Algorithm for traffic grooming of batches of deadline-driven requests

Juliana de Santi, Nelson L. S. da Fonseca and Gustavo B. Figueiredo*
University of Campinas, Brazil
Email: {santi, nfonseca}@ic.unicamp.br
*Federal University of Bahia, Brazil
Email: {gustavo}@dcc.ufba.br

Abstract—This paper introduces a novel traffic grooming algorithm for the connection establishment of deadline-driven requests. The algorithm grooms batches of requests rather than individual requests. Results indicate that the algorithm is able to significantly reduce the blocking ratio and promote a fair distribution of blocking among source destination pairs of nodes.

I. INTRODUCTION

The large availability of bandwidth in Wavelength Division Multiplexing (WDM) optical networks made possible the existence of bandwidth hunger applications such as those in IPTV services and e-Science. These applications usually have QoS requirements established in their Service Level Agreements (SLAs). A common requirement is the deadline to finish the transmission of the application data, which is quite common in e-Science [1] and grid applications [2]. Applications which have such requirements are called deadline-driven.

Requests for connection establishment by deadline-driven applications can be postponed until the time at which the applications data should be transmitted at the maximum available rate to meet the required deadline. The possibility of postponing the connection establishment can benefit both users and service providers. If bandwidth is unavailable, the connection establishment can be postponed avoiding the blocking of requests. Moreover, service providers can choose the rate of a connection which is determined by the amount of data to be transmitted and the connection deadline.

A batch is a set of requests which arrived in a period starting at the arrival time of the oldest request and ending at the earliest time one of the request should start transmission in order to meet its deadline. The flexibility of setting the starting time of a connection allows the creation of batches of requests that can be scheduled as a whole rather than as a sequence of individual requests. At the scheduling time of a batch, each connection will demand a specific transmission rate which can be groomed on a set of existing or potentially allocable lightpaths. The scheduling of batches allows greater combinations of connection in the grooming decision. To illustrate the benefit of batch scheduling, suppose that 11Gbps and 22Gbps are the available capacity in different lightpaths. If an arriving request for connection demanding 5Gbps is groomed in a lightpath with 22Gbps of available capacity, a subsequent request demanding 20Gbps will be blocked. This blocking could be avoided if the connections were scheduled at once by postponing the scheduling of the request which arrived first.

This paper introduces a new traffic grooming algorithm which grooms batch of requests rather than individual requests. The problem of grooming a batch of requests is a variation of the classical job shop scheduling problem in which connections are represented by jobs and lightpaths by machines. In this case, machines have different capacity and can be occupied by concurrent jobs. As far as we known, there is no existing algorithm for the scheduling of deadline driven requests that schedules batch of requests for connection establishment in WDM networks.

Results derived via simulation show the efficacy of the proposed mechanism to reduce the blocking of connection as well as to promote fair blocking among different source destination pairs.

This paper is organized as follows. Section II overviews related work. Section III introduces a novel dynamic traffic grooming algorithm. Section IV presents a numerical evaluation of the proposed algorithm. Finally, Section V concludes the paper.

II. RELATED WORK

The survey of traffic models for scheduled service in [3] introduces a classification of models based on the information available about the connections.

In [4], the Holding-Time-Aware (HTA) dynamic traffic grooming algorithm was introduced. It aggregates connections with known duration to lightpaths with known lifetime. An algorithm aware of the holding time of the connections with load balancing was presented in [5] and an extension that uses multipath routing for the provision of high capacity connections was introduced in [6].

The work in [7] schedules a set of connection requests by representing the allocation times as an interval graph which is analyzed to detect scheduling conflicts.

The study in [8] considers the provisioning of connections with either fixed rates or adaptive rates. Requests are processed one by one and the aim is the fast release of lightpaths for their reuse by incoming connection requests.

Cavdar et al. [9] defines delay tolerance as the maximum period duration that a request can wait to establish a connection. If there is no available capacity to serve the connection request, it can be postponed (redialed) as many times as...
allowed by its delay tolerance. Unlike the approach in [8], the transmission rate is fixed as well as the holding time. The employment of delay tolerance in scheduling criteria to differentiate service classes in WDM networks was investigated in [10]. The quality of service differentiation was studied based on information about connection setup time [11].

Time flexibility is employed in [12] to implement advanced reservation scheme for delivering quick responses to users. Re-provisioning of requests is used to improve performance. An algorithm for advanced reservation for elastic applications is defined in [13].

Previous work introduced solutions for two type of scenarios. In the first, connections with time requirements are served individually [4]–[6], [8]–[13] and in the second, groups of static requests are served [7], [14].

The algorithm in the present paper grooms connections which requests arrive dynamically and have deadline requirements. Differently than previous work, the proposed algorithm explores the flexibility resulted from different deadline values to create and schedule batches, rather than to schedule individual requests.

III. BATCH GROOMING ALGORITHM

In the scenario considered in this paper, requests for connection establishment arrive dynamically. Each request specifies the amount of bytes to be transmitted as well as the deadline to have the transmission completed.

Connections are not provisioned upon arrival, and a deadline for scheduling is determined by the available capacity in the lightpaths and the amount of bytes to be transmitted. The required transmission rate increases as time passes. If the connection is not scheduled before or at the scheduling deadline, its establishment becomes unfeasible since the required transmission rate becomes greater than the available capacity in any lightpath from source to destination nodes.

The proposed algorithm, called BatchGrooming, takes advantage of the fact that the provisioning of a connection can be delayed in order to create a batch of requests with the same source and destination nodes. This batch will have a maximum scheduling time (scheduling deadline) which is determined by the earliest time after which one of the requests in the batch becomes unfeasible.

By provisioning a batch of requests rather than provisioning a sequence of individual requests, it is expected that different combinations of connections and lightpaths will decrease the number of requests blocked. The BatchGrooming algorithm decides on how to groom these connections on the set of lightpaths.

BatchGrooming tries to groom the highest number of requests in a batch by considering the demanded rate of each connection as well as the available capacity in each lightpath. Before presenting the BatchGrooming algorithm, let us introduce some notation:

- \( s_j \): source node of the connection requested by the request \( j \);
- \( d_j \): destination node of the connection requested by the request \( j \);
- \( N_j \): number of bytes requested to be transmitted by the request \( j \);
- \( D_j \): deadline at which the connection (transmission) requested by the request \( j \) should end;
- \( R_j = (s_j, d_j, N_j, D_j) \): the \( j \)th request;
- \( r_j(t) = \frac{N_j}{D_j - t} \): transmission rate of the connection \( j \) which starts at the time \( t \);
- \( l_{i,j}(t) \): lightpath \( i \) connecting \( s_j \) to \( d_j \) at time \( t \);
- \( c_{i,j}(t) \): available capacity of the lightpath \( l_{i,j} \) at time \( t \);
- \( L_j(t) = \{l_{i,j}\} \): set of lightpaths connecting \( s_j \) to \( d_j \) at time \( t \);
- \( B_j(t) = \{R_k | s_k = s_j \text{ and } d_k = d_j\} \): set of requests at time \( t \) that have the same source and destination nodes of request \( R_j \);
- \( SB_j(t) \): scheduling deadline of the batch \( B_j(t) \) which is given by the earliest starting deadline of a connection in the batch.

The BatchGrooming algorithm is defined in Figure 1.

Algorithm BatchGrooming

```
Input: \( R_j \) - arriving request at time \( t \).
Output: The request \( R_j \) provisioned, blocked or provisioning postponed.
1: \( B_j(t) = \{R_j\} \)
2: if \( \exists R_k | s_k = s_j \text{ and } d_k = d_j \) then
   3: \( B_j(t) = B_k(t) \cup \{R_j\} \)
4: end if
5: At \( SB_j(t) \)
6: Groom-Solver(\( L_j(t) \), \( B_j(t) \))
7: \( B_k(t) = \emptyset \)
8: for \( m = 1...|B_j(t)| \text{ and } i = 1...|L_j(t)| \) do
   9: if \( X_{m,i} = 1 \) then
      10: Establish connection \( m \) on lightpath \( i \)
   11: else
      12: \( B_k(t) = B_k(t) \cup \{R_m\} \)
   13: end if
8: end for
```

Fig. 1. Algorithm BatchGrooming

Algorithm Groom-Solver

```
Input: \( L_j(t) \) and \( B_j(t) \)
Output: \( X = \{X_{m,i}\} \)
1: Solve \( \max \sum X_{m,i} \times r_m(t) \)
2: Subject to \( X_{m,i} \times r_m(t) \leq W_{i,j}(t) \)
3: \( X_{m,i} \in \{0, 1\} \)
4: \( m = 1...|B_j(t)| \)
5: \( i = 1...|L_j(t)| \)
```

Fig. 2. Algorithm Groom-Solver

The BatchGrooming algorithm is solved at the arrival of request \( R_j \). If there is no other existing request with the same source destination (SD) pair, a batch is formed with \( R_j \) (Line
1). Otherwise a batch is formed with existing requests (Line 2 and Line 3). The batch is then scheduled to be groomed at the scheduling deadline of the batch just formed (Line 5) which is the earliest scheduling deadline of a connection in the batch.

The Groom-Solver decides which connection should be groomed on which lightpath (Line 6). Connections are established according to the suggestion given by the Groom-Solver (Line 10) and the requests which received no suggestion for grooming are gathered in a batch (Line 12) for later scheduling.

The Groom-Solver Algorithm is presented in Figure 2. It solves an integer linear programming problem that tries to allocate the maximum number of connections by maximizing the capacity allocated (Line 1). Such allocation is bounded by the available capacity of a lightpath associated to the connection. All compared algorithms use the RWA algorithm presented in [5].

The WDMSim [15] simulator was used in the evaluation. The independent replication method was employed to generate confidence intervals with 95% confidence level. Each simulation run involved 100,000 requests. The ILP problem in Groom-Solver algorithm was solved using the Xpress-MP Suite tool [16].

The NSF topology, with 16 nodes and 25 bidirectional links (Fig. 3) and the USA topology, with 24 nodes and 43 bidirectional links (Fig. 4) were used in the simulation. For both topologies, it was assumed that each fiber has 16 wavelengths, with capacity of an OC-192 carrier (10 Gbps). Each node is a multi-hop partial grooming node with 32 grooming port pairs and no wavelength conversion capability.

IV. PERFORMANCE EVALUATION

The effectiveness of the BatchGrooming algorithm is compared with those of the algorithms MaxRate and MinRate [8], which groom connections one at a time by employing the minimum rate and the maximum rate, respectively. The minimum rate associated to a connection at its arrival time is $\text{minRate}_j = \frac{N_j}{D_j - t}$ and the maximum rate is the largest available capacity of a lightpath associated to the connection. All compared algorithms use the RWA algorithm presented in [5].

The WDMSim [15] simulator was used in the evaluation. The independent replication method was employed to generate confidence intervals with 95% confidence level. Each simulation run involved 100,000 requests. The ILP problem in Groom-Solver algorithm was solved using the Xpress-MP Suite tool [16].

The NSF topology, with 16 nodes and 25 bidirectional links (Fig. 3) and the USA topology, with 24 nodes and 43 bidirectional links (Fig. 4) were used in the simulation. For both topologies, it was assumed that each fiber has 16 wavelengths, with capacity of an OC-192 carrier (10 Gbps). Each node is a multi-hop partial grooming node with 32 grooming port pairs and no wavelength conversion capability.

The minimum rate ($\text{minRate}$) of the connections are distributed according to the following probability distribution: 50/100 for OC-3; 25/100 OC-12; 15/100 for OC-24; 7/100 for OC-48; 2/100 for OC-96 and 1/100 for OC-192. Connection requests are uniformly distributed among all pairs of nodes. The amount of bytes to transmit (File size) are uniformly distributed in the range [0.1-2.5]GB for $\text{minRate}$ OC-3, [0.5-5]GB for $\text{minRate}$ OC-12; [1-10]GB for $\text{minRate}$ OC-24; [1-50]GB for $\text{minRate}$ OC-48, OC-96 and OC-192. The deadline for finishing the transmission of the requested file is determined by the $\text{minRate}$ and the file size. We preferred to draw $\text{minRate}$ values rather than drawing deadline values to involves realistic values of the transmission rates. Requests are still represented by file size and deadline values and the deadline values are given by the ratio File size/$\text{minRate}$.

The metrics considered in the evaluation of the simulations were request blocking ratio and the fraction of unprovisioned bytes, i.e the fraction of non-provisioned bytes to the total bytes requested to be transmitted. In order to evaluate the fairness promoted by the proposed algorithm, the distribution of blocked requests among the source-destination pairs was also evaluated.

Figure 5 presents the blocking ratio values for the NSF topology as a function of the connections arrival rate. Under arrival rate greater than 55 requests per second ($rps$), BatchGrooming algorithm generates blocking ratio lower than that produced by the other algorithms. When comparing BatchGrooming algorithm with the MaxRate algorithm, it is clear that the processing of batch of requests significantly decreases the number of blocked requests. Under arrival rates of 55 $rps$ and 80 $rps$, the BatchGrooming algorithm produces, respectively, blocking ratio 11% and 69% lower than those given by the MaxRate algorithm. Such differences are lower when comparing BatchGrooming and the MinRate algorithm. Moreover, under arrival rate of 120 $rps$, BatchGrooming generates blocking ratio 4% lower than that produced by MinRate and under arrival rate of 90 $rps$, the difference increases to 17%. These results show the benefit of grooming batches of connections.

Figure 6 shows the blocking ratio as a function of the connection arrival rate for the USA topology. Results are similar to those obtained for the NSF topology. Under arrival rate of 65 $rps$, the blocking ratio given by BatchGrooming is lower than that produced by the other algorithms. The values of blocking ratio given by BatchGrooming are 73% lower than that generated by the MaxRate algorithm under arrival rate of 100 $rps$. Under an arrival rate of 120 $rps$, the MinRate algorithm produced blocking ratio 31% greater than that generated by the BatchGrooming algorithm. Moreover,
under arrival rate of 95 rps, this difference is 23% greater than that generated by the BatchGrooming algorithm.

To evaluate the fairness of the results produced by the algorithms, we verified the per S-D pair distribution of blocked requests for a single simulation with an arrival rate of 100 rps for the NSF topology (Figure 7). For some pairs, the MaxRate algorithm produced blocked requests up to 1.6 greater than its mean and up to 4.6 times greater than the mean value (12.9%) given by BatchGrooming. Since both algorithms use the same RWA algorithm, this evinces that grooming of batches of connections is a fair way to balance the establishment of connections among the S-D pairs. When compared to the results given by the MinRate algorithm, the distribution is similar. However, the BatchGrooming is slightly more efficient, since MinRate presents a larger fluctuation around the mean value. Furthermore, the MinRate algorithm generated blocking of 1.3 times greater than its mean blocking value (15.2%) and up to 1.6 times greater than the mean value (12.9%) given by BatchGrooming.

Figure 8 presents the ratio of unprovisioned bytes for the NSF topology as a function of the connection arrival rate. Under arrival rate of 120 rps, the BatchGrooming algorithm decreases the ratio of unprovisioned bytes to a value 4% lower than the values produced by the MinRate algorithm. Such difference rises to 25% under arrival rate of 50 rps. Moreover, the values generated by BatchGrooming algorithm are 23% greater than that produced by the MaxRate algorithm under arrival rate of 120 rps. Under arrival rate of 50 rps the ratio of unprovisioned bytes produced by BatchGrooming algorithm is 89% greater than that given by the MaxRate algorithm. This is a consequence of the size of files blocked. As shown in Figure 9, the MaxRate algorithm tends to block a large number of connections with small files. On the other hand, the MinRate and BatchGrooming algorithms block a greater number of connections with large files.

Connections requesting the transmission of large files demand high transmission rates to ensure that transmissions meet the requested deadline. The employment of the MinRate algorithm implies on the establishment of connections that last long periods which favors the provisioning of simultaneous large number of connections with low transmission rate. However, keeping the lightpaths highly utilized by a long time can unfavors connections requiring the transmission of large files. The BatchGrooming algorithm is sensitive to the transmission of large files since any delay in establishing a connection increases the required transmission rate.

V. CONCLUSION

This paper proposed a novel algorithm that grooms batches of connections requests in WDM mesh networks. The main contribution of this paper is the employment of batches for dynamic traffic grooming. The use of batches takes advantage of the flexibility resulted from having different requested deadlines. The BatchGrooming algorithm produces lower and fairer
fraction of blocking when compared to algorithms that groom request by request. As future work, we plan to consider the network future state, which can be inferred by the connections departure times. Another potential work involves multipath routing.

ACKNOWLEDGMENTS

This work was partially sponsored by CAPES and by CNPq under support code 141246/2009-0.

REFERENCES


