Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 7209414, 17 pages
https://doi.org/10.1155/2023/7209414

Research Article

DIT: A Dynamic Bandwidth Isolated Transmission System for Large-Scale Inter-DC Wireless Communication Network

Shiyan Zhang, Yuchao Zhang, Ran Wang, Xiangyang Gong, and Yongping Xiong

Beijing University of Posts and Telecommunications, Beijing 100089, China

Correspondence should be addressed to Shiyan Zhang; zshiyan@bupt.edu.cn

Received 6 July 2022; Revised 10 August 2022; Accepted 19 September 2022; Published 20 February 2023

Academic Editor: Lei Liu

Large companies are increasingly providing their services over interdomain data centers. Such large-scale wireless communication networks carry various applications, including both online time-sensitive services and offline bandwidth-sensitive services. These applications share the same physical links but have different bandwidth requirements at different time, making the coscheduling problem a key challenge in improving the overall network utility. Most existing commercial systems reserve peak value bandwidth for online time-sensitive services to ensure the quality of service and use the leftover bandwidth to transmit offline data in a best-effort manner. Such systems cannot obtain high link utilizations because of the nonstationarity of online traffic. To solve this problem, we present a dynamic isolated transmission (DIT) system in this paper, where a sliding-*k* algorithm is designed to predict online traffic, and a bottleneck bypass routing scheme is proposed to schedule the mixed traffic in a shared inter-DC wireless communication network. We conduct a series of experiments in an experimental inter-DC wireless communication network to analyze for efficient network operation of the DIT and compare its performance with typical existing fix-bandwidth-based solutions including Microsoft's SWAN. The results show that DIT outperforms existing solutions, improving the interdomain link utilization by 18% and the intradomain link utilization by 65%.

1. Introduction

With the rapid development and widespread application of cloud computing, the construction of global infrastructure is accelerating. The development of cloud computing has stimulated the rapid deployment of interdata center (DC) wireless communication networks, which provide efficient services by connecting geographically distributed DCs [1]. Large-scale Internet service providers, such as Google, Microsoft, Facebook, Baidu, and Alibaba, have commenced the large-scale deployment and operation of geographically distributed DCs to provide better user experience [2].

To improve link utilization, online and offline applications are typically mixed-deployed in a physical inter-DC network [3]. Although the amount of online traffic is moderate, the requirement of effectiveness is high, and traffic bursts occur frequently [4]. On the contrary, the amount of offline traffic is high, while the requirement of timeliness is relatively relaxed. With the large-scale application of cloud technologies and the widespread construction of DCs, enterprise-side information communication technology (ICT) infrastructure is transitioning from the enterprise side to the DC, reflecting the excellent advantages and improved resource utilization of ICT infrastructure to a certain extent. However, the overall inter-DC link utilization of wide-area networks (WANs) remains quite low [5]. In typical cases, even for very busy links, the average utilization rate is only 40% to 60% [6]. Transmitting data in time and meeting the expected service deadline are the main challenges for data transmission through inter-DC WANs.

Although there are many mature mixed-deployment systems, such as Microsoft's SWAN [6] and Google's B4 [7], most of them adopt a fixed-bandwidth separation method [8]. These scheduling schemes can be classified into three categories. The most widely used method for allocating bandwidth in the spatial dimension is the max-min fairness, whereby flow weights are adjusted depending on their priorities [6, 9]; in the time dimension, some systems [10, 11] reserve the bandwidth for future resources; in the storage dimension, some systems [12, 13] adopt the store-andforward method, since most current systems are separated by fixed bandwidth. When online traffic is idle at a low valley, excess bandwidth cannot be released to offline traffic.

The fixed bandwidth separation method is where a threshold is set for the bandwidth that can be utilized by the offline traffic, and the bandwidth that can be occupied by the offline traffic is limited to this range, so that even if there is any spare bandwidth allocated to the online traffic, the offline traffic cannot exceed the range of this threshold to utilize the spare part of the bandwidth. As a result, there is a significant waste of bandwidth. In this study, we aimed to design a dynamic bandwidth isolation transmission system to fully utilize inter-DC links. The simplified core idea is shown in Figure 1. The grey area with bursts indicates online traffic while the green blocks represent offline traffic. An ideal scheduling algorithm should accurately predict the amount of online traffic and then allocate the remaining bandwidth to offline traffic. Thus, the bandwidth reserved for online traffic will not be wasted even when online traffic is in its valley.

To achieve this, we propose a transmission system based on dynamic bandwidth isolation, called the dynamic isolated transmission (DIT) system. In this method, instead of setting a fixed-bandwidth upper limit for offline traffic, the DIT system adjusts the available bandwidth dynamically. With the help of a sliding-k prediction algorithm [14], the system predicts the amount of online traffic in the next cycle and then allocates the remaining bandwidth to the offline traffic.

DIT not only takes into account the elasticity of network bandwidth allocation for offline and online traffic but also maximizes the utilization of inter-DC bandwidth. We dynamically explore the set of paths available in the distributed data center wireless communication network based on traffic demands, including the direct shortest path and other paths with higher bandwidth. By elastically setting the congestion threshold, we dynamically choose bottleneck-bypass mechanisms to fully utilize the inter-DC links.

The following are the contributions of this paper:

(1) A dynamic bandwidth isolation system for large-scale inter-DC data transmission is proposed

(2) A bottleneck bypass scheduling algorithm is designed to efficiently transmit large amounts of offline traffic

(3) Demonstrating the practical benefits of DIT by a real-world pilot deployment in Huawei. DIT system has improved interdomain link utilization by 18% and intradomain link utilization by 65%

The rest of this paper is structured as follows. Section 2 presents a brief overview of related works. Background and problem formulation are given in Section 3. The proposed DIT is presented in Section 4. Section 5 proposes scheduling and routing algorithms. Section 6 gives experimental setup. Section 7 demonstrates the experimental evaluation. At last, Section 8 concludes the paper.

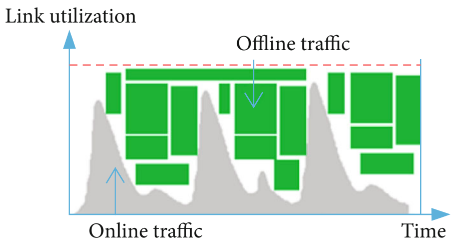


Figure 1: Application layer scheduling.

2. Related Work

Traffic prediction and scheduling schemes for inter-DCs have been widely studied. However, none of them can directly solve our problems in DIT. Therefore, we analyze the current research results from the framework design and scheduling objectives.

2.1. Centralized Architecture. The TE of traditional distributed routing is suboptimal in most cases. For optimal scheduling, centralized scheduling must be used. Google implements global scheduling through centralized TE. Therefore, centralized scheduling of the bandwidth is feasible. In B4 [7], a software-defined network (SDN) gateway integrates the topology from multiple sites into the TE controller. Different application requirements were collected by a bandwidth enforcer (BwE) [9] and then submitted to the controller. With this global information, a centralized controller allocates bandwidth in a max-min fairness manner, which is performed from a global perspective. Baidu pointed out that because of the lack of global information, local scheduling decisions that depend on each server are usually suboptimal. Therefore, Baidu proposed BDS [8], which is a highly centralized structure that maintains the data transmission state of the agent server through a central controller in real time and updates global information in near real-time to respond to dynamic performance changes and routing decisions. Such a global scheduling system is complicated. However, BDS decouples the control algorithm into two parts, namely, scheduling and routing, thereby reducing the computational overhead of the centralized control. The system selects a subset of duplication data in the scheduling step, and the subsequent routing step only considers these selected subsets, thus significantly reducing the search space and making global scheduling more efficient. As the SDN gradually becomes a new specification, it transfers the network control from the distributed protocol to the control plane of logic centralization, and gets increasing attention by virtue of flexible network management and rapidly supports the deployment of new functions. Most schemes are using the SDN paradigm; SWAN [6] collects the network topology and the global view of the traffic request through the SDN and then reconfigures the network data plane to match the current traffic requirements depending on the SDN specifications. The centralized controller directly updates the forwarding status of the switch, while controlling the traffic and flow rate of each service, thereby realizing more bandwidth and increasing the traffic carried by the inter-DC WAN. [15] proposed an SDN-based traffic management system. Centralized control is mainly reflected in collecting network events and traffic statistics reported by OFA-switches through traffic management (TM) to obtain global network status information. TM allocates bandwidth for flow groups (FG). [16] designed an Amoeba system, which consists of a controller and site brokers in a logical set. The controller is responsible for maintaining global information on the network bandwidth and interactive traffic requirements and performing space-time resource allocation. Each DC has a site broker, which regularly predicts and reports to the controller the interactive flow demand of the local DC and realizes the decision of the controller by coordinating the bandwidth execution. [17] presented a hybrid SDN-MPLS architecture in the data center network, with the rerouting procedure to achieve the desired network activity.

2.2. Distributed Architecture. Autopilot [18] is Microsoft's infrastructure for automated DC management. Autopilot is a distributed system, and the components do not depend on each other. It includes functional modules for automated software provisioning and deployment, system monitoring, and performing maintenance operations. The two most important principles of Autopilot design are fault tolerance and simplicity: the objective of fault tolerance is to minimize the dependencies between components and prevent a temporary single point of failure from affecting the entire cluster. IP anycast is the foundation of some critical network infrastructures; however, its efficiency has long been in question. [19] proposed passive and active measurement methods and analyzed the inefficiency in the deployment of root DNS server IP anycast. With a part of the geographical indications in BGP loaded as prompts, more information was provided for the routing process to make path selection among multiple paths of equal length. This approach adopts a typical distributed idea, but it is difficult to advance the large-scale deployment smoothly due to the need to modify the protocol. Some methods have been proposed in the field of P2P research to realize a balance between maximizing link utilization and ensuring the fairness of multicast sessions. [20] designed an application-layer multicast system, called SplitStream, which uses a forest of multicast trees (a group of multicast trees) to distribute data to balance the forwarding load and achieve fault tolerance. Experiments showed that this method outperforms other application-layer multicast systems in increasing the bandwidth of data transmission. [21] proposed a dual distributed algorithm to maximize the performance of multicast trees in multipath scenarios. The fairness between different applications can be achieved by selecting different performance functions for the applications. A general trustworthiness-based evaluation scheme (TVRS) [22] is proposed to analyze vehicle trust and obtain high-quality information. TVRS is studied for data sensing in dynamic distributed networked systems (DNS). Almadani et al. [23] proposed the distributed SDN control plane framework (DSF)—a framework for the East/West interface for heterogeneous, distributed SDN controllers that used a standardized data-centric real-time publish/subscribe paradigm (called Data Distribution Service (DDS)) to synchronize topologies. [24] presented a complete software model of a software-defined data center (SDDC) that provided three different functions such as controller location, routing, and load balancing through different RL-agents in the SDDC.

2.3. Improve Network Utilization. Google realizes that the overconfiguration method makes the average utilization rate of WAN links usually 30%-40%, which will cause a huge waste of bandwidth. To improve bandwidth utilization, Google adopted the centralized TE method in B4 [7]. B4 controls the network edge, utilizes the available network capacity by multipath forwarding based on the application priority, and dynamically redistributes the bandwidth in case of link/switch failure or application demand change. These features enable many links to achieve 100% utilization, and all links can achieve an average utilization of 70% over a long period of time, with a 2-3-time efficiency improvement compared to the current state. TE shares bandwidth for multiple applications among multiple paths that may be used. Its objective is to realize max-min fairness allocation by application, which can achieve fairness on the premise of maximizing link utilization. To achieve max-min fairness, B4 designed its own TE optimization algorithm, including tunnel group generation and tunnel group quantization. Calendaring [10] reserves resources in advance for WAN traffic. The objective of Tempus is to satisfy the transmission requests of long-period flows, while leaving sufficient bandwidth for high-priority flows. Tempus utilizes the LP algorithm to maximize the number of tasks completed before the deadline and maximize total network utilization. [15] proposed the idea of multiclass QoS-guaranteed IDC traffic management model (MCTEQ), which can ensure end-to-end delay of high-priority traffic by allocating bandwidth jointly for traffic of different priority levels. For traffic of the same priority, the piecewise linear approximation of the utility function is used to equitably distribute the flow rate and finally achieve higher bandwidth utilization. Considering the heterogeneous needs of elephant flow requests and mouse flow requests in a data center networks. Sun et al. [25] proposed an efficient orchestration algorithm for online service function chains (SFC) requests, reducing the delay experienced by small flows while improving the acceptance ratio for user requests. The work [26] provided flow splitter (FS), a deep reinforcement learning- (DRL-) based flow scheduler which enables hybrid optical-electrical switching-based data center network (HOE-DCN) to make instant flow scheduling according to the runtime network conditions.

The DIT is a pragmatic and challenging solution compared to existing works. DIT is designed to be a centralized control system that maximizes the use of network resources. Simultaneously, DIT optimizes the inter-DC multicast based on the predicted remaining bandwidth resources of each link on the available paths. DIT develops and integrates all required operations into a unified platform, allowing them to interact and optimize each other to improve the overall resource utilization of the inter-DC network.

3. Background and Problem

We first present insights into the dynamic deployment of hybrid traffic on interdata center networks—dynamic bandwidth isolation mechanism, then initialize some basic definitions of interdata center networks, and finally, consider the scheduling and routing issues of overlay networks introducing the idea of bottleneck links and formulating a set of constraints.

3.1. Background. The traffic in inter-DCs can be characterized as a hybrid deployment of rigid traffic (online traffic) and multicast elastic traffic (offline traffic). The first is a delay-sensitive high-priority traffic; although the proportion of total traffic is small, it needs to be fully met. The second is a delay-insensitive traffic of large-scale transmission, which needs to be completed before the deadline. The DIT system we designed is a mixed deployment system of online and offline. Here, a custom sliding k algorithm is designed for online traffic prediction. The offline scheduling in DIT refers to elastic traffic scheduling. When scheduling, we need to consider the bandwidth limitation of the dedicated line [27] and the unschedulable rigid traffic in the dedicated line. For rigid traffic, we allow the elastic traffic to be sent at a time when the rigid traffic is low, to minimize the peak-to-average ratio of the flow on the line, thereby delaying the line expansion. This process is called traffic reshaping [28].

To prevent delay-sensitive flows from being negatively affected by burst bulk-data transmission, sliding k is used to design and respond to sensitive network changes. In other words, when online traffic is oscillating (sensitive), more attention is paid to any sudden increase or decrease, while referring to the information obtained when historical online traffic remained unchanged (stable). When there is a sudden increase or decrease in online traffic in the network, it is clear that the more recent observations should obviously be assigned a higher weight, while the older observations should be assigned a lower weight. The DIT traffic prediction scheme for dynamic bandwidth separation of elastic traffic can no longer use a bandwidth with a fixed threshold. In real time, for elastic flow adjustment of the available bandwidth, the rigid flow value is constantly predicted, and the scheduling decisions are automatically adjusted on the basis of the forecasted results, flexible response to the network environment, and make full use of bandwidth. When rigid traffic occupies more bandwidth and reaches its peak, the dynamic bandwidth separation mechanism responds to this, making elastic traffic reduce its occupied bandwidth and avoid congestion. When the rigid flow occupies less bandwidth and reaches its valley, the dynamic bandwidth separation mechanism will allocate more bandwidth to the elastic traffic and make full use of the remaining bandwidth. Because the dynamic bandwidth separation mechanism can make full use of the remaining bandwidth, elastic traffic can take up more bandwidth, thus reducing the time required for elastic traffic transmission.

3.2. Preliminary. Let $G = (V, E)$ be a directed acyclic graph, which is used to represent the interdata center network. V denotes the node set, where each node $v \in V$ can be a data center. E denotes the link set, where each $e_{st} \in E$ represents the virtual link from source data center s to destination data center d with corresponding link capacity C_{sd}. We assume that the system consists of some users, n core-DC agents, m severs, and a system controller. The set of core-DC agents is denoted by a vector $M = \{M_1, M_2, \cdots, M_n\}$. The set of severs is denoted by a vector $N = \{N_1, N_2, \cdots, N_m\}$.

Considering the centralized strategy used in the model, we assume that each core-DC agent independently assigns its tasks to the server according to the probability distribution. Thus, core-DC agent M_i should find probability ρ_{ij} assigned tasks to server j to minimize the expected execution time of its tasks. The probability of agent M_i is denoted by vector $\rho_i = \{\rho_{i1}, \rho_{i2}, \cdots, \rho_{im}\}$, and $\sum_{j=1}^{m} \rho_{ij} = 1$. We assume that on a particular link e, A transfer represents a Tenants' data delivery demand from the source data center to the destination data center, which is specified as a tuple $f_i = \{D_i, T_i^s, T_i^d, v_i\}$, where D_i is the traffic volume, T_i^s is the transmission starting time, T_i^d is the deadline, and v_i is the expected transfer rate. v_i can be obtained by

$$v_i = \frac{D_i}{T_i^d - T_i^s}. \tag{1}$$

Further, at a specific time slot t, we organize all existing transfers that are waiting to be transmitted from the source sever as a set F_t and denote the total of transfers as $|F_t|$. Similarly, the transmitting transfer set is represented as $\mathbb{F}_t$, and we use B_t to represent the bandwidth these transfers have occupied, which is calculated as

$$B_t = \sum_{f_i \in \mathbb{F}_t} v_i. \tag{2}$$

Once ρ_{ij} is determined, core-DC agent i will send tasks to server j at a rate $B_{ij}(t)$ given by $B_{ij}(t) = B_i(t)\rho_{ij}(t)$. Moreover, the available processing capacity of each server for core-DC agent M_i is denoted by vector $C_i = \{c_i^1, c_i^2, \cdots, c_i^m\}$, where $c_i^j(t) = c_j(t) - \sum_{k=1,k\neq i}^{n} \rho_{kj}(t) B_k(t)$ indicates the available bandwidth from core-DC agent i to server j.

3.3. Problem Formulation and Constraints. Regarding the scheduling and routing problems of the overlay network, Table 1 summarizes the key symbols used and the physical meaning of the symbols.

We consider a case where the sending node s (source) wishes to send a data block with data file size F to the receiving node d (destination). To take advantage of the overlay path between the source and destination DCs, each data file is split into multiple data blocks and transmitted in batches. The DIT updates the scheduling decision every scheduling period ΔT. Based on the network conditions in the current cycle, an appropriate scheduling method is used to schedule the elastic traffic so that the elastic traffic can make full use of the remaining bandwidth of the available links and improve the utilization of the links.

Table 1: Summary of parameters and their descriptions.

Network model input parameters	
F	The size of a data file
s	Source node
d	Destination node
ΔT	Scheduling cycle
total_traffic	Networking information, that is, the region to which each link belongs, and the total bandwidth of each link
$P_{(s,d)}$	Set of paths between source and destination DC pairs
p	Specific path p between source and destination DC pairs
l	Link l on the path
$\text{onup}_{s,sc}$	In each cycle, the amount of rigid traffic in the uplink direction (DC to supercore direction)
$\text{ondo}_{dc,d}$	In each cycle, the amount of rigid traffic in the downlink direction (supercore to DC direction)
$\text{onre}_{i,j}$	In each cycle, the rigid flow of each link between supercores
Output variables	
$\text{preup}_{s,sc}$	In each cycle, the prediction of the uplink is allocated to the bandwidth of the rigid traffic
$\text{predo}_{dc,d}$	In each cycle, the prediction of the downlink is allocated to the bandwidth of the rigid traffic
$\text{prere}_{i,j}$	In each cycle, the prediction of each link between supercores is allocated to the bandwidth of the rigid traffic
$\text{realup}_{s,sc}$	The actual amount of remaining traffic on the uplink in each cycle
$\text{realdo}_{dc,d}$	The actual amount of remaining traffic on the downlink in each cycle
$\text{realreg}_{i,j}$	The actual remaining traffic of each link between supercores in each cycle
Internal and decision variables for epoch t	
$U_{s,sc}(t)$	The amount of traffic that the uplink can send in time t
$D_{dc,d}(t)$	The amount of traffic that the downlink can receive in time t
$RP_{\text{traffic}}(t)$	Predicted remaining traffic in time t
$R_{\text{traffic}}(t)$	The remaining traffic in time t
$B_p(t)$	Bandwidth allocated for data blocks sent on path p in time t
$C_l(t)$	Link capacity in time t

We need to find the bottleneck link. The bottleneck link includes both intradomain and interdomain bottlenecks. $U_{s,sc}(t)$ and $D_{dc,d}(t)$ represent the traffic that can be sent by the uplink and the traffic that can be received by the downlink in time t, respectively, which we call intradomain bottlenecks. $P_{sc,mc}(t)$, $P_{mc,dc}(t)$, and $N_{sc,dc}(t)$ represent the traffic that can be sent by the interdomain link in time t. The first two items refer to the interdomain bottlenecks of the bypass scheme, and the third item refers to the nonbypass interdomain bottlenecks. Regardless of whether the bottleneck link is an intradomain bottleneck or an interdomain bottleneck, we need to compare the remaining bandwidth of all the interdomain links that bypass and not bypass, find the interdomain bottleneck link, and choose a solution without interdomain bottlenecks.

Input: the input parameters of the DIT are as follows:

Networking information (bandwidth): is total_traffic, including the total bandwidth of the Core-DC, and the total bandwidth of each DC to which each Core-DC belongs.

Original rigid traffic: in which $\text{onup}_{s,sc}$, $\text{ondo}_{dc,d}$ and $\text{onre}_{i,j}$ denote the rigid traffic on the uplink, downlink, and interdomain link, respectively, in each cycle.

Demand information for elastic traffic (demand.txt): including demand number, start time, end time, source IDC, destination IDC, demand traffic size, schedule update interval ΔT.

Output: the output of DIT is as follows:

The kth period T_k, the bandwidth allocation of the predicted rigid traffic: $\text{preup}_{s,sc}$, $\text{predo}_{dc,d}$ and $\text{prere}_{i,j}$ represent the predicted bandwidth allocated to rigid traffic in each link in each cycle for uplink, downlink and interdomain links, respectively.

Calculate the actual remaining bandwidth that can be allocated to elastic traffic, including $\text{realup}_{s,sc}$, $\text{realdo}_{dc,d}$ and $\text{realreg}_{i,j}$, which represent the actual remaining traffic on the uplink, downlink, and interdomain link in each cycle, respectively. Therefore, the actual traffic can be compared with the predicted traffic.

The kth period T_k, the total utilization of each link (rigid traffic plus elastic traffic).

The kth period T_k, the scheduling and routing results of each link's elastic traffic. Scheduling refers to the scheduling of elastic traffic, that is, the amount of data in elastic traffic transmitted on each link in each cycle. Routing refers to the selection of the transmission path of the data blocks scheduled in each cycle, and whether to bypass or not is judged based on the remaining bandwidth of the bottleneck link of the optional path.

3.3.1. Constraint. Regarding the bypass scheme, we need to find the interdomain bottleneck link of the bypass scheme, i.e., we need to compare the predicted residual traffic values of each interdomain link on the bypass path. Because the scheduling process is based on the predicted residual traffic, the segment of the interdomain link that finds the lowest predicted residual traffic is used as the bottleneck of the bypass scheme, that is:

$$P_{\text{bottleneck}(i)}(t) = \min\left(P_{sc,mc(1)}(t), \cdots P_{mc(n-1),mc(n)}(t), P_{mc(n),dc}(t)\right). \tag{3}$$

In each cycle, regardless of the scheme used, finding the interdomain bottleneck link requires calculating the remaining traffic of each interdomain link. This in turn requires comparing the predicted remaining traffic of each interdomain link with the actual remaining traffic and then finding the lowest value as the remaining traffic (the bottleneck link), i.e.,

$$RP_{\text{traffic}}(t) = \text{total\_traffic}(t) - \text{predict}(t), \tag{4}$$

$$R_{\text{traffic}}(t) = \min\left(RP_{\text{traffic}}(t), \text{real}(t)\right). \tag{5}$$

The bandwidth $B_p(t)$ allocated on path p must not exceed the capacity of any link l in p, i.e.,

$$B_p(t) \leqslant \min\left(C_{s,sc}(t), C_{sc,mc(1)}(t), \cdots C_{mc(n-1),mc(n)}(t), C_{mc(n),dc}(t), C_{dc,d}(t)\right). \tag{6}$$

For all paths through a link, the total allocated bandwidth of this link should not exceed its capacity.

$$C_l(t) \geqslant \sum_{p\in P} B_p(t), \quad \forall l \in p. \tag{7}$$

When $D_{\text{traffic}}(\Delta T) \leqslant R_{\text{traffic}}(\Delta T)$, the data blocks of the elastic traffic $D_{\text{traffic}}(\Delta T)$ selected to be sent in each cycle can be transmitted within ΔT, and the predicted and actual remaining traffic are updated.

$$\begin{cases} D_{\text{traffic}}(\Delta T) = 0, \\ RP_{\text{traffic}}(\Delta T) = RP_{\text{traffic}}(\Delta T) - D_{\text{traffic}}(\Delta T), \\ \text{Real}(\Delta T) = \text{Real}(\Delta T) - D_{\text{traffic}}(\Delta T). \end{cases} \tag{8}$$

When $D_{\text{traffic}}(\Delta T) > R_{\text{traffic}}(\Delta T)$, the data blocks of the elastic traffic $D_{\text{traffic}}(\Delta T)$ that is selected to be sent in each cycle cannot be transmitted within ΔT. We need to update the elastic traffic demand and the predicted and actual remaining traffic; thus, the remaining elastic traffic can continue to be transmitted in the next cycle.

$$\begin{cases} D_{\text{traffic}}(\Delta T) = D_{\text{traffic}}(\Delta T) - R_{\text{traffic}}(\Delta T), \\ RP_{\text{traffic}}(\Delta T) = RP_{\text{traffic}}(\Delta T) - R_{\text{traffic}}(\Delta T), \\ \text{Real}(\Delta T) = \text{Real}(\Delta \text{T}) - R_{\text{traffic}}(\Delta T). \end{cases} \tag{9}$$

All elastic traffic needs to be transmitted before the deadline. If not, they will be discarded.

$$D_{\text{traffic}}(t) \leqslant \sum_{k=1}^{N} \sum_{p\in P} B_p(T_k)t \leqslant \text{deadline}_{\text{demand}}. \tag{10}$$

3.3.2. Objective Function.

$$U = \frac{\sum_{k=1}^{N} \sum_{p\in P} B_p(T_k)}{C_l}. \tag{11}$$

The link utilization of link l during a period of time is calculated using Equation (11). Maximizing link utilization as proposed in this paper refers to allocating the maximum available bandwidth resources to a network deployed with a mix of online and offline traffic. This means that the bandwidth resources allocated to online and offline traffic on link l as a percentage of the bandwidth capacity of link l is maximized.

4. DIT System Overview

In this section, we elaborate on the design of the DIT system in detail. It mainly involves the centralized architecture, the design objectives, and the description of the bypass scheme.

4.1. Centralized Architecture. The DIT system design is a centralized control overlay system, including DIT controller, agent monitor, agent, and network agent. The objective of DIT is to maximize the utilization of network resources and optimize inter-DC multicasts based on the remaining bandwidth resources on each link of the predicted available path. To make full use of overlay path optimization scheduling and improve link utilization, batch data transmission is dynamically optimized through time-varying networks. The DIT system uses a centralized controller to obtain the latest global link information. The traditional distributed solution relies on the local reactive decision of a single server. It is difficult to explore all the available coverage paths, because of the inability to obtain global information, and may result in locally optimal scheduling decisions.

The real-time centralized scheduling method needs to maintain global state information, which has a large computational overhead, and the transmission of large amounts of data usually takes several tens of seconds. To achieve near-optimal routing and scheduling based on a global perspective, the slight delay caused by this can be accepted. The overall architecture of the DIT is shown in Figure 2.

Global information is stored and maintained in the centralized control module. First, the demand information of the elastic flow is inputted, and the scheduling task, demand number, start time, end time, source IDC, destination IDC, and size of demand flows are initialized. Second, it needs to store networking information (total_traffic), i.e., the region to which each link belongs, and the total bandwidth of each link. Networking status information, including network uplink, downlink, and interdomain link usage. If it is a rigid traffic amplification mode, the rigid traffic amplification will be performed first, and then, the utilization of each link will be counted. When performing network topology task allocation based on the routing and scheduling decisions of the controller, considering the interaction between the agent and the network topology and actual data transmission, the bandwidth is allocated for each data transmission through the agent.

We must continuously maintain the rigid traffic, actual remaining traffic, and predicted remaining traffic on each link in the current cycle. When the centralized control module is started, the demands will be loaded into the system, and the centralized controller will maintain the demands table through the scheduling of the system. If the parameters in the network configuration indicate that the current mode is the amplification mode, when the centralized control module is started, the network must be amplified, i.e., the

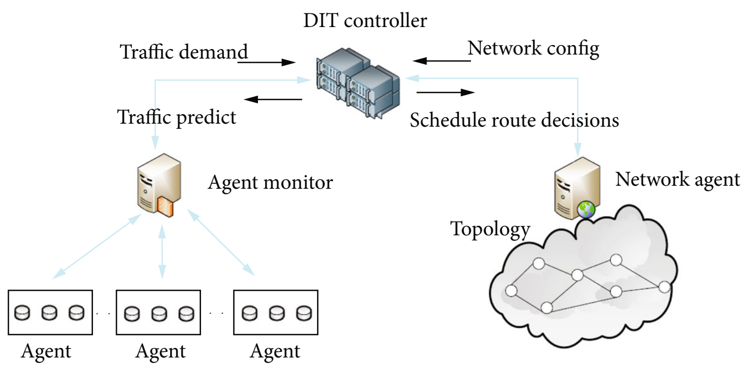


FIGURE 2: DIT Framework.

value of total_traffic is amplified. After each period of scheduling is completed, the centralized control module will calculate the link utilization of each link and the completion of each demand and record it.

Module description of DIT architecture components is as follows.

4.1.1. Centralized Control. The traffic information is read from the agent monitor, including the original rigid traffic and elastic traffic demand, and the predicted remaining traffic, which is the estimated remaining traffic of a certain link calculated by the prediction algorithm in a certain period. The network configuration and WAN topology are obtained from the network agent, and the maintenance traffic needs are updated in real time as the scheduling is unfolded. The global information required for elastic traffic scheduling is obtained, scheduling decisions are calculated, and finally, scheduling and updates are performed. The routing decision determines whether bypass is required and then updates the remaining traffic of the link.

Our DIT controller is a customized controller. First, DIT controller obtains source block_id, block_num, server_status, and other information from the agent monitor and predicts the collected rigid traffic. Then, the DIT controller plans the subtasks based on the network status, existing task status, link capacity, and other information. If the subtask selects a source that is not schedulable within the flow completion time (and the allocated bandwidth of the subtask cannot meet the minimum bandwidth requirement), set the schedule value of the subtask to true and inform the agent. When the DIT controller captures the subtask that needs to be scheduled, if it is the scheduling of a source that cannot be scheduled within the flow completion time, the DIT controller will check the remaining fragments of the subtask and whether there is available bandwidth. When scheduling subtasks with insufficient bandwidth, the DIT controller checks the remaining fragments of the subtasks and whether it can produce a scheduling result that meets the deadline. After a period of time, when no new scheduling results are generated, the DIT controller does not make any modifications to the subtasks. As the new scheduled task arrives, the DIT controller updates the remaining fragments of the subtasks and generates new subtasks. When the subtask completes the transmission of all the fragments, the DIT controller modifies the subtask status to complete, performs persistent storage, and deletes it from the memory. Finally, when the task is completed, the DIT controller changes the task status to complete, performs persistent storage, and deletes it from the memory.

4.1.2. Agent Monitor. Used to poll the status of the agent and task completion and update the status of the Agent and its running tasks to the DIT controller, the DIT controller obtains the configuration that needs to be updated and delivers it to the agent.

The agent monitor obtains the tasks and subtasks that need to be updated from the DIT controller and sends them to the agent. Subsequently, the agent monitor polls the subtasks waiting to be scheduled and at the same time updates the status of the subtasks to the DIT controller. When the agent monitor obtains the update of the subtask, it informs the agent. Once the agent monitor has polled all the agents, it informs the DIT controller of all the updated statuses, and the DIT controller knows that the subtask has been completed. Finally, the agent monitor informs the agent that the subtask can be deleted. The release process of the task is the same as the release process of the subtask.

4.1.3. Agent. Receiving instructions from the DIT controller, each agent generates traffic requirements and runs locally, checking the local status, including the data transfer status (which data blocks have arrived, which data blocks have not been completed) and the availability of the agent server and disk failure. Agent represents the management software running on each server. The agent controls the local management information which uses Linux Traffic Control (tc) to force a limit on the total bandwidth usage of inter-DC multicast traffic.

Agent submits new task requirements to DIT controller through agent monitor. At the same time, the agent continuously records tasks and subtasks and then establishes connections with other agents or sources based on the subtask information to initiate transmission. For the subtask with schedule=true, the agent will generate a callback message to the DIT controller whenever the specified number of steps (schedule_step) is completed. Subsequently, the agent stops the transmission of the subtask (DISCUSS: continue to transmit the next schedule_step fragments to avoid wasting the bandwidth in the waiting time, but only one schedule_ step can be transmitted). However, if the agent is told to continue the subtask, it will transmit according to the original fragment order. When a subtask update is encountered, the transmission is performed according to the modified subtask. Once the agent checks that all the fragments of a task have been completed, the status of the task is set to complete.

4.1.4. Network Agent. Tracking topology and network resource utilization: information about topology changes is sent to the DIT controller every $\Delta T = 5$ s, and information is collected and reported on traffic to the relevant topology. The network agent is also responsible for reliably updating and switching to different network topologies as required by the controller.

The network agent dynamically obtains the information of each link of the topology file, including the total traffic of each link of intra-DC (total_traffic), inter-DC link traffic (regions_traffic), upload traffic (up_traffic), and download traffic (down_traffic).

4.2. Design Objectives. We discuss our design objectives and summarize their implementation.

4.2.1. Effective Usage of Remaining Bandwidth. In the case of mixed deployment scenario, online traffic and offline traffic share the same bandwidth resources. When a fixed bandwidth separation strategy is adopted, a threshold is set for the bandwidth available for elastic traffic. The amount of bandwidth that can be occupied by elastic traffic is limited to this range. Even if there is a surplus of bandwidth allocated to rigid traffic, elastic traffic cannot exceed this threshold to use the remaining traffic. As a result, a large amount of bandwidth is wasted. We further propose a dynamic bandwidth separation DIT. With the continuous prediction of online traffic, the scheduling decisions are automatically adjusted, and the available bandwidth for bulk data transmission is corrected in real time, thereby fully utilizing the network bandwidth.

4.2.2. Global Optimal. In a distributed architecture, the path calculation strategy of data flow is rigid, and traffic scheduling lacks flexibility. It is difficult to dynamically adapt to network state changes. Moreover, a distributed control strategy cannot make full use of the limited resources of the network to achieve dynamic optimization of network performance, particularly in an environment where the growth rate of network traffic far exceeds the speed of network infrastructure replacement. Maximize DIT is a completely centralized architecture of an application level multicast overlay network that allows a central controller to maintain an up-to-date global view of the server's data transfer status. Dynamically balancing the availability of various data blocks can help significantly improve the overlay multicast performance, which is critical to achieving near-optimal performance, while further accelerating the speed of multicast mass data transmission.

4.2.3. Reduce the Overhead of Global Scheduling. With the increase in the number of inter-DC overlay paths, the individual servers cannot explore all the available overlay paths based on local measurements alone. This widens the search space for finding usable paths and exponentially increases the centralized computing overhead. The computing overhead of the centralized controller can be significantly reduced by decoupling scheduling and routing steps. Since the scheduling step selects a subset of blocks and only considers the selected blocks in subsequent routing steps, the search space is significantly reduced. To avoid link saturation, a congestion threshold is introduced. A bypass strategy is used to fully utilize hidden resources to achieve load balancing.

Among them, (1) efficient use of the remaining bandwidth corresponds to the sliding-k algorithm, as detailed in [14]; (2) global optimality corresponds to the centralized control module of the DIT architecture; (3) reduction of the global scheduling overhead corresponds to the design of the scheduling and routing decoupling in Section 5.

4.3. Inter-DC Bypass Scheme. Geographically distributed DCs are connected through a dedicated WAN to transmit a large amount of inter-DC traffic. The services carried by the DC network are dominated by multicast services, such as multimedia (music, video, etc.) and cloud computing (electronic maps, social networks, etc.). This multicast traffic will cause a large amount of data and content copying. Application-level overlay networks [29] can be used to reduce the completion time of inter-DC multicast. However, the overlay network of the traditional multicast routing method only uses a path between the sender and the receiver and selects a low-latency link for routing, ignoring other network links that can be used with a longer delay. With the increase in the amount of data transmitted on the Internet, the traditional multicast overlay network method will cause congestion and packet loss, making it necessary for retransmission.

Considering the overall bandwidth usage, our objective was to make full use of the unused remaining bandwidth to improve IDC resource utilization, thus minimizing inter-DC congestion, so as to transfer data backup and update data transmission between different DCs. By combining multipath technology with overlay multicast, we proposed a DIT system to divide large chunks of data into several blocks. The data blocks are scheduled across multiple paths and multiple hops. Scheduling is driven by predicting the remaining bandwidth available on the link. DIT periodically (by default every 5 s) updates routing and scheduling decisions in a centralized manner to maximize the use of remaining resources.

The overlay network is to build a virtual network on top of the existing physical network. The nodes of the overlay network are connected through the underlying network, so path exploration and maintenance can be actively performed. The bottleneck link of a path is defined as the link with the smallest capacity in the path. The overlay routing [30] can be used to deal with the link bypass problem. Our DIT scheduling scheme uses the bottleneck-bypass method,

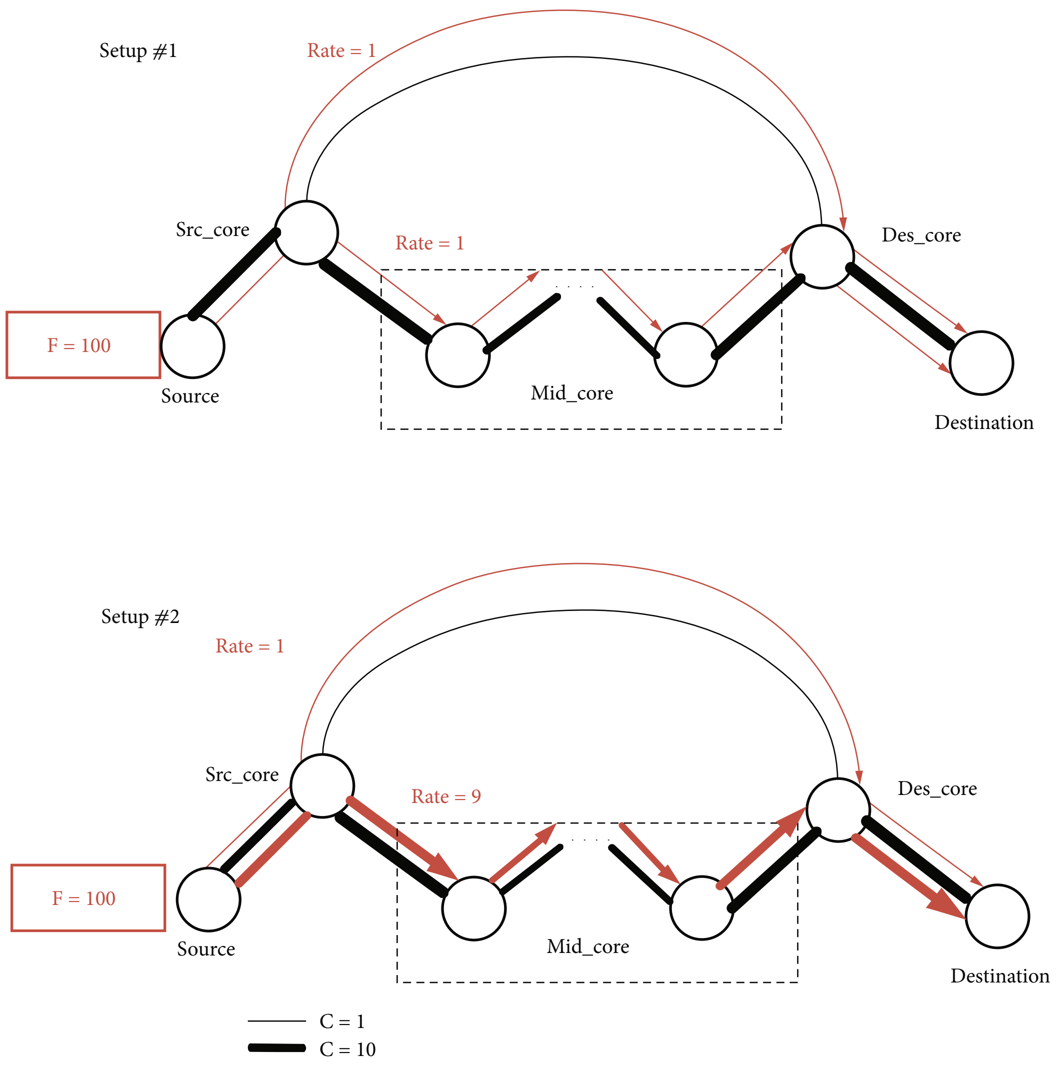


Figure 3: Examples of bottleneck-bypass overlay paths.

which can quickly find the best available path, thereby avoiding congestion and improving WAN performance and fairness.

Figure 3 shows the available paths for sending data traffic from the source DC to the destination DC (including bypass and nonbypass schemes). An example of delivering a large object F (shown by red line, the arrow represents the direction of propagation of the flow instance) from source to destination has a volume of 100 units. We have two types of links with capacities of 1 and 10 units of traffic per time unit (shown by the black line). In setup #1, we can use a multicast paths to connect the source to destination which allow us to transmit at the bottleneck rate of 1 to bypass and nonbypass schemes. In the setup #2, assigning a rate of 1 to the nonbypass attached to destination DC and a rate of 9 to the bypass to destination DC will achieve this goal while respecting link capacity over all links (the Soure to SRC_core link and Des_core to Destination link are the bottleneck). In this paper, we aim to avoid congestion and improve the utilization of network resources by selecting bottleneck-bypass mechanism.

5. Scheduling and Routing Algorithms

Considering the motivations and challenges discussed above, show the design details of our DIT system scheduling and routing. In addition, we explain how to rationally select bypass paths.

5.1. Traffic Scheduling. The scheduling cycle of the DIT system is 5 s, using the deadline closer to the principle of priority to schedule elastic traffic. In each round of the cycle, the time interval of the selected demand includes all the elastic traffic requirements of the cycle for scheduling. The higher the deadline, the higher the priority. If the elastic traffic demand is greater than the remaining traffic of the bottleneck link on the demand path within the current period, the elastic traffic demand is scheduled depending on the size of the remaining traffic, and the unscheduled part of the demand continues to participate in the scheduling of subsequent cycles. In particular, when the remaining traffic of the bottleneck link on the demand

```
Require: Load initial remaining traffic information for each link globally; Load Rigid traffic (total_traffic) between IDCs.
Ensure:  Routing result
1: $running\_time = len(c.demands)$
2: while $ts < \max\_ts$ do
3:     //Initialize:
4:     demand_sorted=sorted(demands.end_time)
5:     demand_id=demand_sorted[i].id
6:     start_t=c.demands[demand_id].start
7:     end_t=c.demands[demand_id].end
8:     demand_traffic=c.demands[demand_id].tra
9:     source= c.demands[demand_id].src
10:    destination= c.demands[demand_id].des
11:    if  (start_t≥ts)and(end_t≤ts)and(demand_traffic>0)
   then
12:        running_time[demand_id-1]+=schedule_cycle
13:    end if
14:    //Find bottleneck, determine whether to transfer
15:    bottleneck=min(sourcecore_medicore_traffic, medicore_descore_traffic)
16:    if  (bottleneck>sourcecore_descore_traffic)and
17: (c.real_regions[sourcecore_descore])/
18:(total_region∗schedule_cycle<0.4)and BYPASS then
19:        flag=True
20:    else
21:        flag=False
22:    end if
23:    //Schedule
24:    if  (src==soure_core)and(des==des_core) then
25:        select_schedule( )
26:    else if  (src ≠ soure_core)and(des==des_core) then
27:        select_schedule(src_srccore_traffic )
28:    else if src == soure_core)and(des ≠ des_core) then
29:        select_schedule( descore_des_traffic )
30:    else
31:        select_schedule(src_srccore_traffic,
32:        descore_des_traffic )
33:    end if
34:    ts+=schedule_cycle
35:end while
```

Algorithm 1: Schedule Algorithm.

path during the current period is zero, the scheduling of the demand will be delayed to the next period, and so on. If an elastic traffic demand still has unscheduled traffic after its deadline, this part of the demand will be discarded and no longer scheduled. If the elastic traffic demand is less than the remaining traffic of the bottleneck link on the demand path within the period, the elastic traffic demand is fully scheduled.

5.2. Routing. The routing process is decoupled from the scheduling process mentioned above, which significantly reduces the computational overhead of the centralized controller. In each cycle of the scheduling process, a subset of data files is selected to be scheduled until all the data files are scheduled or the unscheduled parts are discarded beyond the deadline. The routing process decides which path to select and allocates bandwidth to transfer the selected data files. In each routing process, we only consider the data block selected in the current scheduling cycle for routing, so the search space is significantly reduced. When the following two conditions are met, bypassing is considered to obtain a greater value, and detour links should be selected: (1) the utilization rate of the interdomain link planned to be used when not bypassing has exceeded 60%; (2) the remaining traffic of the bottleneck link in the bypass scheme is greater than the remaining traffic of the link used when nonbypassing.

5.3. Algorithm. Algorithm 1 describes the entire scheduling and routing process of the DIT. First, the elastic flow demand that conforms to the time range of the current cycle is selected for scheduling. In lines 4-13, the demand to be scheduled is sorted by the urgency of the deadline, and the demand list and scheduling time of each demand are then updated. Lines 15-22 are the routing processes. We need to find the bottleneck link and judge whether the demand needs to be bypassed. The remaining traffic of the bypass and nonbypass interdomain links is calculated separately, and the scheme with the largest remaining traffic is selected. In addition,

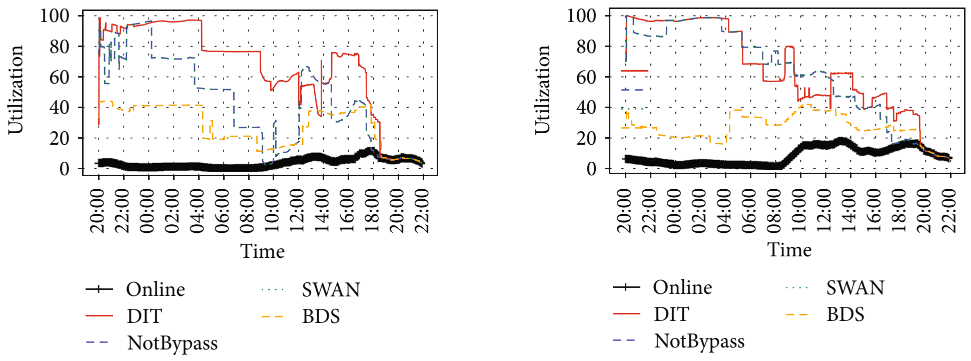
Deployed		
North district	Beijing	50G
East district	Nanjing	50G
South district	Dongguan	50G

North district	Beijing
Langfang	60G
Tianjin	200M
Jinan	800M
Haerbin	100M
Changchun	40M
Wulanchabu	400M
Taiyuan	50M
Shijiazhuang	160M
Shenyang	110M
Qingdao	40M
Huhehaote	80M
Zhengzhou	120M

East district	Nanjing
Shanghai	40G
Hangzhou	40G
Suzhou	40G
Wuhan	40G
Ningbo	2M
Nantong	2M
Hefei	50M
Wuxi	200M
Wenzhou	2M
Yancheng	2M
Xuzhou	2M

South district	Dongguan
Shenzhen	100G
Haiwai	20G
Kunming	100M
Guangzhou	800M
Foshan	50M
Changsha	800M
Nanning	200M
Guiyang	40G
Fuzhou	130M
Nanchang	150M
Xiamen	40M

Figure 4: Original networking information.



(a) Average of all uplink-Nanjing link utilization values (b) Average of all Dongguan-downlink link utilization values

Figure 5: The first set of intradomain link analysis.

before selecting the bypass scheme, it is necessary to meet the link utilization of the nonbypass scheme that has exceeded 60% (avoid frequent switching between the bypass and non-bypass schemes). Lines 24-33 are the scheduling processes. It is necessary to determine whether the source and destination DC are DCRs and then call the select_schedule() function. Because of different source and destination conditions, the condition of the remaining traffic of the links that needs to be updated during the scheduling period is different. In the last 34 lines, we need to update ts and discard unscheduled demand that is beyond the deadline.

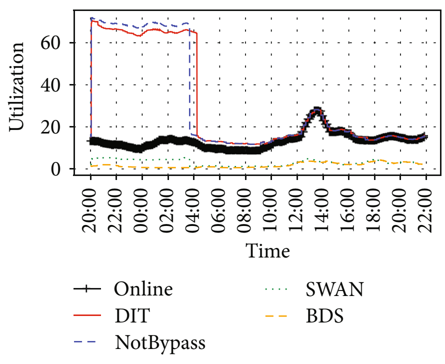
The time complexity for the DIT to find the bottlenecks and decide whether to undertake the bypass solution is $O(N)$. The DIT needs to rely on the agent for traffic information, and in the normal case, the message complexity is $O(N^2)$ due to the all-to-all communication model. In a centralized control process, the communication cost rises to $O(N^3)$ for secure transmission to the agent. Therefore, the time complexity of the DIT system when running is $O(N^3)$.

6. Setup

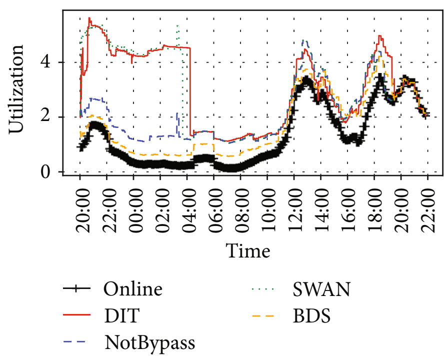
We implemented DIT and deployed it in Huawei's experimental Intranet. There are three super_cores (DCRs) in North District (Beijing), East District (Nanjing), and South District (Dongguan). There are 100 servers in each DC of the region where each super_core is located. The uplink and downlink rates are the same, i.e., 100 MB/s, the scheduling period is 5 s, the total bandwidth of the interdomain link is 50 G, and the total bandwidth of the interdomain link in the augmentation mode is 60 G.

Our testbed consists of the DIT controller, agent monitor, agents, and network agent. The duplications of the DIT controller are implemented on three different geo-located DCRs.

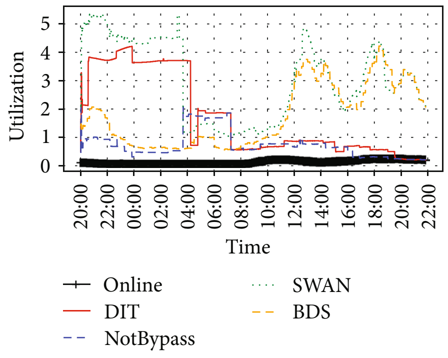
The agent monitor sends control messages between the controller and agent servers. Agent monitor uses the HTTP POST protocol to send control messages between the controller and the servers. The topology information of Huawei's network is dynamically collected by the network monitor. Network agent tracks topology and traffic with the help of switches. The network agent pass messages about topology changes to the controller immediately and collect and report information about flows at the granularity of OpenFlow rules ($\Delta T = 5$ s). DIT reads the elastic traffic and rigid traffic datasets, where the rigid traffic is the real datasets of each link of Huawei's network, and the elastic traffic is a simulation datasets of 300 demands randomly generated, and the congestion threshold by setting–limit-bottleneck in each data block schedule and router.



(a) Nanjing-Dongguan link utilization

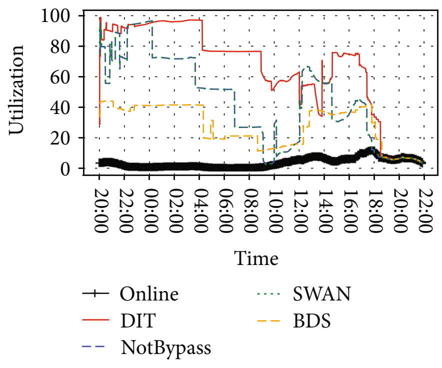


(b) Nanjing-Beijing link utilization

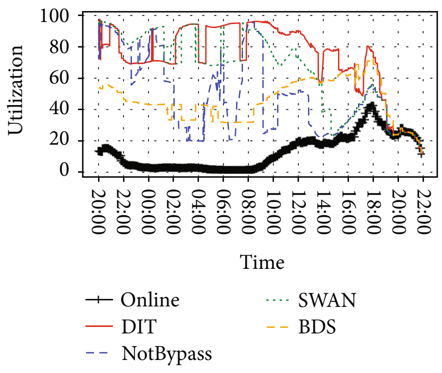


(c) Beijing-Dongguan link utilization

Figure 6: The first set of interdomain link analyses.



(a) Average of all uplink-Nanjing link utilization values (first group)



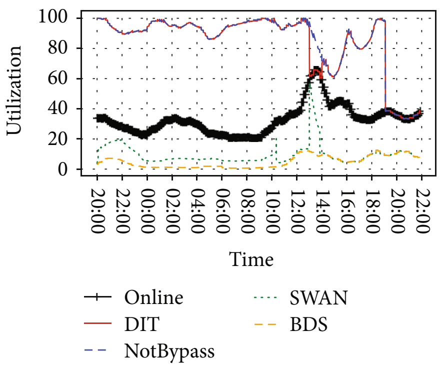
(b) Average of all uplink-Nanjing link utilization values (second group)

Figure 7: The second set of intradomain link analysis.

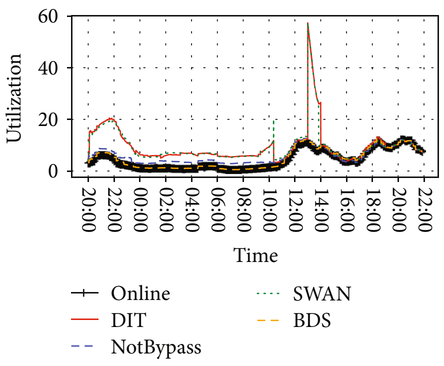
The DIT experimental platform is built with Python. Simulations are conducted on a PC with a Core i7, 3.4 GHz CPU, 16 GB RAM, Windows 10, and Python3. DIT can be seamlessly integrated with any inter-DC communication patterns. First, the original rigid traffic datasets need to be processed to calculate the actual rigid traffic of each link in the network and then send to the centralized control module. In the case of networking and rigid traffic amplification mode, the data preprocessing module will perform the rigid traffic amplification before performing statistics. We speculate that our implementation should be applicable to other companies' DCs too.

7. Experimental Evaluation

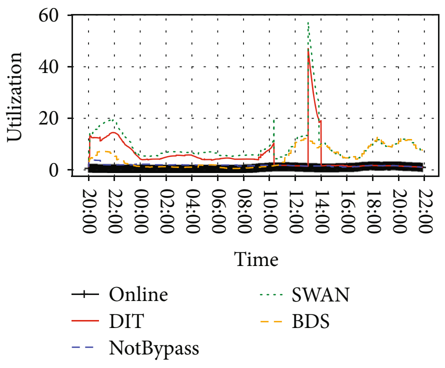
To explore the performance of time-varying systems that set a safety threshold for link utilization under the centralized control platform, we should explore under what conditions the interdomain link will become a bottleneck and whether the bottle-bypass scheme will improve system performance. For this system, four sets of experiments are designed based on the two thresholds of link utilization security threshold and whether the network and rigid traffic are amplified. Security threshold for link utilization: 80% or 100% (with or without a threshold). When the utilization of a link



(a) Nanjing-Dongguan link utilization

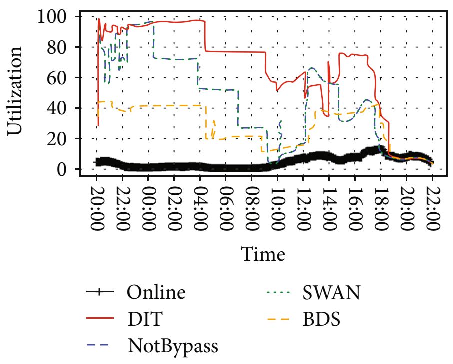


(b) Nanjing-Beijing link utilization

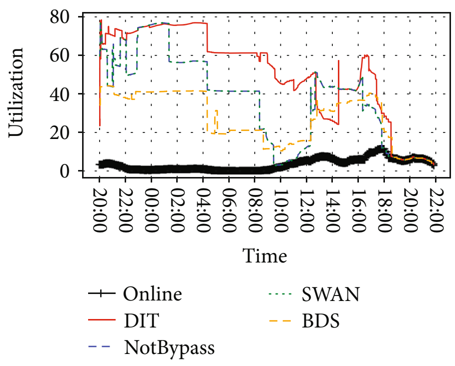


(c) Beijing-Dongguan link utilization

Figure 8: The second set of interdomain link analyses.



(a) Average of all uplink-Nanjing link utilization values (first group)

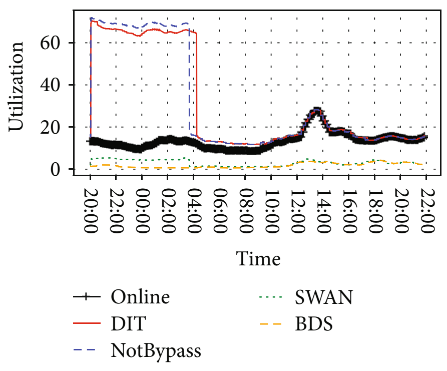


(b) Average of all uplink-Nanjing link utilization values (third group)

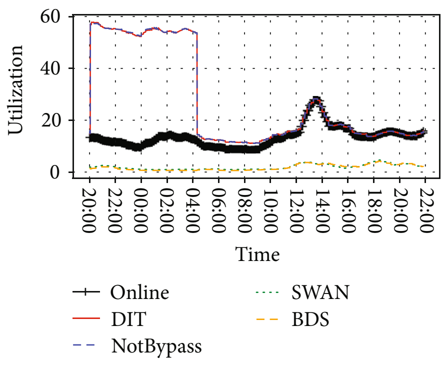
Figure 9: The third set of intradomain link analyses.

reaches this safety threshold, transmission tasks will no longer be scheduled to that link, whether the networking and rigid traffic are amplified: expanded or nonexpanded. Network expansion method: The network is divided into three regions, as shown in Figure 4. During the expansion, a domain in which the total link bandwidth in the domain is centered is found, and it is amplified to a given bandwidth (here, 60 G is selected). The multiple that needs to be amplified is recorded, and the bandwidth of each IDC in each domain is amplified based on this multiple.

Rigid flow amplification method: each original rigid flow file records the rigid flow for a day (24 h) on a path that passes through 1 or 2 segments of the intradomain links and will participate in the statistics of the rigid flow for those links. The rigid flow is amplified to increase the peak value of the rigid flow of links in each domain to approximately 60% of the links. To this end, the multiples of the links in each domain need to be amplified when they are not amplified, and all the original rigid flow files involved in the statistics of the rigid flow of this link are amplified in accordance with this proportion. However, as mentioned earlier, each original rigid traffic file may participate in the statistics of the rigid traffic of links in 1 or 2 segment domains; therefore, the multiple to be amplified in the rigid traffic statistics of different links may not be the same. Therefore, in this experiment, the smaller of the two amplification multiples is taken

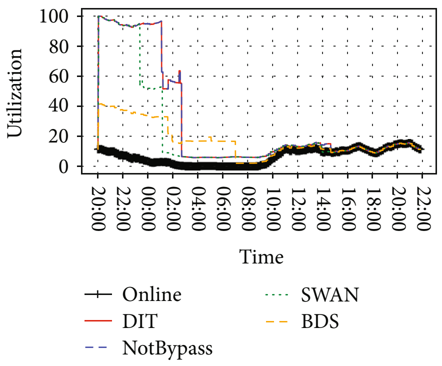


(a) Nanjing-Dongguan link utilization (first group)

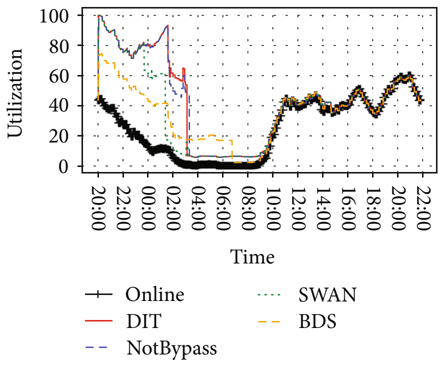


(b) Nanjing-Dongguan link utilization (third group)

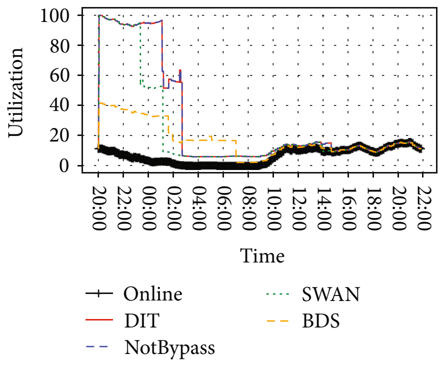
Figure 10: The third set of interdomain link analyses.



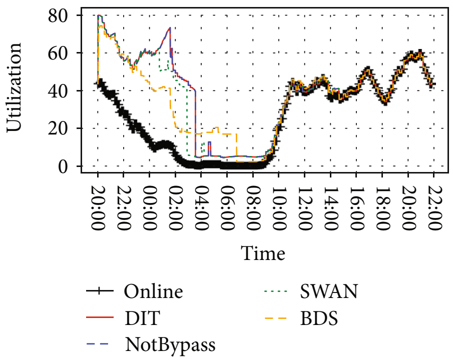
(a) Guangzhou-Dongguan link utilization (first group)



(b) Guangzhou-Dongguan link utilization (second group)



(c) Guangzhou-Dongguan link utilization (third group)



(d) Guangzhou-Dongguan link utilization (fourth group)

Figure 11: The fourth set of intradomain link analyses.

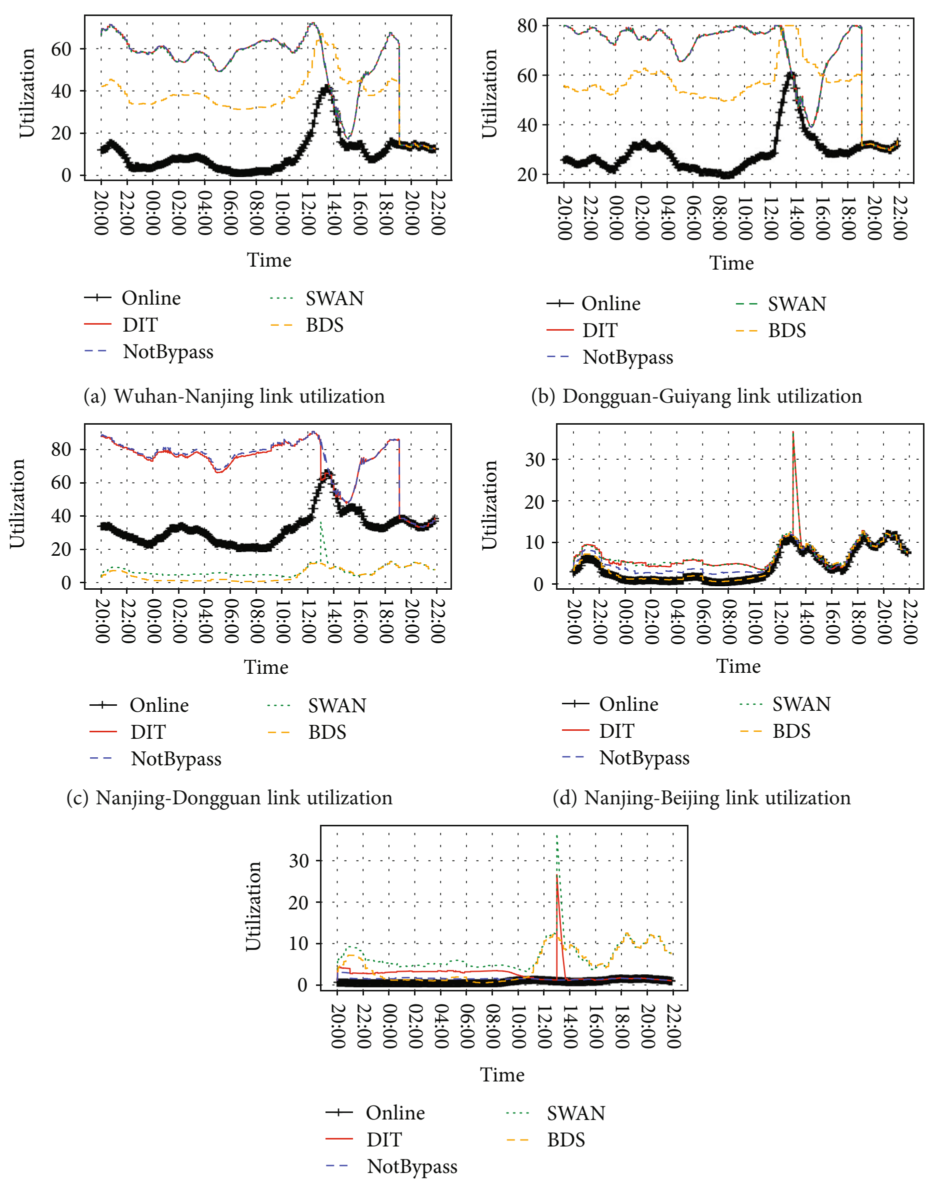
as the amplification multiple of the original rigid traffic file. Thus, the peak of the rigid traffic of the amplified link may be less than 60% of the total link traffic but close to 60%, which can also achieve the purpose of amplification.

7.1. The Value of the Variable in the First Set of Experiments.

The safety threshold of link utilization is 100%; the network and rigid traffic are not amplified.

Intradomain link analysis:

Figures 5(a) and 5(b) show that the system can make full use of the link capacity, and in the face of a large amount of elastic traffic demand, the link utilization can be increased to nearly 100%. Because the prediction is based on rigid traffic, when there is a large fluctuation in rigid traffic, the prediction effect will decrease, resulting in a slight decline in link utilization. This explains why the red line in Figure 5(b) is more jittery than Figure 5(a). The performance degradation of the red line for some of the time periods in Figures 5(a) and 5(b) is due to the arrival of the new transmission demand. The bottleneck is at the other end of the path. The current transmission capacity depends on the bottleneck, so only a relatively small amount of traffic can be transmitted. If there remain other transmission requirements on this link at this moment, and the link is a bottleneck on the transmission demand path, the "gap" at this moment will be filled. Among them, online denotes the resource utilization of online traffic, DIT denotes the dynamic bandwidth isolation scheme proposed in this



(a) Wuhan-Nanjing link utilization

(b) Dongguan-Guiyang link utilization

(c) Nanjing-Dongguan link utilization

(d) Nanjing-Beijing link utilization

(e) Beijing-Dongguan link utilization

Figure 12: The fourth set of interdomain link analyses.

paper, NotBypass denotes the case of no bypass, and SWAN [6] and BDS [8] are both existing schemes for static bandwidth isolation systems. SWAN applies multicommodity flows (MCF) to allocate rates to maximize overall throughput while selecting shorter paths. In contrast, NotBypass uses the Dijkstra algorithm to find the shortest path. When the network topology is relatively simple, the shortest paths found by SWAN and NotBypass may be the same, so the green curve and blue curve almost coincide.

Interdomain link analysis:

Figure 6(a) shows that the utilization of the interdomain link Nanjing-Dongguan is up to approximately 75%, because the size of interdomain traffic is limited by the uplink-Nanjing bottleneck link. We find that the utilization of the Nanjing-Dongguan interdomain link is almost the same (area) in both DIT and NotBypass cases. By looking at Figures 6(b) and 6(c), we can infer that there is a small amount of bypassed traffic on the interdomain link in the DIT and SWAN cases.

7.2. The Value of the Variable in the Second Set of Experiments. The safety threshold of link utilization is 100%, and the network and rigid traffic are amplified.

Intradomain link analysis:

Comparing Figures 7(a) and 7(b), the jitter is more pronounced when the rigid traffic is amplified, further affecting the jitter of the red line. However, the prediction gets better overtime, with DIT significantly outperforming SWAN's 65% intradomain link utilization.

7.3. Interdomain Link Analysis. After the networking and rigid traffic expansion, the remaining traffic on the intradomain link may be greater than the remaining traffic on the interdomain link, so the bottleneck may be on the interdomain link. Figure 8(a) shows that the utilization rate of the interdomain link has reached 100%, indicating that the bottleneck link is the interdomain link between Nanjing and Dongguan. Figures 8(b) and 8(c) observe that the DIT improves interdomain link utilization by an average of 18% over the NotBypass solution when bypassed.

7.4. The Value of the Variables in the Third Set of Experiments. Safety threshold of link utilization is 80%, and the networking and rigid traffic are not amplified.

Intradomain link analysis:

When the link utilization safety threshold is changed from 100% to 80%, the available remaining traffic value of the link in each cycle decreases, and the task requires more cycles to complete. It is intuitively shown that the image becomes "chubby." Specifically, the link in Figure 9(a) is completed at 12:00, and the link in Figure 9(b) is delayed until nearly 13:00.

Interdomain link analysis:

When the upper limit of link utilization is changed from 100% to 80%, the interdomain link utilization drops by approximately 20% in Figure 10. This is because the bottleneck is still in the intradomain uplink-Nanjing under the condition that the networking and rigid traffic are not expanded. The traffic of the interdomain link is limited by the intradomain link. Because the utilization of the intradomain uplink-Nanjing is reduced from 100% to 80%, the transmissible elastic traffic decreases, thereby decreasing the transmissible traffic of the interdomain link. The utilization should also drop from approximately 70% to approximately 50%.

7.5. The Value of the Variables in the Fourth Set of Experiments. Safety threshold of link utilization is 80%; the networking and rigid traffic are amplified. In the above analysis of the mean of the uplink and downlink, we can also analyze the case of a single link.

7.6. Intradomain Link Analysis. A comparison between the fourth group in Figure 11(d) and the second group in Figure 11(b) shows that the transmission completion time is delayed from 5:00 to 5:30.

This result is similar to the comparison between the third and first groups as described above. Compared with the fourth group in Figure 11(d) and the third group in Figure 11(c), the rigid traffic and the network have been expanded. The peak of the curve indicated in black increases to 60%. This result is the same as the second one described above. This result is consistent with the comparison result between the second and first groups as mentioned above. The comparison between the fourth group in Figure 11(d) and the first group in Figure 11(a) shows that rigid traffic and networking have been expanded, while the upper limit of link utilization is changed from 100% to 80%, which is reflected in the fact that the peak value of the black curve is increased to 60%, and the red line is "chubby."

7.7. Interdomain Link Analysis. After the networking and rigid traffic are expanded, the bottleneck may shift from intradomain links to interdomain links. Figure 12(c) shows that when the utilization of the interdomain link Nanjing-Dongguan reaches the upper limit (80%), the link can no longer transmit tasks, so the transmission requirements that originally needed to go through this link will be unable to proceed. However, because this system uses the bottleneck-bypass mode, the requirements that originally had to go through the Nanjing-Dongguan link can choose to go through the Nanjing-Beijing and Beijing-Dongguan links instead.

8. Conclusion

In this paper, in order to fully utilize hybrid interdata center network link bandwidth, we proposed a new adaptive method called DIT for online traffic prediction and offline scheduling. With the help of a sliding-k prediction algorithm, the system predicts the amount of online traffic in the next cycle and then allocates the remaining bandwidth to the offline traffic. We carefully design DIT controller and propose a bottleneck bypass routing scheme to schedule the mixed traffic in the shared inter-DC network. Through simulation, we prove the outstanding performance of DIT compared with other related methods. DIT system has improved interdomain link utilization by 18% and intradomain link utilization by 65%. Our future works include further improving the performance of DIT and modifying the scheduling algorithm to enhance the application-level performance in hybrid inter-DC. In the future, the issues of traffic engineering for interdata center need to be combined with congestion control, load balancing, scalability, etc. It is also an open question to further exploit the benefits of reconfigurable network technology to improve performance metrics and scheduling objectives.

Data Availability

Huawei's private dataset cannot be made public for now.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors' Contributions

Shiyan Zhang is responsible for the conceptualization, design of study, acquisition of data, analysis and/or interpretation of data, writing—original draft, and writing—review and editing; Yuchao Zhang for the conceptualization, design of study, acquisition of data, analysis and/or interpretation of data, and writing—review and editing; and Ran Wang, Xiangyang Gong, and Yongping Xiong for the writing—review and editing.

Acknowledgments

The work was supported in part by the National Key R&D Program of China under grant 2019YFB1802603, the National Natural Science Foundation of China (NSFC) Youth Science Foundation under grant 61802024, and the Fundamental Research Funds for the Central Universities under grant 2482020RC36.

References

[1] J. H. Wang, J. Wang, C. An, and Q. Zhang, “A survey on resource scheduling for data transfers in inter-datacenter WANs,” *Computer Networks*, vol. 161, pp. 115–137, 2019.

[2] J. Xu, “The technical foundations of FinTech: ABCDI and more,” in *The Future and FinTech*, ABCDI and beyond, 2022.

[3] Y. Zhang and K. Xu, *Network Management in Cloud and Edge Computing*, Springer, 2020.

[4] S. A. Hosseini Bidi, *Deadline-Aware Bulk Transfer Scheduling in Best-Effort SD-WANs, [M.S. Thesis]*, University of Calgary, 2021.

[5] L. Luo, H. Yu, K. T. Foerster, M. Noormohammadpour, and S. Schmid, “Inter-datacenter bulk transfers: trends and challenges,” *IEEE Network*, vol. 34, no. 5, pp. 240–246, 2020.

[6] C. Y. Hong, S. Kandula, R. Mahajan et al., “Achieving high utilization with software-driven WAN,” in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pp. 15–26, New York, NY, United States, 2013.

[7] S. Jain, A. Kumar, S. Mandal et al., “B4: experience with a globally-deployed software defined WAN,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[8] Y. Zhang, J. Jiang, K. Xu et al., “BDS: a centralized near-optimal overlay network for inter-datacenter data replication,” in *Proceedings of the Thirteenth EuroSys Conference*, New York, NY, United States, 2018.

[9] A. Kumar, S. Jain, U. Naik et al., “BwE: flexible, hierarchical bandwidth allocation for WAN distributed computing,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, New York, NY, United States, 2015.

[10] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, “Calendaring for wide area networks,” in *Proceedings of the 2014 ACM conference on SIGCOMM*, New York, NY, United States, 2014.

[11] S. Das, K. G. Panda, D. Sen, and W. Arif, “Maximizing last-minute backup in endangered time-varying inter-datacenter networks,” *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2646–2663, 2021.

[12] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. C. Lau, “Orchestrating bulk data transfers across geo-distributed datacenters,” *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 112–125, 2015.

[13] X. Lin, J. Zou, S. Yue, W. Sun, and W. Hu, “SnF scheduling of multicast transfers across inter-datacenter optical networks,” in *ICC 2021-IEEE International Conference on Communications*, pp. 1–7, Montreal, QC, Canada, 2021.

[14] W. Ran, Z. Yuchao, W. Wendong, X. Ke, and C. Laizhong, “Algorithm of mixed traffic scheduling among data centers based on prediction,” *Journal of Computer Research and Development*, vol. 58, no. 6, p. 1307, 2021.

[15] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, “SDN-based multi-class QoS guarantee in inter-data center communications,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 116–128, 2015.

[16] H. Zhang, K. Chen, W. Bai et al., “Guaranteeing deadlines for inter-data center transfers,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 579–595, 2016.

[17] M. Paliwal and D. Shrimankar, “Effective resource management in SDN enabled data center network based on traffic demand,” *IEEE Access*, vol. 7, pp. 69698–69706, 2019.

[18] M. Isard, “Autopilot: automatic data center management,” *ACM SIGOPS Operating Systems Review*, vol. 41, no. 2, pp. 60–67, 2007.

[19] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee, “Internet anycast: performance, problems, & potential,” in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, New York, NY, United States, 2018.

[20] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “Splitstream: high-bandwidth multicast in cooperative environments,” *ACM Sigops Operating Systems Review*, vol. 37, no. 5, pp. 298–313, 2003.

[21] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, “Utility maximization in peer-to-peer systems with applications to video conferencing,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1681–1694, 2012.

[22] T. Li, A. Liu, N. N. Xiong, S. Zhang, and T. Wang, “A trustworthiness-based vehicular recruitment scheme for information collections in distributed networked systems,” *Information Sciences*, vol. 545, pp. 65–81, 2021.

[23] B. Almadani, A. Beg, and A. Mahmoud, “DSF: a distributed sdn control plane framework for the east/west interface,” *IEEE Access*, vol. 9, pp. 26735–26754, 2021.

[24] B. Balakiruthiga and P. Deepalakshmi, “(ITMP)-intelligent traffic management prototype using reinforcement learning approach for software defined data center (SDDC),” *Sustainable Computing: Informatics and Systems*, vol. 32, article 100610, 2021.

[25] G. Sun, Z. Chen, H. Yu, X. du, and M. Guizani, “Online parallelized service function chain orchestration in data center networks,” *IEEE Access*, vol. 7, pp. 100147–100161, 2019.

[26] Y. Tang, H. Guo, T. Yuan et al., “Flow splitter: a deep reinforcement learning-based flow scheduler for hybrid optical-electrical data center network,” *IEEE Access*, vol. 7, pp. 129955–129965, 2019.

[27] G. Mine, J. Hai, L. Jin, and Z. Huiying, “A design of SD-WAN-oriented wide area network access,” in *2020 International Conference on Computer Communication and Network Security (CCNS)*, Xi'an, China, 2020.

[28] J. Song, R. Guérin, and H. Sariowan, “Minimizing network bandwidth under latency constraints: the single node case,” in *2021 33th International Teletraffic Congress (ITC-33)*, Avignon, France, 2021.

[29] Y.-h. Chu, S. G. Rao, and H. Zhang, “A case for end system multicast (keynote address),” *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 1, pp. 1–12, 2000.

[30] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient overlay networks,” in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, New York, NY, United States, 2001.