A Survey on Resource Scheduling for Data Transfers in Inter-Datacenter WANs

Jessie Hui Wang[a,b], Jilong Wang[a,b], Changqing An[a,b], Qianli Zhang[a,b]

[a] *Institute for Network Sciences and Cyberspace, Tsinghua University, China*
[b] *Beijing National Research Center for Information Science and Technology, China*

Abstract

Today, datacenter providers spend a lot of money on inter-datacenter WANs to transmit traffic flows between geographically distributed datacenter sites. It is very important for datacenter providers to design resource scheduling schemes to schedule inter-datacenter networking resources to satisfy data transfer requests and achieve maximum performance or economic benefits. In this article, we conduct a review study on research efforts being conducted on the inter-datacenter networking resource scheduling problem. Our survey study is presented in three parts. The first part focuses on formulating the problem theoretically, including the definition and features of a data transfer, network models of inter-DC WANs, scheduling objectives, and scheduling dimensions. We also summarize a conceptual implementation architecture of scheduling systems. Second, we classify existing schemes according to their objectives and scheduling dimensions and then explain how they formulate and solve their own problems. In the third part, we examine practical challenges in developing scheduling systems and also point out some future directions. Since we do not see any survey focusing on the scheduling problem of inter-DC WANs, we believe this article can provide some help for research developments in this important field by organizing logically the recent research efforts from industry and academia.

Keywords: Inter-DC WAN, Scheduling, Resource Allocation, Traffic Engineering, Datacenter

1. Introduction

In recent years, many Internet Content Providers (ICPs), Internet Service Providers (ISPs) and other corporations deploy large-scale datacenters at multiple geographical locations for scalability, robustness or performance (*e.g.*, latency) considerations. Each of them needs to build its *Inter-DC WAN* to connect its multiple datacenter sites, and measurement study shows that the inter-datacenter traffic has been growing at a very fast rate. For example, Google's WAN is delivering terabits/sec of traffic across the earth. It is reported that the total cost of the inter-DC WAN even exceeds the cost of networking within a datacenter site and becomes one of the major components of the cost for datacenters [1].

However, datacenter providers are unable to fully leverage this investment with only traditional Internet technologies. Traditionally, in the Internet, WAN links are often provisioned at $30-40\%$ on average to protect against traffic bursts, failures and packet loss. Such low utilization is unacceptable for inter-DC WAN links because low utilization and huge traffic volume would make WANs prohibitively expensive.

There are at least two reasons for the poor efficiency of traditional WANs. First, traditional WANs always treat all bits the same. As a result, WAN links have to be over-provisioned to mask link or router congestions and failures from the traffic of sensitive applications. If we are capable to offer satisfactory performance (*e.g.*, low latency, enough bandwidth) for performance-sensitive applications even when the WAN is congested, we can avoid the necessity of over-provisioning. Second, traditional resource allocations are usually implemented in a distributed manner wherein no one has a global view and each node independently manages its own resources, so that the global optimal allocation cannot be reached.

Considering the above two culprits for low efficiency, in order to achieve high efficiency, datacenter providers may want to treat traffic flows of various applications differently according to the applications' own requirements, and the resource allocation should be done in a centralized manner with a global view. For example, intuitively, we can have a central controller which collects deadline requirements of all flows, then some flows without deadline requirements can be postponed to make room for latency-sensitive flows when networking resources are not sufficient. In this way, we do not need to increase provisioning to satisfy the requirements of latency-sensitive flows, and the operation cost of the datacenter provider is reduced.

In fact, the idea of treating traffic flows of applications differently had ever been proposed to improve the efficiency of the Internet. For example, decades ago, researchers proposed Intserv and Diffserv mechanisms to improve efficiency by setting different priorities for different traffic classes in the Internet.But they have never been de-

*Corresponding author
Email address: jessiewang@tsinghua.edu.cn (Jessie Hui Wang)

ployed at large scale, mainly due to scalability and management issues.

Fortunately, inter-DC WANs exhibit a number of unique characteristics which are different from the Internet, and these properties enable us to improve the efficiency of inter-DC WANs by designing proper scheduling schemes. First, one inter-DC WAN is always operated by a single operator, so the complete view of the WAN is available. For private datacenters such as Google, it can even be able to control the sending rates of applications and end hosts. By controlling more factors, inter-DC scheduling schemes can extract more values. Second, generally one inter-DC WAN has about a few dozens of sites, which is far smaller than the Internet, and it makes centralized optimization computationally tractable. Third, there are a lot of elastic traffic demands that offer great flexibility for scheduling. For example, data backup among sites, which is one major source of inter-DC traffic demand, only requires to be completed before a loose deadline, therefore we can allocate it less resource when the WAN is heavy-loaded and compensate it later. Due to these reasons, it is feasible and valuable to optimize the scheduling of data transfers in inter-DC WANs using a centralized mechanism and enabling a reliable global view of network and user status that facilitates coordination among transfers.

1.1. Problem Definition and Motivation

As we described above, it is valuable and feasible to develop innovative scheduling schemes that can schedule inter-DC networking resources for data transfers to exploit inter-DC WANs efficiently instead of dumb over-provisioning. However, although the basic idea to improve efficiency is simple, how to design schemes and implement them still faces significant technical challenges and deserves significant attention from both academia and industry.

First, inter-DC data transfers may have various sizes, deadlines and performance requirements. We have to understand them well to model, categorize, and aggregate them for scheduling. Second, the availability and cost of networking resources can be very different for different datacenter providers. For example, if a datacenter provider deploys a dedicated inter-DC WAN, its networking resource and cost would be fixed, and thus it would like to provide service for as many as possible of data transfers; if the datacenter provider is buying its networking resources from ISPs under a usage-based charging model, the amount of its networking resources can be adaptive to the intensity of data transfers, but it has to consider the cost and revenue of providing service for a data transfer and may reject transfers with negative profit. Third, datacenter providers may have various scheduling objectives, such as priority or fairness among transfers, maximum revenue or social welfare. Fourth, the ability of datacenter providers to control the provision and usage of inter-DC WANs varies greatly. One datacenter provider might be able to control one or more of the following things: routing of data packets, sending rate of each flow, provisioned bandwidth, charges to users, WAN topology, *etc.* We refer to them as *scheduling dimensions* or *decision variables* of the datacenter provider.

Generally speaking, the *resource scheduling* problem of inter-DC WANs is defined as follows:

Given: (a) a set of data transfer requests $\mathcal{D}$, (b) a networking model which includes a fixed or variable amount of networking resources $\mathcal{R}$ with a cost of $\mathcal{C}(\mathcal{R})$, (c) a set of scheduling dimensions $\mathcal{S}$, and (d) a set of given objectives $\mathcal{U}$,

Question: the *resource scheduling* problem of inter-DC WANs is to determine "optimal" values for all scheduling dimensions, such that the given objectives are maximized under the constraints of networking resources.

Here, one scheduling dimension $\mathcal{R}_i$ can be a variable or a vector of variables of the network model, such as the bandwidth or sale price of each link; it can also be a variable or a vector of variables that represents a decision made for every transfer, such as a vector of routing paths or sending rates. $\mathcal{D}$, $\mathcal{R}$, $\mathcal{C}(\mathcal{R})$, $\mathcal{S}$ and $\mathcal{U}$ can be very different in different inter-DC WANs. We will describe them in detail in Section 2.

While there has been a wide body of research published in prestigious journals and conferences (*e.g.*, Sigcomm 2011, Sigcomm 2013-2016, Sigcomm 2018) in recent years, we have not seen any survey articles focusing on the resource scheduling of inter-DC WANs. We are therefore motivated to survey the latest papers in this area and provide a starting ground for interested readers. In this article, we will conduct a review study on recent research efforts from datacenter operators (*e.g.*, Google, Microsoft), ISPs (*e.g.*, Telefonica), and universities. We believe it can provide some help for research developments in this important field by organizing logically the recent research efforts from industry and academia.

1.2. Related Research Directions and Surveys

A datacenter is a set of servers connected by networking devices such as routers or switches. Traditionally, datacenters run applications on these dedicated physical servers directly. After the emergence of server virtualization technologies, most datacenters are virtualized to provide elastic virtual computing, storage and networking resources for various applications, such as cloud computing.

As cloud computing is more and more widely used, the research on datacenter and cloud computing draws a lot of attention, and there have been a lot of surveys focusing on various issues of datacenters, such as infrastructure architecture and routing of datacenter networking (DCN) [2] [3] [4] [5] [6] [7] [8], virtualization technologies [9][10], energy consumption [11][12][13], and SDN for DCN [14] [15].

Among various issues, resource management and scheduling is regarded as one of the key future directions for cloud computing research [16]. The authors of [17] outline a conceptual framework for cloud resource management and

structure related works according to the framework. The article [18] conducts a survey on resource orchestration in terms of concepts, paradigms, languages, models, and tools. There have also been some survey articles on resource scheduling algorithms, but these articles are working on *task/workload scheduling* and *flow scheduling*, and none of them focus on the resource scheduling of inter-datacenter WANs.

Cloud task/workload scheduling. In [19], resource scheduling is defined as "mapping of workloads with appropriate resources". In clouds, cloud applications are run by cloud tenants (or SaaS cloud providers) who apply for virtual cloud resources to satisfy their end users. Conventionally, these requests from end users are referred to as *workloads* or *tasks*. The objective of workload scheduling algorithm is to solve these questions: 1) schedule each task onto a suitable virtual node; 2) place each virtual node on an appropriate physical node and provision it with an appropriate amount of physical resources, *i.e.*, VM placement/consolidation/migration.

In [19], Singh and Chana present a thorough survey on the workload scheduling problem, and they analyze thirteen types of resource scheduling algorithms and eight types of resource distribution policies. The authors of [20] present a taxonomic survey on the algorithms to balance workloads across datacenter resources by assigning workloads to light-loaded nodes or allocating more physical resources to overloaded VMs. The article [21] also reviews load balancing techniques and implementations in cloud computing. The algorithms to determine the mapping between virtual machines and physical machines are surveyed in [10] and [22]. The fairness issue in resource allocation mechanisms is discussed in [23]. Some survey articles focus on a single one particular type of algorithms or approaches for the scheduling of cloud resources. For example, in [24], the authors only focus on evolutionary computation algorithms, such as GA, ACO, and PSO. In [25], the authors review the applications of economic and pricing models to develop adaptive algorithms and protocols for resource scheduling.

The workload scheduling problem mainly focuses on computing, storage and energy resources. Although some algorithms are network-aware, they do not consider the allocation of networking resources. In [20], the authors point out the effective utilization of networking resources is a significant research direction and that the problem has not been properly addressed.

Flow scheduling in DCN. The allocation and scheduling of networking resources has also been discussed in several survey articles, but they mainly focus on intra-datacenter scenarios, which are different from our survey.

End-to-end traffic control via transport control protocols is surveyed in [26]. The authors of [27] also present a tutorial on various datacenter traffic control solutions, such as transmission control, traffic shaping, prioritization, and multi-pathing. The authors of [28] present a survey of solutions for traffic load balancing in data center networks. The article [22] also reviews research efforts in intra-datacenter traffic engineering and end-to-end flow control via transport protocols. Most of these algorithms are designed based on the special data center topologies and traffic characteristics.

Data processing frameworks such as MapReduce and Hadoop are popular applications running in cloud networks. These data-parallel applications generate collections of semantic-related flows (defined as *coflow*) between multi-stages, and the transmission of these coflows has a significant influence on their job-level performance. In [29] and [30], the authors conduct surveys on coflow scheduling algorithms which are to optimize the transmission of data flows generated by those frameworks.

Although inter-datacenter communication is regarded as an evolving research area that requires further attention [27], the articles mentioned above either focus only on intra-datacenter traffic control or just mention resource scheduling in inter-datacenter networks very briefly. Our article presents a complete and detailed survey on inter-datacenter resource scheduling.

1.3. Organization

We base our discussion on an overall classification of surveyed schemes according to their scheduling dimensions and scheduling objectives. As shown in Figure 1, we classify existing research efforts into the following main areas:

1. *deadline-agnostic schemes.* These schemes consider only the spatial dimension, *i.e.*, they try to control the sending rates and sending paths of data flows to achieve their goals.
2. *deadline-aware schemes.* These schemes also consider the temporal dimension, *i.e.*, deadline requirements of data flows. By postponing non-urgent flows to non-peak timeslots, these schemes can further improve the efficiency of WANs.
3. *store-and-forward schemes.* These schemes allow data to be stored in intermediate nodes to wait for free timeslots of the next links.
4. *schemes to minimize cost.* These schemes aim to minimize the cost payed to ISPs for the networking resources of inter-DC WANs.
5. *schemes to optimize welfare or revenue.* These schemes include a pricing module to encourage tenants or applications to report their demands and performance requirements truthfully. The pricing module also helps datacenter providers to optimize their economical goals.
6. *schemes for P2MP requests.* These schemes focus on Point to Multi-Points (P2MP) transfer requests, which are very popular in inter-DC WANs.
7. *schemes with dynamic IP topology.* The schemes in this category are based on the technical development of the optical layer. With the help of optical providers, datacenter providers can further control their IP layer topologies dynamically to optimize their goals.

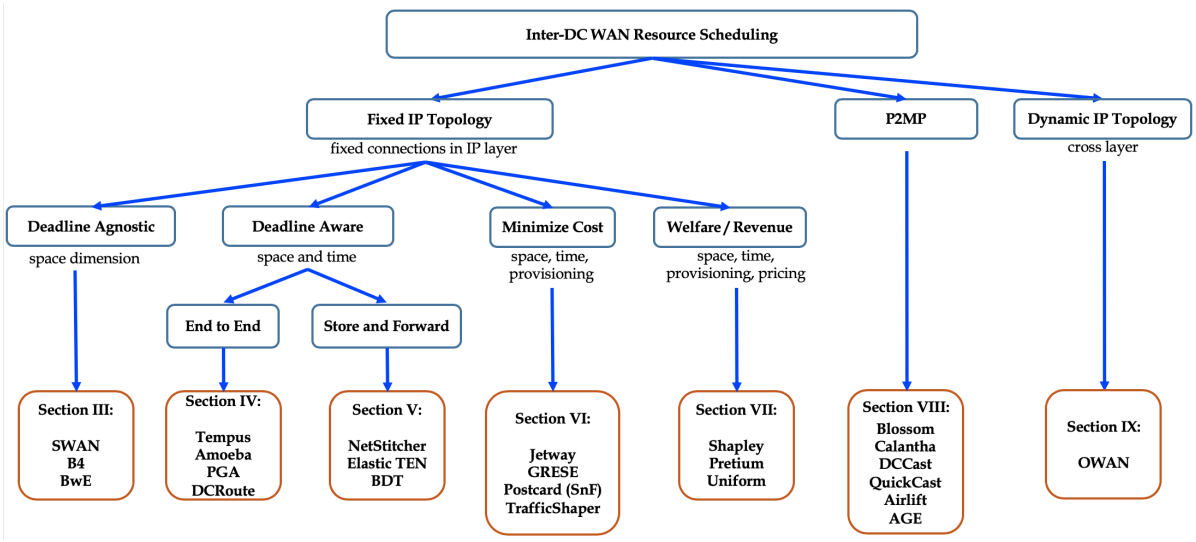

Figure 1: A summary and classification of inter-DC resource scheduling schemes proposed recently.

The remaining part of this article is organized as follows. Section 2 explains the elements in the formulation of the scheduling problem and presents a conceptional architecture of potential solutions. Section 3 discusses deadline-agnostic schemes. Section 4 introduces deadline-aware end-to-end schemes. Store-and-Forward schemes to optimize network efficiency and transfer performance are presented in Section 5. Section 6 and Section 7 describe schemes with economic incentives, wherein one section focuses on cost minimization and the other focuses on welfare/revenue maximization. Section 8 reviews the scheduling schemes for P2MP requests. The only scheme with dynamic IP topology is introduced in Section 9. In Section 10, we summarize some practical issues in developing and implementing a scheduling system, and we also describe how current schemes deal with them. In Section 11, we present a discussion on open issues and possible future directions. The article is concluded in Section 12.

2. The Problem of Resource Scheduling in Inter-DC WANs

Consider a geographically distributed datacenter with many (dozens in general) sites. In each site, there are a lot of end hosts (physical servers or virtual servers) used by different users or applications. They need to communicate with end hosts in other sites. Therefore, the datacenter operator builds an inter-DC WAN to connect these sites together by leasing links or buying bandwidth of links from ISPs.

Figure 2 illustrates such an inter-DC WAN connecting five sites of a geographically distributed datacenter. Particularly, the left part shows its internal structure within one DC site. Depending on the granularity we would like to have, the end hosts can be aggregated into groups in different levels for scheduling, such as job (hosts for the same job), user (hosts for jobs of the same user), and site (hosts in the same site). The smallest circles represent end hosts. These hosts are used by three applications, *i.e.*, App_1, App_2 and App_3, represented by medium-size circles with solid lines. Furthermore, App_1 and App_2 are run by user A, while App_3 is run by user B, illustrated by dotted circles. Here we see a hierarchy of (host, application, user). There can be hierarchies with more levels or different levels to reflect more complicated business models of datacenter operators.

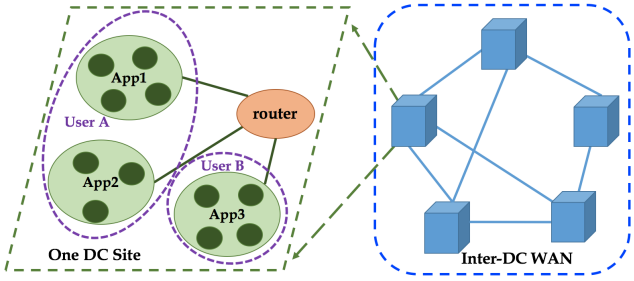

Figure 2: Sites of one datacenter connected by its inter-DC WAN

How to schedule the data transfers generated by users to make use of the networking resources of the inter-DC WAN efficiently and satisfy users' demands is an important problem for the datacenter provider. In this article, we refer to it as the resource scheduling problem. It is naturally an optimization problem. With a particular objective function in mind, under the constraints of provisioned resources, a datacenter provider solves the scheduling problem and sets values for every decision variable, *i.e.*, scheduling dimension. DC operators can have different scheduling objectives due to various business considerations. On the other hand, one DC operator might also have different scheduling dimensions, *e.g.*, routing of data packets, sending rate of each flow, provisioned bandwidth,

charges to users, and even WAN topology.

In this section, we would analyze the resource scheduling problem from several prospectives. We would summarize the types of data transfers, network models of datacenters, possible scheduling objectives and scheduling dimensions. Finally, we would present a conceptual architecture summarized from most reviewed schemes.

2.1. Traffic Features and Types of Data Transfers

Typical inter-DC data transfer sizes range from tens of terabytes to petabytes; deadlines range from milliseconds to a couple of days. Traffic flows may have different requirements on their SLAs. For example, assume site A accepts a user request and it needs to fetch some information from remote sites to produce the response for the user. The inter-DC traffic generated in this scenario is in the critical path of user experience and must be delivered without any delay. If a user is using Hadoop for computation over distributed data sources in multiple sites, the inter-DC traffic generated for remote storage access is "elastic", *i.e.*, it only requires timely delivery before the data being used.

Table 1 summarizes the properties and representative applications of three traffic classes based on their time-sensitivity. The traffic on the inter-DC WAN is a mix of these three classes. In terms of volumes, interactive traffic takes only a small portion of the overall inter-DC traffic, *e.g.*, 5%-15% [31]. While interactive traffic demand is bursty and highly diurnal, the average volume over a 5-minute window is relatively stable and can be largely predicted [32] [33]. It is also reported that background traffic is dominant in Yahoo!'s inter-DC WAN [34]. For Google's B4, we know it is running a large-scale inter-datacenter copy service whose traffic is elastic.

Obviously, interactive traffic must be transmitted immediately, so it is the least flexible for scheduling. Elastic traffic and background traffic are flexible, so they are good objects for scheduling. For example, we can delay some traffic with a farther deadline to make room for traffic with a more urgent deadline.

Scheduling systems can process these traffic in two ways, *i.e.*, aggregated flows and requests. We would explain them in the following paragraphs.

2.1.1. Aggregated Flows

All traffic flows are aggregated into flowgroups for scheduling. Each flowgroup consists of flows with the same source, destination and traffic class (priority). Flowgroup can be defined at different granularities, such as site-fg, cluster-fg, and service-fg. The scheduling system monitors the demanded rate of each flowgroup continuously. Obviously, the demand is always changing. Therefore, the scheduling algorithm runs periodically. At each timeslot, the algorithm predicts the demand of each flowgroup from historical information and allocates networking resources among flowgroups according to their demands.

2.1.2. Requests

Some scheduling systems require users to submit their requests for data transfer. There can be two kinds of requests, *i.e.*, *rate request* which demands a fixed rate from its source to its destination during a period, or *elastic request* which requires to complete a transfer of a specified size from its source to its destination. In this article, "request" is usually used to indicate an elastic request. Either explicitly or implicitly, a request is always associated with a deadline. It can also be associated with a valuation and a set of usable paths. If the user is requesting data transfer from one source to multiple destinations, the request is referred to as a *P2MP request*.

2.2. Network Model: Resource and Cost

In terms of resource, we should know whether storage resources are available at each site. If available, the store and forward mechanism can be exploited and we would have more scheduling flexibility.

In terms of cost, we should know how the DC operator buys networking resources. Generally, there can be three cases. First, the DC pays for direct links among sites, and ISPs are using the *capacity-based charging model*. It means the DC has made a fixed provisioning decision and it pays a fixed cost. The scheduling of data transfers is constrained by link capacities. Second, the DC pays for direct links among sites, and ISPs are using the *usage-based charging model*. In this case, the transmission cost is determined by the scheduling of data transfers. The cost of one link can be calculated on the 95th percentile or the maximum of bandwidth used on the link. Third, the DC pays for the uplinks and downlinks of each site, using either a capacity-based or a usage-based charging model.

Some DC operators can control the optical layer of their WANs, which enables them to conduct cross-layer scheduling. By configuring optical devices differently, they can implement different IP layer topologies.

2.3. Scheduling Objectives

The scheduling system is to allocate networking resources to all data transfers. In general, *fairness* and *efficiency* are two primary concerns of any resource allocation problem. Fairness focuses on the comparison of the payoff each individual achieves. Efficiency focuses on the total payoff achieved by the inter-DC WAN, *i.e.*, all individuals.

Note that payoff can be defined based on different metrics, then fairness or efficiency should be calculated based on the definition accordingly. For example, the payoff of one flow/request can be defined as its achieved sending rate, total bandwidth consumption on all links, completed fraction before its deadline, completion time, net profit (valuation minus transmission cost), *etc.*

A surveyed scheme may aim for a single objective or a tradeoff of multiple objectives, *e.g.*, an objective on efficiency and an objective on fairness. In this subsection, we would introduce some representative scheduling objectives.

Traffic Class	Other Names	Priority	Properties	Representative Applications
interactive traffic	highpri traffic	high	delay-sensitive, volume small and bursty, but average volume relatively stable	client-triggered D2D (dc-to-dc), transit D2C (dc-to-customer), U2D (user-to-dc)
elastic traffic	bulk transfers with deadlines	medium	deadline-aware, deadline is within several minutes or hours, large number of flows	distributed cloud computing such as MapReduce, search index synchronizing
background traffic	large transfers with soft/long deadlines or no deadline	low	bandwidth-intensive, largest volume	data replication for long-term storage

Table 1: Traffic flows on inter-DC WAN links

2.3.1. Fairness and Priority

There are several different ideas on judging whether the result achieved by a scheduling algorithm is fair, such as *Proportional Fairness*, *Max-min Fairness* or *Weighted Max-min Fairness*. Proportional fairness indicates resources should be allocated to users proportionally according to their demands. With max-min fairness, resources are allocated to all users equally until someone's demand has been satisfied or someone's allocation cannot be increased due to constraints (*e.g.*, capacity limitation on some links). Then the system would continue to allocate the remaining resources to unsatisfied active users equally. Weighted max-min fairness is similar, except that each user defines a weight to reflect the incremental value of additional allocation to it and resources would be allocated to unsatisfied active users proportionally to their weights.

Max-min fairness is regarded as more practical than proportional fairness because max-min fairness provides better isolation [35]. In case that a large transfer increases its demand suddenly, the allocation to other transfers would not change and thus their performance would not be affected.

Take weighted max-min fairness as an example. Let w_i denote the weight of user i, r_i is its demand, and R is the amount of resources for allocation. Mathematically, under weighted max-min fairness, the resources allocated to user i, denoted by b_i, is computed as follows,

$$b_i(\alpha) = \min(w_i\alpha, r_i) \tag{1}$$

wherein α is derived by solving $\sum_{\forall i} b_i(\alpha) = R$. Here, $b_i(\alpha)$ is referred to as a *bandwidth function*, and α is known as *fair share*.

Equation 1 is defined for the scenarios where users are competing for a single resource, *e.g.*, the bandwidth of a single link. It is a *link-level fairness* problem, which is easy to solve. In the problem of inter-DC WAN scheduling, data transfers are competing for the bandwidth of multiple links. Hence, it must simultaneously account for multiple potential bottlenecks, which is a *network-wide fairness* problem. Progressive water-filling algorithm is a popular way to solve network-wide fairness problems.

For scalability, one scheduling scheme may target at *hierarchical fairness*. For example, as shown in Figure 3, SWAN allocates networking resources in a max-min fair manner at the granularity of *site-fg*. Traffic flows in one site-p-fg may belong to different services, *i.e.*, *service-fgs*. The resources allocated to one site-p-fg is further assigned to each of its service-fgs in a max-min fair manner. The resource allocation from one site-p-fg to its service-fgs is a link-level fairness problem because the service-fgs are competing for a single resource, *i.e.*, the bandwidth on one path or on a set of paths (if multi-path routing) from the source site to the destination site. In contrast, the resource allocation among site-fgs is a network-wide fairness problem because they are competing for resources of multiple bottleneck links.

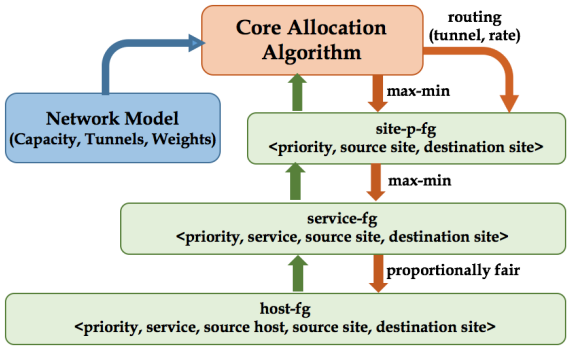

Figure 3: Hierarchical Fairness in SWAN

Weighted max-min fairness can discriminate the significance of users. The user with a larger weight would always receive more allocation. The other way to represent the significance of users is to assign each of them with a priority. In this case, users with high priorities would be satisfied before users with low priorities, while the resource allocation should be fair among users with the same priority.

2.3.2. Efficiency

There are two types of efficiency objectives, namely, *network efficiency* and *economic efficiency*.

Network efficiency objectives mean the inter-DC WAN aims to provide best-effort or guaranteed service for flows or requests as much as possible. Particularly, there are the

following possible metrics.

- maximize the network throughput

  The throughput can be calculated as the total of sending rates of all traffic flows transmitted over the inter-DC WAN. A similar metric is "utilization" of links, *i.e.*, the rate of traffic flows on the link over the link capacity. We can see that increasing utilization of all links is roughly the same as maximizing throughput.

- maximize the accept ratio or success ratio

  Some scheduling schemes have an *admission control* module and accept requests only when they can guarantee the performance for requests. In this case, it is a possible objective to maximize the ratio of accepted requests. Similarly, the success ratio can also be an objective which is calculated as the number of requests completed successfully over the number of accepted (or submitted if no admission control) requests.

- maximize the total of completed fractions

  Some schemes are deadline-aware but not deadline-guaranteed. They may consider the completed fraction before deadline as the payoff of one request. Accordingly, the efficiency objective is to maximize the total of completed fractions of all requests. In this case, the fairness objective can be to maximize the minimum completed fraction of request traffic before deadline.

- minimize the completion time

  Since requests are competing for limited resources, we cannot minimize the completion time simultaneously for all requests. For schemes that schedule all requests simultaneously, they may aim to minimize the average completion time. For schemes that serve in a First-Come-First-Serve (FCFS) manner, they can minimize the completion time for the single new request.

In terms of economic efficiency, datacenter providers have three possible objectives as follows.

- minimize transmission cost

  Under usage-based charging, an inter-DC WAN always wants to minimize the monetary cost that is paid to provision the WAN to transmit traffic of all requests. Specially, datacenter providers can exploit "free slots" to further reduce cost if p-th percentile charging is used by ISPs.

  Under capacity-based charging, since the network provision cannot change frequently, we may consider that the operation cost has been fixed. Therefore, the datacenter provider would focus only on network efficiency objectives.

- maximize social welfare

  Social welfare is defined as the total valuation of all transmitted requests minus the transmission cost. In order to achieve this objective, the scheduling system must be aware of or be able to estimate the valuation of each request. It is a natural objective for private providers which provide services only for internal applications.

- maximize revenue or profit

  Some schemes have *pricing modules* to charge users for their requests. A public inter-DC WAN always wants to maximize its revenue or profit (in case that cost is under consideration). However, in [36], the authors argue that social welfare should be the objective instead of profit even for profit-driven public providers due to the highly competitive market.

2.4. Scheduling Dimensions

In order to achieve their objectives, datacenter providers need to control or affect data packets and users in various dimensions. Basically, they include but not limited to the following dimensions.

- Space.

  It refers to the routing paths of one flow/request and also sending rates on these paths. One scheduling scheme may prefer single-path routing to avoid re-ordering, or it may prefer multi-path routing for network efficiency. The usable paths can be given as an input to reduce the computational complexity of the scheduling problem.

- Time.

  It refers to the timeslots used for the transmission of one elastic request and also sending rates in each timeslot (may also on each of multiple paths). Only elastic requests can be scheduled across the temporal dimension.

- Storage.

  If in-network storage nodes are available, the store-and-forward mechanism can be used to further improve the usage of networking resources. For example, without store-and-forward, a path can be used for transmission only when all links on the path are available. While with store-and-forward, data packets can be transmitted on a link once the link is idle and stored there until idle timeslots of the next hop.

- Provisioning.

  If ISPs are using usage-based charging, datacenter operators can minimize their operation costs paid to ISPs by controlling the provisioning of each link. It is often used together with other scheduling dimensions such as space and time.

- Pricing.

  A datacenter operator needs to charge its users to receive monetary revenue or at least cover its cost. A proper pricing mechanism can be used to achieve various goals. For example, resources in light-loaded timeslots can be priced less to encourage users avoiding busy periods.

- Cross-Layer.

  If the optical layer is intelligent with the support of modern Reconfigurable Optical Add-Drop Multiplexer (ROADMs), datacenter providers can construct IP networks dynamically to adapt to the demands of users.

2.5. Conceptional Architecture of Solutions

We summarize main surveyed schemes in Table 2 and present the main features in their problem formulations, such as network model, scheduling objectives, scheduling dimensions, *etc.* The column of "policy" presents their scheduling policies, wherein "periodical" means that the scheduling decision is made for all active requests at the beginning of each timeslot and "FCFS" indicates that the scheduling decision is made for a request when it arrives and the resource is allocated on a first-come-first-serve basis. The next column "type of transfer" presents how the scheduling system gets the information on users' demand. Some scheduling schemes monitor the flows in the WANs and make scheduling decisions based on their measurements, which is denoted as "flow" in the table; "request" indicates that the scheduling system requires one user to specify the information on its requests explicitly, *e.g.*, size, deadline, or valuation. The concept of "flow" and "request" is introduced in Section 2.1. In some schemes, a user also specifies a set of available paths for each transfer, and the scheduling scheme only needs to select some paths in the set and split the traffic demand among these selected paths. It can greatly reduce the problem complexity. Whether the paths are given is indicated in the column "paths given". The next two columns introduce the network model, *i.e.*, whether the amount of usable resource and the corresponding cost are fixed, and whether in-network storage is available. The concept of network model is introduced in Section 2.2. The column "objectives or payoff" presents the scheduling objective, which is introduced in detail in Section 2.3. The last three columns characterize the possible cases of scheduling solutions, *e.g.*, whether the solution can select multiple paths for a single data transfer, and whether admission control is conducted so that some requests would be rejected when the resources are insufficient. The possible scheduling dimensions are introduced in Section 2.4.

Figure 4 shows the conceptional architecture that we summarize from recent solutions to the inter-DC WAN scheduling problem. It mainly includes the following modules.

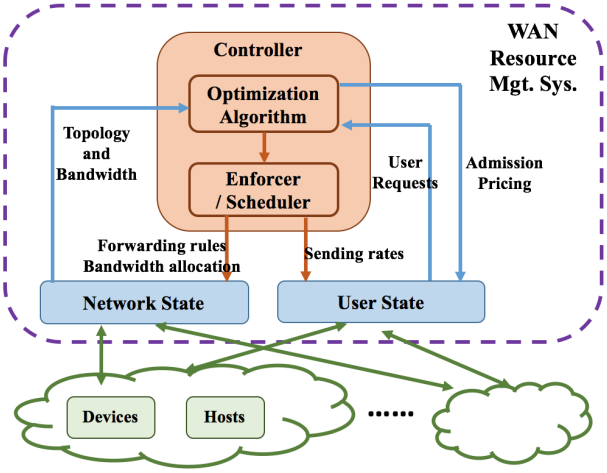

Figure 4: Conceptional Architecture of Scheduling Systems.

- *User State.* This module monitors the behaviors of users (*e.g.*, hosts, services, and requests) and provides information about current user requests to the Optimization Algorithm. In some solutions, it is also responsible for predicting users' future behavior. After the Optimization Algorithm computes the optimal allocation, the User State module may receive decisions such as admission control (accept or reject), pricing, and sending rate of each user. Then it has to make sure each user cannot exceed its assigned sending rate. Please note the User State module is often implemented distributedly. For example, each service can have its own "user state" module, *i.e., service broker*.

- *Network State.* This module learns the state of the network continuously and reports the up-to-date global view of the network topology, available resources and traffic demands to the Optimization Algorithm. In some solutions, it is also responsible for predicting the dynamics of the network. After the Optimization Algorithm computes the optimal allocation, the Network State module may receive requests to configure rate limiting and forwarding rules on network devices. Then it has to make sure these configurations take effect in all sites of the datacenter. Similar to User State, Network State module is often implemented distributedly. For example, each site can have its own "network state" module, *i.e., site agent*.

- *Optimization Algorithm.* Given the information on networks and users, considering various goals, this module computes the optimal scheduling decisions, including but not limited to, admission decisions, pricing decisions, sending rates, routing paths and allocated bandwidth for flows or requests. The computed results are sent to Enforcer for the enforcement in the datacenter.

Table 2: Surveyed Schemes and Their Problem Formulation.

Schemes	policy	Input Information		Network Model		Objectives or Payoff	Solution		
		type of transfer	paths given?	provisioning	storage		multi-path?	admission control?	scheduling dimensions
SWAN [32]	periodical	flow	Y	fixed	N/A	throughput fairness	Y	N	space
B4 [37]	periodical	flow	Y	fixed	N/A	throughput fairness	Y	N	space
BwE [35]	periodical	flow	Y	fixed	N/A	throughput fairness	Y	N	space
Tempus [33]	periodical	request	Y	fixed	N/A	completed fraction	Y	N	space, time
Amoeba [31]	FCFS	request	Y	fixed	N/A	deadline-guaranteed	Y	Y	space, time
PGA [38]	FCFS	request	Y	fixed	N/A	deadline-guaranteed	Y	Y	space, time
DCRoute [39]	FCFS	request	N	fixed	N/A	deadline-guaranteed	N	Y	space, time
NetStitcher [40]	FCFS	request	N	fixed	Y	completion time	Y(chunk)	N	space, time, storage
Elastic TEN [41]	periodical	request	N	fixed	Y	congestion	Y	N	space, time, storage
Elastic TEN [42]	periodical	request	N	fixed	Y	completed fraction	Y	N	space, time, storage
BDT(HEU) [43]	FCFS	request	N	fixed	unilimited	throughput	Y(chunk)	Y	space, time, storage
Jetway [44]	periodical	request (rate)	N	usage-based	N/A	cost	Y	N	space, time, provisioning
GRESE [45]	periodical	request	N/A	usage-based	N/A	cost	N/A	N	space, time, provisioning
Postcard [46]	periodical	request	N	usage-based	Y	cost	Y	N	space, time, storage, provisioning
TrafficShaper [47]	periodical	request	N/A	usage (up, downlinks)	N/A	cost	N/A	N	space, time, provisioning
Shapley [48]	FCFS	request (valuation)	Y	usage-based	N/A	welfare	N	auction	space, time, provisioning, pricing
Pretium [36]	both	request (valuation)	Y	usage-based	N/A	welfare	Y	auction	space, time, provisioning, pricing
Uniform [49]	N/A	request (valuation)	N/A	fixed	N/A	revenue	N/A	auction	pricing
Blossom [50]	periodical	request (P2MP)	Y	fixed	N/A	throughput	Y	N	space
DCCast [51]	FCFS	request (P2MP)	N	fixed	N/A	completion time	N	N	space, time
Airlift [52]	periodical	request (P2MP)	N	fixed	N/A	throughput	Y	N	space, time
AGE [53]	N/A	request (P2MP)	N	fixed	N/A	the number of receivers	N/A	N	space, time
OWAN [54]	periodical	request	N	N/A	N/A	throughput	Y	N	crossN/Alayer, space, time

- *Enforcer or Scheduler*. This module is responsible for the enforcement of decisions in the datacenter. For example, we may need to configure tunnels to implement routing paths, update forwarding states of switches, and configure rate limits on hosts and devices. Moreover, frequently re-configuring the network can cause transient congestion, even packet loss, that can heavily hurt latency-sensitive traffic. This Enforcer module needs to orchestrate all these configuration activities to avoid such situations and send configuration requests to service brokers and network agents.

Since the state of users and networks is dynamic, obviously the scheduling system should run online to adapt to the changes of network and user requests. It can be request-driven or periodical.

Request-driven. The scheduling decision is recomputed when one new transfer request comes. If the new request can only use the leftover resources from existing requests, the scheduling is referred to as *FCFS*. FCFS can retain the scheduling of existing requests, thus their performance would not be affected by later requests. It also reduces computational complexity, but may not be able to reach the optimal solution.

Periodical. Such scheduling systems discretize time into timeslots and recompute the decision at the beginning of each timeslot. Then the bandwidth allocation is fixed within a timeslot but can vary across different timeslots. The length of timeslot must be set properly to achieve a reasonable tradeoff between performance and overhead. If it is too long, the algorithm cannot adapt to network dynamics quickly and the scheduling performance would degrade. If it is too short, the algorithm would run too frequently which incurs a lot of costs on computation and configuration update. Moreover, since the network needs time to converge after a scheduling decision is implemented, the possible timeslot length must be longer than the convergence time. In general, it is much coarser than TCP time-scale, (*e.g.*, minutes *vs.* RTT of $100ms$).

Although the scheduling can be formulated as optimization problems intuitively, how to solve the formulated problems is often a challenge, partly because there are a lot of decision variables and constraints. Therefore, researchers have to design techniques and heuristics to solve them efficiently. Further, we must consider various practical design and implementation issues, such as accuracy of predictions, untrusted users, and unexpected failures.

3. Spatial Scheduling: Maximize Utilization with Fairness

DC providers always would like to increase inter-DC link utilization to extract more revenue or operate at a lower cost. As we have explained, link utilization in the traditional Internet cannot be too high because it can result in serious packet loss which is deadly for performance-sensitive traffic flows. In an inter-DC WAN, such a problem can be solved by providing services with different priorities, *i.e.*, high priorities for sensitive traffic and low priorities for other flows. Packets with high priority are always processed immediately in a preemptive manner. In this way, DC providers can push its link utilization to a high value by scheduling the sending rates and routing paths of traffic flows, without the need to worry about network performance.

Based on this idea, researchers from Microsoft and Google proposed three scheduling schemes, namely, SWAN, B4 and BwE. They aggregate traffic flows into FGs (flow groups) at specified granularities, *e.g.*, source-destination site pair. Then they take as inputs the demand of each FG, the capacity of each link in the inter-DC WAN, and the usable paths (tunnels) for each FG. The scheduling algorithms of these schemes are to determine the rate of each FG that can be transmitted over the WAN, and the fraction of FG traffic to be forwarded along each usable path, *i.e*, split ratio.

3.1. SWAN: Iterative MCF for Max-min Fairness [32]

SWAN allocates resources by invoking its scheduling algorithm separately for classes in priority order. After a class is allocated, its allocation is removed from the remaining link capacity. Therefore, the high priority traffic flows are satisfied before the low priority traffic flows get their allocations.

For traffic flows within a priority class, SWAN sets two parameters, α (> 1) and U, and computes the solution with network-wide max-min fairness iteratively.

- In each iteration step k, SWAN solves a MCF (multi-commodity flow) problem with the objective function to maximize total throughput of all flows and constraints that flows are allocated rates in the range $[\alpha^{k-1}U, \alpha^k U]$ but no more than their demands. Solving this MCF problem formulated by SWAN would give a rate allocation and a routing decision for each flow.

- A flow's allocation is frozen at step k when it is allocated its full demand or it receives a rate smaller than $\alpha^k U$ due to the capacity constraints. The frozen flows have got their final allocations and would not take part in later steps.

In terms of efficiency, the above algorithm can maximize the total throughput of all flows by solving MCF in each step. In terms of fairness, it achieves approximated max-min fairness by starting the iteration from small allocation and freezing the flows constrained by link capacities at each step. It can be proved to be an α-approximation algorithm.

The two parameters, $\alpha > 1$ and $U > 0$, are used to tune the tradeoff between fairness and runtime. The algorithm runs $T = \lceil \log_\alpha \frac{max(d_i)}{U} \rceil$ steps (d_i is the demand of fg_i),

therefore its runtime decreases if we use a larger α and a larger U. On the other hand, the resulting allocation is more close to the max-min fair solution as α is more close to 1. The authors state its allocation is highly fair and takes less than a second combined for all priorities.

3.2. B4: Progressive Water-filling for Weighted Max-min Fairness [37] [55]

In Sigcomm 2013, where SWAN was published, researchers from Google also published a paper on boosting the utilization of inter-datacenter networks by centrally controlling traffic flows. Their inter-DC WAN is called as B4 [37]. Different from many other proposals, B4 had been in deployment for three years when the work was published. B4 consists of more than ten sites, and over 90% of Google's internal application traffic runs across B4.

Instead of specifying multiple priority classes, B4 specifies a weight for each application, which means linear bandwidth functions are used. It then aggregates bandwidth functions for applications into piece-wise bandwidth functions for site-fgs and delivers weighted max-min fair allocations to site-fgs.

B4 also assumes that a set of usable paths has been known for each FG. It allows DC providers to specify a cost for every edge and assumes that the shortest path with available bandwidth is always preferred.

B4 uses the idea of *progressive filling* to solve its *network-wide* weighted max-min fair allocation problem. The algorithm starts with the fair share $\alpha = 0$ and increases α gradually. For a given α, the allocated sending rate of each FG can be derived according to its bandwidth function. At the beginning, all FGs choose the shortest path in their own path sets. As the rate and the path of each FG have been known, the traffic rate on every link can be computed as a function of α.

With the increase of α, the traffic rate on one link would hit its capacity limit, and the link becomes a bottleneck link. Since there is no more resource on this link, all FGs transmitted on this link should be frozen on their corresponding paths, and each of them needs to choose a new shortest path from remaining unfrozen paths for more transmission rate. If all paths of one FG have crossed bottleneck links, the FG has reached its final allocated sending rates on all usable paths. Other FGs can continue to receive more allocations on their paths without bottlenecks. The algorithm continues until all FGs have frozen all their paths. We can see that B4 just pushes traffic flows to inter-DC links as much as possible, instead of maximizing throughput by solving maximization problems directly as in SWAN.

In order to enforce the scheduling on hosts and switches, the solution computed by B4 is presented in the format of a sending rate and a split ratio for each FG. The desired split ratios may not be implementable on switches due to the hardware capability limitation, therefore the resulting split ratios have to be approximated. For example, a switch that only supports a splitting granularity of 0.5 has to approximate a resulting ratio $(0.3, 0.7)$ to either $(0, 1)$ or $(0.5, 0.5)$. Such an approximation can affect the efficiency and fairness of the final solution. BwE in the next subsection can take the approximated ratios as input and compute a new sending rate for each FG under the implementable ratios.

3.3. BwE: Hierarchical Multi-Path Fair Allocation [35]

In Sigcomm 2015, researchers from Google presented a paper on BwE (bandwidth enforcer) and disclosed more details on its inter-DC WAN. It also uses the idea of the progressive filling algorithm to find out the allocation with global max-min fairness. But it enables more complex features, thus it can be regarded as a complement and extension to B4.

Hierarchical Allocation with Policies at Two Levels. As shown in Figure 5, BwE implements a hierarchical host-based allocation architecture. Hierarchical architecture is used in many scheduling schemes, but BwE has some unique features, which might be due to Google's business considerations.

BwE considers inter-cluster (intra-site) links may also be bottlenecks, so the allocation should be done at the cluster-level instead of the site-level topology. Moreover, BwE requires that the allocation should respect policies at both site-fg and user-fg levels at the same time. The bandwidth function of one cluster-fg is derived as the sum of predefined bandwidth function of its child user-fgs, but site-fg's bandwidth function is defined independently instead of summing up cluster-fgs.

BwE designs a *Bandwidth Function Transformation* algorithm to transform cluster-fgs' bandwidth function B_{cf_i} to effective bandwidth function $B^e_{cf_i}$, which is a new cluster-fg's bandwidth function respecting site-fg's priority configurations. Then the allocation problem is solved at the level of cluster-fgs using $B^e_{cf_i}$ by the progressive water-filling algorithm. BwE considers jobs of a user have the same priority and tasks of a job also have the same priority. Therefore, at these two levels, BwE just splits the allocation to the parent node among its children based on their estimated demands in a max-min fair manner.

Complex Bandwidth Functions for Priority Classes. BwE allows operators to specify complex (non-linear) bandwidth functions, while B4 only allows linear functions for applications. For example, BwE can define two regions of fair share, *i.e.*, Guaranteed $(0-2)$ and Best-Effort $(2-\infty)$. Assume a FG fg_1 has a demand of 20Gbps. It requires a guaranteed bandwidth of 10Gbps with a weight of 10, and upon reaching such an allocation, it still requires best-effort bandwidth with a weight of 10. Then fg_1's bandwidth function can be specified as

$$B_1(\alpha) = \begin{cases} \min(10\alpha, 10) & \alpha \in [0, 2] \\ \min(10 + 10 \times (\alpha - 2), 20) & \alpha \in [2, \infty] \end{cases}$$

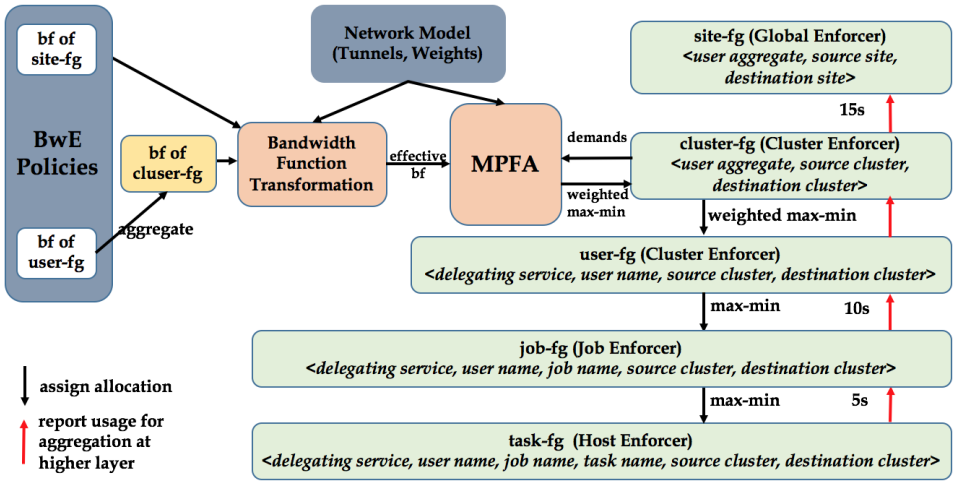

Figure 5: BwE: hierarchical fair allocation and enforcement

In this way, BwE realizes the same goal as SWAN, *i.e.*, allocating bandwidth in strict precedence across different priority levels, and allocating bandwidth fairly within a priority level. The flexibility of bandwidth functions is also useful for applications such as video streaming with adaptive bitrates. For these applications, only sufficient additional bandwidth to enable a higher bitrate is useful.

MultiPath Fair Allocation (MPFA). As shown in Figure 5, BwE takes (tunnels, weights) as input, which includes a set of tunnels for each site-pair and each cluster-pair, and also the split ratios among tunnels in each set, *i.e.*, weights. BwE also uses the idea of the progressive filling algorithm, but data packets of each FG are transmitted over all usable paths at the same time with the given split ratio. As α increases, there are more and more traffic volumes on every usable path. Once there is a saturated link on any path in its path set, the FG will be frozen on all its paths and it has got the final allocation. BwE outputs a sending rate for each FG as the solution.

BwE can improve network utilization compared to the approximated solution produced by B4. Taking the approximated split ratios produced by B4 as input can reduce the complexity of BwE's scheduling algorithm.

4. Spatial-Temporal Scheduling: End-to-End

Given available networking resources and the demands of flows, SWAN, B4, and BwE determine the sending rate and routing paths for each flow in a single timeslot. They run their scheduling algorithms periodically and guarantee that each flow can get a fair allocation in the timeslot under scheduling. However, they cannot guarantee the performance of flows that span multiple timeslots.

Assume a large file or a dataset is required to be sent from one site to the other site before a deadline. Spatial scheduling schemes cannot guarantee its success directly. They have to transform the original deadline requirement to a more stringent requirement, *i.e.*, a bandwidth demand in every timeslot before its deadline. Obviously, it is far from optimal because the elasticity of the data flow is not exploited. Intuitively, we would like to allocate more bandwidth to it in light-loaded timeslots, so that more bandwidth in heavy-loaded timeslots can be used by other urgent transfers. By shaping peaks and troughs, scheduling in the temporal dimension can improve the network efficiency.

We see that elastic transfers should be scheduled in both spatial and temporal dimensions. We refer the schemes that consider both dimensions as *Spatial-Temporal Scheduling*. They exploit temporal features of delay-tolerant transfers and conduct optimization on all timeslots during durations of transfers. In contrast, spatial scheduling schemes only consider a single timeslot in each running.

Spatial-temporal scheduling schemes can achieve two objectives: improve or guarantee deadlines of transfers, and maximize the utilization of networks by exploiting flexibility in both of two dimensions. Some of these schemes are also referred to as *deadline-aware* or *deadline-guaranteed*. In contrast, spatial scheduling algorithms are always *deadline-agnostic*.

Deadline-aware schemes should have a richer service-network interface to allow users to express more details of their transfer requests, such as the total bytes and the deadline of each transfer. Moreover, it is possible that a request is rejected because the system cannot allocate sufficient resources to guarantee the request is satisfied.

Different deadline-aware schemes specify transfer requests in a similar way as follows. One request is defined by a tuple $(b_i, d_i, D_i, s_i, t_i, P_i)$ where b_i is the begin time of the transfer request; d_i is the deadline; D_i is the demand (total bytes) of the request; s_i is the source node of the data flows of the request; t_i is the target node; P_i is the set of admissible paths from s_i to t_i. In some schemes, users can submit one request before the request can be transmitted, then the tuple would have one more element, a_i, the aware time in which the system becomes aware of

the request. Please note knowing the request earlier can help the system to find a better solution. In case that the request is with a soft deadline, a value function replaces the strict deadline d_i.

Most schemes use "transfer" and "request" interchangeably. Some schemes, such as Amoeba, notice some transfers are co-related, *e.g.*, shuffle transfers from several mappers to a reducer in MapReduce. These transfers are useful only after all transfers are completed. Amoeba allows tenants to specify such a demand by submitting a request $R = \{T_1, ..., T_n\}$, where T_is are transfers with identical deadline requirements.

Given the network state (topology and links' capacity) and user requests, a deadline-aware scheme runs a logically centralized algorithm to determine $f_{i,p,t}$, *i.e.*, the amount of bandwidth allocated for request i on path $p \in P_i$ in timeslot t. Please note although the scheduling algorithm runs once a timeslot, each running of it considers all future timeslots within a sliding time window.

4.1. Tempus: Efficiency and Fairness in Terms of Completed Fractions [33]

Tempus is published by researchers from Microsoft in Sigcomm 2014. It focuses on the fraction of completed volume before the deadline of each transfer, say γ_i. Tempus first aims to maximize the smallest fraction, which corresponds to a fairness objective. After it is maximized, some links might remain under-utilized. Tempus then tries to maximize the sum of completed fractions of all requests ($U = \max \sum_{\forall i} \gamma_i$), which corresponds to an efficiency objective. We can see that it does not guarantee the completion of any transfers.

Unfortunately, it is very challenging to solve the above problem because of millions of variables and constraints. Assume the smallest completed fraction is γ, and assume the sum of completed fractions of all requests is at least U. The problem to find a feasible solution $f_{i,p,t}$ to satisfy the given γ and U, denoted as $LP(\gamma, U)$, is in fact a packing-covering problem, where the edge capacities are packing constraints and the request satisfactions are covering constraints. This packing-covering problem can be solved using Young's algorithm [56], which is an iterative solver.

As an iterative solver, Young's method has a very useful property in this scenario, *i.e.*, the solution for the current timeslot is a feasible searching point for the new scheduling problem in the next timeslot. In the next timeslot τ, the allocation for old requests are retained; Tempus conducts a water-filling process, *i.e.*, gradually increase γ from 0, to allocate resources for new coming requests as long as $LP_\tau(\gamma, U)$ is feasible. During this process, U is kept as same as the previous timeslot. Once the water filling process stops, γ is then fixed and Tempus continues to search for the largest feasible U.

Due to the above property, the computation complexity is reduced, which makes Tempus practical in real-world networks. The authors state that Tempus is efficient in terms of running time, memory usage, and updating overhead.

4.2. Amoeba: Deadline-guaranteed and FCFS [31]

Tempus is deadline-aware but not deadline-guaranteed. It maximizes the minimum completed faction of all transfers. However, for many applications, partial data transfer is useless and the deadline must be met for the transfer to be useful. Noticing it, Amoeba sets its objective as to maximize the number of transfers that are completed before their respective deadlines. As a deadline-guaranteed scheme, Amoeba has to reject some requests when the demand of requests is more than the network can accommodate. This is done by the admission control module, which does not appear in SWAN, B4 and Tempus.

Amoeba does not think that fairness is necessary among all requests. It serves requests in a FCFS manner. When a new request comes, re-scheduling all admitted requests may yield optimal result but the algorithm would be time consuming and cannot scale. Amoeba starts from the old solution to search for a new solution using heuristics. In this way, Amoeba balances its scalability and optimality and also makes it more practical.

Time-Expansion Graph Approach. Time-Expansion Graph Approach is used by Amoeba to transform its network layer topology to a temporally expanded flow graph to incorporate two dimensions, *i.e.*, time and space, in one graph. This approach is popularly used by multiple spatial-temporal schemes. Figure 6 shows the example presented in [31].

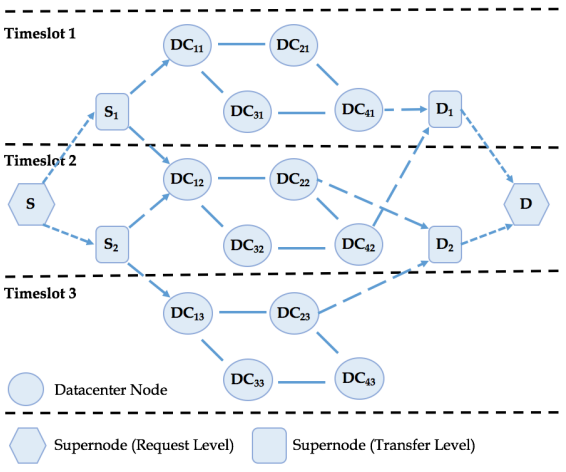

Figure 6: An example of a temporally expanded flow graph (adapted from [31]).

Basically, in this approach, the network topology, together with link capacities, is duplicated multiple copies, each copy for one timeslot. For each transfer T_i, Amoeba adds a pair of supernodes S_i and D_i and connects S_i (D_i) to the source (destination) DCs in the timeslots within T_i's possible transmission period. For example, Figure 6 shows

a request R that includes two transfers, wherein T_1 is from DC_1 to DC_4 within timeslots $[1, 2]$ and T_2 is from DC_1 to DC_2 within timeslots $[2, 3]$.

Opportunistic Rescheduling with Admission Control. When a new request comes, Amoeba first computes whether the new request R can be accommodated without changing the schedules of accepted requests. In order to answer this question, Amoeba assigns weights to the edges from the supernode S_i to the source DCs in different timeslots, and the earlier timeslot has smaller weight. Then Amoeba solves a *min-cost flow problem* (MCFP) on the temporally expanded flow graph. Since earlier timeslots are always preferred due to their lower cost, the solution, which is with minimum cost, gives the earliest timeslot, say t', in which R can be completed.

If t' is later than R's deadline, accepted requests must be rescheduled to try the best to accommodate R. In order to avoid the complexity of solving optimization problems directly, Amoeba depends on two heuristics to select requests for rescheduling. In the first heuristic, Amoeba selects all accepted requests that use links on R's paths within R's starting time and deadline, and it then tries to move their volumes to timeslots before and after R's transmission time window. In the second heuristic, it selects existing requests who have a lot of traffic that goes through R's bottleneck link or have a small overlapping time period with R. The affected traffic of selected requests and R are scheduled together using MCFP to find the shortest completion time.

After trying to reschedule transfers using two heuristics, if the completion time is still later than R's deadline, R should be rejected.

4.3. PGA: Mixture of Hard and Soft Deadlines [38]

Tempus aims to maximize the sum of completed fractions of all requests, and it does not guarantee the completion of transfers. It means Tempus is assuming that all requests are with soft deadlines. On the other hand, Amoeba would reject an arriving request if it cannot find sufficient resources to complete the transfer. It means Amoeba is assuming that all requests are with hard deadlines.

In [38], the authors notice that inter-DC data transfer requests are with a mixture of hard and soft deadlines. They argue that the completion of transfers with hard deadlines should be guaranteed and transfers with soft deadlines should be delivered in a best-effort manner to improve network utilization. Then they propose a revenue model, wherein the datacenter provider can get positive revenue from each deadline-met byte and it needs to pay penalty (negative revenue) for each deadline-missed byte. Two parameters, the revenue coefficient and the penalty coefficient, are specified for each request. The penalty parameter is set to be a very large value for a transfer request with a hard deadline.

With the revenue model, the scheduling problem is formulated as an optimization problem aiming to maximize the provider's revenue. Based on the primal-dual method, the authors design an online algorithm to efficiently make a resource scheduling decision for each transfer at its arrival. Their theoretical analysis proves that the achieved value by the algorithm is at least $(e-1)/e$ of the optimal offline solution with complete knowledge of future requests at a cost of little link capacity augmentation.

4.4. DCRoute: Single-Path Routing and As Late As Possible [39]

DCRoute also serves requests in a FCFS manner with admission control, and it solves the scheduling problem based on heuristics. But it is different from Amoeba in several ways.

First, DCRoute argues that transmitting a single transfer on multiple paths is likely to cause packet reordering and degrade its TCP throughput. Therefore, DCRoute only chooses one path for each transfer, and this routing decision would not be changed during transmission.

Second, DCRoute totally depends on heuristics to find an approximate solution. Please note even Amoeba, which also exploits heuristics, needs to solve MCFP for each new request. The authors argue that a scheduling decision should be made in a time as short as possible, and DCRoute is stated to be at least 200 times faster than techniques based on linear programming.

When a new request arrives, its working procedure is as follows.

Path Selection (spatial scheduling). DCRoute retains the scheduling of existing requests as before, and it selects a most preferable path for the new request. The path selection is based on three metrics: 1) the sum of load (including existing requests) on all links over the selected path during its transmission period. The path with a smaller total load is preferred. 2) the bottleneck load, *i.e.*, the load on the link with the maximum load given the path is selected. The path with a smaller bottleneck load is preferred. 3) the path with a smaller number of hops is preferred. In this way, DCRoute tries to spread traffic more evenly on all links, which is thought to be beneficial to accommodate more future requests.

Admission Control and Temporal Scheduling. After the path is determined, it tries to allocate the demand to the timeslots as late as possible (ALAP). In this way, the new request would use resources only when it is necessary to meet its deadline, and the closest timeslots can be used for requests with the most urgent deadlines. If the resource of the path during its transmission period is insufficient, the request would be rejected.

Pull Back. Obviously, the ALAP scheduling can cause that the current timeslot is underutilized. In order to fully utilize the current timeslot, DCRoute pulls the traffic in the closest timeslots to the current timeslot as much as possible.

Push Forward. After the pullback, some later resources are released, and it is possible that the paths for

some requests become available in later timeslots than their current scheduling. According to the ALAP principle, the traffic of these requests is pushed forward to later timeslots.

5. Spatial-Temporal Scheduling: Store and Forward

No matter deadline-agnostic or deadline-aware, all schemes mentioned before are direct *End-to-End* transfers, which means traffic flows are transmitted from sources to destinations without longtime in-network storage. In other words, all nodes and links on the path must be available for the path be used by a transfer. But what if there is no overlapping available hour for all links on the path?

Let us consider the following scenario. A datacenter site A on the East Coast has free capacity during its early morning hours (*e.g.*, $8-11am$ GMT), and it wants to maximize the volume of data that it can backup to another site B within 24 hours. If B is also free during this period, we can transmit data between these two sites using a direct e2e transfer during these non-peak hours. But what if B is located on the West Coast of USA and its non-peak hours are $12pm-3pm$ GMT? A and B do not have overlapping non-peak hours. All schemes mentioned before cannot help datacenter operators.

The reason that those schemes cannot work is that all of them only consider end-to-end transmissions. If the DC operator also runs a site C in the Central USA, which is free from $10am$ to $1pm$, the *Store and Forward* (SnF) mechanism would ask A to send data to C during $10-11am$, and C relays the data to B during $12pm-1pm$. In this way, the DC operator can exploit the left-over bandwidth to complete the transfer from A to B, with the extra cost of C's uploading and downloading bandwidth and also storage. Furthermore, if there are other sites which can serve as relay nodes, the volume of data transferred from A to B can be further increased.

SnF has been widely used in Delay Tolerant Networks (DTN) [57] wherein it is impossible or difficult to establish end-to-end connections between sources and destinations due to highly unpredictable environments and moving destinations. It can be viewed as a temporal-expansion of routing. In [58] [59], the authors propose a novel business model of Internet Post Offices (IPOs), *i.e.*, collecting delay-tolerant traffic from users and scheduling them efficiently with the support of network attached storage nodes. The nodes can be deployed by a CDN, a federation of access ISPs, or a transit provider. If the network attached storage nodes are widely deployed with proper pricing schemes, SnF could become a popular way to transmit all delay-tolerant traffic flows in the Internet.

The inter-DC scheduling schemes based on SnF generally formulate the scheduling problem as follows. A large file to be transmitted from a source site is split into pieces or chunks, and each individual piece is scheduled across space and time to use all available networking and storage resources of relay nodes to optimize a particular goal. There are also some SnF-based schemes that are designed to schedule requests instead of chunks of a single file. We do not see any significant difference between a chunk-based model and a request-based model.

Obviously, the throughput enabled by the SnF mechanism can only be used for non-real-time applications. In the following subsections, we briefly summarize three representative SnF scheduling schemes.

5.1. NetStitcher: MFP on Time-Expansion Graph [40]

Researchers from Telefonica (which operates a CDN) noticed that the demand of an area followed strong diurnal patterns with high peak to valley ratios, wasting network resources during non-peak hours. In [40], the authors propose NetStitcher, a scheme based on the SnF mechanism, that uses left-over bandwidth at different sites (relay nodes) to minimize the completion time for a non-real-time transfer request.

Minimizing Completion Time. NetStitcher works in a FCFS manner, and each new request only uses the residual resources left by old requests to minimize its transfer time. In order to find the minimum completion time, NetStitcher gradually increases the number of timeslots used for transmission and computes the maximum flow volume that can be sent during these timeslots. Once the maximum flow volume is larger than the demand, the minimum transfer time (number of timeslots) is found. Therefore, the original problem is transformed into a series of MFPs on a time-expansion graph.

Incorporate Storage, UpLinks and DownLinks. As a SnF scheme, the time-expansion graph of NetStitcher should incorporate more factors than Amoeba. The capacity of a link connecting two different nodes in a timeslot is set to be the available bandwidth of the overlay link in the timeslot, and it means the link is carrying data across different geographical locations. This is the same for all time-expansion graphs. Besides it, in NetStitcher, the capacity of a link connecting the same node in two consequent timeslots is set to be the storage size of the node, and it means the node is carrying data across time. Moreover, each relay node has limited downlink bandwidth and uplink bandwidth. In order to incorporate these possible bottlenecks, each node is split into three parts, *i.e.*, the front part is used for modeling the downlink bottleneck, the middle part models the storage capacity, whereas the back part models the uplink bottleneck. The sender node has only a back part and the receiver has only a front part.

After the time-expansion graph is constructed, the MFP problem for scheduling a request can be solved using the *Ford-Fulkerson* (FF) method, which is a well-known greedy algorithm for MFP.

5.2. Elastic TEN: Elastic Time-Expansion Network [41] [42]

Both Amoeba and NetStitcher duplicate the original network for every single timeslot, which unnecessarily results in a very large time-expansion graph and expensive computational cost. In [41] and [42], the authors propose to expand the original network elastically, *i.e.*, one duplication representing a time window including a different number of timeslots.

In [41], the authors argue that the congestion of all links should be minimized as much as possible instead of only minimizing the congestion of the bottleneck. Therefore, they define the concept of "lexicographical minimization", which means the congestion of the bottleneck should be minimized first, then the congestion of the next bottleneck should be minimized, and so on. In [42], the authors aim to provide max-min fairness among requests in terms of completed fractions. Although these two works are proposed for different objectives, their formulated problems can be solved by similar algorithms.

Determine Elastic Time Windows. They first aggregate consecutive timeslots with the same amount of networking and storage resources into a time window. Second, they further split these time windows by the start time and deadline of each request. In this way, within one time window, both the demand and resource are constants, and then the scheduling on the basis of time window instead of timeslot would not lose any optimality. After elastic time windows are determined, it is easy to construct an elastic time-expanded graph.

Iterative Min-Max (Max-Min) LP. Now the scheduling can be solved using an iterative Min-Max LP [41] or Max-Min LP [42]. Take the Min-Max for congestion as an example. During the iterations, links are labeled as "unmin" or "min", wherein "min" means the congestion of this link has been minimized and cannot be reduced by any means.

Initially, it has no idea on the congestion of any link, and all links are labeled as "unmin". In each iteration, it minimizes the maximum congestion of all unmin links by solving a minimization problem. Then, the link with maximum congestion is regarded as a "min" link, and it would not put more traffic on this link in the following iterations according to the requirement of min-max. The iteration repeats until all links are minimized, *i.e.*, labeled with "min".

Disjoint Paths to Reduce Complexity. In [42], the authors further design a method to reduce the time complexity by putting a limit on the number of usable paths for each request. Previously, each request can use an unlimited number of paths (a path is a series of network links and storage nodes) to deliver its traffic. The key here is how to select the K paths for each request. The authors devise a k-disjoint shortest paths algorithm and demonstrate that it can produce better results than k-shortest paths.

5.3. BDT: Scheduling Multiple Requests with Unlimited Storage [43]

In [43], the authors propose BDT, which formulates the scheduling problem in a similar way to NetStitcher, *i.e.*, splitting into chunks and transmitting each chunk individually in a store-and-forward manner. BDT tries to maximize the effective throughput of all successfully completed requests.

The authors design an algorithm, named HEU, to make a scheduling decision for a new arriving request. Assume a path r with h_r hops (links) is a possible path for the request. It splits the transmission window (timeslots before the deadline) into h_r timepieces with an equal number of timeslots. If every hop on the path has available resources to transmit a chunk within its own timepiece, this path is viewed as a feasible path.

For each chunk, HEU selects a shortest feasible path. If one link on the path has multiple free timeslots within its own timepiece, HEU selects the later timeslot with higher probabilities for transmission. To some extent, HEU also performs the scheduling policy of "As Late As Possible".

Similar to DCRoute, it pulls traffic from future timeslots back to the current timeslot if underutilized resources are found. Note that HEU can pull back traffic if the link is available because only a single link is involved under SnF. DCRoute can pull back traffic only when all links of the e2e path are available.

HEU is a heuristic algorithm. The authors conduct experiments to compare it with RES, which solves the optimization problem straightforward for each new request. HEU outperforms RES although HEU is a heuristic while RES targets at the optimal solution. It is because RES does not consider unknown future requests while HEU saves resources for urgent future requests.

6. Scheduling to Minimize Transmission Cost

All previous schemes assume that the datacenter provider has made its provisioning decision, *i.e.*, the capacity of each network link is fixed and ISPs are using a *capacity-based model*. In other words, the datacenter would pay a fixed charge according to the capacity it buys, no matter how much traffic is transferred on the inter-WAN links. During scheduling, the constraint is that the traffic on one link cannot exceed its capacity, and the objective on efficiency is equal to increase the utilization rates of links to satisfy requests as much as possible.

On the other hand, there is the other charging model for ISPs to charge datacenters, *i.e.*, *usage-based model*, wherein the transmission cost of one datacenter provider is directly related to the traffic rate on inter-DC links, either the maximum rate or the q-percentile rate. Under the usage-based model, some DCs would like to minimize the transmission cost incurred by inter-DC traffic requests as much as possible.

In this section, we would review some schemes that aim to minimize the transmission cost of inter-DC WANs.

Some schemes, such as Jetway, is for deadline-agnostic scenarios, wherein each request has a rate demand and wants to maximize its allocated rate, similar to schemes in Section 3. Other schemes, such as GRESE, Postcard and TrafficShaper, focus on deadline-aware scenarios, wherein each request wants to meet its deadline requirement, similar to schemes in Section 4.

6.1. Jetway: Deadline-agnostic Requests [44]

Jetway is designed for scheduling of inter-DC video flows, and each flow is specified with a flow rate requirement. The basic idea of Jetway is to accommodate more flows using the resources that have been paid and buy more resources only when necessary. The authors of [60] also study the resource allocation problem for deadline-agnostic flow requests, but they assume a usage-based cost model, which means they can always choose available paths with the least cost for flows. The problem formulation is quite straightforward, and their main contribution is on the decomposition of the original optimization problem to develop a distributed method.

Jetway solves the scheduling problem for each independent timeslot by formulating two flow problems as follows. In a timeslot t, the authors assume all historical information is known, and they define the charging volume up to the time interval $(t-1)$ on one link as the *Already Paid Portion of Traffic Volume*. Jetway constructs a graph where each inter-DC WAN link is with a capacity of its already paid portion volume. Obviously, Jetway would like to accommodate inter-DC requests in this graph as much as possible, which is a *maximum concurrent flow problem*. After that, the second step is trying to minimize the additional cost to accommodate all remaining demands. It is exactly in the form of a *minimum-cost multi-commodity flow problem*.

We note that Jetway tries to find the best solution for the current timeslot based on the historical information because the leaving and arriving of flow requests are unknown for future timeslots. It only schedules flows in terms of the spatial dimension.

6.2. GRESE: Partially Elastic Requests [45]

Similar to Jetway, GRESE is also based on the simple idea that the "already paid" bandwidth should be used with the highest efficiency and more bandwidth is bought only when necessary. But GRESE deals with deadline-aware elastic requests instead of flow requests.

The authors of [45] pinpoint that the assumption that the data of one request can be sent at any rate may not hold. For example, it is well known that the maximum achievable throughput of a TCP connection has a ceiling related to the RTT and the packet loss of its transmission path. Therefore, some requests are not "completely elastic", and each request should be specified with three more parameters: a minimum rate B_{min}, a maximum rate B_{max}, and a flex Δ (maximum variation of rates in two consecutive timeslots).

In [45], the authors design a scheduler, named GRESE, to decide the sending rate in each timeslot for each "partially elastic" request with the objectives of deadline-guaranteed and cost minimization. They solve this problem using some very simple heuristics.

In each timeslot, each request has a minimum sending rate, which is the maximum between its B_{min} and the required smallest rate to complete before its deadline, *i.e.*, the remaining transfer size over the remaining time. GRESE allocates to each request its minimum sending rate in an order wherein the non-preemptable requests (whose rate must be non-zero due to Δ) are satisfied first and the remaining requests are sorted by multiple attributes such as flex, deadline and remaining demand size. If the resource is insufficient for this allocation, GRESE would increase the link capacity by 1% and try again. If there is excess capacity after the allocation, GRESE allocates more resources to the requests with tight deadlines or large flex.

We can see that GRESE decides to buy more resources only when it is necessary. Since ISPs usually charge DC providers based on maximum bandwidth (or 95 percentile), once GRESE decides to increase link capacity, the increased capacity can be used till the end of one billing period.

6.3. Postcard: SnF to Minimize Cost [46]

By enabling the SnF mechanism, Postcard can postpone some traffic to reduce the peak traffic of some links, and thus it can further minimize the operation cost charged by ISPs.

The authors also construct a time-expanded graph to formulate and solve the cost minimization problem. In Postcard, all nodes are duplicated one copy for each timeslot, and there is no link within the same copy. Let i^n denote the node i in the timeslot n. A node can only be connected with nodes in neighboring timeslots. If there is a link from i to j $(i \neq j)$ in the original network, there would be a link from i^n to j^{n+1} $(\forall n)$ associated with a particular capacity limitation and a particular cost per traffic unit. There is the other type of links, from i^n to i^{n+1}, representing that traffic can be stored at node i in timeslot n. Postcard associates these storage links with unlimited capacity and zero cost.

We can see that the way to construct the graph in Postcard is different from Amoeba and NetStitcher. It is because they have different assumptions on their data transportation. Postcard and BDT assume that data can be transmitted only one hop in one timeslot. In other words, when a node receives a piece of file, it must cache this piece and wait until the next timeslot to send it out. In contrast, NetStitcher and Amoeba allow packets traversing multiple hops as a consecutive flow in a single timeslot if no storage is required.

Then the cost minimization problem becomes a static traffic allocation problem with only linear constraints on the time-expanded graph, which can be solved by clas-

sic algorithms, *e.g.*, subgradient projection methods and interior-point methods.

6.4. TrafficShaper: Exploiting Percentile Charging [47] [61]

Exploiting the 95% percentile charging model to reduce cost has been studied in inter-domain traffic engineering for traditional ISP networks [62]. Under this charging model, some (say 5%) timeslots with the heaviest load can be viewed as "free" because they do not affect the cost of this charging period. Therefore, we can set some timeslots as "burstable" or "free" timeslots and move traffic to these timeslots as much as possible as long as the capacity constraints are not violated. TrafficShaper tries to use the same idea in inter-DC WANs with the additional consideration to guarantee successful completion of most requests.

Different from other schemes wherein one DC operator should pay for the inter-DC dedicated links or paths between two sites, TrafficShaper assumes that the DC operator pays for the uplink and downlink of every site under a usage-based charging model.

Although the problem is formulated as a centralized optimization, the authors make efforts to design a distributed system, which enables that one site can make its scheduling decision only based on previous timeslots (temporal decomposition) and local information (spatial decentralization).

Temporal Decomposition. Scheduling decisions in different timeslots are correlated due to the percentile function and the deadline requirements. Using Lyapunov optimization, the authors represent all capacity and deadline constraints by congestion of queues and transform the objective function as a tradeoff between cost and deadline misses. Then the problem is decomposed into several sequential subproblems, each of that is to be solved in a single timeslot. In this way, at the beginning of each timeslot, the decision process only needs to consider historical information in previous timeslots.

Spatial Decentralization. The authors further propose a decentralized implementation, with which each site can determine the rate of each transfer on its uplink and downlink using its local information. The destination site of one request would send its planned receiving rate to the source site, and the source site chooses the minimum of sending rate and receiving rate to send traffic for this request. In this way, the time-complex centralized computation with global information is avoided.

7. Scheduling with Pricing

Some researchers noticed that scheduling schemes had to incorporate proper pricing mechanisms to charge users according to their requirements such as priority, deadline, and the size of demand. Otherwise, users always have inclinations to submit inflated priorities and tightened deadlines, and the performance of scheduling would be degraded drastically due to the false information provided by users.

In Amoeba, the authors mentioned some simple principles of pricing models. For example, higher price should be set for better service; charging for a request should be able to reflect the pressure the request exerts to the network, which might be evaluated by the average bandwidth requirement instead of the volume. Obviously, it is far away from enough. For example, transmitting a file in non-peak hours should be less expensive than peak hours, so that resources during non-peak hours can be utilized more and the inter-DC WAN may get more revenue. In this example, pricing models are used to encourage users to behave properly to optimize a particular economic objective.

In this section, we would briefly summarize several scheduling schemes with pricing mechanisms. Roughly speaking, they formulate the scheduling problem as follows. Each request is specified with a *valuation* by the user to reflect its importance. After the system receives the request, it would compute a spatial-temporal scheduling decision and also a price for the request. If the valuation is also sent to the scheduling system as a *willing-to-pay price* or a *bidding price*, the system can make an admission control decision by comparing the computed price and the willing-to-pay price. If the valuation should be kept by the user as a secret, it can be viewed as an *auction*, wherein the computed price is returned to the user and based on a comparison the user would make a decision, such as whether to transmit or how much volume to transmit.

There might be two natural economic objectives when it computes the scheduling and the price. A public datacenter provider may want to maximize its revenues from users. At least it wants to distribute the cost it pays to ISPs among users and make sure that the revenue is larger than the cost. A private datacenter provider only provides services for its own applications. Its objective can be modeled as maximizing *social welfare*, which equals the total valuation of accepted requests minus the ISP cost that the datacenter provider should pay to complete transmission of these requests. Furthermore, in [36], the authors argue that all DC providers will be driven to optimize social welfare due to high competition in the market.

7.1. Shapley: Welfare Maximization and Shapley-based Charging [48]

Shapley value is an approach in the cooperative game theory to allocate aggregated profit/cost generated by a coalition to individuals in the coalition fairly and efficiently [63]. The basic idea is to allocate profit in accordance with the surplus contribution of each individual. It has been proven that Shapley value is the only allocation approach which achieves all the three nice properties, *i.e.*, efficiency, symmetry, and fairness [64].

In [48], the authors assume that the WAN is charged by ISPs using a usage-based model. The cost should be shared by all requests in the accounting period. They propose that the price of a request should be linear with its Shapley value to reflect the cost it incurs to the WAN. The linear coefficient, which is a parameter set by the WAN,

should be larger than 1 to guarantee budget balance. On the other hand, it cannot be too large because the network needs to accept as many requests as possible for more revenue.

The authors state that its objective is to maximize social welfare. We notice that the authors argue that the network always can buy more bandwidth as long as it is willing to pay more cost. So they assume that the WAN always has sufficient bandwidth on all links to satisfy all requests. Under this assumption, there is no resource contention, and welfare maximization is equal to accept every request whose Shapley value is less than valuation.

Here we only introduce its online algorithm because offline algorithms cannot be used in real networks. The authors propose a two-step procedure as follows.

Traffic Scheduling. When a request arrives, the system assumes the request is accepted and tries to find an optimal schedule. Although it aims to maximize welfare, since future requests are unknown, the authors just use a very simple scheduling algorithm. In terms of the spatial dimension, it assumes there is only one usable path for each request. In terms of the temporal dimension, it distributes the traffic of one request equally over the entire transmission time window, arguing that this simple smoothing can avoid high traffic peak. It replaces a deadline requirement with a bandwidth requirement, which means the system is putting a more stringent requirement for each request.

Price Computation. The price of a request should be calculated from its offline Shapley value instead of online Shapley value because the authors argue that the charge to one request should not be affected by its arrival time. In other words, the extra cost caused by this request must be calculated under each of arrival sequences (permutations of all requests), and the average of all these extra costs would be used for charging.

The challenge is that future request arrivals are unknown when this request is received. We know that the online Shapley value can be easily computed by subtracting the cost incurred by old requests from the total cost incurred by this new request and all old requests together. The authors propose two ways to estimate the offline Shapley value from the online Shapley value. One is to produce request arrivals based on the Monte Carlo simulation and then calculate using all the produced request arrivals. The other way is scaling the online Shapley value by a factor calculated from the data of the previous accounting period.

7.2. Pretium: Pre-computing Link Prices of Each Timeslot [36]

In Sigcomm 2016, researchers from Microsoft presented *Pretium*, which also aims to maximize social welfare. At the beginning of every time window (*e.g.*, a day), based on the requests and the corresponding scheduling in the last time window, Pretium calculates the optimal prices for every link/timeslot under the optimal scheduling to maximize the social welfare of the previous time window. These prices are stored in the system and will be used to produce price quote for requests coming in this time window.

Pretium assumes the per-byte value is fixed for one request, which does not hold for transfers that are useless until the whole transfer is completed. The authors state that Pretium can be extended for non-linear utilities but do not explain how to extend.

Let us denote the price to use a link e in a timeslot t as $P_{e,t}$. Pretium works as follows.

Preliminary Scheduling and Price Quote Calculation. Since all $P_{e,t}$ have been known, Pretium can easily calculate the price of each usable path in each timeslot. In the preliminary scheduling, Pretium always steers traffic of one request to the minimum-price path/timeslot with available bandwidth. In other words, it uses the minimum-price path/timeslot until the path in the timeslot is saturated, and then it starts to use the path/timeslot with the next lower price. Therefore, the price quote is a piece-wise linear function of the transmitted volume. The price quote also includes a capacity bound, *i.e.*, the maximum volume that the network can accommodate for the request before its deadline.

Demand Size by the User. Each user has a secret per-byte valuation for its request. It would like to transfer more bytes as long as the volume is less than the capacity bound and the marginal price to transfer the byte, which can be calculated from the price quote function, is less than its per-byte valuation. The user notifies Pretium of its decision on the data size to transfer.

Optimal Scheduling in Each Timeslot. The preliminary scheduling only achieves approximate local optimal for a single new request. At each timeslot, Pretium would reschedule all requests together for global optimal social welfare. There are two issues here. First, Pretium does not know the values of requests, which is required in calculating the achieved social welfare. Second, if the ISP charges the datacenter provider based on usage, the $95th$ percentile usage-based charging would make the optimization problem NP-hard. For the first issue, Pretium can have a rough estimation from its final transfer size and the piece-wise linear price quote function. For the second issue, the authors propose to approximate the 95 percentile charges using a linear function and transform the original NP-hard problem to a linear program. The linear program has a lot of constraints, and sorting-network inequalities are used by Pretium to reduce the number of constraints.

7.3. Uniform Price Auction: Value-based Allocation [49]

In [49], the authors propose a scheme which aims to maximize revenue for the datacenter provider. Here, one request is specified with its demand D_i and its willing-to-pay prices v_i. The scheme allocates resources to requests using the formula $q_i = (1 - \frac{p}{v_i})D_i$ wherein p is the price charged for per unit of bandwidth.

It is a *uniform price auction* game [65], and the seller (datacenter provider) can find the optimal price p to achieve maximum revenue, $p \times \sum_i q_i$, through a simple computation.

Under this allocation scheme, if there are sufficient resources for allocation, *i.e.*, the capacity is larger than $\sum_i q_i$, the total allocated bandwidth would be 50% of the aggregate demand from users, which means each user only gets a half of its demand satisfied in average. Obviously, it works only for monopolist datacenter providers, which can set a very high price for its revenue maximization.

If there are insufficient resources for allocation according to the formula, it means the maximum-revenue price is too low and the seller must set a higher price to reduce the allocation of each user. The best price in this case is the market clearing price, which can allocate all resources to users.

One desired property of this scheme is that the request with a higher value will be allocated more resources. Moreover, the dominant strategy of each request is to submit its valuation as its willing-to-pay price, which means truthfulness in bidding value.

It has two undesired properties. First, no request can get its demand fully satisfied. Second, resources can be wasted seriously because social welfare is not considered at all.

Furthermore, this scheme works only for the allocation of a single resource, *e.g.*, a lot of requests are competing for the resource of a single link. It should be extended for scenarios with multiple resources to solve the real-world inter-DC scheduling problem.

8. Scheduling of Inter-DC P2MP Requests

Point to Multi-Points (P2MP) transfer requests are very popular in inter-DC WANs. For example, one DC operator would like to make several copies at different sites for one file or a dataset, *i.e.*, geo-replication for performance or fault tolerance.

Obviously, transforming a P2MP request into multiple independent point-to-point transfers, *i.e.*, sending the data from the source to every destination directly, can yield poor efficiency because the file may be transmitted for multiple times on one link. Traditional IP layer multicast protocols are more efficient than multiple unicasts, but they are with complex management cost and may not be feasible across ISP networks. Researchers propose various centralized scheduling schemes for inter-DC WANs to improve the efficiency of P2MP transfers.

The scheduling problem for P2MP requests can be summarized as follows. Each request is specified with a source, a set of destinations (instead of a single destination), and a demand (either a rate demand or a transfer size with a deadline). The data would be transmitted on forwarding trees (instead of paths). The set of usable forwarding trees can be given as an input, or it is needed to be found by the scheduling systems. A scheduling decision includes a set of selected forwarding trees and corresponding sending rates on each selected tree in each timeslot.

In this section, we would review several P2MP scheduling systems. Blossom and Calantha are designed for rate requests (deadline-agnostic as in B4 and SWAN), and they schedule multiple requests simultaneously in the spatial dimension. They take candidate forwarding trees as input, and each forwarding tree consists of all destinations. Multiple trees are used to maximize the throughput of the request. DCCast and QuickCast work in a FCFS manner, which means they only schedule a single request at a time. They aim to complete the request as soon as possible and schedule the request in both spatial and temporal dimensions. In order to avoid packet reordering, they only use a single tree for one destination. But the destinations can be partitioned to multiple groups, and each group is assigned with an individual forwarding tree. Airlift is different from these schemes. It exploits a feature of network coding and transforms one P2MP request to multiple unicast requests to find a final scheduling solution.

8.1. Blossom [50] and Calantha [66]: Max-Min Fair Multi-Tree Multicast

Blossom aims to maximize network utilization while maintaining max-min fairness in terms of satisfaction among all P2MP transfers. It assumes the candidate trees for every request have been known, then it is very easy to formulate the optimization problem mathematically. But solving the problem using standard LP solvers is time-consuming. With the help of the variable-size increment technique and the dual problem, the authors design an algorithm to find an approximate solution in polynomial time. The basic idea is that the dual to link capacity constraint c_e is defined as link length d_e. Blossom gradually allocates more traffic to the minimum spanning tree of each request, and d_e is updated accordingly. The saturation of a link can be tested by comparing d_e with $1/c_e$. Once a P2MP request is saturated on all usable spanning trees, it is excluded from later phases. The algorithm stops when all requests are excluded.

Calantha proposes to reduce the computational complexity of Blossom by using only *"hop-constrained"* forwarding trees. It reduces the size of the set of candidate trees for selection and makes sure that the maximum number of hops between the sender and any receiver cannot be too large, which is beneficial to constrain the bandwidth waste caused by deep trees. The idea of using only trees of limited depth has also been exploited in other scenarios such as [67]. Experiments show that it has been enough to achieve good network efficiency with only 2-hop constrained trees or 3-hop constrained trees.

8.2. DCCast [51] and QuickCast [68]: Minimizing Completion Time

DCCast and QuickCast work in a FCFS manner, and they aim to complete a request as soon as possible (transfer

performance) while using bandwidth as little as possible (network efficiency) for completing the request. These two objectives can be conflicting with each other. For example, a longer path with a larger free capacity would result in shorter completion time and more consumed bandwidth. DCCast and QuickCast just try to make a trade-off between them using heuristics.

Determine a Single Forwarding Tree. DCCast defines the link weight as the total load on a link, which is the load of all existing requests and the new requests that would be assigned to paths including this link. When a P2MP transfer arrives, DCCast selects the *minimum weight Steiner Tree* as its forwarding tree. An approximate minimum weight Steiner Tree can be found by GreedyFLAC at a quite fast rate. In order to complete the request as soon as possible, the sending rate is set to be the minimum available bandwidth of all links on the forwarding tree in all closest timeslots before the request is completed.

By the definition of link weight, the selected tree can avoid heaviest loaded links and longer paths. Furthermore, a transfer with a larger size tends to be assigned to a tree with a smaller number of links, and a transfer with a smaller size tends to use a tree with more residual bandwidth.

The forwarding tree can be implemented using Group Tables of SDN switches.

Partition Receivers to Accelerate. The completion time of a transfer is constrained by the bottleneck link on the tree. For example, assume the link between a receiver j and its parent node is heaviest loaded, then the transfer must be slowed down to the rate allowed by this bottleneck, thus all nodes would be affected because they are on the same single forwarding tree.

Noticing this disadvantage, QuickCast [68] proposes to group receivers into multiple subsets and each subset is assigned to a separate forwarding tree. In this way, j and j's child nodes can use the same forwarding tree as before, while other nodes can have a different tree without this bottleneck and receive the data faster than in DCCast. Then, the average completion time is reduced at a cost of additional bandwidth overhead.

How to partition receivers is the key problem here. QuickCast clusters receivers according to the distance between any two nodes, *i.e.*, the hop number of the shortest path between two receivers. This clustering process repeats until the number of groups is equal to the predefined desired value. In this way, it ensures that the total number of links in all forwarding trees is minimized, which helps to reduce bandwidth consumption.

Increasing the number of groups would consume more bandwidth, and it can even result in longer completion time due to the resource contention among trees. QuickCast chooses to partition receivers into at most two groups. It compares the single-tree solution and the two-tree solution in terms of the total load of all links on the selected forwarding trees. If the increasement of the total load is less than a predefined threshold, the two-tree solution would be accepted. Otherwise, the additional overhead is too much and the single-tree solution should be used.

QuickCast also uses the minimum weight Steiner Tree for each group. Since there can be resource contention among groups, QuickCast has to take fairness among groups into consideration when determining the sending rate of each forwarding tree.

8.3. Airlift [52]: Multicast Using Network Coding

Airlift is designed to schedule flows of multiple multi-party video conferences running on an inter-DC WAN. These flows are with stringent delay constraints, and higher bit rates are preferred. Therefore, Airlift aims to achieve three objectives: satisfying delay constraint, maximizing throughput (bit rate), and achieving fairness among conferences.

In order to improve scalability, flows of conferences are aggregated according to the source site and the set of destination sites of each flow. Each aggregation is referred to as a *session*.

The basic idea of Airlift is that a multicast session can be transformed into multiple unicast conceptual flows and then Airlift only needs to solve the scheduling problem for these special unicast flows.

Conceptual Flows Based on Network Coding. According to [69], if a source can send data at a rate x to every destination independently, the rate x must be implementable for all destinations using a multicast session with network coding. Based on it, a multicast session (aggregated from multiple conferences) can be transformed to multiple unicast conceptual flows, wherein each flow is from the source of the session to one destination of the session, and these flows would not compete for bandwidth. When a conceptual flow is scheduled, the bandwidth allocated to other conceptual flows in the same session can also be used by it. In other words, assume Airlift allocates resources x to a multicast session, then every unicast conceptual flow of this session can use all of x.

Exclude Paths with Longer Delay. Airlift assumes that the delay of each link has been known, and an end-to-end path delay is equal to the sum of delays of links on the path. Therefore, all feasible paths for each conceptual flow can be found using a depth-first search algorithm.

Now each conceptual flow has had a set of usable path. The scheduling problem is similar to the problems in Section 3, except that Airlift needs to revise capacity constraints. The formulated problem can be solved by standard LP solvers.

Transform the Scheduling Solution for the Initial Problem. The derived scheduling decision for conceptual flows should be transformed back into a feasible scheduling decision for original multicast sessions. Let us illustrate using a simple example. Assume a session S_1 is from the site D_1 to D_2 and D_3, so there are two conceptual flows $f_{1,2}$ and $f_{1,3}$. Assume the derived scheduling is as follows: $f_{1,2}$ is sending at a rate of 10 on the path $(1, 4, 2)$,

and $f_{1,3}$ is sending at a rate of 6 on the path $(1, 4, 3)$ and a rate of 4 on the path $(1, 4, 5, 3)$. Then the final multicast tree should be: sending 10 on the link $(1, 4)$, and the site D_4 would replicate two copies, wherein one copy would be sent to D_2 and the other copy sent to D_3 and D_5 with a split ratio $(4 : 6)$. D_5 then forwards the ratio it receives to D_3.

8.4. AGE [53]: P2MP Bulk Transfers

In [53], the authors also study the P2MP transfer problem, but the scenario they concern is different from the P2MP schemes mentioned above. First, they focus on bulk transfers instead of streaming transfers. A streaming transfer requires that the same rate should be allocated to the transfer over all links of its forwarding tree to reach all the destinations at the same time at any timeslot. A bulk transfer just requires the file can be transferred to the destinations before the specified deadline. Therefore, a destination node that has completed the transfer can serve as a source and deliver the file to uncompleted destinations. The problem formulated by them can be viewed as finding an overlay forwarding tree with store-and-forward enabled. Second, their goal is to maximize the number of destinations that can receive the file successfully before the specified deadline. In the paper, the authors just try to find the optimal scheduling decision for a single transfer request instead of optimizing the overall performance of multiple requests, therefore there is no need to consider the resource contention among transfers and the solution can be figured out by solving the corresponding MIP problem directly.

9. Cross-Layer Scheduling: Optical Layer and IP Layer

Schemes discussed before all assume a given and fixed network-layer topology. In recent years, if a network-layer WAN is built on an intelligent optical layer, the network-layer topology can be constructed dynamically by reconfiguring optical devices [70][71][72][73]. Therefore we can further improve the performance of inter-DC WANs by jointly optimizing both optical layer and network layer.

9.1. Basics of Reconfigurable Optical Layer

One inter-DC WAN built on an intelligent optical layer consists of network routers, optical switches called Reconfigurable Optical Add-Drop Multiplexer (ROADMs), regenerators, and fibers. A WAN link is an optical circuit with a specific wavelength, which is established by properly configuring the wavelength switching in ROAMs along the link.

Modern ROADMs can be configured remotely in hundreds of milliseconds, and the time can be reduced to tens of milliseconds and even lower in the future. Enabling by today's ROADM technology, we can change how routers are connected for particular goals via reconfiguring how wavelengths are switched in ROADMs.

If the distance between two ROAMs is too long, a regenerator is necessary to regenerate the optical signal because a wavelength normally has limited transmission range which is called *optical reach*. Optical-network providers usually pre-deploy some regenerators at certain concentration sites to enable transmission between any two ROADMs. The number and locations of pre-deployed regenerators are important constraints during reconfiguring topologies.

9.2. OWAN [54]: Dynamic Network Layer Topology

In Sigcomm 2016, some researchers from academia proposed OWAN, a centralized system to optimize the performance of one inter-DC WAN and transfers on it by jointly controlling the optical circuit configurations (OC) and the routing configuration (RC). OC is configured on optical devices and it determines network topology. RC includes routing paths and sending rates and it should be installed on routers and end hosts.

OWAN runs its reconfiguration algorithm periodically in each timeslot. Obviously, the optimization problem it formulates is with more decision variables and constraints than the cases with fixed topologies, and some constraints are integral. In order to reduce time complexity, the authors design a greedy algorithm with a reduced search space.

OWAN exploits the *Simulated Annealing* algorithm to greedily search for the approximately optimal topology that can produce the highest total throughput. The simulated annealing is an iterative algorithm. It starts from the current topology. At each iteration, it constructs a neighbor state (a new topology) and decides whether to transit to the new state according to the throughput of the new topology. We describe the algorithm in detail as follows.

Constructing a Neighbor State. In each iteration, OWAN randomly selects two links, *e.g.*, (u, v) and (p, q), and changes one wavelength for (u, v) and (p, q) to (u, p) and (v, q). Let θ be the capacity of one wavelength. The above change means the capacity of each old link is reduced by θ and each new link has its capacity increased by θ. In this way, the ingress and egress capacity of each router is kept unchanged, which is a constraint related to router performance.

The simulated annealing algorithm only searches neighbor states, so the search space is significantly reduced. Moreover, by starting the first iteration from the current state, the algorithm tends to find a target topology that is close to the current topology, which helps OWAN minimize the disturbance caused by configuration updates across timeslots.

Computing Throughput of the New State. OWAN uses the maximum throughput that can be achieved by the given state as the *energy* of the state, which is a key factor to decide whether to transit to the state. Therefore,

it is necessary to compute the maximum throughput of a network layer topology.

It should be noticed that not all topologies can be implemented due to the constraints of the optical layer such as fibers, wavelengths and regenerators. So the first step is to find an optical layer configuration (OC) that can implement the given state. OWAN designs a greedy algorithm to search for the OC. It is possible that the greedy algorithm finds that the capacity of one or more links in the given topology cannot be satisfied. In that case, the link capacity has to be decreased to form an implementable topology.

In the second step, OWAN schedules transfers to optimize their performance and computes the corresponding network throughput under such scheduling. It is the traditional scheduling problem in the schemes with fixed topologies, which is known to be hard. OWAN uses heuristics to solve it. Roughly speaking, it prefers the shortest path even for different transfers, which helps in achieving a larger throughput. For transfers whose shortest paths are with the same length, it uses classic scheduling policies like shortest job first (SJF), which helps in minimizing the average completion time, or earliest deadline first (EDF), which helps in completing transfers before deadlines as much as possible.

Transition to a New State. Simulated Annealing algorithm has two basic concepts, *i.e.*, energy and temperature. In OWAN, the *energy* of a state is defined as the maximum achieved throughput of the state. The initial *temperature* is set to be the initial energy, and the temperature is decreased by a factor of α at each iteration.

OWAN decides to transit to a new state with a probabilistic function P. If the new state is with a higher energy, $P = 1$; otherwise, $P = e^{(e_{current} - e_{neighbor})/T}$, where $e_{current}$ and $e_{neighbor}$ are the energy of the current state and the new state respectively.

Stopping the Search. The algorithm stops when the temperature is less than a threshold ϵ. α and ϵ are parameters to control the number of iterations.

10. Practical Issues

Whether a proposed scheme can achieve the best result depends on many factors. For example, whether the problem is properly formulated to reflect the intentions of datacenter operators and users correctly and model the network environment comprehensively? Whether the optimization problem can be solved online or how far the solution given by greedy algorithms is from global optimal? Even the solution is optimal, the final achieved result still depends on many practical issues, *e.g.*, hardware capability and imperfect knowledge.

In this section, we would summarize some practical challenges and briefly describe how current schemes deal with them. We can see that most solutions in current schemes are intuitive. Researchers can improve the scheduling performance by studying these issues more deeply in the future.

10.1. Time-Complexity and Scalability

In real networks, the scheduling system should run online, which means it should be able to find the desired scheduling solution within a short time. However, most mathematical problems formulated are very complicated with a lot of constraints and decision variables, even NP-hard. Therefore researchers make a lot of efforts to reduce the time-complexity of algorithms. We summarize their ideas as follows.

Well-known LPs. A lot of schemes formulate linear programs, such as multi-commodity flow problem, min-cost flow problem, and packing-covering problem. These problems are well-known, and researchers have proposed many approximate algorithms for them. Some of them even have standard solvers. For example, the problem of maximizing throughput with fairness in mind has been studied in [74] and [75]. It is worthwhile to study whether their algorithms can be exploited to solve the problems formulated by B4 and SWAN.

It is reasonable to conjecture that approximate algorithms can work well if the number of flows or requests is not very large or the system works in a FCFS manner. Otherwise, the time-complexity can still be a problem and it is needed to analyze in more detail case by case.

Greedy Algorithms and Heuristics. Some schemes depend on greedy algorithms or heuristics instead of formulating mathematic problems. For example, B4 uses a progressive water-filling algorithm to push traffic to all usable paths as much as possible. DCRoute schedules the arriving request to timeslots as late as possible. Since the schemes cannot have perfect information about future requests, some experiments demonstrate that these greedy algorithms, if well designed, can outperform the optimal solution derived by solving optimization problems straightforward.

Some schemes depend on heuristics to reduce complexity but still need to solve some LPs. For example, Amoeba uses some heuristics to select some requests to reschedule instead of rescheduling all requests. Then it needs to solve a maximization problem to find a scheduling decision for these selected requests. In this way, Amoeba tries to achieve a tradeoff between optimality and time-complexity.

Aggregation or Selection. Schemes can reduce the number of decision variables by aggregation or selection. Both BwE and SWAN aggregate flows at the site/cluster level and solve their scheduling problems for flowgroups. Since the number of flowgroups is limited, we do not need to worry about time-complexity too much. Furthermore, BwE only enforces 10% of the flows but these flows can account for 94% of the traffic. Experiments show that we can focus only on the subset of flows that contribute most to link utilizations and congestions.

10.2. Imperfect Knowledge and Unexpected Failures

In [43], their experiments show that a heuristic outperforms the algorithm that solves the optimization problem straightforward because the heuristic greedily saves resources for future requests while the optimization problem cannot handle unknown future requests. It demonstrates that we must pay attention to the influence of imperfect knowledge and unexpected failures.

High-priority Traffic. All schemes should deal with dynamic network condition. Bulk transfers are using the left-over resource of high-priority traffic flows. Since the volume of high-priority traffic varies, the resource that can be used by bulk transfers is also changing. For example, Netsticher uses the *Sparse Periodic Auto-Regression* (SPAR) estimator to derive a prediction for the next 24 hours from historical information collected by bandwidth monitoring tools. Fortunately, NetStitcher finds that measurements show its environment is with predictable periodic patterns. Tempus and Amoeba also predict high-priority traffic demand from historical usage.

Demand Estimation. The spatial schemes to schedule flowgroups need to estimate the demand of each flowgroup in the next timeslot. Roughly speaking, they measure the demand of each flowgroup continuously, and so they can estimate future demand from usage history. For example, BwE empirically finds that $Demand = max(max_{\Delta t}(usage) \times scale, min\_demand)$ with $\Delta t = 120s$, $scale = 1.1$ and $min\_demand = 10Mbps$ works well for $user\text{-}fgs$ for Google's network applications.

Untrust Users. The temporal schemes to schedule transfers generally ask users to specify the detailed information on their requests. There is no estimation error, but the scheme should ensure that users have no inclination to overstate or understate their demands. For example, in the Shapley scheme, the system runs its algorithm with different deadlines earlier than the deadline provided by the user, and it chooses the deadline with which the request can be accepted and also yield the smallest charge to the user. The traffic schedule and auction price are derived from the chosen deadline instead of the deadline given by the user. In this way, it brings the lowest price and users have no incentive to bid an earlier deadline.

Future Requests. The other issue of online temporal scheduling systems is that they have no idea about request arrivals in future timeslots. Some schemes, such as Amoeba, just work in the FCFS manner, wherein a new request can only use the resource left by old requests and the new quest would be rejected if the left resource is insufficient. These schemes do not provide fairness among request arriving at different timeslots, and they also cannot achieve global optimal performance. Other schemes consider that all requests should be able to receive the same service quality no matter when they come. For example, in Tempus, future network capacity is systematically underallocated to leave room for future requests. The room it leaves is larger for far future than near future timeslots. It makes that all capacity in the current timeslot can be fully used by existing requests, and the room in far future timeslots would be used gradually as the time moves.

Deal with Unexpected Failures. There is no way to avoid unexpected network failures. All schemes must define how to react to failures. Many schemes just propose to recompute their scheduling solutions. For schemes with admission control, it is possible that some accepted transfers cannot be completed successfully due to failed nodes or links, so they must decide which requests can be rejected. For example, Amoeba removes all the requests on the failed link and runs its algorithm to decide whether to admit the requests one by one according to their arrival times. NetStitcher is more complex when failures occur because of in-network storage. NetStitcher assigns a new demand (equals to the demand not delivered yet) at the source and assigns a new demand (equals to the volume it currently stores) at each intermediate storage node. Then NetStitcher needs to solve a multiple source maximum flow problem. Furthermore, after most of the file is received, NetStitcher asks some storage nodes to keep "inactive" replicas of already forwarded pieces. Once an unexpected failure occurs and the transmission of last pieces is affected, the receiver can get subpieces simultaneously from the source and the storage nodes. In this way, it avoids performance degradation caused by the last delayed pieces.

Deal with Imperfect Knowledge. Besides running the scheduling algorithm with updated information sufficiently frequently, schemes also develop different simple ways to mitigate the waste caused by imperfect knowledge and improve network utilization. For example, BwE estimates an upper level demand by aggregating its lower level demands, ignoring the benefit brought by statistical multiplexing of burst flows. Thus it multiplies the allocated bandwidth by a burstiness factor (≥ 1) before redistributing the resource to lower level flows. In Pretium, the link prices in each timeslot are determined using historical data. Obviously, the prices may not be optimal. Pretium proposes short-term adjustments, *e.g.*, increasing the price of a link when the link is heavily loaded. After admitting a new request, Pretium doubles the prices for links whose utilizations are more than 80%. Amoeba detects if there is unallocated bandwidth at the beginning of each timeslot and pulls traffic from the future timeslots if spare resources are found.

10.3. Enforcement of Scheduling

After the optimal scheduling solution is computed, the solution should be enforced in the WAN. In most schemes, the scheduling solution means a set of usable routing paths and the sending rate on each path for each transfer or traffic flow in each timeslot. Enforcement is not easy under the current network model. For example, Amazon AWS plans to allocate a particular data rate to a user class. It implements a rate limiting mechanism, but a measurement study shows the throughput varies a lot in different time

periods. We summarize some ways proposed for scheduling enforcement in this subsection.

Enforcement of Priority. The traffic flows of interactive services should be specified with high priority and sent out as soon as possible. Both SWAN and BwE propose to ask hosts to tag data packets with DSCP bits to indicate the priorities of data packets. In networking devices, priority queues should be configured to accelerate the processing of packets with high priority.

Enforcement of Sending Rate. The rate limiting can be enforced on hosts/supervisors or on networking devices. Roughly speaking, host-based enforcement is preferred because it is considered to be more scalable than network-based enforcement. For example, SWAN, BwE and Amoeba exploit token bucket or Hierarchical Token Bucket (HTB) to enforce rate limiting on hosts. By modifying Linux networking stack, each outgoing packet is intercepted and marked using the *nfmark* field of *sk_buff* to uniquely identify its owner (task, service or user) for rate limiting. By experiments, Amoeba states it performs accurate real-time enforcement, and the difference between the scheduling target and the actual throughput is less than 5% for more than 95% of requests.

However, host-based rate limiting requires full control of end hosts. If source hosts are not under the control of the datacenter operator, switches must be able to detect services that are sending more traffic than allocated from traffic logs. After detection, BwE proposes to notify the owners of the service and re-mark the DSCP bits of excess packets to be the lowest priority to avoid the greedy service using resources allocated to other services.

Enforcement of Routing. The basic ideas to enforce routing are overlay tunnels and source routing, although the implementation details might be different. SWAN pre-installs tunnels, *i.e.*, label-based forwarding rules, on OpenFlow switches. Each data packet is assigned with a label (using VLAN ID) at its source switch, and the label implies a set of tunnels and traffic splitting ratios among these tunnels. Remaining switches just conduct label-based forwarding. B4 works in a similar way to SWAN, except it uses IP-in-IP tunnels. Amoeba implements the routing by explicit path control using Xpath [76]. Each desired path is associated with an IP address, and it is installed in IP LPM (Longest Prefix Match) tables of switches. Then Amoeba leverages the NAT technology to translate the original destination IP address into the desired path ID according to the scheduling decision.

Enforcement of Traffic Splitting. Most schemes allow multipath routing. Among them, some schemes determine the sending rate on each path, while others give as the result the total rate on all paths and a splitting ratio among these paths. In the latter case, it needs only one rate limiting for a single service and the split ratio can be implemented on ingress switches, while the former case needs to conduct rate limiting on each individual path for the service.

Group tables in the OpenFlow pipeline can be used to implement splitting. When it is not supported by the hardware, SWAN proposes to split IP addresses into multiple groups to approximate the desired split. B4 notices that hardware switches may not be able to support any granularity splitting. For example, if the hardware supports splitting at a granularity of 0.5, then a desired ratio $(10 : 25/3 : 5/3)$ has to be approximated to $(0.5 : 0.5 : 0)$. Such approximation may result that the achieved network efficiency is worse than expected. It seems BwE can take the approximated splitting ratio as input and tune the sending rate of each service to improve the result achieved by B4.

10.4. Updating Configurations

Most schemes need to update their scheduling decisions periodically. It is well known that there are some important issues during updating network configurations and we must pay special attention to them. These issues become more important due to the high frequency of updating. It is not easy to ensure that the network can maintain high utilization during updates. In fact, although SWAN incorporates a lot of mechanisms to deal with issues during updates, it still finds that an update frequency of 10 (100) minutes reduces throughput by 5% (30%).

Minimizing Disturbance of Updating. To make schemes practical in real-world networks, all of the previously promised allocations should be retained in future timeslots as much as possible. This is to reduce the cost of updating network configurations and avoid performance variance due to path changes across timeslots. Tempus achieves this property by exploiting the iterative algorithm, *i.e.*, Young's method. Amoeba achieves this property by first computing whether new requests can be accommodated without changing the bandwidth schedules of existing requests. Even if the answer is no, Amoeba only selects a subset of requests for rescheduling using heuristics.

Forwarding Table Limitation. Switches can only support a limited number of rules. It means at least two issues. First, we must constrain the number of usable paths (tunnels) for each flow or request. Most schemes do not consider this limitation, therefore it is possible that their solutions are not implementable. SWAN states that setting the number of usable paths as the number of priority classes, *i.e.*, 3, has been enough to fully use the network capacity in its network environment. Therefore, SWAN proposes the following procedure. First, it runs the algorithm on all usable paths to derive a scheduling decision. Second, it selects the smallest latency path, after that it repeats to select the remaining path that carries most traffic as long as the rule limitation is not violated. Third, it reruns its scheduling algorithm on the selected paths. Then the final solution is both implementable and efficient.

Second, in order to avoid packet loss during updating, new forwarding rules must be added before the old rules are deleted. It also puts pressure on the size of switch tables. SWAN proposes that some rule capacity should be

left for new rules, and it also designs an algorithm to determine an implementable update sequence of rules. More rule space is left, the smaller number of steps are needed to complete updating.

Data Plane Capacity Limitation. Consider the scenario that all links are fully used. In this case, it is impossible to move traffic from one path to a new path without congestion. Therefore, SWAN proposes to leave about 10% of capacity unused at each link to accommodate moving traffic. Even with reserved capacity, it is still needed to design an algorithm to find a proper update sequence to realize a congestion-free update.

Dependencies of Configuration Operations. We have noticed that the sequence of updates should be designed to avoid transient overloads during changes. The other issue in determining the sequence is the dependency of operations. For example, B4 must configure the new path before it moves traffic to the new path. Similarly, it must move all traffic to the new paths before it removes the old paths from switches. In OWAN, a routing path cannot be used until the optical circuits for all links on the path have been configured. In summary, we should issue the operation after all operations it depends have been completed. Dionysus [77] allows operators to build a dependency graph and produce a proper update sequence. But it cannot handle cross-layer updates. OWAN extends Dionysus to represent dependencies of elements in different layers by introducing circuit nodes into its dependency graph.

10.5. General Challenges in SDN

Most schemes are using the SDN paradigm, therefore they can enjoy the benefits of SDN. On the other hand, they are also forced to solve the challenges brought by SDN, such as the coordination of states across distributed controllers, dependencies of multiple operations and failures of operations.

11. Discussions and Open Issues

Although quite a few resource scheduling schemes for data transfers in inter-DC WANs have been proposed as introduced in this article, the problem still remains an issue and far from being well solved. As datacenter technologies evolve and geographically distributed datacenters are more and more widely deployed, further research on the scheduling of inter-DC networking resources is necessary and valuable. In this section, we would like to highlight some open issues and future directions.

1) measurement study to improve understanding and prediction. Efficiently scheduling the inter-DC networking resources requires a good understanding of traffic characteristics on inter-DC WANs, but we only see very few measurement results on it. For example, knowing more about the applications that generate inter-DC traffic flows and their performance requirements can help us formulate the scheduling problem, especially the scheduling objective, more reasonably. Moreover, when computing the scheduling decision, some schemes need to predict the amount of available resources and arriving requests in future timeslots. Research works on measurement and analysis of inter-DC traffic flows are beneficial for us to develop advanced prediction algorithms.

2) nonconventional approaches to computing optimal solutions. As we described in Section 10, most mathematical problems formulated are very complicated with a lot of constraints and decision variables. Solving these optimization problems using conventional approaches is time-consuming, and using heuristics to derive approximate solutions cannot ensure optimality. In the surveyed articles, we do not see any efforts to compute optimal solutions using nonconventional approaches, such as evolutionary computation and machine learning. Evolutionary computation has been used in many cloud task scheduling schemes [24], and machine learning techniques have also been used to solve some types of optimization problems efficiently [78] [79]. Machine learning may also be used to learn scheduling objectives for allocating inter-DC networking resources.

3) interference between transport protocols and centralized scheduling decisions. As we know, transport layer protocols such as TCP are in fact completing the task of distributed resource allocation. It suggests the possibility that a traffic flow might not be able to really acquire the resources assigned by scheduling decisions. In addition, if the resource share assigned to a traffic flow varies a lot across two consecutive timeslots, the throughput of TCP connection might degrade unexpectedly due to the slow start phase and the AIMD algorithm of TCP. A lot of transport layer protocols have been proposed for intra-datacenter communication based on features of intra-datacenter topologies and traffic characteristics, and it is interesting to conduct a study on the influence of various centralized scheduling decisions on the throughput of transport layer traffic flows.

4) dynamic pricing strategies for inter-DC networking resources. A carefully designed dynamic pricing strategy can be an effective way to encourage tenants to use inter-DC networking resources efficiently. But pricing resources is always a hard problem. Particularly, it is very difficult to model user behaviors, *e.g.*, how they react to different prices? how much risk they are willing to take? The pricing scheme proposed in [48] is assuming the players are playing a cooperative game and they just aim to recover the total cost. In [36], the prices are pre-computed using historical data. It is interesting to design pricing strategies wherein usage prices are dynamically set by pricing agents to achieve particular objectives.

12. Conclusion

In recent years, as geographically distributed datacenters are more and more widely deployed, how to efficiently

utilize networking resources of inter-DC WANs and provide better performance for data transfers deserves significant attention. Researchers from both industry and academia have proposed various resource scheduling schemes to solve this problem. In this article, we first introduce how existing schemes understand and formulate the scheduling problem for data transfers in inter-DC WANs, *e.g.*, the scheduling objectives, scheduling dimensions and scheduling policies. Then we introduce each representative scheme, mainly focusing on how it solves the optimization problem with a lot of decision variables and constraints or how it finds reasonable heuristics to design its scheduling algorithm. Finally, we further examine the practical challenges in developing real-world scheduling systems and point out some open issues and future directions. Although most scheduling schemes share the same basic idea, which is to exploit the elasticity of transfer requests in inter-DC WANs and the optimality enabled by a centralized architecture, these schemes are developed for different scenarios (different network models and business objectives) and the ways in which they make their algorithms to be able to work in an online fashion can be very different.

In the real world, we can see most inter-DC WANs are still using traditional traffic engineering techniques, and most schemes reviewed in this article have not been deployed in any inter-DC WAN. It indicates that more research efforts are necessary and valuable for the scheduling problem. We hope this article can provide some help for research developments in this important field.

Acknowledgement

The authors thank the editors and anonymous reviewers for taking time to review this paper and for their suggestions that helped improve this paper. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0801302 and in part by the National Natural Science Foundation of China under Grant 61202356.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[2] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Computer Networks*, vol. 91, no. C, pp. 528–547, Nov. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2015.08.040

[3] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks," *Journal of Parallel and Distributed Computing*, vol. 96, no. C, pp. 45–74, Oct. 2016. [Online]. Available: https://doi.org/10.1016/j.jpdc.2016.05.009

[4] J. Moura and D. Hutchison, "Review and analysis of networking challenges in cloud computing," *Journal of Network and Computer Applications*, vol. 60, pp. 113 – 129, 2016.

[5] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 640–656, Firstquarter 2017.

[6] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: insights and future directions," *IEEE Network*, vol. 25, no. 4, pp. 6–10, July 2011.

[7] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi, "A survey of wireless data center networks," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, March 2015, pp. 1–6.

[8] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1021–1036, Fourth 2012.

[9] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 909–928, Second 2013.

[10] I. Pietri and R. Sakellariou, "Mapping virtual machines onto physical machines in cloud computing: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 49:1–49:30, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2983575

[11] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Computer Communications*, vol. 40, pp. 1 – 21, 2014.

[12] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, Firstquarter 2016.

[13] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, U. S. Khan, A. Abbas, N. Jalil, and S. U. Khan, "A taxonomy and survey on green data center networks," *Future Generation Computer Systems*, vol. 36, pp. 189 – 208, 2014.

[14] B. Dai, G. Xu, B. Huang, P. Qin, and Y. Xu, "Enabling network innovation in data center networks with software defined networking: A survey," *Journal of Network and Computer Applications*, vol. 94, pp. 33 – 49, 2017.

[15] J. Son and R. Buyya, "A taxonomy of software-defined networking (SDN)-enabled cloud computing," *ACM Computing Surveys*, vol. 51, no. 3, pp. 59:1–59:36, May 2018. [Online]. Available: http://doi.acm.org/10.1145/3190617

[16] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojicic, C. Varela, R. Bahsoon, M. D. D. Assuncao, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, and H. Shen, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Computing Surveys*, vol. 51, no. 5, pp. 105:1–105:38, Nov. 2018. [Online]. Available: http://doi.acm.org/10.1145/3241737

[17] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, Jul. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9307-7

[18] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A taxonomy and survey of cloud resource orchestration techniques," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 26:1–26:41, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3054177

[19] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016.

[20] A. Thakur and M. S. Goraya, "A taxonomic survey on load balancing in cloud," *Journal of Network and Computer Applications*, vol. 98, pp. 43 – 57, 2017.

[21] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends," *Journal of Network and Computer Applications*, vol. 71, pp. 86 – 98, 2016.

[22] F. P. Tso, S. Jouet, and D. P. Pezaros, "Network and server resource management strategies for data centre infrastructures: A survey," *Computer Networks*, vol. 106, pp. 209–225, 2016.

[23] P. Poullie, T. Bocek, and B. Stiller, "A survey of the state-of-the-art in fair multi-resource allocations for data centers," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 169–183, March 2018.

[24] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol. 47, no. 4, pp. 63:1–63:33, Jul. 2015. [Online]. Available: http://doi.acm.org/10.1145/2788397

[25] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 954–1001, Secondquarter 2017.

[26] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Network*, vol. 27, no. 4, pp. 22–26, July 2013.

[27] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1492–1525, Secondquarter 2018.

[28] J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, "Load balancing in data center networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2324–2352, thirdquarter 2018.

[29] K. Wang, Q. Zhou, S. Guo, and J. Luo, "Cluster frameworks for efficient scheduling and resource allocation in data center networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3560–3580, Fourthquarter 2018.

[30] S. Wang, J. Zhang, T. Huang, J. Liu, T. Pan, and Y. Liu, "A survey of coflow scheduling schemes for data center networks," *IEEE Communications Magazine*, vol. 56, no. 6, pp. 179–185, June 2018.

[31] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 579–595, Feb 2017.

[32] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486012

[33] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 515–526. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626336

[34] Y. Chen, S. Jain, V. K. Adhikari, Z. L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! datasets," in *2011 Proceedings IEEE INFOCOM*, April 2011, pp. 1620–1628.

[35] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787478

[36] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: ACM, 2016, pp. 73–86. [Online]. Available: http://doi.acm.org/10.1145/2934872.2934893

[37] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486019

[38] L. Luo, H. Yu, Z. Ye, and X. Du, "Online deadline-aware bulk transfer over inter-datacenter wans," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 630–638.

[39] M. Noormohammadpour, C. S. Raghavendra, and S. Rao, "DCRoute: Speeding up inter-datacenter traffic allocation while guaranteeing deadlines," in *Proceedings of the 2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*, Dec 2016, pp. 82–90.

[40] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 74–85. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018446

[41] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Computer Networks*, vol. 68, pp. 123 – 137, 2014, communications and Networking in the Cloud.

[42] S. Su, Y. Wang, S. Jiang, K. Shuang, and P. Xu, "Efficient algorithms for scheduling multiple bulk data transfers in inter-datacenter networks," *International Journal of Communication Systems*, vol. 27, no. 12, pp. 4144–4165, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.2603

[43] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. C. M. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 112–125, Jan 2017.

[44] Y. Feng, B. Li, and B. Li, "Jetway: Minimizing costs on inter-datacenter video traffic," in *International Conference on Multimedia*, 2012.

[45] T. Nandagopal and K. P. N. Puttaswamy, "Lowering inter-datacenter bandwidth costs via bulk data scheduling," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 244–251. [Online]. Available: https://doi.org/10.1109/CCGrid.2012.70

[46] Y. Feng, B. Li, and B. Li, "Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, June 2012, pp. 43–50.

[47] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "Trafficshaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2018.

[48] W. Shi, C. Wu, and Z. Li, "A shapley-value mechanism for bandwidth on demand between datacenters," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 19–32, Jan 2018.

[49] W. K. Tan, D. M. Divakaran, and M. Gurusamy, "Uniform price auction for allocation of dynamic cloud bandwidth," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 2944–2949.

[50] Y. Li, H. Xie, and Y. Liao, "Blossom: Content distribution using inter-datacenter networks," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[51] M. Noormohammadpour, C. S. Raghavendra, S. Rao, and S. Kandula, "DCCast: Efficient point to multipoint transfers across datacenters," in *Proceedings of the 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*. Santa Clara, CA: USENIX Association, 2017.

[52] Y. Feng, B. Li, and B. Li, "Airlift: Video conferencing as a cloud service using inter-datacenter networks," in *Proceedings of the 2012 20th IEEE International Conference on Network Protocols (ICNP)*, ser. ICNP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–11. [Online]. Available:

http://dx.doi.org/10.1109/ICNP.2012.6459966

[53] L. Luo, H. Yu, and Z. Ye, "Deadline-guaranteed point-to-multipoint bulk transfers in inter-datacenter networks," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[54] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford, "Optimizing bulk transfers with software-defined optical wan," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: ACM, 2016, pp. 87–100. [Online]. Available: http://doi.acm.org/10.1145/2934872.2934904

[55] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, K. N. B., C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, S. Padgett, F. Rabe, S. Ray, M. Tewari, M. Tierney, M. Zahn, J. Zolla, J. Ong, and A. Vahdat, "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: ACM, 2018, pp. 74–87. [Online]. Available: http://doi.acm.org/10.1145/3230543.3230545

[56] N. E. Young, "Sequential and parallel algorithms for mixed packing and covering," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, Oct 2001, pp. 538–546.

[57] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 145–158. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015484

[58] N. Laoutaris and P. Rodriguez, "Good things come to those who (can) wait or how to handle delay tolerant traffic and make peace on the Internet," in *Proceedings of ACM HotNets-VII*, 2008.

[59] N. Laoutaris, G. Smaragdakis, R. Stanojevic, P. Rodriguez, and R. Sundaram, "Delay-tolerant bulk data transfers on the Internet," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1852–1865, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1109/TNET.2012.2237555

[60] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, "Cost-minimizing bandwidth guarantee for inter-datacenter traffic," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[61] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "More peak, less differentiation: Towards a pricing-aware online control framework for inter-datacenter transfers," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2105–2110.

[62] D. K. Goldenberg, L. Qiuy, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing cost and performance for multihoming," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 79–92. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015478

[63] L. S. Shapley, "A value for n-person games," in *Contributions to the Theory of Games II*, H. W. Kuhn and A. W. Tucker, Eds. Princeton: Princeton University Press, 1953, pp. 307–317.

[64] S. Dobzinski, A. Mehta, T. Roughgarden, and M. Sundararajan, "Is shapley cost sharing optimal?" in *Algorithmic Game Theory: First International Symposium*, B. Monien and U.-P. Schroeder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 327–336.

[65] I. Kremer and K. G. Nyborg, "Underpricing and market power in uniform price auctions," *Review of Financial Studies*, vol. 17, no. 3, pp. 849–877, 2004.

[66] Y. Li, L. Zhang, Y. Jia, Y. Liao, and H. Xie, "Calantha: Content distribution across geo-distributed datacenters," in *Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 724–729.

[67] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: A low-delay multi-party conferencing solution," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 155–164, September 2013.

[68] M. Noormohammadpour, C. S. Raghavendra, S. Kandula, and S. Rao, "QuickCast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts," in *Proceedings of Infocom 2018*, 2018.

[69] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[70] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. L. Woodward, and J. Yates, "Bandwidth on demand for inter-data center communication," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 24:1–24:6. [Online]. Available: http://doi.acm.org/10.1145/2070562.2070586

[71] S. J. B. Yoo, Y. Yin, and K. Wen, "Intra and inter datacenter networking: The role of optical packet switching and flexible bandwidth optical networking," in *2012 16th International Conference on Optical Network Design and Modelling (ONDM)*, April 2012, pp. 1–6.

[72] Y. Yin, L. Liu, R. Proietti, and S. J. B. Yoo, "Software defined elastic optical networks for cloud computing," *IEEE Network*, vol. 31, no. 1, pp. 4–10, January 2017.

[73] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Network*, vol. 29, no. 5, pp. 36–42, September 2015.

[74] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *Proceedings - IEEE INFOCOM*, 03 2012, pp. 846–854.

[75] M. Allalouf and Y. Shavitt, "Centralized and distributed algorithms for routing and weighted max-min fair bandwidth allocation," *IEEE/ACM Transactions on Networking*, vol. 16, no. 5, pp. 1015–1024, Oct 2008.

[76] S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo, "Explicit path control in commodity data centers: Design and applications," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2768–2781, October 2016.

[77] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 539–550. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626307

[78] A. Lodi and G. Zarpellon, "On learning and branching: a survey," *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 25, no. 2, pp. 207–236, July 2017.

[79] N. Rosenfeld, E. Balkanski, A. Globerson, and Y. Singer, "Learning to optimize combinatorial functions," in *ICML*, 2018.