Hierarchical Congestion Control (HCC): Cooperation of Uncorrelated Flows for Better Fairness and Throughput

Shiva Ketabi, Yashar Ganjali
Department of Computer Science, University of Toronto

***Abstract*—Congestion control protocols face several challenges for achieving max-min fairness and high throughput. First, each flow has a limited view of the network state. In the absence of a centralized congestion control entity, coordination is left (directly or indirectly) to individual flows. Second, most flows are very volatile by nature: flow rates/demands change significantly from one instant to another. In this poster, we present a hierarchical congestion control scheme to tackle these challenges. We aggregate flows with low correlation in a hierarchical manner, and recursively compute and allocate rates to these flows. Our preliminary experimental results show significant promise in terms of fairness and throughput.**

I. Introduction

A common weakness of most congestion control protocols is lack of cooperation and information sharing among flows. Congestion control protocols that are distributed by design delegate the task of congestion detection to individual flows. These individual flows are required to estimate the congestion state based on limited congestion signals they receive. These signals, ranging from packet loss to variations in packet round-trip latencies, are limited by nature given the limited information provided (single or a few bits of information).

This limited view of individual flows and lack of coordination among flows make reaching optimal state challenging. In particular, it is extremely difficult to reach optimum fairness and throughput in short periods of time, especially for short-lived flows. In contrast, let us imagine a congestion control framework which has access to all congestion control signals in the network, and can use this information to accurately infer congestion state of individual flows. This, for example, can happen if we have a logically centralized controller in the network that collects all individual congestion signals and globally allocates packet injection rates to individual flows, (*e.g.* by solving an optimization problem aiming at maximizing throughput while respecting fairness). Unfortunately, such a centralized congestion controller has its own challenges as gathering and processing information from a large number of flows in a timely manner is not scalable.

A novel approach in this direction is RackCC [1] which uses tunneling to aggregate all flows between pairs of top of rack switches (*i.e.* ToR-ToR traffic) as a single JumboFlow. Rack-level flow aggregation helps flows to combine their individual limited knowledge of the congestion state and to collectively agree on their optimum rates. Moreover, new flows, with no prior knowledge of the network, can acquire the congestion state of older flows and immediately set a near-optimal sending rate instead of wasting time and resources searching for the optimum rate.

In this poster, we propose Hierarchical Congestion Control (HCC) a model for hierarchical aggregation of congestion signals among flows sharing bottlenecks in the network. HCC is a generalization of RackCC which allows arbitrary aggregation of flows rather than limited ToR-ToR flow aggregation. In addition to being more general, HCC eliminates a fundamental problem in RackCC: flows that share the same source and destination racks are usually correlated. This is due to the locality of applications (*i.e.* to enhance latency) and due to the fact that these flows usually receive similar congestion signals from the network. Instead, HCC aggregates flows based on their correlation: the lower the correlation of any pair of flows, the higher their chance of being aggregated in the same group. Intuitively, the central limit theorem [2] suggests that by aggregating an adequately large number of independent flows, we will have a smooth and predictable distribution of demand rates, as opposed to significant fluctuations in aggregate demands of correlated flows.

II. Flow Aggregation, Rate Allocation

HCC consists of a *flow aggregation* phase running every T_{agg} seconds, and a *rate allocation* phase running every T_{alloc} seconds. We assume $T_{\text{alloc}} << T_{\text{agg}}$.

Flow aggregation. The flow aggregation phase iteratively merges the flows and produces an aggregation graph. It starts with the set of all flows $F^0 = \{f_1^0, f_2^0, \dots, f_N^0\}$ as input, generating a leaf node for each flow in the aggregation graph. During the i-th iteration, it merges flows sharing a path whose pairwise correlation is less than a threshold θ_i, calculated using Pearson Correlation Coefficient [2]. The flows in each group $f_{j_1}^i, f_{j_2}^i, \dots, f_{j_M}^i$, are replaced with a single flow f_j^{i+1} in the next iteration. The node representing f_j^{i+1} in the aggregation graph is the parent of all flows it replaces.

Each flow f_j^i is represented by (s_j^i, d_j^i, p_j^i) representing its source, destination, and a path between source and destination. For the merged flow f_j^{i+1}, the associated path is the path shared amongst all its children, and the first and last node of this path are considered to be the source and destination of this flow.

The aggregation phase is finished when no more flows can be aggregated, or when i reaches a predefined threshold $i_{\max}$. In the following two cases, we explain each of these termination conditions.

Case 1. Let us assume the aggregation is terminated at the k-th iteration since there are no shared links among flows $f_1^k, f_2^k, \ldots, f_{N_k}^k$. Here, each congestion bottleneck can belong to only one of these k-th level flows. Therefore, we can assign the total bandwidth of the bottleneck to the corresponding flow, and then recursively share the bandwidth amongst the children of this flow at level $k-1$. Since each flow in level $k-1$ might be included in more than one flow in level k, *i.e.* it might have more than one parent, it picks the minimum assigned rate among all its parents as its rate. Each node then shares its rate with its parents so that any excess bandwidth can be redistributed to other flows which need it.

Case 2. If the iterations stop with $i = i_{\max}$, we can use any known congestion control algorithms to allocate flow rates to the super-flows generated by the flow aggregation phase. In other words, the set of flows in this iteration, *i.e.* $F^i = \{f_1^i, f_2^i, \ldots, f_{N_i}^i\}$, can be considered as the only flows in the network and we can run any known congestion control scheme to allocate rates to them. RackCC [1] is a simple example here where $i_{max} = 1$. After one level of aggregation, RackCC uses known congestion control mechanisms to allocate rates to these ToR-ToR JumboFlows.

Rate allocation. The rate allocation phase consists of two different rounds: congestion signal collection and congestion reaction. These rounds are performed based on the child-parent relations between flows in the aggregation graph. In the congestion signal collection round, each parent receives congestion signals form all its children, aggregates them, and sends the aggregated congestion signal to its parents. In the congestion reaction round, the action taken by each flow in response to the congestion signals is pushed from each parent to its children.

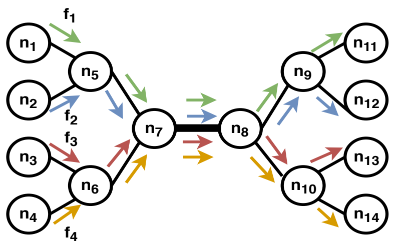

Fig. 1. An example topology

Example. Let us consider Figure 1 as an example. We have a bottleneck link (n_7, n_8), and four flows f_1, f_2, f_3, and f_4. Let us compare the following two different groupings:

Grouping 1. Merge f_1^0 and f_2^0 sharing the path (n_5, n_7, n_8, n_9) and create $f_1^1 = \{f_1^0, f_2^0\}$. Merge f_3^0 and f_4^0 sharing the path (n_6, n_7, n_8, n_{10}) and create $f_2^1 = \{f_3^0, f_4^0\}$. Note that this type of grouping does not follow our proposed approach since it aggregates flows from the same rack which are most likely correlated. In the next step, merge f_1^1 and f_2^1 since they share the path (n_7, n_8) and create $f_1^2 = \{f_1^1, f_2^1\}$. Figure 2(a) shows the resulting aggregation graph.

Grouping 2. Merge f_1^0 and f_3^0 sharing the path (n_7, n_8) and create $f_1^1 = \{f_1^0, f_3^0\}$. Merge f_2^0 and f_4^0 sharing the path (n_7, n_8) and create $f_2^1 = \{f_2^0, f_4^0\}$. Note that here we are grouping uncorrelated flows. In the next step, merge f_1^1 and f_2^1 sharing the path (n_7, n_8) and create $f_1^2 = \{f_1^1, f_2^1\}$. Figure 2(b) shows the resulting aggregation graph.

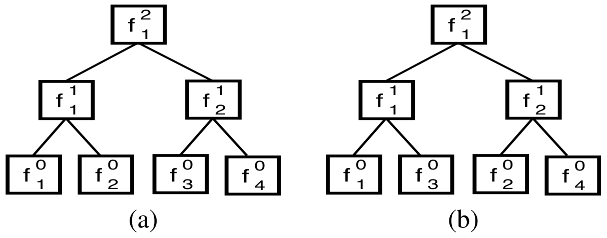

Fig. 2. Aggregation graphs. (a) correlated grouping, (b) uncorrelated grouping

III. Evaluation

We implemented HCC in ns-2, and created a scenario similar to Figure 1 with 100 flows (instead of 4 flows). The simulations were run for 500 seconds, measuring fairness (using Jain index [3]) and throughput of each flow. Figure 3(a) shows the CDF of fairness among all flows. When uncorrelated flows are grouped, 50% of flows have Jain index higher than 0.75 as opposed to 0.6 for correlated grouping (the higher the Jain index, the more fair the algorithm is). Figure 3(b) shows grouping uncorrelated flows also leads to higher throughput in general.

IV. Conclusion

Our hierarchical design makes the overhead of coordination of flows manageable, as we only need to deal with aggregated flows rather than individuals. Also, since we aggregate flows with low correlation, we are able to limit the variations in the aggregate demand (central limit theorem). Coordination among aggregate flows is also easier due to slower changes in aggregate flow demands.

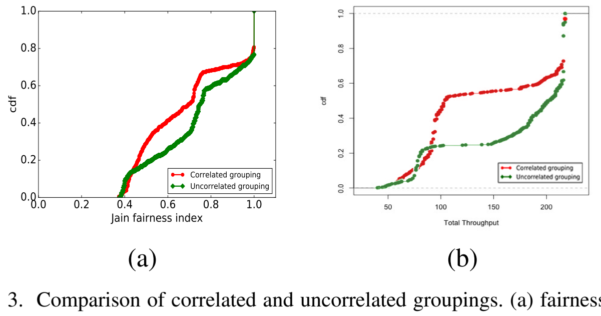

Fig. 3. Comparison of correlated and uncorrelated groupings. (a) fairness (b) throughput

References

[1] D. Zhuo, Q. Zhang, V. Liu, A. Krishnamurthy, and T. Anderson, "Rack-level congestion control," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 148–154.

[2] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "Carpo: Correlation-aware power optimization in data center networks," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 1125–1133.

[3] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," in *ATM Forum contribution*, vol. 99, no. 45, 1999.