Multi-Commodity Flow with In-Network Processing

Moses Charikar[1], Yonatan Naamad[1], Jennifer Rexford[1], and X. Kelvin Zou[1]

[1]Department of Computer Science, Princeton University
{*moses,ynaamad,jrex, xuanz*}*@cs.princeton.edu*

1 Introduction

1.1 Background Networks have evolved beyond a simple packet forwarding. Many types of in-network flow processing are ubiquitous today. The flow processing usually take place at vertices, in which we install one or a few appliances, so called middleboxes(MBoxes), examples including firewall, encryption and compression. In many cases the flow has to go through a sequence of MBox processing in order to fulfill the function or security requirements.

Traditionally, these MBoxes use specialized hardware and are installed at predetermined locations(vertices). In terms of the locations, some design places MBoxes at juncture points(e.g., gateways) in special network topologies such as FatTree[1], but it leads to inefficient routing in cases where the both source and sink are sharing the same predecessors. It also fails to cover generalized networks or enforce the policy for a sequence of MBoxes.

The recent development of MBox virtualization enables more flexible deployment. MBoxes are no longer hardware dependent and can be spin up(or down) at any physical server[6, 7]. In this new architecture, the processing capacities are homogeneous and the problem becomes simply where to place MBox servers. There are two choices: 1. separate the server placement and routing decision: place MBox servers first and make routing decisions based on MBox locations[8]; 2. have a joint optimization scheme where the server locations are decided based on potential routing decisions given the flow demand. We argue that the latter is better in terms of efficiency but imposes the hardness for the server placement.

1.2 The Problem We can abstract the flow in-network process problem in the following way: there is a flow demand with multi-sources and multi-sinks, and each flow requires a certain amount of in-network (MBox) processing. The in-network processing required for a flow is proportional to the flow size and without losing generality, we assume one unit of flow requires one unit of processing. For a flow from a source to a sink, we assume it is an aggregate flow so the routing and in-network processing for a flow are both divisible. In this model there are two types of constraints: edge capacity and vertex capacity, which represents bandwidth and MBox processing capacity. A feasible flow pattern satisfies: the sum of flows on each edge is bounde by the edge capacity, the sum of in-network process done at each vertex is bounded by the vertex capacity, and the processing done at all vertices for a flow should be equal to flow size and the processing has to be carried out by the flow.

Network design problems involve finding a network with a minimum cost that satisfies various properties, often related to routing and flow feasibility. The network design problem in this model is: for a multi-source and multi-sink flow demand, and a fixed amount of budget to install the processing capacities in the vertices, what is the optimal way to allocate them such that the flow pattern satisfies all the constraints.

Our model is a superset of multi-commodity flow model[5], that is, if we can solve this problem, we can naturally solve multi-commodity flow problem: simply assign each vertex with an infinite capacity

it becomes an MCF problem. Many classic MCF design properties can be extended to this problem, such as MC-BB(multi-commodity buy-at-bulk) problem[2, 3, 4].

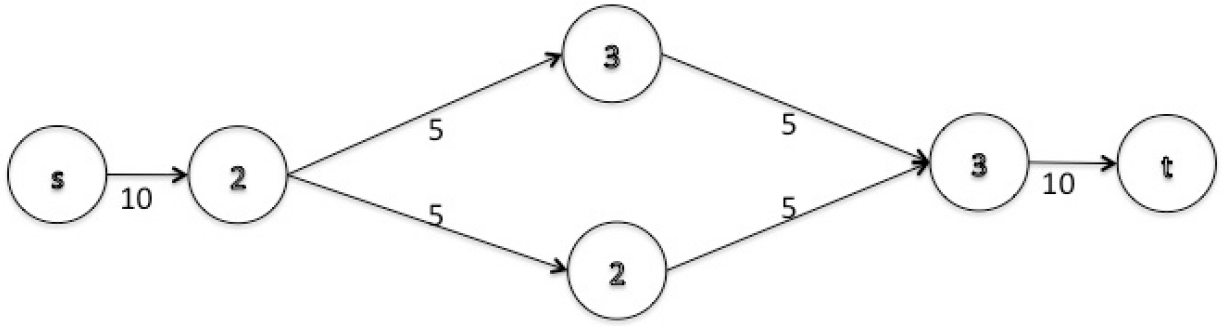

Figure 1: **Example for the Graph Model: given the edge and vertex constraints,** {5,10} **are edge capacities and {2,3} are vertex capacities, the maximum flow from source to sink in the example is 10, in this example all edges and vertices will be saturated.**

1.3 Notations To capture both the capacitated vertices and edges, we use the following notations: a directed graph $G(V, E)$ where each edge $e \in E$ has an edge bandwidth $B(e)$, and each vertex $v \in V$ has a processing capacity $C(v)$.

1.4 The results in this paper

2 Processing Capacity Placement in fractional manner

We formulate a generalized linear programming model in this problem in an edge based formulation. Notations: $C(v), B(e)$ represents the processing and edge capacity at node v and edge e, $f_i(e)$ represents the flow size for each source-sink pair on edge e, $w_i(e)$ represents process demand for flow i carried out at edge e, $f(C(v))$ is an concave cost function of $C(v)$:

$$\text{Minimize:} \sum_v f(C(v)) \tag{2.1a}$$

$$\text{Subject to:} \forall v \in V - \{s,t\}, \forall i, \sum_{in} f_i(e) = \sum_{out} f_i(e); \tag{2.1b}$$

$$\forall e, \sum_i f_i(e) \leq B(e); \tag{2.1c}$$

$$\forall e, \forall i, 0 \leq w_i(e) \leq f_i(e), \tag{2.1d}$$

$$\forall v, \forall i, 0 \leq \sum_{in} w_i(e) - \sum_{out} w_i(e) \leq C(v). \tag{2.1e}$$

$$\forall i; \forall (s,v), w_i = f_i \ \& \ \forall (v,t), w_i = 0. \tag{2.1f}$$

References

[1] Al-Fares, M., Loukissas, A., and Vahdat, A. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev. 38*, 4 (Aug. 2008), 63–74.

[2] Awerbuch, B., and Azar, Y. Buy-at-bulk network design. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science* (1997), pp. 542–547.

[3] Charikar, M., and Karagiozova, A. On non-uniform multicommodity buy-at-bulk network design. In *Proc. of ACM STOC* (2005), ACM Press, pp. 176–182.

[4] Chekuri, C., Hajiaghayi, M. T., Kortsarz, G., and Salavatipour, M. R. Approximation algorithms for node-weighted buy-at-bulk network design. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1265–1274.

[5] Ford, L. R., and Fulkerson, D. R. A suggested computation for maximal multi-commodity network flows. *Management Science 5*, 1 (1958), 97–101.

[6] Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. Clickos and the art of network function virtualization. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)* (Apr. 2014), USENIX Association, pp. 459–473.

[7] Patel, P., Bansal, D., Yuan, L., Murthy, A., Greenberg, A., Maltz, D. A., Kern, R., Kumar, H., Zikos, M., Wu, H., Kim, C., and Karri, N. Ananta: Cloud scale load balancing. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (2013), SIGCOMM '13, ACM, pp. 207–218.

[8] Qazi, Z. A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., and Yu, M. Simple-fying middlebox policy enforcement using sdn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (2013), SIGCOMM '13, ACM, pp. 27–38.