4
Adapting Jumbo Frames in Data center

4.1 INTRODUCTION

Nowadays data centers are attracting attention of researchers in a big way. The performance of the data centers holds the key in the success of cloud computing. As dimensions of cloud computing are enhanced by offering multiple services, the responsibility of service provider escalates many fold. Even hosting a single service per data center requires promising conduct round the clock. Incompetence in fulfilling such a vow may lead to disappointment in customers. Customers are very sensitive to delay and long delay may result in losing of customers. Therefore delay in data centers adversely affects on business revenue. This chapter attempts to focus on how to knock down this delay in data centers with the help of oversubscribed topology. The proposed idea in this chapter, Jumbo Frame Enabled Packet Merge (JFEPM), allows in-network data aggregation and with the help of jumbo frames it utilizes topology architecture of the data center. This idea is based on jumbo size frames supported by Gigabit and Fast Ethernet switches. This scheme requires modification to the service policy of queue on the edge switch. This chapter uses oversubscribed data center topology architecture and transmits large sized frames in higher layers of data center network.

4.2 MOTIVATION

The designers of data center have proposed the idea of oversubscription in which racks must be connected with higher capacity links. Oversubscription not only reduces the cost of the infrastructure but also provides a better resource utilization. Generally ToR switches in data center are connected to aggregation switches with 10 Gbps link. These high speed links require less time to transmit packets. As discussed earlier in Chapter 1, the data center is owned and operated by a single organization, and environment in the data center is very much under control compared to the Internet. In the former, dealing with a large size packet is not an issue.

4.2.1 Data center Topology

A data center topology is a tree like structure as shown in Figure 4.1. Each server is connected to ToR switch with 1 Gbps link. In this structure, core switches are the root of the tree and the end hosts or servers are leaf nodes of the tree. As discussed earlier, a rack contains multiple servers. Each server is connected to a ToR switch with 1 Gbps link. Servers within the rack can communicate with each other at 1 Gbps. To make a location independent of server assignment for a service, it is necessary that a server should be able to communicate at its full speed to any other server in the data center. In other words, if bandwidth between two servers is not dependent on where they are located then the servers for a given service can be distributed anywhere in the data center[Greenberg et al., 2008]. To make location independent server assignment possible, it is necessary that racks must be connected with higher capacity links. In every generation topology, switch-to-switch links are faster then server-to-switch links. On the other hand, high capacity switch ports have their own limitation, and they are expensive. Sometimes higher bandwidth links are underutilized also. Therefore, it is not reasonable to allow all switches with high capacity link at all ports. The idea of oversubscription helps the designers of data center to reduce the cost of infrastructure and improve the utilization of resources.

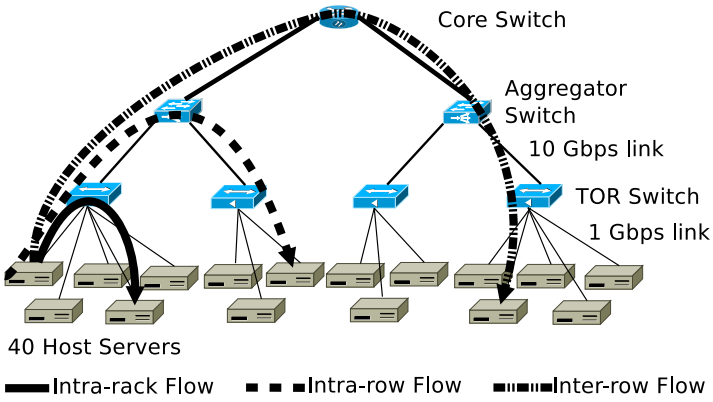

Figure 4.1. : Data center topology with oversubscription

4.2.2 Jumbo frames in Ethernet

Traditional Ethernet supports MTU of 1500 bytes but now with the jumbo frame it is possible to send a frame of size 9000 bytes. Gigabit Ethernet and Fast Ethernet switches support jumbo frames. Jumbo frames are not part of the official IEEE 802.3 Ethernet standard. Network interface card providers like Intel, Broadcom, Netxen, Neterion will support jumbo frames. Most modern switches are capable of jumbo frames, and much of the networking hardware within the data centers are capable of jumbo frames [Hogg, 2013]. Jumbo frames lower network overheads because fewer packets and, therefore, fewer headers, are required to move data from the source to the destination [Murray et al.] [Divakaran et al., 2009]. Jumbo frames also increase the efficiency of a network because jumbo frames spend more time in the transmission of payload. Besides these advantages of the jumbo frame, adoption in the Internet is questionable as the designers were worried about the effectiveness of error detection technique (CRC 32) with 9000 bytes frame. Additionally in the Internet packet fragmentation is required because of different size of MTU of different networks. Topology in data center is structured and all operations are controlled by a single administrator. Hence, allowing large size packets in data center is feasible.

4.2.3 Latency Vs Packet Size

Several ways of reducing latency by designing of network protocols were discussed in Chapter 2. Average Flow Completion Time (AFCT) is a measure of network latency [Hong et al., 2012][Munir et al., 2013][Xu and Li, 2013]. The effect of packet size on average flow completion time has not been analyzed yet. To determine the effect of packet size on average flow completion time we simulate the topology shown in Figure 4.1 with 40 servers in each rack in NS-2. In this simulation, different TCP versions were used as transport layer protocol. With a single FTP flow per ToR, mice flows are simulated with an average size of 100 KB. The flows arrival time follows Poisson process. The size of the flow is chosen from a uniform distribution within the interval of [2KB,198KB][Munir et al., 2014]. In simulation studies, after a certain load, as packet size is increased, both average flow completion time (AFCT) and number of packets dropped decreased. Figure 4.2 and Figure 4.3 show the average flow completion time and the number of packets dropped with increasing packet sizes. These results motivate us to take an initiative for capitalizing the effect of packet size in order to reducing latency.

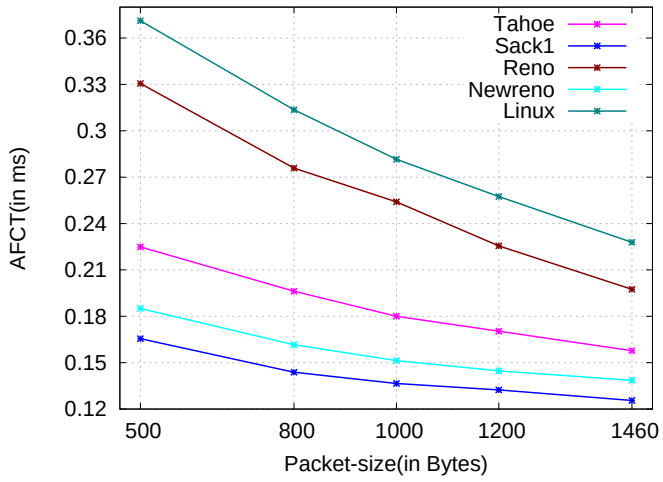

Figure 4.2. : Average flow completion time Vs Packet size

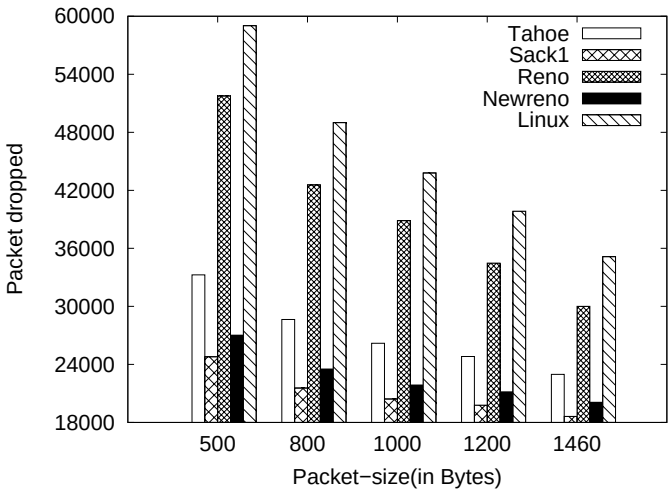

Figure 4.3. : Number of packets dropped Vs Packet size

4.3 JFEPM: JUMBO FRAME ENABLED PACKET MERGE

As discussed in the previous section, end server can communicate at 1 Gbps rate. However ToR switches can deliver frames at 10 Gbps, these links are also jumbo frame enabled. For a frame to go through a link of 1 Gbps, it requires more time compared to a 10 Gbps link. A ToR switch, for the same size of the frame, can transmit comparatively faster to end server. The use of jumbo frames enables a high capacity switch to match transmission time of frame with end server. This feature of jumbo frame empowers a switch to investing more time in the transmission of data. This approach is also helpful in minimizing frequent switching activities [Salyers et al., 2007]. The basic idea of JFEPM is to reduce habitual switching through the jumbo frame at the aggregation and core switches. Aggregation and core switches in the topology architecture of data centers use higher capacity links. JFEPM utilizes this data center topology architecture to reduce the latency. JFEPM may increase queuing time for some packets due to transmission of the jumbo frames,

regardless of guarantee. As guarantee in a data center is at the flow-level not at the packet-level, this scheme will not hamper such a guarantee.

JFEPM assumes that all applications are transferring data only through fixed sized packets. As discussed in Section 4.2.3 flow completion time and number of packets dropped, which are shown in Figure 4.2 and Figure 4.3 respectively, are being reduced with an increase in packet size. Allowing end servers to use bigger packet at 1 Gbps might seize a link for a longer time, as well as increase the waiting time for other applications. JFEPM provides a way to increase the packet size for a path segment between intermediate devices. These intermediate devices are connected with high capacity links in the data center. This scheme is not an end-to-end scheme like in [Prakash et al., 2013], and no change is required at the end servers. In this approach of JFEPM, only ToR switch is responsible for combining and separating the packets. When a ToR switch gets a packet from a sender and if enough space is available, the packet is simply inserted at the rear of the queue. Service policy is changed for removing a packet from the queue; the packet from the front of the queue was served first. While searching the next slot for forwarding a packet, it also searches for other packets of the same flow in the queue as shown in Figure 4.4. Data center uses small size buffers due to small RTT. These small sized buffers help in reducing latency[Alizadeh et al., 2010]. Hence, the cost of the packet searching is less O(buffer size) because the time complexity of the search is linear in nature. During the search if identical packets of the same flow are found, then the switch removes those packets from the queue and merge them after extracting headers of each as shown in Algorithm 1. The maximum allowable size of the packet in this scheme is 8960 bytes. Once either packet size reached a threshold or traversal of the queue is completed, the merged packet will be transmitted with a single header. This new header contains information about total number of merged packets as well as some transport layer information like sequence number and a time stamp. This newly constructed jumbo frame is forwarded to an aggregation switch.

Algorithm 1 Deque method

```
Remove a packet from front
Flow ID:= Packet.Flow ID
Size:= Packet.Size
if Within rack flow then
  return Packet
end if
for Each packet in queue do
  Temp:= Packet
  if Flow ID==Temp.Flow ID && Size ≤ 8960 then
    Remove temp from queue
    Remove header from temp
    Merge temp with packet
    Number of packet merge += 1
    Size += Temp.Size
    Drop temp
  end if
end for
return Packet
```

At the receiver side, ToR switch receives a jumbo frame from an aggregation switch. The recognition of a jumbo frame is based on the field name, as the number of packets merged, in the header. This field is set to 0 for normal packets and more than 1 for jumbo frames. Switches treat a normal packet and jumbo frame differently; the procedure to handle jumbo frame is described in Algorithm 2. Whenever free buffer space at the switch is not adequate to accommodate the packet, it simply drops the packet. On the other hand, the giant packet is split into original and equal size

packets. Before inserting packets in the queue, transport layer information of the packets is also put in appropriate header fields. Once packets are reconstructed in their original form, all packets are inserted at the rear of a queue and the jumbo frame is dropped. The merging and splitting process of packets is invisible to the source and destination nodes.

Algorithm 2 Enque method

```
if Packet size ≤Available buffer space then
  Find number of packets merged
  if Number of packets merged ≥ 1 then
    Remove header of jumbo packet
    Split original PDU into equal size PDUs
    for Each small PDU do
      Attach header with PDU
      Set number of packets merged field to 0
      Insert at the rear end of the queue
    end for
    Drop original jumbo packet
  else
    Insert into the queue
  end if
else
  Drop packet
end if
```

For simplicity, an example shown in Figure 4.4 is discussed here. The packets of different flows (1, 2, and 3) occupy the queue at the ToR switch on the sender side. Packet of flow 1 is pointed by the front pointer and packet of flow 2 is at the rear of the queue. The switch removes the packet from the front (in this case packet of flow 1) for forwarding to next slot and also searches for packets of the same flow. Other packets of the same flow (flow 1) will be found at position 3, 4, and 6 of the queue. JFEPM removes all packets of flow 1 from the queue and merges them after extracting the headers. This merged packet is sent to aggregation switch after affixing new header. This new header also carries information about number of packets merged (4 in this situation). On the ToR switch at the receiver side, when this jumbo frame is reached, the switch seeks for the number of packets merged information. Upon getting this information, the jumbo PDU is divided into original size PDUs of equal size. In the figure PDU is divided into 4 original size packets. Original headers are included in these all PDUs, and number of packets merged field is set to zero. All these packets are inserted at the rear of the queue.

Aforementioned scheme of the jumbo frame is not applicable for intra-rack communication. Packets of such flows only need to traverse ToR switch through 1 Gbps link; ergo, the only beneficiary of this scheme is the flows going outside of racks. Jumbo frame based approach can yield profit only between ToR's due to high rate links. This scheme is helpful in reducing processing time at aggregation and core switches.

4.4 ANALYTICAL CHARACTERIZATION

This section illustrates theoretical characterization of the total processing time of the packets sent by a host to all the hosts in the same rack (intra-rack), same row (intra-row) and other rows (inter-row). Keeping simplicity in mind we analyze JFEPM for the topology shown in Figure 4.1 with two aggregation switches and two ToR switches per aggregation switch. This reasoning can be extended for any larger topology also. Symbols used for this purpose are listed in Table 4.1. This

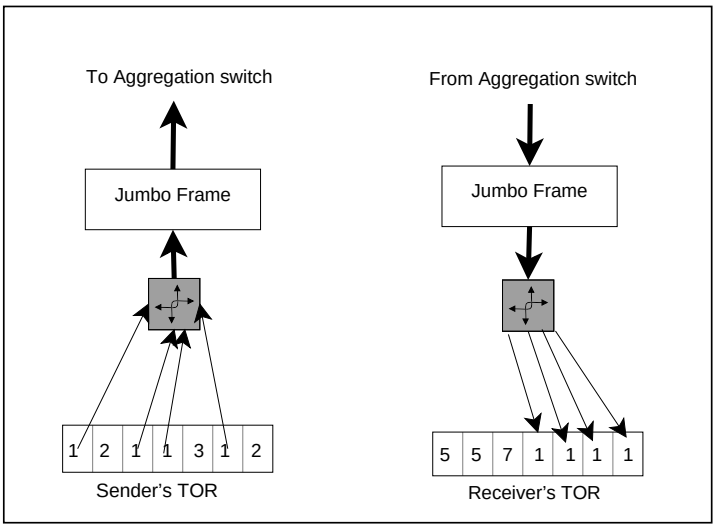

Figure 4.4. : Packet merge and split at ToR switch

analysis considers a situation where a node in the data center wants to send N packets to all other hosts.

Symbol	Description
n	Number of hosts per ToR
N	Number of packets of each flow
M_f	Merge factor (Max number of packets merge)
B_f	Buffer size (in bytes) of a switch
T_f	Forwarding time of a packet at a switch
TP_o	Total processing time with original size packets
TP_j	Total processing time with jumbo size packets
α	Time required to find a packet belongs to a particular flow
S_p	Size of original packet (in bytes)
S_j	Size of Jumbo frame (in bytes)
$E[.]$	Expectation function

Table 4.1. : Symbols used for analysis

Here the term merge factor (M_f) defines the maximum number of original packets that can be carried by a jumbo frame, which is the ratio of size of the jumbo frame and the original packet.

$$M_f = \left\lfloor \frac{\text{Size of Jumbo frame } (S_j)}{\text{Size of original packet } (S_p)} \right\rfloor$$

The term $M_f > 1$ because $S_j > S_p$ (size of jumbo frame is larger than size of original packets) and is always an integer. The expectation of the variable M_f is defined as the average number of packets carried by a single jumbo frame.

$$E[M_f] = \frac{1}{M_f} \sum_{i=1}^{M_f} i$$

4.4.1 Processing time without Jumbo frame (Original size)

The processing time required for intra-rack flows is

$$(n-1)NT_f$$

The processing time required for intra-row flows is

$$3nNT_f$$

Similarly the processing time for inter-row flows is

$$10nNT_f$$

The total processing time required for sending N packets to all other hosts in the network by a single host can be obtained as

$$TP_o = (14n-1)NT_f$$

4.4.2 Processing time with Jumbo frame (JFEPM)

The processing time required for intra-rack flows is

$$(n-1)NT_f$$

The processing time required for intra-row flows is

$$\frac{nN}{E[M_f]}\left\{\alpha\left(\frac{B_f}{S_p}\right)+2T_f\right\}+nNT_f$$

Similarly the processing time for inter-row flows is

$$\frac{2nN}{E[M_f]}\left\{\alpha\left(\frac{B_f}{S_p}\right)+4T_f\right\}+2nNT_f$$

The total processing time required for sending N packets to all other hosts in the network by a single host with JFEPM is

$$TP_j = NT_f\left[(4n-1)+\frac{3nk}{E[M_f]}+\frac{10n}{E[M_f]}\right]$$

where

$$k = \alpha\left(\frac{B_f}{S_p}\right)\frac{1}{T_f}$$

Here α is a constant because it requires comparison of few fields of packet header and T_f is O(n) because of forwarding table will have maximum of n entries. Hence $\frac{\alpha}{T_f} \cong \frac{1}{n}$, and if the ratio of size of buffer and size of original packet i.e. $\frac{B_f}{S_p} = n$ then $k \cong 1$.

Dividing TP_o by TP_j we get

$$\frac{TP_o}{TP_j} = \frac{14n-1}{4n + \frac{10n}{E[M_f]} + \frac{3nk}{E[M_f]} - 1}$$

So for large n, rewrite the equation as

$$\frac{TP_o}{TP_j} = \frac{14}{4 + \frac{10}{E[M_f]} + \frac{3k}{E[M_f]}}$$

Here $E[M_f] > 1$ which is the average number of packets combined in a single jumbo frame. As the ratio of the sizes of jumbo frame and original packet increases, this expectation also increases. When buffer size $\frac{B_f}{S_p}$ is equal to the number of hosts per ToR then $k \cong 1$. In this situation $\frac{TP_o}{TP_j} > 1$, which is the ratio of the total processing time without jumbo frame and with jumbo frame. Hence, one can argue that the total processing time in case of jumbo frame is less as compared to the original case; if the number of hosts per ToR is large and buffer size at the switches is not larger than the number of hosts per ToR.

This buffer size restriction limits the cost of searching in this technique. However, it does not mean that smaller buffer size gives more advantage. As the buffer size is reduced, jumbo frame will merge fewer packets which also limits its performance. Hence, a buffer size which is equal to the number of hosts per rack is an appropriate choice for this scheme.

4.5 ARGUMENT ON THE APPROACH OF PACKET MERGE

Store and forward technique with small size packets provide a way to increase parallelism. Due to small size packets, any intermediate switching element can collect and pass on packets simultaneously. At first look, it seems like packet merging scheme is unfriendly to parallelism because merged packets seize the link for more time, and earnings made by parallelism are diminished. With JFEPM, the benefits of parallelism obtained are maintained intact by the virtue of the topological structure in the data center. High capacity links from one ToR switch to another are supportive to sustain the gain of the parallel pipeline. The transmission time of the merged packet at ToR switch is never more than the transmission time of the original packet at end server. In this approach, only merged packet go through high capacity links. Merged packets are divided into original size packets at the receiver side ToR switch so that any single packet can not seize the link for a longer time. As discussed earlier in Section 4.2.1 links between the end servers and the ToR switch are of 1 Gbps. Hence, the final remark about JFEPM is, proposed scheme is not a foe of parallelism with oversubscribed topology structure.

Service policy used in this scheme for merging packets requires different structures. Removal of the packets in the middle of a queue is not possible with First In First Out (FIFO) discipline. For merging packets, this approach requires removal of packets from anywhere in the queue. This unsystematic removal of a packet from queue requires additional overhead to maintain consistency.

Packets after merging take a longer time to transmit than usual owing to the bigger size. Due to this longer transmission time, a few of the packets may get delayed to get service, which increases queuing time of these packets. On the other hand, this approach serves some of the packets earlier than expected time. This packet merging scheme is responsible for disturbing the order of packets being served as well as delayed or advanced reception of packets. However, this uncertainty of one way delay at the packet-level increases jitter at the receiver; the flow completion time is still not influenced by such fluctuations. Flow completion time is defined as the time duration from the beginning to the end of flow. Fortunately service level agreements in a data center require guarantee at flow-level not at the packet-level.

4.6 EVALUATION

JFEPM is evaluated using NS-2 simulator. A 3-tier topology with ToR switches, aggregation switches and single core switch is simulated as shown in Figure 4.1. Such a topology is commonly used for data center simulation purpose [Munir et al., 2014][Vamanan et al., 2012][Wilson et al., 2011]. The total number of hosts in this topology is 160, which are interconnected by 4 ToR switches that are connected through 2 aggregation switches. Single core switch facilitates connections between aggregation switches. Each link between the end host and ToR has 1 Gbps, and all other links are 10 Gbps. Hence, the topology is oversubscribed by 4:1 for up link from ToR switch. In the simulation, the minimum round trip propagation delay (without queuing) for flows between end hosts going through core switch is $300\mu s$.

4.6.1 Simulation Setup

In simulation, flow arrivals follow Poisson process. The size of the flow is chosen from a uniform distribution within the interval of [2KB,198KB] [Munir et al., 2014]. With the help of load, the arrival rate of flows is changed. Such a traffic pattern is already used in previous study [Hong et al., 2012][Munir et al., 2014][Wilson et al., 2011]. These flows are the replication of mice flows in data center. For simulating elephant flows long-lived FTP flows are generated. In this simulation drop-tail queue with a maximum size of 40 packets (number of hosts per ToR) is used. The maximum size of jumbo frame allowed here is 9000 bytes. For comparison purpose different versions of TCP (Tahoe, Reno, Newreno, Sack) were used. JFEPM is compared with the different version of TCP besides different packet sizes (800, 1000, 1200, 1460 bytes) at different loads (0.1 to 0.95). The load was divided into three categories moderate (0.1 - 0.4), average (0.5 - 0.7) and heavy (0.8 - 0.95). The Average Flow Completion Time (AFCT) and the number of packets dropped are the performance metrics to compare JFEPM with the original version of TCP.

4.6.2 Simulation Results

In the graphs, one can note that JFEPM is more beneficial when packet sizes are small; because in this case, a jumbo frame can merge more small packets. Here Max difference represents the maximum FCT difference between JFEPM and original version of TCP in the microseconds. The load at which the maximum difference is achieved is also shown after '/' sign. For example, when packet size is 800 bytes JFEPM can reduce maximum 33 μs at load 0.6 in Tahoe (Table 4.2). Avg gain represents an average reduction of FCT, irrespective of load. In this section for better explanation of results, the term percentage change is defined as

$$\frac{\text{original FCT} - \text{jfepm FCT}}{\text{original FCT}} \times 100$$

which represents comparative change. Max percentage at load is the maximum gain in percentage change and Avg percentage is average percentage change.

TCP-Tahoe

Tahoe is the simplest version of TCP. The proposed scheme, JFEPM, performs better with TCP-Tahoe compared to original one. These graphs are 3-d graphs that represent the load on one axis and packet size in bytes on the other axis. For clarity, tables of profit gained by JFEPM are given.

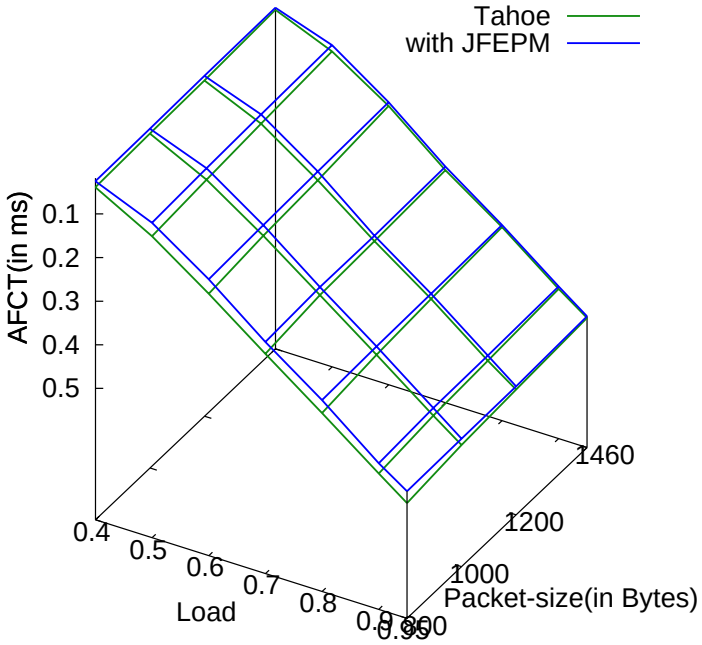

Figure 4.5. : AFCT of Tahoe

Size	Max difference (in μs)/load	Avg gain (in μs)	Max% /load	Avg gain %	Reduction factor
800	33/0.6	19.58	42.17/0.1	21.33	0.90
1000	26.5/0.5	13.81	31.90/0.4	14.38	0.93
1200	21.2/0.5	9.98	33.39/0.4	11.76	0.94
1460	14.4/0.5	5.89	28.16/0.3	13.11	0.96

Table 4.2. : Difference of AFCT between JFEPM and Tahoe

The performance of JFEPM with Tahoe for AFCT is shown in Figure 4.5 and Table 4.2. As shown in Table 4.2, Avg gain for AFCT of JFEPM decreases with the increase of packet size. The maximum FCT difference is achieved on average load while the maximum percentage is achieved on moderate load. When the packet size is 800 bytes JFEPM is able to reduce flow latency by a factor of 0.90. Similarly for packet size of 1000 bytes the latency is reduced by a factor of 0.93; for 1200 bytes packets this factor is 0.94; for 1460 bytes packets this factor is 0.96 for FCT.

Size	Max difference /load	Avg difference (packets)	Max difference %/load	Avg difference %	Reduction factor
800	5477/0.95	3015.4	60.66/0.1	24.82	0.92
1000	3266/0.6	1963.8	32.74/0.4	15.72	0.94
1200	2509/0.6	1150.8	34.36/0.4	12.09	0.96
1460	1732/0.5	375.3	35.71/0.1	14.60	0.98

Table 4.3. : Difference of dropped packets between JFEPM and Tahoe

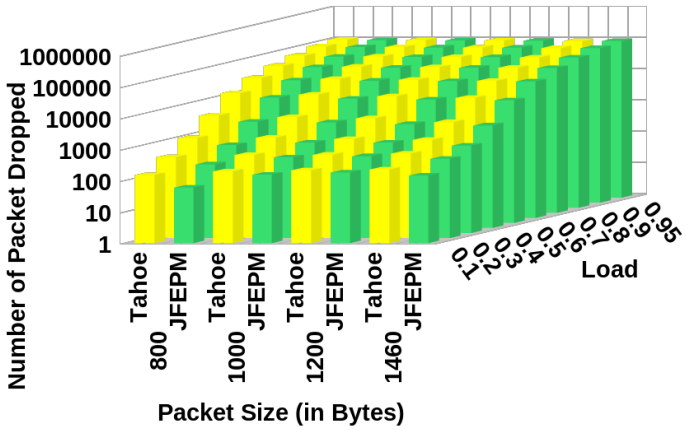

Figure 4.6. : Number of packets dropped of Tahoe

The performance of JFEPM with Tahoe for the number of packets dropped is shown in Figure 4.6 and Table 4.3. As shown in Table 4.3, Avg difference for the number of packet dropped between JFEPM and original Tahoe decreases with the increase of packet size. The maximum percentage is achieved on a moderate load. When the packet size is 800 bytes JFEPM is able to reduce the number of dropped packets by a factor of 0.92. Similarly for packet size 1000 bytes same factor is 0.94; and for 1200 bytes packets this factor is 0.96; for 1460 bytes packets this factor is 0.98 for dropped packets.

TCP-Reno

TCP-Reno can detect the packet loss before a timeout with three duplicate acknowledgments. Reno after detecting loss sets the congestion window to half of the current congestion window.

The performance of JFEPM with Reno for AFCT is shown in Figure 4.7 and Table 4.4. As shown in Table 4.4, Avg gain for AFCT of JFEPM is reduced with the increase of packet size. The maximum FCT difference is achieved on average load while maximum percentage reduction of FCT is achieved on moderate load. When the packet size is 800 bytes JFEPM is able to reduce flow latency by a factor of 0.82. Similarly for packet size of 1000 bytes this factor is 0.86; for 1200 bytes packets same factor is 0.89; for 1460 bytes packets this factor is 0.91 for FCT.

The performance of JFEPM with Reno for number of packet dropped is shown in Figure 4.8 and Table 4.5. As shown in Table 4.5, Avg difference for number of packet dropped between JFEPM and original Reno is reduced with the increase of packet size. When the packet size is 800

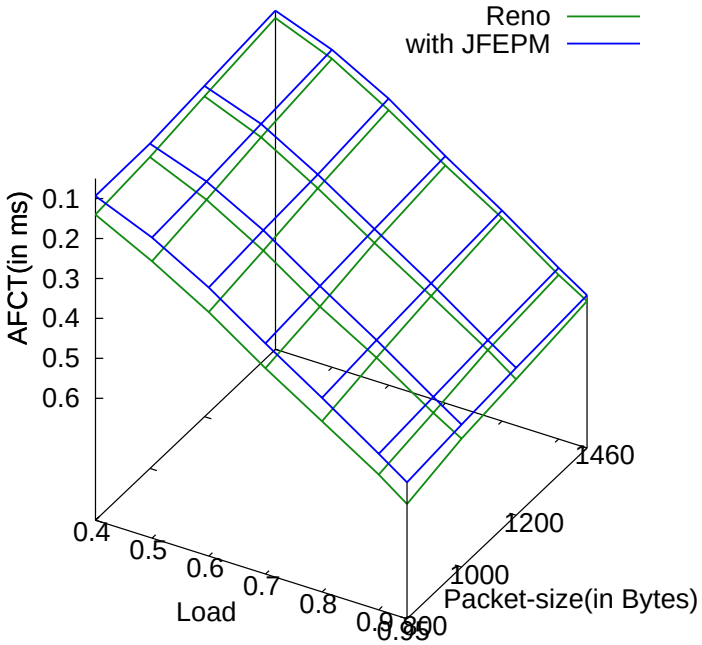

Figure 4.7. : AFCT of Reno

Size	Max difference (in μs)/load	Avg gain (in μs)	Max% /load	Avg gain %	Reduction factor
800	61.94/0.7	49.91	54.74/0.1	26.82	0.82
1000	51.62/0.7	38.13	39.43/0.1	21.22	0.86
1200	34.67/0.7	27.40	36.76/0.1	19.0	0.89
1460	28.87/0.6	18.52	45.49/0.1	18.84	0.91

Table 4.4. : Difference of AFCT between JFEPM and Reno

bytes JFEPM is able to reduce the number of dropped packets by a factor of 0.84. Similarly for packet size 1000 bytes same factor is 0.88; for 1200 bytes packets this factor is 0.90; for 1460 bytes packets this factor is 0.93 for dropped packets.

TCP-Newreno

TCP-Newreno is able to detect multiple packet losses and improves re-transmission during the fast recovery of Reno. Newreno does not exit fast recovery until all the data that was transmitted before congestion episode is acknowledged, so reduction of the congestion window is only once even there are multiple packets dropped in Newreno.

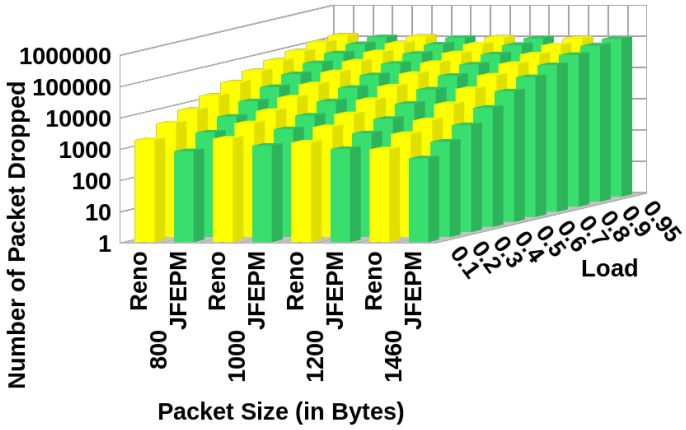

Figure 4.8. : Number of packets dropped in Reno

Size	Max difference /load	Avg difference (packets)	Max difference %/load	Avg difference %	Reduction factor
800	13778/0.95	7790.3	56.57/0.1	27.37	0.84
1000	9263/0.8	5730	40.65/0.1	21.31	0.88
1200	6800/0.95	4077.5	38.02/0.1	19.28	0.90
1460	4142/0.6	2521.2	48.52/0.1	19.40	0.93

Table 4.5. : Difference of dropped packets between JFEPM and Reno

Size	Max difference (in μs)/load	Avg gain (in μs)	Max% /load	Avg gain %	Reduction factor
800	20.27/0.5	10.67	38.39/0.3	19.19	0.94
1000	18.17/0.5	7.14	33.34/0.4	13.60	0.96
1200	14.58/0.5	6.82	32.61/0.4	11.88	0.96
1460	9.72/0.9	4.51	27.89/0.2	12.16	0.97

Table 4.6. : Difference of AFCT between JFEPM and Newreno

The performance of JFEPM with Newreno for AFCT is shown in Figure 4.9 and Table 4.6. As shown in Table 4.6, Avg gain for AFCT of JFEPM decreases with the increase of packet size. As shown in Table 4.6, The maximum percentage is achieved on moderate load. When the packet size is 800 bytes JFEPM is able to reduce flow latency by a factor of 0.94. Similarly for a packet size 1000 bytes this factor is 0.96; for 1200 bytes packets same factor is 0.96; for 1460 bytes packets this factor is 0.97 for FCT.

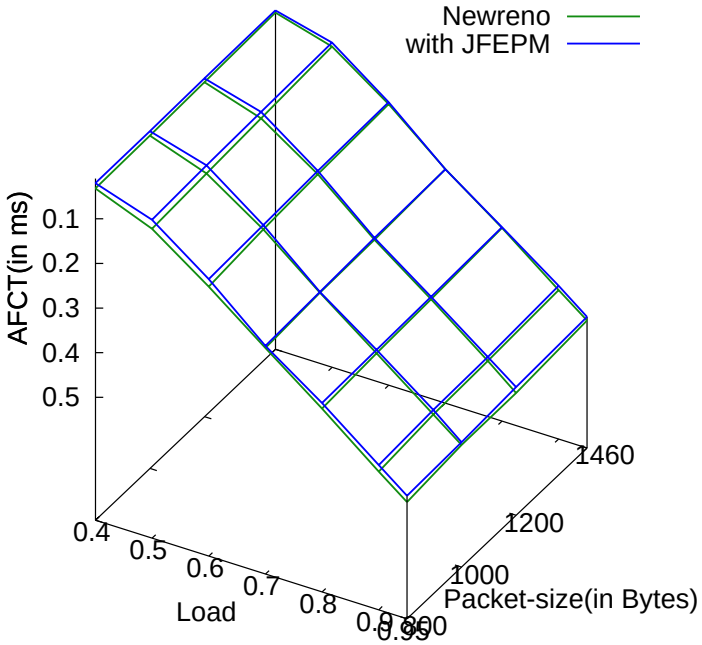

Figure 4.9. : AFCT of Newreno

Size	Max difference /load	Avg difference (packets)	Max difference %/load	Avg difference %	Reduction factor
800	4196/0.95	1969.4	53.90/0.1	24.20	0.94
1000	2504/0.9	1308.9	35.83/0.4	15.80	0.96
1200	2311/0.95	1041.4	35.59/0.4	13.28	0.96
1460	1123/0.5	478.8	35.68/0.2	14.29	0.98

Table 4.7. : Difference of packets dropped between JFEPM and Newreno

The performance of JFEPM with Newreno for the number of packet dropped is shown in Figure 4.10 and Table 4.7. As shown in Table 4.7, Avg difference for the number of packet dropped between JFEPM and original Newreno is reduced with the increase of packet size. When the packet size is 800 bytes JFEPM is able to reduce the number of dropped packets by a factor of 0.94. Similarly for packet of size 1000 bytes this factor is 0.96; for 1200 bytes packets same factor is 0.96; for 1460 bytes packets this factor is 0.98 for dropped packets.

TCP-Sack

Sack uses selective acknowledgment for minimizing re-transmission. Due to selective acknowledgment, the sender comes to know about lost packets and then transmit only the lost packets.

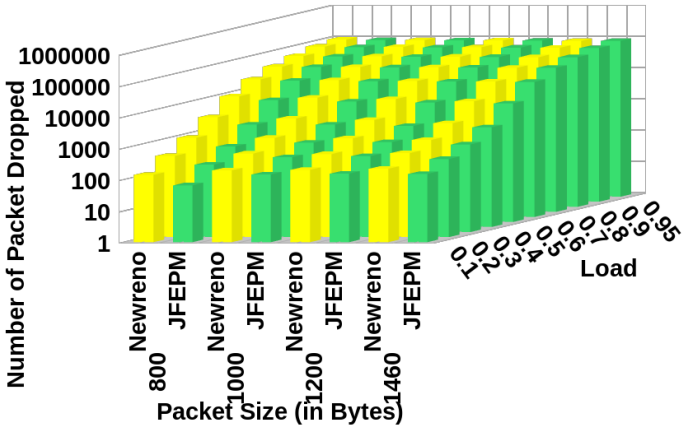

Figure 4.10. : Number of dropped packets in Newreno

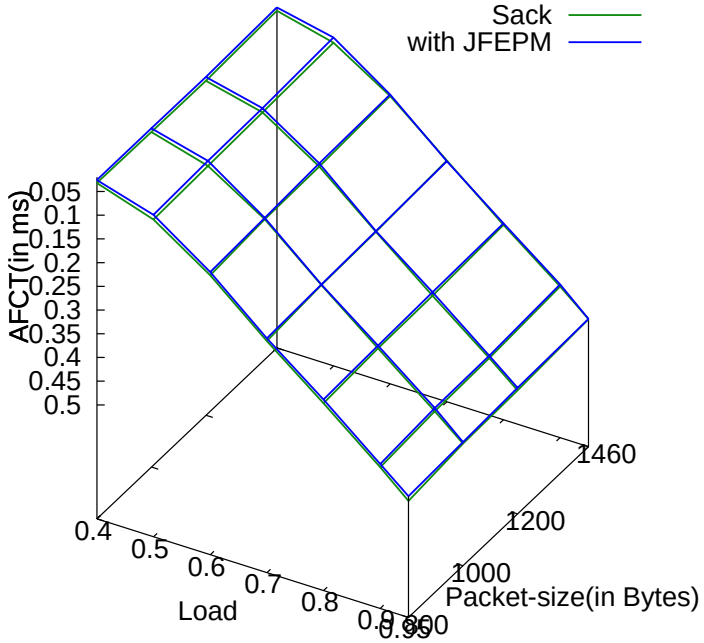

Figure 4.11. : AFCT of Sack

The performance of JFEPM with Sack for AFCT is shown in Figure 4.11 and Table 4.8. As shown in Table 4.8, the maximum percentage is achieved on moderate load. When the packet size is 800 bytes JFEPM is able to reduce the flow latency by a factor of 0.96. Similarly for packet size 1000 bytes the latency is reduced by factor of 0.96; for 1200 bytes packets this factor is 0.96;

Size	Max difference (in μs)/load	Avg gain (in μs)	Max% /load	Avg gain %	Reduction factor
800	9.83/0.95	6.77	33.76/0.2	13.97	0.96
1000	10.57/0.5	5.85	37.86/0.1	14.36	0.96
1200	8.77/0.5	6.61	30.31/0.2	13.40	0.96
1460	11.02/0.5	5.23	44.03/0.1	16.33	0.96

Table 4.8. : Difference of AFCT between JFEPM and Sack

for 1460 bytes packets this factor is 0.96 for FCT.

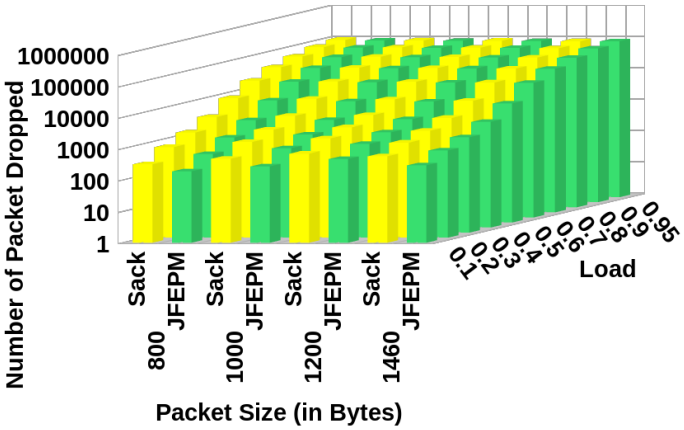

Figure 4.12. : Number of dropped packets in Sack

Size	Max difference /load	Avg difference (packets)	Max difference %/load	Avg difference %	Reduction factor
800	5264/0.95	2013.3	40.88/0.1	17.81	0.93
1000	3130/0.9	1465.6	43.18/0.1	17.15	0.95
1200	2646/0.95	1335.1	33.33/0.2	15.46	0.95
1460	1407/0.5	806.3	48.94/0.1	18.37	0.97

Table 4.9. : Difference of packets dropped between JFEPM and Sack

The performance of JFEPM with Sack for number of packets dropped is shown in Figure 4.12 and Table 4.9. As shown in Table 4.9, Avg difference for number of packets dropped between JFEPM and original Sack is reduced with the increase of packet size. The maximum difference of packet drop is achieved on heavy load while the maximum percentage is achieved on moderate load. When the packet size is 800 bytes JFEPM is able to reduce number of dropped packets by a factor of 0.93. Similarly for packet size 1000 bytes the same factor is 0.95; for 1200 bytes packets this factor is 0.95; for 1460 bytes packets this factor is 0.97 for dropped packets.

4.7 SUMMARY

JFEPM opportunistically utilizes the oversubscribed topology structure of data center. This chapter shows that the approach of jumbo frames is helpful to reduce the latency of the flows in the data center networks when the buffer size at the switch is equal to the number of hosts in the rack. With the help of jumbo frames, frequent switching on higher level switches can be reduced. JFEPM does not require any change at end hosts. However it requires modification in buffer management policy at ToR switch only. JFEPM does not require any prior information about flow like deadline or size. This scheme treats all types of flows equally, so it is a deadline agnostic approach to reducing flow completion time. For the sake of validation of the scheme a simple analytical reasoning is presented, as well as simulation is performed with multiple well known TCP versions. Simulation results show that JFEPM is helpful to scale down the latency in data center. The amount of benefit given by JFEPM is different for different TCP versions. This reduction in average flow completion time would encourage application developers to invest more time for processing. As well service provider can also meet their service level agreements.

...