Modeling the internet for maximum flow when capacity varies with local time

Elizabeth Varki
University of New Hampshire
email: varki@cs.unh.edu

Abstract

Maximum flow is the research problem underlying transmission of bulk data sets on the internet since a key task is to find an optimal capacity route between the sender and receiver. In order to generate this route, it is necessary to construct the underlying flow network. In a discrete time model of the internet, the arc between 2 nodes represents the bandwidth capacity for flow between the networks at each flow instant t= 0, 1, ... T-1. A dynamic internet flow model consists of nT nodes and (n+m)T arcs; the internet is completely connected, so m=n(n-1). Thus, the state space count of the graph theoretic flow model is (n+m)T=n^2T. This paper develops a new data structure called clock net that models the dynamic flow network of the internet with nT states. The bandwidth capacity available for flow varies with traffic intensity which is a function of local time - bandwidth availability increases during users' sleep cycle. At each flow instant t, the local time at the nodes depends on their geographic placement; only some nodes have local times in the sleep phase. A feature of the clocked flow network is that at every flow instant t, given any local time, the nodes with this local time can be found in O(1) time. Since local time determines capacity, highest capacity paths can be found in O(1) time at every flow instant t. This paper develops a maximum flow algorithm for bulk transmission with complexity O(nT). As comparison, Netstitcher, a bulk transmission protocol based on the graph theoretic flow network uses Ford Fulkerson's algorithm which has complexity of O($|f|n^2Tlog(T)$) where $|f|$ is the maximum flow value. The contribution of this paper is clock net, a new data structure for indexing time expanded graphs that model systems where local time determines capacity.

1 Introduction

A dynamic flow network models maximum flow problems in systems where time is parameterized. There are several examples of systems modeled by dynamic flows [1]; this paper models dynamic flows in the context of bulk transmissions over the internet. In bulk transmissions, the maximum flow problem relates to the determination of the largest data set size that can be transmitted from a sender to a receiver during a given time period. This paper uses a discrete time model where flow time t is incremented in unit steps 0, 1, 2, ..., T-1. The flow network has n nodes that represent data centers with storage capacity and m arcs that represent links for flow between two nodes. An arc's bandwidth capacity and a node's storage capacity may vary with time depending on internet traffic.

The discrete time flow network is modeled as a time-expanded graph. A time expanded graph contains a copy of the node set of the underlying network for every discrete time step. Each set of n nodes and m arcs models the flow network at an instant t, where t=0,1,..,T-1 represents the elapsed time since flow initiation at t=0; there are a total of nT nodes and mT arcs. In addition, there is an arc from each node at t to its counterpart node at t+1 which carry excess flow from the node at t.

There are several types of dynamic flow problems depending on the settings of input parameters. For example, time can be discrete or continuous, capacity can be fixed or time varying, arc traversal times may vary, and nodes may allow storage. The classes of dynamic flow problems are presented in an overview paper by Kotnyek [5]. We consider flow problems that are discrete time with zero arc traversal times and flow capacities that vary with time. The property of capacity being dependent on time is generally a reflection of the dependence of traffic on time-of-day. The internet is an example of such a system since available bandwidth has a diurnal wave distribution with high bandwidth availability during the early morning hours [6].

Maximum flow in the context of bulk transmissions can be evaluated with existing algorithms. Most solution methods reduce the dynamic problem to a static form and then employ static maximum flow algorithms. Laoutaris et al. [7] have designed a transmission protocol using Ford and Fulkerson's algorithm [4] with time expanded graphs. Hoppe and Tardos [3] have developed a polynomial algorithm for the transshipment problem with discrete flow time when traversal times and capacities are fixed. Wilkinson [9] has presented a non-polynomial algorithm for universal maximal flow with the objective of earliest arrival time when time is discrete. Universal maximum flow is a flow that maximizes the flow at the sink at each time t. Ogier [8] and Fleischer [2] have evaluated universal flow in a network with zero transit time where node and storage capacities are piece-wise constant functions with T breakpoints. The run time of Ogier's universal maximum flow algorithm is determined by the time to solve O(nT) maximum flow problems on a network with nT vertexes and (n+m)T arcs. Fleischer's maximum flow algorithm is faster since it involves only one preflow-push maximum flow computation on a network with nT vertexes and (n+m)T arcs; the run time is O(T^2mnlog(Tn^2/m)).

We develop a new data structure called clock net to model the flow network underlying bulk transmissions. A clock net is essentially an index of the time expanded graph by local time - given a local time instant i, all nodes and paths with local time instant i are found in O(1) time. During each flow instant t, the maximum flow algorithm directs flow along the nodes and paths in the suitable local time zone.

The contributions of the paper are the following: 1) development of clock net, a new data structure that indexes dynamic flow networks by local time instants such that nodes and paths at a given local time instant can be found in O(1) time; 2) development of a clocked graph model for dynamic flow networks that has nT states as opposed to (n+m)T = n^2T states in the graph model; and 3) development of an O(nT) maximum flow algorithm that searches for paths based on local time of the nodes.

2 Motivating application

The motivating application is bulk transmission, which is the transmission of a large file from a sender to a receiver via the internet. The rate of transmission depends on bandwidth capacity; more capacity is available when internet traffic is low. For example, suppose backbone bandwidth can be purchased cheaply during the hours of 3:00 AM and 6:00 AM, the sender is a data center in Boston and the receiver is a data center in Alaska. Since Alaska is 4 hours behind Boston, there is no overlap of high capacity times. In order to take advantage of the cheap bandwidth, it is necessary to transmit portions of the file to data centers in times zones across the US, such as Chicago, Phoenix, and LA. The file portions are later transmitted to Alaska when bandwidth opens up. The next example presents a comprehensive transmission schedule when the sender and receiver are separated by several time zones.

Example 1: The objective is maximal flow from Chicago to Japan with the constraint that backbone bulk transmissions are allowed only during the hours of 12:00 AM to 12:00PM. The flow capacity that can be transmitted is as follows: 12:00 AM-3:00 AM: 10 flow units; 3:00 AM-6:00 AM: 20 flow units; 6:00 AM-9:00 AM: 18 flow units; 9:00 AM-12:00 PM:8 flow units. Assume that the data center networks have ample bandwidth, and the bottleneck is the backbone networks. The maximum flow that can be transmitted out of Chicago in a day is 56 units. However, this flow cannot be directly transmitted to Japan since Japan is ahead of Chicago by 15 hours; when it is 12:00 AM in Chicago, it is 3:00 PM in Japan. The only times of direct flow from Chicago to Japan is from 12:00 AM-3:00 AM Japan time which corresponds to 9:00 AM-12:00 PM Chicago time; a total of 8 flow units can be transmitted during this time period.
For this example, let data centers be located in time zones that are 3 hours apart. Starting from Chicago, the data centers, in UTC zone order are located in: Argentina, UK, Jordan, Bhutan, Japan, New Zealand, and Alaska. (Argentina is 3 hours ahead of Chicago, Alaska is 3 hours behind.)
The goal is to transmit 56 units of flow from Chicago to Japan within a 24 hour period. The transmission schedule in Table 1 satisfies this goal. Transmission from Chicago starts at 12:00 AM Chicago time (t=0). A time unit is 3 hours. The first column of Table 1 gives the transpired time t, the third column gives the transmissions fired at t, the second and fourth column gives the stored flow in the sending and receiving data centers, respectively, corresponding to each transmission. The last column lists the data centers that are in the time periods 12:00 AM to 12:00 PM. The transmission completes when 56 units of flow arrive in Japan 21 hours after the transmission was started in Chicago. □

In the example, there is exactly 1 data center (node) per time zone. In general, there can be 0 or more nodes per time zone; the link capacity of nodes may vary. The example assumes that all nodes have sufficient storage capacity, but storage capacity may also have a distribution which is dependent on clock time.

There are several maximum flow problems of interest. Some of them are:

1. compute the maximum flow for a given transmission start time and end time.

   If the transmission start time for the Chicago to Japan transmission is changed to 9:00 AM, then the maximum flow in the 24 hour period need not be 56 units of flow.

2. compute the transmission start time that results in maximum flow in a given time duration.

3. find optimal (time zone) locations for intermediate data centers that result in maximal flow.

   For example, transmission from NY to Alaska is maximized by data centers across the US; transmission from Alaska to NY is maximized by data centers in Asian and European time zones.

Table 1: Maximum flow routing from Chicago to Japan; transmission starts when it is 12:00 AM in Chicago and ends when it is 12:00 PM in Japan. Each flow instant t represents 3 hours

t	S flow	S – arc flow –> R	R flow	Nodes lying in local time range 12:00AM-12:00PM
0	56	Chicago – 2 –> UK	2	Chicago, Argentina, UK, Jordan
		Chicago – 8 –> Jordan	8	
1	46	Chicago – 10 –> Alaska	10	Alaska, Chicago, Argentina, UK
		Chicago – 4 –> Argentina	4	
		Chicago – 6 –> UK	8	
2	26	Chicago – 10 –> NZ	10	NZ, Alaska, Chicago, Argentina
		Chicago – 8 –> Alaska	18	
	4	Argentina – 4 –> Alaska	22	
3	8	Chicago – 8 –> Japan	8	Japan, NZ, Alaska, Chicago
	22	Alaska – 12 –> NZ	22	
	10	Alaska – 2 –> Japan	10	
4	8	Alaska – 8 –> Japan	18	Bhutan, Japan, NZ, Alaska
	22	NZ – 12 –> Japan	30	
	10	NZ – 2 –> Bhutan	2	
5	8	NZ – 8 –> Japan	38	Jordan, Bhutan, Japan, NZ
	2	Bhutan – 2 –> Japan	40	
	8	Jordan – 8 –> Japan	48	
6	8	UK – 8 –> Japan	56	UK, Jordan, Bhutan, Japan

4. for a given set of intermediate data centers, find the minimum bandwidth and storage capacity in order to ensure maximum flow from sender to server;

   If there are several data centers within the US, but only a couple of data centers across Asia and Europe, more bandwidth is required in the European and Asian data centers and backbone networks.

5. for a given file size, what is the start time that results in the quickest transmission;

   If the file size is less than 8 flow units, then starting the transmission at 9:00 AM Chicago time will result in direct transfer to Japan.

6. for a given file size and transmission completion time, what is the latest start time.

Sometimes, the only invariants in bulk transmission are the sender and receiver, and the goal is to transmit the file quickly and cheaply. The locations and bandwidth capacities of the intermediate nodes, the backbone bandwidth capacities, and the optimal start time are to be determined by the maximum flow algorithm. With each change in the value of an input parameter, a new flow model must be constructed and solved. Finding the optimal solution requires several executions of the maximum flow algorithm with various combinations of the input parameters.

3 Mapping local time and flow time

The graph theoretic flow network of bulk transmission models each data center as a node and the arcs represent links for flow between the nodes. The arc capacity of (n1,n2) is computed as the minimum of 4 components,

namely, 1) uplink capacity of n1, 2) backbone capacity of n1, 3) backbone capacity of n2, and 4) downlink capacity of n2. The arc's capacity varies with local time. Each node has storage capacity that may vary with local time. In the discrete time model, the flow is computed over T time units, t = 0, 1, ..., T-1. The flow starts transmitting from the sender at time t=0, and the receiver can receive flow until t=T. The flow network is modeled as a time expanded graph; there is one set of nodes for each flow instant t. For details on a time expanded graph model of bulk transmission, a reader is referred to NetStitcher [7].

In the time expanded graph model, the only time that is parameterized is flow time. However, local time is modeled implicitly in arc capacities and storage capacities. In Example 1, at flow instant t=0, it is 12:00 AM Chicago time and 3:00 PM Japan time. The time expanded graph models the local times at nodes through arc and node capacities. The effect of local time is implicitly modeled in the time expanded graph, but local time is not explicitly parameterized. Consequently, it is not possible to search the graph by local time. This paper develops a data structure called clock net that parameterizes local time by explicitly mapping flow instant t to local time instant i.

The value of i refers to the local time instant in node x at flow instant t. If local time at node x is i at t, then local time at node x is i+1 at t+1. At each unit increment of t, there is a unit increment of i. It is possible to find the value of the unit increment from the problem statement. In Example 1, the flow unit is 3 hours since bandwidth capacity is given in increments of 3 hours. Without loss of generality, let i=0 refer to 12:00 AM (0:00). The mapping from local time instant to hours of the day: i=0 to 0:00, i=1 to 3:00 AM, i=2 to 6:00 AM, ..., i=7 to 9:00 PM (21:00). At t=0, Chicago has i=0 (12:00 AM), Japan has i=5 (3:00 PM). At t=1, Chicago's i=1, Japan's i=6.

A flow instant maps to a local time instant which maps to hours of the day. At every instant, time is incremented by unit number of hour(s). After 24 hours, the capacity distribution of the flow network resets to the distribution at t=0. For simplicity of exposition, this paper considers flow during a 24 hour period. Let T represent the number of flow instant increments for the system to cycle 24 hours; every increment of t and i is equal to 24/T hour(s). Thus, the flow network uses modulo T arithmetic.

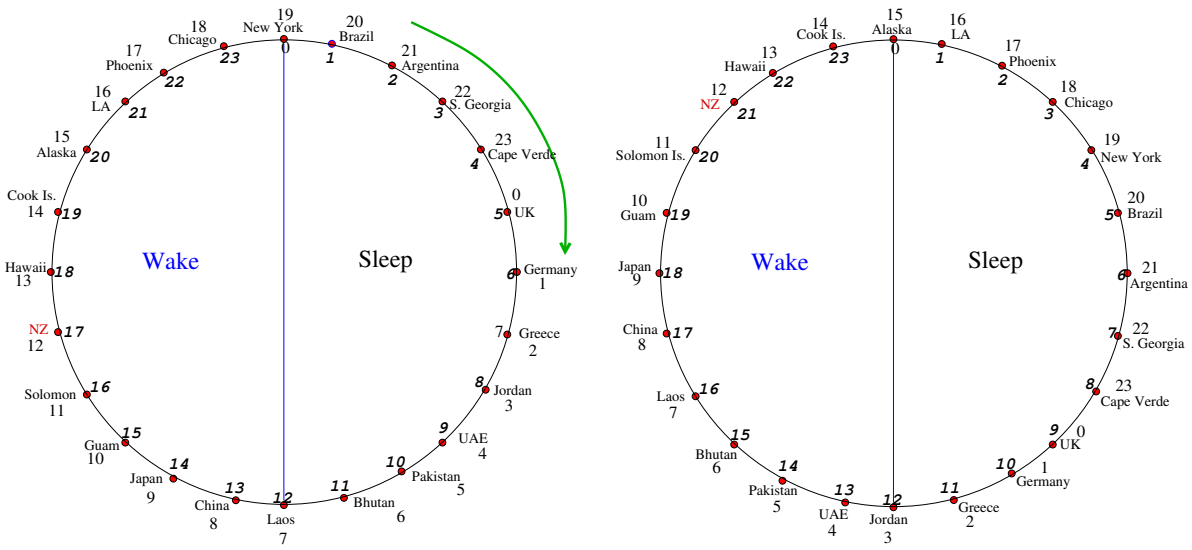
Definition 1 *For node x, if the local time at flow instant t is i, then the local time at instant t+1 is i+1; t=0,1, ..., T-1 and i=0, 1, ..., T-1. The local time i represents* $\frac{i \times 24}{T}$ *hours.*

In next section we present the global clock, a data structure which maps global positioning to local time and flow instants. The global clock forms the basis of clock net.

4 Global clock

The geographic map is partitioned into time zones, where all locations in a time zone have the same local time. When the geographic map is divided into 24 time zones, the local time at every location in a zone is an hour away from the local time at every location in its adjacent time zones. Without loss of generality, let Greenwich time zone (UTC=0) be designated as zone 0 and zone numbers increase by 1 in the clockwise direction; UTC + 1 is zone 1 and UTC - 1 is zone 23. All geographic positions in a time zone have the same local time. The geographic map is divided into UTC time zones; on this geographic map the minimum time difference between two UTC zones is 30 minutes. For example, Pakistan and India's local time differ by 30 minutes. The UTC zone is a geographic positioning parameter.

The global clock is a tool that maps global positioning and local time. The global clock is divided into zones, similar to the geographic map with UTC zones. However, the minimum time difference between the clock zones is not limited to 30 minutes. A global clock can be divided into any number of zones. If the



Configuration at t=0 when clock is started. Configuration at t=4.

Figure 1: Global clock with 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 zone numbers rotate clockwise while the circle remains fixed. The zones have shifted 4 positions to the right in the second figure.

zones are 5 minutes apart, then there are 288 zones where each zone is 5 minutes apart from its adjoining zone; when the local time in zone i is 5:10 AM, then the local time in zones i-1 and i+1 are 5:05 AM and 5:15 AM, respectively. By mapping global positioning to time zones, a global clock is a 1-1 function that maps positioning and local time. Henceforth, positioning and zones are used interchangeably.

The function of the global clock is to track the local time at all zones during 24 hours. The local time and zones are tracked at every flow instant, so local time and zones take discrete values 0, 1, ..., T-1. For example, consider the first clock in Figure 1; the clock has 24 zones, so the flow period modeled by this graph is t=0, 1, ..., 23. In the figure, the local times are the numbers inside the circle's circumference, the zones are the numbers outside. The name of a country/city in the zone is also shown in the figure. For example, it is 1:00 AM in Brazil, so the local time is 1 in zone 20. When the clock is divided into 288 zones, the local time and zones take discrete values from 0, 1, ..., 287, and T=288. Note that if the first clock in Figure 1 is divided into 288 zones where each zone is 5 minutes apart, then the zone and local time numbering would change. For example, Brazil would now be at local time (instant) 12; if UK remains as zone 0, Brazil would be zone 240; when it is 1:00 AM in Brazil, local time is 12 in zone 240 when T=288. (With 288 zones, not all clock zones have country names which map to UTC zones.) Reconsider the first clock in Figure 1 with T=8 and UK remaining as zone 0; the remaining zones would be in order, Jordan: zone 1, Bhutan: zone 2, Japan: zone 3; NZ: zone 4, Alaska: zone 5, Chicago: zone 6, and Argentina: zone 7. The zones are 3 hours apart, so if Chicago is at local time 0, UK would be at local time 2 (6:00 AM). Note that Brazil is no longer represented in this clock; for Brazil to be selected in a 8 zone map, the selected zones would be Brazil, Cape Verde, Greece, Pakistan, China, Solomon, Cook Is., and Phoenix.

The clock starts at time t=0. The initial zones to local time mapping is the configuration at time t = 0. The first clock in Figure 1 shows an example initial configuration at t=0. Knowing the local time of any single zone at t=0 is sufficient to find the local time at all zones in the clock at t=0. For example, if T=288 and it is known

that zone 0 is at local time 60, then zone 10 would be at local time 180 and zone 20 would be at local time 12 (mod 288 arithmetic).

Let t = 0, 1, ..., T-1 represent *clock time* which is the transpired time since the clock starts. After every 24/T hours, the zones shift 1 position to the right, and the clock's transpired time gets incremented by 1 (*i.e.,* t=t+1). The circle remains fixed while the zones rotate clockwise by 1 time unit for every unit increment of t. Each right shift is referred to as a clock tick which sets t=t+1. At each clock tick, the local time at each zone increments by 1 (in mod T arithmetic). The second clock in the figure shows the clock configuration at t = 4 when it is 5:00 AM in Brazil (local time in zone 20 is 5). The *cycle time* of the clock is T and is the time to execute a complete rotation of C. The cycle time may also be referred to as the clock's *period*.

The global clock is a tool that maps zones to local time at every clock tick. Given a zone's local time at clock time t, the zone-to-local time mapping for all zones can be computed for all past and future clock ticks in O(1) time. The initial configuration is defined by the mapping of any one zone to its local time. For example, in the first clock of Figure 1 the initial configuration can be set by mapping a single zone to its local time, say zone 20 (Brazil) to 1. The first clock of Figure 1 shows the configuration of all zones at time t=0. The second clock shows the configuration of all zones at t=4.

Definition 2 *The global clock, C, maps zones to their local time where the local time and zones take discrete values 0, 1, 2, ..., T-1. The local time i represents* $\frac{i \times 24}{T}$ *hour(s), and each zone is 24/T hour(s) away from its adjoining zones.*

C is modeled by inverse functions l and z where l(i,t) gives the local time in zone i at t, and z(j,t) gives the zone number with the local time j at t. The clock time $t = 0, 1, \cdots, T-1$ *represents time transpired from the instant C starts rotating.*

The initial configuration of C is given by l(i,0)=j, assigning a zone i to local time j at t=0.

The time unit of C is 24/T hour(s), so $t = t + 1$ *every 24/T hour(s).*

The configuration of C at t is given by $l(i,t) = j \iff z(j,t) = i; \;\; i, j \in \{0, 1, 2, \cdots, T-1\}$.

The rotation of C during time $0 \leq t < T$ *is defined by:* $l(i,t) = j \iff l(i, t+1) = j + 1$.

The relative positioning of time zones is defined by: $l(i,t) = j \iff l(i+1, t) = j + 1$.

in mod T arithmetic.

In Figure 1, T=24. In the first clock it is 1:00 AM in Brazil when t=0, so $l(20, 0) = 1, z(1, 0) = 20$. In the second clock, it is 5:00 AM in Brazil when t=4, and $l(20, 4) = 5, z(1, 4) = 16$.
By construction, for $\theta > 0$,

$$l(i, t+\theta) = l(i,t) + \theta; \quad z(i, t+\theta) = z(i,t) - \theta. \tag{1}$$

The functions l() and z() permit the tracking of zones and local times at every flow (clock) instant t.

5 Clock net

A clock net is a global clock with nodes, where nodes model data centers in UTC zones. A clock net maps nodes to local time and vice versa via the global clock - nodes (UTC zones) are mapped to clock zones, the

global clock maps clock zones and local time. Before presenting the formal definition of a clock net, we first explain the clock net and its variables with examples.

Example 1 (continued): The objective is maximal flow from Chicago to Japan. The nodes are located in Chicago, Argentina, UK, Jordan, Bhutan, Japan, New Zealand, and Alaska, ordered by UTC zones. At any instant, each node's local time is three hours away from its adjoining node's local time. The flow capacity that can be transmitted is as follows: 12:00 AM-3:00 AM: 10 flow units; 3:00 AM-6:00 AM: 20 flow units; 6:00 AM-9:00 AM: 18 flow units; 9:00 AM-12:00 PM:8 flow units.
The nodes are numbered from 0 to 7 in UTC order. Without loss of generality, let UK be node 0. It follows that Jordan is node 1, Bhutan is node 2, ..., Chicago is node 6, and Argentina is node 7.
Construction of clock net: Set T=24/3=8, where 3 equals the smallest UTC time difference between two nodes. In this simple example, the number of (clock) zones equals the number of nodes, with one node in each zone. The bandwidth distribution of node x at local time i is given by b[x,i], so b[x,0]=10, b[x,1]=20, b[x,2]=18, b[x,3]=8, b[x,4]=0, b[x,5]=0, b[x,6]=0, b[x,7]=0; x=0,1, ..., 7. A node x is mapped to (clock) zone i by p[x]=i, so p[0]=0, p[1]=1, p[2]=2, p[3]=3, p[4]=4, p[5]=5, p[6]=6, p[7]=7. □

Example 2: Reconsider Example 1, with an additional node in Phoenix. The nodes are in UTC order, UK (0), Jordan (1), Bhutan (2), Japan (3), New Zealand (4), Alaska (5), Phoenix (6), Chicago (7), Argentina (8).
Construction of clock net: The bandwidth distribution is in 3 hour periods, but the minimum UTC difference between two nodes is 1 hour since Phoenix is an hour behind Chicago. Set T=24/1=24. Since T is 24, the flow capacity for every hour must be computed. The array b[x,i] $\forall x = 0, 1, .., 8$ is defined as: b[x,0]=3.3; b[x,1]=3.3; b[x,2]=3.3; b[x,3]=6.6; b[x,4]=6.6; b[x,5]=6.6; b[x,6]=6; b[x,7]=6; b[x,8]=6; b[x,9]=2.6; b[x,10]=2.6; b[x,11]=2.6; b[x,12]=b[x,13]=...=b[x,23]=0;
Referring to Figure 1, the mapping from nodes to (clock) zones is: p[0]=0; p[1]=3; p[2]=6; p[3]=9; p[4]=12; p[5]=15; p[6]=17; p[7]=18; p[8]=21. Note that p[x]=i assignment ensures that the UTC positioning of nodes aligns with the (clock) zone positioning. □

Let n represent the total number of nodes. Nodes are numbered in UTC relative positioning order from 0 to n-1, with node 0 being chosen arbitrarily. For example, node 0 could be a node in UTC zone 0 or node 0 could be the sender node. A (clock) zone can have zero or more nodes. If a zone has more than one node, then the nodes in the zone are numbered consecutively. In Example 2, if there are three data centers in the Chicago UTC zone, then they are numbered as nodes 7, 8, 9, and Argentina's node is numbered 10 (T=24, n=10).

Definition 3 *The clock net N = (C, b, d, p) is modeled by global clock C, a 2-dimensional array b[x,i] storing the bandwidth capacity of node x at local time i, a 2-dimensional array d[x,i] storing the storage capacity of node x at local time i, and an array p[x] storing the zone positioning of node x, x=0, 1, ..., n-1.*
The nodes are numbered by their positioning, so $p[x] \leq p[x+1]$*; if p[x]=p[x+1] then nodes x and x+1 are in the same UTC zone.*
The start configuration of N(t=0) = C(t=0); the local time of node x at t=0 is given by l(p[x], 0).
The bandwidth and capacity of node x at t is given by b[x, l(p[x],t)], d[x, l(p[x],t)]; $t = 0, 1, \cdots, T-1$.
The period of N (and C) is T. During each clock tick, a node's local time is incremented, so l(p[x],t+1)=l(p[x],t)+1.

The array p[] is the mapping from UTC (geographic) zones to clock zones. The start configuration of clock net N is defined by the start configuration of C. A clock C has T possible start configurations since zone j can be set to local time 0, 1, ..., T-1. Since the sender has T start times, N has T start configurations.

5.1 Comparison of clock flows and graph flows

Bulk transmission is the motivating problem behind the development of clock net. Consequently, the design of clock net is closely related to the modeling of maximum flow over the internet. The internet's system design allows the implicit definition of arc capacity, so arcs are not defined explicitly. However, the definition of clock net can be generalized to include a graph G, so that N=(G, C, p). The clock net can be viewed as a graph theoretic flow network where nodes are numbered to indicate relative UTC positioning, and positioning array p[] maps nodes to zones.

Appendix A defines flows in a clock net defined for bulk transmission with arcs defined implicitly. When clock net is defined as (G, C, p), the flow definitions are similar to those defined for graph theoretic flow networks.

6 Algorithm

Before explaining the details of Algorithm 1, we present the underlying flow logic. Local time determines capacity, so at every flow instant, the algorithm searches for paths whose local times lie in the suitable range. The search takes O(1) time since a clock net has functions that map local time to nodes. The fundamental difference between a graph theoretic flow network and a clock theoretic flow network is the following: given a local time i, a clock theoretic flow network has functions that return nodes (and paths) with local time i; a graph theoretic flow network has to be searched to find nodes/paths with local time i.

A defining feature of the clocked flow network is the mapping from local time to nodes. In order to carry out this mapping, we define two arrays, lnode[0..T-1] and hnode[0..T-1] where lnode[j] stores the lowest node number located in zone j and hnode[j] stores the highest node number located in zone j. When there are no nodes in a zone j, lnode[j]=hnode[j]=-1. The arrays lnode[] and hnode[] are initialized during flow net construction along with the arrays b[], d[], and p[]. Recall that the clock net variable p[x]=j assigns a node x to zone j. This implies that lnode[j] $\leq$ x $\leq$ hnode[j]. The next two results follow from clock net construction.

Result 1 *Given a node x, the local time of x at flow instant t is l(p[x],t).*

Result 2 *Given a local time i, the nodes with local time i at flow instant t are lnode[z(i,t)]* $\leq x \leq$ *hnode[z(i,t)].*

The high level design of Algorithm 1 is as follows: each node keeps track of residual flow, a measure of the degree by which this node is time synchronized with the receiver r for the remaining flow duration. A positive residual value represents the number of flow units that can be transmitted to this intermediate node for later transmission to the receiver. The absolute value of a negative residual represents flow stored in the node that cannot be directly transmitted to the receiver. In order to ensure that this flow is not lost, this negative flow must be transmitted to a node with positive residual flow. At t=0, sender node s has a negative residual value representing the total flow units that have to be transmitted to the receiver r. The receiver node r has a positive residual value indicating the inflow bandwidth capacity of r during the flow period. During each flow instant, the algorithm scans nodes in the suitable local time range. The first stage of the algorithm is a push phase, where flow is pushed out to nodes that are more time synchronized with r. At the start of each flow period, the nodes with negative residual, starting with nodes that are at the end of their high capacity local times, transmit their flow to nodes with positive residual; these receiving nodes are also scanned in order starting with nodes at the end of their high capacity local times. Once receiving nodes with positive residual are exhausted, the nodes with negative residual transmit to nodes with negative residual; the sender nodes are closer to completing their high capacity local times than the receiving nodes. The receiving negative residual nodes have more time than

Algorithm 1: MAXIMUM FLOW WHEN CAPACITY IS DEPENDENT ON LOCAL TIME.

```
initialize ls
initialize e2e[x]; x=0,..,n-1
res[x]=e2e[x]; x=0,..,n-1
for t=0 to T-1 do
    // update flow that can be transmitted from x to r after this flow instant.
    update res[0..n-1];
    // push flow to nodes times synchronized with receiver.
    y=r;
    for x=z(ETIME) to z(STIME) do
        //push flow from nodes with negative residual flow.
        while b[x,l(x)]>0 and res[x] < 0 do
            // push flow to nodes with positive residual.
            // transfer() moves flow from x to y and updates b[x,t], b[y,t], f[x], f[y]
            if b[y,l(y)]>0 and res[y] > 0 then
                transfer(x,y);
            else
                if y==r then
                    y=z(ETIME);
                else
                    y- -;
                    // no more nodes with positive residual at t
                    if l(y) < STIME then
                        break;
        // push to nodes with < 0 residual; local time of receiving nodes are further away from ETIME.
        y=z(STIME);
        while b[x,l(x)]> 0 and res[x] < 0 do
            if b[y,l(y)] > 0 then
                transfer(x,y);
            else
                y++;
                // no more receiving nodes with negative residual earlier in the high capacity time range.
                if y ≥ x then
                    break;
    // pull to r from nodes that have bandwidth >0 and f[x]>0.
    x=z(ETIME);
    while b[r,l(r)] > 0 do
        if f(x) > 0 then
            transfer(x,r);
        x- -;
        if l(x) < STIME then
            break;
    // rotate clock net by 1 time unit.
    ls++;
```

the sender node in which to transmit their flow to nodes that are time synchronized with r. The second stage of the algorithm is a pull phase, where r initiates inflow from nodes that have stored flow units. At the end of each flow instant, the clock time is incremented and the next flow instant begins.

The first step in the algorithm is the construction of the flow network. During the flow network construction, the arrays b[] and p[] are initialized. The arrays lnode[] and hnode[] are generated from p[]. The sender s and receiver r nodes are inputs to the algorithm; the local time of the sender at t=0 is also an input parameter. As explained in Section 4, mapping a single zone to its local time is sufficient information to set the configuration of the entire clock (*i.e.,* the mapping of all zones to their local times). Let ls represent the start time of the sender's zone at time t; ls is initialized at t = 0, and at each increment of t, ls = ls + 1. The function l(j) which returns the local time of zone j is defined as l(j)=ls + (j - s) in mod T arithmetic. The function z(i) which return the zone with local time i is defined as z(i)=s + (i - ls) in mod T arithmetic.

We explain the main variables defined in Algorithm 1. Define constants STIME and ETIME to represent the start and end local times during which bulk flow is permitted. In Example 1, flow is permitted from 12:00 AM to 12:00 PM, so STIME=0 and ETIME=11 when T=24; STIME=0 and ETIME=3 when T=8. If flow is permitted during all times, then set STIME=0 and ETIME=T-1. Set b[x,i]=0 when i $>$ ETIME and i $<$ STIME since flow is not permitted outside the local time range STIME $\leq$ i $<$ ETIME. Define an array e2e[0..n-1] which stores the maximum end-to-end flow that is possible from node x to node r during flow period 0..T-1. Let δ=r-x. The array initialization is:

$$e2e[x] = \sum_{\theta=STIME}^{ETIME} \text{MIN}\{b(r,\theta), b(x,\theta+\delta)\}$$

The array e2e[] represents that maximum flow that can be transmitted from node x to r during the flow period T-1. This array is static and its value is unchanged for the algorithm duration. At flow instant t, the array res[0..n-1] represents the dynamic state of e2e[] by storing the remaining residual flow from node x to r. At the start of each flow instant res[x]=res[x]-b[x,l(p[x],t)]-f[x], where array variable f[x] represents the flow units stored in node x. When flow enters (or departs) x, update res[x] = res[x] - f(y,x,t) (or res[x] = res[x] + f(x,y,t)). The variable f[x] corresponds to flow f(x,t) defined in Section A. The computations of f(x,t) and f(x,y,t) are presented in Section A

Algorithm 1 presents the code outline. For notational convenience and code readability, the algorithm assumes that there is at most one node per zone so that there is a direct mapping from node to local time and vice versa (without using p[], lnode[], hnode[]). Bandwidth is set to zero when there are no nodes in a zone to allow the one-to-one mapping between node number and local time. The algorithm assumes that storage capacity at nodes is not a bottleneck.

7 Conclusions

The clocked flow network is a graph theoretic flow network with additional data structures to track the local time of nodes. In this paper, arcs are implicitly modeled, but the clock net definition could include both arcs and nodes. The implicit modeling of arcs is a consequence of this paper's focus on modeling the internet, not an inherent feature of clock net. A graph can be put into the framework of a clock net by capturing the local time/zone/node relation. The defining feature of clock net is the mapping from local time to nodes and vice versa. A graph theoretic flow network only records the relationship between nodes and flow instants, not nodes and local time. A clock net is an index by local time on graphs. For example, given a local time, the nodes with this local time can be found in O(1) time for every flow instant. The storage cost of the index are the arrays p[], lnode[], hnode[] that store the mapping between nodes and zones; the mapping between zones and local time is done by functions l() and z() and bear no storage cost.

References

[1] J. E. Aronson. A survey of dynamic network flows. *Ann. Oper. Res.*, 20(1-4):1–66, August 1989.

[2] Lisa Fleischer. Universally maximum flow with piecewise-constant capacities. In *IPCO*, pages 151–165, 1999.

[3] Bruce Hoppe and Éva Tardos. The quickest transshipment problem. In *SODA*, pages 512–521, 1995.

[4] L.R. Ford Jr. and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[5] Balzs Kotnyek. An annotated overview of dynamic network flows, 2003.

[6] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. In *SIGMETRICS'04*, pages 61–72, 2004.

[7] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. In *Proceedings of the ACM SIGCOMM 2011*.

[8] Richard G. Ogier. Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities. *Networks*, 18(4):303–318, 1988.

[9] W.L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.

A Flow in clock net

Node numbering reflects relative node positioning. This feature models paths between different nodes of a clock net modeling bulk transmission as explained here. Two nodes that are not adjacent to each other are connected via backbone networks over the intermediate zones that lie between the nodes. There are two paths from a sender node to a receiver node, one in the clockwise direction and another in the anti-clockwise direction. For example, the clockwise path from NY to Germany is via Brazil, Argentina, Cape Verde, and UK; the anti-clockwise path is via Chicago, Phoenix, ..., Jordan, and Greece.

To simplify the flow definitions, we assume that node storage capacity is not a bottleneck. Let f(x,y,t) represent flow between node x and node y at time t. Without loss of generality, let $p[x] < p[y]$. There exists 2 disjoint paths between x and y, one path is in the clockwise direction from x to y and the other path is in the anti-clockwise direction from x to y. The clockwise path is $\langle x, x+1, x+2, \cdots, y-1, y\rangle$ and the anti-clockwise path is $\langle x, x-1, \cdots, y+1, y\rangle$. Flow can be directed along either path. The clockwise flow from x to y at time t is given by:

f(x,y,t) $\leq$ MIN { b[x,l(p[x],t)], b[x+1,l(p[x+1],t)], $\cdots$, b[y,l(p[y],t)] }

The anti-clockwise flow from x to y is given by

f(x,y,t) $\leq$ MIN { b[x,l(p[x],t)], b[x-1,l(p[x-1],t)], $\cdots$, b[y,l(p[y],t)] }

Let the local times in x and y be τ_x and τ_y. That is, $l(p[x], t) = \tau_x$ and $l(p[y], t) = \tau_y$. From Equation 1, the flow equations can be written as:

f(x,y,t) $\leq$ MIN { b[x,τ_x], b[x+1,τ_x+1], $\cdots$, b[y,τ_y] }

The anti-clockwise flow from x to y is given by

f(x,y,t) $\leq$ MIN { b[x,τ_x], b[x-1,$\tau_x - 1$], $\cdots$, b[y,τ_y] }

The total outflow (inflow) of an end node at t cannot exceed the bandwidth capacity of the node at t. This gives,

$$\sum_{\forall y \neq x} f(x,y,t) \leq b(x,\tau_x), \text{ and } \sum_{\forall y \neq x} f(y,x,t) \leq b(x,\tau_x)$$

The above constraint on total flow at t only applies to end nodes in the end-to-end flow since the flow is modeled in context of the internet. There are one or more backbone links between two end networks, so each end-to-end flow at t between nodes x and y could be routed along different links. The only invariant nodes in the end-to-end flow are the end nodes x and y. The clockwise and anti-clockwise paths are relevant in that they represent the zone coverage of the backbone networks and exchange points over which the flow passes enroute from sender to receiver. If this assumption is not valid for the application being modeled, then the flow definition should add constraints on total flow at t for all nodes along the end-to-end flow path.

The flow stored in node x at time t, f(x, t), equals the total inflow into node x minus the total outflow out of x by time t. By the definition of f(x,y,t), it follows that

$$f(x,t) = \sum_{\forall y \neq x; \theta \leq t} f(y,x,\theta) - \sum_{\forall y \neq x; \theta \leq t} f(x,y,\theta)$$

At start of clock net rotation, only the sender has stored flow. That is, at $t = 0^-$, $f(s,0^-) > 0, f(x,0^-) = 0, \forall i \neq s$. At the end of clock net rotation, $f(r,T)$ is the value of flows sent from s to r within the cycle time T.

A clock net N has T possible start configurations since the flow start time at the sender can be set to T local times. Each start configuration results in a different flow network, so the total flow at the receiver may change. In Example 1, 56 flow units are transmitted from Chicago to Japan when flow is initiated from Chicago at 12:00 AM; if the flow start time is changed to some other time, say 6:00 AM Chicago time, the flow model changes and a maximum flow of 26 units is transmitted (this can be computed using Algorithm 1).

For a given start configuration, let λ specify a set of paths from sender to receiver, and f(λ, t) be the total flow with solution λ within the time limit $t < T$. That is, λ is the sender to receiver transmission routes generated by a routing algorithm. Then,

$$f(\lambda,t) = \sum_{i \neq r, \theta \leq t} f(i,r,\theta)$$

and

$$f(\lambda,t) = \sum_{i \neq s, \theta \leq t} f(s,i,\theta) - \sum_{i \neq s,r} f(x,t)$$

The notation $f(\lambda,T)$ refers to the value of flows sent from s to r within the time limit T using routing λ. For bulk transmission, $f(s,0^-)$ is the size of the bulk data set at the sender node just before transmission at time t=0, and $f(\lambda,T)$ is the size of the data set transmitted from s to r using routing paths λ within time T. Dynamic flow algorithms address the following optimization problems:

1. maximize the flow units that can be transmitted from sender to receiver within cycle period for all possible flow start times; and

2. find the start time and routing paths that minimizes the time to transmit a given set of flow units from sender to receiver.

In the domain of clock nets, the problems translate to the following objective functions.

Objective function 1 *For a given sender s and receiver r, generate* λ^{max} *such that* $f(\lambda^{max}, T) \geq f(\lambda, T), \forall \lambda \in N, \forall 0 \leq l(p[s], 0) < T$.

Objective function 2 *For a given* $f(s, 0^-)$*, find l(p[s],0) and generate* λ^* *where* $f(s, 0^-) = f(\lambda^*, \tau) > f(\lambda, t)\ \forall \lambda$ *in N when* $t < \tau$.