Stochastic Fluid Models for Cache Clusters

Florence Clévenot, Philippe Nain

INRIA, BP 93, 06902 Sophia Antipolis, France.

Keith W. Ross

Polytechnic University, Six MetroTech Center, Brooklyn, NY 11201, USA

Abstract

Clusters of Web caches are extensively used by different types of organizations, including companies, universities, ISPs, and CDNs. To model Web caches, we must account for two types of stochastic events: objects being pulled into/out of the cache cluster at random times, and caches going up and down at random times. Detailed stochastic models of such complex systems quickly become intractable. In this paper we propose a stochastic fluid model which captures the salient characteristics of a cache cluster. The stochastic fluid model replaces the object arrivals to the cluster and departures (object modification/expiration) with a fluid flow, but maintains the up/down dynamics of the original system. The model can be applied to a variety of cluster routing policies, and provides a simple means to estimate the hit rate. We compare the results of the stochastic fluid model with that of a simulation of the real system. We find the fluid model to not only be a close approximation, but also to exhibit the key qualitative properties of the original system. We conclude that stochastic fluid models show great potential in modeling a variety of content distribution systems.

Key words: Web caching, cache clusters, fluid model, stochastic processes, Palm calculus.

1 Introduction

Beginning with the seminal work of Anick, Mitra and Sondhi in 1982 [1], stochastic fluid models have been successfully applied to a variety of packet-

Email addresses: Florence.Clevenot@sophia.inria.fr (Florence Clévenot), Philippe.Nain@sophia.inria.fr (Philippe Nain), ross@poly.edu (Keith W. Ross).

switching systems over the past 20 years (e.g., see [2–7]). In these papers, detailed models of system behavior, which involve random arrivals of packets of discrete size to network nodes, are replaced by macroscopic models that substitute fluid flows for packet streams. The rates of the fluid flows are typically modulated by a stochastic process (such as a Markov process), thereby resulting in a "stochastic fluid model". Although the resulting stochastic fluid models ignore the detailed, packet-level interactions, often they are mathematically tractable and accurate, providing significant insight into the qualitative properties of the original system.

We believe that stochastic fluid models are promising tools for modeling content distribution systems. In this paper we illustrate the approach by focusing on clusters of Web caches. Instead of replacing packet streams with fluid (as is done in the packet network modeling literature), we are proposing replacing content – such as images, MP3s, text files, video clips – with fluid. The motivation for this approach is that caches up/down dynamics introduce high complexity to the discrete model. Hence, classical tools such as Markovian analysis and simulation appear to be untractable for the dynamic system. The model that we discuss in this paper is also appropriate for P2P cooperative caches, for which objects are cached cooperatively in a community of up/down peers [8].

Previous studies on cache cluster modeling and analysis mainly focus on static cluster population, i.e., the number of caches in the cluster is constant [9,10]. [11] provides an analytical model for comparing ICP and hash-based cache clusters. The model includes nodes dynamics to compare two hashing schemes but only at join/leave instants, and the expected hit probability is chosen as a parameter and not as a metric. In this paper, we compute the expected hit rate in the presence of cache dynamics.

In Section 2 we provide an overview of hash-based cache clusters. We describe in Section 3 a reference detailed model, then we introduce a general stochastic fluid model which can model a wide variety of content distribution systems. A specialized version of the model is appropriate for modeling cache clusters. Section 4 provides the principal contributions of the paper. We describe the fluid evolution and show that the hit rate can be easily obtained, from a tridiagonal linear system of dimension N where N is the number of caches in the cluster. We provide explicit, closed-form expressions for $N = 2$ in Section 5, which provide insight into performance issues of cache clusters. Our analysis shows that two key systems parameters largely determine the performance of the system. We also use the results of the stochastic fluid model to compare two natural direction policies, namely, "partitioning" and "winning". In Section 6 we compare the theoretical results from our fluid model with a simulation of the detailed model introduced in Section 3.1. We find that the fluid model is largely accurate and has the same qualitative behavior as the detailed model.

2 Overview of Cache Clusters

A cache cluster consists of N nodes (that is, caches). The nodes hold objects. Each of the caches may go down at random times. There are three event types in the cache cluster: node going up, node going down, and request for an object. When a client in the organization makes a request for an object, the request is sent to one of the up caches. If the up cache receiving the request has the object, it immediately sends a copy to the client. Otherwise, the cache retrieves a copy of the object from the origin server, puts a copy in its storage, and sends a copy to the client. Because caches are going up and down at relatively slow time scales compared to requests, we assume throughout that each client always knows which caches are up, that is, each client tracks the set of active caches. (This is typically done by configuring each browser to retrieve a *proxy automatic configuration (PAC) file* each time the browser is launched. The PAC file indicates which caches are currently up, and also implements the direction policy as discussed later in this section.)

It remains to specify how a client, requesting a particular object, determines to which cache it should direct its request. This is specified by the *direction policy.* Ideally, to avoid object duplication across caches, we want requests from different clients for the same object to be directed to the same cache in the cluster. This ensures that at most one copy of any object resides in the cache cluster. Also, we would like the request load to be evenly balanced among the caches in the cluster. These two goals are often achieved by using a common mapping at all the clients. When a client wants an object, it maps the object name (typically a URL) to a specific cache in the cluster, and directs its request to the resulting cache. This mapping can be created with a hash function as follows. Let $h(\cdot)$ be a hash function that maps object names to the set $[0, 1)$. Let i be the number of up caches. Partition $[0, 1)$ into i intervals of equal length, $\Psi_1 = [0, 1/i)$, $\Psi_2 = [1/i, 2/i)$, …, $\Psi_i = [1 - 1/i, 1)$. Associate one up cache with each of these intervals. Then when a client makes a request for object o, it calculates $h(o)$ and determines the interval Ψ_j for which $h(o) \in \Psi_j$. It then directs its request for object o to the jth cache. We refer to this direction policy as *partition hashing.* If the hash function has good dispersion properties, partition hashing should balance the load among the caches in a more-or-less equitable manner.

Partition hashing has a serious flaw, however. When a new cache is added or goes down, approximately 50% of all the cached objects are cached in the wrong caches [11]. This implies that after an up/down event, approximately 50% of the requests will be directed to the wrong up cache, causing "misses" even when the object is present in the cluster. Furthermore, partition hashing will create significant duplication of objects after an up/down event. (Because the caches employ cache replacement policies, such as *least recently used*

(LRU), this duplication will eventually be purged from the system.)

To resolve this problem, independent teams of researchers have proposed refined hashing techniques, including CARP and consistent hashing, which route requests to their correct caches with high probability even after a failure/installation event [12,13]. Such robust hashing techniques have been used in Microsoft and Netscape caching products, and also appear to have been implemented in the Akamai content distribution network. We now briefly describe CARP; consistent hashing is similar. CARP uses a hash function $h(o, j)$ that is both a function of the object name o and the cache name j. When the client wants to obtain object o, it calculates the hash function $h(o, j)$ for each j, and finds the cache j^* that maximizes $h(o, j)$. We henceforth refer to this technique as *winning hashing*. The principal feature of winning hashing is that relatively few objects in the cluster become misplaced after an up/down event [11]. Specifically, when the number of active caches increases from j to $j + 1$, only the fraction $1/(j + 1)$ of the currently correctly-placed objects become incorrectly placed; furthermore, when the number of up nodes decreases from $j + 1$ to j, none of the currently correctly-placed objects become misplaced.

The model that we describe in the subsequent section is also appropriate for when a community of users (for example, employees in a large corporate campus) pool their peer computers to create a cooperative cache [8]. The resulting large storage capacity is highly desirable for caching large audio and video files. In such a P2P cache, the individual nodes go down when peers disconnect.

3 A Model for Cache Clusters

3.1 Dynamic Microscopic Model

Let $O = \{o_1, o_2, \dots, o_c\}$ be the collection of all objects. We assume that each cache has a capacity of B objects. Let O_j be the set of all objects currently in node j. Let U be the set of nodes in $\{1, \dots, N\}$ that are currently up. We assume that nodes go up and down independently of each other, the time until a given up (resp. down) node goes down (resp. up) is exponentially distributed with rate μ (resp. λ). The resulting process $N(t) = |U| \in \{0, 1, \dots, N\}$ is a particular birth-death process, known in the literature as the *Ehrenfest* model. Let N^∞ denote the stationary number of caches which are up. Setting $\rho = \lambda/\mu$, we have [14, p. 17]

$$P(N^\infty = i) = \binom{N}{i} \frac{\rho^i}{(1+\rho)^N}. \tag{1}$$

In particular, the expected number of caches which are up in steady-state is

$$\mathbb{E}\left[N^{\infty}\right] = \frac{N\rho}{1+\rho}. \tag{2}$$

Requests for objects arrive at random times $a_1 < a_2 < \cdots$. We assume that this random point process (e.g. a Poisson process) has an intensity σ. Moreover, different objects are requested according to a given probability distribution (typically a Zipf distribution [15]). Last, we assume that each cached document has a time-to-live (TTL) associated with it (see Section 3.2 for details).

The description of the dynamic model is completed by defining a direction policy (see Section 2) and specifying an associated hash function. The resulting detailed model, although accurately reflecting the real-world system, is unwieldy for traditional microscopic (including Markovian) analysis. Discrete-event simulation of this complex system can also be very slow.

3.2 Stochastic Fluid Model

We now propose a broad class of fluid models which capture many of the salient characteristics of content distribution systems. The basic observation is that requests for objects occur at a much faster time scale than the nodes up/down events. We are therefore motivated to replace the arrivals of new objects to the up nodes – which occur at discrete times in reality – with a fluid flow. Specifically, let x_j denote the number of objects currently stored in node $j \in U$. In between up/down events we suppose that x_j grows at a continuous rate. This growth corresponds to a request directed to node j and not being immediately satisfied; node j then retrieves the object from the origin server and stores a copy, causing x_j to increase. This growth is slowed down by object expirations, which can also be modeled as a fluid flow going out of the system.

We further simplify the fluid model by supposing that the direction policy balances the load across all of the up nodes in the cluster. With this simplified description, the state becomes (i, x), where i, $i = 0, 1, \ldots, N$, is the number of up nodes and x, $0 \leq x \leq c$ is the total amount of fluid in the system. Also, c denotes the total amount of fluid in the universe.

We now define the dynamics of our fluid model. Similarly to the detailed model described in section 3.1, when there are i up nodes, nodes go down at rate $i\mu$ and nodes go up at rate $(N-i)\lambda$. In between up/down events, requests occur at rate σ and each cached object expires at rate θ.

Let us denote $p(i, x)$ as the steady-state probability of a hit when the state is

(i, x). (We shall indicate how $p(i, x)$ is modeled shortly.) The content increases whenever there is a miss event. Therefore, a natural model for the rate at which the fluid increases in the system between up/down events is $\sigma[1-p(i, x)]$. However, the content decreases at the constant rate θ due to object expirations. Therefore, the fluid increase rate in state (i, x) is actually $\sigma[1-p(i, x)]-\theta x$.

Now consider how the fluid level changes at an up/down event. In the general model, we allow for an instantaneous change in the fluid level after an up/down event. Specifically, when in state (i, x) and a node goes down, a fraction of the current fluid is instantaneously removed from the content distribution system. The remaining amount of fluid is $\Delta_d(i)x$. Similarly, when in state (i, x) and a node goes up, the fraction $\Delta_u(i)$ of the current fluid remains in the system.

Our general stochastic fluid model is now defined. The parameters in the model are N, c, λ, μ, σ, θ, B, $p(i, x)$, $\Delta_u(i)$ and $\Delta_d(i)$. The general stochastic fluid model can model a variety of content distribution systems, including server farms, cache clusters, and P2P file-sharing systems.

We now specialize this model to cache clusters. Recall that c is the total number of objects. A natural model for the hit probability is therefore $p(n, x) = x/c$. Thus, for $i \geq 1$, the rate at which the fluid increases between up/down events is given by $\sigma[1-x/c]-\theta x$. An alternative model for $p(n, x)$, which is not studied in the current paper, is to use $p(n, x) = f(x)$, where $f(x)$ is an increasing concave function. Such a "concave" model would capture the fact that the cached objects tend to be the more popular objects.

It remains to define $\Delta_u(i)$ and $\Delta_d(i)$. As discussed in Section 2, for partition hashing it is natural to define $\Delta_d(i) = 1/2$ for $i = 1, \ldots, N$ and $\Delta_u(i) = 1/2$ for $i = 0, \ldots, N-1$. For winning hashing, it is natural to define $\Delta_d(i) = (i-1)/i$ when $i > 0$ and $\Delta_u(i) = i/(i+1)$ for $i < N$.

In the next section we will determine the hit rate for general $\Delta_u(i)$ and $\Delta_d(i)$, and use the result to compare partition hashing with winning hashing.

4 Hit Rate Analysis

In this section we compute the hit rate associated with the fluid model introduced at the end of the previous section. We now define more precisely the fluid model under consideration. Let $X(t) \in [0, c]$ be the total amount of correctly cached fluid at time t and $N(t) \in \{0, 1, \ldots, N\}$ be the number of caches which are up at time t. Let $0 \leq T_1 < T_2 < \cdots$ be the successive jump times of the process $\{N(t), t \geq 0\}$. We will assume that the sample-paths of $\{N(t), t \geq 0\}$ and $\{X(t), t \geq 0\}$ are left continuous. Hence, $N_n := N(T_n+)$

and $X_n := X(T_n+)$ are the number of up caches and the amount of correctly cached fluid, respectively, just *after* the n-th jump. Let $\pi_i = \lim_{n\uparrow\infty} \mathbb{P}[N_n = i]$ be the steady-state probability that there are i active caches just after a jump. We show in appendix A that

$$\pi_0 = \frac{1}{2(1+\rho)^{N-1}} \tag{3}$$

$$\pi_i = \frac{i+\rho(N-i)}{2i(1+\rho)^{N-1}} \binom{N-1}{i-1} \rho^{i-1}, \quad 1 \le i \le N. \tag{4}$$

4.1 Infinite Storage Capacity

Througout this section we assume that each cache can store an unlimited number of objects. (Currently, disk storage capacity is abundant for most caching systems, and capacity misses are very rare as compared to freshness misses. Nevertheless, in the next section we take explicitly into account finite storage capacity.) The fluid arrival process is defined as follows: in (T_n, T_{n+1}) the fluid arrival rate is $\sigma(1 - X(t)/c) - \theta X(t)$ if $N_n \in \{1, 2, \ldots, N\}$ and is equal to 0 if $N_n = 0$. Thus, the rate at which the fluid increases in the cache cluster is

$$\frac{\partial}{\partial t} X(t) = \sigma \left(1 - \frac{X(t)}{c}\right) - \theta X(t) = \sigma - \left(\frac{\sigma}{c} + \theta\right) X(t) \tag{5}$$

for $T_n < t < T_{n+1}$ and $N_n \in \{1, 2, \ldots, N\}$.

The following new parameters will play a role in understanding the system behavior [1]

$$\alpha := \frac{\theta c}{\sigma} \quad \text{and} \quad \gamma := \frac{\sigma}{\mu c} \ . \tag{6}$$

For sake of convenience we also introduce

$$\eta := \frac{c}{1+\alpha}. \tag{7}$$

Integrating (5) gives

$$X(t) = \eta + (X_n - \eta)\, e^{-(t-T_n)\sigma/\eta} \tag{8}$$

for $T_n < t < T_{n+1}$ provided that $N_n \in \{1, 2, \ldots, N\}$. If $N_n = 0$ then $X(t) = 0$ for $T_n < t < T_{n+1}$. At time T_n a jump occurs in the process $\{X(t), t \geq$

[1] The system is defined in terms of 6 parameters: N, c, ρ, μ, θ, σ; definitions in (6) will allow us to express the hit rate only in terms of 4 parameters, namely, N, ρ, α and γ, as shown in Proposition 4.1.

$0\}$ as described in Section 3.2. Note that from (8), $0 \leq X(t) < \eta$ for all $t > 0$ as long as $0 \leq X_0 < \eta$. From now on we will assume without loss of generality that $N_0 = 0$ and $X_0 = 0$ a.s. Under the aforementioned assumptions $\{(N(t), X(t)), t \geq 0\}$ is an irreducible Markov process on the set $\{0, 0\} \cup \{\{1, 2, \ldots, N\} \times [0, \eta)\}$. Denote by X the stationary regime of $X(t)$.

Our objective in this section is to compute the hit rate H, defined as

$$H = \frac{\mathbb{E}[X]}{c}. \tag{9}$$

Proposition 4.1 *Assume that*

$$0 \leq \Delta_u(i)\Delta_d(i+1) \leq 1, \quad \textit{for } i = 0, 1, \ldots, N-1. \tag{10}$$

The hit rate H is given by

$$H = \frac{1}{(1+\alpha)(1+\rho)^N} \sum_{i=1}^{N} \binom{N}{i} \rho^i \, v_i \tag{11}$$

where the vector $\mathbf{v} = (v_1, \ldots, v_N)^T$ is the unique solution of the linear equation

$$A\,\mathbf{v} = \mathbf{b} \tag{12}$$

with $\mathbf{b} = (b_1, \ldots, b_N)^T$ a vector whose components are given by $b_i = \gamma(1+\alpha)$ for i=1,2,...,N, and $A = [a_{i,j}]_{1 \leq i,j \leq N}$ a $N \times N$ tridiagonal matrix whose non-zero elements are

$$\begin{aligned} a_{i,i} &= \gamma(1+\alpha) + i + \rho(N-i), \quad 1 \leq i \leq N \\ a_{i,i-1} &= -i\Delta_u(i-1), \quad 2 \leq i \leq N \\ a_{i,i+1} &= -\rho(N-i)\Delta_d(i+1), \quad 1 \leq i \leq N-1. \end{aligned}$$

Proof. Let Y_n be the amount of correctly cached fluid just before the $(n+1)$-st event (i.e. $Y_n = X_{T_{n+1}-}$). We first compute $\mathbb{E}[Y_n | N_n = i]$ for $1 \leq i \leq N$. With (8) we have

$$\begin{aligned} \mathbb{E}[Y_n \,|\, N_n = i] &= \mathbb{E}\left[\eta + (X_n - \eta)\, e^{-(T_{n+1}-T_n)\sigma/\eta} \,|\, N_n = i\right] \\ &= \eta\, \frac{\gamma(1+\alpha) + (\rho(N-i) + i)\, \eta^{-1}\, \mathbb{E}[X_n \,|\, N_n = i]}{\rho(N-i) + i + \gamma(1+\alpha)}. \end{aligned} \tag{13}$$

To derive (13) we have used the fact that, given $N_n = i$, the random variables X_n and $T_{n+1} - T_n$ are independent, and $T_{n+1} - T_n$ is exponentially distributed with parameter $(N-i)\lambda + \mu i$.

Let us evaluate $\mathbb{E}[X_n \mid N_n = i]$.

Define $v_i := \eta^{-1} \lim_{n\uparrow\infty} \mathbb{E}\left[Y_n \mid N_n = i\right]$. Conditioning on N_{n-1} we have

$$\begin{aligned}
&\lim_{n\uparrow\infty} \mathbb{E}[X_n|N_n = i] = \\
&\quad \lim_{n\uparrow\infty} \mathbb{E}\left[X_n|N_n = i, N_{n-1} = i-1\right] P\left(N_{n-1} = i-1|N_n = i\right) \\
&\quad + \lim_{n\uparrow\infty} \mathbb{E}\left[X_n|N_n = i, N_{n-1} = i+1\right] P\left(N_{n-1} = i+1|N_n = i\right) \mathbb{1}_{[i<N]} \\
&= \Delta_u(i-1) \lim_{n\uparrow\infty} \mathbb{E}[Y_{n-1}|N_{n-1} = i-1] \frac{\pi_{i-1}}{\pi_i} \frac{\rho(N-i+1)}{\rho(N-i+1)+i-1} \\
&\quad + \Delta_d(i+1) \lim_{n\uparrow\infty} \mathbb{E}[Y_{n-1}|N_{n-1} = i+1] \frac{\pi_{i+1}}{\pi_i} \frac{i+1}{\rho(N-i-1)+i+1} \mathbb{1}_{[i<N]} \\
&= \eta \, \frac{\Delta_u(i-1) v_{i-1}\, i + \Delta_d(i+1) v_{i+1}\, \rho(N-i)}{\rho(N-i)+i} \qquad (14)
\end{aligned}$$

by using (4) and the definition of v_i. Finally, introducing (14) into (13) yields

$$(\rho(N-i)+i+\gamma(1+\alpha))\, v_i = \gamma(1+\alpha)+i\Delta_u(i-1)v_{i-1}+\rho(N-i)\Delta_d(i+1)v_{i+1} \quad (15)$$

for $i = 1, 2, \ldots, N$, or equivalently (12) in matrix form with $\mathbf{v} = (v_1, \ldots, v_N)$. The uniqueness of the solution of (12) is shown in appendix B.

The vector $\mathbf{v}$ in (12) gives the conditional stationary expected amount of fluid correctly cached just before jump epochs (up to a multiplicative constant). However, the hit rate H in (9) is defined in terms of the stationary expected amount of fluid correctly cached at *arbitrary* epochs. The latter metric can be deduced from the former one by using Palm calculus, through the identity (see e.g. [16, Formula (4.3.2)])

$$\mathbb{E}\left[X\right] = \Lambda \mathbb{E}^0 \left[\int_0^{T_1} X(t)dt\right] \quad (16)$$

where $\mathbb{E}^0$ denotes the expectation with respect to the Palm distribution [2], T_1 denotes the time of the first jump after 0, and where Λ denotes the global rate of the Engset model, i.e.

$$\Lambda = \frac{1}{\mathbb{E}^0[T_1]}. \quad (17)$$

From now on we assume that the system is in steady-state at time 0. Under the Palm distribution we denote by N_{-1} and Y_{-1} the number of up caches and the amount of correctly cached fluid respectively, just before time 0 (i.e. just before the jump to occur at time 0).

[2] i.e. assuming that a jump occurs at time 0 and that the system is in steady-state at time 0

We first compute $1/\Lambda$. We have

$$\frac{1}{\Lambda} = \sum_{i=0}^{N} \pi_i \, \mathbb{E}^0[T_1 \,|\, N_0 = i] = \frac{1}{\mu} \sum_{i=0}^{N} \frac{\pi_i}{\rho(N-i)+i} = \frac{1+\rho}{2N\rho\mu} \tag{18}$$

by using (3)-(4).

Let us now determine $\mathbb{E}\,[X]$. From (16), (8), (18) we find

$$\begin{aligned}
\mathbb{E}\,[X] &= \Lambda \sum_{i=1}^{N} \pi_i \, \mathbb{E}^0 \left[\int_0^{T_1} \left(\eta + (X_0 - \eta)\, e^{-t\sigma/\eta} \right) dt \,|\, N_n = i \right] \\
&= \Lambda\eta \left[\sum_{i=1}^{N} \pi_i \mathbb{E}^0[T_1 | N_0 = i] + \frac{1}{\sigma} \sum_{i=1}^{N} \pi_i \mathbb{E}^0 \left[(X_0 - \eta) \left(1 - e^{-T_1\sigma/\eta} \right) | N_0 = i \right] \right] \\
&= \Lambda\eta \Bigg[\mathbb{E}^0[T_1] - \pi_0 \mathbb{E}^0[T_0 \,|\, N_0 = 0] + \frac{1}{\sigma} \sum_{i=1}^{N} \pi_i \left(\mathbb{E}^0[X_0 \,|\, N_0 = i] - \eta \right) \\
&\quad \times \left(1 - \mathbb{E}^0 \left[e^{-T_1\sigma/\eta} \,|\, N_0 = i \right] \right) \Bigg] \qquad (19) \\
&= \Lambda\eta \left[\frac{1}{\Lambda} - \frac{1}{2N\rho\mu\,(1+\rho)^{N-1}} + \frac{1}{\mu} \sum_{i=1}^{N} \pi_i \, \frac{\eta^{-1}\mathbb{E}^0[X_0 \,|\, N_0 = i] - 1}{\rho(N-i) + i + \gamma(1+\alpha)} \right] \\
&= \frac{c}{1+\alpha} \left[1 - \frac{1}{(1+\rho)^N} + \frac{2N\rho}{(1+\rho)} \sum_{i=1}^{N} \pi_i \, \frac{\eta^{-1}\mathbb{E}^0[X_0 \,|\, N_0 = i] - 1}{\rho(N-i) + i + \gamma(1+\alpha)} \right] \qquad (20)
\end{aligned}$$

By definition, $\mathbb{E}^0[X_0 \,|\, N_0 = i] = \lim_{n\uparrow\infty} \mathbb{E}[X_n \,|\, N_n = i]$, which has been computed in (14). By combining (14) and (15) we obtain

$$\mathbb{E}^0[X_0 \,|\, N_0 = i] = \eta \, \frac{(\rho(N-i) + i + \gamma(1+\alpha))v_i - \gamma(1+\alpha)}{\rho(N-i)+i}.$$

Plugging this value of $\mathbb{E}^0[X_0 \,|\, N_0 = i]$ into the r.h.s. of (20), and using (4), yields after some straightforward algebra

$$\begin{aligned}
\mathbb{E}\,[X] &= \frac{c}{1+\alpha} \left[1 - \frac{1}{(1+\rho)^N} + \frac{N\rho}{(1+\rho)^N} \sum_{i=1}^{N} \binom{N-1}{i-1} \frac{\rho^{i-1}}{i} (v_i - 1) \right] \\
&= \frac{c}{(1+\alpha)(1+\rho)^N} \sum_{i=1}^{N} \binom{N}{i} \rho^i v_i. \qquad (21)
\end{aligned}$$

According to (9) it remains to divide both sides of (21) by c to get (11). This concludes the proof. ◇

Conditions (10) in Proposition 4.1 ensure that the system (12) has a unique solution (see Section A). They are satisfied for both winning hashing (since $\Delta_u(i)\Delta_d(i+1) = (i/(i+1))^2$ for $i < N$) and for partition hashing (since $\Delta_u(i)\Delta_d(i+1) = 1/4$ for all $i < N$) schemes – see discussion at the end of Section 3.

Remark 4.1 *Since A is a tridiagonal matrix, (12) can be solved in only $\mathcal{O}(N)$ operations, once the mappings Δ_u and Δ_d are specified.*

4.2 Finite Storage Capacity

We assume in this section that the amount of fluid in each cache cannot exceed a given constant B. More specifically, we assume that $X(t) \leq BN(t)$ for all $t \geq 0$. In this setting the time-evolution of $X(t)$ between two consecutive jump times of the process $\{N(t),\, t \geq 0\}$ is given by

$$X(t) = \min\left(BN_n,\, \eta + (X_n - \eta)\, e^{-(t-T_n)\sigma/\eta}\right) \tag{22}$$

for $T_n < t < T_{n+1}$. When $B = \infty$ then we retrieve (8).

Unfortunately, when B is finite the computation of $\mathbb{E}\left[Y_n | N_n = i\right]$ introduces a non-linearity due to the minimum operator in (22). Therefore, unlike in the case when $B = \infty$, it is not possible to find a closed-form expression for the hit rate H. An alternative approach to compute H is to use an hybrid equation-based/discrete-event simulator that uses (22). This can be done as follows: First, run a discrete-event driven simulation of the process $\{(N_n, T_n),\, n \geq 1\}$. Then, use $\{(N_n, T_n),\, n \geq 1\}$ in (22) to evaluate $\mathbb{E}[X]$. We expect this solution to be much more time-efficient than a classical discrete-event driven simulation of the entire system since the hybrid approach will only have to simulate events (up/down events) on a slow time-scale. This method is discussed in Section 6.2.

5 Performance of the Caching System

In this section we use Proposition 4.1 to analyze cache clusters. First, we show how the result provides qualitative insight for the hit rate. We then use the result to compare the hit rates of partition hashing and winning hashing.

5.1 Qualitative Behavior

For small N, we can compute the hit rate in closed-form. We do this now for winning hashing.

For $N = 2$ we have

$$H = 2\gamma \frac{\rho}{(1+\rho)^2} \frac{2\gamma\alpha + \rho\gamma\alpha + 2\gamma + \rho\gamma + \rho^2 + 4 + 3\rho}{2\gamma^2 + 4\gamma^2\alpha + 6\gamma + 2\gamma^2\alpha^2 + 6\gamma\alpha + 4 + 2\rho\gamma + 2\rho\gamma\alpha + 3\rho}.$$

We observe that the hit rate only depends on the parameters α, γ, defined in Section 4 (see (6)), and ρ. This result actually holds for any value of N since a glance at Proposition 4.1 indicates that the components of A and $\mathbf{b}$ depend on the model parameters only through α, ρ and γ. Interestingly enough, the fact that the hit rate for a given rate of change depends on the parameters σ and c only through the ratio σ/c was observed in [10] in a slightly different context. This is an indication that our fluid model is able to capture some of the main features of the caching system.

Figure 1 shows how the hit rate depends on γ, ρ and α for small clusters (i.e. when N is small). The concave shapes on Figure 1(a) shows that increasing γ (through σ, for instance) can offer a large performance gain in the smaller range, in this case when $\gamma \leq 20$. This can be related to empirical observations in [17,18,10]: for small client population sizes, the authors found that the hit rate increases in a log-like[3] fashion of the population size. Moreover, Duska, Marwood and Feeley showed that request rate is a better metric of performance than client population size, and that the hit rate also increases with request rate [19]. Our model exhibits similar shapes, although the hit rate is a rational function of γ rather than a logarithmic function of γ. Moreover, it explains analytically these properties, and also includes the caches dynamic behavior through μ in the γ definition.

Parameter ρ is fairly new in cache cluster analysis. It typically represents one aspect of the dynamic behavior of the system, as apparent in (1) and (2).

Figure 1(b) represents the hit rate as an increasing function of ρ, converging rapidly to its maximum value ($\approx 40\%$ in Figure1b). The curves flatten to become almost constant for high values of ρ; the larger N the quicker the curve flattens. Therefore, except for very small values, in which case the hit rate drops very quickly, ρ has very little influence on the hit rate. This can be easily explained. Indeed, we see from (1) that with the probability $1/(1+\rho)^N$ all caches are down. Therefore, all caches are down with very high probability when when ρ is small, yielding very low hit rates, as shown in Figure 1(b).

[3] The hit rate is either a logarithm or a small power of the population size.

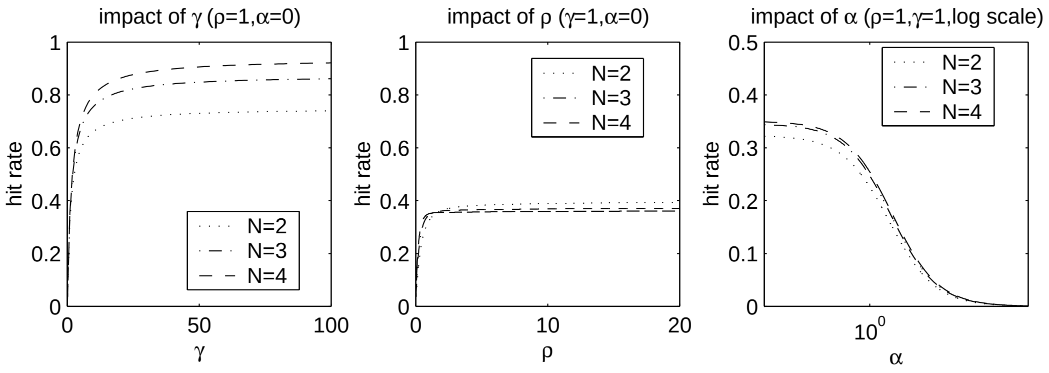

Fig. 1. Impact of ρ, γ and α on the hit rate for small clusters.

On the other hand, when ρ is large there is always at least one cache up with a high probability which prevents the hit rate from dropping to zero. Under these circumstances, the limiting factor for the hit rate will be the removal of documents in caches, modeled by parameters γ (as described above) and α.

Figure 1(c) shows how α impacts the hit rate for $\rho = 1$ (which is large enough to avoid long periods of total unavailability since in this case 50% of the caches are up on the average as shown in (2)) and $\gamma = 1$. The curve is obviously decreasing since α is proportional to the rate of change (or expiration) of cached documents. The highest hit rate is therefore obtained with $\alpha = 0$. Also observe that the hit rate drops significantly as α increases.

From Figure 1 we infer that the key parameters of the system are γ and α, which almost determine the hit rate as long as ρ is not too close to zero. This can be explained by the fact that for high values of ρ, γ and α capture the main interactions between object population, request rate, document rate of change and cache dynamics — which correspond to document losses and misplacements in the cluster.

Since γ and α are the two only limiting factors for realistic systems (where $P(N_\infty = 0) \approx 0$), we may want to compare their influence on the system. On Figure 2 we observe that the domain where the hit rate is high (above 40%) is very small ($\alpha \leq 1$, $\gamma \geq 10$). In fact, γ has a real impact on the hit rate when $\alpha \leq 1$. The concave shape observed in Figure 1(a) for $\alpha = 0$ is still present for positive values of α but it is less and less pronounced as α increases. This can be explained analytically from the fact that $X_t \leq \eta$ for all $t > 0$ (provided that $X_0 < \eta$) as already observed in Section 4.1, which implies that H is bounded from above by $1/(1 + \alpha)$.

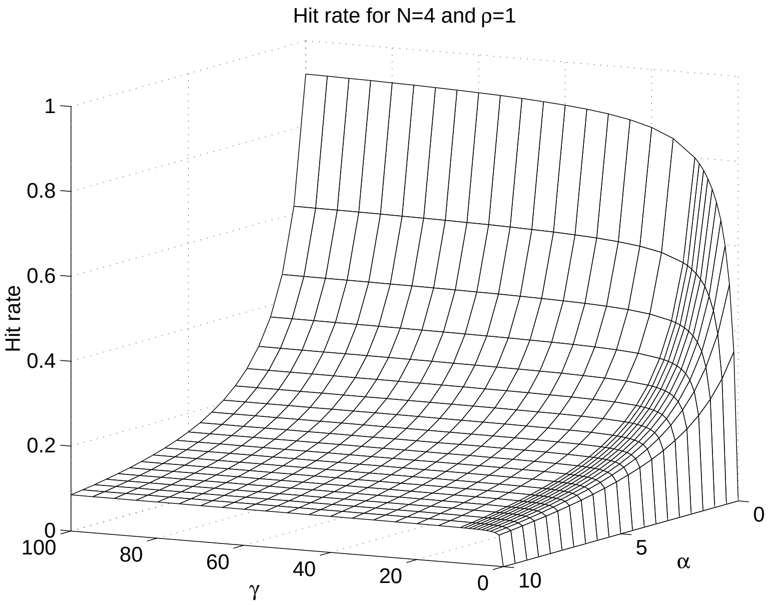

Fig. 2. H as a function of γ and α for $\rho = 1$

5.2 Comparison of Partition Hashing and Winning Hashing

Figure 3 compares the hit rate for partition hashing with that of winning hashing when $\alpha = 0$, i.e. when documents do not expire. The performance difference is striking, especially for small γ and $\rho > 1$: for any set of parameters, winning hashing always exhibits much higher hit rates than does partition hashing scheme. For instance, at $\rho = 50$ and $\gamma = 1$ (see Figure 3(b)), the hit rate for winning hashing is 36%, which is 50% higher than that for partition hashing, i.e., 24%.

6 Numerical Results

In this section, we compare quantitatively our macroscopic fluid model with a discrete-event driven simulation of the microscopic model described in Section 3.1 for $N = 10$ caches. Throughout this section we use winning hashing: the CARP hashing function [12] is implemented in the simulator while the corresponding values of Δ_u and Δ_d are used in the fluid model. Experimental results are given with 99% confidence intervals.

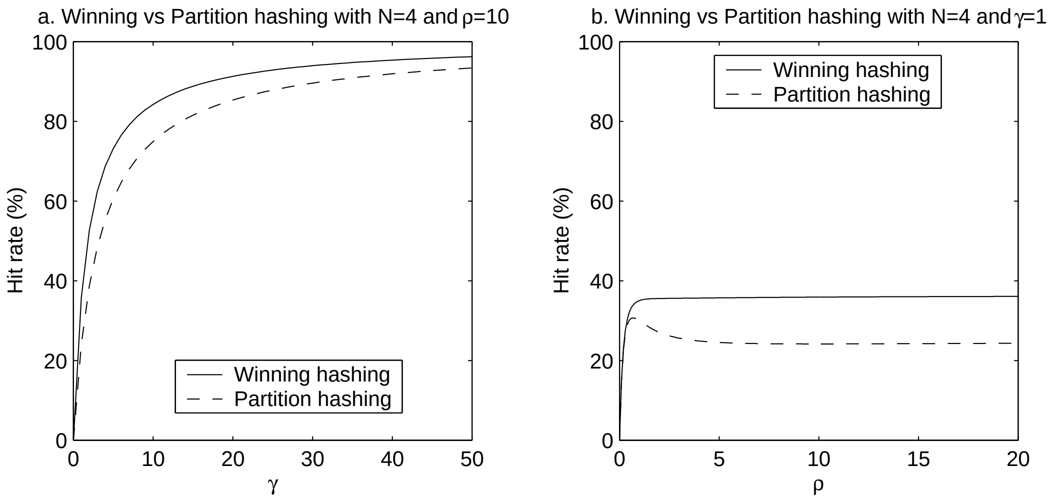

Fig. 3. Comparison of winning hashing and partition hashing for $N = 4$ and $\alpha = 0$

6.1 Unlimited Capacity

In this section, the fluid model estimation is computed using Proposition 4.1. We begin with a validation of parameter γ. Table 1 shows results for various values of the (σ, μ, c) triple with a constant ratio $\gamma = 2$, and $\rho = 1$. Of course, the fluid model provides identical values, i.e., 50.9%, for all experiments (see Section 5). We observe that the discrete-event simulation is almost insensitive to the variations of the number of documents, request arrival rate or failure rate when γ is constant: even when they vary by several orders of magnitude, the hit rate remains between 50% and 52%, which is close to the fluid model value. We conclude that our analytical evidence that the system is characterized by parameters ρ and γ is confirmed here by simulation results. Of course, when σ becomes of the same order of magnitude as μ, which is highly unlikely to happen in real systems, discrete-event simulation does not see enough requests to create reliable statistics.

Table 1
Hit rate (%) for $\gamma = 2$ and $\rho = 1$.

c	2000			20,000		
σ	0.2	2	20	0.2	2	20
μ	5×10^{-5}	5×10^{-4}	5×10^{-3}	5×10^{-6}	5×10^{-5}	5×10^{-4}
Simulation	50.0±1.5	50.8±1.9	51.8±1.7	51.0±1.2	50.3±2.4	50.4±1.9
Fluid Model	50.9	50.9	50.9	50.9	50.9	50.9

We now consider the impact of γ on the hit rate. Figure 4 displays the hit rate as a function of γ with $\rho = 1$, for two different values of α. We observe that the fluid model curves closely follow the same shapes as the discrete-event simulations and therefore mimics the discrete system behaviour very accurately. An important feature appearing on Figure 4 is the range of H when $\alpha = 0$ and ρ not too small: H increases with γ from zero to almost 100%. Although this observation is not true for very small values of N, as shown in Figure 1, the upper bound seems to increase with N and is already very close to 100% for $N = 10$. Therefore, for small values of α and $\rho \geq 1$, it is possible to reach almost any desired hit rate by increasing γ accordingly. This validates our finding that γ determines the hit rate of the system when $\alpha = 0$. Also, the curves comparing our fluid model to discrete-event simulation when $\alpha = 1$ clearly shows how this second parameter limits the hit rate even for large values of γ, which is rather intuitive: indeed, α represents the time needed for the system to cache all existing documents (filling time) divided by the time-to-live of the cached documents, while γ is the ratio of the average lifetime of a cache and this filling time. It is clear that if the document modification rate is high with regard to the filling time, fewer documents will become misplaced upon failure events.

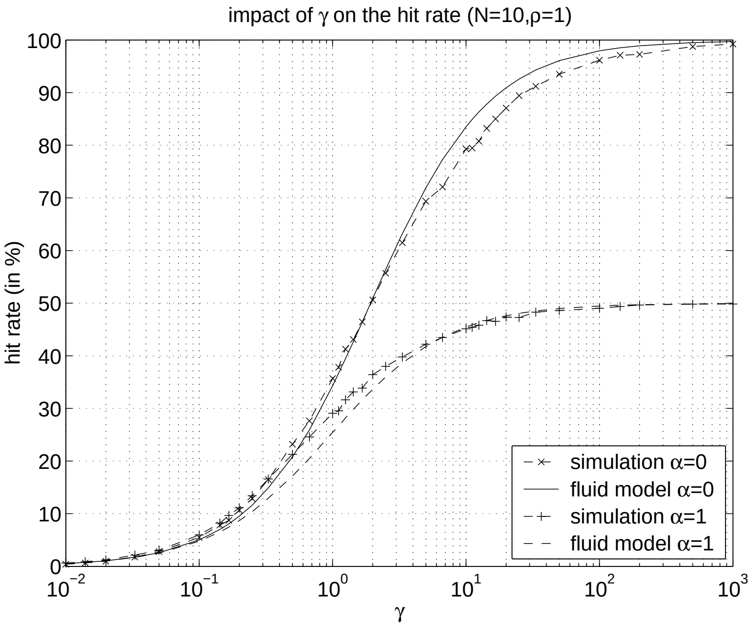

Fig. 4. Fluid model vs simulation: impact of γ (with $N = 10$ and $\rho = 1$).

Finally, we examine the influence of ρ on the hit rate. Figure 5 shows that both the fluid model and the simulation exhibit a steep slope for small values of ρ and an almost flat shape for $\rho \geq 1$. This validates the fact that ρ has very little influence on the hit rate except when it is close to zero.

We conclude that the fluid model provides an accurate approximation for the actual hit rate of the discrete system and more importantly, highlights the key parameters and properties of the system.

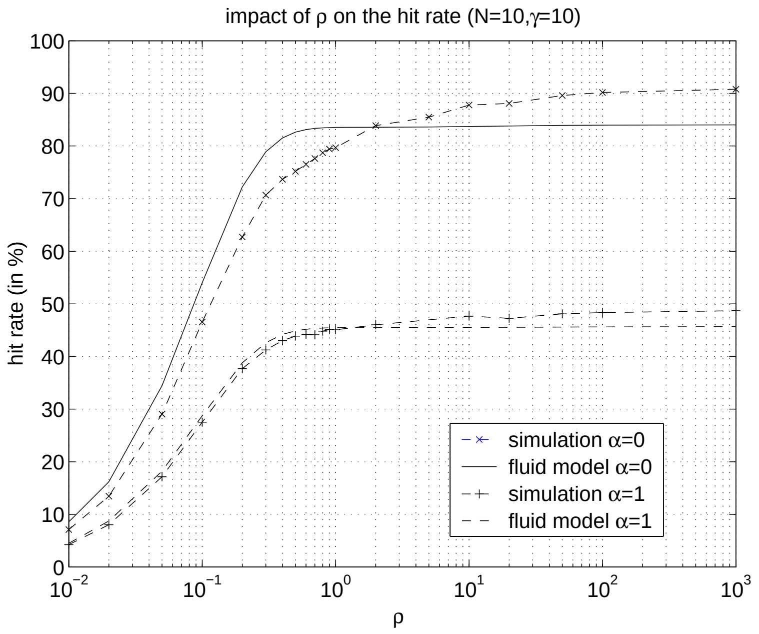

Fig. 5. Fluid model vs simulation: impact of ρ (with $N = 10$ and $\gamma = 10$).

6.2 Validation for Finite Storage

Figure 6 compares the results obtained with the equation-based simulator that uses (22) with that of a discrete-event driven simulator as a function of the average storage capacity $N_{\text{moy}}B$ for $\gamma = \rho = 1$ and $\alpha = 0$, where $N_{\text{moy}} = N\rho/(1+\rho)$ is the mean number of active caches (see (2)).

The flat shape after a threshold around $1.5c$ when $\alpha = 0$, indicates that increasing capacity beyond a certain value does not improve the cache performance, limited by other factors such as cache dynamics. This phenomenon is even more obvious when $\alpha > 0$, because object expirations happen faster than cache filling, and justifies the infinite capacity assumption in Section 4.1.

Figure 6 shows that the equation-based simulator results not only exhibits the same shape as the discrete system hit rate, but also provides an accurate numerical approximation. This strengthens the conclusion that the fluid model is able to capture the main features of the discrete system.

Moreover, the equation-based simulator is a much faster tool than the discrete-event driven simulation of the system, especially for large values of request rates. In addition to an obvious efficiency gain, it provides higher accuracy by allowing the simulation of a much larger number of up/down events, thereby approaching more closely the stationary state. Also, the equation-based simulation method can easily be extended to other equations than (22), for instance to take into account document popularity as discussed in Section 3.2.

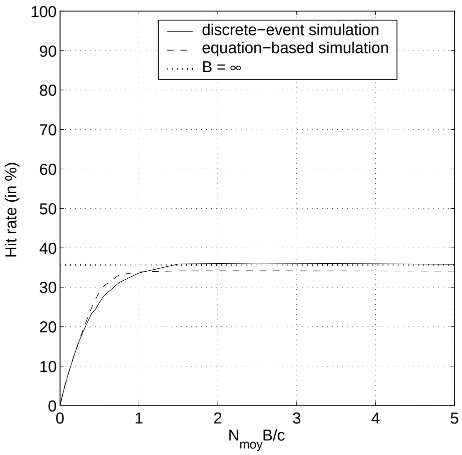

Fig. 6. Impact of B on the hit rate.

7 Concluding Remarks

In this paper we have considered a complex system consisting of multiple nodes that randomly go up and down and which store new objects arriving randomly from origin servers. The system exhibits randomness on two time scales: object arrivals on a fast time scale, and cache up/down events on a slower time scale. To analyze this complex system, we have approximated the system with a stochastic fluid model, which although non-trivial, turns out to be mathematically tractable. Comparison with simulation results has shown that the hit rate provided by the solution to the model is a good approximation of the actual hit rate. Also, the solution highlights the key characteristics of the actual system.

Our stochastic fluid model in Section 3 is quite general and can be used to model a variety of content distribution systems. Many open problems remain. For example, it is of interest to extend the theory to cover object-popularity (e.g., Zipf distribution). It is also of interest to extend the theory to model content dynamics in P2P file sharing and in CDNs.

A Stationary distribution of the Engset model at jump times

In this section we compute the limiting distribution of the Markov chain $\{N_n, n \geq 1\}$. Let $P = [p_{i,j}]_{0 \leq i,j \leq N}$ be its transition probability matrix. We have $p_{i,i+1} = \rho(N-i)/(\rho(N-i)+i)$ for $i = 0, 1, \ldots, N-1$, $p_{i,i-1} = i/(\rho(N-i)+i)$ for $i = 1, 2, \ldots, N$ and $p_{i,j} = 0$ otherwise.

Since this Markov chain[4] has a finite-state space and is irreducible, it is positive recurrent [20, Cor. 5.3.19, 5.3.22]. Therefore, it possesses a unique stationary distribution $\pi = (\pi_0, \cdots, \pi_N)$ given by the (unique) solution of the equation $\pi P = \pi$ such that $\sum_{i=0}^{N} \pi_i = 1$ [21, page 208].

We proceed by induction to compute π. From the equation $\pi P = \pi$ we find that $\pi_1 = (\rho(N-1)+1)\pi_0$ and $\pi_2 = \frac{\rho(N-2)+2}{2}\rho(N-1)\pi_0$. This suggests that

$$\pi_j = \frac{\rho(N-j)+j}{j}\frac{\rho^{j-1}}{(j-1)!}\frac{(N-1)!}{(N-j)!}\pi_0 \tag{A.1}$$

for $j = 1, 2, \ldots, N$. Assume that (A.1) holds for $j = 1, 2, \ldots, i < N-1$. Let us show that it still holds for $j = i+1$. We have

$$\begin{aligned}
\pi_{i+1} &= \frac{\rho(N-(i+1))+i+1}{i+1}\left(\pi_i - \frac{\rho(N-(i-1))}{\rho(N-(i-1))+i-1}\pi_{i-1}\right) \\
&= \frac{\rho(N-(i+1))+i+1}{i+1}\left(\frac{(\rho(N-i)+i)\,\rho^{i-1}}{i!(N-i)!}\right. \\
&\quad \left. - \frac{\rho(N-(i-1))}{\rho(N-(i-1))+i-1}\frac{(i-1+\rho(N-i+1))\,\rho^{i-2}}{(i-1)(i-2)!(N-i+1)!}\right)(N-1)!\pi_0 \qquad \text{(A.2)} \\
&= \frac{\rho(N-(i+1))+i+1}{i+1}\frac{\rho^i(N-1)!}{i!(N-(i+1))!}\pi_0
\end{aligned}$$

where (A.2) follows from the induction hypothesis. The constant π_0 is computed by using the normalizing condition $\sum_{i=0}^{N} \pi_i = 1$; we find $\pi_0 = 1/(2(1+\rho)^{N-1})$ as announced in (3). Plugging this value of π_0 into (A.1) gives (4). $\diamond$

B Uniqueness of the solution of (12)

The linear system (12) defined in Proposition 4.1 admits a unique solution if and only if $\det(A) \neq 0$. Since A is a tridiagonal matrix we can use the LU decomposition [22, Sec. 3.5] $A = LU$ with

$$L = \begin{pmatrix} l_1 & 0 & \cdots & 0 \\ \beta_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \beta_n & l_n \end{pmatrix} \qquad U = \begin{pmatrix} 1 & u_1 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & u_{n-1} \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

[4] Note that this chain is period (with period 2).

where l_i's and u_i's are defined as follows:

$$\begin{aligned} a_{1,1} &= l_1 \\ a_{i,i} &= l_i + a_{i,i-1}u_{i-1}, \; i = 2, \ldots, N \\ l_i u_i &= a_{i,i+1}, \qquad\qquad i = 1, \ldots, N-1. \end{aligned}$$

Both matrices L and U being bidiagonal matrices it follows that $\det(A) \neq 0$ if and only if $l_i \neq 0$ for $i = 1, 2, \ldots, N$.

We use an induction argument to show that $l_i \neq 0$ for $i = 1, 2, \ldots, N$. We have $l_1 = \gamma + \rho(N-1) + 1$. Assume that $l_i > \gamma + \rho(N-i)$ for $i = 1, 2, \ldots, n < N-1$ and let us show that $l_{n+1} > \gamma + \rho(N-n-1)$. We have

$$\begin{aligned} l_{n+1} &= a_{n+1,n+1} - \frac{a_{n+1,n}a_{n,n+1}}{l_n} \\ &= \gamma + \rho(N-n-1) + (n+1)\frac{l_n - \rho(N-n)\Delta_u(n)\Delta_d(n+1)}{l_n} \\ &> \gamma + \rho(N-n-1) \end{aligned}$$

by using the induction hypothesis together with the fact that $0 \leq \Delta_u(n)\Delta_d(n+1) \leq 1$.

◇

References

[1] D. Anick, D. Mitra, M. M. Sondhi, Stochastic theory of data-handling systems with multiple sources, Bell Systems Technical Journal 61 (1982) 1871–1894.

[2] A. Elwalid, D. Mitra, Fluid models for the analysis and design of statistical multiplexing with loss priorities on multiple classes of bursty traffic, in: Proc. IEEE INFOCOM '92, Florence, Italy, 1992, pp. 415–425.

[3] A. Elwalid, D. Mitra, Effective bandwidth of general markovian traffic sources and admission control of high speed networks, IEEE/ACM Transactions on Networking 1 (1993) 329–343.

[4] K. Kumaran, M. Mandjes, Multiplexing regulated traffic streams: design and performance, in: Proc. IEEE INFOCOM 2001, Anchorage, Alaska, 2001, pp. 527–536.

[5] F. LoPresti, Z. Zhang, D. Towsley, J. Kurose, Source time scale and optimal buffer/bandwidth trade-off for regulated traffic in an ATM node, in: Proc. IEEE INFOCOM '97, Kobe, Japan, 1997, pp. 676–683.

[6] R. Boorstyn, A. Burchard, J. Liebeherr, C. Oottamakorn, Statistical service assurances for traffic scheduling algorithms, IEEE Journal on Selected Areas in Communications, Special Issue on Internet QoS 18 (2000) 2651–2664.

[7] M. Reisslein, K. W. Ross, S. Rajagopal, A framework for guaranteeing statistical QoS, IEEE/ACM Transactions on Networking 10 (1) (2002) 27–42.

[8] S. Iyer, A. Rowstron, P. Druschel, Squirrel: A decentralized, peer-to-peer Web cache, in: Proc. of ACM Symposium on Principles of Distributed Computing (PODC 2002), Monterey, California, 2002, pp. 213–222.

[9] P. Rodriguez, C. Spanner, E. Biersack, Analysis of web caching architectures: Hierarchical and distributed caching, IEEE?ACM Transactions on Networking 9 (4) (2001) 404–418.

[10] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, H. Levy, On the scale and performance of cooperative Web proxy caching, in: 17th ACM Symp. on Operating Systems Principles (SOSP '99), Kiawah Island, South Carolina, 1999, pp. 16–31.

[11] K. W. Ross, Hash-routing for collections of shared Web caches, IEEE Network Magazine 11 (1997) 37–45.

[12] V. Valloppillil, K. W. Ross, Cache array routing protocol (CARP), Internet Draft (June 1997).

[13] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, Y. Yerushalmi, Web caching with consistent hashing, in: 8th International WWW Conference, Toronto, 1999.

[14] F. P. Kelly, Reversibility and Stochastic Networks, Wiley, Chichester, 1979.

[15] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: Evidence and implications, in: Proc. IEEE INFOCOM '99, New York, 1999, pp. 126–134.

[16] F. Baccelli, P. Brémaud, Elements of Queueing Theory: Palm-Martingale Calculus and Stochastic Recurrences, Springer Verlag, 1994.

[17] P. Cao, S. Irani, Cost-aware WWW proxy caching algorithms, in: Proc. USENIX Symp. on Internet Technologies and Systems, Monterey, California, 1997, pp. 193–206.

[18] S. Gribble, E. Brewer, System design issues for internet middleware services: Deductions from a large client trace, in: Proc. USENIX Symp. on Internet Technologies and Systems, Monterey, California, 1997, pp. 207–218.

[19] B. Duska, D. Marwood, M. Feeley, The measured access characteristics of World Wide Web client proxy caches, in: Proc. USENIX Symp. on Internet Technologies and Systems, Monterey, California, 1997, pp. 23–35.

[20] E.Çinlar, Introduction to Stochastic Processes, Prentice Hall, 1975.

[21] G. Grimmett, D. Stirzaker, Probability and Random Processes, Oxford University Press, 1992.

[22] R. A. Horn, C. R. Johnson, Matrix Analysis, Cambridge University Press, 1985.