TCP over High Speed Variable Capacity Links: A Simulation Study for Bandwidth Allocation

Henrik Abrahamsson[1], Olof Hagsand[2], and Ian Marsh[1]

[1] SICS AB, Kista S-164 29, Sweden,
{henrik,ianm}@sics.se
[2] Dynarc AB, Kista S-164 32, Sweden,
hagsand@dynarc.se

Abstract. New optical network technologies provide opportunities for fast, controllable bandwidth management. These technologies can now explicitly provide resources to data paths, creating demand driven bandwidth reservation across networks where an applications bandwidth needs can be meet almost *exactly*. Dynamic synchronous Transfer Mode (DTM) is a gigabit network technology that provides channels with dynamically adjustable capacity. TCP is a reliable end-to-end transport protocol that adapts its rate to the available capacity. Both TCP and the DTM bandwidth can react to changes in the network load, creating a complex system with inter-dependent feedback mechanisms. The contribution of this work is an assessment of a bandwidth allocation scheme for TCP flows on variable capacity technologies. We have created a simulation environment using ns-2 and our results indicate that the allocation of bandwidth maximises TCP throughput for most flows, thus saving valuable capacity when compared to a scheme such as link over-provisioning. We highlight one situation where the allocation scheme might have some deficiencies against the static reservation of resources, and describe its causes. This type of situation warrants further investigation to understand how the algorithm can be modified to achieve performance similar to that of the fixed bandwidth case.

Keywords: TCP, DTM, rate control, rate adaption

1 Introduction

Reliable transfer of data across the Internet has become an important need. TCP [Pos81] is the predominant protocol for data transfer on the Internet as it offers a reliable end-to-end byte stream transport service. Emerging optical networking technologies provide fast, cheap and variable capacity bandwidth links to be setup in milliseconds allowing data-driven virtual circuits to be created when needed. One example of an application that could use such a service is the backup of critical data.

Exact allocation of bandwidth to TCP flows would alleviate complex traffic engineering problems such as provisioning and dimensioning. Allocating bandwidth to TCP is a complex problem; the TCP congestion control mechanism

G. Carle and M. Zitterbart (Eds.): PfHSN 2002, LNCS 2334, pp. 117–129, 2002.

plus network dynamics can make exact allocation for TCP data flows difficult. Te contribution of this paper is the performance evaluation of an estimation algorithm, which measures the rate of TCP flows and allocates capacity on a DTM network.

Dynamic Synchronous Transfer Mode [GHP92] [BHL+96] is a gigabit ring based networking technology that can dynamically adjust its bandwidth. DTM offers a channel abstraction, where a channel consists of a number of slots. The number of slots allocated to a channel determines its bandwidth. The slots can be allocated statically by pre-configured parameters, or dynamically adjusted to the needs of an application. In DTM it is possible to allocate a channel to a specific TCP connection, or to multiplex several TCP connections over the same channel. We mostly investigated cases where each TCP connection is assigned to a separate channel, but show one case in which two TCP connections compete for a single channel. The DTM link capacity is only allocated in the forward direction in this study, we have not performed any allocation for TCP acknowledgements.

TCP uses an end-to-end congestion control mechanism to find the optimal bandwidth at which to send data. In order to get good throughput with TCP operating over a technology such as DTM, it is important to understand the dynamic behaviour of the two schemes, especially when evaluating a bandwidth allocation strategy. TCP is capable of adjusting its *rate* whilst DTM is capable of changing its *capacity*. In dynamically interacting systems, it is possible to create unwanted oscillations resulting in under allocation or over allocation of bandwidth to TCP flows. In order to evaluate the performance of the DTM bandwidth allocator, we have implemented the algorithm in the network simulator ns-2. We have performed a number of simulations that include single and multiple TCP flows, links with varying delay characteristics, different buffer sizes, plus TCP Reno and Tahoe variants.

Section 2 outlines DTM and our estimation algorithm, simulation experiments are given in Section 3, related work follows in Section 4, and finally conclusions and a discussion are given in Section 5.

2 Dynamic Synchronous Transfer Mode

DTM uses a TDM scheme where time slots are divided into control and data slots. The control slots are statically allocated to a node and are used for signalling. Every node has at least one control slot allocated that corresponds to 512 kbps of signalling capacity. The data slots are used for data transmission and each slot is always owned by a node. A node is only allowed to send in slots that it owns. The ownership of the slots is controlled by a distributed algorithm, where the nodes can request slots from other nodes. The algorithms for slot distribution between the nodes affect the network performance. Each slot contains 64 bits and the slots are grouped in 125 microsecond long cycles. The bit rate is determined by the number of slots in a cycle, so one slot corresponds to a bit rate of 512 kbps. By allocating a different numbers of slots, the transmission rate for a channel can be changed in steps of 512 kbps.

2.1 TCP Rate Estimation and DTM Capacity Allocation

TCP's rate is simply estimated as the number of *incoming* bytes per second. The algorithm which is presented next calculates the rate by dividing the number of bytes by the time elapsed. The rate of each flow is calculated ten times per second, i.e. every 100 ms. This value has been chosen as a compromise between good measurement granularity and processing overhead. DTM technology however, has the ability to sample flows up to gigabit speeds, i.e. at sampling rates higher than 100 ms. Actual slot allocation or changes are done only *once* every second, this is slightly coarser due to the overhead of nodes potentially having to negotiate slots.

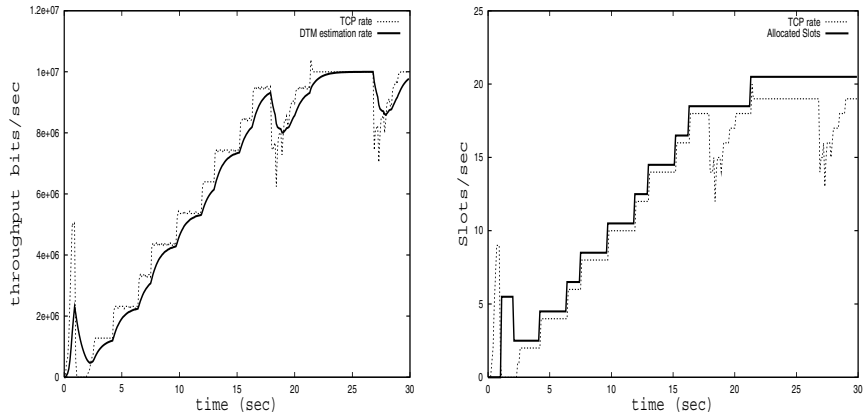
We now describe the TCP bandwidth estimator. Figure 1 shows the algorithm used to estimate the rate of a given flow. As stated, every 100 ms the estimator measures the rate `new` in bits per second and compares it with the previous value, `current`. A delta of the difference is reduced by `DTM_SHIFT` in the algorithm. Note this delta is simply shifted, keeping the complexity of the calculation to a minimum. In this case it is three, so the current value is changed by one eighth towards the recently measured flow value, as shown in the first half of the algorithm. This shift effectively determines how aggressively TCP's rate can be tracked. This default value has been chosen experimentally, as DTM is a deployed technology. The technical report version of this paper shows the affect

```
dtm_calc_bw ( new ) {
DTM_SHIFT = 3
MARGIN = 0.75
CORRIDOR = 2

/* first half - Move last estimate closer */
diff = new - current
if ( diff < 0 ) {
   diff = (-diff) ≫ DTM_SHIFT
   current = current - diff // Decreasing
} else {
   diff = diff ≫ DTM_SHIFT
   current = current + diff // Increasing
}
curr_slot = current / slot_bw

/* Second half - Last estimate within bounds ? */
if ( curr_slot > upper_bound ) || (curr_slot < lower_bound ) {
   dynBw = curr_slot + MARGIN + ( CORRIDOR / 2 )
   /* only change bw once per sec */
   change_link_bw (dynBw)
   }
}
```

Fig. 1. Algorithm for bandwidth estimation

of using other values [AM01].Finally the units are changed from bits per second to slots per second by dividing the rate by the channel bandwidth and assigning this value to the variable `curr_slot`.

The second half of the algorithm determines whether it is necessary to change the slot allocations. The current slot value is compared to upper and lower bounds before making any changes. An offset, 0.75 of a slot, `MARGIN` equivalent to 394 kbits, is added to the TCP throughput estimate so the DTM allocation will be a little over the estimated rate. Figure 2 shows two plots using the topology shown in Figure 3, the leftmost plot is the actual measured bandwidth of a single TCP flow. The right plot shows the effect of adding `MARGIN` and measuring the rate in slots. If the allocation was based purely on this estimate it would under allocate bandwidth, causing TCP reduce its window because of congestion on the link. The rightmost graph is coarser due to the second granularity of the bandwidth changes. The plots illustrate how the estimation can be used to give TCP the bandwidth it needs and hence maximise throughput. One can also see in this figure that estimation starts after 100 ms but a change is not applied to the offered bandwidth before the first second. Note also the y-axis in Figure 2b) is in slots per second and not bits per second as in the left figure. Additionally,



(a) Estimating the TCP Rate (b) DTM slot allocation

Fig. 2. Measured Throughput and Slot Allocation

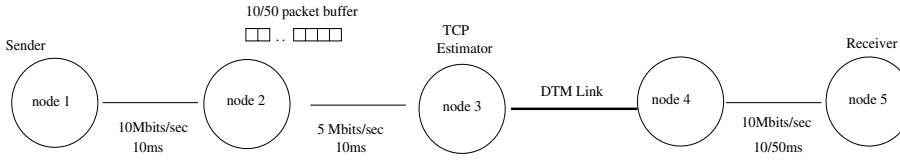


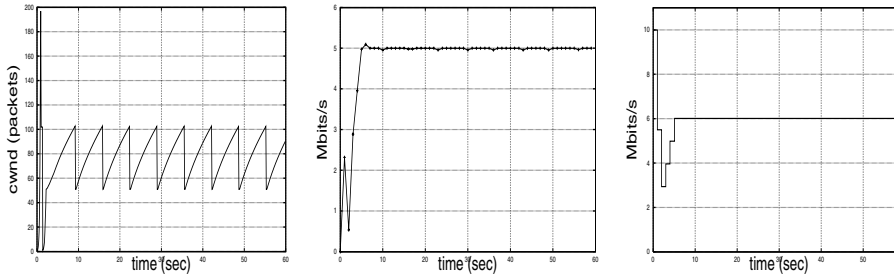
Fig. 3. Simulation topology

a `CORRIDOR` is an amount the estimate is allowed to vary before slots are added or decreased for a channel. This is not visible in the plots but will be illustrated later. The purpose is to avoid small fluctuations causing unnecessary costly slot allocation changes. As mentioned, slot changes can be time consuming due to the distributed nature of DTM [AMMS98].

3 Simulation Tests

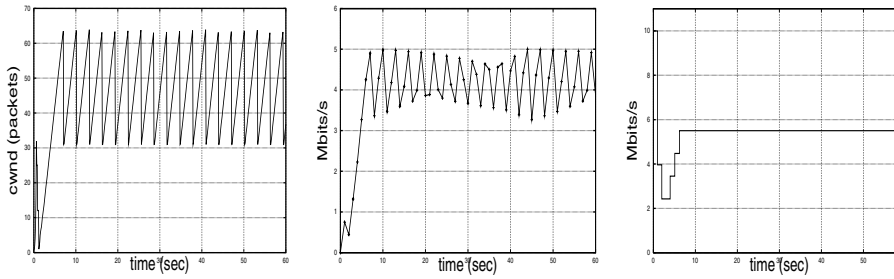
This section presents simulation results that show how the DTM estimation algorithm adapts the offered bandwidth to TCP flows. Figure 3 shows the topology we used for the following simulations. The 5 Mbits per second link between nodes two and three is the bottleneck link. The link between nodes three and four is the DTM link with dynamically allocated bandwidth. Initially the DTM link is set to 10 Mbits per second. This value was chosen simply for convenience, since simulating a 622 Mbits per second link with large bandwidth flows is not feasible in a packet level simulator like ns-2. The other two links also have a capacity of 10 Mbits per second. A bulk transfer TCP Reno flow was setup between nodes one and five and the throughput measured at node three, in order to allocate bandwidth on the outgoing DTM link. In this first simulation the queue length in node 2 was set to 50 packets, figure 4 shows the result. In congestion avoidance the TCP flow increases the congestion window by the maximum segment size bytes each RTT seconds. However, the increase is not made each RTT. Instead TCP will increase $MSS/congestion$ window bytes each time an ACK is received. This means that after RTT seconds, the congestion window was increased by MSS bytes. This continues until the TCP flow has filled the buffer space at the bottleneck link, resulting in a packet drop. TCP Reno, using fast retransmit and fast recovery, then reduces the congestion window by half and continues with congestion avoidance. The congestion window, therefore, follows a sawtooth curve. If enough buffer space is available at the bottleneck link, the rate of the TCP flow, perceived after the second link, is not affected when the congestion window is reduced. This mechanism and result can be seen in left and middle plots of Figure 4. The rightmost plot shows the dynamically allocated bandwidth on the DTM link. It can be seen that TCP actually manages to get about one Megabit per second more on the DTM link due to the extra capacity allocated to the flow through the addition of `MARGIN`. It should be stated in a real deployment of TCP over DTM that this value is settable by network operators. Its affect can be tested in simulation environments such as this if necessary.

Figure 5 shows the results when the queue size at the bottleneck link is limited to ten packets. This could be the case if a static allocation over the DTM network has been setup. Now the rate of the TCP flow changes with the congestion window, but the changes are too small to affect the dynamic allocation of bandwidth. This is due to the corridor mentioned earlier to avoid small changes from incurring changes in the slot allocation scheme. Figure 6 shows the case in which the simulation with a small queue size and a 50 ms link delay has been repeated using TCP Tahoe instead of TCP Reno. TCP Tahoe



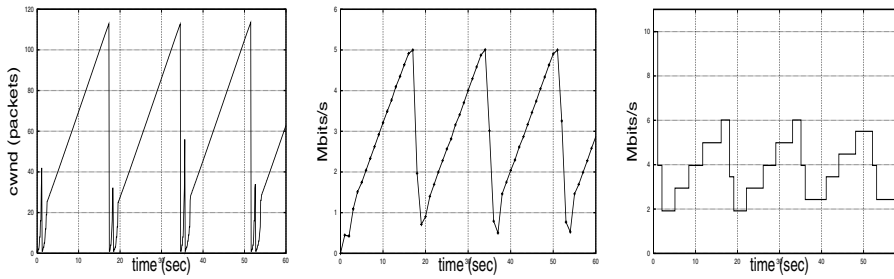
(a) Congestion Window (b) Throughput (c) DTM allocation

Fig. 4. Dynamically allocated bandwidth on a DTM link (50 packet queue)



(a) Congestion Window (b) Throughput (c) DTM allocation

Fig. 5. Dynamically allocated bandwidth on a DTM link (10 packet queue)



(a) Congestion Window (b) Throughput (c) DTM allocation

Fig. 6. Tahoe TCP on a DTM link (10 packet queue)

only relies on the retransmission timer and does not use fast retransmit. When a packet is dropped, the congestion window is set to one and slow-start is invoked. We can see that the allocation on the DTM link closely follows the sharp saw tooth behaviour of TCP Tahoe Figure 6c).

3.1 Two Flows per Channel and Small Router Buffers

So far, we have shown cases where the dynamic allocation of bandwidth has allowed TCP to maximise its throughput. We illustrate one case next when the algorithm has weaknesses to allocate sufficient bandwidth to two TCP Reno flows. In this scenario the fixed link case performs better. Figure 7 shows the topology that we used. It differs from previous simulations in that the flows have their own input buffer at node two but share a common output buffer in the same node. This buffer is also served ten times faster than in previous cases by the fact that the link feeding the DTM network was set to 100 Mbits per second. In this case the queue length of a DTM link, node three, was limited to ten packets.

Figure 8 shows the results when the link capacity between nodes 3 and 4 was fixed at 10 Mbits per second. We can see that both flows manage to reach their 5 Mbits throughput, effectively sharing equally one DTM channel. If we now turn our attention to the same simulation but replace the static link between nodes three and four with a variable one the results are quite different. Figure 9 shows the dynamically allocated bandwidth on the DTM link. Neither of the flows manage to reach 5 Mbits per second on their output links. In this case packets are being dropped in the output buffer of node three. This can be seen in the congestion windows of the two flows, they never manage to maintain the size of the static case, about 100 segments. The problem in this case is the estimation algorithm should not *decrease* the estimation *if* packets are being dropped. The algorithm is symmetric, it increases or decreases depending on the measured rate. Additionally the effect of the short queue does not help, there is not sufficient pressure with a small queue to keep the rate up, with a larger buffer there is more pressure due to accumulated packets. Interestingly, the algorithm actually

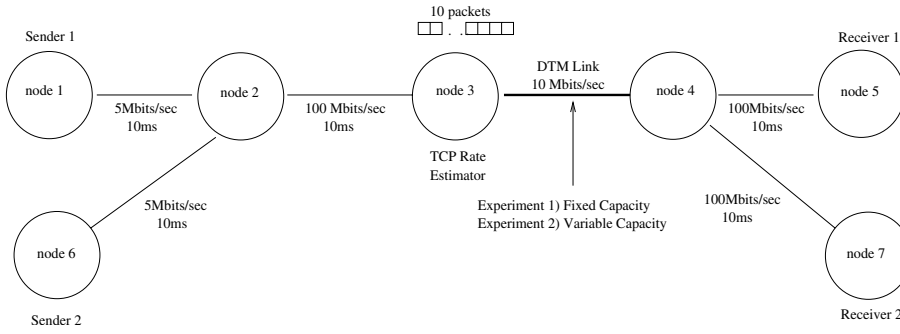
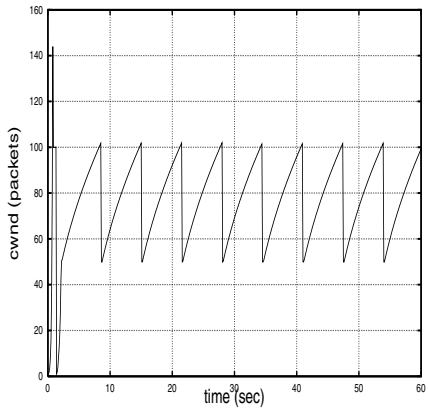
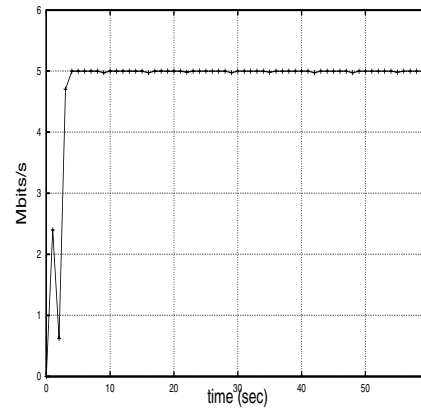


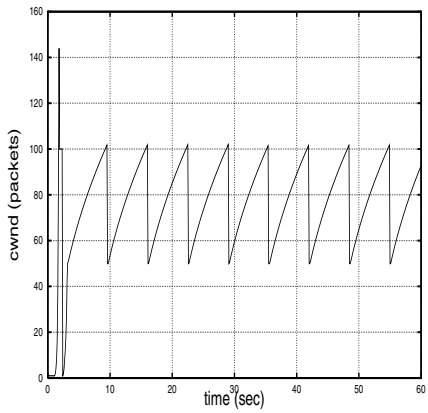
Fig. 7. Simulation topology with two flows



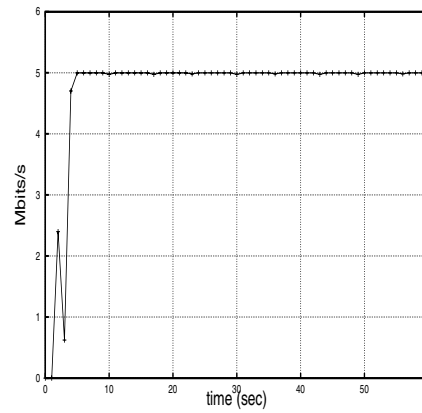
(a) Window Flow 1



(b) Rate Flow 1



(c) Window Flow 2

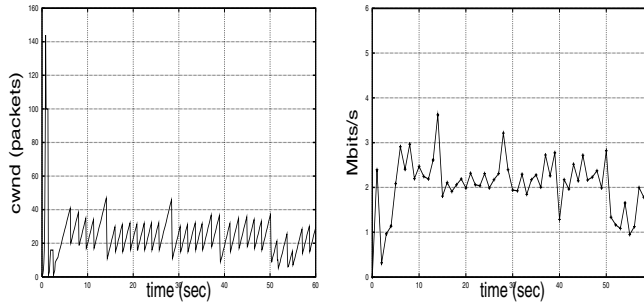


(d) Rate Flow 2

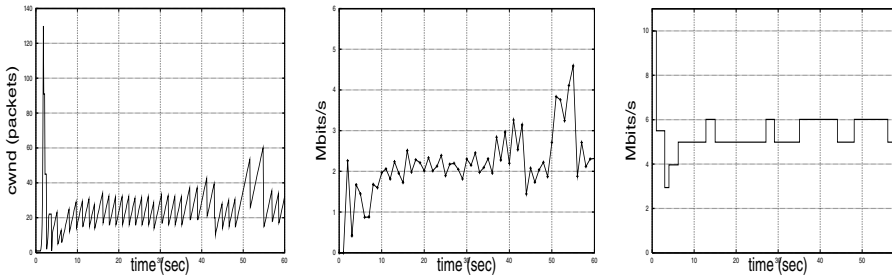
Fig. 8. Experiment 1 static link: The senders do not drop packets at the ingress node and achieve their constrained link throughput 5Mbits per second.

correctly allocates for the observed throughput, however does not maximise the TCP throughput.

Some researchers have put forward TCP variants which are capable of estimating the bandwidth such as TCP Westwood [MG98], which do not solely rely on packet loss for congestion. It is not clear whether TCP variants such as this will improve on the situation above without substantial simulations. However other TCP variants would be worthy of investigation. The important point is note is it is important to detect loss early and this can be done by monitoring



(a) Window Flow 1 (b) Rate Flow 1



(c) Window Flow 2 (d) Rate Flow 2 (e) Aggregated DTM rate

Fig. 9. Experiment 2 Dynamic Link: Allocated bandwidth on DTM link. Congestion window and measured rate for two TCP Reno flows with limited queue size at the DTM link.

queues for the local node or even via mechanisms such as RED or ECN for upstream nodes.

Our conclusion is that in the face of loss at a node the estimation algorithm should not decrease to allow TCP to recover. So a special case for loss could be introduced, *during loss* the rate could be estimated but no action is taken to adjust the bandwidth. One other solution would be to adjust the rate more slowly when allocating *less* bandwidth. This is trivial in the present scheme, the `DTM_SHIFT` variable can be split into `DTM_SHIFT_INC` and `DTM_SHIFT_DEC` where a decrease in bandwidth takes places at a slower rate. This might have adverse affects on a normal behaving system so, once again would need to be validated with further simulations. We note that [Kri01] also performed comparisons with the fixed link case and show in one experiment that the dynamically allocated link was not able to achieve the throughput of a fixed link case. With a certain selection of parameters it was only possible to allocate slightly over half of the

fixed link case. We conclude that it is not always possible to perform as the fixed link case, however comparisons, wherever possible should be performed. The savings of allocating just the required bandwidth, however can be considerable. This was the methodology employed during our investigation.

4 Related Work

Work on estimating and maximising TCP throughput for variable capacity links is relatively scarce. However comprehensive studies have been done related to the performance of TCP on ATM networks [Bon98] [MG95] [CLN96]. The main conclusions of the works are similar, the traffic classes of ATM are poorly suited to the bursty needs of TCP, due to the traffic contracts needed by ATM classes. The conclusion of [Bon98] is that the complexity of choosing traffic parameters for ABR is not in proportion to the benefits of carrying TCP/IP traffic. The CBR class is too simple for TCP, as only the peak rate is specified. Most of the DTM research in this area focuses on the distributed slot allocation for example [AMMS98].

Clark and Fang propose a framework for allocating bandwidth to different users during congestion [CF98]. The focus of the work is TCP bulk-data transfers. The authors attempt to keep TCP flows in congestion avoidance in the best case, and fast recovery phase in the worst case, by avoiding dropping several packets of the same flow in the same RTT. The conclusions of the given work are similar to those of [Bon98], that TCP connections can have difficulties to fill their alloted bandwidth. The work resembles ours in that they attempt to allocate bandwidth between different flows in a fair manner. It differs from ours in that we assume that the network can change its offered bandwidth and we focus on maximising TCP throughput, rather than trying to maintain a TCP state in the face of adverse network conditions. In addition, we allocate bandwidth to flows not only when the network is congested but also in normal situations as well.

Sterbenz and Krishnan investigate TCP over Load-Reactive Links in a ICNP publication [KS01] and a technical memorandum [Kri01]. They use a hysteresis control mechanism for capacity allocation. Buffer levels are monitored (as in [Lun98]) and if the occupancy is greater than a threshold the capacity is increased and vice versa. This approach is not the same as ours, we measure the rate of incoming TCP flows at the router before the DTM link rather than the buffer level in the router at the outgoing DTM link. Their scheme is dependent on keeping buffers occupied all the time, otherwise the link capacity will fall and hence the throughput. A single TCP flow is simulated and the authors state that the control parameters should be carefully chosen. Poor parameter choice can have the opposite effect, resulting in TCP not being able to operate, as stated previously. The work resembles ours in that a method is presented to react to network load and allocate bandwidth for TCP accordingly. It differs in that we measure the throughput of individual flows and allocate bandwidth from this measurements, where they use the buffer length as a measure of the load. Our

system is less scalable, but more accurate, as we can ascertain exactly the bandwidth of the incoming TCP connections. Also there is no need to keep buffers occupied to allocate bandwidth for TCP connections. Also there is no potential interaction between several dynamic allocation schemes running on the same node.

Lundqvist evaluates different algorithms for bandwidth allocation for DTM channels transporting IP traffic [Lun98]. The algorithms were assessed with respect to throughput, delay and bandwidth changes per second. TCP rate adjustment is done by placing the incoming packets into a buffer and adding and removing slots if the level of the buffer exceeds continuously maintained threshold values. He concludes that adaptive strategies are recommended for TCP, however too frequent changes can be undesirable in a DTM network due to the processing cost. The main conclusion from this work is that the choice of algorithm can play a significant role in the performance. This work is similar to ours in that the goal is a slot allocation for TCP traffic over DTM. We also agree it is important to keep the computational complexity low and DTM bandwidth changes as infrequent as possible. It differs from ours in that we measure the rate of each TCP flow, whilst he looks at the outgoing buffer length as a sign to increase or decrease the number of slots. We look more into network scenarios such as different link delays, buffer lengths and use two different TCP types, TCP Reno and Tahoe.

5 Conclusions

We have analysed a complex problem, allocating bandwidth to a protocol that can adapt its rate. The benefits of guaranteeing throughput for an application using TCP can be very beneficial, in particular the cost savings when paying per unit of transmission. The goal was to investigate the behaviour of our bandwidth estimation scheme, its affect on TCP *and* on a network that can vary its capacity, in this case DTM. Our work however is not only limited to DTM technology, we can draw the same conclusions about TCP performance on any high speed network technology that offers variable capacity.

We have written a simulation environment using ns-2, and found that in almost all cases, TCP could be allocated a share of the channel identical to its measured throughput on a fixed network. We identified one scenario in which the algorithm could be improved, when packets are dropped at a router with a small buffer. In this situation the estimation algorithm should not reduce the offered bandwidth further, resulting in less offered bandwidth and further packet loss. Instead it should allow TCP to find the new capacity available in the network. The combination of the small buffer size plus high speed input link aggravates this observed deficiency.

There are some other open issues, the system as described relies on measuring individual TCP flows, therefore methods that encapsulation such as MPLS would hinder us from measuring single flows. An alternative is monitor and measure aggregate flows, however this was not the focus of our work. DTM is a fast MAN

technology and can monitor flows at gigabit speeds, however if a bandwidth allocation scheme would be used in a non-DTM environment, especially in a backbone, some consideration would be needed for the sampling and measuring rate one can achieve with thousands of TCP flows. Aggregation of flows in this context would be a viable alternative.

We have only considered bulk data transfers, as the scheme measures flow bandwidths, it is not feasible to allocate bandwidth for all TCP flows, particularly http transfers as most data is transferred during the slow-start phase of a TCP connection, e.g. banners, buttons etc. Another issue is capacity provisioning for ACKs, we assumed the return path for acknowledgements is not constrained. In our experiments we had a return channel of 512 kbits per second which was more than adequate to support the forward data rates we were using, a maximum of 100Mbits per second. Further investigation is needed to state where problems could arise as well potential solutions.

We have not considered well known sceneries such as satellite links with large bandwidth-delay products or more interestingly, where the control loops are sensitive to delay. We believe some benefit would be gained by looking at this problem (and others with time sensitive mechanisms) from a control theory perspective rather than the traditional networking approach, Westwood referenced earlier, takes exactly this approach.

In a simulation environment the parameter space is large. Due to space limitations we have only discussed a key subset of possible buffer sizes, link bandwidths, link delays and TCP variants. Parameters that are worthy of further investigation include sampling times and estimation thresholds. Further results, plus validation tests for using ns-2 in these kind of simulations, can be found in the technical report [AM01].

Acknowledgements

We would sincerely like to acknowledge the Computer and Network Architecture lab at SICS, and the Laboratory of Communication Networks at KTH, Royal Institute of Technology, Stockholm.

References

[AM01] Henrik Abrahamsson and Ian Marsh. Dtmsim – dtm channel simulation in ns. Technical Report T2001:10, SICS – Swedish Institute of Computer Science, November 2001.

[AMMS98] Csaba Antal, József Molnár, Sándor Molnár, and Gabor Szabó. Performance study of distributed channel allocation techniques for a fast circuit switched network. *Computer Communications*, 21(17):1597–1609, November 1998.

[BHL+96] Christer Bohm, Markus Hidell, Per Lindgren, Lars Ramfelt, and Peter Sjödin. Fast circuit switching for the next generation of high performance networks. *IEEE Journal on Selected Areas in Communications*, 14(2):298–305, February 1996.

[Bon98] O. Bonaventure. *Integration of ATM under TCP/IP to provide services with minimum guaranteed bandwidth.* PhD thesis, University of Liege, 1998.

[CF98] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.

[CLN96] E. Chan, V. Lee, and J. Ng. the performance of bandwidth allocation strategies for interconnecting atm and connectionless networks, 1996.

[GHP92] L. Gauffin, L. H. Hakansson, and B. Pehrson. Multi-gigabit networking based on DTM. *Computer Networks and ISDN Systems*, 24(2):119–130, April 1992.

[Kri01] Rajesh Krishnan. Tcp over load-reactive links. Technical Memorandum 1281, BBN, February 2001.

[KS01] Rajesh Krishnan and James Sterbenz. Tcp over load-reactive links. In *International Conference on Network Protocols (ICNP)*, 2001.

[Lun98] Henrik Lundqvist. Performance evaluation for IP over DTM. Master's thesis, Linköping University, Linköping, 1998.

[MG95] Kjersti Moldeklev and Per Gunningberg. How a large ATM MTU causes deadlocks in TCP data transfers. *IEEE/ACM Transactions on Networking*, 3(4):409–422, 1995.

[MG98] Saverio Mascolo and Mario Gerla. TCP congestion avoidance using explicit buffer notification. Technical report, University of California, Los Angeles, California, February 1998.

[Pos81] J. Postel. Transmission control protocol. Request for Comments 793, Internet Engineering Task Force, September 1981.