GPSonflow: Geographic Positioning of Storage for Optimal Nice Flow

ELIZABETH VARKI, University of New Hampshire

This article evaluates the maximum data flow from a sender to a receiver via the internet when all transmissions are scheduled for early morning hours. The significance of early morning hours is that internet congestion is low while users sleep. When the sender and receiver lie in proximal time zones, a direct transmission from sender to receiver can be scheduled for early morning hours. When the sender and receiver are separated by several time zones such that their sleep times are non-overlapping, data can still be transmitted during early morning hours with an indirect store-and-forward transfer. The data are transmitted from the sender to intermediate end networks or data centers that serve as storage hops en route to receiver. The storage hops are placed in zones that are time proximal to the sender or the receiver so that all transmissions to and from storage hops occur during low-congestion early morning hours. This article finds the optimal locations and bandwidth distributions of storage hops for maximum nice internet flow from a sender to a receiver.

CCS Concepts: • **Networks** → **Network performance modeling**; *Peer-to-peer networks*; *Public Internet*; • **Computing methodologies** → **Model development and analysis**;

Additional Key Words and Phrases: System-graph model, flow network, bulk data transfers, inter-datacenter transfers, internet application, maximum flow problem

ACM Reference format:
Elizabeth Varki. 2018. GPSonflow: Geographic Positioning of Storage for Optimal Nice Flow. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 3, Article 12 (June 2018), 29 pages.
https://doi.org/10.1145/3197656

1 INTRODUCTION

Consider this scenario: *A user wants to transfer 1TB of data to another user by a given deadline. The sender and receiver are at end networks, such as campus networks, and share end network resources with other users. The sender and receiver have access to 10Gb/s from 1:00 AM to 6:00 AM, and 100Mb/s for the rest of the day. With 100Mb/s, 1TB is transmitted in 22.23 hours; with 10Gb/s, 1TB is transmitted in 0.23 hours. Consequently, the transfer should be scheduled between 1:00 AM and 6:00 AM. There's just one caveat: The sender is in UK and the receiver is in Japan—the sender is 8 hours behind the receiver.*

There are several programs, such as CERN's particle collider, SETI, and genomic sequencing (Bot et al. 2012), that would benefit from a tool for transmitting terabytes of data. Transferring terabyte-scale data over the internet, however, requires sustained access to large bandwidth, which is challenging given that bandwidth is a shared resource. The internet's organization and protocols are

Author's address: E. Varki (corresponding author), University of New Hampshire, Department of Computer Science, Kingsbury Hall, 33 Academic Way, Durham, NH 03824-2619; email: varki@cs.unh.edu.

https://doi.org/10.1145/3197656

not designed for terabyte- and petabyte-scale transfers. Therefore, terabyte-scale data are typically transferred by slow mail on portable storage devices (Azure 2017; Google 2017; Snowball 2017). Amazon's Snowball (Snowball 2017) allows clients to download their data onto Amazon's portable storage devices, which are then shipped. Microsoft's Azure (Azure 2017) and Google's cloud platform (Google 2017) are other examples of shipping services for terabyte-scale data. Currently, shipping storage devices is the cheapest and fastest method for transferring terabyte-scale data.

A large number of organizations store their data in the cloud, and data exchanges between client organizations and data centers are on the rise. The demand on cloud providers to manage their clients' big data has led to a need for hardware (i.e., gigabit speed connections) and software (i.e., internet protocols and apps) that support terabyte-scale electronic transfers. Both industry and academia are analyzing and developing hardware/software for large transfers (Date 2016). In the future, it is expected that, with a wider deployment of gigabit connections, electronic terabyte-scale transfers will be available to internet users (Date 2016). In the future, what is not expected to change is the shared nature of the internet and the scarcity of bandwidth. As the number of gigabit connections increases, both the number of users and the number of bandwidth-greedy internet applications (such as Netflix streaming) are also expected to increase. Therefore, bandwidth will continue to be a limited and expensive resource. Since terabyte-scale transfers are bandwidth-intensive, a bandwidth efficient transfer would be cost-effective.

Let's assume that large end-to-end transfers are feasible: For example, consider an app that transmits a large data set in small segments by opening several FTP/HTTP connections from various servers at the sender's network; the app monitors transmissions, restarts failed transmission, and reassembles the complete data set once all segments arrive at the receiver. Even with gigabit speed connections and internet apps for terabyte-scale transfers, there is no avoiding the shared nature of the internet. The internet traffic pattern at an end network depends on its users. Traffic at end networks and Internet Exchanges (IXs) increases during the day, peaks around 8:00 PM, and then drops off around 1:00 AM; network traffic has a wave distribution that mimics users' sleep–wake cycle (AMS 2017; DECIX 2017; LINX 2017; Lakhina et al. 2004; Villa and Varki 2011). End networks purchase fixed internet bandwidth based on peak usage, so there is more unused bandwidth when traffic tapers off. It is faster, cheaper, and nicer (to other end network users) to schedule greedy, bandwidth-intensive, sustained transmissions for early morning hours when few users are on the internet. A focus of this work is a scheduler for terabyte-scale transfer apps; the scheduler ensures that all end-to-end transmissions are scheduled during sleep times at the corresponding end networks.

When the sender and receiver are in different time zones, their low traffic times are out-of-sync. Reconsider the UK–Japan scenario where both sender and receiver have access to 10Gb/s from 1:00 AM to 6:00 AM and 100Mb/s during other times. A direct file transfer from sender to receiver has a rate of 100Mb/s at all times and takes 22.23 hours. Even if the users in the UK and Japan have access to 10Gb/s for 24 hours each day, the transmission rate is lower than 10Gb/s due to congestion in long-haul networks and IXs. When the sender and receiver are in different time zones, it is faster to transfer the file indirectly via one or more storage hops. These storage hops may be data centers or end user clients (similar to a BitTorrent client) that store data temporarily en route to the receiver. When all end-to-end transmissions are contained within low-traffic time zones, the bandwidth-intensive transfer is nice to other users. In the UK–Japan example, a nice indirect transfer is as follows: Select three storage hops where hop1 is 4 hours behind the UK, hop2 is 4 hours behind hop1, hop3 is 4 hours behind hop2 (and 4 hours ahead of Japan). At 5:00 AM, UK transmits to hop1 (where it is 1:00 AM); at 5:00 AM, each storage hop transmits to the hop that is 4 hours behind it; when it is 1:00 AM in Japan, hop3 (at 5:00 AM) transmits the file to Japan.

This indirect transfer consists of four end-to-end transmissions that are all contained within low-traffic time zones. The file transfer time is 12.23 hours: 11.31 hours for storage plus 0.92-hour transmissions.

The goal is to select "nice" data centers/clients such that every end-to-end transmission of an indirect transfer lies in the low-traffic time zones. We refer to the problem of data center/client selection as **GPSonflow**: **G**eographic **P**ositioning of **S**torage for **o**ptimal **n**ice **flow**. The fundamental difference between data center hops and end network hops is bandwidth availability: Data centers have ample bandwidth (gigabit-scale), while end network clients have scarce bandwidth (megabit-scale). This article models both data center-supported and client-supported terabyte-scale flow.

2 GPSONFLOW

GPSonflow deals with the problem of moving a terabyte-scale dataset from a sender's network to a receiver's network via the internet with application-level transmission protocols. Before defining the GPSonflow problem statement, we explain the system setup. The sender and receiver are end users and have permission to access available internet uplink/downlink bandwidth at their end networks. The available bandwidth capacity varies, with more bandwidth available during early morning hours. The transfer of data from sender to receiver need not be direct; storage hops, which are end networks or data centers, may be used as store-and-forward hubs. The dataset may be divided into *segments* where each segment has its own transfer path from sender to receiver via zero or more hops. A segment transfer schedule specifies the segment size, the storage hops on the path, and transmission start time from each hop. For example, suppose the sender is in Chicago and the receiver is in Japan: The sender may transmit a segment at 12:00 UTC (6:00 AM Chicago time) to a data center in Alaska (3:00 AM local time); at 15:00 UTC, Alaska (6:00 AM local time) transmits the segment to New Zealand (3:00 AM local time); at 21:00 UTC, New Zealand (9:00 AM local time) transmits the segment to the receiver in Japan (6:00 AM local time). This segment transfer path consists of three end-to-end transmissions: From Chicago to Alaska; from Alaska to New Zealand; and, finally, from New Zealand to Japan. The end-to-end transmissions are carried out with application-level protocols such as HTTP, FTP, or gridFTP.

The sender and receiver are bandwidth-constrained but not storage-constrained. The available bandwidth at the sender and the receiver varies with time of day. At any instant, transmissions must not exceed the bandwidth limit. GPSonflow is an application-level transfer with no control of network level packet routing; however, the transfer has control of application-level routing decisions such as segment sizes, segment transfer paths via zero or more hops, and start time of each end-to-end transmission along a segment transfer path.

The goal of GPSonflow is to find the best transfer start time, the best global locations for the hops, and the minimum bandwidth distribution at each hop that will allow maximum flow from the sender to the receiver. A constraint on hop selection is the following: The storage hops must all be data centers or they must all be end network clients. In client- supported flow, the segment sizes are in the megabyte range and are referred to as *micro-segments*. Data centers have plenty of storage. Clients have limited storage and limited bandwidth, and a client is permitted to participate in at most one micro-segment transfer. The terabyte-scale transfer may use any and all of the end networks of the internet as client hops. Since a client is permitted to participate in the transfer of at most one micro-segment, a large number of clients may be required for the terabyte-scale transfer. Consequently, we refer to client-supported flow as *crowd-* supported flow.

GPSonflow problem statement: *Find the maximum number of bytes that can be transmitted from a sender's end network to a receiver's end network via a store-and-forward transfer, given the earliest transfer start time and the latest transfer completion time (or given the duration of flow). The locations*

and available bandwidth distributions at the sender's end network and the receiver's end network are provided. The sender and receiver have ample storage capacity. For data center-supported flow, the storage hops are all data centers with ample storage. For crowd-supported flow, the storage hops are all end network clients, where each client may participate in the transfer of at most one micro-segment; the micro-segment size is given. Moreover, all transmissions to and from clients are restricted to sleep times; the allowable client transmission times are provided in the problem statement.

Along with maximum flow, the solution should output the transfer schedule, which includes segment/micro-segment transfer paths with transmission start times.

The problem statement is open-ended with respect to input data on storage hops: Some problems may provide locations and bandwidth distributions for every hop, while other problems may only specify whether the flow is data center-controlled or crowd-controlled.

Example 1. What is the maximum number of bytes that can be transmitted from the UK to Japan if the earliest start time is midnight in UK and the latest completion time is 7:00 AM Japan's local time. Both the UK and Japan have 2Gb/s from midnight until 9:00 AM local time.

The solution technique should work for all the following scenarios regarding input hop data:

(1) The data centers have x Gb/s from 1:00 AM until 5:00 AM local time and y Gb/s for the rest of the day. What are the best locations for data centers?
(2) Data centers are located in X, Y, Z; each center's bandwidth distribution is provided. What is the maximum flow?
(3) Data centers are located in X, Y, Z. What is the minimum bandwidth requirements for X, Y, Z that allow maximum flow?
(4) Data centers may participate only from midnight until 6:00 AM local time. What are the best locations and minimum bandwidth capacities for data centers that allow maximum flow from sender to receiver?
(5) The flow is crowd-supported with a micro-segment size of 2MB. Clients are allowed to transmit from 3:00 AM until 6:00 AM local time. Where should clients be placed for maximum flow?

2.1 Solution Technique

A maximum flow algorithm is a graph search algorithm that finds all paths from sender to receiver such that there is maximum transmission of data (Cai et al. 2007; Cormen et al. 2009). The input model to the algorithm is a directed graph—a flow network—of all flow paths from sender to receiver. Thus, GPSonflow can be solved by formulating it as a maximum flow problem, which involves two iterative steps: The first is the construction of the flow network, and the second is the computation of maximum flow. Varying any input parameter, such as the flow start time, changes the flow network and the computed maximum flow value. To find the maximum flow from a sender to a receiver may require several iterations of flow network construction and computation of maximum flow.

There are several algorithms that compute maximum flow; we use the Edmonds-Karp algorithm (Edmonds and Karp 1972). The output of the algorithm is only as good as the input flow network, and this article constructs the flow network for application-controlled terabyte-scale internet transfers via storage hops. The flow graph of GPSonflow not only models the sender and receiver nodes but also the storage hop nodes. And this is the knotty issue with GPSonflow: *To find the optimal storage hop locations, the flow graph modeling all the storage hops and all the flow paths from sender to receiver must be constructed.* For data center-supported flow, an open-ended problem statement may provide insufficient data to construct the flow graph. For crowd-supported

flow, it is possible to construct a flow graph along the lines of the Global Internet Map (GIG 2017), but the size of this flow graph makes the solution exorbitantly expensive.

This article is about modeling: How does one construct a feasible flow graph of terabyte-scale, application-controlled internet transfers when the selection of storage hops is open-ended? Some problem statements provide all details of hops; others provide none. Also, how does one develop a model for both data center-supported flows and crowd- supported flows? We want to construct a flow graph that is general, inclusive, and compact (i.e., has minimum size complexity).

The solution to GPSonflow has to find the start time (and, sometimes, the duration of flow) that results in maximum flow. However, for each start time, the selection of hops for maximum flow may be different. To find the best start time, optimal hop placement, and optimal hop bandwidth, several flow graphs have to be constructed and evaluated. We want to automate the construction of flow graphs.

The **contributions** of this article are (i) the construction of an application-level internet flow graph of terabyte-scale transfers using a minimal set of input parameters that are publicly available and (ii) the development of an algorithm that automatically constructs the GPSonflow graph. The flow graph is unbounded since the duration of transfer is unbounded and can last several days. The defining feature of our algorithm is that it constructs the unbounded GPSonflow network from a constant number of input parameters.

3 RELATED WORK

We are not the first to identify that unused bandwidth during sleep hours can be used for large file transfers. Prior papers (Agapi et al. 2009; Feng et al. 2012; Laoutaris et al. 2011, 2013; Shi et al. 2011) have experimentally shown that indirect transfer via storage hops, where each end-to-end transmission is constrained to sleep hours, is superior to direct transfer when transmitting thousands of gigabytes between two users in different time zones. Thus, these papers have provided proof of concept for GPSonflow.

Research on terabyte-scale transfers can be broadly classified into application-level transfers and network-level transfers. The application-level transfers assume no knowledge or control of network-level packet routing (Cho and Gupta 2011; Feng et al. 2012; Laoutaris et al. 2011, 2013), unlike the network-level transfers (Agapi et al. 2009; Feng et al. 2016, 2017; Lee and Rhee 2017; Lin et al. 2016; Maille et al. 2016). The difference between application- and network- level schedulers is the flow graph input to the maximum flow algorithm. The network-level flow graph incorporates paths along which packets are routed, so the nodes in the graph include edge servers in long-haul networks; this graph assumes knowledge and control of routing paths within long-haul networks. The application-level flow graph assumes no knowledge of the internet beyond publicly available information; therefore, the nodes in the flow graph represent end networks or data centers whose internet uplink and downlink capacities are provided.

Prior papers (Agapi et al. 2009; Cho and Gupta 2011; Feng et al. 2012; Laoutaris et al. 2011; Shi et al. 2011) have used the sleep—wake traffic pattern to identify that the flow model is a dynamic flow network whose arc capacities vary with time. *All prior papers assume complete knowledge of the locations and bandwidth distributions of the participating storage hops; thus prior papers have all the information required to construct the flow graph.* This article only requires that the bandwidth distributions of the sender and receiver be provided. If storage hop data are not provided in the input, then this article generates the minimum bandwidth distribution and the best locations for storage hops.

While related papers have mentioned the challenge of flow network construction, this is the first work to address this challenge. We have developed an algorithm to construct the flow network from a minimal set of system parameters. Earlier papers have used the sleep–wake bandwidth

distribution to identify the model to be a dynamic flow network where bandwidth varies with time; we have used the sleep–wake bandwidth distribution to construct the model automatically.

We are the first to evaluate client-supported flow. Client machines, unlike data centers, have limited bandwidth and storage, so segments are smaller and each client machine is constrained to participate in the transfer of at most one micro-segment. Thus, a crowd of clients must participate in terabyte-scale flows. The size of the flow network depends on the number of data centers or clients. Consequently, the model for data center-supported flow is expected to be smaller than the model for crowd-supported flow. The size complexity of our model is O(T), where T is the flow duration, so the size of our model does not depend on the number of data centers or clients. Our model, unlike previous models, generates transfer paths from sender to receiver where the storage hops are either crowds of client machines or data centers.

4 DIRECT TRANSFER

First, we formulate the simpler scenario of direct transfers from sender to receiver when there are no storage hops; each transfer consists of one or more end-to-end transmissions from sender to receiver. There is sufficient information to construct the flow graph since the bandwidth distributions and the locations of the sender's and receiver's networks are provided in the input statement. The maximum flow is the largest file that can be transmitted directly from the sender's network to the receiver's network in the given duration. There are two possibilities: (i) the available bandwidth distributions at the sender/receiver networks are constant, and (ii) the available bandwidth distributions at the sender/receiver networks vary with time. We construct a system model and map it to a graph model (the flow network).

The parameters of the system model are the internet uplink and downlink bandwidth capacities of end networks. End networks pay for $95th$ percentile backbone bandwidth usage and has access to paid-for backbone uplink and backbone downlink bandwidth (Laoutaris et al. 2013). The following assumption follows from the preceding feature:

Assumption 1. *For GPSonflow, the end networks are the bottleneck, not the internet.*

Constant Bandwidth

The simplest case is direct transfer when the sender and receiver have constant bandwidth for GPSonflow.

System model: The sender end network is represented by s and the receiver end network by r. By Assumption 1, the internet is not a bottleneck, and the bandwidth capacity for transfer is determined by the internet uplink/downlink capacities of the end networks. Let the internet be represented by eXchange node x: s has uplink bandwidth, bw(s,x), to the internet, and r has downlink bandwidth, bw(x,r), from the internet. The problem statement provides the values for bw(s,x) and bw(x,r).

Graph model: The flow network has three nodes, namely, s, r, and x and two arcs: The arc from s to x is the uplink from the sender and the arc from x to r is the downlink to the receiver. The flow network is a star graph with internal node x and two leaf nodes s and r. Let c() represent the capacity function along an arc:

$$c(s, x) = \text{bw(s,x)}; \qquad c(x, r) = \text{bw(x,}r\text{)}.$$

By Assumption 1, the maximum flow from s to r is given by:

$$| f_{max} | = \text{minimum}\{c(s, x), c(x, r)\}.$$

Example 2. What is the maximum number of bytes that can be transmitted from the UK to Japan during twelve hours, given that the sender has bandwidth capacity of 10Gb/s and the

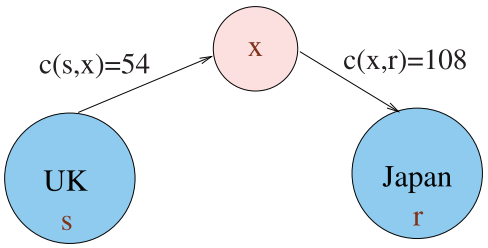

Fig. 1. Star graph corresponding to Example 2: constant bandwidth.

receiver has bandwidth capacity of 20Gb/s. For the example, bw(s,x) = 54TB/12 hours, bw(x,r) = 108TB/12 hours, so maximum flow = minimum{54, 108} = 54TB/12 hours. Figure 1 depicts the flow network for this example.

The unit for bandwidth capacity is bits/second; the unit for file size is bytes. The GPSonflow problem deals with transfer over a period of several hours during which terabytes may be transferred. Mapping from bits to bytes to terabytes is routine, yet distracting, as seen in the preceding example. Therefore, we drop units altogether when units are understood from context. With reference to Example 2, bw(s,x) = 54, bw(x,r) = 108, and the maximum file size is 54.

Variable Bandwidth

Next, we consider direct transfer when the sender and receiver have varying bandwidth by time of day. The end-to-end transmission from sender to receiver depends on the bandwidth capacities at the sender and receiver at the same instant. For constant bandwidth, an instant is equal to the entire duration of transfer. In Example 2, the duration of transfer is 12 hours, and end-to-end transmission is computed for an instant lasting 12 hours. When bandwidth capacity changes with time, the end-to-end transmission rate changes over the duration of transfer, and flow computed at a single instant is incorrect. In the preceding example, if the sender has 10Gb/s in the first 6 hours and 20Gb/s in the last 6 hours of transmission (the receiver has 20Gb/s for 12 hours), then the maximum flow is greater than 54; in the first 6 hours, flow is 27TB and in the last 6 hours, flow is 54TB, for a total flow of 81TB. In this updated example, bandwidth varies with time, and flow is computed by specifying bandwidth at two time instants of 6-hour duration. The model for constant bandwidth does not specify time. To model the variance in bandwidth distribution, it is necessary to model time.

4.1 Modeling Time

The goal is to construct the flow graph enumerating the sender's uplink and the receiver's downlink during each *instant* from start time until end time. First, we have to define an instant of time. Second, we have to define start time instant and duration of flow T (or end time instant). Next, the flow graph must enumerate network bandwidth at all flow instants, t, from start instant until end instant, t = 0, 1, 2, . . . T − 1. To refer to bandwidth at flow instants, we modify the notation c() by adding the parameter t; for example, c(s,x,t) refers to sender's uplink capacity (along arc $\overrightarrow{sx}$) at flow instant t.

Another issue is that flow instant t ranges from t = 0 to t = T − 1, but the problem statement specifies an earliest start time with reference to time of day (or clock time). Therefore, the flow instant t must be mapped to time of day. There are two specifications of time of day: Local time λ and coordinated universal time (UTC) τ. The local time is always in the context of geographic positioning; if the sender and receiver are in different time zones, then the local time instant at a

transmission instant is different for the sender and for the receiver. The universal time, however, is the same in every location. For example, at 00:00 UTC, it is 9:00 AM local time in Japan (ahead by 9 hours, same day), it is 7:00 PM local time in Boston (behind by 5 hours, previous day), and it is 12:00 AM midnight in the UK (GMT zone 0 in winter).

The flow network enumerates bandwidth capacities from sender to receiver during transmission instants. The problem statement, however, provides bandwidth capacities by time of day. A transmission instant t equates to different local time instants, λ, at the sender and receiver, but a transmission instant t equates to the same UTC instant τ at the sender and the receiver. Thus, the model must have notation to specify bandwidth at end networks during UTC instants; we refer to bandwidth at UTC instants by U(). For example U(s,x,τ) refers to a sender's uplink capacity at UTC instant τ. In this section, we explain how U() is mapped to c().

The available bandwidth at a network depends on the sleep–wake cycle, which is in the context of local time. Therefore, the input bandwidth distributions provided in the problem statement are specified by local time. We specify bandwidth at local time instants by the notation L(). For example L(s,x,λ) refers to a sender's uplink capacity at local time instant λ; L(λ) refers to bandwidth capacity at local time instant λ when the network (sender or receiver) is inferred from context. To construct the flow graph, L() is mapped to c().

We refer to local time λ and UTC τ as *system time* and flow time t as *graph time*. System time is cyclical and bounded by the length of day, while flow time is unbounded (since it depends on flow duration). Therefore, bandwidth by system time—U() and L()—are functions of finite and bounded domain; bandwidth by graph time—c()—is a function of unbounded domain. To construct the flow graph, we map graph time t to system time and then map bandwidth by system time (L, U) to bandwidth by graph time c(). *The essence of automating the construction of the GPSonflow model is the mapping of unbounded graph time instants to bounded system time instants and then mapping the periodic bandwidth functions* L*(),* U*() to graph bandwidth function* c*().* In this section, we develop the mappings. The notation is summarized in Table 1.

4.1.1 Time Instant δ. We parameterize time as a discrete variable. When time is discrete, the 24-hour day is divided into discrete time instants each of length δ. The number of time instants in a day, Γ, depends on the time unit δ: If δ is 60 minutes, then the day is divided into 24 time instants ($\Gamma = 24$); if δ is 1 minute, then the day is divided into 1440 time instants ($\Gamma = 1440$); if δ is 180 minutes, then the day is divided into 8 time instants ($\Gamma = 8$). Each time instant represents a time period of one time unit.

The selection of time unit δ must allow the division of the hour/day into equal time periods; for example, a time unit of 7 minutes is unsuitable. The bandwidth is constant during a time instant. The modeler may choose the time unit based on the variance of bandwidth. If available bandwidth changes every few milliseconds, then δ of 1 second balances accuracy and speed; if the bandwidth changes gradually with time, then δ of 1 or more minutes is appropriate. A time unit of 3 to 5 minutes captures the variance in available bandwidth (Laoutaris et al. 2013). The bandwidth capacity at a time instant is the total bandwidth capacity for the time period represented by this time instant. Without loss of generality, in this article, δ is specified in minutes, and $1 \leq \delta \leq 60$. Note that in this article, due to space constraints, examples use $\delta = 180$ minutes (so a day only has 8 time instants as opposed to 24 time instants with $\delta = 60$ minutes). The time instants are specified by $0, 1, 2, 3, \ldots$, where each instant specifies a duration of δ minutes.

4.1.2 Graph Time - Flow Instant t. The graph specifies arc capacities c() (i.e., bandwidth at end networks) from flow start time t = 0 until flow end time t = T − 1, where T is the flow duration. Each flow instant, $t = 0, 1, 2, \ldots, T - 1$, represents a time duration of length δ. For example, suppose δ is

Table 1. Summary of Notation

Notation	Meaning
δ	time unit: the duration of a time instant (in minutes)
Γ	*system parameter*: total number of δ time instants in a day
T	*graph parameter*: total number of δ flow instants during data transfer
τ	*system parameter*: UTC time instant $\tau = 0, 1, \ldots, \Gamma - 1$ $\tau = 0$ maps to duration starting at 00:00 UTC to (00:00+δ) UTC $\tau = 1$ maps to duration starting at (00:00+δ) UTC to (00:00+2δ) UTC $\tau = \Gamma - 1$ maps to duration starting at (00:00-δ) UTC to 00:00 UTC
λ	*system parameter*: Local time instant $\lambda = 0, 1, \ldots, \Gamma - 1$ $\lambda = 0$ maps to duration starting at 12:00 AM (midnight) to (12:00+δ) AM $\lambda = 1$ maps to duration starting at (12:00+δ) AM to (12:00+2δ) AM $\lambda = \Gamma - 1$ maps to clock duration starting at (12:00-δ) PM to midnight
θ	*system parameter*: UTC instant when flow starts from the sender
t	*graph parameter*: flow time instant $t = 0, 1, \ldots, T - 1$ $t = 0$ maps to UTC instant θ when flow starts $t = 1$ maps to UTC instant $\theta + 1$ modulo Γ $t = T - 1$ maps to UTC instant $\theta + t$ modulo Γ when flow completes
$\mathsf{bw}(u, v)$	*system parameter*: constant bandwidth from u to v
$\mathsf{c}(u, v)$	*graph parameter*: constant capacity along edge (u,v)
$\mathsf{U}(u, v, \tau)$	*system parameter*: bandwidth from network u to v at UTC instant τ
$\mathsf{L}(u, v, \lambda)$	*system parameter*: bandwidth from network u to v at local time instant λ
$\mathsf{c}(u, v, t)$	*graph parameter*: capacity along edge (u,v) of **dynamic** flow graph at flow instant t
$\mathsf{c}(u_t, v_t)$	*graph parameter*: capacity along edge (u,v) of **time expanded**flow graph at flow instant t
$\mathsf{z}(v)$	number of hours separating local time in v from UTC time
$\mathsf{d}(v)$	number of time instants (of length δ) separating the local time in v from UTC time

60 minutes and t = 0 starts at 8:00 AM; t = 0 is [8:00-9:00), t = 1 is [9:00-10:00), t = 2 is [10:00-11:00), and so on.

4.1.3 System Time - UTC Instant τ. Let parameter τ refer to a universal time instant: The UTC instants range from $\tau = 0, 1, 2, \ldots, \Gamma - 1$, where Γ is the number of time instants in a day. Discrete time steps $0, 1, 2, 3, \ldots$ map to clock time intervals; time instant $\tau = 0$ maps to time interval of duration δ starting at 00:00 UTC, and $\tau = \Gamma - 1$ refers to the last UTC time interval ending at 00:00 UTC. For example, if time unit δ is 60 minutes, then $\Gamma = 24$ and link capacities are defined for each hour in [00:00-23:00] UTC; $\tau = 0$ represents [00:00-01:00) UTC, $\tau = 1$ represents [01:00-02:00) UTC, $\ldots$, and $\tau = 23$ represents [23:00-00:00) UTC. If time unit δ is 5 minutes, $\Gamma = 288$, $\tau = 0$ represents [00:00-00:05) UTC, $\tau = 1$ represents [00:05-00:10) UTC, $\ldots$, and $\tau = 287$ represents [23:55-00:00) UTC.

4.1.4 Mapping Graph Time t to UTC Instant τ with UTC Start Instant θ. Let θ represent the UTC instant at which transfer is initiated from the sender. Suppose $\delta = 180$ minutes; $\tau = 0$ at [00:00-03:00) UTC, $\tau = 1$ at [03:00-06:00) UTC, $\ldots$ $\tau = 7$ at [21:00-00:00) UTC. If transfer is initiated at

09:00 UTC ($\tau = 3$) then $\theta = 3$. Suppose transfer duration is 6 time instants (i.e., t = 0, 1, 2, 3, 4, 5); the UTC instants relating to transfer duration are $\tau = \theta = 3, \tau = \theta + 1 = 4, \tau = \theta + 2 = 5, \tau = \theta + 3 = 6, \tau = \theta + 4 = 7, \tau = \theta + 5 = 0$. Thus, transfer runs from 09:00 UTC to 03:00 UTC the next day.

Each flow instant t corresponds to a time of day UTC instant τ. The flow instants $t = 0, 1, 2, \ldots, T-1$ correspond to UTC instants $\tau = \theta, \theta + 1, \ldots, \theta + T - 1$. When flow starts at $\tau = \theta$, the UTC instant τ corresponding to flow instant t is given by:

$$\tau = (\theta + t) \; modulo \; \Gamma \tag{1}$$

4.1.5 Mapping U() to c():. The flow graph consists of nodes and arcs that model the bandwidth capacity of *s* and *r* at every flow instant from t = 0 to t = T − 1. The number of input parameters for model construction is proportional to the length of the array c(): c(0), c(1), . . . c(T − 1). The array U() is different from the array c() even though they both represent available bandwidth capacity. The length of array U() is capped by the number of time instants in a day, Γ, while the length of array c() is dependent on the duration of flow T. The parameter U(s,x,0) is the sender's uplink flow capacity at 00:00 UTC. The parameter c(s,x,0) is the sender's uplink flow capacity at the start of flow (t = 0). The start of flow could be at any time: For example, flow could start at 08:00 UTC; thus, U(0) is not equal to c(0) when start of flow is not 00:00 UTC. The system parameter U(i) refers to capacity at UTC instant *i*, while graph parameter c(i) refers to capacity at flow instant *i*. The relationship between arc capacity c() and network bandwidth capacity U() is given by:

$$\begin{aligned} c(v, x, t) &= U(v, x, \theta + t) \\ c(x, v, t) &= U(x, v, \theta + t), \end{aligned} \tag{2}$$

where $0 \leq \theta < \Gamma$ is the UTC instant when flow is initiated and $\tau = (\theta + t)$ modulo Γ from Equation (1).

The input bandwidth distribution is by local time, not UTC. Therefore, we develop the relationship between c() and L() by mapping L() to U().

4.1.6 System Time - Local Time Instant λ. Let λ represent a local time instant of length δ; $\lambda = 0, 1, 2, \ldots, \Gamma - 1$, where Γ is the number of time instants in a day. Discrete time intervals $0, 1, 2, 3, \ldots$ map to clock time intervals; time instant $\lambda = 0$ maps to time interval of duration δ starting at midnight, and $\tau = \Gamma - 1$ refers to the last local time interval ending at midnight. For example, if time unit δ is 60 minutes, then Γ = 24, and link capacities are defined for each hour in a day [12:00 AM-12:00 AM); $\lambda = 0$ represents [12:00 AM-1:00 AM), $\lambda = 1$ represents [1:00 AM-2:00 AM), . . . , and $\tau = 23$ represents [11:00 PM-12:00 AM). If time unit, δ, is 5 minutes, Γ = 288, $\tau = 0$ represents [12:00 AM-12:05 AM), $\lambda = 1$ represents [12:05 AM-12:10 AM), . . . , and $\lambda = 287$ represents [11:55 PM-12:00 AM).

4.1.7 Mapping UTC τ to Local Time λ with Time Zone z(). Note that $\lambda = 0$ starts at midnight while $\tau = 0$ starts at 00:00 UTC. We map UTC instants to local time instants by using the time zones of the sender and receiver. Without loss of generality, we classify the globe into 24 time zones $0, 1, \ldots, 23$; all cities fall in exactly one time zone. Let z(v) refer to the time zone of an end network v. (We use v to refer to an end network, either the sender or receiver.) If z(v) = i, then local time of v is i hours away from its UTC time (in the clockwise direction): in time zone 0, the UTC instant is equal to the local time instant; in time zone 1, the local time is 1 hour away from UTC (i.e., 1:00 AM is 00:00 UTC); in time zone 23, the local time is 23 hours away from UTC (i.e., 11:00 PM is 00:00 UTC). In this article, the UK is the country synonymous with time zone 0: The UK is in time zone 0, Japan is in time zone 9 (9 hours ahead of UK) and New York is in time zone 19 (5 hours behind the UK).

At any UTC time, the local time at a network can be computed from its time zone. Let d(v) refer to the number of time instants (of length δ) separating the local time instant in v from the UTC instant:

$$\mathsf{d}(v) = \mathsf{z}(v) \times \frac{60}{\delta}. \tag{3}$$

Note that δ is the length of a time instant in minutes, so $\frac{60}{\delta}$ gives the number of time instants in an hour. In the UK to Japan example, when $\delta = 180$ minutes: z(UK) = 0 so d(s) = 0 and z(Japan) = 9 so d(r) = 3; if v is in zone 21 (or 22 or 23), then d(v) is 7. If time unit δ is 1 minute, then d(r) = 540. For a given UTC instant τ, the local time instant λ of v is given by:

$$\lambda = (\mathsf{d}(v) + \tau) \; \mathit{modulo} \; \Gamma. \tag{4}$$

4.1.8 Mapping L() to U(). The array L(λ), $\lambda = 0, 1, 2, \ldots, \Gamma - 1$ represents bandwidth by local time; the array U(τ), $\tau = 0, 1, 2, \ldots, \Gamma - 1$ represents bandwidth by UTC. The length of each array is Γ, the number of time instants in a day. The parameter L(v,x,0) represents node v's uplink bandwidth during duration [12:00 AM-12:00 AM+δ); the parameter U(v,x,0) represents node v's uplink bandwidth duration UTC [00:00-00:00+δ). The two arrays are identical only for networks in time zone 0 where $\lambda = \tau$. For example, suppose δ is 60 minutes: If the downlink for receiver Japan is 10 Gb/s from 12:00 PM to 1:00 PM (i.e., [03:00-04:00) UTC), then L(x,r,12) = U(x,r,3) = 4500 GB; for the UK in time zone 0, if the uplink is 10 Gb/s from 12:00 PM to 1:00 PM (i.e., [12:00-13:00) UTC), then L(s,x,12) = U(s,x,12) = 4500 GB. For a given τ,

$$\begin{aligned} \mathsf{U}(v, x, \tau) &= \mathsf{L}(v, x, \mathsf{d}(v) + \tau) \\ \mathsf{U}(x, v, \tau) &= \mathsf{L}(x, v, \mathsf{d}(v) + \tau), \end{aligned} \tag{5}$$

where d() is computed by Equation (3), and λ is set to (d(v)+ τ) modulo Γ from Equation (4).

4.1.9 Mapping L() to c(). From Equation (5) mapping L() to U() and from Equation (2) mapping U() to c():

$$\begin{aligned} \mathsf{c}(v, x, t) &= \mathsf{L}(v, x, \mathsf{d}(v) + \theta + t) \\ \mathsf{c}(x, v, t) &= \mathsf{L}(x, v, \mathsf{d}(v) + \theta + t), \end{aligned} \tag{6}$$

where λ is set to (d(v) +θ + t) modulo Γ from Equations (1) and (4).

Example 3. Suppose the sender's network in the UK and the receiver's network in Japan permit large transfers from local time midnight to noon [12:00 AM-12:00 PM). Let a time instant represent a 3-hour duration; so $\delta = 180$ minutes and $\Gamma = 8$. (For brevity, we use time units >60 minutes in examples.) The sender and receiver have identical bandwidth distribution by local time, L():

from [12:00 AM-3:00 AM), L(0) = 10, from [3:00 AM-6:00 AM), L(1) = 20,
from [6:00 AM-9:00 AM), L(2) = 18, from [9:00 AM-12:00 PM), L(3) = 8,
from [12:00 PM-12:00 AM), L(4) = L(5) = L(6) = L(7) = 0.

Example 4 (Mapping L*() to* U*()).* In Example 3, the sender and receiver have identical distribution by local time, but they are in different time zones, so their bandwidth distribution by UTC differs.

The UK is in time zone 0 where local time equals UTC time. Therefore, its distribution by local time equals its distribution by UTC. The uplink, U(s,x,τ), from the UK is given by:

U(s,x,0) = L(s,x,0) = 10; U(s,x,1) = L(s,x,0) = 20; U(s,x,2) = L(s,x,0) = 18; U(s,x,3) = L(s,x,0) = 8;
U(s,x,4) = L(s,x,0) = 0; U(s,x,5) = L(s,x,0) = 0; U(s,x,6) = L(s,x,0) = 0; U(s,x,7) = L(s,x,0) = 0.

Japan is in time zone 9, so the local time is 9 hours ahead of UTC time (*i.e.,* 3 time units ahead). Therefore the downlink, U(x,r,τ), to Japan is given by:

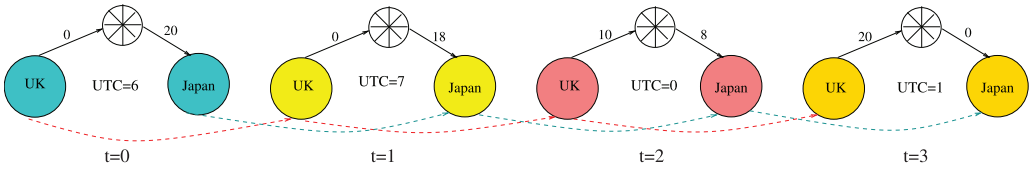

Fig. 2. Time-expanded flow network for Example 6 when UTC start instant θ= 6 (t = 0). Four subgraphs represent bandwidth at sender and receiver at flow instants t = 0, 1, 2, 3. The dashed lines represent storage capacity of end networks; what is not transmitted at t is stored and available for transmission at t = t+1.

U(x,r,0) = L(x,r,3) = 8; U(x,r,1) = L(x,r,4) = 0; U(x,r,2) = L(x,r,5) = 0; U(x,r,3) = L(x,r,6) = 0;
U(x,r,4) = L(x,r,7) = 0; U(x,r,5) = L(x,r,0) = 10; U(x,r,6) = L(x,r,1) = 20; U(x,r,7) = L(x,r,2) = 18.

Example 5 (Mapping U*() to graph arc capacity* c*()).* Suppose transfer is initiated from the UK to Japan for four time instants (t = 0, 1, 2, 3), and the transfer starts at 18:00 UTC. Thus, θ = 6 and τ = 6, 7, 0, 1.

c(s,x,0) = U(s,x,6) = 0, c(s,x,1) = U(s,x,7) = 0, c(s,x,2) = U(s,x,0) = 10, c(s,x,3) = U(s,x,1) = 20;
c(x,r,0) = U(x,r,6) = 20, c(x,r,1) = U(x,r,7) = 18, c(x,r,2) = U(x,r,0) = 8, c(x,r,3) = U(x,r,1) = 0.

Next, suppose the transfer is started at 3:00 UTC for four time instants. Thus, θ = 1 and τ = 1, 2, 3, 4.

c(s,x,0) = U(s,x,1) = 20, c(s,x,1) = U(s,x,2) = 18, c(s,x,2) = U(s,x,3) = 8, c(s,x,3) = U(s,x,4) = 0;
c(x,r,0) = U(x,r,1) = 0, c(x,r,1) = U(x,r,2) = 0, c(x,r,2) = U(x,r,3) = 0, c(x,r,3) = U(x,r,4) = 0.

4.2 Computing Maximum Flow

Unbounded graph model: The maximum flow from s to r during **time duration T** is given by
$| f_{max} | = \sum_{t=0}^{T-1}$ minimum{c(s, x,t), c(x, r,t)}

System model: The maximum flow from s to r **in a day** is given by
$| f_{max} | = \sum_{\tau=0}^{\Gamma-1}$ minimum{U(s, x, τ), U(x, r, τ)}

System-graph model: When flow starts at UTC instant θ for T time units:
$| f_{max} | = \sum_{t=0}^{T-1}$ minimum{U(s, x, θ+t), U(x, r, θ+t)}, where θ+t is computed in modulo Γ arithmetic.

In the preceding equation, c() is equated to U(). The arc capacity c() can be equated to L() (instead of U) using Equation (6). The flow network is still unbounded insofar as duration T, but the flow network can be constructed with Γ inputs. Appendix A presents the system-graph theoretic definition of the flow network.

Example 6 (Reconsider Example 5). When transfer is initiated from the UK to Japan at UTC time instant θ = 6 for 4 time instants:

max flow at t = 0 is minimum{c(s,x,0), c(x,r,0)} = 0, max flow at t = 1 is minimum{c(s,x,1), c(x,r,1)} = 0,
max flow at t = 2 is minimum{c(s,x,2), c(x,r,2)} = 8, max flow at t = 3 is minimum{c(s,x,3), c(x,r,3)} = 0.
The total flow from the UK to Japan is 8. Figure 2 shows the time-expanded flow network. (Refer to Appendix A for time-expanded flow network.)
When θ = 1 (3:00 UTC) and T = 4, maximum flow is 0. Figure 3 shows the time-expanded flow network.

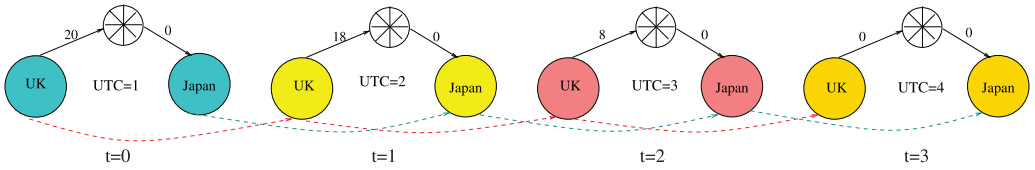

Fig. 3. Time-expanded flow network for Example 6 when θ=1.

As Examples 5 and 6 show, a consequence of including variable bandwidth distribution in the GPSonflow model is that start instant θ becomes significant to the maximum file size computation. With each start time, a new flow network must be constructed since arc capacities c() change (see Figures 2, 3).

5 INDIRECT TRANSFER

When the sender and receiver are in different time zones, their high-capacity time instants are asynchronous. With reference to Example 6, maximum flow from the UK to Japan is 8 even though each end network has a total bandwidth capacity of 56 per day. A maximum of 56 flow units can be transmitted from the UK to Japan if a suitable storage hop is found. Suppose a data center in Germany has uplink and downlink capacity of 20 flow units per time instant for the entire day. The UK starts transmitting at UTC instant 0 when the UK has an uplink capacity 10 and Japan has a downlink capacity 8. At UTC 0, the UK transmits 8 flow units to Japan and the remaining 2 flow units to Germany. During the next three instants, the UK transmits to Germany since Japan has no bandwidth capacity for large transfers. When bandwidth becomes available in Japan, Germany transmits to Japan. In this section, we develop the indirect transfer flow model.

For direct flow, the star graph models two end networks—the sender and the receiver—as leaf nodes and the internet as the internal node; therefore, the size of a subgraph modeling network's capacity at a time instant is constant. The size (i.e., number of subgraphs) of the time-expanded graph is determined by T, the unbounded flow duration. For indirect flow, the size of the graph is dependent not only on the flow duration but also on the number of data centers/clients. The number of data centers/clients is unbounded and can be a very large number: For crowd-supported flow, every end network of the internet can potentially participate in the flow. If the number of data centers/clients is n, then the size of the graph is O(nT).

We develop a model for data center/crowd-supported flow where the size of the graph is O(T). We do not restrict the number of data centers/clients participating in the flow. However, we restrict the number of hops in the model to a maximum of 24.

System model: *In addition to the sender node s, receiver node r, and eXchange node x, the indirect flow model has at most 24 hop nodes, where each hop node represents all the data centers/clients located in its time zone. A hop node u has two parameters, zone number* z(*u*) *and bandwidth distribution by local time* L(*u*)*; the bandwidth of a hop is equal to the sum of the bandwidth of all the data centers/clients in* z(*u*)*. For two hop nodes u and v,* $z(u) \neq z(v)$ *when* $u \neq v$.

Figure 4 shows the indirect model when there are 24 storage hops, one in each zone. The sender node has uplink and the receiver has downlink; all the other nodes have uplink and downlink edges. (In the figure, the two edges are not shown explicitly, but are implicitly represented by a single edge with no arrows.) The capacities along arcs vary with time of day.

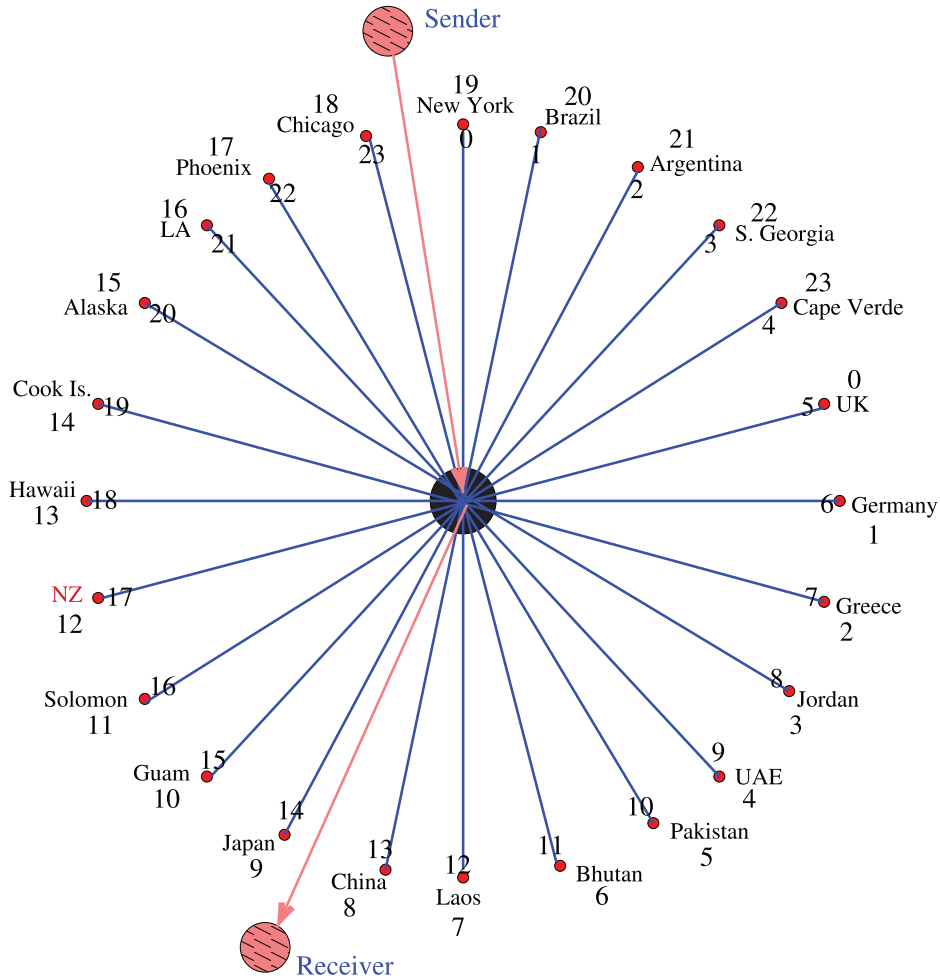

Fig. 4. System model: Two nodes representing sender and receiver, and 24 storage hops, one in each UTC zone. Each node has three labels: The outer number represents zone number; the country/city label adds meaning by signifying global positioning of the node; the inner number represents local time (time instant is 60 minutes) at each node during the flow instant.

5.1 Data Center-Supported Flow

Our model specifies at most one hop in each of the 24 UTC zones 0, 1, . . . , 23; the hop is defined by two parameters: The zone number, z(), and the bandwidth distribution by local time, L(). For the sender and receiver, the problem statement provides the locations and the bandwidth distributions by local time. For data centers, however, the problem statement is open-ended: The distributions and locations of all, some, or none of the data centers may be provided. If the problem statement does not provide information on the placements and bandwidth distributions of all or some of the data centers, we have to fill in the blanks to construct the flow graph.

It is reasonable to assume that information on data centers is not provided when there are no constraints or restrictions on the data centers. Thus, when the problem statement states nothing about data centers, the flow graph models a hop with unlimited bandwidth in every time zone. The maximum flow algorithm outputs the transfer paths and maximum flow, and, from

this output, the best placement and minimum bandwidth for data centers are determined. For example, consider a model with 24 hops, each having large (unlimited) bandwidth; suppose the maximum flow algorithm outputs only two transfer paths: A segment of size 10 transmitted via hops in zones 2 and 5, and a segment of size 6 transmitted via a hop in zone 21. From this output, we determine that data centers should be placed in zones 2, 5, and 21; the minimum bandwidth distribution of hops is determined by the segment size (10 for hops in zone 2, 5; 6 for hop in zone 21) and hop transmission times. A comprehensive example is provided in Section 7.

Assumption 2. *When the locations of data centers are not provided, the flow graph models a hop in each of the 24 time zones.*

When the bandwidth of a data center is not provided, the flow graph assumes that the hop corresponding to the data center has unlimited bandwidth (i.e., *the hop's bandwidth is not a bottleneck).*

For data center-supported flow, we explain the model for different scenarios allowed by the open-ended problem statement (refer to Example 1 in Section 2):

(1) the locations and bandwidth distributions of all data centers are given: The model will have at most 24 hops, one for each zone; and the bandwidth of a hop, L(), is the sum of the bandwidth of all data centers in the zone.
(2) no information on data centers is provided: The model constructs one hop in every zone, and each hop has unlimited (large) constant bandwidth.
(3) the locations of all data centers are provided: Construct one hop node in each of the zones where data centers are located, and each hop has unlimited bandwidth.
(4) the times during which data centers may transmit are provided: The model constructs one hop in each of the 24 zones; each hop has unlimited constant bandwidth during the transmission times and zero bandwidth during other times.
(5) the bandwidth distribution by local time, L(), of a data center is provided, but the locations of the data centers are not provided: The model constructs one hop node in every zone; all hops have identical bandwidth distribution by local time L() (provided in problem statement).
(6) the locations of all data centers are provided, but bandwidth distribution by local time of only a single data center is provided: The model constructs one hop in each of the specified zones; all hops have identical bandwidth distribution by local time L() (provided in problem statement).

Example 7 (Reconsider Example 3). Suppose data centers (hops) h0, h1, h2, . . . , h7, are placed in zones 0, 3, 6, 9, 12, 15, 18, and 21. Thus, z(h0) = 0, z(h1) = 3, z(h2) = 6, z(h3) = 9, z(h4) = 12, z(h5) = 15, z(h6) = 18, z(h7) = 21. Suppose bandwidth distribution by local time of hops is identical to that of the end networks in the UK and Japan.

Storage hop 1 is in zone 3, so local time is 3 hours ahead of UTC time (*i.e.,* 1 time unit ahead). Using Equation (5) to map L() to U():

$$U(h1,x,0) = L(h1,x,1) = 20;\ U(h1,x,1) = L(h1,x,2) = 18;\ U(h1,x,2) = L(h1,x,3) = 8;\ U(h1,x,3) = L(h1,x,4) = 0;$$
$$U(h1,x,4) = L(h1,x,5) = 0;\ U(h1,x,5) = L(h1,x,6) = 0;\ U(h1,x,6) = L(h1,x,7) = 0;\ U(h1,x,7) = L(h1,x,0) = 10.$$

Storage hop 2 is in zone 6, so local time is 6 hours ahead of UTC time (*i.e.,* 2 time units ahead).

$$U(h2,x,0) = L(h2,x,2) = 18;\ U(h2,x,1) = L(h2,x,3) = 8;\ U(h2,x,2) = L(h2,x,4) = 0;\ U(h2,x,3) = L(h2,x,5) = 0;$$
$$U(h2,x,4) = L(h2,x,6) = 0;\ U(h2,x,5) = L(h2,x,7) = 0;\ U(h2,x,6) = L(h2,x,0) = 10;\ U(h2,x,7) = L(h2,x,1) = 20.$$

The uplink/downlink capacities of other storage hops are similarly computed.

5.2 Crowd-Supported Flow

In crowd-supported flow, clients have limited bandwidth and limited storage, and each client may participate in the transfer of at most one micro-segment. The problem statement gives the size of the micro-segment and the times during which clients may participate in transmissions. The problem statement may limit the number of clients in each zone that may participate in a flow. If the problem statement does not provide information on the number of clients in a zone, then, given the scale of the internet and its large user base, it is reasonable to assume that the number of clients in each zone is unlimited (i.e., a large number).

Assumption 3. *When the number of clients in a zone is not specified, the flow model assumes that the sum of bandwidth of all clients in the zone, during the times that clients are permitted to transmit, is not a bottleneck.*

Next, we explain how this assumption results in the crowd-supported flow model being equivalent to the data center -supported flow model with at most 24 hops, one in each zone. The single hop in each zone represents all the clients in the zone. Since a client may transmit at most one micro-segment, the bandwidth of a hop is equal to the number of clients (in the zone) times the micro-segment size. If the number of clients in a zone is not provided, then the hop corresponding to the zone has unlimited bandwidth during the transmission times (by Assumption 3).

There is one issue: Namely, when this model is input to a maximum flow algorithm, the outputs are segment transfer paths, not micro-segment transfer paths. For example, suppose the model of Example 7 is input to a maximum flow algorithm, the output of the algorithm would be segment transfer paths (e.g., segment of size 5 along path *UK to h2 to h4 to Japan*). We map from segments to micro-segments by multiplexing each segment transfer path into micro-segment transfer paths. Reconsidering the example: Suppose the micro-segment size is 2, then the segment (of size 5) is divided into three micro-segments—two of size 2, and one of size 1. All three micro-segments travel along the same path *UK to h2 to h4 to Japan*. Each micro-segment uses a unique client in the zone; referring to the example, there would be three clients in zone 2 and three clients in zone 4. Thus, the transfer satisfies the constraint that each client participates in the transfer of at most one micro-segment.

Instead of multiplexing segments into micro-segments, one could also set time instant δ to be so small that the maximum segment size that can be transmitted in an instant does not exceed the size of a micro-segment. For example, suppose the micro-segment size is 3 and δ is set to a minute; suppose a maximum of 2 units of data can be transmitted from the sender in a minute (or received by the receiver in a minute). Every transfer path from the sender to the receiver via zero or more hops would transmit at most 2 units. Therefore, the output of the algorithm would be micro-segment transfer paths.

Example 8. Referring to Example 3, where a time instant is 3 hours: suppose a micro-segment size is 7. If the time instant is reduced to 1 hour, the sender/receiver's bandwidth distribution would be

$$L(0) = L(1) = L(2) = \tfrac{10}{3};\quad L(3) = L(4) = L(5) = \tfrac{20}{3};\quad L(6) = L(7) = L(8) = \tfrac{18}{3};$$
$$L(9) = L(10) = L(11) = \tfrac{8}{3};\quad L(12) = L(13) = \ldots L(23) = 0.$$

Since the maximum data that can be transmitted in an instant is less than 7 units, the maximum flow algorithm would output micro-segment transfer paths.

5.3 System-Graph Model

The dynamic model is a star graph with at most 26 leaf nodes (sender, receiver, and at most 24 storage hop nodes). The arc capacities vary with time of flow, which is mapped to system time via

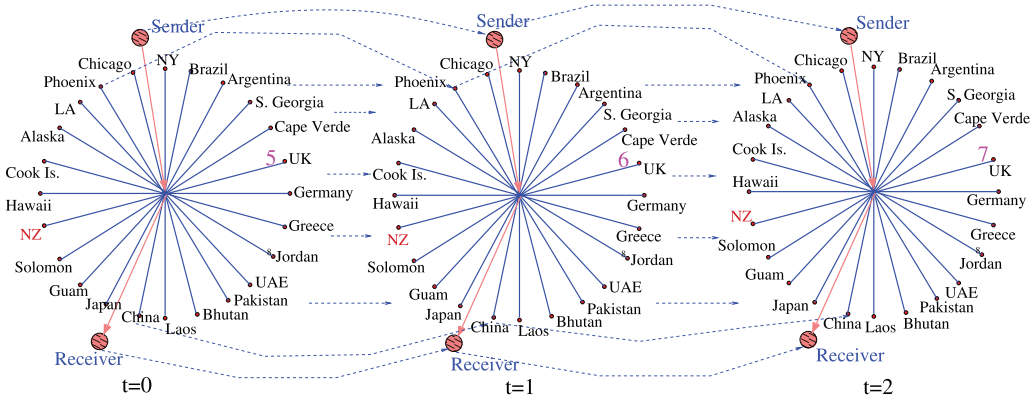

Fig. 5. Time-expanded star graph corresponding to system model of Figure 4: Three subgraphs representing flow instants t = 0, 1, 2. At t = 0, 1, and 2, local time instant in the UK is 5, 6, and 7, respectively; so UTC instant (at all nodes) is also 5, 6, and 7.

UTC start instant. The dynamic flow network has to be transformed to the static time-expanded network for flow computation by maximum flow algorithms. Figure 5 presents the time-expanded flow network of Figure 4 when the duration is three time instants. Appendix B presents the system-graph theoretic definitions.

6 MODEL CONSTRUCTION WITH GCLOCK

A contribution of this article is that our model bounds the number of nodes in a subgraph regardless of the unbounded number of data center or clients participating in the flow. Our model divides the globe into a maximum of 24 longitudinal time zones and places at most one hop in each zone. For finer-grained modeling, each time zone could be further divided into a fixed number of latitudinal regions; each region could be assigned at most one hop representing all the data centers/clients in the region. This effect could be achieved by two methods:

(1) Leave the model unchanged but multiplex the transfer paths so that hops refer to data centers/clients in different latitudinal regions. For example, a transfer path for segment of size 10 via a hop in zone 19 could be assigned to data centers in different regions of zone 19, say Boston and Panama; a segment of size 5 could be transferred via a data center in Boston, and a segment of size 5 could be transferred via a data center in Panama.
(2) Explicitly model hops in latitudinal regions within a zone. For example, the model could have two hops in zone 19, one corresponding to Boston and another corresponding to Panama.

In either case, the size complexity of the model remains O(T), so the size of the flow network is dependent on the flow duration. With δ of 5 minutes and duration of 24 hours, there are 288 subgraphs in the time-expanded flow network for a total of 14400 arcs (without accounting for holdover arcs). With a time unit of 1 minute and a duration of 24 hours, there are 72000 arcs in the time-expanded graph. Thus, manual construction of the graph is error prone and tedious.

Often, the only invariants in the GPSonflow problem statement are the sender's and receiver's bandwidth distributions L(); the optimal start time and the duration of flow have to be determined. To determine the optimal start time, maximum flow at each start time must be computed. Changing the start time changes the flow network because c() is dependent on the start time: For example, if flow starts at 8:00 AM, then c(s,x,0) specifies bandwidth during instant starting at 8:00 AM; on

the other hand, if flow starts at 9:00 PM, then c(s,x,0) specifies bandwidth during instant starting at 9:00 PM. Changing the flow duration also changes the flow network because the length of array c() depends on T (c(0), c(1), . . . , c(T − 1)). Thus, solving GPSonflow often requires the construction of several flow networks.

For each start instant θ and for each duration T, a new flow network must be constructed since c() is dependent on θ and T. We develop an algorithm to construct a flow graph automatically from a fixed set of input parameters, namely, the zone numbers z() and the bandwidth by local time, L(), for the sender, receiver, and hop nodes. *The basis of the algorithm is the following: The input* L*() is independent of both* θ *and T.* Therefore, $\forall\ \theta$ and T, c() can be generated from L() by Equation (6) (in Section 4.1.9).

We develop a data structure, **Graph clock (Gclock)**, for mapping from the unbounded flow instant t to the bounded local time instant for every node. Gclock follows from Equations (1), (3), and (4).

Definition 1. Gclock C(δ, θ,T) tracks the local time instant $\lambda = 0, 1, \ldots, \Gamma - 1$, in every UTC time zone $\mu = 0, 1, \ldots, 23$, at each flow instant t = 0, 1, . . . , T − 1 of a flow graph.

Let lt(μ,t) represent the local time instant in zone μ at flow instant t. The initial configuration of Gclock C at t = 0 when UTC instant $\tau = \theta$ is given by

$$\text{lt}(\mu, 0) = \left(\mu \times \frac{60}{\delta} + \theta\right) \bmod \Gamma, \forall \mu \in [0, 24)$$

At each clock tick, t = t + 1 and

$$\text{lt}(\mu, t) = (\text{lt}(\mu, t-1) + 1) \bmod \Gamma, \forall \mu \in [0, 24).$$

With Gclock, the local time in a zone, μ, at any flow instant, t, is given by
lt(μ,t) = (lt(μ,0) + t) mod Γ.
From Equation (6), the bandwidth capacity at any node, v, at any flow instant, t, is given by
c(v,x,t) = c(v_t, x_t) = L(v,x,lt(z(v),t)).

Gclock maps graph flow instant t to local time of nodes in any time expanded flow graph once the flow start time t = 0 is mapped to the UTC start instant θ (the local time in zone 0). The graph clock is not specific to GPSonflow, so the data structure would be useful for other applications that are modeled by dynamic flow networks where graph parameters are functions of system time. The next example illustrates graph construction with Gclock.

Example 9. Reconsider Example 7. Suppose flow starts at $\theta = 6$; at θ= 6, t = 0 so c(0) = U(6):

c(s_0, x_0) = U(s,x,6) = L(s,x,6) = 0 since z(s) = 0; c(x_0, r_0) = U(x,r,6) = L(x,r,1) = 20 since z(r) = 9;
c($h1_0, x_0$) = U(h1,x,6) = L(h1,x,7) = 0 since z(h1) = 3; c($h2_0, x_0$)= U(h2,x,6) = L(h2,x,0) = 10 since z(h2) = 6

At t = 1, 2, . . . T − 1, the arc capacities are computed similarly by mapping t to system time (c(1) = U(7); c(2) = U(0); c(3) = U(1)...).

If the start time is changed to $\theta = 2$, then a new graph has to be generated since c(0) = U(2).

Figures 6 and 7 present a visual representation of Gclock. The first clock in Figure 6 depicts Gclock at t = 0 when δ is 60 minutes. Each colored point on the circle represents a zone: The zone numbers are shown outside the circle and the local time instants are shown inside the circle. Each zone is also labeled by the name of a country/city within the time zone. As the figure shows, Gclock tracks the local time instant in all zones at a flow instant. The value of UTC instant is given by the local time in zone 0 (UK); in the first clock, the UTC instant is 5. At each clock tick, t is incremented to t + 1, the UTC instant τ is incremented to τ + 1 mod Γ, and the local time instant for each zone is also incremented by 1. Referring to the figure, at each clock tick, the zones move

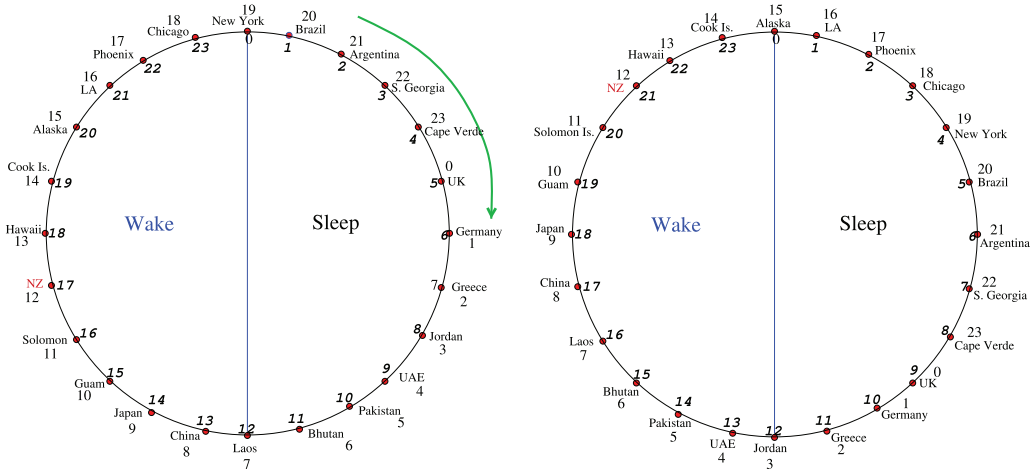

Fig. 6. Gclock where $\delta = 60$ minutes. There are 24 zones. The numbers outside the circle represent the zone number; the numbers inside the circle represent local time at the zones. Each zone is also referred to by the name of a country/city in the UTC zone. The UTC clock instant $\tau = 5$, the local time in zone 0 (UK). The zone numbers rotate clockwise while the circle remains fixed. The zones have shifted 4 positions to the right in the second figure, so $\tau = 9$.

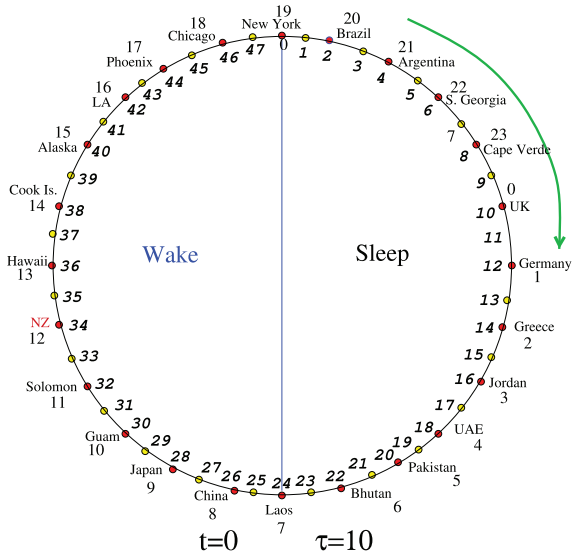

Fig. 7. Gclock where time unit $\delta = 30$ minutes.

one local time instant in the clockwise direction, while the clock itself—the circle, the local time instants—stays fixed. The second clock in Figure 6 shows the local time in all zones when t = 4, four clock ticks after initialization; the UTC instant is 9. Figure 7 depicts Gclock, where δ is 30 minutes.

Algorithm 1 shows the construction of the GPSonflow time-expanded graph using Gclock. When an input parameter, say start time θ, is changed, the flow network can be constructed by

changing the initial configuration of Gclock. Thus, Gclock automates model construction. The motivating example of Section 7 illustrates the practicality and usefulness of Gclock and the algorithm.

ALGORITHM 1: Constructing the time-expanded flow graph

```
Input: Gclock C(δ,θ,T);
Input: ∀ u ∈ V, z(u);
Input: ∀ u ∈ V, L(u,x,λ), L(x,u,λ), λ = 0, 1, . . . , Γ-1;
for t=0 to T-1 do
    for ∀ u ∈ V do
        c(u_t, x_t) = L(u,x,lt(z(u),t));
        c(x_t, u_t) = L(x,u,lt(z(u),t));
        if t<T-1 then
            c(u_t, u_t+1) = ∞;
        end
    end
end
```

7 EVALUATION

Here, we demonstrate the usefulness of Gclock and Algorithm 1 with a simple example: The objective is maximal flow from Chicago (Ch) to Japan (Jp) when the sender in Chicago and the receiver in Japan are end networks with identical bandwidth distribution by local time. In the problem statement, it is given that data centers are placed in zones 0, 3, 6, 12, 15, and 21 (zones 9 and 18 have the receiver and the sender). Starting from Chicago, the zones where data centers are placed are Argentina (Ag), United Kingdom (UK), Jordan (Jd), Bhutan (Bh), New Zealand (NZ), and Alaska (Ak). The problem statement provides two possible bandwidth distributions:

Bandwidth distribution 1: The bandwidth distribution by local time at Chicago and Japan: [12:00 AM-3:00 AM): L(0) = 10; [3:00 AM-6:00 AM): L(1) = 20; [6:00 AM-9:00 AM): L(2) = 18; [9:00 AM-12:00 PM): L(3) = 8; [12:00 PM-3:00 PM): L(4) = 3; [3:00 PM-6:00 PM): L(5) = 1; [6:00 PM-9:00 PM): L(6) = 2; [9:00 PM-12:00 AM): L(7) = 4.

Bandwidth distribution of data centers: [12:00 AM-3:00 AM: L(0) = 5; [3:00 AM-6:00 AM): L(1) = 5; [6:00 AM-9:00 AM): L(2) = 5; [9:00 AM-12:00 PM): L(3) = 5; [12:00 PM-12:00 AM): L(4) = L(5) = L(6) = L(7) = 0.

Thus, data centers have far less bandwidth than either the sender or the receiver. (The upload and download distributions are identical.)

Bandwidth distribution 2: The sender, receiver, and data centers have identical bandwidth distribution by local time. All nodes are only allowed to transmit from midnight until noon: L(0) = 10, L(1) = 20, L(2) = 18, L(3) = 8, L(4) = L(5) = L(6) = L(7) = 0;

Both bandwidth distributions model the sleep–wake diurnal cycle with more bandwidth available during the early morning hours. The fundamental difference between the two distributions is that in distribution 1, the data centers have less bandwidth than the sender/receiver. Also, in distribution 1 (unlike distribution 2), the sender and receiver are permitted to transmit during high-traffic times (with lower bandwidth).

Objective: Find the maximum flow from Chicago to Japan when flow duration is 24 hours.

To find the maximal flow, the best start time with respect to both distributions must be found. Once the best start time is found, the best location and minimal bandwidth distribution for each hop is identified from the detailed transfer schedule.

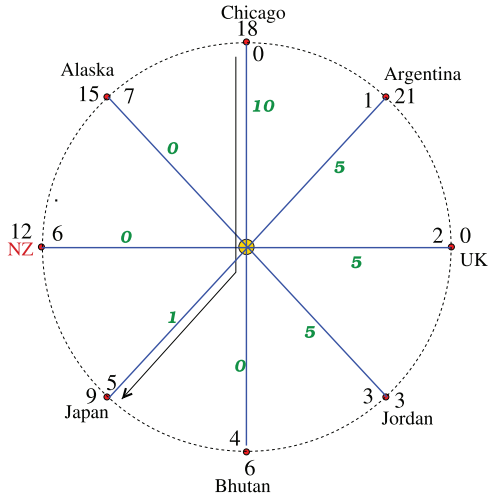

Fig. 8. The numbers outside the circle represent zone numbers while the numbers just inside the circle represent the local time instant at the corresponding zone. (The time instant $\delta = 180$ minutes and $\Gamma = 8$.) Model shows local time instants in zones when UTC instant $\tau = 2$, the local time in UK (zone 0). The numbers along the links represent bandwidth capacity of the node at the modeled instant (based on distribution 1). The circle is an abstract representation of GPS positioning.

Evaluating the effect of data center placement on size of a single subgraph of a flow graph: The problem statement provides data center locations; the data centers lie in six unique zones. Therefore, a dynamic flow network has a total of nine nodes: a sender node, a receiver node, an internal exchange node, and the six hop nodes. The problem statement provides two different bandwidth distributions, so two scenarios have to be evaluated, where each scenario is represented by a dynamic flow graph with nine nodes. Figure 8 shows a star subgraph of GPSonflow with distribution 1 at a single time instant.

Evaluating the effect of bandwidth variance and flow duration on size of a single flow graph: The problem statement provides total bandwidth during 3-hour periods; thus, bandwidth variance is modeled by setting time instant $\delta = 180$ minutes. (Number of time instants in a day $\Gamma = 8$.) Since flow duration is 24 hours, T = 8 and there are eight star subgraphs (each representing flow time t, $0 \leq t < 8$) in every flow graph. If flow duration is changed to 36 hours, T = 12 and there would be 12 subgraphs.

If the time instant is set to a more realistic 3 minutes, then the input bandwidth array L() would have length 480. A single flow graph would have 480 subgraphs for a flow duration of 24 hours, and 720 subgraphs for a duration of 36 hours. If the time instant is set to 1 minute, a single flow graph would have 1440 subgraphs for a flow duration of 24 hours.

Evaluating the effect of bandwidth variance on number of flow graphs that must be evaluated: For a given duration, the maximal flow may vary with each start instant. For a duration of 24 hours and a time instant of 3 hours, there are eight possible start time instants. For each start time, a flow graph has to be constructed before maximum flow is computed. For the two distributions, a total of 16 flow graphs have to be constructed and solved for maximum flow.

If the time instant is 3 minutes, then there are 480 start instants in 24 hours, which implies that 480 flow graphs must be evaluated for each distribution. (Note that each flow graph has

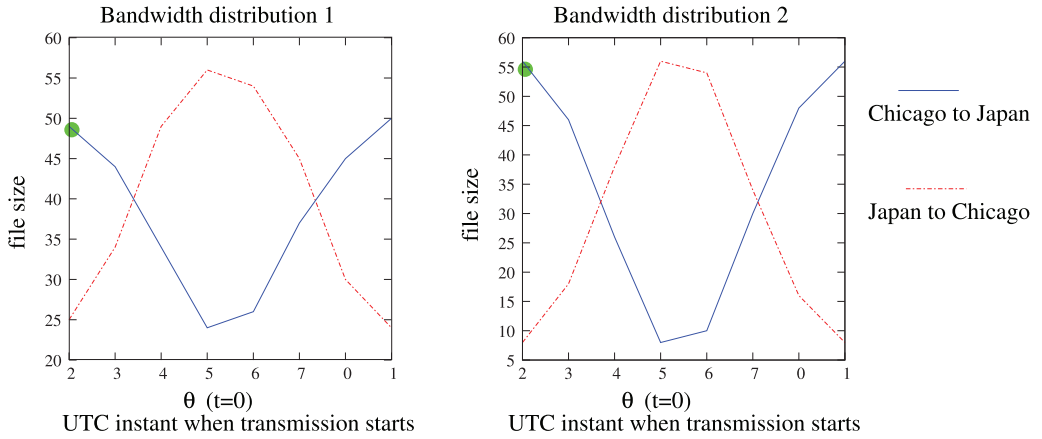

Fig. 9. Impact of start time on size of transmitted file over duration of at most 24 hours ($0 \leq t < 8$). The marked points on the plots correspond to total flow shown in the last row, first column of Tables 2 and 3.

480 subgraphs for a 24-hour duration.) For both distributions, a total of 960 flow graphs, each with 480 subgraphs, must be evaluated.

Figure 9 plots the maximum flow over a duration of 24 hours at each start instant: The first graph corresponds to distribution 1 and the second graph corresponds to distribution 2. The solid line plots the Chicago to Japan flow, and the dashed line plots the Japan to Chicago flow. (The Japan to Chicago flow is not required for solving the problem,r but is shown for comparison. These plots required construction of an additional 16 graphs.)

From the graphs, the maximum flow with distribution 1 is 50 units and occurs at flow start instant $\theta = 1$; the maximum flow with distribution 2 is 56 units and occurs at flow start instants $\theta = 1$ and $\theta = 2$. The time $\theta = 2$ corresponds to 06:00 UTC (midnight in Chicago). At $\theta = 2$, the maximum flow of 56 units takes 21 hours, while at $\theta = 1$ the maximum flow of 56 units takes 24 hours. The shorter duration of 21 hours is not indicated in the graph, but from the transfer schedule output by the maximum flow algorithm (Table 2). **From the graphs, we conclude that the best option is distribution 2: 56 units can flow from Chicago to Japan over a duration of 21 hours; the optimum start time is midnight in Chicago.**

Evaluating Table 2 to find best placement and minimum bandwidth of data centers: The transfer schedule for each se gment is shown in the table. From the transfer schedule, the minimum bandwidth distribution of the data centers is:

Alaska: U(3) = 10; U(4) = 12; U(5) = 14; U(6) = 8; U(0) = U(1) = U(2) = U(7) = 0
Argentina: U(3) = 4; U(4) = 4; U(0) = U(1) = U(2) = U(5) = U(6) = U(7) = 0
UK: U(0) = 8; U(2) = 2; U(3) = 6; U(1) = U(4) = U(5) = U(6) = U(7) = 0
Jordan: U(2) = 8; U(7) = 8; U(0) = U(1) = U(3) = U(4) = U(5) = U(6) = 0
Bhutan: U(6) = 2; U(7) = 2; U(0) = U(1) = U(2) = U(3) = U(4) = U(5) = 0
NZ: U(4) = 10; U(5) = 12; U(6) = 14; U(7) = 8; U(0) = U(1) = U(2) = U(3) = 0

For comparison purposes, in Table 3, we show the transfer schedule with distribution 1 and a maximum flow of 49. Data centers in Argentina and Bhutan do not appear on the schedule. Thus, the optimum placement of data centers with respect to this transfer schedule are in zones corresponding to the UK, Jordan, New Zealand, and Alaska. The optimum bandwidth for data center in the UK

Table 2. Bandwidth Distribution 2: Maximum Flow from Chicago to Japan with Start Instant at UTC 06:00 and Flow Duration of 21 Hours

Segment size	UTC 06:00 (τ=2,t=0)	UTC 09:00 (τ=3,t=1)	UTC 12:00 (τ=4,t=2)	UTC 15:00 (τ=5,t=3)	UTC 18:00 (τ=6,t=4)	UTC 21:00 (τ=7,t=5)	UTC 00:00 (τ=0,t=6)
8				Ch-to-Jp			
2		Ch-to-Ak		Ak-to-Jp			
8		Ch-to-Ak			Ak-to-Jp		
10			Ch-to-NZ		NZ-to-Jp		
8	Ch-to-Jd					Jd-to-Jp	
2			Ch-to-Ak	Ak-to-NZ	NZ-to-Jp		
2	Ch-to-UK						UK-to-Jp
6		Ch-to-UK					UK-to-Jp
6			Ch-to-Ak	Ak-to-NZ		NZ-to-Jp	
2		Ch-to-Ag	Ag-to-Ak	Ak-to-NZ		NZ-to-Jp	
2		Ch-to-Ag	Ag-to-Ak	Ak-to-NZ	NZ-to-Bh	Bh-to-Jp	
56*	0	0	0	10	20	18	8

Each row represents a segment flow path from sender to receiver via one or more hops. The first column of the last row presents the total units transferred from t = 0 until t = 6; other columns in the last row present the units transferred to Japan at the corresponding time instant.

Table 3. Bandwidth Distribution 1: Maximum Flow Output for Chicago to Japan Transfer with Start Time UTC 06:00 (Local Time in Chicago 12:00 AM) and Flow Duration of 24 Hours

Segment size	UTC 06:00 (τ=2,t=0)	UTC 09:00 (τ=3,t=1)	UTC 12:00 (τ=4,t=2)	UTC 15:00 (τ=5,t=3)	UTC 18:00 (τ=6,t=4)	UTC 21:00 (τ=7,t=5)	UTC 00:00 (τ=0,t=6)	UTC 03:00 (τ=1,t=7)
1	Ch-to-Jp							
2		Ch-to-Jp						
4			Ch-to-Jp					
8				Ch-to-Jp				
2		Ch-to-Ak		Ak-to-Jp				
3					Ch-to-Jp			
3		Ch-to-Ak			Ak-to-Jp			
5			Ch-to-NZ		NZ-to-Jp			
2			Ch-to-Ak		Ak-to-Jp			
1						Ch-to-Jp		
5	Ch-to-Jd					Jd-to-Jp		
2							Ch-to-Jp	
4	Ch-to-UK						UK-to-Jp	
1		Ch-to-UK					UK-to-Jp	
3			Ch-to-Ak	Ak-to-NZ		NZ-to-Jp		
3								Ch-to-Jp
49*	1	2	4	10	13	9	7	3

Each row presents a segment flow path from Chicago to Japan via one or more storage hops.

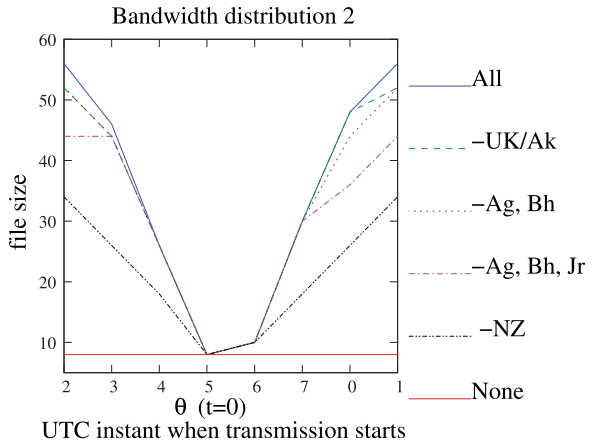

Fig. 10. Impact of positioning of storage hops on size of transmitted file. All: With all zones; –UK/Ak: minus either UK or Alaska; –Ag,Bh: minus both Argentina and Bhutan; –Ag,Bh,Jr: minus Ag, Bh, and Jordan; –NZ: minus NZ; None: no hops, direct sender to receiver.

is: 4 from [6:00-9:00), 1 from [9:00-12:00), 5 from [00:00-3:00). Similarly, the minimum bandwidth requirements for data centers in other zones can be computed.

Evaluating the impact of hops on maximum flow: In the problem statement, both distributions 1 and 2 place storage hops in zones 0, 3, 6, 12, 15, and 21. With distribution 2, a maximum flow of 56 units occurs over a duration of 21 hours (or 24 hours). Next, we evaluate the impact of hop placement on maximum flow. Figure 10 shows the impact of removing hops from one or more zones. The removal of the hop in New Zealand has a major negative impact on maximum flow. Removing one or more of the other hops also affects maximum flow. When the flow is direct (with no hops), the start time has no effect on maximum flow since 8 units are transmitted over 24 hours with each start instant.

Evalu ating the impact of sender/receiver on maximum flow: Figure 9 plots maximum flow from Chicago to Japan and from Japan to Chicago. The flow from Chicago to Japan is almost (anti)symmetrical to the flow from Japan to Chicago. This need not be true for other cases as we show in Figure 11, which plots the flow from Chicago to Argentina and from Argentina to Chicago for various start times with distribution 2. (A hop is placed in Japan and the hop in Argentina is removed.) The maximum flow from Argentina to Chicago is only 44 units, while the maximum flow from Chicago to Argentina is 56 units.

Evaluating Table 2 with reference to crowd-supported flow: Suppose the problem statement specifies crowd- supported flow with a micro-segment size of 2. To map from data center- to crowd-supported flow, segments must be transformed to micro-segments. Therefore, the segment of size 10 transferred from Chicago to Japan via New Zealand would be divided into five micro-segments. There would be five clients in New Zealand. All other segments of sizes greater than 2 would be similarly mapped to micro-segments, and hops would be mapped to clients.

7.1 GPSonflow App

This article has developed an application-level flow model for transfer of terabyte-scale data using standard file transfer protocols such as FTP, gridFTP, and HTTP. We expect terabyte-scale transfers will be handled by corporations (internet providers or cloud players) rather than by individual

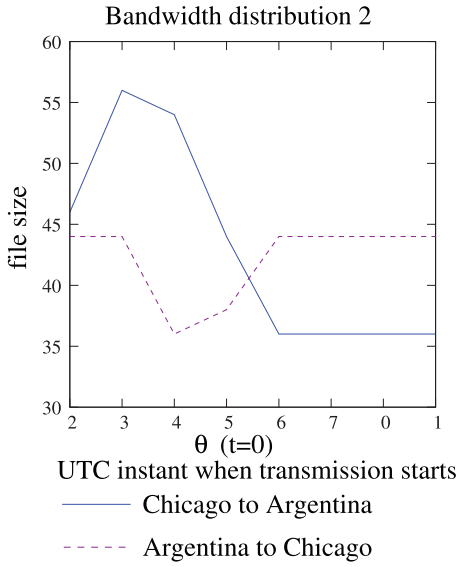

Fig. 11. Impact of start time on size of transmitted file over duration $\leq = 24$ hours ($0 \leq t < 8$).

users since there are several issues to be managed. First, the GPSonflow app has to monitor segment transfers and restart failed transmissions. Next, the app has to handle tasks such as division of a dataset into segments, encryption and compression at the sender, decryption and decompression at the receiver, and verification and recombination of segments into the original dataset. In addition to managing segment transfers, the GPSonflow app has to determine the inputs to Algorithm 1: For a given start time, duration of flow, and input bandwidth distribution, Algorithm 1 generates a flow graph, which is input to a maximum flow algorithm that generates the maximum flow and transfer schedule. To find the best transfer schedule for a sender and receiver, another program, *supervisor*, has to generate inputs to Algorithm 1 (value of time instant, start times, duration of flow, possible hop locations and bandwidths); the supervisor then evaluates the outputs and selects the best option.

The corporation (e.g., Amazon, Google or Microsoft) would run the supervisor program (and, therefore, Algorithm 1 and maximum flow algorithms) to determine the best transfer schedule for their sender and receiver clients. The sender's client program would handle file-to-segment division, encryption, compression, segment transmission at the appropriate times, and monitoring and retransmissions of failed transmissions. The receiver's client program would receive segments, decompress and decrypt, verify correctness, and recombine segments into the original dataset. Each of the data centers would also have programs that receive segments, verify correctness, and transmit on schedule.

If several users from the same end network want to transmit terabyte-scale data at the same time, there would be congestion even during the sleep cycle. Note that transfer schedules in both tables (of Section 7) have slack in some segment paths, where a *slack* refers to the time difference between segment arrival and segment transmission from storage hops. At a slack, it is possible for a segment to arrive later than its scheduled arrival time without impacting the maximal flow duration. The output of GPSonflow is a schedule, which could be input to a PERT/CPM algorithm to estimate latest arrival times of segments at hops. Thus, if terabyte-scale transfers at an end network are coordinated by the same organization, it may be possible to reduce congestion even when there are several terabyte-scale transfers.

8 CONCLUSION

This article evaluates GPSonflow, which is the problem of cheaply transmitting thousands of gigabytes from one end network to another end network via the internet. We extend prior work by focusing on model construction: We are the first to identify that the flow network for GPSonflow routing is a star graph, not a complete graph that mimics the dense, mesh architecture of the internet. Our insight has reduced the number of edges in the flow network from $O(n^2T)$ to $O(nT)$ where n is the number of storage hops and T is the duration of flow. Further, we model all data centers/clients in a zone by a single storage hop node, thereby bounding the number of hops to 24, the number of global time zones. Therefore, our flow network has a size complexity of O(T). Our flow network models both crowd- and data center-supported terabyte-scale transfers using bandwidth parameter values that are publicly available.

The flow network's parameters, such as arc capacities, are functions not only of graph time (i.e., flow instants) but also of time of day; this is the first work to formalize the mapping from graph time to system time. We have developed a data structure, Gclock, that outputs the local time instants of all nodes at every flow instant. Since arc capacities are functions of local time—a bounded domain—Gclock facilitates the automatic construction of the graph from a fixed number of arc capacities. We have developed an algorithm that constructs the O(T) flow network from $K \times \Gamma$ input parameters, where Γ is the number of time instants in a day and $2 \leq K \leq 26$ is the number of modeled nodes (sender + receiver + at most 24 hops).

To our knowledge, no previous papers have explicitly integrated the mapping from graph time to system time into graph construction and graph searching algorithms. Gclock is not specific to GPSonflow, so this is a useful tool for other applications where the graph parameter values vary with system parameter values.

Since arc capacities are function of time of day, the GPSonflow graph is periodic: The subgraph at t is identical to the subgraphs at $\Gamma + t$, $2\Gamma + t$, $3\Gamma + t \ldots$ The complexity of a maximum flow algorithm is dependent on the size of the graph. It might be possible to use graph periodicity to lower search complexity; such an algorithm might be useful for applications other than GPSonflow.

APPENDIXES

A FLOW NETWORK FOR DIRECT TRANSFER

The flow network for constant bandwidth is represented by a single star graph with two leaf nodes. The flow network for variable bandwidth is also represented by a single star graph with two leaf nodes, with one major difference: The capacities along the arcs (s, x), (x, r) vary with time. The leaf nodes s and r have storage capacity; the hub node has no storage capacity. This ensures that the flow network models end-to-end flow; a flow initiated from leaf node s at time t will arrive at a leaf node r at time t (or it could stay in node s until next flow instant t + 1). When flow is initiated from leaf node s to leaf node r, the flow cannot exceed the minimum of c(s,x,t) and c(x,r,t).

Dynamic flow model: For a flow duration t = 0 until t = T − 1 (UTC instants $\tau = \theta$ until $\tau = \theta +$ T − 1 mod Γ), the flow network $G^\star$=(V∪x, E,T) where

- V is the set of 2 nodes representing the the sender s and the receiver r
- x represents a single hub node; and
- E is the set of arcs $\{(s, x), (x, r)\}$.

$G^\star$ is a star network where the leaves are in node set V and the hub is node x. The nodes in V have infinite storage capacity, and the node x has zero storage capacity. The arcs in E have bandwidth capacity:

$c(s, x, t) = U(s,x,\tau)$
$c(x, r, t) = U(x,r, \tau)$
$\forall t = 0, 1, \ldots, T - 1$ where $\tau = \theta$ + t in modulo Γ arithmetic.

The majority of maximum flow algorithms ignore time by assuming instantaneous flow from sender to receiver; the corresponding flow network is static. The flow network for GPSonflow has arc capacities that vary with time; such a flow network is dynamic (Cai et al. 2007). Ford and Fulkerson (1958)) proposed the following solution for dynamic flows: Convert a dynamic flow to a static flow using a *time-expanded* flow network. The time- expanded flow network is a static network in which there is a copy of the graph for each time instant $0 \leq t <$ T. Figures 2 and 3 show the time-expanded version of Figure 1 when bandwidth varies with time, and duration is 4 time units. In these figures, there are 4 subgraphs, each representing an instant of time. The dashed lines are called *holdover* arcs and they represent storage capacity at leaf nodes. The holdover arc (s_t, s_{t+1}), models s at t storing data for transmission at $t + 1$. Storage capacity at end networks is not a bottleneck, so the capacity of holdover arcs is set to infinity.

Static (time-expanded) flow model: The dynamic flow network $G^\star$=(V ∪x,E,T) transforms to the static flow network G=($V_T \cup X_T$, $E_T \cup H_T$) where

- V_T is the set of end nodes s_t, r_t, t = 0, 1, . . . , T − 1;
- X_T is the set of hub nodes x_t, t = 0, 1, . . . , T − 1;
- E_T is the set of arcs (s_t, x_t), (x_t, r_t), t = 0, 1, . . . , T − 1;
- H_T is the set of holdover arcs (s_t, s_{t+1}), (r_t, r_{t+1}) t = 0, 1, . . . , T − 2.

The arcs have capacity:

$c(s_t, x_t) = c(s, x, t)$, $c(x_t, r_t) = c(x, r, t)$,
$c(s_t, s_{t+1}) = c(r_t, r_{t+1}) = \infty$.

The dynamic flow from s to r is equivalent to a corresponding static flow from s_0 to r_{T-1} (Ford and Fulkerson 1958). Therefore, finding maximum flow in the dynamic network can be solved by finding maximum flow in the corresponding static time-expanded network.

A flow in time-expanded G is a function
f: $(V_T \cup X_T) \times (V_T \cup X_T) \longrightarrow \mathbb{N}_0$ satisfying the properties of capacity constraint, skew symmetry, and flow conservation (Cormen et al. 2009). The value $| f_{max} |$ of the maximum flow f in G is
$| f_{max} |$ = f(s_0, x_0) + f(s_0, s_1) = f(x_{T-1}, r_{T-1}) + f(r_{T-2}, r_{T-1}) = $\sum_{t=0}^{T-1}$ minimum{c(s_t, x_t), c(x_t, r_t)}.

B FLOW NETWORK FOR INDIRECT TRANSFER

Dynamic flow network: For a flow duration t = 0 until t = T − 1 (UTC instants $\tau = \theta$ until $\tau = \theta$ + T − 1, $0 \leq \tau, \theta <$ Γ), $G^\star$=(V∪x, E,T) where

- V is the set of 26 nodes representing the sender s, the receiver r, and the storage hops in 24 time zones.
- x represents a single hub node; and
- E is the set of arcs $\{(s, x), (x, r)\} \cup \{(v, x), (x, v) \mid \forall v \in$ V-s-r }.

$G^\star$ is a star network where the leaves are in node set V and the hub is node x. The nodes in V have infinite storage capacity, and the node x has zero storage capacity. The arcs in E have bandwidth capacity:

$$c(v, x, t) = U(v, x, \theta + t) \qquad \forall (v, x) \in E$$
$$c(x, v, t) = U(x, v, \theta + t) \qquad \forall (x, v) \in E$$

$\forall t = 0, 1, \ldots, T - 1$ and θ + t is in modulo Γ arithmetic.

Static (time-expanded) flow network: The dynamic flow network $G^{\star}$=(V $\cup x$,E,T) transforms to the time-expanded flow network G=($V_T \cup X_T$, $E_T \cup H_T$) where

- V_T is the set of end nodes u_t, $\forall u \in V$ and t = 0, 1, . . . , T − 1;
- X_T is the set of hub nodes x_t, t = 0, 1, . . . , T − 1;
- E_T is the set of arcs (u_t, x_t), (x_t, u_t), $\forall (u, x), (x, u) \in E$ and t = 0, 1, . . . , T − 1;
- H_T is the set of holdover arcs (u_t, u_{t+1}), $\forall u \in V$ and t = 0, 1, . . . , T − 2.

The arcs have capacity:
$c(u_t, x_t) = c(u, x, t)$, $c(x_t, u_t) = c(x, u, t)$, and $c(u_t, u_{t+1}) = \infty$

ACKNOWLEDGMENTS

I thank the editors, Carey Williamson and Chuan Wu, for guiding the paper to publication. I thank the reviewers for their careful reading of the paper; I am grateful for their insightful comments and suggestions, which have greatly improved the paper. A special thanks to Adam Villa for his contribution in early stages of this research.

REFERENCES

A. Agapi, S. Soudan, M. Pasin, P. Vicat-Blanc Primet, and T. Kielmann. 2009. *Optimizing Deadline-Driven Bulk Data Transfers in Overlay Networks*. IEEE.

AMS. 2017. AMS IX Amsterdam Internet Exchange; statistics. Retrieved from https://www.ams-ix.net/.

Azure. 2017. Microsoft Azure data transfers pricing details. Retrieved from https://azure.microsoft.com/en-us/pricing/details/data-transfers/.

Jan Bot, Migiel de Vos, Sander Boele, Marcel Reinders, and Joost Kok. 2012. Enabling large genomic data transfers using nation-wide and international dynamic lightpaths. In *Proceedings of the 2012 IEEE 8th International Conference on E-Science* (2012), 1–2. DOI: http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/eScience.2012.6404458

Xiaoqiang Cai, Dan Sha, and C. K. Wong. 2007. *Time-Varying Network Optimization (International Series in Operations Research & Management Science)*. Springer.

Brian Cho and Indranil Gupta. 2011. Budget-constrained bulk data transfer via internet and shipping networks. In *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC'11)*. ACM, New York, 71–80. DOI: http://dx.doi.org/10.1145/1998582.1998595

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3 ed.). MIT Press, Cambridge, MA.

Sachin Date. 2016. Should you upload or ship big data to the cloud? *Communications of the ACM* 59, 7 (2016), 44–51. DOI: http://dx.doi.org/10.1145/2909493

DECIX. 2017. DE-CIX Where Networks Meet; Statistics. Retrieved from http://www.de-cix.net/about/statistics/.

Jack Edmonds and Richard M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM* 19, 2 (April 1972), 248–264. DOI: http://dx.doi.org/10.1145/321694.321699

Da Feng, Weiqiang Sun, and Weisheng Hu. 2017. Joint provisioning of lightpaths and storage in store-and-transfer wavelength-division multiplexing networks. *IEEE/OSA Journal of Optical Communications and Networking* 9, 3 (March 2017), 218–233.

Da Feng, Weiqiang Sun, Peng Wu, Xiaojian Zhang, Junmin Wu, and Weisheng Hu. 2016. Dimensioning of store-and-transfer WDM network with limited storage. In *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS'16)*. 214–221. DOI: http://dx.doi.org/10.1109/NOMS.2016.7502815

Yuan Feng, Baochun Li, and Bo Li. 2012. Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward. In *Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 43–50.

GIG. 2017. Global Internet Map. Retrieved from https://www.telegeography.com/telecom-maps/global-internet-map.html.

Google. 2017. Google cloud storage transfer service. Retrieved fromhttps://cloud.google.com/storage/transfer/.

L. R. Ford Jr. and D. R. Fulkerson. 1958. Constructing maximal dynamic flows from static flows. *Operations Research* 6 (1958), 419–433.

Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft. 2004. Structural analysis of network traffic flows. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance*. New York, NY, USA, 61–72.

Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. 2011. Inter-datacenter bulk transfers with Netstitcher. In *Proceedings of the ACM Special Interest Group on Data Communication SIGCOMM 2011 Conference*. New York, NY, 74–85.

Nikolaos Laoutaris, Georgios Smaragdakis, Rade Stanojevic, Pablo Rodriguez, and Ravi Sundaram. 2013. Delay-tolerant bulk data transfers on the internet. *IEEE/ACM Transactions on Networking* 21, 6 (December 2013), 1852–1865.

Chankyun Lee and June-Koo Kevin Rhee. 2017. Efficient design and scalable control for store-and-forward capable optical transport networks. *Journal of Optical Communications Networks* 9, 8 (Aug 2017), 699–710. DOI: http://dx.doi.org/10.1364/JOCN.9.000699

Xiao Lin, Weiqiang Sun, Malathi Veeraraghavan, and Weisheng Hu. 2016. Time-shifted multilayer graph: A routing framework for bulk data transfer in optical circuit-switched networks with assistive storage. *Journal of Optical Communications Networks* 8, 3 (Mar 2016), 162–174. DOI: http://dx.doi.org/10.1364/JOCN.8.000162

LINX. 2017. LINX London Internet Exchange. Retrieved from https://www.linx.net.

Patrick Maille, Gwendal Simon, and Bruno Tuffin. 2016. Vertical integration of CDN and network operator: Model and analysis. *Proceedings of the 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'16)*, 189–195. DOI: http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/MASCOTS.2016.24

Cong Shi, Mostafa H. Ammar, and Ellen W. Zegura. 2011. IDTT: Delay tolerant data transfer for P2P file sharing systems. In *GLOBECOM'11*. 1–5.

Snowball. 2017. Amazon's AWS Snowball. Retrieved from https://aws.amazon.com/snowball/.

Adam H. Villa and Elizabeth Varki. 2011. The feasibility of moving terabyte files between campus and cloud. In *Proceedings of the 23rd IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*. Dallas, TX, December 2011.

Received July 2017; revised January 2018; accepted March 2018