Congestion Control Scheme in Optical Packet Switched Networks

Zheng Lu, and David K. Hunter
Department of Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, UK
{zlu, dkhunter}@essex.ac.uk

Abstract: This paper proposes a congestion control scheme for optical packet switched networks, for use with edge smoothing and reduced core optical buffers. Simulation results indicate that it interacts favourably with TCP's congestion control mechanism.

OCIS codes: (060.4250) Networks

1. Introduction

Optical packet switching provides the benefits of flexibility and high capacity; however, it implies problems in a realistic network using TCP/IP, such as interaction with the elastic TCP congestion control mechanism. If congestion occurs in the optical network core, any associated loss may be resolved either via the TCP acknowledgement process or by a timeout. Many optical packets are lost before TCP can detect it, due to the high optical transmission speed. We now resolve this problem by proposing a TCP aware congestion control scheme, with intelligent edge buffering to reduce TCP segment loss under congestion, without modification to existing core switch buffer structures [1]. In [2], the authors use a link-utilization-based detection and transmission rate adjustor for TCP ACK segments. In our work, congestion control is based on interactively changing the sending rate under congestion at the time-scale of OPS. It treats all TCP/IP traffic as encapsulated data with no direct interaction with TCP segments. Therefore, it also works transparently with other protocols. Our TCP Aware Congestion Control will be referred to as TACC for short. The high switching speed of OPS often results in many optical packets being lost, due to the relatively slow operation of TCP. In slotted OPS, the slot time is several nanoseconds, whereas, the TCP round trip time (RTT) for a wide area network (WAN) ranges from several milliseconds to several hundreds of milliseconds, while TCP's retransmission timeout (RTO) is typically several seconds. Therefore, the total number of slots lost is $P_{overflow}(A) \times \min\{T_{RTT}, T_{RTO}\}/\tau$, with minimum loss rate of $P_{overflow}(A) \times 10^6$. A is the offered load, and τ is the OPS slot size. This has a serious effect on performance, especially under high load, since $P_{overflow}(A)$ increases with load. This is a problem in OPS networks due to the lack of large optical buffers. However, if congestion occurs, we do not merely buffer the traffic and wait the whole TCP RTT or timeout to recover it, since it is possible that the network will become less congested before this. For these reasons, TACC operates at the OPS time-scale – not the TCP time-scale. In our previous work [3, 4], the rate of traffic is predicted, and it is smoothed at the edge with negotiated sending rate. This is a prerequisite for the design of this congestion control scheme.

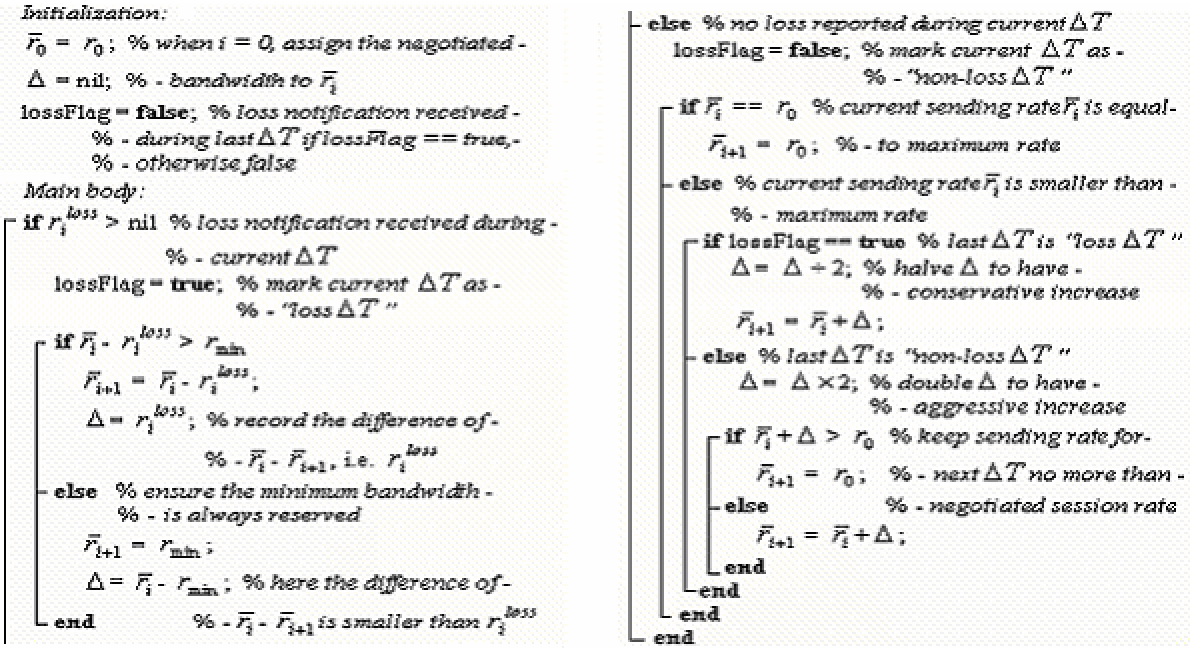
2. Traffic smoothing and TACC scheme

For traffic smoothing [3, 4], the future sending rate at the edge switch is estimated as: $R_{i+1} = \nu \hat{R}_i + (1-\nu)(R_i + B/\tau)$ with $\nu = \min\{(R_i - R_0)/R_{avail}, 1\}$ and $R_f = \Sigma_{\forall i}^{\lfloor T/\tau \rfloor} R_i / \lfloor T/\tau \rfloor$. R_0 is the present sending rate. R_i is the measured data rate during the ith estimation interval of length τ, which is much larger than an optical packet slot. R_{i+1} is the estimated sending rate for the next interval, estimated from previous traffic patterns. R_f is the mean estimated sending rate. ν is a factor which determines to what extent the estimate is dependent on previous experience; it also reflects the relationship between user traffic requests and network capabilities. B/τ is the extra rate required to empty the edge buffer, where B is the buffer occupancy. R_{avail} is the available bandwidth in the network, and R_{thresh} is the threshold for adopting the new estimated rate as the sending rate. If $|R_f - R_0| > \Delta R$, the sending rate changes to R_f; that constitutes a successful negotiation and traffic is sent into the core network with this estimated constant sending rate, otherwise it keeps the present sending rate of R_0. ΔR is the tolerable variation that is permitted without triggering re-negotiation; it reflects the maximum acceptable cost per negotiation. This edge traffic smoothing allows constant rate transmission into the core network, with stepwise changes between negotiation sessions with length $T_{session}$. TACC operates within this inter-negotiation interval.

Below, notation is defined to facilitate our discussion of TACC:

- r_0 – The reserved sending rate for the current session, i.e. the successfully negotiated rate for the current flow.
- $r_{\min}$ – The minimum allowed sending rate, reflecting the minimum bandwidth required for a traffic flow.
- ΔT – The interval during which TACC waits for possible loss notification, with a length of several OPS RTTs.
- $\bar{r}_i$, $\bar{r}_{i+1}$ – The sending rate during the current ΔT and the next ΔT respectively, with: $\bar{r}_i$, $\bar{r}_{i+1} \in [r_{\min}, r_0]$, $i \in \{ 0, 1, 2, 3, \ldots T_{session} / \Delta T \}$; r_i^{loss} – The loss rate during the current ΔT .
- Δ – The difference between $\bar{r}_i$ and $\bar{r}_{i+1}$, i.e. $\Delta = \bar{r}_i - \bar{r}_{i+1}$.

This algorithm is initialized at the beginning of the current negotiation session (of length $T_{session}$); at the end of each ΔT , the sending rate $\bar{r}_{i+1}$ for the next ΔT is found. For this reactive control algorithm, the rate set for the switch ports is (r_1 , r_2 , r_3 ,… r_{N-1} , r_N), where N is the number of ports. A simplified version of the algorithm is



```
Initialization:
  $\bar{r}_0 = r_0$; % when i = 0, assign the negotiated -
  $\Delta$ = nil;  % - bandwidth to $\bar{r}_i$
lossFlag = false; % loss notification received -
        % - during last $\Delta T$ if lossFlag == true,-
        % - otherwise false
Main body:
if $r_i^{loss}$ > nil  % loss notification received during -
                % - current $\Delta T$
    lossFlag = true;  % mark current $\Delta T$ as -
                % - "loss $\Delta T$ "
    if $\bar{r}_i - r_i^{loss} > r_{\min}$
        $\bar{r}_{i+1} = \bar{r}_i - r_i^{loss}$;
        $\Delta = r_i^{loss}$; % record the difference of -
                % - $\bar{r}_i - \bar{r}_{i+1}$, i.e. $r_i^{loss}$
    else  % ensure the minimum bandwidth -
          % - is always reserved
        $\bar{r}_{i+1} = r_{\min}$;
        $\Delta = \bar{r}_i - r_{\min}$; % here the difference of -
    end          % - $\bar{r}_i - \bar{r}_{i+1}$ is smaller than $r_i^{loss}$
else % no loss reported during current $\Delta T$
    lossFlag = false; % mark current $\Delta T$ as -
                % - "non-loss $\Delta T$ "
    if $\bar{r}_i$ == $r_0$  % current sending rate $\bar{r}_i$ is equal-
        $\bar{r}_{i+1} = r_0$;  % - to maximum rate
    else  % current sending rate $\bar{r}_i$ is smaller than -
          % - maximum rate
        if lossFlag == true  % last $\Delta T$ is "loss $\Delta T$ "
            $\Delta = \Delta \div 2$; % halve $\Delta$ to have -
                % - conservative increase
            $\bar{r}_{i+1} = \bar{r}_i + \Delta$;
        else % last $\Delta T$ is "non-loss $\Delta T$ "
            $\Delta = \Delta \times 2$; % double $\Delta$ to have -
                % - aggressive increase
            if $\bar{r}_i + \Delta > r_0$  % keep sending rate for-
                $\bar{r}_{i+1} = r_0$;   % - next $\Delta T$ no more than -
            else              % - negotiated session rate
                $\bar{r}_{i+1} = \bar{r}_i + \Delta$;
            end
        end
    end
end
```

Fig.1. Logic of TACC algorithm

presented in Fig. 1. Conditions I_1 and I_2 are defined as: $I_1 : \sum_{k=1}^{N} r_k \geq 1$ and $I_2 : L > 0$ respectively (L denotes the number of optical packets dropped). I_1 reflects congestion, whereby the sum of all port rates is equal or greater than system capacity and I_2 implies that packets in the system exceed the maximum number of slots the FIFO feed-forward FDL system we are considering can simultaneously accommodate. When I_1 and I_2 are satisfied, we have a *loss control process*, and if either I_1 or I_2 are not satisfied, we have a *recovery process*. With either process, TACC operation during the first ΔT is different from that in subsequent intervals. For a *loss control process*, the initial rate starts from r_0 , i.e. the successfully negotiated rate for the current flow, and the loss rate is subtracted upon each new interval until no loss is notified, or until a minimum rate is reached. For a *recovery process*, the sending rate during the first interval is conservatively computed by adding half the last loss rate, in order to probe congestion conditions before fast recovery. If zero loss continues, the sending rate continues to increase exponentially. Otherwise, TACC switches to a *loss control process*. This increased sending rate may not exceed the current successfully negotiated rate, in order TACC and edge traffic smoothing are independent. The performance of this TCP aware congestion control is evaluated in the following simulations.

3. Simulation results

Simulation is carried out by the OPNET simulator [5], with a 14-node NSFNET backbone topology and a 16-node Pan-European network topology. "Baseline scenario" refers to the situation without TACC. Each node acts as both an edge switch and core switch depending on whether the incoming traffic is external or internal. Each node routes to each other destination node with equal probability. The core switches use a feed-forward FDL structure.

Fig. 2 shows the influence of edge switch buffer size on the overall throughput under different load conditions. A small edge buffer performs poorly under high load in baseline nets, indeed throughput decreases with offered load under some conditions. Also, a small edge buffer in TACC does not yield significantly better performance than a baseline network with large edge buffers, since a certain amount of edge buffering is required for TACC to operate correctly. However, with large edge buffers, TACC obviously outperform baseline nets, assuming equal edge buffers. This is because with small edge buffers, more packets are lost, due to buffer overflow during congestion. A certain amount of buffering is required for edge congestion control and TCP congestion control to react, however the edge switch size must not be too large but appropriate for correct operation.

Fig. 3 shows delay performance of both baseline and TACC nets. The delay is calculated as the sum of sojourn times of all generated packets in the network (including retransmitted packets), divided by the offered traffic. For baseline nets, delay decreases with load at low loads, and increases dramatically after a load of 0.4. This is because, at a low load of 0.1, delay is mostly from the packet containerization process, since under low traffic optical packets can not be efficiently containerized and are mostly triggered by timeout. As traffic increases from 0.2 to 0.4, there is sufficient traffic to avoid timeout during containerization, therefore, delay decreases slightly; after that, delay increases dramatically, since at higher load, contention and network congestion become dominant; the lost traffic has to be retransmitted. For TACC nets, at low load, the negotiation interacts with the filling of slots, therefore there is no decreasing trend; the delay in TACC nets steadily increases with load, due to the increased delay is from retransmitted traffic. By comparing these scenarios, it is apparent that at low and medium traffic load, baseline nets have better performance due to their "fill and go" nature; however, at medium and high load, TACC outperforms baseline due to its implementation of congestion control.

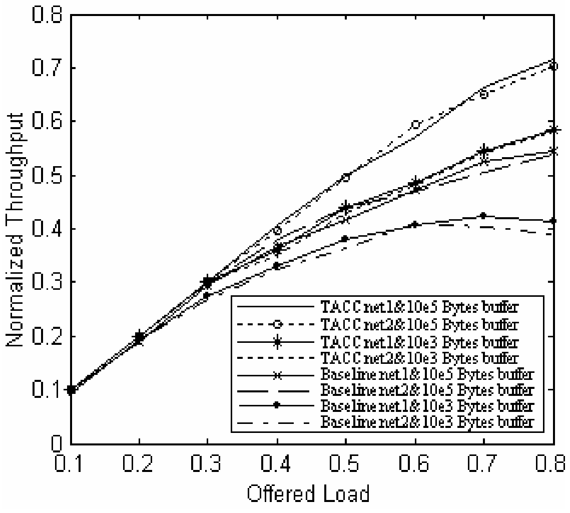


Fig.2. Throughput versus load and edge buffer

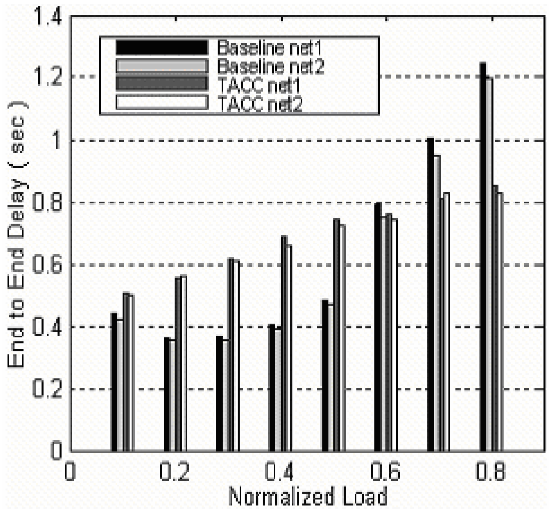


Fig.3. End to End Delay plot

4. Conclusion

In this paper, we proposed a congestion control algorithm that is implemented in the edge switch and interactively works with core loss notification to serve the two purposes: reducing the congestion and loss for core OPS at optical timescales and interactively assigning traffic to the edge buffers to wait for TCP to react if the network is saturated. Simulation results show conclusively that it improves throughput while increasing edge queuing size. However, this can be accommodated by readily available electronic buffers. This congestion control scheme pushes the processing burden from core to the edge, where powerful electronic processing is available.

5. References

[1] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in Optical Packet Switches", *IEEE JLT,* vol. 16, pp. 2081–2094, Dec. 1998.

[2] Fei Xue, S. J. B. Yoo, "TCP-aware congestion control in Optical Packet Switched Networks," *IEEE/OSA OFC'03*, Atlanta, March 2003.

[3] Z. Lu, D. K. Hunter, "Edge Traffic Smoothing in Optical Packet Switched Networks", *LCS*, London, September 2005.

[4] Z. Lu, D. K. Hunter, "Contention Resolution Scheme for Slotted Optical Packet Switched Networks", *ONDM 2005*, February 2005, Milan, Italy.

[5] www.opnet.com.