Abstracting Networks with Measurable Guarantees

Vitalii Demianiuk
Ariel University, Israel

Kirill Kogan[†]
Ariel University, Israel

Antonio Fernández Anta
IMDEA Networks Institute, Spain

***Abstract*—To simplify definitions of network-wide behaviors (e.g., in datacenter transports), networks are often represented by virtual switches. In most cases, the buffering architecture of a *representing* virtual switch is inherited from analytic models implementing the desired properties, and is completely decoupled from the *represented* network topology. Thus, it is unclear how well the network infrastructure is exploited. This paper makes the first attempt in understanding which buffering architectures can best represent a given network, and how buffer management decisions can be mapped back to a represented network.**

I. Introduction

Recently, there has been a significant amount of effort to devise efficient network-wide behaviors optimizing performance in modern datacenters [1–6]. The design process of such schemes is complex and, as a result, usually consists of the following simplifying steps: (1) represent a given network topology G by a virtual switch S; (2) for switch S, select algorithm A^S optimizing the desired objective according to specific traffic characteristics; (3) construct network-wide behavior A^G on G, emulating the behavior of A^S. For instance, in pFabric [1]: (1) a *combined input-output queued* (CIOQ) switch S represents a network topology G; (2) the approximation algorithm A^S minimizing average flow completion time in S is selected; (3) to emulate the behavior of A^S in G, the proposed pFabric transport exploits remaining flow sizes as priorities during the processing of flow packets in individual switches. In this example, the choice of a *representing* virtual switch is mostly motivated by the existence of a buffer management policy for the considered objective, and is completely decoupled from the *represented* network topology. Such virtual switch representations mostly guide design principles of network-wide behaviors and do not allow to map decisions in the representing virtual switches to decisions in the corresponding network-wide behaviors with the same performance guarantees.

In this paper, we are going in the opposite direction, and propose transformations of given networks to virtual switches that incorporate network topology constraints and do not depend on specific buffer management policies. In the long term, we are interested in building a procedure for automating the process described above. This procedure automatically constructs a virtual switch S representing a given topology G, and selects an algorithm A^S minimizing the desired objective on S (Steps 1 and 2). Then, the proposed automation system maps the buffer management decisions of A^S to the scheduling decisions of A^G preserving measurable guarantees (throughput, latency, fairness, etc.) on the originally given network G (Step 3). To implement this mapping automatically, link capacities in G should be somehow represented in S. This can be naturally achieved by tuning buffer sizes in S, where buffers in our case represent bottleneck capacities of paths connecting sources with destinations. This paper is devoted to the initial deep understanding of this network virtualization process.

[†] Kirill Kogan died on 13 March 2021 from complications of the COVID-19 disease.

[$] The work was supported in part by: Israeli Innovation Authority under the Knowledge Transfer Commercialization Program (MAGNETON), grant 71249; Ariel Cyber Innovation Center in cooperation with the Israel National Cyber Directorate in the Prime Minister's Office; Regional Government of Madrid (CM) grant EdgeData-CM (P2018/TCS4499, cofunded by FSE & FEDER); Spanish Ministry of Science and Innovation grant ECID (PID2019-109805RB-I00, cofunded by FEDER)

Annex to ISBN 978-3-903176-39-3© 2021 IFIP

II. Network as a virtual switch

Consider a virtual network represented by a weighted directed graph $G = (V, E, c)$. The set V consists of *source* vertices having only outgoing *source* links, *destination* vertices having only incoming *destination* links, and intermediate nodes. Traffic in G is transmitted from sources to destinations across simple paths. For simplicity, we assume that the time is slotted, and in every time slot each source receives a set of unit-sized packets destined to various destinations. The capacity $c(u, v)$ of a link $(u, v) \in E$ defines the maximum number of packets that can be transmitted through (u, v) per time slot. Since several virtual networks can share the same physical infrastructure, that is tailored for specific application requirements, link capacities in G can be highly heterogeneous even for regular physical datacenter topologies. The scheduling algorithm A^G on G transmits packets from sources to destinations minimizing a desired objective. Each time slot consists of two phases: (1) for each link (u, v) in G, A^G chooses a subset of packets to be sent through (u, v) (*selection phase*); (2) A^G transmits the selected packets (*transmission phase*).

Our goal is to transform network G to a virtual switch S satisfying two conditions:

- *feasibility* – for any work-conserving buffer management policy A^S on S, there is scheduling algorithm A^G on G, delivering the same subset of packets;
- *completeness* – for any scheduling algorithm A^G on G, there is buffer management policy A^S on S delivering the same subset of packets.

The feasibility condition guarantees that one can pick up a buffer management algorithm A^S, and the infrastructure will be able to map its decisions to A^G in the represented network G. The completeness condition guarantees that a virtual switch representation does not restrict the algorithmic decisions on G. We show how to construct a virtual switch representation satisfying these two conditions for networks containing at most two bottlenecks on each path from a source to a destination.

2-width networks. First, we consider 2-*width networks*, where every path connecting a source with a destination contains at most two links. The set of such network topologies includes commonly used star topologies, bipartite graphs, and *leaf-spine* topologies [7]. In the following, $v_1, v_2, \dots v_t$ is a set of intermediate nodes connecting sources s_1, s_2, $\dots s_k$ to destinations d_1, d_2, $\dots d_n$ in a 2-width network G (see Fig. 1). Every path in G going from a source to a destination contains at most one intermediate vertex.

We represent 2-width network G by CIOQ switch S, where each source link (s_i, v_j) corresponds to a separate input port $I_{i,j}$ in S with its attached *input* queue, and each destination link (v_j, d_i) corresponds to an output port $O_{i,j}$ in S with its attached *output* queue (see Fig. 1). The port rates and the queue sizes are equal to the capacities of the corresponding links. For every queue Q in S, a buffer management policy A^S defines an admission policy deciding which packets are admitted to Q. That is further translated into the selection phase of the corresponding scheduling algorithm A^G for the link represented by the port of Q. The transmission policy of Q defines the transmission phase of the corresponding link.

Observe that in a 2-width network G, there may be multiple paths between each source and each destination. The path for each packet either (1) is selected in advance, or (2) should be chosen by the scheduling algorithm A^G. In the first case, for each incoming packet, the corresponding input queue in S is already defined. In the second case, for each packet p arrived at source s_i, buffer management policy A^S should also select a queue that will contain p among queues corresponding to source links originated in s_i; this selection defines how s_i chooses a path in G for packet p. Note that intermediate nodes do not make any routing decisions, since every path from an internal node to a destination consists of a single link.

Assume that, in Fig. 1, at the current time slot, s_1 has ten packets ready for transmission to d_1 and all sources have no other packets ready for transmission. First, s_1 selects four packets for the transmission through (s_1, v_1) and two packets through (s_1, v_2) according to the routing decisions made by A^S and the admission policies of the corresponding queues in S. In the next time slot, v_1 selects three packets from the four received according to the admission policy of the queue assigned to $O_{1,1}$ and sends them to d_1, while v_2 transmits all received packets to d_1.

The proposed transformation of 2-width network G to the virtual switch S satisfies both feasibility and completeness conditions since port rates and queue sizes are equal to the capacities of corresponding links. Note that this would not be the case if the port rate of input port $I_{i,j}$ and the size of the attached queue are both strictly greater than $c(s_i, v_j)$: the feasibility condition is not satisfied since there is a buffer management policy A^S that in one time slot can fetch more packets from $I_{i,j}$ than the number of packets that can be transmitted simultaneously through link (s_i, v_j) by any algorithm A^G. If the size of $I_{i,j}$'s queue would be strictly less than $c(s_i, v_j)$, the completeness condition will not be satisfied since A^G can simultaneously transmit $c(s_i, v_j)$ packets through (s_i, v_j), while any A^S can fetch at most $c(s_i, v_j) - 1$ packets from $I_{i,j}$ in a single time slot. The same observations hold for output ports and output queues in S.

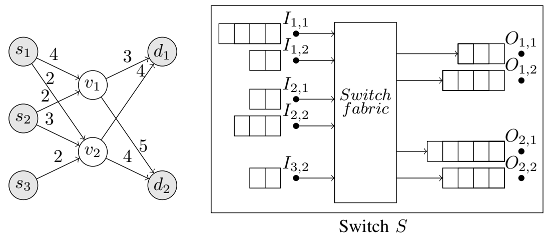

Figure 1. Representing a 2-width network G by virtual CIOQ switch S.

Note that a 2-width network G may contain link (s_i, d_j) directly connecting source s_i with destination d_j. The capacity constraint for such a link is independent from the capacity constraints of other links in G. Hence, (s_i, d_j) is represented in S by a separate port $P_{i,j}$ with the attached queue of size $c(s_i, d_j)$; the port rate of $P_{i,j}$ equals to $c(s_i, d_j)$. In S, packets fetched from $P_{i,j}$ do not traverse the switching fabric. Hence, we can omit such queues and the corresponding links from the future consideration.

Representing bottlenecks. Many recent datacenter transports, as Homa [5], pHost [4], pFabric [1], etc., were designed under the assumption that in the network, only source and destination links can be bottlenecks, i.e., the network can transmit every set of packets satisfying capacity constraints of source and destination links. Such network G can be represented by a virtual switch S in almost the same way as a 2-width network: (1) every source/destination link is represented by the input/output port with the attached input/output queue; (2) the port rates and sizes of the attached queues are equal to the capacities of the corresponding links; (3) the admission policy of the queue defines a selection phase of the scheduling algorithm A^G for the corresponding link in G. As in the case of 2-width networks, for a packet p without predefined path, a buffer management policy A^S must also select input and output queues that will hold p, among the queues corresponding to links outgoing from the source of p and links incoming to the destination of p, respectively; this selection defines all the bottlenecks appearing on a path of p.

III. Simplifying virtual switch

Both the resource and operational complexities of the buffering architecture inside a virtual switch S depend on the number of queues in S and their total size. Here, we propose simplifying transformations of a virtual switch S representing a 2-width network G. Note that these transformations can be extended to representations of general networks containing bottlenecks only at source and destination links.

Simplifying transformations. Let $c_{in}(v_j)$ be the total capacity of all source links incoming to v_j. Consider output port $O_{i,j}$ such that $c(v_j, d_i) \geq c_{in}(v_j)$. In this case, node v_j can always transmit all received packets going to d_i. It means that the selection phase of A^G for link (v_j, d_i) is trivial and does not require any additional computational resources. A specification of an admission policy for the corresponding output queue is not required, since it always accepts all given packets. Hence, we can assume that the queue assigned to

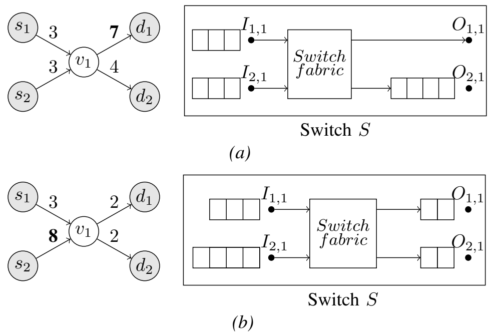

Figure 2. Simplifying transformations: (a) removing the output queue; (b) reducing the size of the input queue.

port $O_{i,j}$ is not required. In Fig. 2a, the maximum number of packets that can be transmitted from all sources to v_1 per time slot is six, which is less than the maximum number of packets that d_1 can receive from v_1 in one time slot. Hence, the queue assigned to port $O_{1,1}$ can be safely removed.

In the extreme case, this transformation can remove all output queues in the virtual switch representation. In such scenario, the switching fabric inside the virtual representation is not required, and a given 2-width network can be represented by a *single queue per-port* (SQ) switch S that consists of multiple independent ports with attached queues corresponding only to source links. For instance, consider a network G whose link capacities are the same as in Fig. 2a except the capacity of (v_1, d_2) that now equals to six. This network can be represented by a SQ switch, since the capacities of the destination links do not impose any constraints on the traffic transmission.

Let $c_{out}(v_j)$ be the total capacity of all destination links outgoing from v_j. For a source s_i in G, it makes no sense to send more than $c_{out}(v_j)$ packets to v_j in one time slot, since all packets traversing link (s_i, v_j) and reaching destinations at time t can be transmitted to v_j at time $t-1$ if $c(s_i, v_j) \geq c_{out}(v_j)$. Hence, for each source link (s_i, v_j) with $c(s_i, v_j) > c_{out}(v_j)$, we assign to the corresponding input port a buffer of the size $c_{out}(v_j)$ instead of $c(s_i, v_j)$. In this case, the total size of a buffering architecture becomes smaller but the number of queues in S remains the same. In Fig. 2b, the size of the queue assigned to $I_{2,1}$ is $c_{out}(v_1) = 4$.

Modifying link capacities. The modification of link capacities can additionally reduce the number of output queues in a virtual switch S representing a 2-width network G. For instance, to remove the output queue assigned to port $O_{i,j}$ in S we can increase $c(v_j, d_i)$ to be no less than $c_{in}(v_j)$. In Fig. 2a, we can remove the queue assigned to port $O_{2,1}$ if we increase $c(v_1, d_2)$ by 2. Note that extra capacity is a scarce resource that should be allocated wisely to simplify S.

An additional approach for removing output queues arises from the reduction of source link capacities. In particular, the capacity reduction of source link (s_i, v_j) decreases $c_{in}(v_j)$ and hence increases the applicability of the transformation that removes output queues corresponding to destination links outgoing from v_j. For instance, in Fig. 2a, the reduction of $c(s_1, v_1)$ by two allows to remove the output queue assigned to $O_{2,1}$. The total value of source link capacity reduction controls a fundamental tradeoff between network bandwidth capabilities and the simplicity of the representing virtual switch.

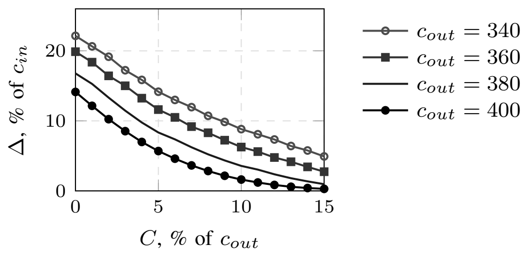

Figure 3. The total capacity reduction Δ (in %) of source links required in a leaf-spine network G to represent G by a SQ switch using C (in %) extra capacity; network G contains 10 sources, 10 destinations, and 2 intermediate nodes; the total capacity c_{in} of source links in G equals 200; the total capacity c_{out} of destination links in G varies from 340 to 400; each destination can receive packets from at most two sources.

To represent a 2-width network G with a SQ switch S, we can use both of the approaches described above. First, to decrease the values of $c_{in}(v_j)$ in G, we reduce capacities of source links. Then, we increase capacities of destination links to make them no less than the reduced values of $c_{in}(v_j)$. For instance, to represent a network from Fig. 2a with a SQ switch, we can reduce $c(s_2, v_1)$ by one and increase $c(v_1, d_2)$ by one.

Extra knowledge about traffic patterns can further simplify representing virtual switches. For instance, assume that in given network G only sources in set $\mathcal{S}_i$ can transmit packets to destination d_i. In this case, we can remove the queue attached to port $O_{i,j}$ in S representing G even if $c(v_j, d_i)$ is smaller than $c_{in}(v_j)$ and no less than the total capacity of all links connecting sources in $\mathcal{S}_i$ with v_j. Fig. 3 shows that, under such partial knowledge about traffic patterns, a leaf-spine topology can be represented by a SQ switch after the reduction of source link capacities by 2% in total and the increase of destination link capacities by 10% in total. We omit the description of the heuristics selecting links in G for the capacity reduction and extra capacity allocation due to the lack of space.

IV. Conclusion

In this work, we show how virtual switches can abstract given network topologies and propose transformations simplifying virtual switch representations. This paper makes the first steps towards this direction. In the future, we will explore representations of networks containing multiple bottlenecks that are not necessarily source or destination links.

References

[1] M. Alizadeh *et al.*, "pFabric: Minimal Near-Optimal Datacenter Transport," in *SIGCOMM*, 2013, pp. 435–446.

[2] V. Arun *et al.*, "Copa: Practical Delay-Based Congestion Control for the Internet," in *NSDI*, 2018, pp. 329–342.

[3] M. Dong *et al.*, "PCC: Re-architecting Congestion Control for Consistent High Performance," in *NSDI*, 2015, pp. 395–408.

[4] P. X. Gao *et al.*, "pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric," in *CoNEXT*, 2015, pp. 1–12.

[5] B. Montazeri *et al.*, "Homa: A Receiver-driven Low-latency Transport Protocol Using Network Priorities," in *SIGCOMM*, 2018, pp. 221–235.

[6] M. Alizadeh *et al.*, "Data Center TCP (DCTCP)," in *SIGCOMM*, 2010, pp. 63–74.

[7] M. Alizadeh *et al.*, "On the Data Path Performance of Leaf-Spine Datacenter Fabrics," in *HOTI*, 2013, pp. 71–74.