Information Engineering Express
International Institute of Applied Informatics
2015, Vol. 1, No. 1, 20 – 28

A Time-Dependent Graph Model for Describing Evolving Networks

Take-Yuki Nagao[*] , Antoine Bossard[*]

Abstract

A growing network like the Internet has a characteristic that the number of nodes is indefinite and may change with time. This characteristic makes it difficult to understand quantitatively and qualitatively how data are transferred among computers. In this article, we propose TFGM, which is a dynamic graph model to describe data transfer inside a network of indefinite size. The network is assumed to be time-dependent and allowed to change by discrete events. TFGM enables us to depict how data are sent from a server and to clients by using data densities, which depend continuously on time. As an application, fairness of load balancing is formulated and theorems are proven concerning an upper bound for the number of simultaneous clients for periodic polling.

Keywords: Dynamic graphs, load balancing, time-dependent graphs, time-evolution.

1 Introduction

In this article we develop a novel graph model, named TFGM (for Time Function Graph Model), that is intended to describe a growing network like the Internet with an increasing number of computers. The growth of a network makes it difficult to capture the nature of the network as a whole — in practice and in theory as well. Everybody knows that there are only finitely many computers connected to the Internet but nobody is able to tell the actual number of them at any time. In other words, the network is time-dependent and evolves with time. The problem addressed here is to develop a simple and practical theory to describe the data transfer over such evolving networks.

2 Graph as a Snapshot of the Network

TFGM presented here is designed to depict a system consisting of an indefinite number of nodes connected to each other by edges. The set of nodes and edges are denoted by V and E, respectively. Nodes of V are, for instance, servers that are connected to the network. The servers that will be connected to the network in the future are also contained in V. It is reasonable to regard V as a countable set but not assumed to be finite. Similar consideration applies to E.

[*] Advanced Institute of Industrial Technology, Tokyo, Japan

Nodes and edges are allowed to change with time but it is assumed that there is only a finite number of them at each moment. A time-line is needed to put this assumption into the theory. We use the set of real numbers $\mathbb{R}$ to represent the time-line with the origin fixed at a certain point in time. A graph in TFGM is a (total) function G with the following type.

$$G : \mathbb{R} \longrightarrow P(V) \times P(E)$$

Here $\times$ means a Cartesian product and $P(X)$ is the powerset of X in general. At any time $t \in \mathbb{R}$, the value $G(t)$ represents the snapshot of the network at t. $G(t)$ can be written as $G(t) = (V(t), E(t))$ for subsets $V(t) \subset V$ and $E(t) \subset E$. It is assumed that $G(t)$ is a directed graph for any t, and also that $V(t)$ and $E(t)$ are finite for any t.

The snapshot $G(t)$ may change at some time, for example by events like the connection of a new node to the network. We assume that such events are sparse in time, meaning that there is an increasing sequence $\{t_j\}_{j\in\omega}$ indexed by the set $\omega = \{0, 1, 2, \cdots\}$ of natural numbers such that $G(t)$ is constant except at t_j's and that $t_j \to \infty$ as $j \to \infty$.

3 Data Density for Modeling the Network

The data transfer among nodes can be modeled by introducing certain functions. To this end, let us write $(v \to w) \in E(t)$ to mean that there is an edge e starting from v to w at time t (e and $v \to w$ are regarded identical). We suppose that the node v is sending data to w via the connection e. The data density at the sender v is denoted by $\rho_e^{\text{out}}(t)$ (in bytes per seconds), so that the size of the data emitted from v within a time interval $[a, b]$ can be computed by the integral below.

$$\int_a^b \rho_e^{\text{out}}(t)dt$$

In a real network, the data transfer is subject to the delay, which is affected by factors, such as protocols, bandwidth, transactions, routing, transformation, access concentration, and others. In TFGM, the delay is modeled by the function $\ell_e(t)$, which depends on e and t. One byte of data transmitted from v at time t, reaches w at time $\tau = t + \ell_e(t)$. It is convenient to consider the function $f_e(t) = t + \ell_e(t)$. The role of f_e is to map the server time t to the client time τ. Let us call f_e the client time function. The behavior of f_e depends on the network condition and it is unknown in general. We model a general setting by supposing that $f_e(t)$ is a (strictly) increasing right-continuous function of $\mathbb{R}$ that satisfies $f_e(t) \geq t$ (or $\ell_e(t) \geq 0$ equivalently) for all t and $f_e(t) \to -\infty$ as $t \to -\infty$. We define $\rho_e^{\text{in}}(t)$ by

$$\int_a^b \rho_e^{\text{out}}(t)dt = \int_{f_e(a)}^{f_e(b)} \rho_e^{\text{in}}(\tau)d\tau \tag{1}$$

for all a and b with $a < b$ (in this article we write $\int_a^b$ to mean the integral over the half open interval $(a, b]$). The equation (1) means that no data are lost during the transactions. If $f_e(t)$ is a smooth function, then the equation (1) can be written with differentials:

$$\rho_e^{\text{out}}(t)dt = \rho_e^{\text{in}}(\tau)d\tau, \quad \tau = f_e(t). \tag{2}$$

This means that $\rho_e^{\text{in}}(\tau)$ is the data density received by w at client time τ. In general, we regard $d\tau$ as the Lebesgue-Stieltjes measure [1] associated with f_e, so that (2) holds in the sense of measures. We call ρ_e^{out} and ρ_e^{in} the outgoing and incoming density functions, respectively. If one knows either of them (by experiments for instance), then the other can be computed by the change of coordinates given in (2). For example, if we know the incoming density, we can write

$$\rho_e^{\text{out}}(t) = \rho_e^{\text{in}}(f_e(t))f_e'(t).$$

Here f_e' is the Radon-Nikodym derivative of $d\tau$ with respect to the Lebesgue measure dt. One of the main difficulties of network-related problems is that we have to control the clients and the servers without explicit knowledge of the function f_e.

A node has, in general, limitations about the bandwidth of network connections and about the number of edges that can be used simultaneously for data transfer. We model these constraints for each node v by the following inequality:

$$\sup_{a\in\mathbb{R}} \frac{1}{T}\int_a^{a+T} \rho_v^{\text{out}}(t)dt \le K_v, \tag{3}$$

where

$$\rho_v^{\text{out}}(t) = \sum_{e\in E_v(t)} \rho_e^{\text{out}}(t)$$

and $E_v(t)$ is the set of all $e \in E(t)$ such that $e = v \to w$ for some $w \neq v$. The time average is used here to model an observation experiment of duration T starting at time a, which is carried out to estimate the total outgoing data volume from v. The constant K_v (in bytes per seconds) represents the total capacity of the node v and $\rho_v^{\text{out}}(t)$ the total outgoing data density. Let us refer to the inequality (3) as the average capacity constraint. A sufficient condition for the average capacity constraint to be satisfied is the following inequality

$$\sup_t \rho_v^{\text{out}}(t) \le K_v. \tag{4}$$

Let us refer to (4) as the strong capacity constraint. The condition (3) is easier to verify than (4) in theory and in applications.

4 Achieving Load Balancing by Fairness

In this section we apply our framework to load balancing. We consider the situation where multiple clients poll data periodically from a single server. This type of data transfer can be used to distribute data — such as sensor data collected from various sources — by using a pull-protocol like HTTP (Hyper-Text Transfer Protocol). A common problem about this kind of data synchronization is the access concentration. We solve it by scheduling the requests from the clients. It is shown below that our solution is *fair* for all clients, with the fairness being rigorously formulated using the data density.

Suppose that $v \in V$ represents a server and that all other nodes are clients connected directly to v. The server v is required to send data uniformly to all clients $w \in V \setminus \{v\}$ (see Fig. 1). To achieve load balancing, the server must be *fair* in the sense that there is no special client that has a higher priority than another. There are multiple options to define what is meant by *fair*. In general, fairness is related with the behavior of clients regarding when and how to retrieve data from the server. Ideal fairness would be achieved if one could mediate all clients to cooperate so that $\rho_e^{\text{in}}(t)$ is constant for all t and e. This is unrealistic since the quantity $\rho_e^{\text{in}}(t)$ is subject to the noise caused by the network conditions, the failure of data transfer due to hardware and/or software, and numerous other reasons. We have a

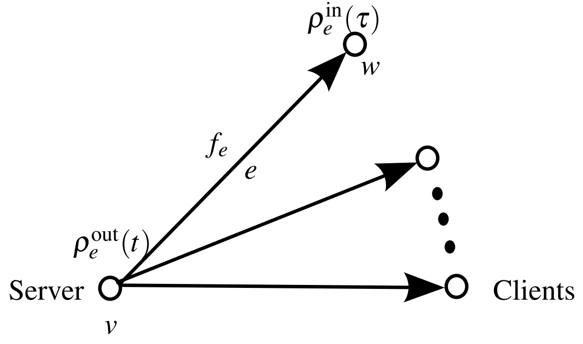

Figure 1: Periodic data transfer from the server v to clients (the outgoing data density ρ_e^{out} is related to the incoming data density ρ_e^{in} via the client time function: $\tau = f_e(t)$)

reasonable alternative using the mean value. Namely, we say that the data transfer is fair if the mean value

$$M^{\text{in}}(T,e,a) = \frac{1}{T}\int_a^{a+T} \rho_e^{\text{in}}(t)dt$$

is constant as a function of $a \in \mathbb{R}$ for a certain fixed value $T > 0$. The time average on the right hand side of the above definition can be observed at node w by recording the number of bytes transferred from v. As before, the parameter T is the duration of the observation. The mean value $M^{\text{in}}(T,e,a)$ computed at each node w, can be used to audit the status of load balancing — if it is varying rapidly with time, then the load balancing is not working. The duration T should be chosen large enough so that such an audit experiment can easily be carried out by any node. At least, T should be larger than all delays $\ell_e(t)$. In application, the condition that M^{in} is constant can be implemented by the condition $M^{\text{in}}(T,e,a) \in [M_0 - \varepsilon, M_0 + \varepsilon]$ for some fixed constant M_0 and a small $\varepsilon > 0$.

5 Fairness of Periodic Polling

We now model the periodic polling by specifying the incoming data density. Our key observation is that the outgoing data density can be estimated when we know the incoming data density. In other words, if we could control the timing and the data density of the requests from the clients, then we would be able to manage efficiently the data transfer for the polling.

Let P be the polling period in seconds. Suppose that a client $w \in V \setminus \{v\}$ initiates the k-th transaction at time $\tau_w + kP$ for $k = 0,1,2,\cdots$ with $\tau_w \in [0,P)$. We only consider the traffic from v to w, the communication in the other direction being ignored for simplicity. We assume further that the transactions do not overlap in time and that P is chosen large enough so that multiple transactions can be completed within the duration P. The incoming data density $\rho_e^{\text{in}}(\tau)$ at w from the server v can be decomposed into the sum

$$\rho_e^{\text{in}}(\tau) = \sum_{k=0}^{\infty} \rho_{w,k}^{\text{in}}(\tau),$$

where $\rho_{w,k}^{\text{in}}(\tau)$ is the incoming data density associated with the k-th transaction. Note that this summation is effectively finite since there can be at most one request at a time over the connection e. The periodicity of the requests is modeled by the following relation

$$\rho^{\text{in}}_{w,k}(\tau) = \psi(\tau - \tau_w - kP),$$

where ψ is a continuous and nonnegative function such that $\psi(t) = 0$ if and only if $t \notin (0, \delta)$ for some positive constant $\delta \ll P$. δ is the duration of the transaction, which is assumed to be independent of time. We remark that $\rho^{\text{in}}_e(\tau)$ is P-periodic with these settings. We further suppose that $\ell_e(t) \le \delta$ (or equivalently $f_e(t) \le t + \delta$).

Lemma 1. *Let g be a nonnegative periodic function on $\mathbb{R}$ with period P. Suppose that $0 < P \le T$. Then we have*

$$\left| \frac{1}{T} \int_a^{a+T} g(t)dt - \overline{g} \right| \le \overline{g}/n$$

with $n = \lfloor T/P \rfloor$ and $\overline{g} = \int_0^P g(t)dt/P$.

Applying this lemma to the incoming density function, we obtain the inequality

$$|M^{\text{in}}(T,e,a) - \mu| \le \mu/n, \quad \mu = \int \psi/P$$

with the same n as in Lemma 1. This means that the load balancing is fair if $P \ll T$, provided that the total outgoing data density $\rho^{\text{out}}_v(t)$ satisfies the capacity constraint.

6 Average Capacity Constraint

We now derive an inequality to describe the relationship among the number of clients, the network delay, and the total outgoing density function. To this end, let us denote by $c_v(T,a)$ the maximum number of clients connected to v within the interval $(a, a+T]$, i.e.,

$$c_v(T,a) = \sup_{t \in (a,a+T]} |E_v(t)|.$$

The change of delay for each connection can be measured by the difference quotient of f_e, which we define by

$$q_e(T,a) = \frac{f_e(a+T) - f_e(a)}{T}.$$

To describe the change of delay inside the whole network, it is convenient to use the supremum

$$q_v(T,a) = \sup_e q_e(T,a).$$

Remark that q_e and q_v are nonnegative. Moreover, by assumption on f_e, we have $q_e(T,a) \le 1 + \frac{\delta}{T}$ and thus $q_v(T,a) \le 1 + \frac{\delta}{T}$. We have $q_v = 1$ when the network has no delay at all. A larger value of q_v means greater change of delay.

The periodicity of ρ^{in}_e implies

$$\int_{f_e(a)}^{f_e(a+T)} \rho^{\text{in}}_e(\tau)d\tau \le (f_e(a+T) - f_e(a) + P) \int \psi/P$$

and so the summation over e yields the following

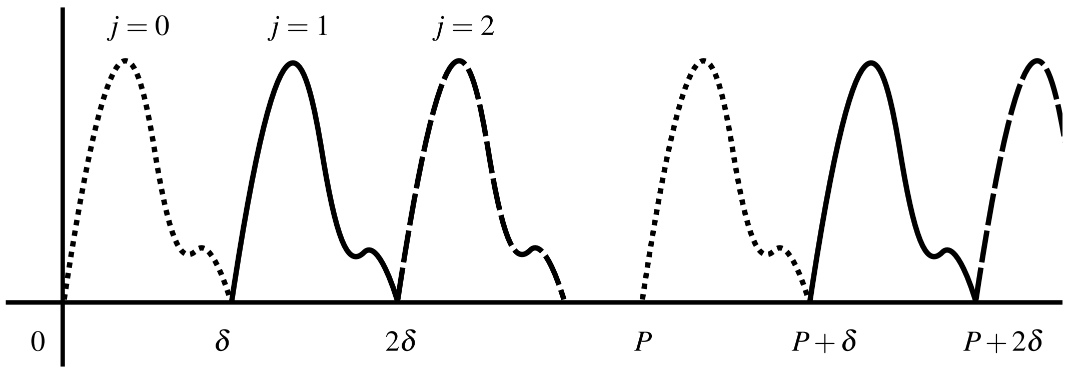

Figure 2: The aligned incoming density functions $\rho_e^{\text{in}}(\tau)$ for periodic polling ($m=3$)

Theorem 2. *For the periodic polling model, the following inequality holds with any $T,a>0$.*

$$\frac{1}{T}\int_a^{a+T}\rho_v^{\text{out}}(t)dt \le c_v(T,a)\left[\frac{q_v(T,a)}{P}+\frac{1}{T}\right]\int\psi \tag{5}$$

Theorem 2 has been formulated to facilitate the verification of the average capacity constraint, which can be done by comparing the capacity K_v with the right hand side of (5). Also, this theorem enables us to estimate the upper bound for the number of clients, given the capacity K_v. For example, it tells us that the server can serve up to

$$\frac{K_v}{L}\left[\frac{q_v(T,a)}{P}+\frac{1}{T}\right]^{-1} \tag{6}$$

clients, where $L=\int\psi$ represents the data size of each transaction. It should be noted that $q_e(T,a)$ can be computed at runtime by recording the timestamp of requests at the server. This means that the quantity (6) can be used to alter the maximum number of clients dynamically, depending on the delay observed at runtime.

Almost no property has been assumed so far about the start time τ_w for polling. We consider here how to choose τ_w for each client w in order to suppress the peak throughput of the network. It will be shown below that this can be achieved by aligning τ_w suitably. The worst alignment is the case where τ_w is a constant, say $\tau_w=0$ for all w. In this case, the data are sent from the server to all clients almost simultaneously. Such a concentration is inefficient since we are assuming the duration of transaction δ to be much smaller than the period P. The idea is to split the polling interval $[0,P)$ into the disjoint union

$$[0,P)=[m\delta,P)\cup\bigcup_{j=0}^{m-1}[j\delta,(j+1)\delta),\quad m=\lfloor P/\delta\rfloor$$

of length at most δ and to scatter the values of τ_w among these intervals. We model the general case of such scattering by a function $\xi:\mathbb{R}\times E\longrightarrow\omega$ and set $\tau_w=\xi(t,e)\delta$. The value of ξ is the index j of the sequence of intervals $\{[j\delta,(j+1)\delta]\}_{j=0}^{m-1}$ (see Fig. 2). It is convenient to have a quantity or an index to describe how uniformly the values of ξ are distributed. For this purpose, we use the level sets of ξ. Namely, we denote by $E_v(t,j)$ the set of all $e\in E_v(t)$ such that $\xi(t,e)=j$. Now consider the following quantities

$$\Phi_\xi(t)=\sup_j\frac{|E_v(t,j)|}{|E_v(t)|},\quad \Psi_\xi(t)=\sum_j\sup_{e\in E_v(t,j)}\rho_e^{\text{out}}(t).$$

The quantity $\Phi_\xi(t)$ is the maximum number of collisions of ξ relative to the number of edges. In general, a smaller value of $\Phi_\xi(t)$ means better uniformity of ξ. Remark that

$$|\mathrm{im}\xi|^{-1} \le \Phi_\xi(t) \le 1$$

for all t, where $\mathrm{im}\xi$ means the image (or the range) of ξ. With these quantities in hand, we can estimate the outgoing density:

$$\rho_v^{\mathrm{out}}(t) = \sum_j \sum_{e\in E_v(t,j)} \rho_e^{\mathrm{out}}(t) \le |E_v(t)|\Phi_\xi(t)\Psi_\xi(t).$$

We have thus proven the

Lemma 3. *Let $\xi : \mathbb{R}\times E \longrightarrow \omega$ be a function. Then the outgoing density satisfies the following inequality for all $t > 0$.*

$$\rho_v^{\mathrm{out}}(t) \le |E_v(t)|\Phi_\xi(t)\Psi_\xi(t)$$

For the periodic-polling model, we can estimate $\Psi_\xi(t)$ from above. Indeed, if e, j and t satisfy $e \in E_v(t, j)$, then we have

$$\rho_e^{\mathrm{out}}(t) = \rho_e^{\mathrm{in}}(f_e(t))f_e'(t) = \sum_{k=0}^{\infty} \psi(f_e(t) - j\delta - kP)f_e'(t).$$

Note that the inequality $t \le f_e(t) \le t+\delta$ implies

$$\psi(f_e(t) - j\delta - kP) \le \psi_\infty(t - j\delta - kP) + \psi_\infty(t-(j-1)\delta - kP),$$

where

$$\psi_\infty(t) = \begin{cases} \sup_t \psi(t) & t\in[0,\delta] \\ 0 & \text{otherwise.}\end{cases}$$

It follows that $\Psi_\xi(t) \le 2\sup_{e\in E_v(t)} f_e'(t)\sup_t \psi(t)$. We have proven the following

Theorem 4. *For the periodic polling model, the following inequality holds for all $t > 0$:*

$$\rho_v^{\mathrm{out}}(t) \le 2|E_v(t)|\Phi_\xi(t) \sup_{e\in E_v(t)} f_e'(t) \sup_t \psi(t).$$

In application, Theorem 4 can be used to verify the strict capacity constraint. Moreover, the theorem tells that

$$\frac{K_v}{2\Phi_\xi(t)\sup_{e\in E_v(t)} f_e'(t)\sup_t \psi}$$

clients can be served simultaneously under a given capacity K_v. This means that one should choose the scattering function ξ so that $\Phi_\xi(t)$ is close to its lower bound $|\mathrm{im}\xi|^{-1} = 1/m$.

7 Application to a Real Network

For implementing data distribution using periodic polling, one can define the scattering function ξ by using the IP address assigned to each node in combination with a hash function [2]. A simplest implementation is to take an exclusive or, $\mathrm{IP}(v)\oplus\mathrm{IP}(w)$, of the server

IPv4 address $\mathrm{IP}(v)$ (considered as a 32-bit unsigned integer), and the client IPv4 address $\mathrm{IP}(w)$ and set

$$\xi(t, v \mapsto w) = h(\mathrm{IP}(v) \oplus \mathrm{IP}(w)) \mod m.$$

Here the function h maps a 32-bit unsigned integer to another such number, and can be chosen from known hash functions [2]. The scattering function ξ is time-independent in this case. The IPv6 addresses can be used similarly. For a hash function with good uniformity, one expects $\Phi_\xi(t) \sim 1/m$. Moreover, if the distribution of the values of $\mathrm{IP}(v)$ and $\mathrm{IP}(w)$ is known, and if there are multiple options for the hash functions, then one can choose a hash function that minimizes the time average of $\Phi_\xi(t)$.

8 Related Work

TFGM proposed in this article falls into a class of dynamic edge networks [3]. Our approach is original in that we have chosen the weight f_e from functions and not from a number system. The role of the client time function f_e is to change time variables between the ends of each edge. Although we have modeled the evolution of a dynamic graph using a sequence of static graphs following [3], the data densities and the client time function can depend continuously on time. These time-dependent quantities are crucial for describing how data are transferred among nodes inside an evolving network.

One of the benefits of TFGM is that one can estimate the total outgoing density ρ_v^{out} by aggregating all the incoming densities ρ_e^{in}. This fact has been exploited when modeling the periodic polling and was a key to prove Theorems 2 and 4. Similar results for a different model were shown by the first author in a previous work [4].

9 Conclusion

The outgoing and incoming data densities of TFGM describe how data are transferred between edges. They are important quantities that allow us to better understand networks consisting of an indefinite number of nodes. Indeed, as we have seen above, the data densities allow us to quantitatively formulate such notion as the fairness of load balancing. Although we have focused on the data density in this article, the cost or value density of data can equally be considered. For example, the value of data decays with time in a streaming system like a video conference system — a ten-minute-old video frame is no longer relevant to the users. TFGM can also be used to model such real-time communication systems. TFGM is an optimistic model in the sense that no packets are assumed to be lost during transmission. It is an interesting question as to how to extend the model to accommodate for network with faults.

References

[1] G. B. Folland, Real Analysis, John Wiley & Sons, 1999.

[2] C. Henke, C. Schmoll, and T. Zseby, “Empirical evaluation of hash functions for multipoint measurements,” SIGCOMM Computer Communication Review, ACM, 2008; doi:10.1145/1384609.1384614.

[3] F. Harary and G. Gupta, “Dynamic graph models,” Mathematical and Computer Modelling, vol. 25, no. 7, 1997, pp. 79–87.

[4] T. Nagao, “A method to estimate data volume of awareness information using a continuous time model,” Bulletin of Advanced Institute of Industrial Technology, vol. 3, 2010, pp. 183–188.