A Simple Fluid Model for the Analysis of the Squirrel Peer-to-Peer Caching System

Florence Clévenot, Philippe Nain
INRIA
BP 93
06902 Sophia Antipolis, France
{clevenot,nain}@sophia.inria.fr

Abstract— **Peer-to-peer (P2P) systems are complex to analyze due to their large number of users who connect intermittently and to the frequency of requests for files or Web objects.**

In this paper we propose a mathematical model in which request streams are represented as fluid flows and then apply this model in an analysis of Squirrel: a recent P2P cooperative Web cache.

Our fluid model provides a low-complexity means to estimate the performance of Squirrel (hit probability and latency) and exhibits the key qualitative properties of this system. The accuracy of our model is validated by a comparison with discrete-event simulation.

I. Introduction

Peer-to-peer systems exhibit continuously increasing complexity in several dimensions, such as number of users, number of available documents, and access speed, etc. Performance analysis of peer-to-peer systems suffers from this complexity and often requires costly numerical methods or model simulations [1], [2].

Inspired by the seminal work of Anick, Mitra and Sondhi in 1982 [3] and the subsequent success of fluid modeling and simulation of packet networks (see for instance [4]–[10] and references therein), in this paper we explore a fluid approach for modeling content distribution systems where content (e.g. shared files) is replaced by a fluid.

We have successfully used this approach in [11] to study the performance (hit rate) of a cluster of Web caches. In this paper we use a stochastic fluid model to investigate the performance of Squirrel [12]: a novel peer-to-peer (P2P) cooperative Web cache. The principle of Squirrel is to replace a corporate dedicated Web cache by making client desktop machines cooperate in a peer-to-peer fashion in order to act globally as an efficient distributed Web cache.

This analysis differs from that in [11] in that within this new system, clients and caches are the same entities, and consequently the request rate now depends on the number of active users.

Alternative approaches include Markovian analysis and event-driven (or trace-driven) simulations. These approaches have merit and we do not want to systematically oppose fluid models to more traditional approaches. Our take-home message is that simple (macroscopic) fluid models, whenever they apply, may give fairly accurate qualitative and quantitative results and at a low numerical complexity. Content distribution networks appear to be good candidates to illustrate our approach as they typically involve a large number of users and many parameters. In turn, these characteristics imply large state spaces and a high numerical complexity when one uses detailed (microscopic) models (such as the Markovian) and simulations.

In Section II we provide an overview of Squirrel. Our fluid model is introduced in Section III and we use it in Section IV to compute the main performance of Squirrel (hit probability, latency, etc.). In particular, we provide a simple expression for the hit probability, whose complexity is linear in the number of nodes in the Squirrel network. We show in Section V that our model provides substantial insight into performance issues of P2P cooperative Web caches such as Squirrel. Our analysis shows that two key parameters largely determine the performance of the system. In Section VI we compare results obtained with the fluid model to results obtained with a discrete-event simulation of Squirrel. We find that the fluid model is both qualitatively and quantitatively accurate. We conclude in Section VII with possible extensions of our fluid model.

II. Overview of Squirrel

Squirrel [12] is a decentralized, peer-to-peer Web cache that uses Pastry [13] as a location and routing protocol. When a client requests an object it first sends a request to the Squirrel proxy running on the client's machine. If the object is uncacheable [1] then the proxy forwards the request directly to the origin Web server. Otherwise it checks the local cache, like every Web browser would do, in order to exploit locality and reuse. If a fresh copy of the object is not found in this cache, then Squirrel tries to locate one on some other node. To do so, it uses the distributed hash-table and the routing functionalities provided by Pastry. First, the URL of the object is hashed to give a 128-bit object identity (a number called *object-Id*) from a circular list; then the routing procedure of Pastry forwards the request to the node with the identity (called *node-Id*; this number is assigned randomly by Pastry to a participating node)

[1] An object can be considered uncacheable if, for example, its URL contains "cgi-bin", or if its freshness lifetime is zero (see e.g. [14] for details).

the closest to *object-Id*. This node then becomes the *home node* for this object. Squirrel then proposes two schemes from this point on: *home-store* and *directory* schemes.

In the home-store scheme, objects are stored both at client caches and at its home node. The client cache may either have no copy of the requested object or a stale[2] copy. In the former case the client issues a GET request to its home-node, and it issues a *conditional* GET (cGET) request in the latter case. If the home-node has a fresh copy of an object then it forwards it to the client or it sends the client a not-modified message depending on which action is appropriate. If the home-node has no copy of the object or has a stale copy in its cache, then it issues a GET or a cGET request, respectively, to the origin server. The origin server then either forwards a cacheable copy of the object or sends a not-modified message to the home-node. Then, the home-node takes the appropriate action with respect to the client (i.e. send a not-modified message or a copy of the object).

In the directory scheme the home-node for an object maintains a small directory of pointers to nodes that have recently accessed the object. A request for this object is sent randomly to one of these nodes. We will not go deeper into the description of this scheme since from now on we will only focus on the home-store scheme. We do so mainly because the latter scheme has been shown to be overall more attractive than the directory scheme [12]. In addition, the home-store scheme is more amenable to a fluid analysis than the directory scheme.

In a Squirrel network (a corporate network, a university network, etc.), like in any peer-to-peer system, clients arrive and depart the system at random times. There are two kinds of failures (or departures): abrupt and announced failures. Each failure has a different impact on the performance of Squirrel. An abrupt failure will result in a loss of objects. To see this, assume that node i is the home-node for object O. If node i fails, then a new home-node for object O has to be found by Pastry, as explained above, the next time object O is requested. Assume that the copy of object O was fresh when node i failed and consider the first GET request issued for O after the failure of node i. The GET request is therefore forwarded to the new home-node for object O (say node j); this request will result in a miss if j has no copy of O or if its copy is stale. In this case, the failure of node i will yield a degradation in the performance since node j will have to contact the origin server to get a new copy of object O or a not-modified message, as appropriate. If a node is able to announce its departure and to transfer its content to its immediate neighbors in the node-Id space before leaving Squirrel (announced failure), then no content is lost when the node leaves.

When a node joins Squirrel then it automatically becomes the home node for some objects but does not store those objects yet (see details in [12]). In case a request for one of those objects is issued, then its two neighbors in the node-Id space transfer a copy of the object, if any. Therefore, we can consider that there is no performance degradation in Squirrel due to a node arrival, since the transfer time between two nodes is supposed to be at least one order of magnitude smaller than the transfer time between any given node and the origin server.

[2]A node determines the freshness of a copy either through an explicit value provided by HTTP fields (MAX_AGE, EXPIRES) or by using a heuristic, as detailed later.

From now on, the terms “node” and “client” will be used interchangeably.

III. A Model for Squirrel

A. Modeling the client dynamics

To capture the dynamic behavior of the Squirrel nodes we use the following model: we assume that there are $N < \infty$ clients who join and leave Squirrel independently of each other. The time until a given client joins (resp. leaves) is exponentially distributed with rate $\lambda > 0$ (resp. $\mu > 0$). If we denote by $N(t) \in \{0, 1, 2 \ldots, N\}$ the number of participating (i.e. connected) clients at time $t \geq 0$, then $\{N(t)\}_t$ is a birth and death process, known in the literature as the *Ehrenfest* (or *Engset*) model [15].

Let $\mathbb{N}^\infty$ denote the stationary number of participating clients. Setting $\rho = \lambda/\mu$, we have [15, p. 17]

$$\mathbb{P}[N^\infty = i] = \binom{N}{i} \frac{\rho^i}{(1+\rho)^N}, \quad 0 \leq i \leq N. \tag{1}$$

From now on we will consider that the process $\{N(t)\}_t$ is in steady-state at time $t = 0$. Let $0 \leq T_1 < T_2 < \cdots$ denote the successive *jump times* of the (stationary) process $\{N(t)\}_t$. Introduce $N_n \overset{\text{def}}{=} N(T_n+)$, the stationary number of participating clients just *after* the occurrence of the n-th event/jump (i.e. join or leave of a client). Let π_i be the stationary probability that there are i participating nodes just after a jump. It is shown in Appendix III that

$$\begin{aligned} \pi_i &= \binom{N-1}{i-1} \frac{i + \rho(N-i)}{2i(1+\rho)^{N-1}} \rho^{i-1}, \quad 1 \leq i \leq N \\ \pi_0 &= \frac{1}{2(1+\rho)^{N-1}}. \end{aligned} \tag{2}$$

B. Modeling the content dynamics

We assume that each participating client issues requests for objects at a constant rate $\sigma > 0$, and therefore model the request process of each client by a fluid flow. The motivation for this fluid approximation is that requests occur at a much faster time scale (typically a few requests per hour for each client) than nodes join/leave events (at most once a day for each client, much less frequently for abrupt failures). Therefore, we expect the long-run average performance of the fluid model to be similar to that of the real, discrete-time system, where requests occur with any distribution with mean inter-request time $1/\sigma$. (This claim is experimentally validated for Poisson arrivals in Section VI.)

In particular, our model represents the instantaneous set of cached objects in the whole Squirrel network by a global amount of fluid called $X(t)$ at time t. Indeed, the Pastry substrate provides sufficient load balancing among nodes to assume the amount of fluid at each node is an equal share of the total fluid in the system, thereby allowing us to summarize this distributed state by the single variable $X(t)$. This quantity of fluid will increase when objects are downloaded in the network from the origin server and added to their home node, i.e. whenever there is a cache miss. On the other hand, the amount of fluid will decrease as cached objects become stale. We assume that cached objects have the same constant time-to-live in cache, given by $1/\theta$; this assumption is made both for the sake of simplicity and because most caches use a time-to-live calculation heuristic for objects without any specified expiration date (about 70% of requested objects [14]), which is generally subject to a default maximum value. The usual default value is 24 hours (see [14] for more details).

We assume that each node can store an unlimited number of objects. Though individual nodes would probably not dedicate too much memory to the collaborative cache, even reasonable cache sizes are sufficient to avoid losses due to a full cache; one reason for this is that cached objects become stale fast enough to avoid continuous increase of the content. For centralized caches, the largest size needed to avoid most capacity misses is dictated by the clients request rates [16] and is fairly small. We expect this property to be applicable to the Squirrel decentralized cache system.

As a result, the variation of the fluid is proportional to the miss rate and to the expiration rate θ when the nodes population is constant[3]. If we call $\mathbb{P}[\text{hit}|i,x]$ the hit probability when there are i connected nodes containing the fluid x, then the variation rate of the amount of fluid is

$$\frac{dx}{dt} = i\sigma\left(1 - \mathbb{P}[\text{hit}|i,x]\right) - \theta x$$

where $i\sigma$ is the overall request rate in the Squirrel network when there are i connected nodes.

We now define an appropriate model for the hit probability function $\mathbb{P}[\text{hit}|i,x]$. Let us first call c the total number of objects that can be requested (in our model, the total amount of existing fluid in the universe). Since x is the quantity of cached fluid, a very simple model for the hit probability is $\mathbb{P}[\text{hit}|i,x] = \frac{x}{c}$. However, this linear function does not take into account the fact that some objects may be requested more often than others and thus are more likely to be present in the network. Since the popularity of Web objects follows a Zipf-like distribution [17], we can also model $\mathbb{P}[\text{hit}|i,x]$ as a concave function of the type $\mathbb{P}[\text{hit}|i,x] = \left(\frac{x}{c}\right)^{\beta}$, which reflects the fact that:

- When the amount of fluid is low, popular documents are quickly retrieved, resulting in a fast increase of the fluid.
- When most popular objects are present in the system, the fluid can then only increase with requests for rare objects.

It remains to specify how the node join and leave events impact the performance of our fluid model. We have seen in Section II that join events will probably not affect the performance of the system. On the other hand, we consider all failures (leaves) to be abrupt failures; this assumption is discussed in Section IV-D. Therefore, when a node leaves its share of objects is lost to the system. If we assume that the requests are well balanced across all nodes of the network (property of the Pastry hashing technique), then a fraction $1/i$ of the total amount of fluid is lost when a leave occurs if there were i nodes connected just before this leave event. This value has been confirmed empirically in [12].

For the sake of generality we introduce two mappings, $\Delta_u(i)$ and $\Delta_d(i)$, that give the fluid *reduction* generated by a node up and down event, respectively, given that i nodes were connected before this event. For Squirrel, $\Delta_d(i) = \frac{i-1}{i}$ and $\Delta_u(i) = 1$ as discussed above. In other words, if the amount of fluid is x and that i nodes are connected before a leave (resp. join) then the amount of fluid just after this event will be $x\Delta_d(i)$ (resp. $x\Delta_u(i)$).

A glossary of the model parameters is provided in Table I.

TABLE I
SYSTEM PARAMETERS

N	Maximum number of nodes
λ	Birth rate of each Squirrel node
μ	Death rate of each Squirrel node
ρ	λ/μ
π	Stationary distribution of $\{N_n\}_n$
σ	Request rate per client
θ	Expiration rate of cached objects
c	Total number of objects in the universe (i.e. total amount of fluid)
$\Delta_d(i)$	Fluid reducton after a node failure when there were $i \geq 1$ connected nodes. Default value: $(i-1)/i$ (cf. [12])
$\Delta_u(i)$	Fluid reduction after a node join when there were $i \geq 0$ connected nodes. Default value: 1 (cf. [12])

IV. PERFORMANCE ANALYSIS OF THE SQUIRREL P2P CACHE SYSTEM

In this section we provide a simple closed-form expression for the hit probability of the Squirrel system, under the main assumption that all objects are all equally popular. Although somewhat unrealistic, this assumption leads to a clearer analysis and highlights the parameters interactions of the system. For practical numerical results, we show how this assumption can be relaxed in Section IV-C. The end-to-end latency reduction offered by the Squirrel system, which might be a more meaningful metric than the hit probability, can easily be derived from the following results as shown in Section IV-B. Finally, we discuss the possible sources of inaccuracy of this model in Section IV-D and try to identify remedies whenever possible.

A. Hit probability analysis

Under the equal popularity assumption, the hit probability is a linear function of the amount of cached fluid, as shown in Section III-B. Our first task will be to characterize the fluid process $\{X(t)\}_t$.

Recall the definition of $N(t)$ and N_n, the number of connected nodes at time t and at time T_n+ (i.e. just after

[3]Assuming, of course, that at least one node is present.

the n-th jump time), respectively (see Section III-A). We assume that the sample-paths of $\{N(t)\}_t$ and $\{X(t)\}_t$ are right-continuous. Hence, $X_n \overset{\text{def}}{=} X(T_n+)$ is the amount of cached fluid just *after* the n-th jump in the process $\{N(t)\}_t$. For easy reference, the main definitions and notation have been collected in Table II.

TABLE II
VARIABLES

$X(t)$	Total amount of fluid in the system
$N(t)$	Number of connected nodes at time t
$\{T_n\}_n$	Jump times of the process $\{N_t\}_t$
$N_n = N(T_n^+)$	Number of connected nodes just after the n-th jump.
$X_n = X(T_n^+)$	Total amount of fluid just after the n-th jump.
$Y_n = X(T_{n+1}^-)$	Total amount of fluid just *before* the $n+1$-st jump. (at the end of the n-th period)
v_i	$\lim_{n\to\infty} \mathbb{E}\left[Y_n \mid N_n = i\right]/c$
η_i	$c/(1+\alpha/i)$
γ	$\sigma/(\mu c)$
α	$(\theta c)/\sigma$

The fluid process is defined as described in Section III-B: between two consecutive jumps (T_n, T_{n+1}) of $\{N(t)\}_t$ the fluid increases at rate

$$\frac{d}{dt}X(t) = \sigma N_n \left(1 - \frac{X(t)}{c}\right) - \theta X(t) \tag{3}$$

provided that $N_n > 0$. Integrating (3) gives

$$X(t) = \frac{\sigma N_n}{\frac{\sigma N_n}{c} + \theta} + \left(X_n - \frac{\sigma N_n}{\frac{\sigma N_n}{c} + \theta}\right) e^{-(t-T_n)(\theta + \frac{\sigma N_n}{c})}$$

for $T_n \leq t < T_{n+1}$ provided that $N_n > 0$. If $N_n = 0$ then $X(t) = 0$ for $T_n \leq t < T_{n+1}$.

If T_n corresponds to a node leave (resp. join) then the amount of cached fluid is reduced as follows

$$X_n = \Delta_d(N_n) X(T_n-) \text{ (resp. } X_n = \Delta_u(N_n) X(T_n-))$$

Therefore, $\{X(t)\}_t$ is a piecewise (exponential) process, with randomness at jump times $\{T_n\}_n$. A sample path of the process $\{(N(t), X(t))\}_t$ is represented on Fig. 1.

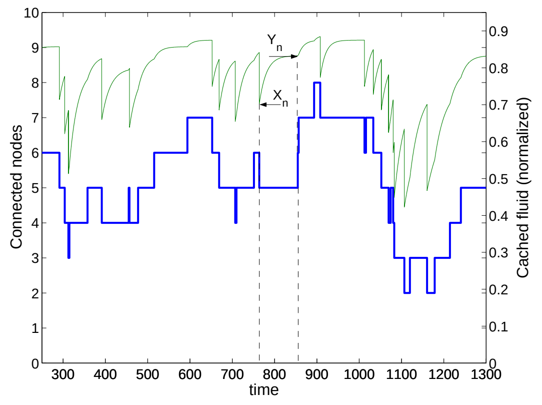

Fig. 1. Sample path of $\{(N(t), X(t))\}_t$.

For the sake of convenience we introduce

$$\eta_i \overset{\text{def}}{=} \frac{c}{1 + \frac{\theta c}{i\sigma}}. \tag{4}$$

We can now re-write the solution of (3) as

$$X(t) = \eta_{N_n} + (X_n - \eta_{N_n})\, e^{-(t-T_n)\frac{\sigma N_n}{\eta_{N_n}}},\ T_n \leq t < T_{n+1} \tag{5}$$

The process $\{(N(t), X(t))\}_t$ is an irreducible Markov process on the set $\{0, 0\} \cup \{\{1, 2, \ldots, N\} \times [0, c)\}$. Denote by X the stationary regime of $\{X(t)\}_t$.

We define the steady-state hit probability p_H as

$$p_H = \frac{\mathbb{E}\left[X\right]}{c} \tag{6}$$

We give a simple formula for p_H in Proposition 4.1. This formula is expressed in terms of the following new parameters that will play a key role in the understanding of the system behavior[4]:

$$\alpha \overset{\text{def}}{=} \frac{\theta c}{\sigma} \quad \text{and} \quad \gamma \overset{\text{def}}{=} \frac{\sigma}{\mu c}\ . \tag{7}$$

Proposition 4.1: Assume that for $i = 0, \ldots, N-1$,

$$0 \leq \Delta_u(i)\Delta_d(i+1) \leq 1. \tag{8}$$

The hit probability p_H is given by

$$p_H = \frac{1}{(1+\rho)^N} \sum_{i=1}^{N} \binom{N}{i} \rho^i\, v_i \tag{9}$$

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

$$A\,\mathbf{v} = \mathbf{b} \tag{10}$$

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

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

Proof: The idea of the proof is to first compute the expected amount of cached fluid just before a jump in the process $\{N(t)\}_t$ conditioned on the value of $N(t)$ just before this jump, and then to invoke Palm calculus to deduce the expected amount of cached fluid at *any time*.

Let Y_n be the amount of cached fluid just before the $(n+1)$-st jump in the process $\{N(t)\}_t$ (i.e. $Y_n = X(T_{n+1}-)$). We first compute $v_i \overset{\text{def}}{=} \lim_{n\to\infty}(1/c)\mathbb{E}\left[Y_n \mid N_n = i\right]$ for $1 \leq i \leq N$. We show in Appendix I that v_i satisfies the following recursive equation:

$$\begin{aligned} v_i\left(\rho(N-i) + \alpha\gamma + (\gamma+1)i\right) &= i\Delta_u(i-1)v_{i-1} \\ &+ \rho(N-i)\Delta_d(i+1)v_{i+1} + i\gamma \end{aligned} \tag{11}$$

[4] The system is defined in terms of 6 parameters: N, c, ρ, μ, θ, σ; definitions in (7) will allow us to express the hit probability only in terms of 4 parameters, namely, N, ρ, α and γ, as shown in Proposition 4.1.

for $i = 1, 2, \ldots, N$, or equivalently (10) in matrix form with $\mathbf{v} = (v_1, \ldots, v_N)$.

The uniqueness of the solution of (10) is shown in Appendix IV. The vector $\mathbf{v}$ in (10) gives the conditional stationary expected amount of cached fluid just before jump epochs (up to a multiplicative constant).

However, the hit probability p_H in (6) is defined in terms of the stationary expected amount of cached fluid at *arbitrary* epochs. The latter metric can be deduced from the former by using Palm calculus, through the identity (see e.g. [18, Formula (4.3.2)])

$$\mathbb{E}[X] = \Lambda \mathbb{E}^0 \left[\int_0^{T_1} X(t) dt \right] \tag{12}$$

where $\mathbb{E}^0$ denotes the expectation with respect to the Palm distribution[5], T_1 denotes the time of the first jump after 0, and where Λ denotes the global rate of the Engset model, i.e.

$$\Lambda = \frac{1}{\mathbb{E}^0[T_1]}. \tag{13}$$

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 connected nodes and the amount of cached fluid respectively, just before time 0 (i.e. just before the jump to occur at time 0).

We first compute $1/\Lambda$. Using (2) we find

$$\begin{aligned} \frac{1}{\Lambda} &= \sum_{i=0}^{N} \pi_i \, \mathbb{E}^0[T_1 \mid N_0 = i] = \frac{1}{\mu} \sum_{i=0}^{N} \frac{\pi_i}{\rho(N-i)+i} \\ &= \frac{1+\rho}{2N\rho\mu}. \end{aligned} \tag{14}$$

We show in Appendix II that

$$\mathbb{E}[X] = \frac{c}{(1+\rho)^N} \sum_{i=1}^{N} \binom{N}{i} \rho^i v_i. \tag{15}$$

Dividing both sides of (15) by c, we get (9), which concludes the proof. ■

Conditions (8) in Proposition 4.1 ensure that the system (10) has a unique solution (see Appendix III). They are satisfied for the home store scheme (since $\Delta_u(i)\Delta_d(i+1) = i/(i+1)$).

Remark 4.1: Since A is a tridiagonal matrix, (10) can be solved in only $\mathcal{O}(N)$ operations, once the mappings Δ_u and Δ_d are specified.

B. Latency reduction

The expected delay to fetch a document can easily be derived from the hit probability as follows: let T_e and T_i be the external and internal latency, respectively. The internal latency is the average delay induced by the Squirrel network, and thus is experienced by clients even in case of home node hits. The external latency is caused by network bottlenecks and Web server delays outside the organization, and is added to the internal latency in case of a miss when an object has to be retrieved from the origin Web server by its home node and is then sent to the client through the Squirrel network. Typically, T_e accounts for most of the total latency in the absence of caching (e.g. 77%, and up to 88% for a geographically located network [19]). The total expected delay with Squirrel is

$$\mathbb{E}[T] = T_i \, p_H + (T_i + T_e)(1 - p_H).$$

The Squirrel cache system reduces the average delay by saving the external latency whenever there is a hit. The relative latency reduction observed with Squirrel is thus

$$\frac{T_i + T_e - \mathbb{E}[T]}{T_i + T_e} = p_H \frac{T_e}{T_i + T_e}.$$

C. Zipf popularity

A concave model for the hit probability such as $\mathbb{P}[\text{hit}|i, x] = (x/c)^\beta$ makes the differential equation (3) nonlinear and without a known closed-form solution. An alternative approach is to consider K classes of decreasing popularity: the first class contains the c_1 most popular objects, and the K-th class contains the c_K least popular ones. With c being the total number of existing objects, we have $\sum_{k=1}^{K} c_k = c$. Let us call o_k the index of the least popular object of class k in the ordered set of objects, that is,

$$o_k = \sum_{l=1}^{k} c_l. \tag{16}$$

Using the Zipf-like popularity distribution [17], the probability of each class of fluid can easily be obtained as

$$p_k = \mathbb{P}[\text{request for class } k] = \frac{o_k^{1-\beta} - o_{k-1}^{1-\beta}}{c^{1-\beta}} \tag{17}$$

for class k, where β is the skew factor of the Zipf distribution.

The request rate for each class is given by $\sigma_k = \sigma p_k$ for $k = 1, 2, \ldots, K$, where σ is the average request rate per node (regardless of object popularity).

The hit probability $\hat{p}_H$ is now defined as the weighted sum of the conditional hit probabilities within each class, namely,

$$\hat{p}_H = \sum_{k=0}^{K} \frac{\mathbb{E}[X_k]}{c_k} p_k \tag{18}$$

where X_k is the (stationary) amount of cached fluid of class k. Similar to the derivation of the hit probability p_H in Proposition 4.1, we find that

$$\hat{p}_H = \sum_{k=0}^{K} p_k \frac{1}{c_k(1+\rho)^N} \sum_{i=1}^{N} \binom{N}{i} \rho^i \, v_i^{(k)} \tag{19}$$

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

$$A^{(k)} \, \mathbf{v}^{(\mathbf{k})} = \mathbf{b}^{(\mathbf{k})} \tag{20}$$

with $\mathbf{b}^{(\mathbf{k})} = (b_1^{(k)}, \ldots, b_N^{(k)})^T$ a vector whose components are given by $b_i^{(k)} = \frac{i\sigma_k}{\mu}$ for $i = 1, 2, \ldots, N$, and $A^{(k)} =$

[5]The Palm distribution is the distribution of the process $\{X_t\}_t$ assuming that a jump occurs at time 0 and that the system is in steady-state at time 0.

$[a_{i,j}^{(k)}]_{1\le i,j\le N}$ a $N \times N$ tridiagonal matrix whose non-zero elements are

$$\begin{array}{lll} a_{i,i}^{(k)} = & \frac{\theta}{\mu} + \frac{\sigma_k i}{\mu c_k} + i + \rho(N-i), & 1 \le i \le N \\ a_{i,i-1}^{(k)} = & -i\Delta_u(i-1), & 2 \le i \le N \\ a_{i,i+1}^{(k)} = & -\rho(N-i)\Delta_d(i+1), & 1 \le i \le N-1. \end{array}$$

D. Discussion and extensions

We now discuss some specific features that were not explicitly taken into account in the analysis of Section IV-A, apart from the popularity of documents.

The first remark is that the model assumes that every requested object is saved in the cooperative cache when downloaded a first time from the origin server. However, a non-negligible fraction (around 28%, cf. [14]) of the requested objects is in practice non-cacheable (mainly, expiration date before current date, but also explicit non-cacheable). We can take into account the uncacheability in our model as follows: let u be the fraction of objects that are uncacheable. So far, we have considered that the fluid increases after each miss, thereby implicitly assuming that all objects are cacheable. The uncacheability can be incorporated in our model by considering that only a fraction $1-u$ of misses will yield a fluid increase. This gives rise to the following equation

$$\begin{aligned} \frac{d}{dt}X(t) &= (1-u)\sigma N_n \left(1 - \frac{X(t)}{c}\right) - \theta X(t) \\ &= (1-u)\sigma N_n - \left(\frac{(1-u)\sigma N_n}{c} + \theta\right) X(t) \end{aligned} \qquad (21)$$

for $T_n < t < T_{n+1}$ and $N_n \in \{1, 2, \ldots, N\}$, since only requests for cacheable objects will lead to a fluid increase. Therefore, uncacheable objects can be added to the model simply by modifying the request rate accordingly.

Secondly, the impact of node join and leave events, modeled through the mappings Δ_u and Δ_d, may be slightly different from the values described in Section III. Indeed, two factors need to be taken into account. On the one hand, some nodes may announce their intention to disconnect, thereby avoiding performance degradation (see Section II). This would change the value of $\Delta_d(i)$. Though Proposition 4.1 provides an expression for general values of $\Delta_d(i)$, this would require a re-estimation of $\Delta_d(i)$. Since it would not exceed one, condition (8) would still be satisfied. On the other hand, the individual Squirrel caches may be stored either on disk or in memory. In the first case, a node i may join with a previously stored set of documents that have not been removed from its disk cache when the node went down. This would possibly add fluid into the network, if the corresponding objects have not been retrieved by the system while i was down and if i has not announced its last departure. The problem would not only be to re-estimate $\Delta_u(i)$, but also that $\Delta_u(i)$ might be greater than one, making condition (8) more difficult to verify. However, we expect that node i will stay down for a minimum time that will be orders of magnitude greater than requests inter-arrival times (the reboot time is typically a few minutes). Meanwhile, most of the objects stored in node i will be requested again and added to their new home nodes. As a result, when i will join again the Squirrel network with its own full cache, it will probably not add any fluid in the system, thus guaranteeing that $\Delta_u(i) \le 1$, and thereby the validity of assumption (8).

Finally, formula (9) involves binomial coefficients $\binom{N}{i}$ and an exponential in N. Therefore, computing p_H accurately for very large values of N might reveal difficult. Nonetheless, we would like first to mention that though we have occasionally encountered such problems, Proposition 4.1 is clearly tractable for the order of magnitude of several thousands of nodes, where a simulation would be untractable for high-confidence results. In addition, for much larger values of N, we believe that the node dynamics can be approximated by an $M/M/\infty$ model instead of an Engset model. This extension is a work in progress.

V. Qualitative insight in the Squirrel system

Proposition 4.1 shows that the performance of the Squirrel system exhibits only four degrees of freedom: N, ρ, γ, α while our model introduced six parameters $(N, \lambda, \mu, \sigma, \theta, c)$. We now examine the relative importance of these new parameters and how they characterize the Squirrel system behavior.

We first rapidly examine the influence of ρ. Fig. 2 shows that while there is a sharp drop of the hit probability for very small values of ρ (smaller that one), the performance is almost constant when ρ increases. Therefore, except when it is close to zero, ρ has very little influence on the performance of the Squirrel system. Also, it is very unlikely that ρ will be really small, since it would mean a non-negligible probability of reaching a state when all nodes are down (a very unrealistic situation). Under these circumstances, the limiting factors for the hit probability will be parameters N, γ and α.

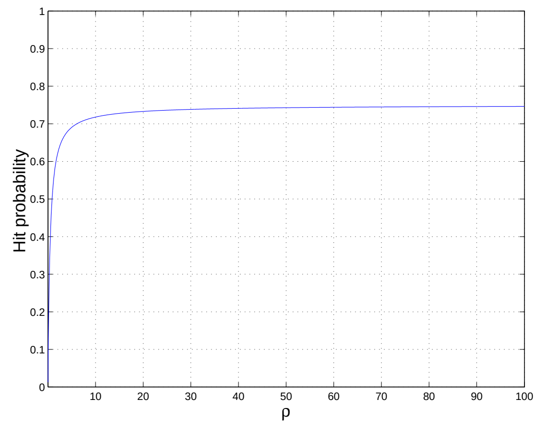

Fig. 2. Impact of ρ (with $N = 3$, $\alpha = 1$ and $\gamma = 2$).

In Fig. 3 we examine the comparative influence of γ and α on the hit probability. We find that for fixed α, the hit probability as a function of γ follows a concave shape, and can reach almost one when $\alpha = 0$. (This is consistent with our observation that ρ does not limit the hit probability when greater or equal to one.) Recall that $\gamma = \sigma/(\mu c)$ where σ is the individual request rate of the nodes. This concave shape in

γ reminds us the log-like[6] performance of a centralized Web cache (or Web cache cluster) as described in [16], [20], [21].

However, we observe that the hit probability is high on a very narrow domain ($\alpha \leq 1$, $\gamma \geq 10$). Indeed, there is a strong attenuation in α, so γ only has a real impact on the performance when α is small.

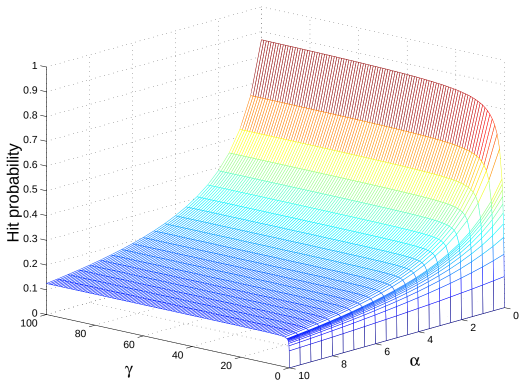

Fig. 3. Impact of γ and α on the hit probability (with $N = 3$ and $\rho = 1$). (Note that α is decreasing.)

These observations suggest possible methods to improve the performance of the Squirrel system. The best possible improvement will be to reduce parameter $\alpha = \theta c/\sigma$. Since the total number of existing objects, c, cannot be modified, there are two options:

- Reduce the expiration rate θ as much as possible: increase the default value of the maximum allowed value (denoted by CONF_MAX) in the freshness calculation heuristic for example [7], especially since most cGET requests (e.g. 90%) are responded with Not-Modified message [14]. Another solution can be the refreshment policy proposed by Cohen and Kaplan in [23].
- Increase the request rate σ, for instance by using prefetching techniques. We believe that prefetching can be incorporated to the fluid model, which will allow us to quantify the gain of using it. Intuitively, although increasing the request rate will increase the load in the system, it will also increase the rate at which objects are retrieved to the Squirrel network. This phenomenon is already known in the context of centralized caches [16].

Finally, if the global shape of the hit probability does not depend on N, the optimal values of γ and α vary with N. As a result any optimization of the system requires a realistic estimation of the maximum number of nodes in the network.

[6]The hit rate is either a logarithm or a small power of the global request rate.

[7]In the typical freshness calculation heuristic, the lifetime is min(CONF_MAX, CONF_PERCENT$\times$(Date-LastModified)) where CONF_PERCENT is a fraction typically limited to 10% and CONF_MAX a value typically equal to a day, since HTTP/1.1 specifies that a cache must attach a warning to any response whose age is more than 24 hours [22].

VI. Experimental results

In this section we compare quantitatively our macroscopic fluid model with a discrete-event driven simulation of the Squirrel home-store system. Request arrivals are Poisson and object time-to-live are taken to be all constant and all identical. We also assume that nodes follow the same time-evolution as in the fluid model, i.e. an Engset model. The external latency is taken into account whereas the internal latency is considered to be zero (corresponding to instantaneous internal transfers). Simulation results are given with 99% confidence intervals.

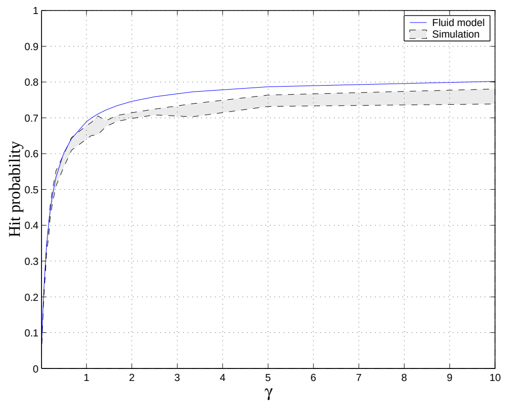

Fig. 4. Fluid model vs discrete-event simulation. ($N = 10$, $\rho = 1$ and $\alpha = 1$).

Fig. 4 displays the hit probability as a function of γ with $\rho = 1$ and $\alpha = 1$. We observe that the fluid model curves closely follow the same shapes as the discrete-event simulations and therefore mimics the simulated system behavior very accurately. We conclude that the model is robust to assumptions such as the request rate distribution (which we assumed constant in Section IV-A), and although microscopic features such as objects replication and local hits (requests not forwarded to home node) are being ignored, the fluid model provides an accurate approximation for the actual performance of the Squirrel system.

Furthermore, we would like to emphasize that each simulation, even for very small networks (10 nodes), ran for several hours (typically 20 hours or more) on a Pentium 4 running at 2.66GHz. Even if our code may not be fully optimized, it is clear that in comparison with the instantaneous results provided by the fluid model, simulation is very slow and limited to very small network sizes.

We show in Fig. 5 how the hit probability would look like for large networks, since simulation of such systems would be either too slow or statistically irrelevant. Since Fig. 4 validated the accuracy of our model for small network sizes, we expect the results for large networks to be as relevant – though we do not have simulation results to demonstrate it. We observe the same shape as in Fig. 4, though on a larger range (thanks to the low complexity of the model results), which suggests that Squirrel scales with the same type of behavior, and that

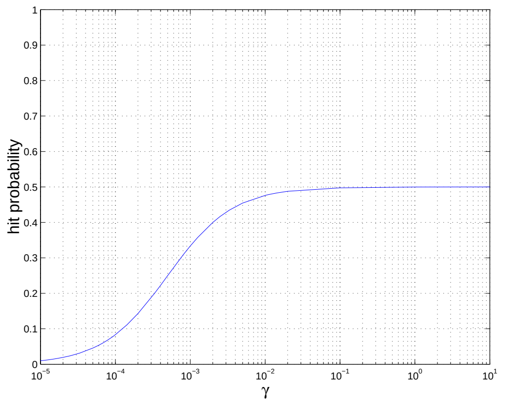

Fig. 5. Hit probability for large networks ($N = 2000$ and $\alpha = 1000$).

the characteristics observed in Section V should be valid for large networks.

VII. Conclusion and perspectives

In this paper we have proposed and studied a fluid model for the performance analysis of the Squirrel cooperative cache system. To cope with the large number of users that join and leave the cache system randomly, we have approximated the request streams of the individual nodes by a fluid flow. Our resulting stochastic fluid model turns out to be mathematically tractable, and has allowed us to provide a simple and very low-complexity procedure for computing the hit probability. Moreover, the analysis has emphasized the key characteristics of the Squirrel system and allows a better understanding of its performance. Comparison with simulation results has shown that the hit probability provided by the solution to the model is an accurate approximation of the actual hit rate and has validated the qualitative conclusions driven by the model results.

It is worth noticing that our analysis does not strictly limit to Squirrel, but can also be straightforwardly applied to other systems based on distributed hash tables such as Chord, CAN or Tapestry ([24]–[26]) for example. The necessary conditions assumed in this work are the load balancing (provided by Pastry), and above all the absence of replication that characterizes the home-store scheme.

Future work will focus on extending the model to handle prefetching techniques and larger populations of peers. We also intend to quantify the accuracy of the approach in the Zipf-like popularity model. Finally, the accuracy and tractability of this fluid model suggest that it could be adapted to analyze content distribution systems involving document replication, such as P2P file sharing applications and CDNs.

Acknowledgment

The authors thank K. W. Ross for useful discussions during the course of this work.

Appendix I
Proof of equation (11)

Recall that $v_i = \lim_{n\to\infty} \frac{\mathbb{E}\left[Y_n \mid N_n = i\right]}{c}$ for $1 \le i \le N$. With (5) we have

$$\begin{aligned} v_i &= \frac{1}{c}\lim_{n\to\infty} \mathbb{E}\left[Y_n \mid N_n = i\right] \\ &= \frac{1}{c}\lim_{n\to\infty} \mathbb{E}\left[\eta_i + (X_n - \eta_i)\, e^{-(T_{n+1}-T_n)\frac{\sigma i}{\eta_i}} | N_n = i\right] \\ &= \frac{1}{c}\lim_{n\to\infty}\left(\eta_i \times \frac{\alpha\gamma+\gamma i}{\alpha\gamma+\gamma i+\rho(N-i)+i}\right. \\ &\quad \left. + \frac{(\rho(N-i)+i)\,\mathbb{E}\left[X_n \mid N_n = i\right]}{\rho(N-i)+i+\alpha\gamma+\gamma i}\right). \end{aligned} \tag{22}$$

To derive (22) 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 now evaluate $\lim_{n\to\infty} \mathbb{E}[X_n \mid N_n = i]$ for $1 \le i \le N$. Conditioning on N_{n-1} we have

$$\begin{aligned} &\lim_{n\to\infty} \mathbb{E}[X_n \mid N_n = i] = \\ &\qquad \lim_{n\to\infty} \mathbb{E}\left[X_n | N_n = i, N_{n-1} = i-1\right] \\ &\qquad \times \mathbb{P}[N_{n-1} = i-1 | N_n = i] \\ &\qquad + \lim_{n\to\infty} \mathbb{E}\left[X_n \mid N_n = i, N_{n-1} = i+1\right] \\ &\qquad \times \mathbb{P}[N_{n-1} = i+1 \mid N_n = i]\, \mathbb{1}_{[i<N]} \\ &= \Delta_u(i-1) \lim_{n\to\infty} \mathbb{E}\left[Y_{n-1} \mid N_{n-1} = i-1\right] \\ &\qquad \times \frac{\pi_{i-1}}{\pi_i}\frac{\rho(N-i+1)}{\rho(N-i+1)+i-1} \\ &\qquad + \Delta_d(i+1) \lim_{n\to\infty} \mathbb{E}\left[Y_{n-1}, \mid N_{n-1} = i+1\right] \\ &\qquad \times \frac{\pi_{i+1}}{\pi_i}\frac{i+1}{\rho(N-i-1)+i+1}\mathbb{1}_{[i<N]} \\ &= c\frac{i\Delta_u(i-1)v_{i-1} + \Delta_d(i+1)v_{i+1}\rho(N-i)}{\rho(N-i)+i} \end{aligned} \tag{23}$$

by using (2) and the definition of v_i. Finally, dividing both sides by c and introducing (23) into (22) yields (11). ■

Appendix II
Proof of equation (15)

Let us determine $\mathbb{E}\left[X\right]$ from the v_is. We use the Palm formula and condition on the value of N_0. From (12), (5), (14) we find

$$\begin{aligned} \mathbb{E}\left[X\right] &= \Lambda\sum_{i=1}^{N}\pi_i\mathbb{E}^0\left[\int_0^{T_1}\left(\eta_i + (X_0 - \eta_i)e^{-t\frac{\sigma i}{\eta_i}}\right)dt | N_0 = i\right] \\ &= \Lambda\left[\sum_{i=1}^{N}\pi_i\eta_i\mathbb{E}^0[T_1 \mid N_0 = i] + \sum_{i=1}^{N}\frac{\pi_i\eta_i}{\sigma i}\right. \\ &\qquad \left. \times\mathbb{E}^0\left[(X_0 - \eta_i)\left(1 - e^{-T_1\frac{\sigma i}{\eta_i}}\right) \mid N_0 = i\right]\right] \\ &= \Lambda\left[\sum_{i=1}^{N}\pi_i\eta_i\frac{1}{\lambda(N-i)+\mu i} + \sum_{i=1}^{N}\frac{\pi_i\eta_i}{\sigma i}\right. \end{aligned}$$

$$\times \left(\mathbb{E}^0[X_0 \,|\, N_0 = i] - \eta_i\right)$$
$$\times \left(1 - \mathbb{E}^0\left[e^{-T_1 \frac{\sigma i}{\eta_i}} \,|\, N_0 = i\right]\right)\Bigg] \tag{24}$$
$$= \frac{\Lambda}{\mu}\left[\sum_{i=1}^{N} \pi_i \eta_i \frac{1}{\rho(N-i)+i} + \sum_{i=1}^{N} \pi_i \frac{\mathbb{E}^0[X_0 \,|\, N_0 = i] - \eta_i}{\rho(N-i)+i+\alpha\gamma+\gamma i}\right]$$
$$= \frac{2N\rho}{1+\rho}\sum_{i=1}^{N} \pi_i \left[\eta_i \frac{1}{\rho(N-i)+i} + \frac{\mathbb{E}^0[X_0 \,|\, N_0 = i] - \eta_i}{\rho(N-i)+\alpha\gamma+(\gamma+1)i}\right]. \tag{25}$$

By definition, $\mathbb{E}^0[X_0 \,|\, N_0 = i] = \lim_{n\to\infty} \mathbb{E}[X_n \,|\, N_n = i]$, which has been computed in (23). By combining (23) and (11) we obtain

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

Plugging this value of $\mathbb{E}^0[X_0 \,|\, N_0 = i]$ into the r.h.s. of (25), and using (2), yields after some straightforward algebra:

$$\mathbb{E}\,[X] = \frac{c}{(1+\rho)^N}\sum_{i=1}^{N}\binom{N}{i}\rho^i v_i.$$

which is nothing but (15). ■

Appendix III
Stationary distribution of the Engset model at jump times

In this section we compute the stationary distribution of the Markov chain $\{N_n, n \geq 1\}$. Since this chain is periodic with period 2, it has no limiting distribution. Our goal is to compute the time average of $\mathbb{P}[N_n = i]$.

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 $0 \leq i \leq N-1$, $p_{i,i-1} = i/(\rho(N-i)+i)$ for $1 \leq i \leq N$ and $p_{i,j} = 0$ when $|i-j| \neq 1$.

Since this Markov chain has a finite-state space and is irreducible, it is positive recurrent [27, Cor. 5.3.19, 5.3.22]. Therefore, it possesses a unique stationary (i.e., invariant) distribution $\pi = (\pi_0, \cdots, \pi_N)$ given by the (unique) solution of the equation $\pi = \pi P$ such that $\sum_{i=0}^{N} \pi_i = 1$ [28, page 208].

We proceed by induction to compute π. From the equation $\pi = \pi P$ 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

$$\pi_j = \frac{\rho(N-j)+j}{j}\frac{\rho^{j-1}}{(j-1)!}\frac{(N-1)!}{(N-j)!}\pi_0 \tag{26}$$

for $j = 1, 2, \ldots, N$. Assume that (26) holds for $j = 1, 2, \ldots, i < N-1$. Let us show that it still holds for $j = i+1$. We have

$$\begin{aligned}
\pi_{i+1} = {} & \frac{\rho(N-(i+1))+i+1}{i+1} \\
& \times \left(\pi_i - \frac{\rho(N-(i-1))}{\rho(N-(i-1))+i-1}\pi_{i-1}\right) \\
= {} & \frac{\rho(N-i-1)+i+1}{i+1}\Bigg(\frac{\rho(N-i)+i}{i!}\frac{\rho^{i-1}}{(N-i)!} \\
& -\frac{\rho(N-i+1)}{\rho(N-i+1)+i-1} \times \frac{i-1+\rho(N-i+1)}{i-1} \\
& \times \frac{\rho^{i-2}}{(i-2)!(N-i+1)!}\Bigg)(N-1)!\pi_0 \qquad (27) \\
= {} & \frac{\rho(N-i-1))+i+1}{i+1}\frac{\rho^i(N-1)!}{i!\,(N-i-1))!}\pi_0
\end{aligned}$$

where (27) 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 (2). Plugging this value of π_0 into (26) gives the general expression of (2). ■

Appendix IV
Uniqueness of the solution of (10)

The linear system (10) 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 [29, Sec. 3.5] $A = LU$ with

$$L = \begin{pmatrix} l_1 & 0 & \cdots & 0 \\ \beta_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \beta_n & l_n \end{pmatrix}$$

and

$$U = \begin{pmatrix} 1 & u_1 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & u_{n-1} \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

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

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

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 $l_1 = \rho(N-1)+1+\gamma(1+\alpha)$. Assume that $l_i > \gamma(i+\alpha)+\rho(N-i)$ for $i = 1, 2, \ldots, n < N-1$ and let us show that $l_{n+1} > \gamma(n+1+\alpha)+\rho(N-n-1)$.

We have

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

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

■

REFERENCES

[1] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proc. IEEE INFOCOM 2003*, San Francisco, California, April 2003.

[2] L. Zou and M. Ammar, "A file-centric model for peer-to-peer file sharing systems," in *Proc. ICNP 03*, Atlanta, Georgia, USA, November 2003.

[3] D. Anick, D. Mitra, and M. M. Sondhi, "Stochastic theory of data-handling systems with multiple sources," *Bell Systems Technical Journal*, vol. 61, pp. 1871–1894, 1982.

[4] A. Elwalid and 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.

[5] ——, "Effective bandwidth of general markovian traffic sources and admission control of high speed networks," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 329–343, June 1993.

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

[7] F. LoPresti, Z. Zhang, D. Towsley, and 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.

[8] R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn, "Statistical service assurances for traffic scheduling algorithms," *IEEE Journal on Selected Areas in Communications, Special Issue on Internet QoS*, vol. 18, pp. 2651–2664, December 2000.

[9] M. Reisslein, K. W. Ross, and S. Rajagopal, "A framework for guaranteeing statistical QoS," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 27–42, February 2002.

[10] B. Liu, D. Figueiredo, Y. Guo, J. Kurose, and D. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[11] F. Clévenot, P. Nain, and K. W. Ross, "Stochastic fluid models for cache clusters," INRIA, Sophia Antipolis, Tech. Rep. 4815, May 2003.

[12] S. Iyer, A. Rowstron, and 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.

[13] A. Rowstron and P. Druschel, "Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems," in *Proc. Int. Conf. on Distributed Systems Platforms (Middleware)*, Heideberger, Germany, November 2001.

[14] E. Cohen and H. Kaplan, "The age penalty and its effect on cache performance," in *Proc. 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, California, 2001, pp. 73–84.

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

[16] B. Duska, D. Marwood, and 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.

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

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

[19] T. Kroeger, D. D. E. Long, and J. C. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching," in *Proc. USENIX Symp. on Internet Technologies and Systems*, Monterey, California, 1997, pp. 13–22.

[20] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and 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.

[21] S. Gribble and 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.

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – – HTTP/1.1," RFC 2616, June 1999.

[23] E. Cohen and H. Kaplan, "Refreshment policies for web content caches," in *Proc. IEEE INFOCOM '01*, Anchorage, Alaska, April 2001.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM 2001*, San Diego, California, August 2001, pp. 149–160.

[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM 2001*, San Diego, California, August 2001.

[26] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UCB, Tech. Rep. UCB/CSD-01-1141, April 2000.

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

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

[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.