Stochastic Models of Pull-Based Data Replication in P2P Systems

Xiaoyong Li and Dmitri Loguinov*
Texas A&M University, College Station, TX, 77843, USA
xiaoyong@cs.tamu.edu, dmitri@cs.tamu.edu

***Abstract*—We consider pull-based data synchronization issues between a source and its replicas in P2P networks. Under continuous information change and lazy synchronization, these systems are highly susceptible to serving outdated content, which negatively affects their performance and user satisfaction. To understand these scenarios, we first introduce a novel model of interaction between two stochastic point processes – updates at the source and downloads at the replica – and derive the probability that a random query against the replica retrieves fresh content. Unlike prior work, we assume non-Poisson dynamics and determine statistical properties of the replication process that make it perform better for a given download rate. The second half of the paper applies these results to several more difficult algorithms – cascaded replication, cooperative caching, and redundant querying from the clients. Surprisingly, we discover that optimal cooperation involves just a single peer and that redundant querying can hurt the ability of the system to handle load (i.e., may lead to lower scalability).**

I. Introduction

P2P file sharing systems have received tremendous interest in recent years among both Internet users and computer networking professionals. In the study of these networks, one fundamental problem is to handle peer overload that may arise due to the highly popular nature of certain content and/or temporal fluctuations in demand (e.g., flash crowds). A common solution for *static* files is to replicate them from each source to multiple peers, which distributes the load and thus improves file-query efficiency. Examples include protocols that replicate content close to the owner [17], [44], [50], near the requester [27], and along query paths [16], [43], [58].

With real-time operation of certain P2P applications, such as online auctions [24], decentralized collaboration [60], web caching [30], and online games [56], replication faces new challenges related to *data churn* (i.e., periodic content updates at the source). To provide accurate and reliable query results in these systems, replicated material must be continuously synchronized with that at the source. Without an effective consistency-maintenance strategy, these applications may suffer from degraded performance and lower user participation.

The majority of existing P2P synchronization methods are *push-based*. To allow the source easy discovery of replica location, these networks often employ structured P2P networks to establish a mapping from file IDs to nodes where they can be found [5], [7], [28], [33], [43], [55], [58]. Since the source must track the status and location of each replica, as well as reconfigure the distribution tree, these methods may suffer from high maintenance overhead, especially when the network structure is volatile (i.e., under high user churn).

In unstructured P2P networks, management of replicas is usually achieved by message spreading [11], [18], [26], [36], which may generate large amounts of redundant traffic and even lead to network collapse when search rates become sufficiently high. To address this problem, several studies [36], [45], [46], [47], [52] propose *pull-based* consistency control, which allows replicas to self-manage their membership in replication paths and decide when to download content from the source. Pull-based techniques have also been used in pub/sub systems [6], [22] and hybrid push/pull methods found application in decentralized online social networks [49].

In databases, it is well-known [31], [59] that pull-based synchronization improves both scalability and availability of the data, but at the expense of increased age of the content served to clients. As the source evolves, replicas in these networks go through periods of staleness, during which they provide outdated responses that do not reflect the true condition of the source. To measure system performance, it is generally accepted that the *probability of freshness* (i.e., likelihood that the most-recent version is available to consumers) accurately reflects the quality of a replication strategy. Although this metric has been considered by researchers in web-based systems [4], [9], [12], [42], [41], [53], it has never been explored in the context of P2P systems. We aim to fill this void below.

A. Contributions

We start by considering a single source driven by an update process N_U and a single replica with the corresponding download process N_D, which is independent of N_U. Our first contribution is to propose a general framework for modeling freshness under arbitrary renewal processes (N_U, N_D). This allows us to derive the freshness probability p in closed-form as a function of inter-update distribution $F_U(x)$ and inter-download distribution $F_D(x)$. Our formula for p generalizes all previous analytical results in the literature [2], [3], [8], [9], [10], [12], [21], [23], [25], [29], [32], [35], [37], [38], [39], [48], [51], [57], which were predominantly limited to Poisson N_U and two simple cases of $F_D(x)$.

Given a fixed download rate λ, our second contribution is to obtain a condition that allows comparison of freshness achieved by different download strategies $F_D(x)$. We show that freshness improves if the inter-synchronization interval

*Supported by NSF grants CNS-1017766 and CNS-1319984.

978-1-4799-6201-3/14/$31.00

becomes *stochastically larger in second order*. This allows us to prove that constant delays are optimal against *all* N_U. For the same p, they require 33% less bandwidth than exponential inter-download delays and 50% less than Pareto.

Based on these results, our third contribution is to analyze *cascaded* synchronization, where replicas receive content from other replicas along a fixed multi-hop path from the source. A common arrangement covered by this model is a b-way replication tree, which limits the source to b concurrent downloads, but keeps client-scalability arbitrarily high depending on the depth of the tree. Assuming independent operation among the replicas, we derive a recursive model that provides freshness at each level i. Our results show that in certain cases p decays exponentially fast as a function of the depth, suggesting an interesting coupling between system size and staleness.

Our fourth contribution is to propose a model of *cooperative caching*, in which m replicas form a single layer, in which each participant can synchronize not only with the source, but also k other replicas. For a target p, the goal is to determine the optimal (m, k) that maximizes the service rate of the entire system. The main caveat of this model is that it takes into account bandwidth constraints at the source and each replica. We show that making k or m too large is detrimental to performance; instead, *each parameter has a unique optimal value that achieves the highest service rate*, which can be $2-7$ times larger than under non-cooperative replication.

Our last contribution is to examine a scenario we call *redundant querying*, in which consumers have access to multiple independent caches. Issuing parallel queries to k replicas out of the m available, the hope is to improve freshness by selecting the most up-to-date copy of the source. We first show that freshness in this case can be computed using the original update process and a superposition of download processes from each of the contacted replicas. However, taking bandwidth into account, this analysis also leads to a surprising conclusion that *redundant querying with $k \geq 2$ sometimes produces lower performance than non-redundant*.

II. Related Work

Consistency maintenance in existing P2P networks can be classified into *push-based* and *pull-based*. In the former, the source is responsible for sending updates to replicas whenever it deems necessary. To achieve this, the source has to know the location of all of its replicas either by utilizing a rigid network structure that maps files to nodes [5], [7], [43], [55], [58] or randomly flooding the graph [11], [18], [26], [36]. In pull-based methods, nodes become replicas, discontinue being such, and adjust their download policy independently of the source. Existing work in this direction [45], [46], [47] focuses on determine the polling frequency using a family of linear-increase multiplicative-decrease (LIMD) algorithms.

In other fields, pull-based data synchronization has also been studied. In the context of web systems [4], [9], [12], [53], sources are typically HTML pages modified by their owners. Replicas can be search engines that use web-crawlers to periodically reload content and refresh their indexes; however, additional applications are possible as well – online monitoring systems [41], [42] of highly dynamic web streams (e.g., stock market, traffic), traditional web caching [13], [14], [15], RSS feed aggregation [48], and many others.

Fig. 1. System model. Arrows represent pull-based requests for information.

In the analytical literature, freshness p was first proposed by Coffman *et al.* [12] and later used by much of the follow-up work [4], [9], [8], [38], [53]. Other ways to capture staleness include information divergence between the source and its replica [39], age of the content served to clients [9], the number of missing updates at the replica [19], [29], and their combined age [34], [35], [48]. In all of these cases, models are derived under the assumption that the update process N_U is Poisson. There has been only one attempt [53] to relax this constraint, but it required deterministic knowledge all download instances. While appropriate in some cases, this model is difficult to evaluate in practice when inter-download delays are random and given by their distribution $F_D(x)$.

III. Single-Hop Replication

In this section, we consider a single source and one of its replicas. We first introduce our assumptions and notation, define freshness p, and derive its closed-form model. We then use simulations to highlight several examples.

A. System Model and Notation

We assume a model of data generation, replication, and consumption in Fig. 1. During system operation, the source experiences random updates in response to either external events (e.g., price bids in e-auctions, status changes in online games) or some internal computation (e.g., indexing, MapReduce output). In either case, each update represents certain tangible information that manipulates the current state of the source. The replica has no direct knowledge of these updates and must infer their occurrence only through periodic downloads. Its goal is to provide consumers with up-to-date responses to various types of queries.

Let u_i denote the time when the i-th update occurs at the source. Define $N_U(t) = \max\{i : u_i \leq t\}$ to be the number of updates in $[0, t]$ and $U_i = u_{i+1} - u_i$ as the i-th inter-update delay. Similarly, let d_k be the k-th download instance, $N_D(t) = \max\{k : d_k \leq t\}$ the number of such points in $[0, t]$, and $D_k = d_{k+1} - d_k$ the k-th inter-synchronization delay. To keep the system tractable, assume that N_D and N_U are independent renewal processes. This means that update intervals $\{U_i\}_{i=1}^{\infty}$ and synchronization delays $\{D_i\}_{i=1}^{\infty}$ are two sets of independent and identically distributed (iid) variables. This allows us to replace them with random variables $U \sim F_U(x)$ and $D \sim F_D(x)$, respectively.

B. Performance Measure

Observe that a local copy at the replica is *fresh* if and only if the last update time $u_{N_U(t)}$ is smaller than the last download

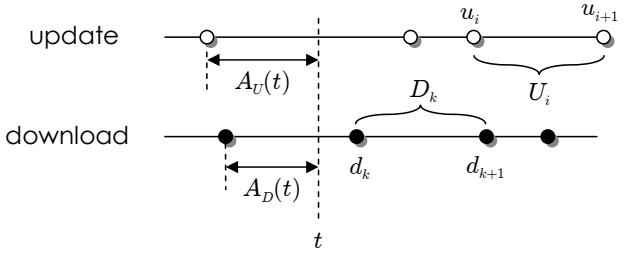
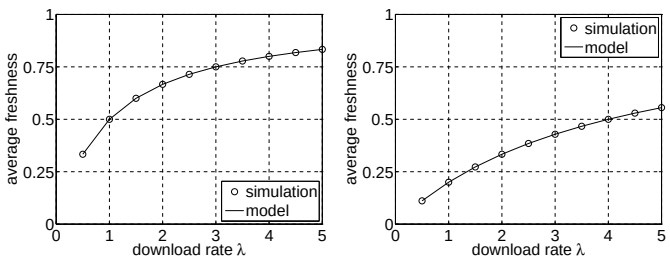


Fig. 2. Illustration of age and other variables.



(a) Pareto U, constant D (b) constant U, Pareto D

Fig. 3. Simulation results of (8) under $\mu = 2$.

time $d_{N_D(t)}$. Freshness of the data can thus be modeled by an alternating (ON/OFF) process:

$$\phi(t) = \begin{cases} 1 & u_{N_U(t)} < d_{N_D(t)} \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where t represents the time of a consumer's request to the replica. In practice, it is more convenient to express $\phi(t)$ as a function of age. Define the *download age* at t to be the time lag from the last synchronization to t:

$$A_D(t) = t - d_{N_D(t)} \tag{2}$$

and the *update age* to be that from the last update:

$$A_U(t) = t - u_{N_U(t)}. \tag{3}$$

These definitions are illustrated in Fig. 2, where the empty circles are update events and the solid ones are download instances. Using the figure, it is not difficult to see that a copy is fresh if and only if the download age is smaller than the update age, which means that:

$$\phi(t) = \begin{cases} 1 & A_U(t) > A_D(t) \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

We model the consumer as querying the replica at some large time t by which the system can be considered stationary. As $t \to \infty$, $A_U(t)$ and $A_D(t)$ converge to their equilibrium versions, which we call A_U and A_D, respectively. Define $\mu = 1/E[U]$ to be the update rate and let $\lambda = 1/E[D]$ be the download rate. Then, from renewal theory [54], the two ages have well-known distributions:

$$G_U(x) := P(A_U < x) = \mu \int_0^x (1 - F_U(y))dy \tag{5}$$

and

$$G_D(x) := P(A_D < x) = \lambda \int_0^x (1 - F_D(y))dy. \tag{6}$$

This allows us to formulate the main metric of system performance – the limiting probability that consumers encounter a fresh copy:

$$p := \lim_{t\to\infty} P(\phi(t) = 1) = P(A_U > A_D). \tag{7}$$

To keep notation simple and prevent unnecessary explanation, we generally use $F_X(x)$ to represent the distribution of variable X, $G_X(x)$ to denote its age distribution similar to (5)-(6), and lower-case functions $f_X(x)$ and $g_X(x)$ as the corresponding densities. We also use $\bar{F}_X(x) = 1 - F_X(x)$ as the CCDF (complementary cumulative distribution function) of variable X and replace $X(t)$ with its limiting variable X as $t \to \infty$, whenever doing so is appropriate.

C. Freshness Probability

Our next result directly follows from (4).

Theorem 1: Freshness experienced by consumers in steady-state is given by:

$$p = E[\bar{G}_U(A_D)] = \int_0^\infty \bar{G}_U(x) g_D(x) dx. \tag{8}$$

To perform a self-check against prior results with Poisson N_U, observe that (8) simplifies to $p = \lambda(1 - e^{-\mu/\lambda})/\mu$ under constant D and $\lambda/(\lambda + \mu)$ under exponential D, which is in agreement with [9], [12]. We use simulations to examine model accuracy in more interesting cases of general renewal processes. Since U and D are non-negative random variables defined on $(0, \infty)$, our Pareto CDF is $1 - (1 + x/\beta)^{-\alpha}$ for $\alpha > 1$ and $\beta > 0$. The mean of this distribution is $\beta/(\alpha - 1)$, where α is kept at 3 throughout the paper. We simulate each process to large enough t to reach stationarity of the underlying processes. This typically requires a few hundred units of time.

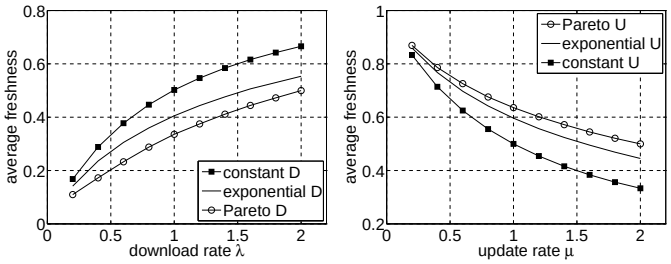
Observe in Fig. 3 that the model matches simulations very well, with constant download intervals performing significantly better against Pareto update cycles in (a) than the other way around in (b). For instance, using download rate $\lambda = 3$, which is 50% faster than the update rate $\mu = 2$ in the figure, part (a) achieves 75% freshness, while part (b) only 43%. It is unclear, however, whether constant D is always better than Pareto and what impact $F_U(x)$ has on the resulting p. We address this question next.

IV. Best Download Strategy

In this section, we study conditions under which one combination of processes (N_U, N_D) performs better than another.

A. Basics

Noticing from (6) that $g_D(x) = G'_D(x) = \lambda(1 - F_D(x))$, the result in (8) shows that expected freshness p is impacted by not only the product of update and download rates $\mu\lambda$, but also the entire functions $F_D(x)$ and $F_U(x)$. To establish order



(a) Pareto U (b) Pareto D

Fig. 4. Ordering of freshness under different families of distributions.

between distributions, we need the next definition commonly used in game theory and statistics.

Definition 1: Variable X is said to be *stochastically larger than* Y, which is written as $X \geq_{st} Y$, if $\bar{F}_X(x) \geq \bar{F}_Y(x)$ for all $x \geq 0$. Variable X is *stochastically larger than* Y *in second order*, which is written as $X \geq^2_{st} Y$, if $\int_0^x \bar{F}_X(y)dy \geq \int_0^x \bar{F}_Y(y)dy$ for all $x \geq 0$.

Since $\bar{G}_U(x)$ is a monotonically non-increasing function, it is easy to show that (8) produces the following result.

Lemma 1: For two download strategies driven by inter-synchronization delays X and Y, freshness $p_X \geq p_Y$ when the age of Y stochastically dominates that of X, i.e., $A_Y \geq_{st} A_X$.

To make this result useful, we next translate stochastic dominance between ages to that between the corresponding variables.

Lemma 2: Assuming $E[X] = E[Y] > 0$, variable X is stochastically larger than Y in second order, i.e., $X \geq^2_{st} Y$, if and only if the age of Y is stochastically larger than that of X, i.e., $A_Y \geq_{st} A_X$.

Proof: Let $G_X(x)$ and $G_Y(x)$ be the CDFs of A_X and A_Y, respectively. Define:

$$J(x) = \bar{G}_Y(x) - \bar{G}_X(x) \tag{9}$$

and expressed it as:

$$J(x) = \frac{\int_0^x \bar{F}_X(y)dy}{E[X]} - \frac{\int_0^x \bar{F}_Y(y)dy}{E[Y]}. \tag{10}$$

Since both means are positive and equal to each other, we get that $A_Y \geq_{st} A_X$ iff $J(x) \geq 0$ for all $x \geq 0$, which holds iff $X \geq^2_{st} Y$. ∎

Combining these observations, we obtain the following.

Theorem 2: For a given N_U and two download distributions $F_X(x)$ and $F_Y(x)$ with the same rate λ, freshness $p_X \geq p_Y$ when X stochastically dominates Y in second order. Similarly, for a given N_D and two update distributions $F_W(x)$ and $F_Z(x)$ with the same rate μ, freshness $p_W \geq p_Z$ when Z stochastically dominates W in second order.

B. Examples

Our last result shows that freshness is improved when D becomes stochastically larger in second order or U becomes the opposite, i.e., smaller. To put this in perspective, consider three classes of distributions often observed in practice. The first one is NWU (new worse than used), which means that conditioned on the fact that an interval is at least y time units, its surplus length beyond y is stochastically larger than the original interval size, i.e., $P(X > x + y|X > y) \geq P(X > x)$ for all $x, y \geq 0$. While many heavy-tailed distributions belong to NWU, two most common examples are Pareto and Weibull. If the inequality is reversed, we obtain what is known as NBU (new better than used). Examples include uniform and constant. Finally, if $P(X > x + y|X > y) = P(X > x)$, the distributions are called *memoryless* (i.e., exponential).

Suppose X is NWU, Y is memoryless, and Z is NBU such that $E[X] = E[Y] = E[Z]$. It then follows [20] that $Z \geq^2_{st} Y \geq^2_{st} X$. Applying this observation to Fig. 4(a), it is no wonder than Pareto D produces worse freshness than exponential, which in turn is worse than constant. For a fixed probability $p = 0.5$, the optimal case requires $33 - 50\%$ less bandwidth than the other two distributions. This relationship is reversed in application to U in Fig. 4(b) – Pareto is the best, while constant is the worst.

C. Optimality

From the discussion above, NBU download delays are better than NWU and exponential; however, it is unclear which NBU distribution is the best and whether other classes may be better than NBU. We are now ready to seek answers to these questions.

Lemma 3: For a given mean, a constant stochastically dominates all other random variables in second order.

Proof: Suppose l is the fixed mean of all distributions under consideration. Let $F_X(x) = \mathbf{1}_{x>l}$ be the CDF of a constant and $F_Y(x)$ be the CDF of another random variable Y such that $E[Y] = l$. Define:

$$\delta(x) := \int_0^x (\bar{F}_X(y) - \bar{F}_Y(y))dy \tag{11}$$

and observe that it suffices to prove that $\delta(x) \geq 0$ for all $x \geq 0$. For $x \leq l$, notice that $\bar{F}_X(x) = 1$ and thus:

$$\delta(x) = \int_0^x F_Y(y)dy \geq 0. \tag{12}$$

For $x > l$:

$$\begin{aligned}\delta(x) &= \int_0^x \bar{F}_X(y)dy - \int_0^x \bar{F}_Y(y)dy \\ &= l - \int_0^x \bar{F}_Y(y)dy \geq l - \int_0^\infty \bar{F}_Y(y)dy = 0,\end{aligned}$$

where we use the fact that $\int_0^\infty \bar{F}_Y(y)dy$ equals the mean $E[Y]$ of a non-negative random variable Y. ∎

This leads to the main result of this section.

Theorem 3: For a fixed download rate λ, constant inter-synchronization delays are optimal under all N_U.

V. Cascaded Replication

We begin this section by introducing our cascaded model and show that freshness at each level can be determined if we know the residual distribution of ON cycles of $\phi(t)$. We then derive a recursive formula for the freshness at each layer i and finish this section with simulations and discussion.

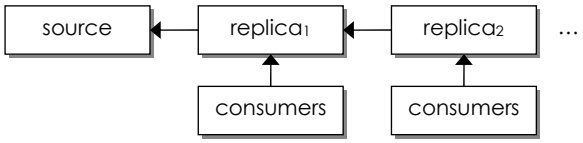


Fig. 5. Cascaded replication at depth two.

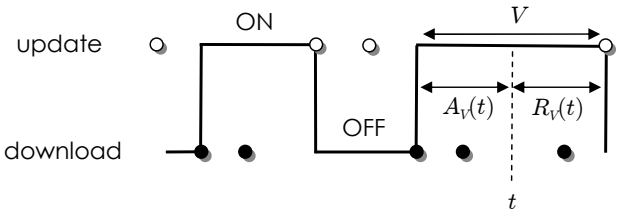


Fig. 6. The age and residual of V in process $\phi(t)$.

A. Objectives

Motivated by the fact that replication trees are a common mechanism to scale the system to a large number of clients, we next consider a *cascaded model* in which caches at level i download content from those at level $i-1$. As before, replicas operate independently of the source and each other. As illustrated in Fig. 5, it suffices to consider a single branch of the tree that starts from the source (at level 0) and traverses towards the leaves.

We assume that each level i of the tree operates using inter-download delays $D^{(i)} \sim F_D^{(i)}(x)$ and has its own freshness process $\phi_i(t)$. This allows nodes near the root to synchronize faster or slower than those near the leaves (e.g., due to bandwidth constraints of the source or other reasons). Our task is to derive the average freshness p_i for queries directed towards replicas at depth i. Unlike (7), which is a simple function of two ages, there is no obvious way (yet) to express how p_i depends on the parameters of the system. The next subsection builds enough results to perform just that.

B. Freshness Residuals

For now, assume the single-layer case. Given the freshness process $\phi(t)$ in Fig. 6, define the ON durations (i.e., periods when $\phi(t)=1$) to be given by some variable V. Note that $\phi(t)$ transitions from OFF to ON upon the first download following an update. It similarly goes from ON to OFF at the first update following a download. At time t, define the age $A_V(t)$ and residual $R_V(t)$ as the backward and forward delays, respectively, to the end of the ON segment.

Our later sections will require the distribution of $R_V(t)$, which is our focus here. Let the residual of the update process at time t be the interval from t to the next update event:

$$R_U(t) = u_{N_U(t)+1} - t. \tag{13}$$

When t lands in an ON cycle, Fig. 6 shows that $R_V(t)$ is the same with $R_U(t)$, which yields:

$$\begin{aligned} P(R_V(t) < x) &= P(R_U(t) < x | \phi(t) = 1) \\ &= P(R_U(t) < x | A_U(t) > A_D(t)). \end{aligned} \tag{14}$$

This is a subtle point, but $R_V(t)$ and $R_U(t)$ have different distributions, unless U is exponential. Conditioning on the ON state of $\phi(t)$ introduces bias into $R_V(t)$, which in certain cases makes it stochastically larger than update residuals $R_U(t)$ and at other times smaller (see below). In order to simplify (14), we need the following lemma.

Lemma 4: Consider a renewal process with interval lengths $X \sim F_X(x)$. As $t \to \infty$, the probability that the age of this process at time t is greater than a and simultaneously the residual is greater than b is:

$$\lim_{t\to\infty} P(A_X(t) > a, R_X(t) > b) = \bar{G}_X(a+b). \tag{15}$$

Armed with Lemma 4, we are ready to obtain the residual distribution of ON durations.

Theorem 4: The residual distribution of V is given by:

$$G_V(x) := P(R_V < x) = 1 - \frac{E[\bar{G}_U(A_D + x)]}{E[\bar{G}_U(A_D)]}. \tag{16}$$

Proof: Re-writing (14) and applying Lemma 4:

$$\begin{aligned} \bar{G}_V(x) &= \lim_{t\to\infty} P(R_U(t) > x | A_U(t) > A_D(t)) \\ &= \frac{1}{p} \lim_{t\to\infty} P(R_U(t) > x, A_U(t) > A_D(t)) \\ &= \frac{1}{p} \int_0^\infty g_D(y) \lim_{t\to\infty} P(R_U(t) > x, A_U(t) > y) dy \\ &= \frac{1}{p} \int_0^\infty g_D(y) \bar{G}_U(x+y) dy. \end{aligned} \tag{17}$$

Recalling (8) and collapsing the integral, we get (16). ■

Fig. 7(a) shows that the model matches simulations very accurately under Pareto U. As mentioned earlier, when the update distribution is exponential, i.e., $F_U(x) = G_U(x) = 1 - e^{-\mu x}$, from (16) we get:

$$G_V(x) = 1 - \frac{E[e^{-\mu(A_D+x)}]}{E[e^{-\mu A_D}]} = 1 - e^{-\mu x}, \tag{18}$$

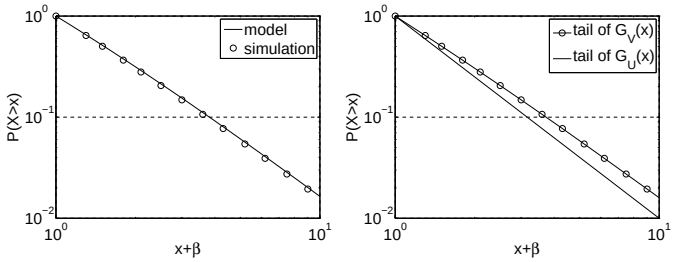
which indicates that $G_V(x)$ remains exponential with the same rate μ. However, this is not true for non-exponential cases. To see this, we plot in Fig. 7(b) the tails of R_U and R_V for Pareto U. Observe in the figure that the latter is more heavy-tailed than the former. In fact, it can be shown that $R_V \geq_{st} R_U$ for NWU update distributions and the opposite for NBU.

Recalling (5)-(6), analysis above allows easy access to the CDF of ON durations:

$$F_V(x) := P(V < x) = 1 - \frac{g_V(x)}{g_V(0)} = 1 - \frac{E[g_U(A_D + x)]}{E[g_U(A_D)]}$$

and the average amount of time the replica stays fresh:

$$E[V] = \frac{1}{g_V(0)} = \frac{E[\bar{G}_U(A_D)]}{E[g_U(A_D)]}. \tag{19}$$



(a) simulation of (16) (b) comparison of tails

Fig. 7. Residual distribution $G_V(x)$ under Pareto U ($\alpha = 3, \beta = 1$) and exponential D ($\lambda = 2$).

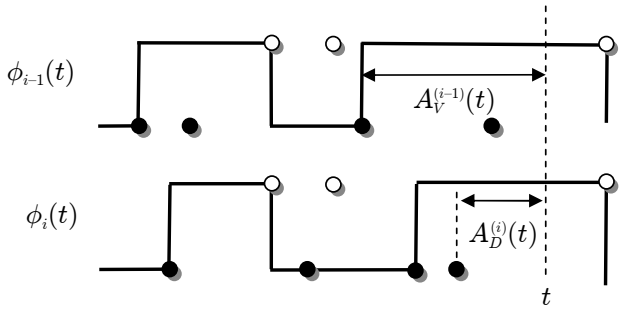


Fig. 8. Processes $\phi_{i-1}(t)$ and $\phi_i(t)$ in cascaded replication.

C. Cascaded Freshness

We now return to the main problem of this section and reactivate usage of sub/super-scripts i to denote the depth of the replica in the tree. As illustrated in Fig. 8, the copy at level i is fresh at time t if and only if the copy at level $i-1$ is fresh and the download age $A_D^{(i)}(t)$ is smaller than the current age $A_V^{(i-1)}$ of the ON duration at level $i-1$:

$$\phi_i(t) = \begin{cases} 1 & A_V^{(i-1)}(t) > A_D^{(i)}(t), \phi_{i-1}(t) = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (20)$$

Our first result allows p_i to be expressed as a function of the residual ON duration within the previous level $i-1$.

Lemma 5: The expected freshness at depth i is given by:

$$p_i = E[r_{i-1}(A_D^{(i)})], \quad (21)$$

where

$$r_i(y) = \lim_{t\to\infty} P(R_V^{(i)}(t) > y, \phi_i(t) = 1). \quad (22)$$

Proof: Using (20) and recalling from renewal theory that $A_V^{(i-1)}(t)$ has the same distribution as $R_V^{(i-1)}(t)$:

$$\begin{aligned} p_i &= \lim_{t\to\infty} P(\phi_i(t) = 1) \\ &= \lim_{t\to\infty} \int_0^\infty P(R_V^{(i-1)}(t) > y, \phi_{i-1}(t) = 1) g_D^{(i)}(y) dy \\ &= \int_0^\infty r_{i-1}(y) g_D^{(i)}(y) dy = E[r_{i-1}(A_D^{(i)})], \end{aligned} \quad (23)$$

where $r_i(y)$ was given earlier in (22). ∎

Our next step is to recursively expand $r_i(y)$.

Lemma 6: For all $i \geq 1$:

$$r_i(y) = E\Big[\bar{G}_U\Big(y + \sum_{k=1}^{i} A_D^{(k)}\Big)\Big]. \quad (24)$$

Proof: First, observe from Fig. 8 that residual ON durations at levels i and $i-1$ are the same as long as the i-th replica is fresh:

$$P(R_V^{(i)}(t) > y, \phi_i(t) = 1) = P(R_V^{(i-1)}(t) > y, \phi_i(t) = 1).$$

On the right-hand side of this result, expanding event $\phi_i(t) = 1$ using (20) and applying Lemma 4, we get:

$$\begin{aligned} &P(R_V^i(t) > y, \phi_i(t) = 1) \\ &= P(R_V^{(i-1)}(t) > y, A_V^{(i-1)}(t) > A_D^{(i)}(t), \phi_{i-1}(t) = 1) \\ &= P(R_V^{(i-1)}(t) > y + A_D^{(i)}(t), \phi_{i-1}(t) = 1). \end{aligned} \quad (25)$$

Letting $t \to \infty$ and closely examining the last equation, notice that it provides a recursive formula on $r_i(y)$:

$$r_i(y) = r_{i-1}(y + A_D^{(i)}) = r_1\Big(y + \sum_{k=2}^{i} A_D^{(k)}\Big), \quad (26)$$

where the last result is obtained by repeatedly expanding $r_{i-1}(.)$. Since $r_1(y) = P(R_V > y)p$, we can combine (16) and (8) to obtain:

$$r_1(y) = E[\bar{G}_U(y + A_D^{(1)})]. \quad (27)$$

Merging (26)-(27), we immediately get (24). ∎

This leads to our main result.

Theorem 5: The probability of freshness at depth i is:

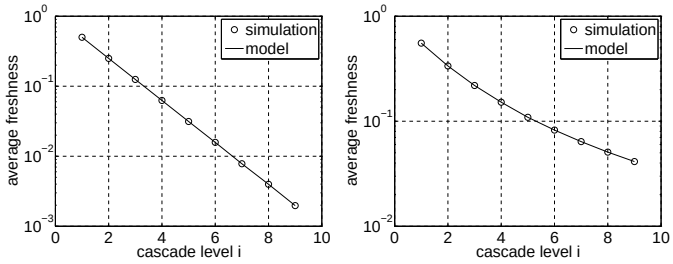
$$p_i = E[\bar{G}_U(Q_i)], \quad (28)$$

where $Q_i = A_D^{(1)} + A_D^{(2)} + \ldots + A_D^{(i)}$.

To perform a self-check, notice that p_1 in (28) reduces to p in (8). For $i \geq 2$, the model is also quite simple – it says that the freshness at level i is given by that of a single-layer system in which the download process N_D operates using intervals D whose age A_D is the summation of ages at levels $1, 2, \ldots, i$. For example, using constant $D^{(i)} = d$, each of the ages $A_D^{(i)}$ is uniform in $[0, d]$. For non-trivial i, their convolution Q_i is approximately Gaussian with mean $id/2$.

D. Discussion

Analyzing (28), first notice that it makes no difference in which order the replicas form the chain – freshness p_i only depends on the summation $Q_i = \sum_{k=1}^{i} A_D^{(k)}$. Therefore, placing high-rate download processes towards the top of the tree and slow towards the bottom produces exactly the same freshness as doing vice versa. Keeping source overload in mind, the best strategy may then be to design trees with high branching factors b (i.e., low depth) and slow λ near the root, placing faster processes towards the leaves.



(a) exponential U (b) Pareto U

Fig. 9. Cascaded freshness with exponential D and $\lambda = \mu = 2$.

When the update process at the source is Poisson, i.e., $G_U = 1 - e^{-\mu x}$, the tail CDF of U decomposes into a product:

$$p_i = \prod_{k=1}^{i} E\Big[\bar{G}_U(A_D^{(k)})\Big] = \prod_{k=1}^{i} p(A_D^{(i)}), \qquad (29)$$

where $p(A_D^{(i)})$ is the freshness probability of $N_D^{(i)}$ working directly with the source. Therefore, if we know that replicas A and B separately achieve freshness p_A and p_B, their cascaded performance will produce freshness $p_A p_B$ at level 2. If additionally the download processes are all homogeneous, i.e., $F_D^{(i)}(x) = F_D^{(j)}(x)$ for all (i, j, x), the freshness value p_i is an exponentiation of the single-step model (8).

Note that multiplicative reduction in p_i as a function of i presents an interesting tradeoff – as the tree size scales exponentially up, freshness scales exponentially down. In order to prevent $p_i \to 0$, one must increase the download rate λ_i at depth i such that $\sum_{i=1}^{\infty} \log p(A_D^{(i)}) > -\infty$. One of the slowest growing functions that satisfies this condition is $p(A_D^{(i)}) = 1-(i+1)^{-\rho}$, where ρ is slightly larger than 1. Using exponential D as an example, we get $p(A_D^{(i)}) = \lambda_i/(\lambda_i + \mu)$, which translates into $\lambda_i = \mu((i+1)^{\rho} - 1)$, showing that bandwidth requirements *must scale super-linearly, but at least not exponentially*, with i.

We next use simulations with homogeneous download delays to compare the decay rate in p_i for exponential and Pareto U. Fig. 9 shows that model (28) is quite accurate and that the Pareto case in (b) decreases much slower than the exponential in (a). This suggests that NWU distributions of U are easier to scale than either exponential or NBU. This can be confirmed by noticing that the heavier the tail $\bar{G}_U(x)$, the slower p_i decays in (28).

VI. Cooperative Caching

In this section, we consider a novel cooperative model in which all replicas are at level 1, but they are allowed to communicate with each other. We first introduce the details of this configuration and the corresponding notation. We follow that up with analyzing parameter selection and formulating optimality conditions that lead to best freshness.

A. Model and Notation

As shown in Fig. 10, suppose there are m replicas in the system that form a cluster. As before, $N_D^{(i)}$ is the download process between each replica i and the source; however, we now additionally assume that $N_C^{(i)}$ represents the *communication* process that each replica uses to poll other nodes within the cluster. At each point of $N_C^{(i)}$, the node contacts k other peers and selects the freshest response for download. All processes are renewal and independent of each other. Let $C \sim F_C(x)$ describe the length of cycles in each communication process and $\nu = 1/E[C]$ be its rate.

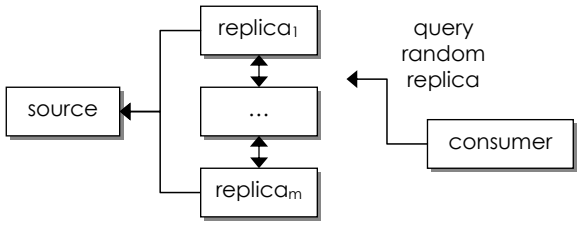


Fig. 10. Cooperative replication.

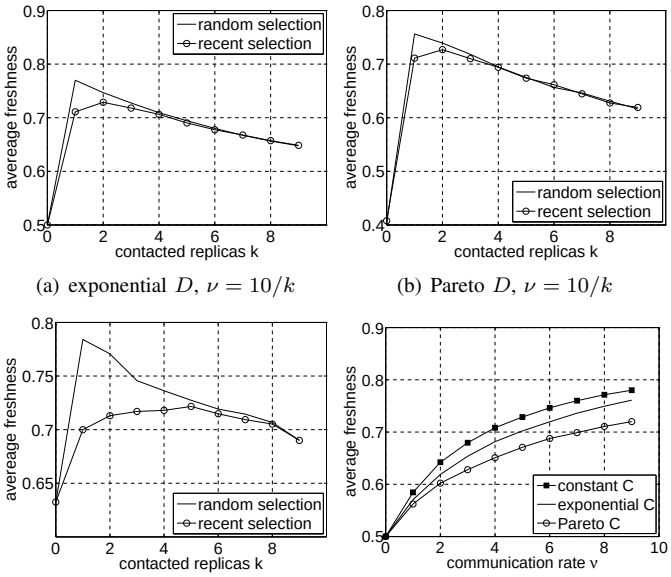
In order for a replica to cooperate with others, it has to know their location. One option is to require that the source maintain a replica list ordered by the most-recent download timestamp. This list can then be disseminated to replicas upon each contact with the source. In order to choose k peers among m, we consider two strategies: *random* and *recent*. The former selects k peers in m uniformly randomly (assuming global knowledge or other mechanisms) and the latter selects those with the largest contact timestamps.

B. Simple Scenario

With a fixed per-node intra-cluster communication bandwidth $k\nu$, the first question is how to choose k and ν such that they provide the highest freshness. With cooperation, closed-form derivations are difficult because downloads follow random cascaded chains (i.e., $s \to x_1 \to x_2 \to \dots$ where s signifies the source and x_k the k-th replica ID along this chain). We therefore use simulations to study this problem.

For random selection, observe from Fig. 11(a)-(c) that the expected freshness p achieves a global maximum at $k = 1$, which is noticeably higher than in non-cooperative cases (i.e., $k = 0$). Freshness then monotonically decreases as the number of contacted nodes k increases. This can be explained by the fact that the replicas' freshness processes $\{\phi_i(t)\}_{t\geq 0}$ are highly correlated, i.e., an update at time t makes all of them transition from ON to OFF. Thus, the benefit of contacting $k \geq 2$ peers at lower rate ν is smaller than contacting one peer with higher ν.

Interestingly, recent selection in Fig. 11(a)-(c) peaks at later points $k \geq 2$, but then succumbs to the same effect. In fact, as k gets larger, the most-recent list becomes essentially composed of random nodes and both methods converge. In all studied cases, random selection beats recent. The intuition is that the latter is biased towards certain peers that were the most up-to-date at time d_k, but are no longer the freshest by the next download instance d_{k+1}. Instead, results show that a random peer has fresher information when we average over the entire interval $[d_k, d_{k+1}]$. This indicates that *the optimal*



(a) exponential D, $\nu = 10/k$ (b) Pareto D, $\nu = 10/k$

(c) constant D, $\nu = 10/k$ (d) random selection $k = 1$

Fig. 11. Effect of k and ν in cooperative replication ($\mu = \lambda = 1, m = 10$). Exponential U and C.

strategy is to choose exactly one uniformly random peer (i.e., $k = 1$) and hence keep ν as large as possible.

C. Convergence of Freshness

We now fix $k = 1$ and vary ν in Fig. 11(d) to analyze the effect of C and ν. As before, NBU synchronization performs the best, followed by exponential and NWU. Reasoning similar to that used in previous sections suggests that constant C remains optimal for cooperative caching. In the figure, p starts at 0.5 and gradually improves to $0.72-0.78$ depending on the distribution of C; however, what happens to freshness as intra-cluster bandwidth ν becomes very large? We address this next.

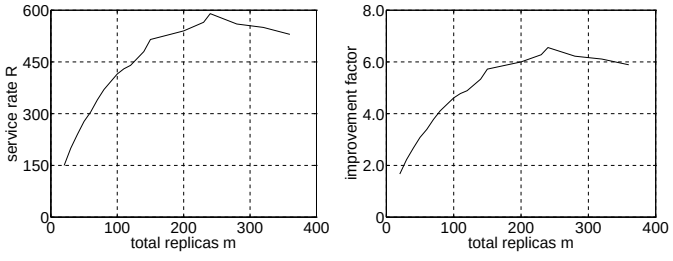
Theorem 6: As $\nu \to \infty$, freshness under random selection equals that computed using the original update process N_U and a superposition N_D^* of m download processes $\{N_D^{(i)}\}_{i=1}^m$.

Note that this result holds for all $k \geq 1$. We are now able to compute the limiting freshness probability for the case in Fig. 11(d) with exponential C. Since a superposition of Poisson process remains Poisson, we get that $p_\infty = m\lambda/(m\lambda + \mu) = 0.91$. As an alternative to raising ν, freshness can be increased by allowing *bidirectional* communication between replicas. When A requests updates from B, it can offer its own content for upload to B. This effectively doubles rate ν in our model, but keeps it fully applicable to this technique as well.

D. Full System

We now consider a more realistic cooperative replication system that has two conflicting goals – serving clients and maintaining freshness. The tradeoff arises from bandwidth consumption – larger ν leads to higher freshness, but reduces the ability of each node to answer consumer queries.

Let s be the service rate that each replica can offer to its clients and suppose that all peers have the same bandwidth



(a) service rate R (b) service rate ratio

Fig. 12. Effect of m on service rate R ($\epsilon = 0.5, \mu = 1, B = 10$). All distributions are exponential.

constraint B (including the source). The system is considered usable if the average freshness is no smaller than $1 - \epsilon$, where $\epsilon > 0$ is some design parameter. Define $p(\lambda, k, \nu)$ to be the freshness probability achieved by a cluster using the source download rate λ, k-way cooperation, and intra-cluster synchronization rate ν. Then, the objective is to maximize the combined service rate $R := ms = m(B - \lambda - k\nu)$ subject to:

$$\begin{cases} m\lambda \leq B \\ p(\lambda, k, \nu) \geq 1 - \epsilon \end{cases}. \tag{30}$$

Since all peers are homogenous, the source bandwidth should be shared equally, which means that the first line of (30) reduces to $m = B/\lambda$. For non-cooperative replication, both k and ν are zero and thus $R = B^2/\lambda - B$. The only unknown parameter is λ, which can be determined from the freshness condition of (30) as $\lambda = q^{-1}(1-\epsilon)$, where $q(x) = p(x, 0, 0)$. For example, with exponential U and D:

$$q(x) = \frac{x}{x + \mu} = 1 - \epsilon, \tag{31}$$

from which we get:

$$\lambda = \frac{\mu(1-\epsilon)}{\epsilon} \quad \text{and} \quad R = \frac{B^2\epsilon}{\mu(1-\epsilon)} - B. \tag{32}$$

For cooperative replication, the previous subsection shows that the optimal case is $k = 1$. Now suppose we fix m. This allows us to determine the remaining parameters of the system using the model above. Indeed, $\lambda = B/m$ and $\nu = q_2^{-1}(1-\epsilon)$, where $q_2(x) = p(\lambda, 1, x)$. To understand why there is an inherent tradeoff in this system, notice that larger m affords the system more combined bandwidth $R = m(B - \lambda - k\nu)$; however, this also leads to lower source-synchronization rate $\lambda = B/m$, which decreases freshness. To compensate for lower freshness, replicas must communicate with other peers at a higher rate ν, which in turn lowers R.

Thus, there must be an optimal m that maximizes the service rate for a given set (ϵ, B, N_U). To demonstrate this, we plot simulation results in Fig. 12(a), where R hits a peak at $m = 240$ and then drops monotonically. The improvement ratio between cooperative and non-cooperative R in Fig. 12(b) reaches 4 by $m = 90$ and 6 by $m = 200$. Table I shows additional scenarios. Observe that cooperative replication is more effective when the target freshness is smaller. The benefit

TABLE I
OPTIMAL SERVICE RATE COMPARISON BETWEEN COOPERATIVE AND NON-COOPERATIVE REPLICATION

$1-\epsilon$	Non-cooperative R	Cooperative R	Ratio
0.4	270	1865	6.9
0.5	90	590	6.6
0.6	50	134	2.7
0.7	33	42	1.3

decreases as $\epsilon \to 0$ since it becomes progressively more difficult to find peers with exceedingly fresh copies.

VII. REDUNDANT QUERYING

To obtain fresher results, existing work [1] suggests that consumers contact multiple replicas. As illustrated in Fig. 13, this model uses a single-layer non-cooperative caching with redundant queries to achieve higher robustness against staleness. Contacting $k \geq 2$ random replicas and retrieving the freshest copy, consumers effectively replace a single download process N_D with a superposition N_D^* of k processes $\{N_D^{(i)}\}_{i=1}^k$. This observation is similar to Theorem 6 except the number of superposed processes is k rather than m.

A. Analysis

As before, suppose $1-\epsilon$ is the target freshness at the user and $p(\lambda)$ is that achieved by a non-redundant system (i.e., using $k=1$). Then, recalling the previous section, there is a unique download rate $\lambda = p^{-1}(1-\epsilon)$ that determines the optimal cluster size $m = B/\lambda$ and the combined service rate:

$$R = m(B-\lambda) = \frac{B^2}{\lambda} - B. \tag{33}$$

We now contrast this with a redundant configuration. Define λ' to be the download rate of each replica and $\lambda^* = k\lambda'$ to be that of the superposed process N_D^*. Since freshness is now determined by $\lambda^* = p_*^{-1}(1-\epsilon)$, where $p_*(x)$ is the freshness probability under N_D^*, we can lower each λ' and hopefully increase system size $m^* = B/\lambda' = Bk/\lambda^*$ beyond m. However, the main caveat is that each replica now serves k times more traffic to the clients, which means that the best possible query rate of the new system is:

$$R^* = \frac{m^*(B-\lambda')}{k} = \frac{B^2}{\lambda^*} - \frac{B}{k}. \tag{34}$$

B. Discussion

In the simplest case of exponential D, both N_D and N_D^* are Poisson, which immediately leads to $\lambda^* = \lambda$ since functions $p(x)$ and $p_*(x)$ are the same. This shows that the redundant system can support exactly $m^*/m = k$ times more servers, but the service-rate ratio R^*/R stays pretty close to 1, i.e., there is virtually no benefit. To tackle more complex cases, assume k is sufficiently large, in which case the superposition process N_D^* tends to Poisson from the Palm-Khintchine theorem [40]. From (32), we know that $\lambda^* = \mu(1-\epsilon)/\epsilon$, which leads to a closed-form service rate:

$$R^* = \frac{B^2\epsilon}{\mu(1-\epsilon)} - \frac{B}{k}. \tag{35}$$

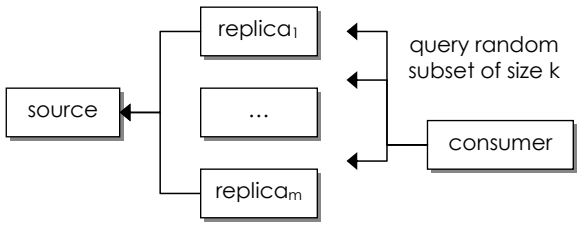


Fig. 13. Redundant querying with $k = 3$.

TABLE II
IMPROVEMENT FROM REDUNDANT QUERYING COMPARED TO NON-REDUNDANT

$1-\epsilon$	Pareto D		exponential D		constant D	
	m^*/m	R^*/R	m^*/m	R^*/R	m^*/m	R^*/R
0.1	21.8	1.1	20	1	18.0	0.90
0.3	26.6	1.3	20	1	14.6	0.73
0.5	32.8	1.6	20	1	12.5	0.63
0.7	42.3	2.1	20	1	11.2	0.56

If D has an NWU distribution, k-way aggregation makes D^* stochastically larger in second order and thus improves freshness (see the discussion in Section IV-B). Counter-intuitively, however, if D has an NBU distribution, transition to a Poisson N_D^* *makes the redundant case perform worse than the non-redundant*. These effects are shown in Table II for $k = 20$ and $B = 1000$. As target freshness increases, the Pareto case gradually improves and finishes with rates 110% above those in the non-redundant scenario. The opposite trend occurs with constant D, which gradually becomes worse and ends up losing 44% of service capacity in the last row.

VIII. CONCLUSION

We proposed a general framework for modeling lazy synchronization and derived the probability of freshness under general update/download processes. We then extended these results to cascaded and cooperative replication, finding solutions to a number of optimization problems in those contexts. Finally, we examined redundant querying and found cases when doing so was detrimental to system performance.

Future work involves combining the various studied techniques (i.e., cascading, cooperation, and redundancy) into a single architecture that can bring together the benefits of its individual pieces to create a more scalable system.

REFERENCES

[1] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, "Probabilistically Bounded Staleness for Practical Partial Quorums," *VLDB Endow.*, vol. 5, no. 8, pp. 776–787, Apr. 2012.
[2] B. E. Brewington and G. Cybenko, "How Dynamic is the Web," *Computer Networks*, no. 1-6, pp. 257–276, Jun. 2000.
[3] L. Bright, A. Gal, and L. Raschid, "Adaptive Pull-based Policies for Wide Area Data Delivery," *ACM Trans. Database Syst.*, vol. 31, no. 2, pp. 631–671, Jun. 2006.
[4] D. Carney, S. Lee, and S. Zdonik, "Scalable Application-Aware Data Freshening," in *Proc. IEEE ICDE*, Mar. 2003, pp. 481–492.
[5] M. Castro, P. Druschel, A. M. Kermarrec, and A. I. Rowstron, "Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure," *IEEE JSAC*, vol. 20, no. 8, pp. 1489–1499, Sep. 2006.

[6] B. Chandramouli and J. Yang, "End-to-end Support for Joins in Large-scale Publish/Subscribe Systems," *VLDB Endow.*, vol. 1, no. 1, pp. 434–450, Aug. 2008.

[7] X. Chen, S. Ren, H. Wang, and X. Zhang, "Scope: Scalable Consistency Maintenance in Structured P2P Systems," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 1502–1513.

[8] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler," in *Proc. VLDB*, Sep. 2000, pp. 200–209.

[9] J. Cho and H. Garcia-Molina, "Synchronizing a Database to Improve Freshness," in *Proc. ACM SIGMOD*, May 2000, pp. 117–128.

[10] J. Cho and H. Garcia-Molina, "Estimating Frequency of Change," *ACM Trans. Internet Technol.*, vol. 3, no. 3, pp. 256–290, Aug. 2003.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability*, Jul. 2000, pp. 46–66.

[12] E. G. Coffman, Z. Liu, and R. R. Weber, "Optimal Robot Scheduling for Web Search Engines," *Journal of Scheduling*, no. 1, pp. 15–29, Jun. 1998.

[13] E. Cohen and H. Kaplan, "The Age Penalty and Its Effect on Cache Performance," in *Proc. USENIX USITS*, Mar. 2001, pp. 73–84.

[14] E. Cohen and H. Kaplan, "Aging Through Cascaded Caches: Performance Issues in the Distribution of Web Content," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 41–53.

[15] E. Cohen and H. Kaplan, "Refreshment Policies for Web Content Caches," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1398–1406.

[16] R. Cox, A. Muthitacharoen, and R. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," in *Proc. IPTPS*, Mar. 2002, pp. 155–165.

[17] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area Cooperative Storage with CFS," in *Proc. ACM SOSP*, Oct. 2001, pp. 188–201.

[18] A. Datta, M. Hauswirth, and K. Aberer, "Updates in Highly Unreliable, Replicated Peer-to-Peer Systems," in *Proc. IEEE ICDCS*, May 2003, pp. 76–85.

[19] D. Denev, A. Mazeika, M. Spaniol, and G. Weikum, "SHARC: Framework for Quality-Conscious Web Archiving," in *Proc. VLDB*, Aug. 2009, pp. 183–207.

[20] J. V. Deshpande, S. C. Kochar, and H. Singh, "Aspects of Positive Ageing," *J. Applied Probability*, vol. 23, no. 3, pp. 748–758, Sep. 1986.

[21] D. Dey, Z. Zhang, and P. De, "Optimal Synchronization Policies for Data Warehouses," *INFORMS J. on Computing*, no. 2, pp. 229–242, Jan. 2006.

[22] Y. Diao, S. Rizvi, and M. J. Franklin, "Towards an Internet-Scale XML Dissemination Service," in *Proc. VLDB*, Aug. 2004, pp. 612–623.

[23] J. Eckstein, A. Gal, and S. Reiner, "Monitoring an Information Source Under a Politeness Constraint," *INFORMS J. on Computing*, no. 1, pp. 3–20, Jan. 2008.

[24] M. Fontoura, M. Ionescu, and N. Minsky, "Law-Governed Peer-to-peer Auctions," in *Proc. WWW*, May 2002, pp. 109–116.

[25] A. Gal and J. Eckstein, "Managing Periodically Updated Data in Relational Databases: A Stochastic Modeling Approach," *J. ACM*, no. 6, pp. 1141–1183, Nov. 2001.

[26] Gnutella. [Online]. Available: http://en.wikipedia.org/wiki/Gnutella.

[27] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher, "Peer-to-Peer Caching Schemes to Address Flash Crowds," in *Proc. IEEE ICDCS*, Mar. 2004, pp. 360–369.

[28] Y. Hu, M. Feng, and L. N. Bhuyan, "A Balanced Consistency Maintenance Protocol for Structured P2P Systems," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 286–290.

[29] Y. Huang, R. H. Sloan, and O. Wolfson, "Divergence Caching in Client-Server Architectures," in *Proc. IEEE PDIS*, Sep. 1994, pp. 131–139.

[30] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A Decentralized Peer-to-peer Web Cache," in *Proc. IEEE PODC*, Jul. 2002, pp. 213–222.

[31] R. Ladin, B. Liskov, L. Shrira, and S. Ghemawat, "Providing High Availability Using Lazy Replication," *ACM Trans. Comput. Syst.*, no. 4, pp. 360–391, Nov. 1992.

[32] J.-J. Lee, K.-Y. Whang, B. S. Lee, and J.-W. Chang, "An Update-Risk Based Approach to TTL Estimation in Web Caching," in *Proc. IEEE WISE*, Dec. 2002, pp. 21–29.

[33] Z. Li, G. Xie, and Z. Li, "Efficient and Scalable Consistency Maintenance for Heterogeneous Peer-to-Peer Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 12, pp. 1695–1708, Dec. 2008.

[34] Y. Ling and W. Chen, "Measuring Cache Freshness by Additive Age," *SIGOPS Oper. Syst. Rev.*, vol. 38, pp. 12–17, Jul. 2004.

[35] Y. Ling and J. Mi, "An Optimal Trade-off between Content Freshness and Refresh Cost," *Applied Probability*, vol. 41, no. 3, pp. 721–734, Sep. 2004.

[36] X. Liu, J. Lan, P. Shenoy, and K. Ramaritham, "Consistency Maintenance in Dynamic Peer-to-peer Overlay Networks," *Comput. Netw.*, vol. 50, no. 6, pp. 859–876, Apr. 2006.

[37] N. Matloff, "Estimation of Internet File-access/Modification Rates from Indirect Data," *ACM Trans. Model. Comput. Simul.*, vol. 15, pp. 233–253, Jul. 2005.

[38] C. Olston and J. Widom, "Best-Effort Cache Synchronization With Source Cooperation," in *Proc. ACM SIGMOD*, May 2002, pp. 73–84.

[39] C. Olston and S. Pandey, "Recrawl Scheduling Based on Information Longevity," in *Proc. WWW*, Apr. 2008, pp. 437–446.

[40] K. Palm, "Intensitätsschwankungen im Fernsprechverkehr," *Ericsson Technics*, vol. 44, 1943.

[41] S. Pandey, K. Dhamdhere, and C. Olston, "WIC: A General-Purpose Algorithm for Monitoring Web Information Sources," in *Proc. VLDB*, Aug. 2004, pp. 360–371.

[42] S. Pandey, K. Ramamritham, and S. Chakrabarti, "Monitoring the Dynamic Web to Respond to Continuous Queries," in *Proc. WWW*, May 2003, pp. 659–668.

[43] M. Roussopoulos and M. Baker, "CUP: Controlled Update Propagation in Peer-to-Peer Networks," in *Proc. USENIX Annual Technical Conference*, Apr. 2003, pp. 167–180.

[44] A. Rowstron and A. Rowstron, "Storage Management and Caching in PAST, A Large-Scale, Persistent Peer-to-Peer Storage Utility," in *Proc. ACM SOSP*, Nov. 2001, pp. 188–201.

[45] H. Shen, "IRM: Integrated File Replication and Consistency Maintenance in P2P Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 100–113, Jan. 2012.

[46] H. Shen and G. Liu, "A Geographically Aware Poll-Based Distributed File Consistency Maintenance Method for P2P Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2148–2159, Nov. 2013.

[47] H. Shen and G. Liu, "A Lightweight and Cooperative Multifactor Considered File Replication Method in Structured P2P Systems," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2115–2130, Jan. 2013.

[48] K. C. Sia and J. Cho, "Efficient Monitoring Algorithm for Fast News Alerts," *IEEE Trans. Knowl. Data Eng.*, no. 7, pp. 950–961, Jul. 2007.

[49] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan, "Feeding Frenzy: Selectively Materializing Users' Event Feeds," in *Proc. ACM SIGMOD*, Jun. 2010, pp. 831–842.

[50] T. Stading, P. Maniatis, and M. Baker, "Peer-to-Peer Caching Schemes to Address Flash Crowds," in *Proc. IPTPS*, Mar. 2002, pp. 203–213.

[51] Q. Tan and P. Mitra, "Clustering-based Incremental Web Crawling," *ACM Transactions on Information Systems*, no. 4, pp. 1–27, Nov. 2010.

[52] X. Tang, J. Xu, and W.-C. Lee, "Analysis of TTL-Based Consistency in Unstructured Peer-to-Peer Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 12, pp. 1683–1694, Feb. 2008.

[53] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen, "Optimal Crawling Strategies for Web Search Engines," in *Proc. WWW*, May 2002, pp. 136–147.

[54] R. W. Wolff, *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[55] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming Botnets: Signatures and Characteristics," in *Proc. ACM SIGCOMM*, Sep. 2006, pp. 171–182.

[56] A. Yahyavi and B. Kemme, "Peer-to-peer Architectures for Massively Multiplayer Online Games: A Survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 9:1–9:51, Oct. 2013.

[57] M. Yang, H. Wang, L. Lim, and M. Wang, "Optimizing Content Freshness of Relations Extracted From the Web Using Keyword Search," in *Proc. ACM SIGMOD*, Jun. 2010, pp. 819–830.

[58] L. Yin and G. Cao, "DUP: Dynamic-tree Based Update Propagation in Peer-to-Peer Networks," in *Proc. IEEE ICDE*, Apr. 2005, pp. 258–259.

[59] H. Yu and A. Vahdat, "Design and Evaluation of a Continuous Consistency Model for Replicated Services," in *Proc. USENIX OSDI*, Jun. 2000, pp. 305–318.

[60] J. Zhu, J. Gong, W. Liu, T. Song, and J. Zhang, "A Collaborative Virtual Geographic Environment Based on P2P and Grid Technologies," *Inf. Sci.*, vol. 177, no. 21, pp. 4621–4633, Nov. 2007.