Performance Analysis of Packet Scheduling Algorithms

[1] **G. Ramesh Kumar**
[1]**Assistant Professor**
Dept. of Computer Science & Applications
Adhiparasakthi College of Arts & Science
G.B.Nagar, Kalavai – 632 506, Vellore District
Tamil Nadu, INDIA

[2] **P. Srinivasan**
[2]**Research Scholar**
Dept. of Computer Science & Applications
Adhiparasakthi College of Arts & Science
G.B.Nagar, Kalavai – 632 506, Vellore District
Tamil Nadu, INDIA

ABSTRACT

In infrastructure network the connection between the source and the destination is connection oriented. And there is continuous path between the source and the destination. While sending the packets from source to destination loss of packets will occur. This project is going to analysis lower missed deadline ratio and time complexity between the scheduling algorithms like FCFS (First Come First Serve), EDF (Earliest Deadline First), LLF (Least Laxity First), CMA (Chen and Muhlethaler Algorithm), and BPA (Best Effort Packet Scheduling Algorithm).This paper compares Missed deadline ratio is defined as the ratio between the number of packets discarded by total number of packets released from the queue and reduces the packet drop ratio and increases the packet success ratio. These algorithms describe an approach to design an Packet scheduling algorithm for Real-time Switched Ethernet Networks and how the algorithms takes scheduling decision faster.

Background

The Ethernet standard is unsuited for real time application due to the random strategy employed by Ethernet’s Collision Detection (CD) MAC (Media Access Control) protocol. The CD protocol conceptually employs a non-deterministic distributed message scheduling algorithm, where host that attempt to simultaneously transmit messages and thus contend for the network medium, “back-off” for a random period of time, before attempting to transmit again. The back-off strategy is repeated where the randomly determined waiting time exponentially increases with each subsequent collision, until the host is able to transmit. Implementation of these algorithms is usually done by using a hub, where hosts are connected to the different ports of the hub. The hub simply broadcasts each message that it receives from any hosts at any of its port, to every host that is connected to one of its ports. This significantly increases the likelihood of message collision, triggering the CD protocol. Ethernet is attractive for real-time application due to its wide availability, low cost, and high performance such as that offered by the emerging 10 Gigabit Ethernet standard.

1. First Come First Served (FCFS)

FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand. One of the major drawbacks of this scheme is that the average time is often quite long.

The First-Come-First-Served algorithm is rarely used as a master scheme in modern operating systems but it is often embedded within other schemes.

2. Earliest Deadline First (EDF)

Each task in an EDF scheduler is assigned a deadline (e.g. a moment in the future at which the task must be completed). Every time a task is inserted in the system or completed, the scheduler looks for the task which has the closest deadline and selects it for execution. In order to ensure that the scheduler is still able to meet each deadline, evaluate if each new task doesn't overload the system and deny execution if it will do so. Each packet has deadline value, its life time. Sort the packets in increasing order based on the deadline value. Apply the feasibility test for every packet .Feasibility test can be done by checking whether the time of scheduling event(t) plus packet transmission latency(li=bitlength(bi)/Throughput(φ) is lesser than the deadline(di).

```
if ((t+li)<di)
    then feasible packets
else if(((t+li)>di) or (di<0))
    then infeasible packets.
```

Feasible packets are placed in front of the out-queue and infeasible packets are placed end of the queue. Finally deadline miss ratio (or packet drop ratio) and packet guarantee ratio (or success ratio) are calculated. Deadline miss ratio can be defined as the ratio between number of infeasible packets and total number of packets released from the queue. Packet guarantee ratio is the ratio between number of feasible packets and total number of packets released from the queue.

Sample Output:

No of packets available in the outqueue:3

```
deadline for each packets
DL[1]=12
DL[2]=34
DL[3]=25

dl=12 p=l
dl=25 p=3
dl=34 p=2

packet size for each packets
at[1]=120
at[2]=125
at[3]=340

time of scheduling event 25

InFeasible packet: tot=25.000120 dl=12,p=l
InFeasible packet: tot=25.000125 dl=25,p=2
Feasible Packet: tot=25.000340 dl=34,p=3

Completion time for pl=12.000000
Completion time for p3=37.000000
Completion time for p2=71.000000

at[1]=120
at[2]=125
at[3]=340

time of scheduling event  25
InFeasible packet: tot=25.000120 dl=12,p=l
InFeasible packet:tot=25.000125 dl=25,p=2
Feasible Packet:tot=25.000340 dl=34,p=3

Completion time for pl=12.000000
Completion time for p3=37.000000
Completion time for p2=71.000000
Average Completion time=40.000000
waiting time for pl=0
waiting time for p3=12.000000
waiting time for p2=37.000000
average waiting time=16.333334

Packet sending order
Feasible Packets are p2<-null
InFeasible Packets are pl<-p3<-null
Deadline Miss ratio=0.666667
```

3. Least Laxity First (LLF)

Each packet has parameters such as deadline, current time, remaining time and so on. Laxity value of each packet is found using the following formula: Laxity(L) = Deadline - Current Time - Remaining Time. Packets are sorted based on Laxity in increasing order. Always least Laxity packet first enters in to the network segment. Feasibility test can be done by checking whether the time of scheduling event(t) plus packet transmission latency is lesser than the Laxity(lxi), where transmission latency is calculated by the expression li=bitlength(bi)/Throughput(φ)

```
if((t+li)<lxi)
    then feasible packets
else if(((t+li)>lxi) or (lxi<0))
    then infeasible packets.
```

Feasible packets are placed in front of the out-queue and infeasible packets are placed end of the queue. Finally deadline miss ratio (or packet drop ratio) and packet guarantee ratio (or success ratio) are calculated.

Sample Output:

```
No of packets available in the outqueue 3
deadline for each packets
DL[1]=60
DL[2]=45
DL[3]=55

current time for each packets
wc[1]=23
wc[2]=30
wc[3]=20

remaining time for each packets
at[1]=30
at[2]=20
at[3]=30
```

```
Laxity values for each packets
pl->7
p2->-5
p3->5

Laxity values for each packets
pl->7
p2->-5
p3->5

laxity=-5 packet=2
laxity=5 packet=3
laxity=7 packet=l

packet size for each packets
at[l]=45
at[2]=34
at[3]=36

time of scheduling event : 6
Latency [0]=0.000045
Latency [1]=0.000034
Latency [2]=0.000036
Latency [2]=0.000036

InFeasible packets
InFeasible packets
Feasible packets

Completion time for p2=60.000000
Completion time for p3=105.000000
Completion time for pl=160.000000

Average completion time=108.333336ms

waiting time for pl=0
waiting time for packet2=60.000000
waiting time for packet3=105.000000

average waiting time=55.000000ms

Packet sending order
Feasible Packets are pl<-null
InFeasible Packets are p2<-p3<-null
```

4. Chen and Muhlethaler Algorithm(CMA)

Steps:

- Get Precedence matrix dimension
- Assign precedence for all packets
- Based on precedence set boolean value in to the precedence matrix
- if precedence relation (i<j)

  set boolean value as 1
- else if precedence relation (j<i)

  set boolean value as 0
- After construct precedence matrix compute true count value for each row
- Highest true count value send first then marked sent
- Same process repeat until all rows of packet sent

  End process

Chen and Muhlethaler uses the precedence-relation property to define a precedence relation between packets. A packet *i* is said to precede a packet j at a time t, denoted as $i < j$ if $\Delta i,j \geq 0$. .If the precedence relationship between all packets pairs can be determined; the packet that gains the maximum number of precedence is a very good candidate to be scheduled at time t. Thus, Chen and Muhlethaler reason that this maximum-precedence packet can be determined at each scheduling instant and thereby a good scheduling decision can be made.

CMA constructs a precedence matrix at each scheduling instant to store the precedence relations. Each row and column of this matrix represents a packet. Each entry of row the matrix contains a Boolean value that indicates whether there exists a precedence relation between the packets represented on the row and on the corresponding column. Thus, given a precedence matrix I(1, 2…….n) (1, 2……….n) that represents the precedence relationship of n packets at a scheduling instant t, I(i,j) is true, if and only if $i <_t j$; otherwise I(i,j) is false.

Table illustrates this concept. Once precedence matrix is constructed, CMA computes the schedule by examining the rows of the matrix. For each row of the matrix, the algorithm counts the number of true values that are present in the columns of the row.

	1	2	……	i	……	n
1	---	*I*(1,2)	……	*I*(1,i)	……	*I*(1,n)
2	---	---	……	*I*(2,i)	……	*I*(2,n)
……	---	---	……	……	……	……
i	---	---	……		……	*I*(i,n)
……	---	---	……	……	……	……
n	---	---	---	---	---	---

Precedence matrix table

The number of true values in a row *I* indicates the number of packets over which packet i has a precedence relation. Once the "true counts" are determined for rows, CMA selects the packet that has the largest true count, inserts the packet into the schedule, and marks the row of the packet as "examined" in the matrix. The algorithm repeats this process until all packets are inserted into the schedule.

Sample Output:

```
Precedence matrix size 3
Pi<Pj then set Boolean 1
Pj<Pi then set Boolean 0
Packet location
(1,1)   1       (1,2)   1       (1,3)   0
(2,1)   1       (2,2)   1       (2,3)   1
(3,1)   0       (3,2)   1       (3,3)   0

         1      2      3

  1      1      1      0      2

  2      1      1      1      3

  3      0      1      0      1

True count        sorting order Row
3                 2
2                 1
1                 3

Packet Row Sending Order: R2<-R1<-R3<-null
Packet sending Order
P2,1<-  P2,2<-  P2,3<-
P1,1<-  P1,2<-  P1,3<-
P3,1<-  P3,2<-  P3,3<-
```

5. BPA : Best Effort Packet Scheduling Algorithm:

BPA is a packet scheduling algorithm. Thus, packets constitute the "input" to the algorithm. Therefore, if packets have benefit functions, that will enable the algorithm to reason about scheduling packets such that it will maximize the aggregate message-level benefit. This requires us to translate benefit functions of messages into benefit functions for packets. Since the packets of a message do not have any precedence relations, the packets can be transmitted in any order from the source host of the message. Furthermore, the benefit of a message is accrued only when all packets of the message arrive at the destination host and are reassembled. Thus, the packets of a message can simply inherit the benefit function of its parent message.

Thus, BPA reasons that by scheduling packets at outgoing queues of end-hosts and at the switch such that the aggregate packet benefit is maximized, the algorithm can maximize the aggregate message benefit. Thus, in designing BPA, our objective is twofold: (1) compute scheduling decisions faster than CMA's *O*(*n*3) time, and (2) compute scheduling decisions that will yield an aggregate packet benefit that is as close as possible to that of CMA, if not better.

5.1 Sort Packets In Decreasing Order Of Their" Return of Investments"

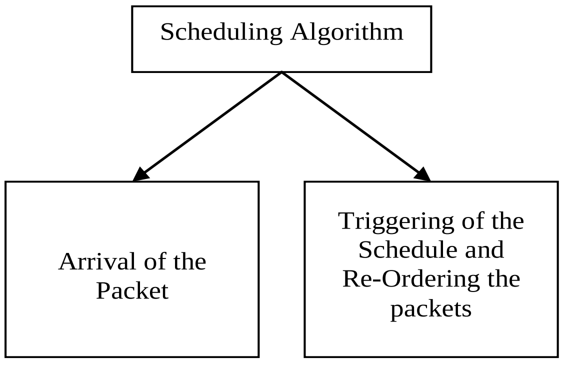
The potential benefit that can be obtained by spending a unit amount of network transmission time for a packet defines a measure of the "return of investment" for the packet. Thus, by ordering packets in the schedule in the decreasing order of their return of investments, we "greedily" collect as much "high return" packets into the schedule as early as possible

Furthermore, since a packet included in the schedule at any instant in time is always the one with the next "highest-return" packet among the set of non-examined packets, we increase our chance of collecting as much "high return" packets into the schedule as early as possible. This will increase the likelihood of maximizing the aggregate packet benefit as packets yield greater benefit if they arrive earlier at their destinations, since all packet benefit functions that we consider are unimodal and non-increasing. The return of investment for a packet can be determined by computing the slope of the packet benefit function. However, computing slopes of arbitrary unimodal benefit functions can be computationally expensive. Thus, we determine the return of investment for a packet as simply the ratio of the maximum possible packet benefit, specified by the packet benefit function, to the packet deadline. This is just a single division, costing O(1) time. We call this ratio, the “pseudo-slope” of a packet. The slope is “pseudo” as it does not represent the correct slope and only gives a crude measure of the slope.

5.2 Move Infeasible Packets toward the End of the Schedule

Infeasible packets are packets that cannot arrive at their destination before their deadlines, no matter what. This is because the transmission time of such packets are longer than the time interval between the scheduling instant the time at which the scheduler is triggered, which is the arrival of a packet into the outgoing packet queue at a host or the switch and the packet deadlines. Packets that are not infeasible are feasible packets. By moving infeasible packets to the end of the schedule, we collect as much feasible packets to the beginning of the schedule as possible. This will increase the likelihood of maximizing the aggregate packet benefit as feasible packets yield greater benefit if they arrive earlier at their destinations since we consider only unimodal benefit functions that are non-increasing. Furthermore, infeasible packets yield zero benefit if they arrive at their destinations after their deadlines. Thus, there is no reason for transmitting them early and jeopardize the potential benefit that can be accrued from feasible packets.

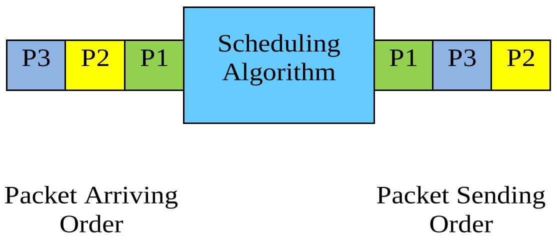
5.3 Module Description



5.3.1 Module 1: Arrival of the Packet

1. Application process send message to lower layer
2. Packets are deposited at host out queue
3. Sense the network medium whether free or not
4. If medium is free then trigger scheduling algorithm else wait until it becomes free.

5.3.2 Module 2: Triggering Of the Schedule and Re-Ordering the Packet



In this module the proposed approach divides available packets in the out queue into two sets namely “feasible packets” and “infeasible packets”. After triggering the scheduling algorithm, in the first iteration packets are sorted into feasible packets. In the second iteration packets are sorted into infeasible packets. “Feasible packets” are placed at the front of the queue and “infeasible packets placed succeeding it. Then the re-ordered packets will reach the destination.

5.4 Pseudo-Code for BPA Algorithm

```
BPA(A, α, t) /* A: set of packets in out queue; α : Number of packets in A; t: time of sched. Event; li:transmission latency */

1. σ=0; /* Intialize packet schedule to empty */
2. For each packets pi belongs to A
   2.1 PseudoSlope(pi) = Bi(0)/Di ;/* Max benefit
       is at time 0; benefit fns are unimodal */
3. Sort packets in A in decreasing order of their
    pseudo-slopes; /* A is now sorted */
4. σ = A;/*packet schedule b is set equal to sorted
    set A*/
5. For k = 1 to α
      5.1 InOrder = TRUE;
      5.2 For i = 1 to α-1
      5.2.1 j=i+1; /* pj is the packet that follows
            pi in schedule σ */
      5.2.2   If (t+li > Di) /* Check for feasibility
              of packet pi */ Move pi to end of
              schedule σ; /* packet pi is not
              feasible */ Continue; /* Skip and
              continue to another iteration */
```

```
        5.2.3  If (t + lj > Dj)   /* Check for feasibility of packet pj */ Move pj to
               end of schedule σ;  /* packet pj is not feasible */ Continue;   /* Skip and
               continue to another iteration */
        5.2.4  Δi,j (t)=[Bi (t+li) + Bj (t+li+lj )] - [Bj (t+lj)+Bi(t+lj+li)] ; /*compute Δ  */
        5.2.5  If (Δ i,j(t) < 0)   /* Out of order, so swap */
                      σ (i) = pj ;
                      σ (j) = pi;
                      t=t+lj ;
                      InOrder = FALSE;
        5.2.6  Else
                      t=t+li;
    5.3 If (InOrder = TRUE)  /* No swaps; so all packets are inorder */
        5.3.1 Break;
    6. σ is the final schedule; return packet σ(1) as the packet selected for transmission;
```

Sample Output:

```
No. of Packets available in Queue: 4

Pname    Benefit          deadline
p1       383.000000       86.000000
p2       777.000000       15.000000
p3       793.000000       35.000000
p4       386.000000       92.000000

Set the Time of Scheduling Event : 25

Pname    Pseudoslope    Benefit      deadline
p2       51.800000      777.00000     5.000000
p3       22.657143      793.00000    35.000000
p1       4.453488       383.00000    86.000000
p4       4.195652       386.00000    92.000000


Enter bit len of p1 : 125
Enter bit len of p2 : 164
Enter bit len of p3 : 178
Enter bit len of p4 : 225

Infeasible packet p2
Feasible packet p3
Feasible packet p1
Feasible packet p4


sending order
p3  22.657143  793.000000        35.000000
pl   4.453488  383.000000        86.000000
p4   4.195652     386.000000     92.000000

Packet sending order
Feasible packets are p3<-pl<-p4<-

InFeasible packets are p2<-


Missed Deadline Ratio=0.250000
Aggregate benefit fn=10.435428
```

6. Comparison between the Scheduling Algorithms.

6.1 FCFS:

In FCFS system whatever packet comes first they are enter first. Major disadvantage of this packet scheduling algorithm Pj packet lesser Deadline than Pi packet. Pi before complete its task, Pj Miss its Deadline

6.2 EDF:

Each task in an EDF scheduler is assigned a deadline (e.g. a moment in the future at which the task must be completed). Every time a task is inserted in the system or completed, the scheduler looks for the task which has the closest deadline and selects it for execution. In order to ensure that the scheduler is still able to meet each deadline, a policer must evaluate if each new task doesn't overload the system and deny execution if it will do so. This EDF system fully based on deadline value

6.3 CMA:

Chen and Muhlethaler uses the precedence-relation property to define a precedence relation between packets. A packet *i* is said to precede a packet j at a time t, denoted as $i< j$ if $\Delta i,j \geq 0$. .If the precedence relationship between all packets pairs can be determined; the packet that gains the maximum number of precedence is a very good candidate to be scheduled at time t. Thus, Chen and Muhlethaler reason that this maximum-precedence packet can be determined at each scheduling instant and thereby a good scheduling decision can be made.

Complexity Analysis of CMA:

Suppose available packet in the out-queue n packets, it would cost the algorithm O(n^2) computations to first construct the precedence matrix. This is followed by O(n^2) number of examinations to determine the packet with the largest true count. The O(n^2) examinations has to be repeated for each of the n packets. Thus, the complexity of CMA is clearly O(n^3).

6.4 Complexity Analysis of BPA

The computational complexity of BPA depends upon the complexity of Step (5). The complexity of all other steps is dominated by this step. The complexity of Step (5) is dominated by that of Step (5.2); all other sub-steps of Step (5) take O (1) time. Step (5.2) can iterate a maximum of 'n' times, and, therefore, costs O(n). Step (5) can iterate a maximum of n times, and thus costs O(n^2).Given m application messages, where each message can be packetized into at most n packets, α = O(mn). Thus, the complexity of BPA is O (m^2n^2).To compare the complexity of BPA with that of CMA, we need to uniformly express the problem size. Thus, given n application packets, BPA has a complexity O (n^2). Given n non-preemptable tasks, which form the input to CMA, CMA has a complexity of O (n^3) . Thus, BPA is an order of magnitude faster than CMA.

7. Conclusion

Thus the 'Best Effort Packet Scheduling Algorithm' performs both the arrival of the packet and Triggering of the schedule and Re-ordering the packet. The algorithm seeks to maximize aggregate packet-level benefit. Worst case Computational complexity of the proposed system is O(n). BPA is suitable for increasing message aggregate benefit function, reducing worst case computational complexity and missed deadline ratio. The computational complexity of the proposed system is O(n^2).

The BPA achieves lesser Deadline miss ratio.. It achieves higher packet guarantee ratio and also it maximizes message level benefit. Thus, the contribution of the project is : (1) The Best effort packet scheduling algorithm that seeks to maximize aggregate message benefit in real-time distributed systems. (2) Reduce the deadline miss-ratio and increase packet guarantee ratio. (3) Taking scheduling decision faster.

8. References:

Journal References:

1. K. Chen and P. Muhlethaler. A Scheduling algorithm for tasks described by time value n 2001.
2. D.L Mills. Improved algorithm for synchronizing computer network clock. IEEE/ACM Transaction on Networking, June 1995.
3. J. W. S. Liu. Real-Time Systems. Prentice Hall, New Jersey, 2000.
4. S. Varadarajan and T. Chiueh. Ethereal: A hosttransparent real-time fast ethernet switch. In Proceedingsof The International Conference on Network Protocols, October 1998.
5. Peter M. Athanas & Scott F. Midkiff "Soft Real-Time Switched Ethernet: Best-Effort Packet Scheduling Algorithm, Implementation, and Feasibility Analysis" PhD thesis, Virginia Polytechnic Institute and State University, September 24, 2002.
6. D. W. Pritty, J. R. Malone, D. N. Smeed, S. K. Banerjee, and N. L. Lawrie, "A real-time upgrade for ethernet based factory networking," in Proceedings of 21st IEEE/IECON International Conference on Industrial Electronics, Control, and Instrumentation, 1995, pp. 1631 –1637.
7. S. K. Kweon and K. G. Shin, "Achieving real-time communication over Ethernet with adaptive traffic smoothing," in Proceedings of IEEE Real-Time Technology and Applications Symposium, 2000, pp. 90 –100.
8. W. Zhao and K. Ramamritham, "A virtual time csma/cd protocol for hard real-time communication," in Proceedings of IEEE Real-Time Systems Symposium, December 1986, pp. 120–127.
9. C. Baek-Young, S. Sejun, N. Birch, and J. Huang, "Probabilistic approach to switched ethernet for real-time control applications," in Proceedings of 7th IEEE International Conference on Real-Time Computing Systems and Applications, 2000, pp. 384 –388.
10. G. C. Buttazzo, "Chapter 5: Fixed priority servers," in Hard Real-Time Computing Systems:Predictable Scheduling Algorithms and Applications. Kluwer Academic Publishers,1997.