On the Capacity Limits of Advanced Channel Reservation Architectures

Niloofar Fazlollahi, Reuven Cohen and David Starobinski
Dept. of Electrical and Computer Engineering
Boston University, Boston, MA 02215
Email: {nfazl,cohenr,staro}@bu.edu

I. INTRODUCTION AND MOTIVATION

The next generation of grid applications demand fast and reliable transfers of extremely large volumes of data between distributed sites around the world. For example, the U.S. Department of Energy's Genomes to Life project relies on ultra high throughput connections between research laboratories and supercomputers for processing and analyzing massive amounts of data.

Despite the seemingly excessive capacity of network backbones, experiments have shown that current TCP/IP architectures do not match the needs of high throughput applications. TCP congestion control policies and the packet switching nature of IP are among of the major causes of this discrepancy. Thus, significant efforts have recently been devoted to develop complementary architectures that support *advanced channel reservation*. Such architectures are specifically tailored for large transfers. Their most important property is to offer hosts the ability to reserve in advance *dedicated* channels (paths) to connect their resources.

Providing efficient algorithms for advanced channel reservation is highly challenging. Variants of this problem have proved to be NP-complete [1]. Thus, most of the work conducted in this area consists of heuristics. What is missing, however, is an absolute benchmark against which the performance of these heuristics can be compared.

Our first contribution in this work is to derive an upper bound on the capacity (throughput) limits of advanced reservation architectures. This bound is based on a maximum concurrent flow optimization [2]. The time complexity to compute this bound is polynomial in the network parameters. No algorithm can exceed this bound and, thus, any heuristic approaching it must necessarily be near-optimal.

Our second contribution is a new competitive algorithm, called `RateComp`. We prove that, for *any* set of connection requests, the maximum delay experienced by each request is within a finite multiplicative factor of the value achieved with an optimal off-line algorithm. Furthermore, `RateComp` reaches the maximum rate (throughput) achievable by any algorithm.

We present simulations results showing that the saturation throughput of `RateComp` is close to the capacity bounds and far higher than that of a simple greedy algorithm.

This research was supported in part by the US Department of Energy under ECPI grant DE-FG02-04ER25605.

II. DERIVATION OF CAPACITY LIMITS

Our model consists of an arbitrary directed network topology, denoted by $G(V, E)$, where V is the set of nodes and E is the set of links connecting the nodes. The capacity of each link $e \in E$ is $C(e)$. A connection request contains the tuple (i, j, s), where $i \in V$ is the source node, $j \in V - \{i\}$ is the destination node, and s is the file size with mean $\bar{s}$. Requests arrive according to an arbitrary, but ergodic, stochastic process.

We assume that requests generate an average demand of $\lambda_{ij} = \alpha_{ij}\lambda$ bits/second from each node i to node j, where α_{ij} is a fixed parameter. We define the *saturation throughput* λ^* as the maximum value of λ such that the average delay experienced by requests is finite (*delay* is defined as the time elapsing between the arrival of a connection request until the completion of the corresponding connection).

Determining analytically the value of λ^* is a difficult task as the answer generally depends on the statistics of the arrival process. Instead, we derive a bound f^* on λ^* that holds for any arrival process. The derivation of the bound rests on the following argument:

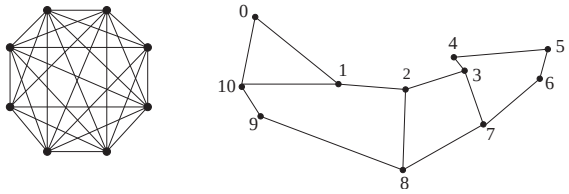
Lemma 2.1: If during any time interval T, each node $i \in V$ sends on average γ_{ij} bits of information per unit time to node $j \in V - \{i\}$, then there exists a multi-commodity flow allocation with flow values $f_{ij} = \gamma_{ij}$.

We note that the above lemma holds for arbitrary values of T. Taking $T \to \infty$, we thus identify γ_{ij} with λ_{ij}.

One can express the multi-commodity flow with a single parameter f by setting $f_{ij} = \alpha_{ij} f$. The maximization of f is a linear planning problem known as the *Maximum Concurrent Flow Problem (MCFP)* [2]. Using Lemma 2.1 we have:

Theorem 2.2: The maximum value of f in MCFP, denoted by f^*, provides an upper bound on the capacity limit of network G, i.e., $\lambda^* \leq f^*$.

Example: Consider the topologies illustrated in Fig. 1, with 20 Gb/s full-duplex links. Topology 1(a) represents an 8-node complete graph while topology 1(b) models the National LambdaRail, a network testbed governed by the U.S. research community. Assuming a uniform traffic distribution ($\alpha_{ij} = 1$), we obtain a capacity bound of $f^* = 20$ Gb/s for each flow in topology 1(a) and $f^* = 1.33$ Gb/s for each flow in topology 1(b). The result for topology 1(a) is expected since the flow between every pair of nodes can be routed through the direct link connecting them.



(a) 8-node clique (b) 11-node National LambdaRail

Fig. 1. Simulations topologies.

III. A COMPETITIVE ON-LINE ALGORITHM

A seemingly natural way to implement advanced channel reservation is to follow a greedy procedure, where, at the time a request is placed, the request is allocated a route guaranteeing the earliest possible completion time. We refer to this approach as `Greedy`. The problem with it is that requests may be allocated very long paths that consume sizable network resources. In fact, it is possible to show that for certain arrival patterns, the saturation throughput of `Greedy` is $\Omega(|V|)$ times smaller than the optimal throughput.

We propose an alternative approach. Instead of immediately reserving a path for each incoming request as in `Greedy`, we accumulate several arrivals in a "batch" and assign a more efficient set of flow paths to the whole batch.

We now present an algorithm, called `RateComp`, that implements this approach. The algorithm is based on the following assumptions:

- Path splitting (i.e., transmitting a flow over multiple paths in parallel) is allowed.
- At the time a request is placed, the algorithm returns only the starting time of the connection. The allocated paths and completion time are returned when the connection starts.

We define the function $\texttt{maxflow}(i,j)$ as the value of the maximum flow in bits per unit time from node i to node j satisfying the link capacity constraints. The algorithm can be described as follows:

1) For the first request (between nodes i and j) arriving at time $t = 0$, give an immediate starting time and an ending time of $t_c = 1/\texttt{maxflow}(i,j)$.
2) When another request arrives at time t:
   - If $t < t_c$, mark t_c as its starting time and add it to the waiting batch.
   - Else, start connection immediately, and update t_c with $t_c = t + 1/\texttt{maxflow}(i,j)$. Go back to step 2.
3) When $t = t_c$, calculate the maximum multi-commodity flow for the waiting batch (if any). Allocate to each job in the waiting batch a set of paths with bandwidths proportional to the multi-commodity flow, and set $t_c \leftarrow t_c + t'$, where t' is the minimum time to end the batch according to the multi-commodity flow. Go back to step 2.

The `RateComp` algorithm satisfies the following theoretical property on the maximum delay:

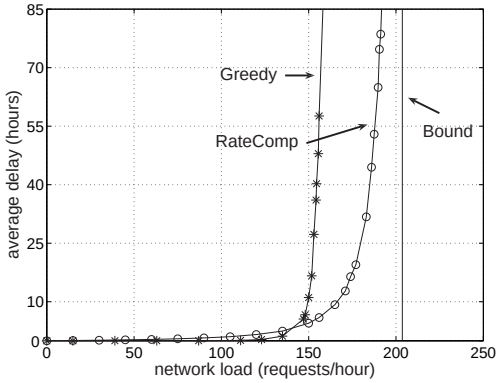


Fig. 2. Average delay of `RateComp` and `Greedy` as a function of the offered load. `RateComp` performs better than `Greedy` at high load and approaches the capacity limits shown by the bound.

Theorem 3.1: The maximum delay from the arrival of any request to the completion of the connection using `RateComp` is no more than $2/\epsilon$ times the maximum delay for an optimal off-line algorithm applied to the same set of requests in a network with link capacities $C(e)/(1+\epsilon)$, for all $e \in E$ and $\epsilon > 0$.

As a corollary of Theorem 3.1, the saturation throughput of `RateComp` is optimal because for any arbitrarily small $\epsilon > 0$, the delay of a request is guaranteed to be at most a finite multiplicative factor larger than the maximum delay of the optimal algorithm in the reduced resources network.

IV. SIMULATION RESULTS

We have developed a simulation tool in C++ to evaluate the performance of `RateComp`. The performance measures of interest are the average delay of requests and the saturation throughput. Clearly, the saturation throughput cannot exceed the capacity limits established in Theorem 2.2.

Figure 2 shows the average delay of `RateComp` and `Greedy` as function of the aggregated offered load for topology 1(a). The offered load is defined as the average number of connection requests per hour aggregated over the entire network. Source and destination of transmissions are selected uniformly at random, and the requested file size follows an exponential distribution with mean $\bar{s} = 2.475$ terabytes. Requests arrive to the system according to a Poisson process. For the purpose of simulation, we define the saturation throughput as the offered load at which the average delay exceeds 85 hours. The value achieved by `RateComp` is above 190 requests/hour which is far higher than the roughly 160 requests/hour obtained by `Greedy`. Furthermore, the saturation throughput of `RateComp` is close to the capacity limits. This result validates the near optimality of `RateComp` for the configuration under consideration.

REFERENCES

[1] A. Banerjee et. al. A Time-Path Scheduling Problem (TPSP) for Aggregating Large Data Files from Distributed Databases using an Optical Burst-Switched Network. In *Proc. ICC*, 2004. Paris, France.

[2] F. Shahrokhi and D. W. Matula. The Maximum Concurrent Flow Problem. *Journal of the ACM (JACM)*, **37**(2), April 1990.