Unfairness in TCP Performance over Lossy Optical Burst-Switched (OBS) Networks

Julie Sullivan, Paul Ramos, and Vinod M. Vokkarane
* *Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747, USA*
E-mail:{u_j7sullivan,pramos1,vvokkarane}@umassd.edu

***Abstract*—One of TCP's primary objectives is to provide fairness to all flows competing for resources in a network. Since the popular flavors of TCP were designed to work with electronic packet-switched networks, they behave differently when used over optical burst-switched networks (OBS). In OBS, burst loss occurs due to random contention of data bursts. Since TCP is unaware of the underlying physical media, the responsibility of fairness must be provided by the burst assembler. We investigate several flow-aware and flow-unaware mechanisms that improve TCP fairness over OBS.**

***Index Terms*—WDM, TCP, OBS, and IP**

I. Introduction

Optical burst switching (OBS) is a promising technology, having a huge potential integrating IP with wavelength division multiplexing (WDM) technology and offering huge bandwidth in the order of 50 THz. OBS has a granularity between optical circuit and optical packet switching networks, making it a pragmatic technology to be employed.

OBS network architecture consists of core nodes and edge nodes. Edge nodes handle all incoming traffic (like TCP/IP and UDP) and aggregate the incoming data into bursts.

Contention occurs when more than one burst is scheduled to go on the same output port at the same time. In electronic packet switching, the primary method for resolving contention is through buffering. In OBS, the contention resolution mechanisms include fiber delay lines, wavelength conversion, deflection routing, and burst segmentation [2].

TCP is the dominant transport protocol over IP and accounts for the majority of data traffic over the Internet. The need for a reliable transport protocol is evident when we think in terms of supporting a multitude of applications, especially ones that require high bandwidth and having long end-to-end delays. The fundamental assumption of the various TCP versions is that the underlying physical medium is electronic and the packets experience queuing delay at IP routers.

In TCP/IP over OBS, multiple TCP packets are assembled into a single burst and transmitted. A single burst loss would generally cause multiple packet losses. TCP reacts to the packet loss via a timeout (TO) or fast retransmit (FR). The specifics of the congestion control algorithms in the different versions of TCP have been discussed thoroughly in [5].

The various techniques TCP employs to deal with and avoid congestion provide fairness across all flows in the network. While these techniques are successful in providing fairness in an electronic network, they are unsuccessful in an OBS network when there are random burst contentions. Since TCP exists at the transport layer, and is unaware of the underlying physical medium, it cannot properly provide fairness for OBS. The responsibility of fairness must then be given to the underlying medium, which in the case of OBS is the burst assembly mechanism.

II. Previous Work and Topologies

Previous work has been done to study TCP fairness over OBS [6]. They show that TCP is generally fair over OBS, using Jain's Fairness index where b_i is the fraction of the capacity of the bandwidth flow i is using:

$$f = \frac{(\Sigma_{i=1}^{n} b_i)^2}{n * (\Sigma_{i=1}^{n} b_i^2)} \quad (1)$$

They use three different kinds of topologies; local aggregation, edge aggregation, and core aggregation. We will only pay attention to local and edge aggregations as they are the only topologies involving burst assembly among different flows. All topologies are dumbbell topologies.

A. Local Aggregation

Local aggregation concentrates all flows on a single TCP sender node. This node is connected to the ingress edge node with one link. At the other end, the egress edge node is connected to the single TCP receiver node.

B. Edge Aggregation

With edge aggregation, each TCP flow is sent from a separate node. There is only one flow per node.

III. Proposed Burst Assembly Mechanisms

We propose different burst assembly mechanisms to spread out the different flows among the bursts. This is intended to create fairness.

These assembly mechanisms would require a buffer for every flow, becoming unrealistic in a typical network with thousands of flows. Thus, we propose a set number of buckets which flows will map into. The burst assembler will take packets from these buckets, which may have many different flows in them (Fig. 1). Several different mapping policies from the flows to the buckets are proposed below.

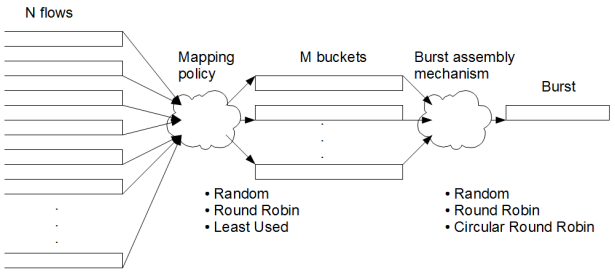


Fig. 1. Proposed Burst Assembly with Buckets

A. Mapping Policies

Here we discuss the different policies we can use to map the flows into the set number of buffers.

1) Flow unaware vs. Flow aware: Here we run into an out-of-ordering issue. If we allow the flows to map into any bucket, some flows may map to several different buckets. For example, Flow 1 may send a packet to Bucket 1, then send its next packet to Bucket 2. The second packet from Flow 1 may be put into the burst before the first packet.

To take care of this problem, we can introduce flow awareness. Once a flow has mapped to a specific bucket, that mapping is tracked, and all later packets from that flow will be sent to the same bucket. The mapping is tracked until the burst is sent, then it is reset. This way, flows will not be stuck with a single bucket for their entire duration. Both flow aware and flow unaware mapping policies will be considered.

2) Round Robin: The round robin (RR) mapping policy will take the flows as they come in and map them to the buckets in order. The first packet will map to the first bucket, then second packet to the second bucket, and so on. The flow-aware RR mapping policy will send the same flows to the same buckets. This mapping policy is expected to evenly distribute the flows among the buckets.

3) Least Used: The least-used (LU) mapping policy maps to the bucket with the least packets in it. This should help ensure that all buckets have flows in them, allowing for a more even distribution. With flow awareness, this mapping policy should be the most able to keep the buckets evenly populated.

4) Random: The random mapping policy will take packets from flows and place them in random buckets. This is the most likely to suffer from out-of-ordering without flow awareness.

B. Burst Assembly Mechanisms

1) Round Robin: A round robin (RR) burst assembler will take one packet from each bucket, starting from the first packet. It will skip any buckets without packets. RR burst assembly will ensure that several different flows end up in a packet, so long as there are several different flows in the different buckets.

2) Circular Round Robin: A circular round robin (CRR) burst assembler acts like a RR burst assembler, except that it changes the order of the buckets that it takes from first after each burst has been completely assembled. It will start off like a regular RR burst assembler, taking a packet from the first bucket first, then going through the rest of the buckets. However, on its next turn, it will start with the second buckets, then go through the rest of the buckets. It will continually change which bucket it starts from, circling through all buckets as the starting bucket. We expect this to be more fair than a RR burst assembler. This assembler guarantees that packets will not end up in the same place in every burst.

3) Random: A random assembler will choose at random which buckets to take packets from. If the picking among the buckets is evenly distributed, this should be a fair assembler. We do not expect it to be as fair as the circular round robin assembler. It may be more fair than the round robin assembler because the flows will continually change positions in the burst.

IV. Simulation Results

We used NS2 for our simulations and we forced random contentions. In our local topology, there is one node with 50 flows being sent from it, and in our edge topology, there are 50 nodes with one flow being sent from each. We have five buckets which the flows will be mapped into for assembly. All the optical links have a bandwidth of 1Gbps and a delay of 0.5ms. There are 16 data and control channels. Bursts are randomly dropped in the core.

Contention greatly affects fairness (Fig. 2).

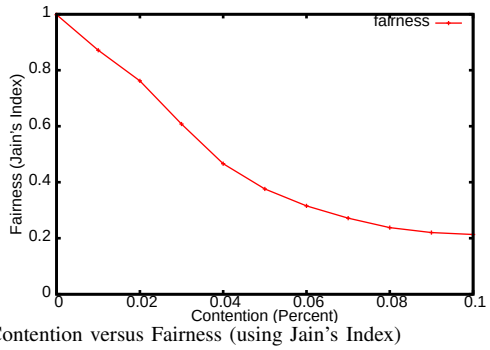


Fig. 2. Contention versus Fairness (using Jain's Index)

We found that the edge aggregation was consistently fairer than local aggregation, no matter what the contention rate was. Edge aggregation always performed very well, almost perfectly fair, for every variable we used. Thus, we will concentrate on local aggregation for the rest of the paper.

Next we compared flow awareness with no flow awareness. We found that no flow awareness performs better than flow awareness with any kind of random assembly or random mapping policy (Fig. 3(a)). Flow awareness performs better than no flow awareness without random assembly or random mapping (Fig. 3(b)).

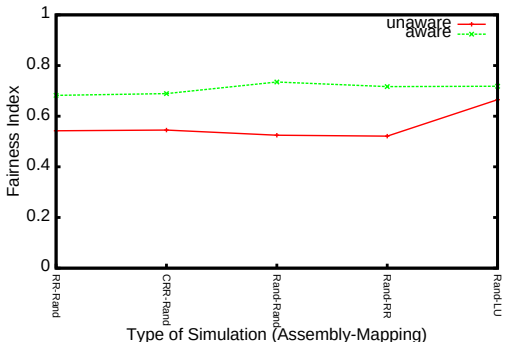
Next we will compare the mapping policies. These performed differently with flow awareness (Fig. 4(a)) than without flow awareness (Fig. 4(b)). When the policies are flow aware, the least-used mapping policy is consistently fairer than round-robin or random mapping policies. The assembly mechanism makes little difference. When the policies are not flow aware, the random mapping policy is consistently fairer than round-robin or least-used policies. The gap in performance closes considerably when we use the random assembly mechanism.

Comparing the assembly mechanisms, we found that when there is flow awareness, all assembly mechanisms perform similarly. Round-robin and Circular round-robin mechanisms perform marginally better than the random mechanism. Without flow awareness, the random assembly mechanism performs significantly better than the other two. The margin closes greatly in conjunction with the random mapping policy.

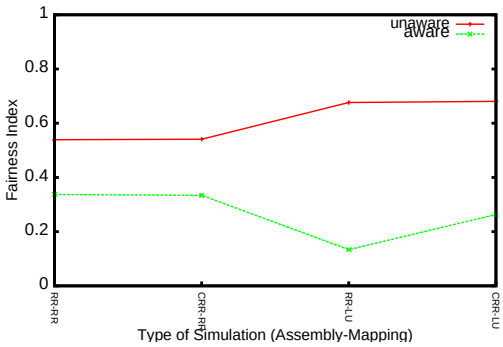
V. Conclusion

We have found that contention makes a big difference in fairness. Also, edge aggregation performs much better than local aggregation due to the fact that all flows are on one node in the local aggregation topology, thus all flows originate in the same place and are more likely to overpower each other.

We found that flow awareness performs much better when there is random mapping or assembly than when the random policy is not used. This is likely due to out-of-ordering. With random mapping and assembly, we are far more likely to have

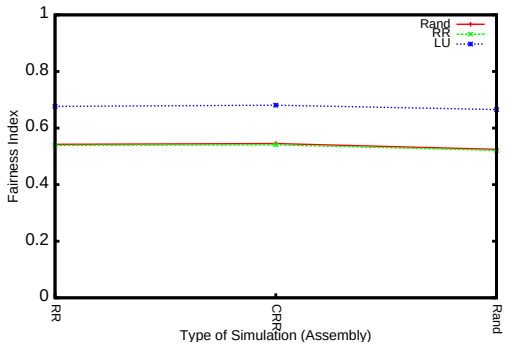


(a) With Random Mapping or Random Assembly

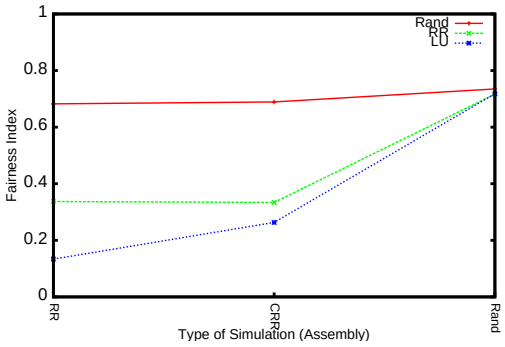


(b) Without Random Mapping or Random Assembly

Fig. 3. Flow Awareness Comparison



(a) Flow Aware



(b) Not Flow Aware

Fig. 4. Mapping Comparison

out-of-ordering, as packets can just go wherever they want to. However, with flow awareness, they are in order, so even though there is randomness, it should not be affected greatly. When there is no randomness, out-of-ordering is much less likely to occur, so flow awareness only puts a burden on the mapping. Thus, no flow awareness performs better when there is no randomness.

When we have flow awareness, the least-used mapping policy outperforms the other two due to the least-used policy's ability to fill the buckets more. With flow awareness, flows are destined to go to certain buckets once they have been mapped to it, so in the case of the random and round-robin mapping policies, some buckets may be empty. In the least-used policy, flows will occupy the emptiest buckets, making it less likely to have any empty buckets, increasing fairness.

When we do not have flow awareness, the random mapping policy outperforms the other two, especially when we use it in conjunction with the round-robin and circular round-robin assembly mechanisms. It is likely more fair to combine the random with a more structured assembly mechanism because we get a better distribution of flows in each burst. We can see this as well in the fact that the random assembly mechanism outperforms the other assembly mechanisms.

Though unexpected, when we do not have flow awareness, we found that the random assembly mechanism better distributes packets in the burst than the circular round-robin or round-robin assembly mechanisms, especially when we use the round-robin or least-used mapping policies. The round-robin and least-used mapping policies will not widely distribute the flows among the buckets like the random mapping policy. So when we use the round-robin or circular round-robin assembly mechanisms, the flows are still in a fairly predictable order entering the burst. Using the random assembly mechanism would mix up this order more, thus creating more fairness.

Using randomness with flow awareness seems to be the fairest approach to using TCP over OBS. Having flow awareness prevents out-of-ordering while random mapping policies or assembly mechanisms distribute the flows widely, creating more fairness.

REFERENCES

[1] C. Qiao et al,"Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *Journal of High Speed Networks*, Jan. 1999.

[2] V. M. Vokkarane et al, "Segmentation-Based Non-Preemptive Channel Scheduling Algorithms for Optical Burst-Switched Networks," IEEE/OSA Journal of Lightwave Technology (JLT), Special Issue on Optical Networks, vol. 23, no. 10, pp.3125-3137, Oct. 2005

[3] K. Fall et al, "Simulation-based Comparisons of Tahoe, Reno and Sack TCP," ACM SIGCOMM Computer Communication Review, vol. 26, pp. 5-21, July 1996.

[4] M. Mathis et al, "TCP Selective Acknowledgement Options," RFC 2018, 1996.

[5] B. Sikdar et al, "Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK," IEEE/ACM Transactions on Networking, vol. 11, no. 6, December 2003 , Pages 959-961.

[6] X. Yu et al, "TCP Performance over OBS Networks with Multiple Flows Input," Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on , vol., no., pp.1-10, 1-5 Oct. 2006