5
Flow-level Adaptive Routing in Data center

5.1 INTRODUCTION

Recent Data center network topologies put forward multiple paths between the end hosts to provide high bisection bandwidth. For the sake of utilizing the multiple paths efficiently many load-balancing techniques are proposed in the literature. These load-balancing proposals mainly deal with distribution of traffic among multiple paths. Further, a good load-balancing technique scales down the latency of flows. While balancing the traffic among multiple paths, a few proposals also take care of congestion. Flow-level solutions avoid flow splitting to prevent the packet re-ordering at the destination. Such adaptive solutions try to re-route (change the path) flows dynamically based on the feedback received from the network. In case of link failures, re-routing of the flow based on the queue size of the remote switch alone may be sub-optimal. The path to be chosen among multiple paths for re-routing, also poses a valid question. This chapter proposes FlowFurl, an adaptive, distributed, end host driven and flow-level load-balancing technique. This adaptive routing technique supports dynamic flow-to-path assignment. This flow-to-path assignment is performed not only based on the congestion at the switch, but also the status of the links along the path. FlowFurl is the first technique to combine link failure and congestion information parameter for re-routing of flows. The proposed technique is deployment friendly, however it incur small amount of overhead. This chapter shows that utilizing connectivity information with congestion information, and choosing a new path for flow based on local information, can help to design better load-balancing scheme.

5.2 MOTIVATION

At the data center networks multiple paths of equal cost exist between end hosts. In adaptive routing a flow can alter the path in response to a change in network conditions. Changing the path, only with congestion information may be sub-optimal. Adaptive routing only based on local congestion signal is not sufficient. Further a naive path selection approaches may lead to increase the number of re-routing events and the latency of the flow.

5.2.1 Data center Topology

Today's data center topology is a multi-rooted tree like structure (Fat-Tree, Clos). In such structures core switches are roots of the tree and end hosts or servers are the leaf nodes of the tree. This topology provides multiple parallel paths of equal cost between any pair of hosts. This work uses Fat-Tree topology [Al-Fares et al., 2008] for protocol description and simulation purpose. A Fat-Tree topology can be divided into four layers, Host or Server, Top of Rack, Aggregation and Core layers. A Fat-Tree topology is built from a large number of *k*-port switches. There are k pods which are interconnected with $k^2/4$ core switches. A pod is a management unit and is a replicable building block of topology. It consists of k/2 aggregation and k/2 Top of Rack switches. All aggregation switches in a pod are connected with all Top of Rack switches. Each aggregation switch in every pod is connected with k/2 core switches and each Top of Rack switch is connected with k/2 hosts or servers. There are a total of $k^3/4$ hosts that can be connected with $5k^2/4$, k-port commodity switches [Al-Fares et al., 2008]. Fat-Tree offers $k^2/4$ equal length paths between any

pair of hosts in different pods. The advantage of the Fat-Tree topology is that it is a well laid-out and symmetric structure, which will help a switch to easily establish connectivity. Figure 5.1 represents a Fat-Tree topology with k = 4 ports. In this figure, there are 4 pods and each pod contains 2 aggregation switches and 2 Top of Rack switches. These pods are connected through 4 core switches. In this topology, there are a total 16 of hosts, with 4 equal cost paths among the hosts of different pods.

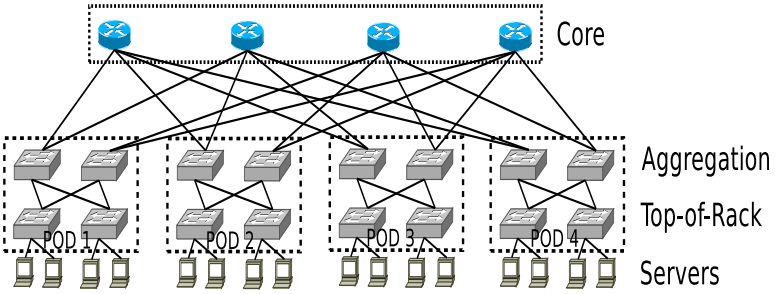


Figure 5.1. : Fat-Tree Data center topology

5.2.2 Adaptive Routing in Data center

Adaptive routing enables switches to select paths according to real-time situation. With adaptive routing a flow can alter the path in response to a change in network conditions. flow-level adaptive routing (DARD, FlowBender, DiFS) changes the path of a flow whenever congestion is detected. The congested switch sends information of local congestion, back along the path, to all the flows, without knowing the connectivity status of remaining path. Based on this information, the load-balancer (end host or switch) of the respective flows tries to re-route (shift) the flows. All the load-balancers receive the same information of congestion and they are not able to differentiate flows based on connectivity of remaining path. Changing the path, only with congestion information may be sub-optimal as shown in Figure 5.2.

As shown in the Figure 5.2, flows F1 and F2 originate from hosts of different pods 1 and 2. These flows are destined for different hosts of pod 4. Flows F1 and F2 share the buffer at switch Agg42 (right aggregation switch) in pod 4. Note that, in pod 4 there is no link between Agg41 (left aggregation switch) and ToR42 (right Top of Rack switch). The congested switch is unaware of this link failure and simply notifies the local congestion to both the sources.

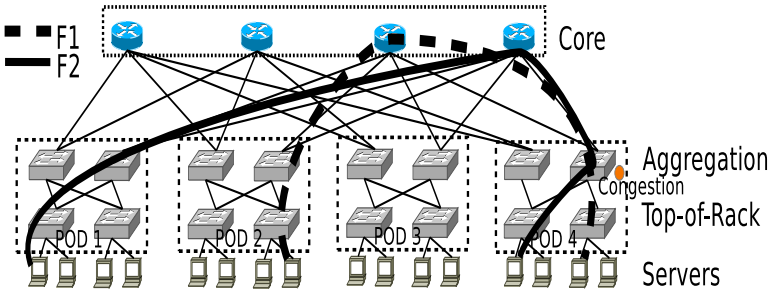


Figure 5.2. : Flow collision in faulty network

In this case, re-routing of flow F1 does not make any sense, because the last switch (ToR42) along the path of this flow is not connected with any other aggregation switch. On the other hand, re-routing of flow F2 is not problematic. Flow F2 can be easily re-routed through other core router

as shown in Figure 5.3. This example clearly illustrates that adaptive routing only based on local congestion signal is not sufficient. Adaptive routing should be able to use link failures for making re-routing decision in a multipath network. By incorporating failure information with congestion notifications, a load-balancer is able to make the correct decision to re-route a flow.

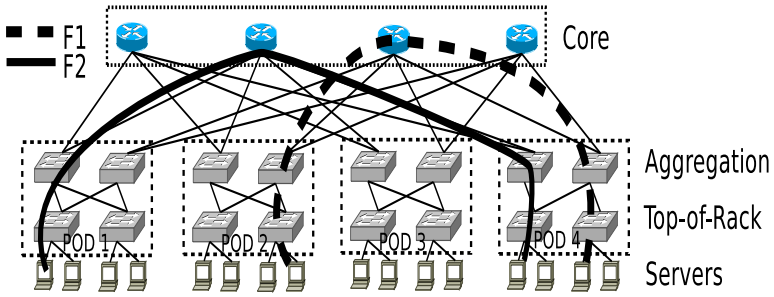


Figure 5.3. : Network after re-routing of flow F2

5.2.3 Naive path selection schemes

A naive scheme[1] decides the new path without prior knowledge of link condition. Such an approach may lead to re-routing of a flow multiple times before finding a suitable path. For example, in Figure 5.4, link B is over utilized and one of its flows is re-routed to other link. The width and color of the link represent utilization of the link. Link B is over utilized, link C and D are very near to full utilization such that small amount of load can lead to congestion, and link A is under utilized. In this situation a naive path selection scheme may choose either link C or D, or in the worst condition it may choose link C and D both (in any order) before opting for the suitable link A. Hence, such naive approaches may lead to congestion in the network and increase re-routing events and latency.

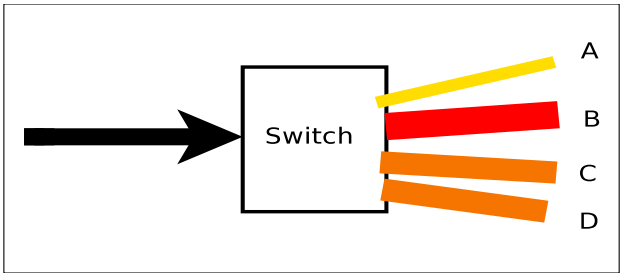


Figure 5.4. : An example of naive path selection scheme

5.3 FLOWFURL DESIGN

FlowFurl is a flow-level, end host driven load-balancing technique. The main contribution of this work is not only to find the congested flow with the help of ECN, but to design more fault-tolerant routing scheme. This fault-tolerant routing scheme provides better re-routing of flows in case of link failures. As it is already known, link failures have an impact on multipath connectivity of the network. Due to link failure, multiple paths between hosts are reduced. FlowFurl provides a better way of re-routing to reduce the effect of link failures. Previously proposed schemes allow re-routing only based on the feedback of congestion. In FlowFurl, the number of available paths

[1]select a new path as the next available path or any path randomly

(i.e., path multiplicity) for a flow is included in the congestion feedback, for taking the decision on re-routing at the sender. The number of times a flow is re-routed is proportional to the number of paths between the source and destination nodes. That is, flows with fewer paths to the destination will change their path less frequently. This proposal requires a small amount of memory at each switch to maintain the link state.

This work is divided into two parts, the first part (Basic) deals with when to re-route a flow and second part (Complete) besides the first one deals with how to decide the appropriate path for the flow. In Basic the source decides, which flow should be re-routed and when to be re-routed. FlowFurl uses ECN (Explicit Congestion Notification) to encode congestion and connectivity information. These ECN-bits are sent back by the receiver in the Acknowledgment (ACK). Further, FlowFurl assumes that every switch is capable to turn-off the ECN-bit, and all the network devices also have some information like layer, topology [2].

5.3.1 FlowFurl-Basic

The working of the FlowFurl-Basic is discussed in the following two subsections. While the first subsection discusses on obtaining the congestion and connectivity feedback from the network, the second subsection discusses the processing of this feedback information for better re-routing of flows.

Network Feedback

In this proposal, a source receives feedback from the network in the form of ECN-bits. Whenever a switch detects its queue length greater than a threshold, it sets the ECN-bit and the Preswitch bit in the packet header. This ECN-bit may be turned-off by the immediately next switch along the path. The purpose of Preswitch bit is to provide ECN turn-off capability to only the next switch. The main idea of FlowFurl-Basic is to turn-off ECN-bit inversely to the link connectivity. Hence, when connectivity is full, no ECN-bit is turned-off and when connectivity is less, ECN-bit is turned-off. In this way, a source receives more ECNs for congested flows having more number of paths and vice versa.

The Fat-Tree is a well defined topology in the data center. In Fat-Tree both ToR and Aggregation switches can be connected with maximum k/2 higher layer switches. Algorithm 3 presents the processing of information and setting ECN-bit and Preswitch bits at switch in Fat-Tree topology. Same idea can also be extended to any other topology with its own structural information.

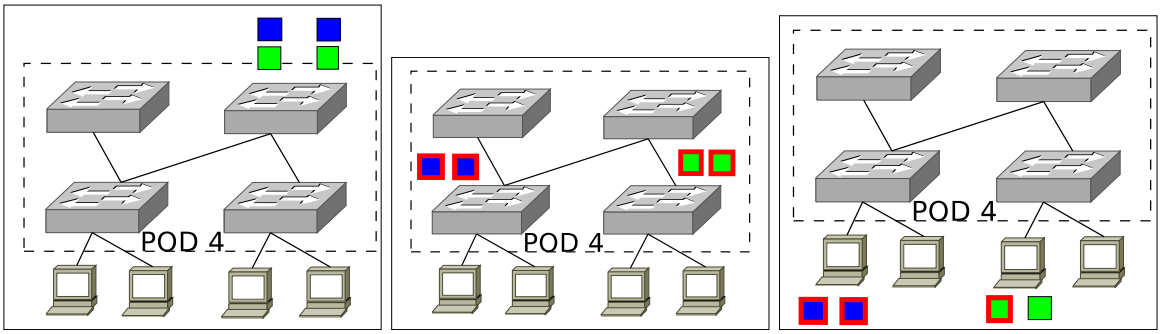
In FlowFurl-Basic, besides marking ECN-bit using threshold, a switch is also capable of checking the connectivity with higher layer switches. The connectivity check takes place at the immediately next switch of the congested switch. For connectivity check, this algorithm uses MyNeighbor function. This function either returns 0 if there is no link, or returns 1 if there exists a link between the two switches. MyNeighbor function can use the next hop field of the routing table or some information exchange protocol for finding the connectivity with neighboring switches. The connectivity information with higher layer switches is maintained in a variable named Totalpaths.

Along with simple ECN marking technique which is based on threshold, this algorithm with the help of Totalpaths measures Connectivity Ratio of a switch, which is the ratio of twice of the Totalpaths and the number of ports per switch k. When the ratio is 1; a switch is connected to all the higher layer switches. The minimum value of Connectivity Ratio is 2/k, which happens when the switch is connected with only one higher layer switch. FlowFurl-Basic randomly turns-off the ECN-bit based on Connectivity Ratio. For this purpose a Uniform Random Number (URN)

[2]Switches and host know about k (ports per switch), its layer (i.e. ToR, Aggregation, Core)

Algorithm 3 Processing at Switch

```
if ECN-bit is set && Preswitch bit is set then
    Totalpaths = 0
    for Each Higher Layer Switch S_i do
        Totalpaths += MyNeighbor(S_i)
    end for
    URN = Uniform Random Number (0,1)
    Connectivity Ratio = (2 * Totalpaths) / Port per Switch(k)
    if URN > Connectivity Ratio then
        Turn-off ECN-bit
    end if
    Turn-off Preswitch bit
end if
if Queue length ≥ Threshold then
    Set ECN-bit
    Set Preswitch bit
end if
```



(a) Before ECN Marking (b) After ECN Marking (c) Effect of Flowfurl

Figure 5.5. : An example of FlowFurl Algorithm

between 0 and 1 is generated. If this variable is higher than the Connectivity Ratio then the switch turns-off the ECN-bit. Finally, the switch always turns-off Preswitch bit irrespective of ECN-bit.

The approach of FlowFurl is reactive, which means Algorithm 3 checks for connectivity only after congestion happen. This reactive nature of FlowFurl is help to reduce the complexity of proposed approach. The complexity of this algorithm is O(Port per switch/2) which is linear. For better understanding of algorithm an example is shown in Figure 5.5. The example shows the working of Flowfurl-Basic. Here we refer the situation in Section 5.2.2, in which there is no link between Agg41 (left aggregation switch) and ToR42 (right Top of Rack switch) in Pod 4 Figure 5.5(a). Switch Agg42 is congested and set ECN-bit and Preswitch bit of 2 packets each of flow F1 and flow F2. ECN-bit enabled packets are represented by the red boundary of packets in Figure 5.5(b). When these packets reach at switches ToR41 and ToR42, they apply Algorithm 3. Switch ToR41 is connected to 2 aggregation switches (connectivity is full). In this case switch ToR41 sends both ECN packets to its destination. Switch ToR42 is connected with only 1 aggregation switch so its connectivity is not full. In this case, switch ToR42 turns-off the ECN-bit of received packets

randomly Figure 5.5(c) and sends to its destination. This algorithm may turned-off an ECN-bit of more than one packet in any order. These ECN-bits are sent back to the source by the destination. In this case, source of flow F2 receives more ECN-bits (within a transmission window) compared to flow F1. Based on these ECN-bits, a source will decide to re-route the flow or not, as described in next subsection. In this scenario chances of the re-routing of flow F2 is high compared to flow F1 because, source of flow F2 gets more ECN-bits compared to the source of flow F1.

Re-routing of Flow

The decision of whether a flow should be re-routed or not is made at the source of a flow. A source collects ECN-bits from ACK for every transmission window, and calculates fraction of ECN marked packets as in Data Center TCP (DCTCP) [Alizadeh et al., 2010]. For decision of re-routing, a source checks two conditions. First, if the fraction of ECN marked packets is higher than $\frac{2}{\text{Port per Switch(k)}}$ and, second if the last re-route decision is taken place before the current round trip time (RTT). When these two conditions are met, the source initiates re-routing of the flow by setting Newpath bit in the next packet header. At the same time, source also updates its local variables like last re-route time and total number of re-routes.

Algorithm 4 Processing at source

```
for Every Transmission Window do
  Compute ECN marked packets
  if (number of ECN marked packets / total packets transmitted) > (2 / Port per Switch(k)) && (current time − last re-route time) > RTT
  then
    Set Newpath bit
    last re-route time = current time
    total re-route++
  end if
end for
```

A source in FlowFurl-Basic depends on the switches for re-routing of the flows. The source sets Newpath bit which is a signal for switches for changing the path of flow. Whenever a switch receives a packet with Newpath bit, it selects the next available port to forward the packets of this flow in FlowFurl-Basic. FlowFurl-Complete, which is discussed in the next subsection, apply some process at the switch for selecting new path. If the switch has succeeded in changing the path of flow, it turns-off the Newpath bit. The switch also updates this flow-to-path assignment in its local record for future purpose.

In Algorithm 4 first condition helps to prevent the frequent path switching and re-routing of a single path flow. As shown in Algorithm 3 ECN-bit of packets are turned-off with the help of a uniform random number. ECN-bit of packets are turned-off inversely proportional to connectivity. For lower connectivity ECN-bit of more packets are turned-off and source of this flow collects remaining ECNs. If a flow has only one path to destination, the fraction of ECN marked packets received at source is always less than $\frac{2}{\text{Port per Switch(k)}}$. In this way re-routing of a flow having a single path can be prevented. Second condition prevents multiple re-routing of a flow in a single RTT (round trip time). This condition ensures RTT time difference between any two re-routing events. Source can receive traffic (congestion) condition of new path at least after an RTT time. Re-routing earlier than RTT may lead to aggressive path switching and improper resource utilization.

Theoretical Analysis

The scheme of ECN with threshold in data center is already used in previous proposals like DCTCP [Alizadeh et al., 2010] and FlowBender [Kabbani et al., 2014]. This analysis is based on the previous work [Alizadeh et al., 2011] [Zhang et al., 2016]. It models the number of ECN marked packet of a flow for FlowFurl-Basic. It also considers N_f synchronized flows responsible for congestion at the switch on a path. We consider Fat-Tree topology with link capacity C (in packets/sec) and propagation delay p_d (in sec). The threshold for marking a packet is denoted by M_T (in packets). In steady state, the window size of flow i at time t (measured in units of RTT) is denoted by $W_i(t)$, where $i \in \{1,2,3,....,N_f\}$. The total number of packets P_t at time t in transit can be written as

$$P_t = \sum_{i=1}^{N_f} W_i(t)$$

The maximum number of packets that can be accommodated on a path of end-to-end host in Fat-Tree topology is

$$W_{Max} = Cp_d + sM_T$$

where s is the number of switches on the path between end hosts. If any packet is injected in path after accommodating W_{Max}, the packet will surely get marked. Similarly, the minimum number of packets, up to that number on packet is marked is given as.

$$W_{Min} = Cp_d + M_T$$

If the total number of packets in transit in a path between two end hosts is fewer than W_{Min} no packet is marked. For $W_{Min} \leq P_t < W_{Max}$, packet marking probability $p(Marking)$ is defined as

$$p(Marking) = \frac{P_t - W_{Min}}{W_{Max} - W_{Min}}$$

Hence,

$$p(Marking) = \begin{cases} 0 & \text{if } P_t \leq W_{Min} \\ \dfrac{P_t - W_{Min}}{W_{Max} - W_{Min}} & \text{if } W_{Min} \leq P_t < W_{Max} \\ 1 & \text{if } P_t > W_{Max} \end{cases}$$

Expectation of this function is

$$E[p(Marking)] = \sum_{P_t = W_{Min}}^{W_{Max}} \frac{(P_t - W_{Min})^2}{W_{Max} - W_{Min}}$$

For simplicity this analysis uses $E[p(Marking)]$ is number of packets when receiver of a flow received ECN marked packet. This analysis assumes, once congestion reaches at certain level in path at t^{th} RTT, then in next $(t+1^{th})$ RTT, the flow will be re-routed and the total number of packets on the path will be reduced. Note that $E[p(Marking)] < W_{Max}$, so theoretically $P_t < W_{Max}$. Then, if no packet is marked at time t^{th} RTT,

$$P_t \leq E[p(Marking)]$$

and the total number of packets on the path in next RTT will be

$$P_{t+1} = \sum_{i=1}^{N_f} W_i(t+1)$$

$$P_{t+1} = \sum_{i=1}^{N_f} (W_i(t)+1)$$

$$P_{t+1} = P_t + N_f$$

Assuming that t^{th} is the RTT just before a congestion event then in next $(t+1)^{th}$ RTT, the total number of marked packets $T(m)$ are

$$T(m) = P_{t+1} - E[p(Marking)]$$

where $T(m)$ is the total number of marked packets. For the i^{th} flow, the proportion of marked packets that its sender calculates at the next $((t+2)^{nd})$ RTT is

$$F_i(t+2) = \frac{T(m)}{N_f(W_i(t)+1)}$$

$$F_i(t+2) = \frac{\sum_{i=1}^{N_f} W_i(t) + N_f - E[p(Marking)]}{N_f(W_i(t)+1)}$$

Here connectivity ratio C_R of a switch is define as a ratio of the number of up links and the total number of links in one direction.

$$C_R = \frac{\text{number of working links}}{\text{total number of links}(\frac{k}{2})}$$

With FlowFurl, a flow in fully connected Fat-Tree topology ($C_R = 1$) receives $F_i(t+2)$ marked packets. However link failure impact on the connectivity, and leads to affect the connectivity ratio of the switch. In FlowFurl, a switch may turn-off the ECN-bit uniformly based on the connectivity ratio. Hence, if connectivity ratio at the next switch of congested switch is less than 1, then the total number of ECNs turned-off by the next switch of flow i along the path are $(1-C_R) \times F_i(t+2)$. The remaining ECN marked packets $(F_i'(t+2))$ reaches at sender of flow i, are

$$F_i'(t+2) = C_R \times \frac{\sum_{i=1}^{N_f} W_i(t) + N - E[p(Marking)]}{N_f(W_i(t)+1)}$$

With help of these marked packets a source will make the re-route decision.

5.3.2 FlowFurl-Complete

In FlowFurl-Basic congested flow is re-routed to the next available path. Assigning next available path as new path after re-routing may lead to multiple times re-routing and reason of performance deterioration of flow. FlowFurl-Complete along with FlowFurl-Basic uses local information of the link to find the most appropriate path as new path while re-routing. The mechanism of the FlowFurl-Complete is divided into two subsections, first subsection deals with what information should be collected and how to maintain this information, and the second deals with how to process this local information to find a better path.

Information Collection and Maintenance

In FlowFurl-Complete each switch is responsible for maintaining the information of links. Every switch stores Rate (R), Last Flow Re-route Time (LRT), Recent Flow Arrival Time (RAT), and Total Number of Re-routes (TR) information of every link. The Rate and Total Number of Re-routes are updated periodically with the help of temporary counters (Temp Rate (T_R), Temp Re-route (T_{NR})) after every Δt amount of time. While forwarding a packet over a link, size of the packet is added to the counter T_R. Similarly, counter T_{NR} is incremented by one when a flow of this link is re-routed to some other link. Hence after every Δt amount of time, Rate and Total Number of Re-routes for every link is calculated using exponentially weighted moving average (EWMA) as

$$R_{new} = \alpha_s * R_{new} + (1 - \alpha_s) * T_R$$

$$TR_{new} = \alpha_s * TR_{new} + (1 - \alpha_s) * T_{NR}$$

where α_s is a constant which represents the weight of new data sample and smoothing factor. The range of α_s is between 0 and 1.

A link with a high Rate or high number of re-routes is inappropriate to choose for new path. Since the chance of congestion is higher with busy and unreliable link. A link with lower Rate and less Number of Re-route is the right candidate for selection of new path. Hence, FlowFurl-Complete chooses ideal and reliable link for re-routing of flow by enforcing lower the better policy on the Rate and Total Number of Re-route information.

Last Flow Re-route Time and Recent Flow Arrival Time are updated when a flow is re-routed to some other link and a flow is assigned to the link. At the time of re-routing, switch updates the Current Time as Last Flow Re-route Time and Recent Flow Arrival Time for previous and new links respectively. For finding an appropriate path, a switch can calculate Time since Last Flow Re-route (Re-route Time) and Time since Flow Arrive (Arrive Time) as

$$\text{Re-route Time} = \text{Current Time} - LRT$$

$$\text{Arrive Time} = \text{Current Time} - RAT$$

Link with low Re-route Time or low Arrive Time is inappropriate to choose for new path. A link is the latest victim of re-routing event, if it re-routes a flow or accommodate a flow recently. The experienced victim links are safe to use as new path because of stability and sustainability. Hence, while opting for a new path, FlowFurl-Complete prefers the links with higher Re-route Time and Arrive Time.

In this approach additional space is required to store this information. This information can be stored using a two dimensional array. The amount of space required to maintain the states of the link is O (Number of links × Number of Link attributes). FlowFurl-complete uses only 4 attributes of a link. Hence, the space required for this scheme is linear.

Information Processing

FlowFurl-Complete uses Analytic Hierarchy Process (AHP)[Kumar, 2014] to find an appropriate path. For a better explanation of process, we consider local information as matrix $M_{i\times j}$, where i is the number of links towards a destination and j is the number of attributes of each link. Every column of this matrix could be considered as a vector of one attribute of all links and n^{th} element of the vector representing value of this attribute for n^{th} link. As we have already discussed that, a link with lower Rate and less Number of Re-routes (lower the better) as well as higher Re-route Time and Arrive Time (higher the better) is most suitable. Hence, in this approach vectors Rate and Total Number of Re-routes are normalized for negative criteria and vector Re-route Time, Arrive Time are normalized for positive criteria. FlowFurl-Complete uses ∞ norm ($\|.\|_\infty$) for normalization of a vector, which is defined as

$$\|\vec{x}\|_\infty = Max\{|x_1|, |x_2|, ..., |x_n|\}$$

where x is $1 \times n$ vector and x_i is i^{th} element of vector.
A vector $\vec{x}$ normalized with positive criteria (higher the better) as

$$\hat{x} = \frac{x_i}{\|\vec{x}\|_\infty}$$

Similarly, normalization for negative criteria (lower the better) is defined as

$$\hat{x} = \frac{\|\vec{x}\|_\infty}{x_i}$$

After this process, FlowFurl-Complete gets normalized matrix $\hat{M}_{i\times j}$.
Here a term weight vector ($W_{j\times 1}$) is defined as importance of different attributes. This vector gives w_i weight to the i^{th} attribute of a link such that $\sum w_i = 1$. The weight vector is the input in this process and provided by the network administrator at setup time. An automatically calculation of w_i at the switch is future scope of this work, in which a switch can learn and adapt the value of w_i. With the help of weight vector, total score (S_i) of link i can be found as

$$S_i = \sum_1^j \hat{M}_{i\times j} * W_j$$

The top scorer link can be found as $Max\{S_i\}$. The top scorer link is suggested as the most appropriate path for re-routing the flow.

5.4 EVALUATION

For simulation of FlowFurl, NS-2 is used and this idea is compared against Equal Coast Multipath (ECMP), Random Packet Scatter (RPS) and FlowBender. The results of our simulations are divided in two subsections FlowFurl-Basic and FlowFurl-Complete. The Fat-Tree network with

8-port switches is simulated as shown in Figure 5.6, hence the network has total 128 servers. These servers are organized into eight pods, each having four Top of Rack (ToR) and four aggregation switches, with sixteen core switches interconnecting the pods. The link failure is incorporated above the ToR layer links randomly using uniform random distribution. For this setup, link failure probability is 0.2.

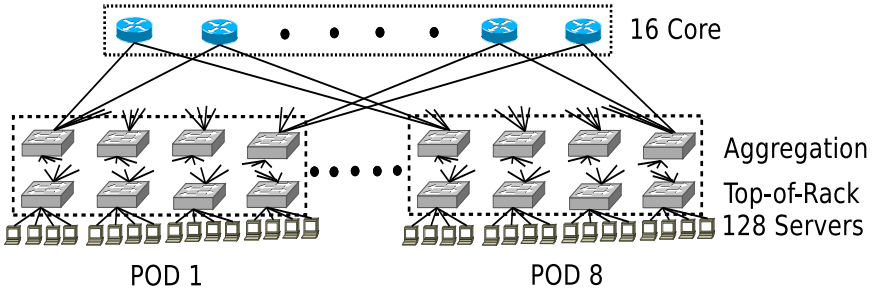


Figure 5.6. : Topology used for simulations

In this simulation 10 Gbps point-to-point Ethernet links are used across the entire network, as 10 Gbps Ethernet is commonly used in today's data centers. The minimum round trip propagation delay (without queuing) for a flow between between any end hosts going through core switch is 90μs, which is realistic in today's data centers.

5.4.1 Simulation Setup

CDF	Number of packets send
0.15	6
0.20	13
0.30	19
0.40	33
0.53	53
0.60	133
0.70	667
0.80	1333
0.90	3333
0.97	6667
1.00	20000

Table 5.1. : Empirical traffic distribution

The traffic workload used in this simulation is already observed in the data center. This workload uses all to all communication traffic pattern that is mainly replication of large scale online services like web search. Flow size distribution in this setup is heavy-tailed and is modeled based on [Alizadeh et al., 2010]. Every source randomly picks destination in the network to send data to, and flows arrive at the sender as a Poisson process with the mean adjusted with load. The load varies in simulation to change the arrival rate of flows and to achieve desired level of load in the network. The deadline of flows follow the exponential distribution with 200ms mean [Alizadeh et al., 2010]. The simulated workload is a mix of mice and elephant flows as shown in Table 5.1. In this workload, the size of 30% of flows is 1-20 MB and these flows are accounted for 95% of the total bytes transferring in the network. Drop-tail queue with a maximum size of 100 packets (100KB) is used in the simulation, the buffer occupancy threshold for setting the ECN-bit, is set to 90 KB.

For better understanding of results, in this section, the traffic is classified by flow size in four categories, (i) less than 10KB, (ii) between 10 KB to 128KB, (iii) between 128KB to 1MB, and (iv) greater than 1MB. The flow distribution at different loads can be noticed in Figure 5.7. One can observe that, roughly 52% of flows are between size of 10KB to 128KB, 26% of flows have size more than 1MB. Other two groups of flows, between 128KB to 1MB and less than 10KB, contribute in total traffic is around 15% and 7% respectively.

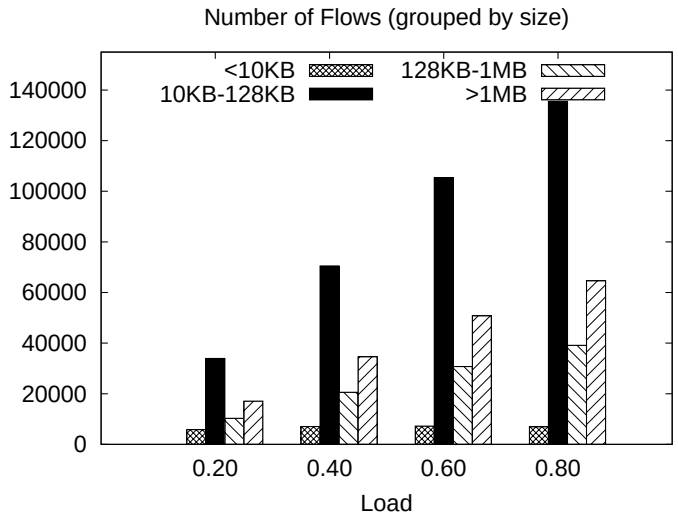


Figure 5.7. : Flow distribution with size at different load

5.4.2 Simulation Results

The number of packets dropped, mean flow completion time (average time to complete a flow), tail flow completion time (highest time to complete a flow) and percentage of deadline achieved are the performance metrics that are used to compare FlowFurl-Basic with different schemes (ECMP, RPS and FlowBender). Further, FlowFurl-Complete is compared with FlowFurl-Basic to analyze the effect of path decision scheme.

FlowFurl-Basic

The working of FlowFurl-Basic is already described in Section5.3.1. In all the graphs of this section, FlowFurl-Basic is termed as FlowFurl. The mean latency of flows is shown in Figure 5.8. In this setup, when link failure probability is 0.2, RPS and FlowBender is able to reduce mean latency up to 79%. Whereas FlowFurl-Basic is able to reduce mean latency up to 86%, which is comparatively 69% of FlowBender.

In distributed computing, completion of any task is highly depended on the tail latency of flows. The tail latency of flows is shown in Figure 5.9. The effect of link failures can be observed in the diagram. FlowFurl-Basic roughly reduces tail latency up to 80%, while RPS and FlowBender are able to reduce it up to 69% only. The average throughput of the flows of FlowBender and FlowFurl-Basic is nearly equal as shown in Figure 5.10. For small size flows, the performance of ECMP is also same as other three. However for large size flows at moderate load, adaptive mechanisms get advantage of re-routing, and performance of ECMP collapses due to static nature.

In any network, large number of packet drop increases timeout events. Timeouts are costly events, which are liable for rising of the flow latency. The total number of packets dropped is shown in Figure 5.11. For small size flows, the number of packets dropped is negligible compared to large

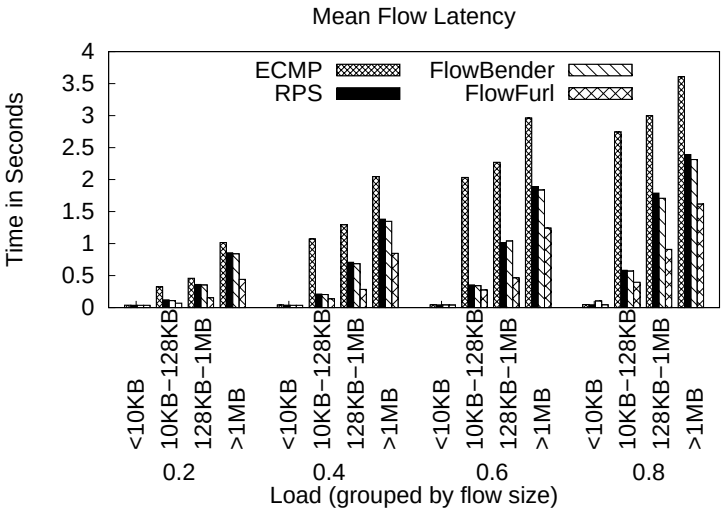


Figure 5.8. : Mean flow latency

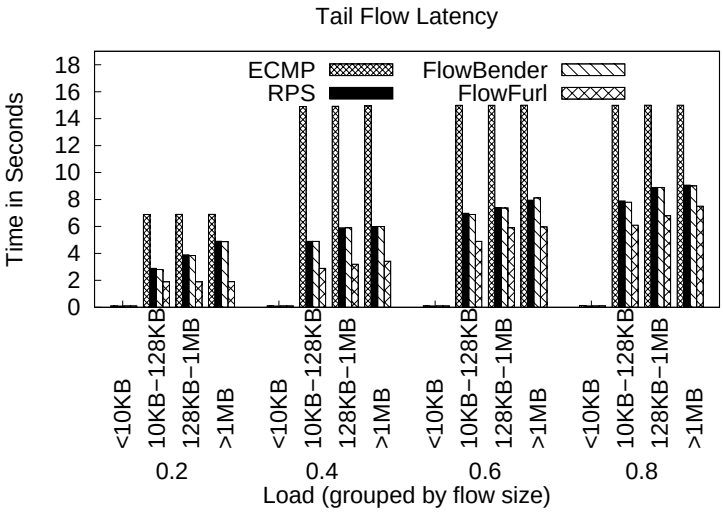


Figure 5.9. : Tail flow latency

flows. FlowFurl-Basic reduces the number of packets dropped up to approx 47%, while RPS and FlowBender are able to reduce the number of packets dropped up to 37% only.

The total number of re-routing events in the network is shown in Figure 5.12. In this setup, when link failure probability is 0.2, path multiplicity (the number of paths between end hosts) of flows is reduced to a small amount. Due to this reason, FlowFurl-Basic initiates re-routing process fewer number of times compared to FlowBender. Re-routing decision in FlowFurl-Basic is a bit conservative than FlowBender.

Figure 5.13 shows the deadline achieved (%) in this setup. For small size flows (less than

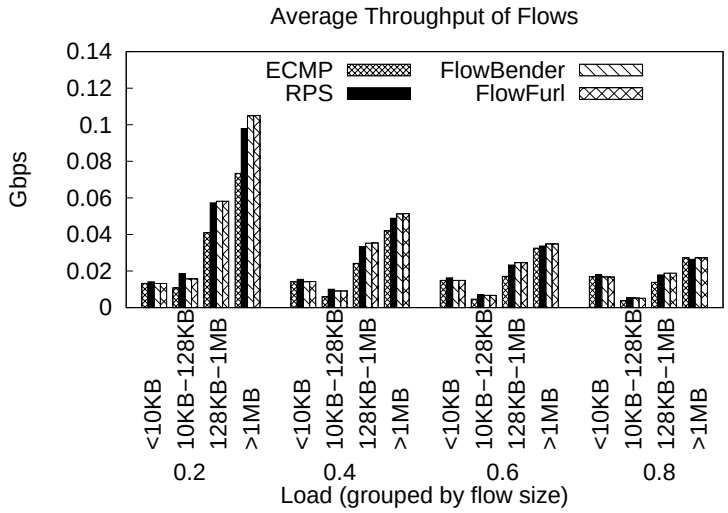


Figure 5.10. : Average throughput of flows

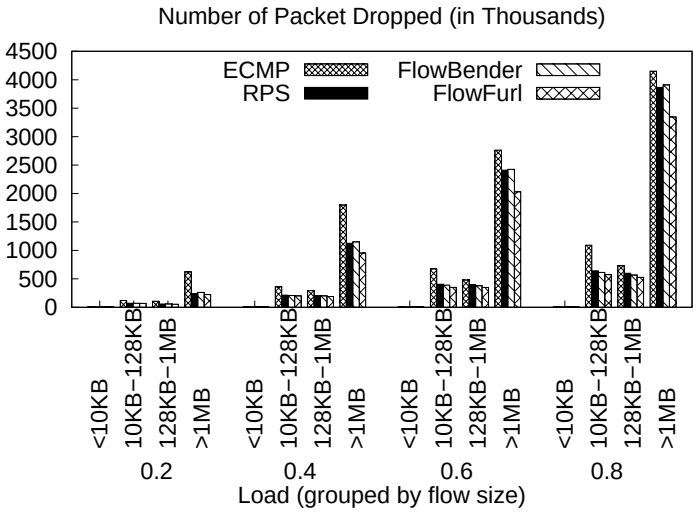


Figure 5.11. : Total number of packets dropped

10KB), the deadline achieved by all schemes are same and more than 90% of flows are achieving the deadlines. However for higher size flows, FlowFurl-Basic achieved more deadlines compared to other schemes.

FlowFurl-Complete

FlowFurl-Complete is already described in Section 5.3.2. For simulation of FlowFurl-Complete the topology and traffic parameter are considered same as of FlowFurl-Basic. Additional parameters used to simulate FlowFurl-Complete is $\alpha_s = 0.5$ and $\Delta t = 0.1s$. In this simulation, equal weights to all attributes of the link are assigned. Hence in this case $w_i = 0.25$, because FlowFurl-

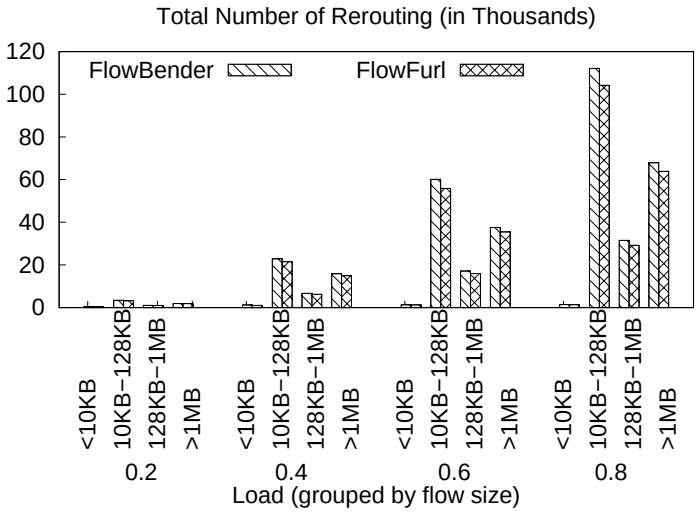


Figure 5.12. : Number of re-routeing events

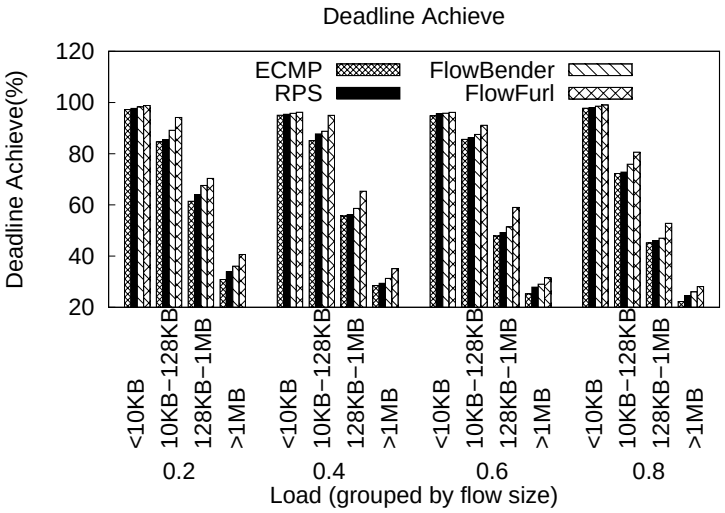


Figure 5.13. : Deadline achieve

Complete uses 4 attributes of the link (Rate, Re-route Time, Arrive Time, and Total Number of Re-routes). FlowFurl-Complete is compared with FlowFurl-Basic to analyze the effect of path decision scheme. In graphs FlowFurl-Basic and FlowFurl-Complete are represented as "Basic" and "Complete" respectively.

Average and tail flow latency are shown in the Figure 5.14 and Figure 5.15 respectively. These latencies are normalized with respect to those of the ECMP. In both the figures, one can observe that at a low load, performance of FlowFurl-Complete is almost the same as FlowFurl-Basic. At low load, when the chance of ideal links is high, shifting a congested flow to the next available link is impacted same as assigning a new path intelligently. At a higher load, when the

chance of congestion is higher due to more number of flows, FlowFurl-Complete performs better than FlowFurl-Basic. Assigning a new path to a congested flow intelligently at higher load (when most of the links are busy) increases the chances of better path assignment.

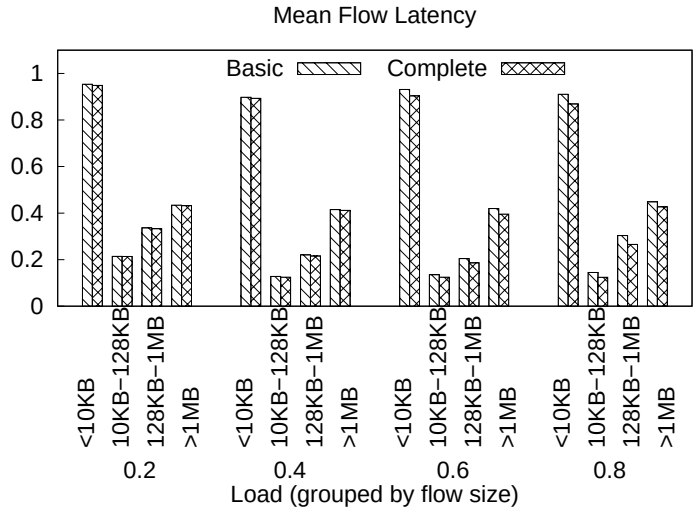


Figure 5.14. : Mean flow latency normalized to ECMP

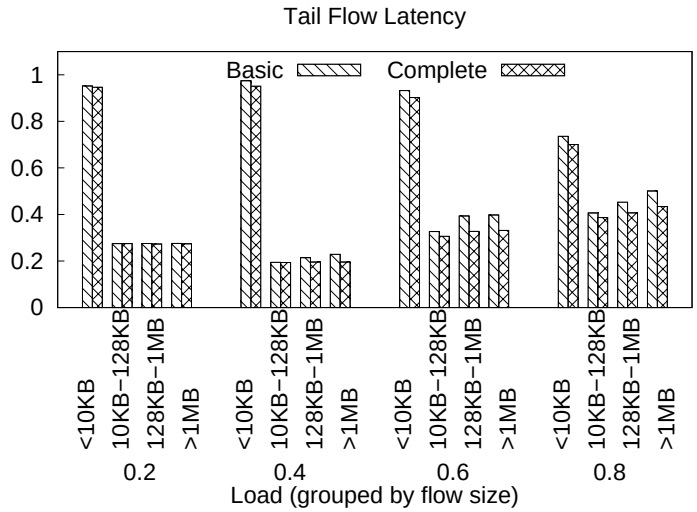


Figure 5.15. : Tail flow latency normalized to ECMP

Figure 5.16 shows the number of re-routing events for FlowFurl-Complete normalized to FlowFurl-Basic. Due to intelligent path selection scheme the total re-routing events are reduced. This path selection approach cuts down the events of multiple re-routing for a congested flow. In this way FlowFurl-Complete finds a good path in fewer attempts, and further helps the flow to finish quickly. In figure, FlowFurl-Complete reduces around 10% of re-routing events compared to FlowFurl-Basic.

In Figure 5.17, one can observe that FlowFurl-Complete achieved more deadlines (in %) compared to FlowFurl-Basic. This intelligent path selection scheme achieved up to 13.7% more

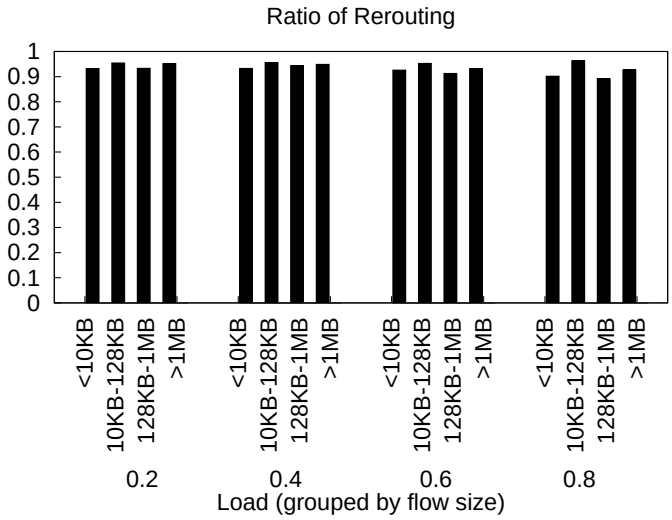


Figure 5.16. : Number of re-routeing events normalized to FlowFurl-Basic

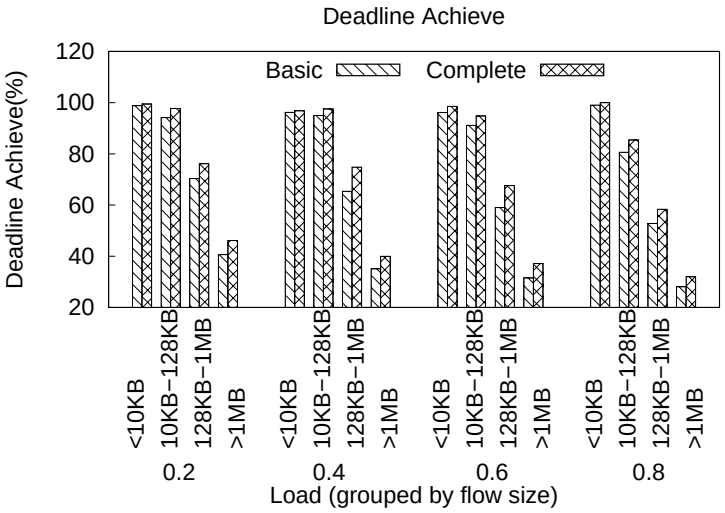


Figure 5.17. : Deadline achieve

deadlines for large size flows. However as flow size decreases the benefit of this path selection is also reduced.

Behavior of flow-level Switching

To better understand the working of this re-routing mechanism, a small scale simulation is performed. In this simulation, a Fat-Tree network with 4-port switches are used, as shown in Figure 5.18. There are total 16 servers in this topology and only two paths exist between any two servers of different pods. In this topology only two core switches are responsible for connecting

different pods, remaining two core switches are disconnected intentionally. Total number of flows in network are four, all flows having originated from pod 1 and terminated in pod 4. The source and destination node of flow 1 and flow 3 are server 0 and server 15 respectively. Similarly, source and destination node of flow 2 and flow 4 are server 2 and server 13 respectively. Flow 1 and flow 2 are going through the same core switch, while flow 3 and flow 4 are using another, core switch. All flows are same in size (1MB) and all started simultaneously at time 0.1 second. In this simulation, bandwidth of all the links are 1 Mbps and the buffer occupancy threshold for setting the ECN-bit, is set to 10 packets.

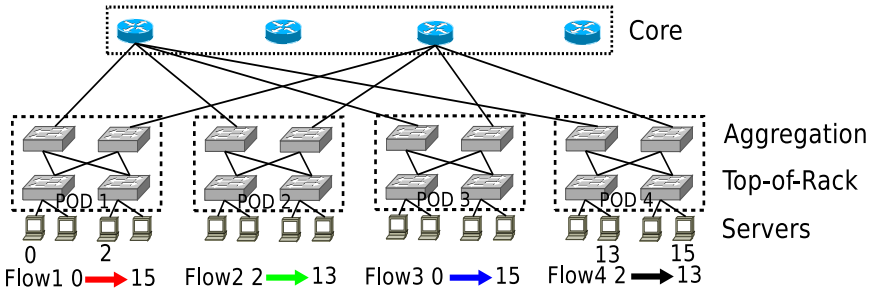


Figure 5.18. : Toy topology with only two path between end hosts

Figure 5.19 shows the number of ECN marked packets received at source for each flow with respect to time. All the flows started at the same time (0.1 second) and after some time ($\sim$ 1.4 second) sources start to receive ECN marked packets. Flow 1 is more aggressive compared to flow 2, because the size of the transmission window of flow 1 is large compared to that flow 2. At the source of flow 2, fraction of ECN marked packets cross the threshold at 9.54 second. After this moment of time, flow 2 changes its path and joins a new path with flow 3 and flow 4. When all three flows (i.e. flow 2, 3 and 4) are sharing the path, at the source of flow 3 the fraction of marked packets crosses the threshold at time 10.25 seconds. Hence, flow 3 changes its path and joins path with flow 1.

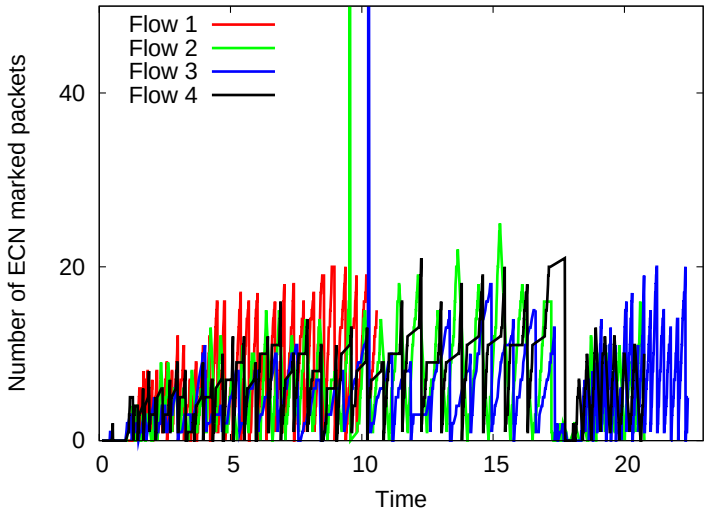


Figure 5.19. : Behavior of flow-level switching

5.5 SUMMARY

This chapter proposes FlowFurl, a flow-level routing scheme to handle congestion, with failures in data center networks. FlowFurl is a distributed, end host based, load aware scheme, and uses ECN as notification of congestion. This scheme does not require any hardware changes or complicated host mechanisms. This load-balancing relies upon congestion notification and connectivity information of the network. The idea is to treat flows differently, based on the connectivity of remaining paths, in such a way, that a source receives more packets with ECN-bit enabled for congested flows having more number of paths and vice versa. Hence re-routing of flows with less number of paths to the destination can be reduced. In the second part of this chapter, FlowFurl-Complete deals with how to choose a new path for re-routing. FlowFurl-Complete applies analytic hierarchy process on link level local information to decide the appropriate path for the flow. Such an intelligent scheme helps to reduce the re-routing events and latency. Simulation results of FlowFurl-Basic shows that combining connectivity information with congestion information helps to decide re-routing of correct flow.

...