Scheduling of Data Transcription in Periodically Connected Databases

Avigdor Gal,* Jonathan Eckstein, Zachary G. Stoumbos†
Department of Management Science and Information Systems
Rutgers University, Piscataway, NJ 08854-8054 USA
{avigal@business, jeckstei@rutcor, stoumbos@andromeda}.rutgers.edu

Abstract

Contemporary data management in applications such as pervasive systems, Web-based supply chain management, data warehouses, and Web crawlers involve the periodic transcription of data onto secondary devices in a networked environment. In this paper, we focus on the scheduling of periodic data transcription in append-only environments, such as e-mail inboxes, newsgroups, technical support bulletin boards, or procurement requests. If the client connects to the server too frequently, the client will nearly always have up-to-date information, but the usage of network resources may be excessive. Conversely, very infrequent connections will conserve network resources, but the client's data may often be significantly out of date, which may also be costly (in terms of lost opportunities, for example). Thus, the best transcription policies should make an optimal trade-off between these costs. Our approach to evaluating this trade-off is to use modeling techniques from the field of stochastic processes. The paper presents a general model for data insertions on the server side, using compound nonhomogeneous Poisson processes, and compares several transcription policies in terms of both transcription cost and obsolescence cost. The comparisons use a validation data set from a real data feed, and our models were calibrated using a separate training set from the same feed. We find that transcription policies based on a nonhomogeneous Poisson arrival model often outperform simpler policies.

Keywords: Poisson models, information systems applications.

1 Introduction

Contemporary data management involves the transcription of data onto secondary devices in a networked environment. Such an environment may require periodic (rather than continuous) synchronization between the database and secondary copies, either due to paucity of resources (*e.g.*, low bandwidth or limited night windows) or the transient characteristics of the connection. Hence, the consistency of the information in secondary copies, with respect to the transcription origin, varies over time and depends on the rate of change of the base data and on the frequency of synchronization. Applications calling for periodic transcription include pervasive systems (*e.g.*,

*Currently on leave from Rutgers, serving as a Senior Lecturer at the Technion, Israel Institute of Technology.

†Zachary G. Stoumbos' work was funded in part by the Law School Admission Council (LSAC) and by a 2001 Rutgers Faculty of Management Research Fellowship. The opinions and conclusions contained in this publication are those of the authors and do not necessarily reflect the position or policy of LSAC.

Microsoft's Mobile Information Server[1] and Café Central[2]), Web-based supply chain management (*e.g.*, virtual retailers), data warehouses, and Web crawlers. For example, consider an online retailer who performs transactions based on inventory information it receives from a distributor. Although the retailer is interested in the most current inventory level, it may become prohibitive to query the distributor inventory database for each inquiry the retailer's customers perform, either due to heavy transaction loads or charges the distributor may impose for accessing its databases. Therefore, periodic transcription of inventory levels onto the retailer's system (*e.g.*, whenever the inventory level of a product becomes low) may be a more desirable solution.

In this paper, we focus on scheduling data transcription in append-only environments, such as e-mail inboxes, newsgroups, technical support bulletin boards, or procurement requests. In such a setting, a server continuously monitors the arrival of new data and a client (*e.g.*, a "road warrior" or a Web crawler) connects to the server at will to transcribe the new data.[3]

If the client connects to the server too frequently, the client will nearly always have up-to-date information, but the usage of network resources may be excessive, and many (or perhaps most) of the connection sessions may find no new information at the server. Conversely, excessively infrequent connections will conserve network resources, but the client's data will be significantly out of date, which may also be costly (in terms of lost opportunities, for example). Thus, the best transcription policies should make some kind of optimal trade-off between these costs. Our approach to evaluating this trade-off is to use modeling techniques from the field of stochastic processes. The paper presents a general model for data insertions on the server side, using compound, nonhomogeneous Poisson processes [37, 45]. Such models capture changes in the rate of insertions over time, *e.g.*, the arrival of new tuples during weekdays as opposed to weekends. It also captures "bulk insertions," in which several new items may arrive simultaneously at a server. This model, while more complex than other models suggested before it (*e.g.*, [11]), is still tractable, and therefore offers a practical tool for decision making.

We present experiments with real data feed to test the usefulness of two special cases of the general model, namely a homogeneous Poisson process and a special case of the compound nonhomogeneous Poisson process. To do so, we present a simple cost model that captures the trade-off between the cost of transcription and the cost of obsolescence. The former is defined using traditional methods in distributed databases, considering each connection's setup cost (which may be substantial) and the latency of transmitting the data. The latter is defined in terms of the time elapsing between arrival of each tuple and its transcription. We acknowledge that obsolescence is subjective, and therefore the obsolescence cost model we have selected reflects user preferences as well. Our findings show that the homogeneous Poisson model, although simple, is limited in the applications it can support. The compound nonhomogeneous Poisson model is far more useful for modeling complex applications and provides a reasonable underlying model to determine the scheduling of transcriptions.

The novelty of this paper is twofold. First, we propose the use of a stochastic model for analyzing data arrivals in append-only environments. We also provide transcription policies, based on the stochastic model, that outperform existing policies. The stochastic model is an early part of an extended project which aims to develop a formal framework for modeling content evolution

[1] http://www.microsoft.com/servers/miserver/

[2] http://www.comalex.com/central.htm

[3] While newsgroups are periodically truncated for space reasons, we assume the clients connect frequently enough to be able to review all newly-arrived messages before they are deleted.

in periodically connected databases subject to tuple insertions, deletions, and modifications. The initial theoretical ideas behind this project are laid out in [17]. This paper validates and applies a portion of this framework empirically.

The rest of the paper is organized as follows. As a background for presenting the model, we discuss related work in Section 2. Section 3 presents the insertion model and some of its simpler derivatives. A simple cost model and a comparison of four transcription policies are given in Section 4. Our experiments and their outcomes are discussed throughout the paper. In particular, we perform a comparative model-fitting validation in Section 3.2. Section 5 provides several possible applications of the proposed model. We conclude with a discussion of further research in Section 6.

2 Related work

The problem of content evolution with respect to materialized views (which may be viewed as a complex form of data transcription) in databases has already been recognized. For example, in [1], the incompleteness of data in views was noted as being a "dynamic notion since data may be constantly added/removed from the view." Yet, we believe that there has been no prior formal modeling of the evolution process.[4] Related research involves the containment property of a materialized view with respect to its base data: a few of the many references in this area include [46, 10, 26, 2, 18]. However, the temporal aspects of content evolution have not been systematically addressed in these works.

Refresh policies for replications and materialized views have been previously discussed in the literature (*e.g.*, [27], [34] and [12]). Typically, materialized views are refreshed using one of the following three policies: immediately upon updates to the base data, at query time (as in [12]), or using snapshot databases (as in [27]). The latter approach, as well as asynchronous replication mechanisms can produce obsolescent data. A combination of all three policies appears in [13]. Our methodology differs in that we do not assume an *a priori* association of a materialized view with a refresh policy, but instead design policies based on their transcription and obsolescence costs (see Section 4).

Several works, including [11, 25], have suggested the use of a Poisson model to model an update process. A preliminary attempt to describe the time dependency of updates in the context of Web management was given in [11], which suggests a simple homogeneous Poisson process to model the updating of Web pages. We suggest instead a nonhomogeneous compound Poisson model, which is far more flexible, and yet still tractable. In addition, the work in [11] supposes that transcriptions are performed at uniform time intervals, mainly because "crawlers cannot guess the best time to visit each site." We show in this paper that our model of content evolution gives rise to other, better transcription policies. In [25], the authors have suggested tuning K, a shifting window that consists of the number of most recent updates to be utilized in computing the arrival rate λ. This approach may capture local λ values, as they vary over time, yet it is more error-prone at the borderline, where λ either sharply increases or decreases. The technique suggested in [25] is identical to the First Arrival policy suggested in Section 4.2.

Recent research in the application area of cache management considers the trade-off between

[4] Orthogonal research efforts involve probabilistic database systems (*e.g.*, [24]), yet probabilistic databases are concerned with uncertainty in the stored data, rather than data evolution.

retrieval time and obsolescence [6]. The proposed trade-off is based on predictions based on the last update of an object, which is a far simpler model than what is proposed herein.

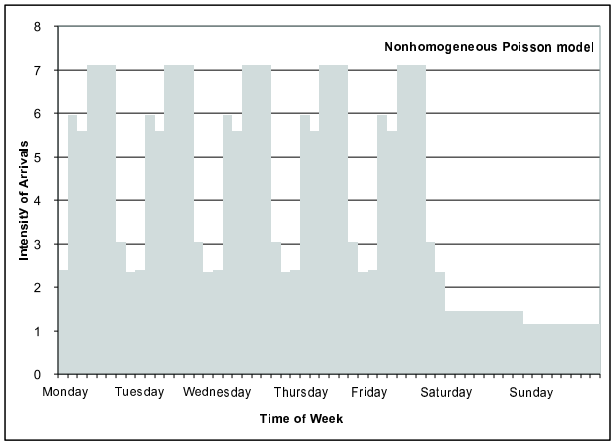
Some related work has dealt with push-based protocols, according to which the server is responsible for "pushing" changes to the client. While catering to user's needs (see for example [9]), push-based technologies may increase the computational load on the server. For example, in [3], it was demonstrated that pull-based protocols can provide significant performance improvement in a broadcast environment, yet unconstrained use by the client can result in scalability problems due to server saturation. Our modeling approach is applicable when a pull-based protocol is required and avoids excessive resource consumption by spacing transcriptions according to the pattern of data arrival.

In [33], a trade-off mechanism was suggested to decide between the use of a cache or recomputation from base data by using range data, computed at the source. While the discussion in [33] revolves around data updates, rather than newly inserted data, the framework has some relevance to our setting. In this framework, an update is "pushed" to a replication site whenever updated data exceeds a predetermined interval of values or whenever a query requires current data. The former requires the client and the server to be in touch continuously, in case the server needs to track down the client, which is not always realistic (either because the server does not provide such services, or because the overhead for such services undermines the cost-effectiveness of the client). The latter requirement puts the burden of deciding whether to refresh the data on the client, without providing her with any model for the evolution of the base data. As discussed in [17], we attempt to fill this gap by providing a stochastic model for content evolution, which allows a client to make judicious requests for current data. It is worth noting that the pull mechanism we suggest in this paper does not require the user's involvement. On the contrary; Transcription decisions are evaluated automatically, based on predefined user's preferences. Other works in related areas (*e.g.*, [4], [8], [14]) have considered various alternatives for pushing updated data from a server to cached data on the client side. Lazy replica-update policies using replication graphs have also been discussed in, for example, [5]. This work, however, does not consider data obsolescence and is primarily concerned with transaction throughput and timely updates, subject to network constraints.

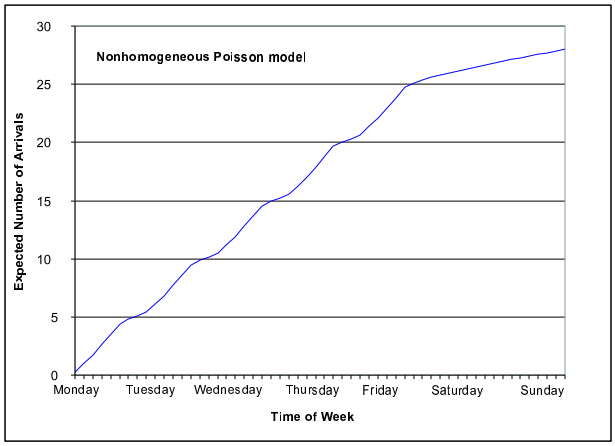
3 The data model

Let the sequence $\{(b_i, e_i)\}_{i=1}^{\infty}$ represent an infinite sequence of connectivity periods between a client and a server. During session i, the client data is synchronized with the state of the server at time b_i, the information becoming available to the client at time e_i. At the next session, beginning at time b_{i+1}, the client is updated with all the information arriving at the server during the interval $(b_i, b_{i+1}]$, which becomes usable at time e_{i+1}, and so forth. We define $b_0 = e_0 = 0$, and require that $0 < b_1 \leq e_1 < b_2 \leq e_2 < \ldots$.

At any time, the client data may be obsolescent [16], that is, out of synchronization with the server. Our ultimate concern is scheduling the connection times $\{b_i\}$. To do so, however, we contend that it is helpful to model the arrival (or insertion) process of data at the server first. Section 3.1 provides the theoretical foundation for the insertion model. In Section 3.2, we test four versions of this model against our test data set, and show that the proposed model, while having a low computational complexity, can fit the data well.



(a)



(b)

Figure 1: Nonhomogeneous Poisson model.

3.1 Modeling insertions

We use a nonhomogeneous Poisson process [36, 45] with instantaneous arrival rate $\lambda : \Re \rightarrow [0, \infty)$ to model the occurrence of *insertion events*. That is, the number of insertion events occurring in any interval $(s, f]$ is a Poisson random variable with expected value $\Lambda(s, f) = \int_s^f \lambda(t)\, dt$. A homogeneous Poisson process may be considered as the special case where $\lambda(t)$ is equal to a constant $\lambda > 0$ for all t, yielding $\Lambda(s, f) = \int_s^f \lambda(t)\, dt = \int_s^f \lambda\, dt = \lambda \cdot (f - s)$.

To illustrate the concept of non-homogeneity, consider Figure 2 and Figure 1. Figure 1(a) shows the changes in the intensity of an arrival rate over a period of one week, using a piecewise-constant model, as will soon be discussed in Example 3. Figure 2(a) provides a pictorial representation of a constant arrival rate, as is the case with a homogeneous Poisson model. Figure 1(b) and Figure 2(b) demonstrate the accumulation of Λ over a period of one week. While the accumulation for the homogeneous Poisson model is linear over time, the accumulation rate of the expected number of arrivals in the nonhomogeneous case changes with fluctuations in the arrival intensity $\lambda(t)$.

We now consider the interarrival time distribution of the nonhomogeneous Poisson process. We first define the nonhomogeneous exponential distribution as follows:

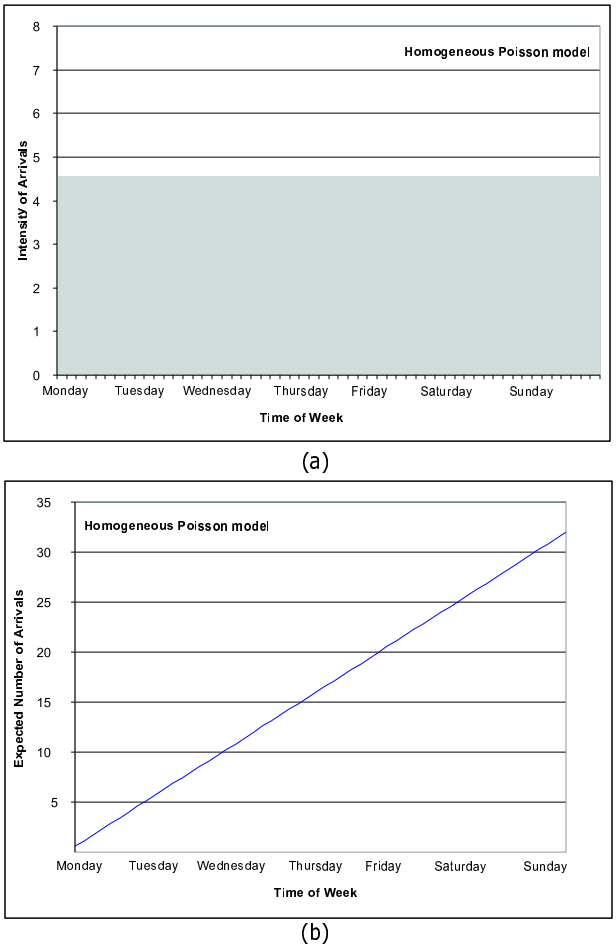


Figure 2: Homogeneous Poisson model.

Definition 1 (Nonhomogeneous exponential distribution) *Let $\phi : \Re \to [0, \infty)$ be a integrable function. Given some $s \in \Re$, a random variable V is said to have a* nonhomogeneous exponential *distribution (denoted by $V \sim \mathrm{Exp}_s(\phi(\cdot))$) if V's density function is*

$$p(\tau) = \begin{cases} \phi(s+\tau) \exp\left(-\int_0^\tau \phi(s+u)\, du\right), & \tau \geq 0 \\ 0, & \tau < 0. \end{cases}$$

It is worth noting that if $\phi(t)$ is constant, $p(\tau)$ is just a standard exponential distribution density. We shall now show that, as with homogeneous Poisson processes, the interarrival time of insertion events is distributed like an exponential random variable, L_s, but with a time-varying density function.

Lemma 1 *At any time s, the amount of time L_s to the next insertion event is distributed like* $\mathrm{Exp}_s(\lambda(\cdot))$. *The probability of an insertion event occurring during $(s, f]$ is* $\mathrm{P}\{L_s < f - s\} = 1 - e^{-\Lambda(s,f)}$.

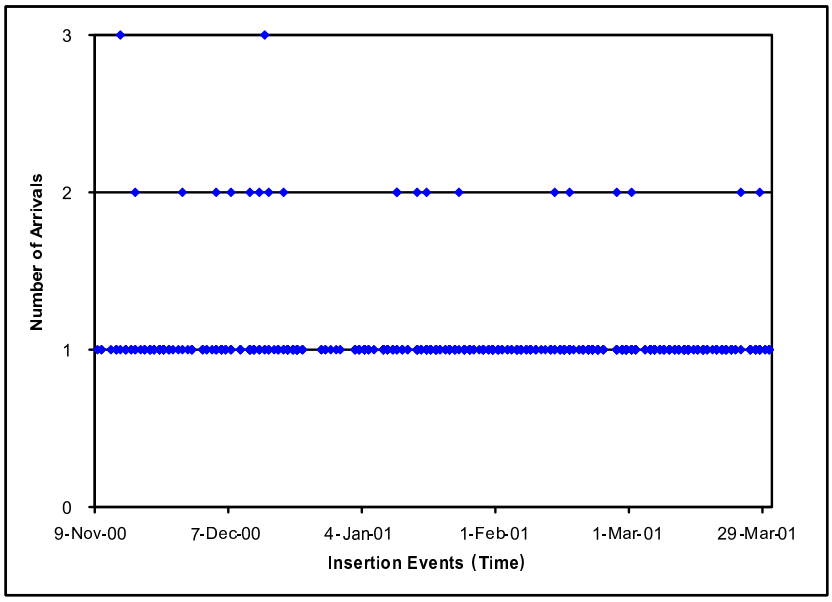


Figure 3: An example of a bulk insertion process.

Proof. Let $\{N(t), t \geq 0\}$ be a nonhomogeneous Poisson process with intensity function $\lambda(t)$, which implies $\mathrm{P}\{N(f) - N(s) = 0\} = e^{-\Lambda(s,f)}$. Now, the chance that no new tuple was inserted during $(s, f]$ is the same as the chance that the process $N(\cdot)$ has no arrivals during $(s, f]$, that is, $e^{-\Lambda(s,f)}$. The chance that a new tuple was inserted during $(s, f]$ is just the complement of the chance of no arrivals, namely,

$$\mathrm{P}\{L_s < f - s\} = \{N(f) - N(s) \geq 1\} = 1 - P\{N(f) - N(s) = 0\} = 1 - e^{-\Lambda(s,f)}.$$

Taking the derivative of this expression with respect to f and making a change of variables, the probability density of the time interval until the next insertion from time s is $p(\tau) = \lambda(s+\tau)e^{-\Lambda(s,s+\tau)}$. Thus, $L_s \sim \mathrm{Exp}_s(\lambda(\cdot))$. ■

At insertion event i, a random number of tuples Δ_i^+ are inserted, allowing us to model bulk insertions. A *bulk insertion* is the simultaneous arrival of multiple data items, and may occur because the items are related, or because of limitations in the implementation of the server. For example, e-mail servers may process an input stream periodically, resulting in bulk updates to a mailbox. Also, multiple pages of a website may be uploaded to a server simultaneously. Figure 3 provides a pictorial example of a bulk insertion process. The vast majority of the data arrive in quanta of 1, while some of the data arrive in quanta of 2 or 3. Assuming that the $\{\Delta_i^+\}$ are independent and identically distributed (IID), then the stochastic process $\{B(t), t \geq 0\}$ representing the cumulative number of insertions through time t is a *compound Poisson* process (*e.g.*, [37], pp. 87-88). We let $B(s, f)$ denote the number of insertions falling into the interval $(s, f]$. The expected number of inserted tuples during $(s, f]$ may be computed via $\mathrm{E}[B(s, f)] = \int_s^f \lambda(t)\,\mathrm{E}[\Delta^+]\,dt = \mathrm{E}[\Delta^+]\int_s^f \lambda(t)dt = \mathrm{E}[\Delta^+]\,\Lambda(s, f)$. Here, Δ^+ represents a generic random variable distributed like the $\{\Delta_i^+\}$.

We now consider three simple cases of this model:

Example 1 (General nonhomogeneous Poisson process) *Assume that* $\mathrm{E}[\Delta^+] = 1$. *The expected number of insertions simplifies to* $\mathrm{E}[B(s,f)] = \mathrm{E}[\Delta^+]\,\Lambda(s,f) = 1 \cdot \Lambda(s,f) = \Lambda(s,f)$. □

Example 2 (Homogeneous Poisson process) *Assume once more that* $\mathrm{E}[\Delta^+] = 1$. *Assume further that* $\lambda(t)$ *is a constant function, that is,* $\lambda(t) = \lambda$ *for all times* t. *In this case, as shown above,* $\Lambda(s,f)$ *takes on the simple form of* $\lambda \cdot (f-s)$. *Thus,* $\mathrm{E}[B(s,f)] = \Lambda(s,f) = \lambda \cdot (f-s)$. *The interarrival times are distributed as* $\mathrm{Exp}(\lambda)$, *the exponential distribution with parameter* λ. □

Example 3 (Recurrent piecewise-constant Poisson process) *A simple kind of nonhomogeneous Poisson process can be built out of homogeneous Poisson processes that repeat in a cyclic pattern. Given some length of time* Q, *such as one day or one week, suppose that the arrival rate function* $\lambda(t)$ *of the recurrent Poisson process repeats every* Q *time units, that is,* $\lambda(t) = \lambda(t - Q\lfloor t/Q \rfloor)$ *for all* t. *Furthermore, the interval* $[0, Q)$ *is partitioned into a finite number of subsets* $J_1, \ldots, J_K$, *with* $\lambda(t)$ *constant throughout each* J_k, $k = 1, \ldots, K$. *Finally, each* J_k *is in turn composed of a finite number of half-open intervals of the form* $[s, f)$. *For instance,* Q *might be one day, with* $K = 24$ *and* $J_1 = [0{:}00, 1{:}00), J_2 = [1{:}00, 2{:}00), \ldots, J_{24} = [23{:}00, 0{:}00)$. *As another simple example,* Q *might be one week, and* $K = 8$, *as given in Figure 1. The subsets* J_1, J_2 *and* J_3 *consist of the three-hour intervals of* $[0{:}00, 3{:}00)$, $[3{:}00, 6{:}00)$, *and* $[6{:}00, 9{:}00)$, *respectively.* J_4 *consists of a firm's normal hours of operation, say* $[9{:}00, 18{:}00)$ *for each weekday.* J_5 *and* J_6 *consist of the three-hour intervals of* $[18{:}00, 21{:}00)$, *and* $[21{:}00, 24{:}00)$, *respectively. Finally,* J_7 *and* J_8 *cover Saturday and Sunday, respectively. Formalisms like those of [31] could also be used to describe such processes in a more structured way. We term this class of Poisson processes to be* recurrent piecewise-constant *— abbreviated* RPC.

*Recurrent processes are a well-known phenomenon, and are apparent in a wide range of applications, including call centers, help desks, reservation systems (such as airline reservations), and retailing. Any database supporting such applications will be subject to such recurrent patterns. For example, Web pages are more likely to be updated during the working hours of the site's Webmaster. Research into time series (*e.g., *[7]) has well-established methodologies to identify and model such cyclic behaviors, and has utilized it for forecasting in data-intensive applications, such as sunspot levels, stock exchange prices, and baseball game outcomes. In Section 3.2 we show the usability of the RPC Poisson model for one application. However, we assert that it is suitable for many more, including the areas just mentioned.* □

It is worth noting that insertion model should be formed from the client's point of view. Therefore, if the server keeps a database from which many clients transcribe data, the modeling of insertions for a given client should only include the part of the database the client actually transcribes. Therefore, if a "road warrior" is interested only in new orders for the 08904 zip code area, the insertion model for that client should concentrate on this zip code, ignoring the arrival orders from other areas.

3.2 Model fitting and validation

Models, in general, are an idealized representation of a process. To be useful, we wish to make accurate predictions regarding the timing of insertions to the database, based on either tractable analytical calculations or simulations. In particular, it is well known that Poisson processes model

a world where data updates are independent from one another. While in databases with widely distributed access, *e.g.*, incoming e-mails, postings to newsgroups, or posting of orders from independent customers, such an independence assumption seems plausible, we still need to validate the model against real data. The experiments we have performed and a discussion of the results will be the focus of this section. We shall demonstrate the capabilities and limitations of the proposed model via two sets of data. Experiments with electronic bulletin board data are reported in Section 3.2.1. Section 3.2.2 discusses modeling of World Cup website log data. We shall demonstrate the model's benefits in generating transcription policies in Section 4.

In the experiments described below, we have used the Kolmogorov-Smirnov goodness of fit test (see for example [19, Section 7.7]). For completeness, we first overview the principles of this statistical test. The Kolmogorov-Smirnov test evaluates the likelihood of a *null hypothesis* that a given sample may have been drawn from some hypothesized distribution. If the null hypothesis is true, and sample set has indeed been drawn from the hypothesized distribution, then the empirical cumulative distribution of the sample should be close to its theoretical counterpart. If the sample cumulative distribution is too far from the hypothesized distribution at any point, this suggests that the sample comes from a different distribution. Formally, suppose that the theoretical distribution is $F(x)$, and we have n sample values $x_1, ..., x_n$ in nondecreasing order. We define an empirical cumulative distribution $F_n(x)$ via

$$F_n(x) = \begin{cases} 0, & \text{if } x < x_1 \\ \frac{k}{n}, & \text{if } x_k \leq x < x_{k+1} \\ 1, & \text{if } x > x_n, \end{cases}$$

and then compute $D_n = \sup_{k=1,...,n}\{|F_n(x_k) - F(x_k)|\}$. For large n, given a significance level α, the test measures D_n against $X(\alpha)/\sqrt{n}$, where $X(\alpha)$ is a factor depending on the *significance level* α at which we reject the null hypothesis. For example, $X(0.05) = 1.36$ and $X(0.1) = 1.22$. The value of α is the probability of a "false negative," that is, the chance that the null hypothesis might be rejected when it is actually true. Larger values of α make the test harder to pass.

3.2.1 Experimenting with electronic bulletin data

Our first data set is taken from postings to the DBWORLD electronic bulletin board. The data were collected over seven months and consists of more than 800 insertions, from November 9^{th}, 2000 through June 13^{th}, 2001. Figure 4 illustrates a data set with 580 insertions during the interval [2000/11/9:00:00:00, 2001/3/31:00:00:00). We used the Figure 4 data as a *training set*, *i.e.*, it serves as our basis for parameter estimation. Later, in order to test the model, we applied these parameters to a separate *testing set* covering the period [2001/3/31:00:00:00, 2001/6/14:00:00:00). In the experiments described below, we tried fitting the training data with two insertion-only models, namely a homogeneous Poisson process and an RPC Poisson process (see Section 3.1). For each of these two models, we have applied two variations, either as a compound or as a non-compound model.

Fitting the homogeneous Poisson process Based on the training set, we computed the parameter for a homogeneous Poisson process by averaging the 580 interarrival times, an unbiased estimator of the Poisson process parameter. The average interarrival time was computed to be 5:15:19, and thus $\lambda = 4.57$ per day. Figure 5(a) provides a pictorial comparison of the cumulative

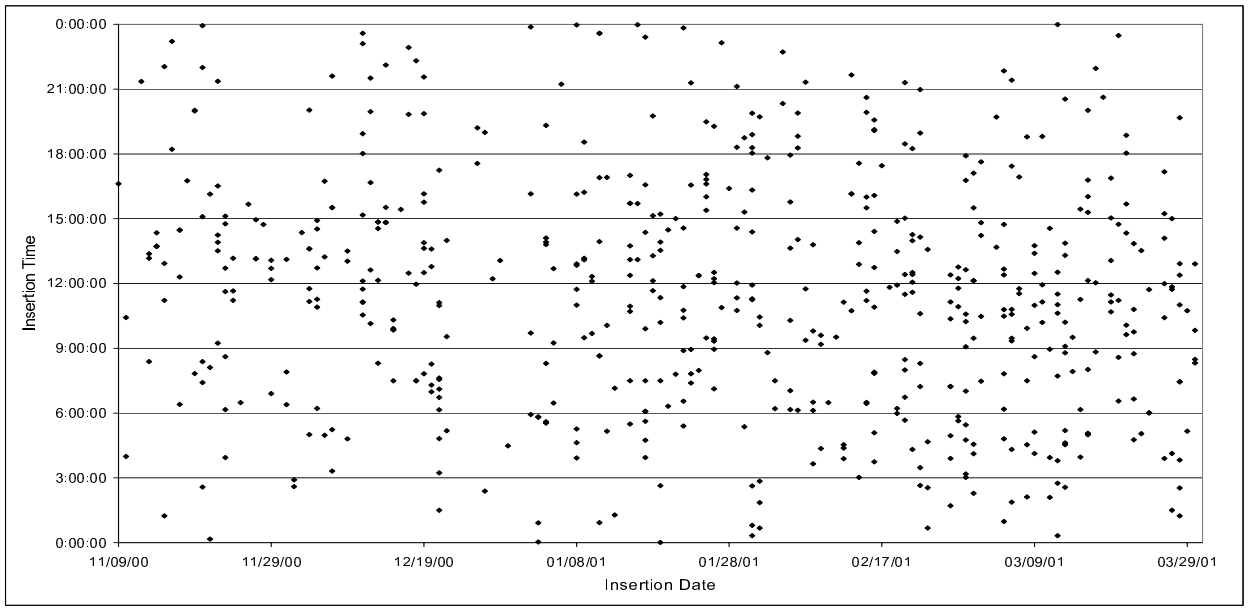


Figure 4: Training data set.

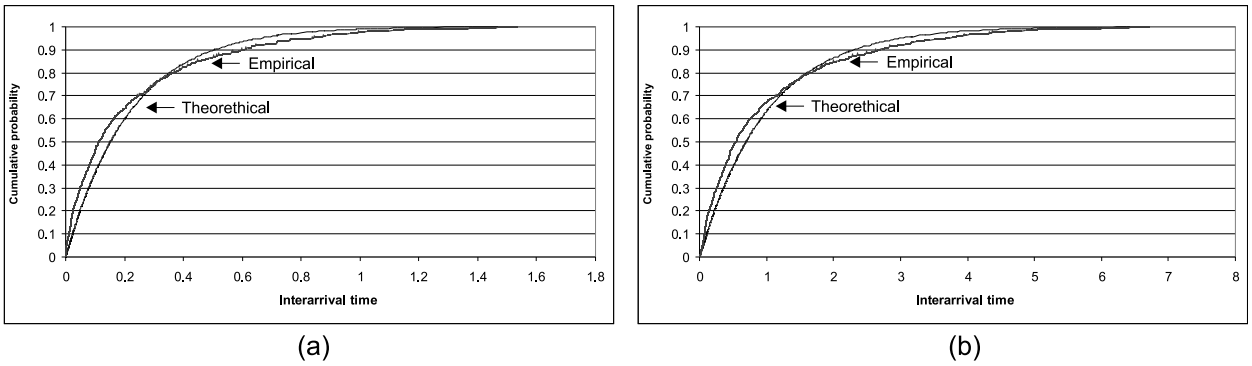


Figure 5: A comparison of a theoretical and an empirical distribution function for the homogeneous Poisson process model (a) and the compound homogeneous Poisson process model (b).

	Weekdays	Saturday	Sunday
[0:00, 3:00)	2.40	1.50	1.15
[3:00, 6:00)	5.96		
[6:00, 9:00)	6.04		
[9:00, 18:00)	7.50		
[18:00, 21:00)	3.03		
[21:00, 24:00)	2.41		

Table 1: Average λ levels for the recurrent piecewise-constant Poisson model.

distribution functions of the interarrival times with their theoretical counterpart. We applied the Kolmogorov-Smirnov test to the distribution of interarrival times, comparing it with an exponential distribution with a parameter of $\lambda = 4.57$. The outcome of the test is $D_n = 0.106$, which means we can reject the null hypothesis at any reasonable level of confidence $\alpha \geq 0.005$ (for $\alpha = 0.005$, the rejection threshold is 0.0718 for $n = 580$). In all likelihood, then, the data are not derived from a homogeneous Poisson process.

Next, we applied a compound homogeneous Poisson model. Our rationale in this case is that DBWORLD is a moderated list, and the moderators sometimes work on postings in batches. These batches are sometimes posted to the group in tightly-spaced clusters. For all practical purposes, we treat each such cluster as a single batch insertion event. To construct the model, any two insertions occurring within less than one minute from one another were considered to be a single event occurring at the insertion time of the first arrival. For example, on November 14, 2000, we had three arrivals, one at 13:43:19, and two more at 13:43:23. All three arrivals are considered to occur at the same insertion arrival event, with an insertion time of 13:43:19. Figure 3 provides an illustration of the number of arrivals in each batch of the training set.

Using the compound variation, the data set now has 557 insertion events. The revised average interarrival time is now 5:28:20, and thus $\lambda = 4.39$ per day. Figure 5(b) provides a pictorial comparison of the cumulative distribution functions of the interarrival times, assuming a compound model, with their theoretical counterpart. We have applied the Kolmogorov-Smirnov test to the distribution of interarrival times, comparing it with an exponential distribution with a parameter of $\lambda = 4.39$. The outcome was somewhat better than before. $D_n = 0.094$, which means we can still reject the null hypothesis at any level of confidence $\alpha \geq 0.005$ (for $\alpha = 0.005$, the rejection threshold is 0.0733 for $n = 557$). Although the compound variant of the model fits the data better, it is still not statistically plausible.

Fitting the RPC Poisson process Next, we have attempted to fit the data to an RPC model. Examining the data, we chose a cycle of one week. Within each week, we used the same pattern for each weekday, with one interval for work hours (9:00-18:00), plus five additional three-hour intervals for "off hours." We treated Saturday and Sunday each as one long interval. This segmentation is the same as described in Example 3. Table 1 shows the arrival rate parameters for each segment of the RPC Poisson model, calculated in much the same manner as for the homogeneous Poisson model. Figure 1(a) provides a pictorial representation of the various λ levels.

The specific methodology for structuring the RPC Poisson model is beyond the scope of this

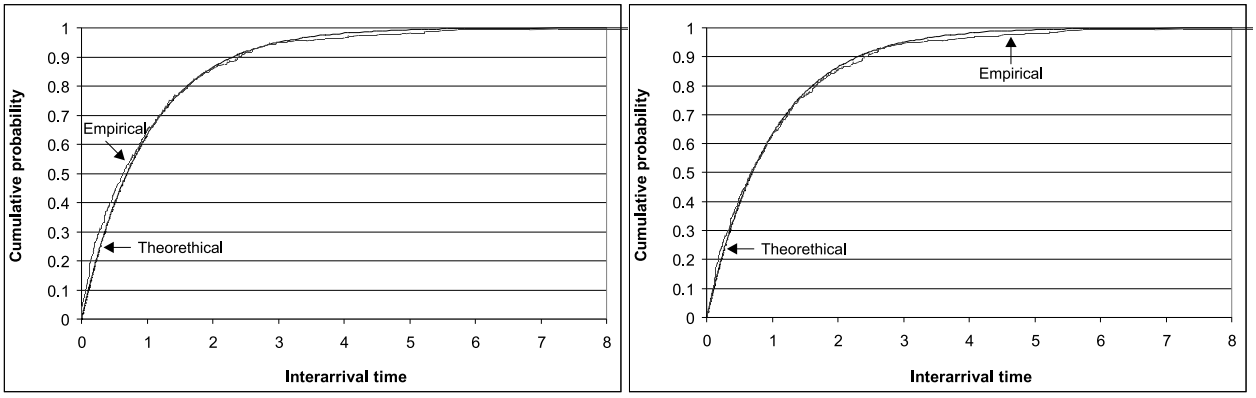


Figure 6: A comparison of a theoretical and an empirical distribution function of U_s for the RPC Poisson model (a) and the compound RPC Poisson model (b).

paper and can range from *ad hoc* "look and feel" crafting to more established formal processes for statistically segmenting, filtering, and aggregating intervals (see, *e.g.*, [21, 48, 43] and references therein). It is worth noting, however, that from experimenting with different methods, we have found that the model is not sensitive to small changes in the interval definitions. Also, the model we selected has only 8 segments, and thus only 8 parameters, so there is little danger of "overfitting" the training data set, which has over 500 observations.

Next, we attempted to statistically validate the RPC model. To this end, we use the following lemma:

Lemma 2 *Given a nonhomogeneous Poisson process with arrival intensity $\lambda(t)$, the random variable $U_s = \int_s^{s+L_{R,s}} \lambda(t)\,dt$, where L_s is as defined above, is of the distribution* Exp(1).

Proof. Let $f_s(t) = \Lambda(s, s+t)$, which is a monotonically nondecreasing function. From Lemma 1, $\mathrm{P}\{L_s < t\} = 1 - e^{f_s(t)}$ for all $t \geq 0$. We have $U_s = f_s(L_s)$. By applying the monotonic function f_s to both sides of the inequality $L_s < t$, one has that $\mathrm{P}\{f_s(L_s) < f_s(t)\} = \mathrm{P}\{L_s < t\} = 1 - e^{f_s(t)}$ for all $t \geq 0$. Substituting in the definitions of U_s and $u = f_s(t)$, one then obtains $\mathrm{P}\{U_s < u\} = 1 - e^{-u}$ for all $u \geq 0$, and therefore $U_s \sim \mathrm{Exp}(1)$. ■

Thus, given an instantaneous arrival rate $\lambda(t)$, and a sequence of observed arrival events $\{t_n\}_{n=0}^{N}$, we compute the set of values $u_n = \int_{t_{n-1}}^{t_n} \lambda(t)\,dt$, $n = 1, \ldots, N$, and perform a Kolmogorov-Smirnov test of these versus the unit exponential distribution.

Figure 6(a) provides a comparison of the theoretical and empirical cumulative distribution of the random variable U. We applied the Kolmogorov-Smirnov test to U, comparing it with an exponential distribution with $\lambda = 1$, based on Lemma 2. The outcome of the test is $D_n = 0.080$, which is better than either homogeneous model, but is still rejected at any reasonable level of significance (recall that for $\alpha = 0.005$, the rejection threshold is again 0.0718 for $n = 580$).

Finally, we evaluated a compound version of the RPC model, combining successive postings separated by less than one minute. We kept the same segmentation as in Table 1, but recalculated the arrival intensities in each segment, as shown in Table 2.

	Weekdays	Saturday	Sunday
[0:00, 3:00)	2.40	1.45	1.15
[3:00, 6:00)	5.96		
[6:00, 9:00)	5.59		
[9:00, 18:00)	7.11		
[18:00, 21:00)	3.03		
[21:00, 24:00)	2.33		

Table 2: Average λ levels for the compound RPC Poisson model.

Model	D_n	**smallest α level for rejection**
Homogeneous	0.106	0.005
Homogeneous+compound	0.094	0.005
RPC	0.080	0.005
RPC+compound	0.050	> 0.100

Table 3: Goodness of fit of the four models.

Next, we recalculated the sample of the random variable U_s for the compound RPC Poisson model, and applied the Kolmogorov-Smirnov test. In this case, we have $D_n = 0.050$, which cannot be rejected at any reasonable confidence level through $\alpha = 0.10$ (for $\alpha = 0.10$, the rejection threshold is 0.0517 for $n = 557$). Figure 6(b) shows the theoretical and empirical distributions of U in this case.

As a final confirmation of the applicability of the compound RPC Poisson model, we attempted to validate the assumption that the number of postings in successive insertion events are independent and identically distributed (IID). In the sample, 536 insertion events were of size 1, 19 were of size 2, and 2 were of size 3, as shown in Figure 3. Thus, we approximate the random variable Δ_R^+ as having a $536/557 \approx .962$ probability of being 1, a $19/557 \approx .034$ probability of being 2, and a $2/557 \approx .004$ probability of being 3. Validating that the observed insertion batch sizes Δ_i^+ appear to be independently drawn from this distribution is somewhat delicate, since they nearly always take the value 1. To compensate, we performed our test on the *runs* in the sample, that is, the number of consecutive insertion events of size 1 between insertions of size 2 or 3. Our sample contains 21 runs, ranging from 0 to 112. If the insertion batch sizes $\{\Delta_i^+\}$ are independent with the distribution Δ_R^+, then the length of a run should be a geometric random variable with parameter $536/557 \approx .962$. We tested this hypothesis via a Kolmogorov-Smirnov test, as shown in Figure 7. The D_n statistic is 0.207, which is within the $\alpha = 0.1$ acceptance level for a sample of size $n = 21$. Although the divergence of the theoretical and empirical curves in Figure 7 is more visually pronounced than in the prior figures, it should be remembered that the sample is far smaller. Thus, the assumption that the insertion batch sizes $\{\Delta_i^+\}$ are IID is plausible.

3.2.2 Discussion

In this section, we have verified that our data feed is modeled reasonably well by a compound RPC Poisson model with eight segments. Table 3 summarizes our experiments by comparing

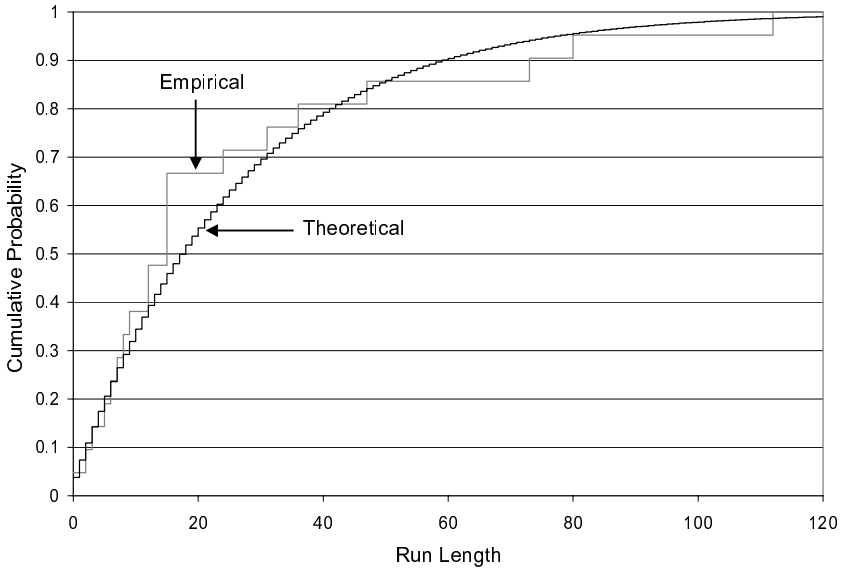


Figure 7: Empirical and theoretical distributions for number of single arrivals between multiple arrivals, compound RCP Poisson model.

the goodness of fit of the four models to the test data. For each of the models, it displays the Kolmogorov-Smirnov test statistic (D_n) and the level α at which one can reject the null hypothesis. The higher the α, the better the fit. The RPC compound Poisson model models the data set best, accepting the null hypothesis at any level up to 0.1. The main conclusion from these experiments is that the simple model of the homogeneous Poisson process is limited to the modeling of a restricted class of applications (one of which was suggested in [11]). Therefore, there is a need for a more elaborate model, as suggested in this paper, to capture a broader range of update behaviors. A nonhomogeneous model consisting of just 8 segments per week, as we have constructed, seems to model the arrivals significantly better than the homogeneous approach. The methodology we have followed, including the decision regarding the size of the training set and the statistical test, is common practice in the area of statistics and can be readily implemented using methods as simple as spreadsheet macros. It is also worth noting that we have received similar results, experimenting with an email inbox. Using seven parameters (five for portions of working days, one for Saturday, and one for Sunday), we were able to model the arrival of emails using a compound RPC Poisson model.

To demonstrate the limitation of the model, we next present our experiments with a data set that consists of all the requests made to the 1998 World Cup Web site between April 30, 1998 and July 26, 1998.[5] This log file describes all requests submitted by users to multiple redundant servers supporting the site. Each data item served is referred to as an *object*.

The log file does not explicitly indicate when each object in the Website was updated. We have used changes in the number of bytes in an object as a surrogate for an update, and estimated the time of update as the first time a download occurred with a new length. Our training set consisted of eight days of data (from June 9, 1998 to June 17, 1998), and the testing set consisted of the

[5]The data was taken from http://ita.ee.lbl.gov/html/contrib/WorldCup.html

Model	D_n	**smallest α level for rejection**	**rejection threshold for $\alpha = 0.001$**
Homogeneous	0.405	< 0.001	0.028
Homogeneous+compound	0.146	< 0.001	0.053
RPC	0.391	< 0.001	0.028
RPC+compound	0.113	< 0.001	0.052

Table 4: Goodness of fit of the four models: the WorldCup data set

following seven days.

We applied the following rules to the data:

1. Dynamically-generated objects (about 1% of all requests) were ignored.
2. When an object changed size, the change was not recorded until most requests had the new object size for at least two minutes after the change was detected. This helped eliminate the effects of inconsistencies at different servers immediately following an update. If most subsequent requests to the object had the old object size, the "change" was assumed to be an error and was not recorded.
3. Objects that frequently "bounced" between two different values were also ignored, since it was not clear they were actually being updated.

10,074 update times of 4,405 objects were analyzed and compared to a homogeneous Poisson model and an RPC Poisson model with eight equally-sized segments per day, assuming a cyclic behavior that repeats daily. As in the DBWORLD data, each model has two variations, simple and compound. For the compound models, we aggregated updates within 30 seconds of one another into a single update event. The data in Table 4 indicates that the World Cup data cannot be modeled using a Poisson model, either homogeneous or RPC. It is interesting to note the improvements of results when shifting from a simple to a compound model. The main reason for the improvement lies in the way we have estimated objects' time of update. Since the trace does not include a last modification time attribute, we had to estimate the update time via request times. Typically, Web requests for complex pages include many objects nearly simultaneously, thus leading our estimation technique to surmise apparently simultaneous updates.

Of course, an RPC model is essentially static over time intervals significantly longer than its period Q. Real-world processes may exhibit a combination of cyclic behavior and long-term changes, such as the gradual growth of a customer base or a sharp lasting rise in the popularity of a discussion topic. Thus, it would be useful to detect when the RPC model should be revised and refitted. The full methodology of model adaptation involves sequential decision making and process control techniques, which are beyond the scope of this paper, yet can be handled by advanced statistical mechanisms, as developed in *statistical process control* (SPC) and *change-point theory*. In these fields, problems of detection of changes (*monitoring*), estimation of the current process parameters (*filtering*), and identifying the change points and regimes (*segmentation*) have been tackled in numerous important application areas, including industrial quality control, automatic fault detection in control dynamical systems (*e.g.*, automatic pilots and robotics), and segmentation and pattern recognition of sound and image signals. See, *e.g.*, [32, 47, 40, 22, 23, 38, 35, 44, 42,

39, 30] and numerous references therein), and can be adopted in our case. The rich machinery supplied by this literature can be used to develop and implement a powerful, data-driven, and computationaly convenient methodology for monitoring the arrival rate λ and identifying its change points, regimes, and intensity levels. However, such a treatment is beyond the scope of this article and will be deferred to a future article.

4 Transcription policies

Equipped with a data arrival model, we shall now study various transcription policies, where the question is how often to generate a remote replica. In Section 4.1, we develop a plausible example cost model for evaluating various policies. A description of four different policies is laid out in Section 4.2. Finally, Section 4.3 describes experiments with the transcription policies, comparing the homogeneous Poisson model with the RPC Poisson model.

4.1 Cost model

A transcription policy aims to minimize the combined cost of *transcription* and *obsolescence* [16]. The former includes the cost of connecting to a network and the cost of transcribing the data, and may depend on the time at which the transcription is performed (*e.g.*, as a function of network congestion), and the length of connection needed to perform the transcription. Obsolescence cost captures the cost of using obsolescent data, and is basically a function of the amount of time that has passed since the last transcription.

Let $C_{\mathrm{u}}(s,t)$ denote the cost of performing a transcription starting at time t, given that the last update was started at time s. Let $C_{\mathrm{o}}(s,t)$ denote the obsolescence cost through time t attributable to tuples inserted at the server during the time interval $(s,t]$. Then the total cost $C(T)$ through time T is

$$C(T) = \rho \sum_{i:b_i \le T} C_{\mathrm{u}}(b_{i-1}, b_i) + (1-\rho) C_{\mathrm{o}}(0,T), \tag{1}$$

where ρ describes the relative importance a user puts on the transcription cost versus the obsolescence cost. Traditionally, $\rho = 0$, and therefore $C(T)$ is minimized for $C_{\mathrm{o}}(0,T) = 0$, allowing the use of current data only. In this section, we shall look into another, more realistic approach, where data currency is sacrificed (up to a level as defined by the *user* through ρ) for the sake of reducing transcription costs. Ideally, one would want to choose the sequence $\{(b_i, e_i)\}_{i=1}^{\infty}$ of connectivity periods, subject to any constraints on their durations $e_i - b_i$, to minimize $C(T)$ over some time horizon T. One may also consider the asymptotic problem of minimizing the average cost over time, $\lim_{T\to\infty} C(T)/T$. We note that the presence of ρ is not strictly required, as its effects could be subsumed into the definitions of the C_{u} and C_{o} functions, especially if both are expressed in natural monetary units. However, we retain ρ in order to demonstrate some of the parametric properties of our model.

In general, modeling transcription and obsolescence costs may be difficult and application-dependent. They may be difficult to quantify and difficult to convert to a common set of units, such as dollars or seconds. Some subjective estimation may be needed, especially for the obsolescence cost. However, we maintain that, rather than avoiding the subject altogether, it is best to try to construct these cost models and then use them, perhaps parametrically, to evaluate transcription

policies. Any transcription policy implicitly makes some trade-off between consuming network resources and incurring obsolescence, so it is best to try to quantify the trade-off and see if a better policy exists. In particular, one should try to avoid policies that are clearly *dominated*, meaning that there is another policy with the same or lower transcription cost, and strictly lower obsolescence, or *vice versa*. Below, for purposes of illustration, we will give one simple, plausible way in which the cost functions may be constructed; alternatives are left to future research, based on, for example, the experience gathered in implementing cost functions in control systems (*e.g.*, [42]).

4.1.1 Transcription costing example

In determining the transcription cost, one may use existing research into costs of distributed query execution strategies. Typically, (*e.g.*, [29]) the transcription time can be computed as some function of the CPU and I/O time for writing the new tuples onto the client and the cost of transmitting the tuples over a network. There is also some fixed setup time to establish the connection, which may be substantial. For purposes of example, suppose that $C_{\mathrm{u}}(s,t) = c + \beta B(s,t)$. Here, $c \geq 0$ denotes the fixed setup cost, $\beta \geq 0$, and $B(s,t)$ is the number of data items that have arrived during $(s,t]$, and thus need to be transferred to the client. We note that, under this assumption

$$\sum_{i:b_i \leq T} C_{\mathrm{u}}(b_{i-1}, b_i) = n(T)c + \beta B(0, b_{i^*(T)}),$$

where $n(T)$ is the number of transcriptions during $[0,T]$, and $i^*(T) = \max\left\{i \;\middle|\; b_i \leq T\right\}$. For large T, one would expect the $\beta B(0, b_{i^*(T)})$ term to be roughly comparable across most reasonable polices, whereas the $n(T)c$ term may vary widely for any value of T. It is worth noting that c and β could be generalized to vary with time or other factors. For example, due to network congestion, certain times of day may have higher unit transcription costs than others. Also, transcribing via airline-seat telephone costs substantially more than connecting via a cellular phone. For simplicity, we have refrained from discussing such variations in the transcription cost.

4.1.2 Obsolescence costing example

We next turn our attention to the obsolescence cost, which is clearly a function of the arrival times of new tuples and the times they are transcribed to the client, as well as the user's subjective perception of obsolescence. Intuitively, the shorter the time between the arrival of a tuple and its transcription to the client, the better off the client should be. As a basis for the obsolescence cost, we suggest a criterion that takes into account user preferences, as well as the content evolution parameters.

We will denote individual tuples by an index r, and let $t(r)$ denote the time r is inserted into the server database. Let $i(r) = \min\left\{i \;\middle|\; b_i \geq t(r)\right\}$, that is, the index of the connection period during which r is transcribed to the client. Then we let $\iota(t,r)$ be a function denoting r's contribution to the obsolescence cost through time t, via

$$\iota(t,r) = \begin{cases} 0 & t \leq t(r) \\ f(t,r) & t(r) < t \leq e_{i(r)} \\ f(e_{i(r)}, r) & t > e_{i(r)}, \end{cases} \tag{2}$$

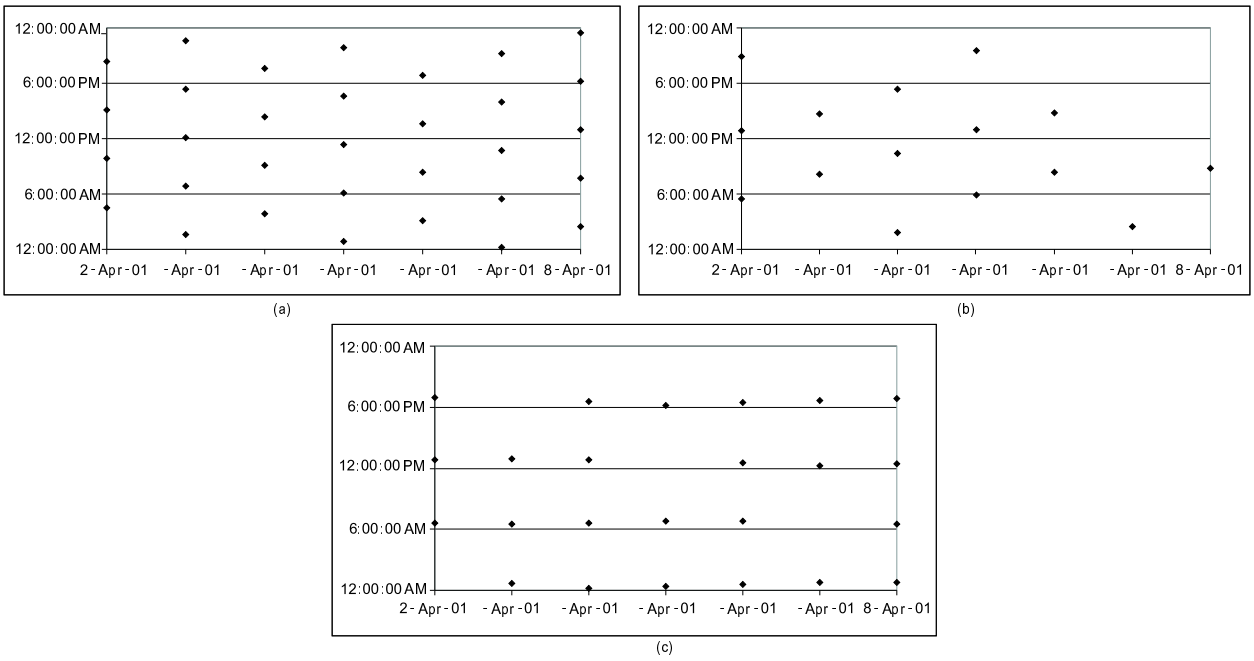


Figure 8: Transcription times for different policies.

where $f(t, r)$ is some application-dependent function, representing the level of importance a user assigns with a message arriving at a time $t(r)$. For example, a user may attach greater importance to messages arriving during official work hours, and a lesser measure of importance to non-work hours (since no one expects her to be available at those times). Thus, one might define

$$f(t, r) = \int_{t(r)}^{t} a(\tau)\, d\tau, \quad \text{where } a(\tau) = \begin{cases} a_1, & \text{if } \tau \text{ is during work hours} \\ a_2, & \text{if } \tau \text{ is after hours,} \end{cases} \tag{3}$$

and $a_1 \geq a_2$. For $a_1 = a_2 = 1$, $\iota(t, r)$ takes a form resembling the age of a local element in [11]. More complex forms of $f(t, r)$ are certainly possible. In this simple case, we refer to a_1/a_2 as the *preference ratio*, rather than specifying a_1 and a_2 individually.

Finally, we define $C_{\mathrm{o}}(s, t) = \sum_{r:t(r)\in(s,t]} \iota(t, r)$.

4.2 Three transcription policies

We next suggest three different transcription policies and compare their performance against our test data. Figure 8 shows a comparison of transcription schedules for the week starting April 2, 2001.

Uniform synchronization point policy: This transcription policy (referred to below as the *USP policy*) was suggested in [11]. According to this policy, the intervals $(b_i, b_{i+1}]$ are equal in size for all i. The decision regarding the interval size $b_{i+1} - b_i$ may be either arbitrary (*e.g.*, once a day) or may depend on λ, the Poisson model parameter (in which case a homogeneous Poisson process is implicitly assumed). The policy may be expressed as $b_{i+1} = b_i + M/\lambda$ for

some multiplier $M > 0$. According to this policy with $M = 1$ (as suggested in [11]), and $\lambda = 4.57$ per day as computed in Section 3.2.1, we would refresh the database every 5 hours, 15 minutes, and 19 seconds. Figure 8(a) introduces the transcription times for the week of $[2001/4/2{:}0{:}00, 2001/4/9{:}0{:}00)$ using the uniform synchronization point policy with $M = 1$ and $\lambda = 4.57$ per day.

Threshold policy: In the *threshold* policy, given that the last connection started at time b_i, we transcribe at time t if the expected obsolescence cost $\mathrm{E}[C_{\mathrm{o}}(b_i, t)]$ exceeds Π, where Π is a threshold that measures the user's tolerance to obsolescent data. Using the definition of $C_{\mathrm{o}}(b_i, t)$ and the properties of nonhomogeneous Poisson processes, we calculate

$$\begin{aligned}
\mathrm{E}[C_{\mathrm{o}}(b_i, t)] &= \mathrm{E}\left[\sum_{r:t(r)>b_i} \iota(t, r)\right] \\
&= \mathrm{E}[B(b_i, t)] \cdot \mathrm{E}\left[\iota(t, r) \;\middle|\; t(r) \in (b_i, t]\right] \\
&= \mathrm{E}\left[\Delta^+\right] \Lambda(b_i, t) \int_{b_i}^{t} \frac{\lambda(t')}{\Lambda(b_i, t)} f(t, t')dt' \\
&= \mathrm{E}\left[\Delta^+\right] \int_{b_i}^{t} \lambda(t') f(t, t')dt'.
\end{aligned}$$

As an example, consider the homogeneous case where $\mathrm{E}[\Delta^+] = 1$ and $\lambda(t) = \lambda$ for all t. Assume further that $a_1 = a_2 = 1$ for all t. In this case,

$$\mathrm{E}[C_{\mathrm{o}}(b_i, t)] = \int_{b_i}^{t} \lambda \cdot (t - t')dt' = \lambda \int_{b_i}^{t} (t - t')dt' = \frac{1}{2}\lambda \cdot (t - b_i)^2$$

Setting $t = b_i + M/\lambda$ and $\Pi = \mathrm{E}[C_{\mathrm{o}}(b_i, t)]$, one has that

$$\Pi = (1/2)\lambda \cdot (t - b_i)^2 = (1/2)\lambda \cdot (M/\lambda)^2 = M^2/2\lambda.$$

Therefore, for $\Pi = M^2/2\lambda$, we recover the homogeneous policy defined above.

Figure 8(b) shows the transcription times for the week $[2001/4/2{:}0{:}00, 2001/4/9{:}0{:}00)$, using the RPC arrival model and the threshold policy with $\Pi = 0.109$ (obtained by setting $M = 1$ and $\lambda = 4.57$ per day, and letting $\Pi = M^2/2\lambda$). Transcriptions are more frequent when the λ intensity is higher (*e.g.*, midweek) and less frequent whenever the arrival rate is expected to be more sluggish (*e.g.*, weekends).

First arrival policy: Suppose the user wishes to refresh her replica whenever the probability of having any new data exceeds some threshold π, regardless of the item's importance, or the total number of items that might be pending transcription. Therefore, the user expects to find one or more new data items upon connection, with probability π. The parameter π reflects the importance a user puts on the timely transcription of any new data item. The lower π gets, the more willing a user is to connect the network in vain, rather then miss on a recent arrival. We call this policy the *first arrival policy* (*FA* for short). From Section 3.1, we know that the probability of a new arrival during $(b_i, t]$ is $\mathrm{P}\{L_{b_i} < t - b_i\} = 1 - e^{-\Lambda(b_i,t)}$. Therefore, a refresh is required at time t if $1 - e^{-\Lambda(b_i,t)} \geq \pi$. For a homogeneous process, we can formulate an equivalent condition on $t - b_i$ as follows:

$$\begin{aligned}
& 1 - e^{-\lambda(t-b_i)} && \geq && \pi \\
\Leftrightarrow \quad & 1 - \pi && \geq && e^{-\lambda(t-b_i)} \\
\Leftrightarrow \quad & \log(1 - \pi) && \geq && -\lambda(t - b_i) \\
\Leftrightarrow \quad & t - b_i && \geq && -\log(1 - \pi)/\lambda.
\end{aligned}$$

Setting $t = b_i + M/\lambda$ and $\pi = 1 - e^{-\lambda(t-b_i)}$, one has $\pi = 1 - e^{-\lambda(M/\lambda)} = 1 - e^{-M}$. Figure 8(c) shows a transcription schedule using the RPC arrival model and $\pi = 0.63$ (computed by setting $M = 1$). As with the threshold policy, transcriptions are unevenly distributed, to reflect the nonhomogeneous nature of the application.

4.3 Comparison of transcription policies

To examine the impact of various transcription policies on the combined cost of transcription and obsolescence, we collected two and a half months of DBWORLD data, with 254 insertions during the time interval [2001/3/31:00:00:00, 2000/6/17:00:00:00). There are three principle variables in this study, namely the transcription policy, the transcription policy parameter M, and the insertion model (either homogeneous or RPC). Since both Π (of the threshold policy) and π (of the FA policy) can be defined in terms of M, we simplify the presentation by parameterizing all policies by M. For each set of tests, we are concerned with the following question: given an importance ratio a_2/a_1 and a ρ, as set by the user, we need to identify an optimal M and to choose a preferred insertion model. The choice of insertion model applies only to the threshold and FA policies, since the USP policy implicitly assumes the homogeneous model. In the rest of this section, we shall present our experiments with the various policies and discuss the results using two different tests. The first is the dominance test (D-test for short), which compares the performance of two models and identifies values of M for which one model outperforms the others, both in terms of number of transcriptions and in terms of obsolescence cost. The second test is the ρ-test, which determines, for a given ρ, which model outperforms the other. In the discussion that follows, we have computed the obsolescence cost using Formulae 2 and 3.

USP Policy Figure 9 displays the results of experimenting with the USP policy. As discussed earlier, this policy implicitly assumes a homogeneous model. We have used $\lambda = 4.57$ per day, as calculated above. Figure 9(a) illustrates the number of transcriptions and the obsolescence cost for various choices of M (some values of M are indicated to enhance readability). As expected, the number of transcription falls as M grows, while the obsolescence cost increases. Figure 9(b) displays the combined cost function normalized to a $[0, 1]$ scale, taking the number of transcriptions as representative of the transcription cost. Using Figure 9(b) and given the user's preferred ρ, an appropriate M can be selected to minimize the combined cost. For example, for $\rho = 0.5$, the best choice would be with $M = 3$.

The threshold policy: homogeneous versus RPC insertion We next performed experiments comparing the performance of the threshold policy for the homogeneous Poisson model and the RPC Poisson model. Figure 10 shows representative results. Figure 10(a) displays the obsolescence cost and the number of transcriptions for various M values, with a user preference ratio of 3:1. Running the D-test for individual M's has the following results: for $M = 1$, the RPC model dominates the homogeneous model: the RPC model performs 158 transcriptions during the experiment periods, while the homogeneous model performs 352 transcriptions; the RPC model has an obsolescence value of 43.23, as opposed to 52.05 of the homogeneous model. Therefore, not only has the RPC model saved network resources, but these savings did not come at the expense of higher obsolescence cost. For all other M values, there is no dominant model.

Figure 10(b) displays partial results of the ρ-test. It provides a comparison of the combined

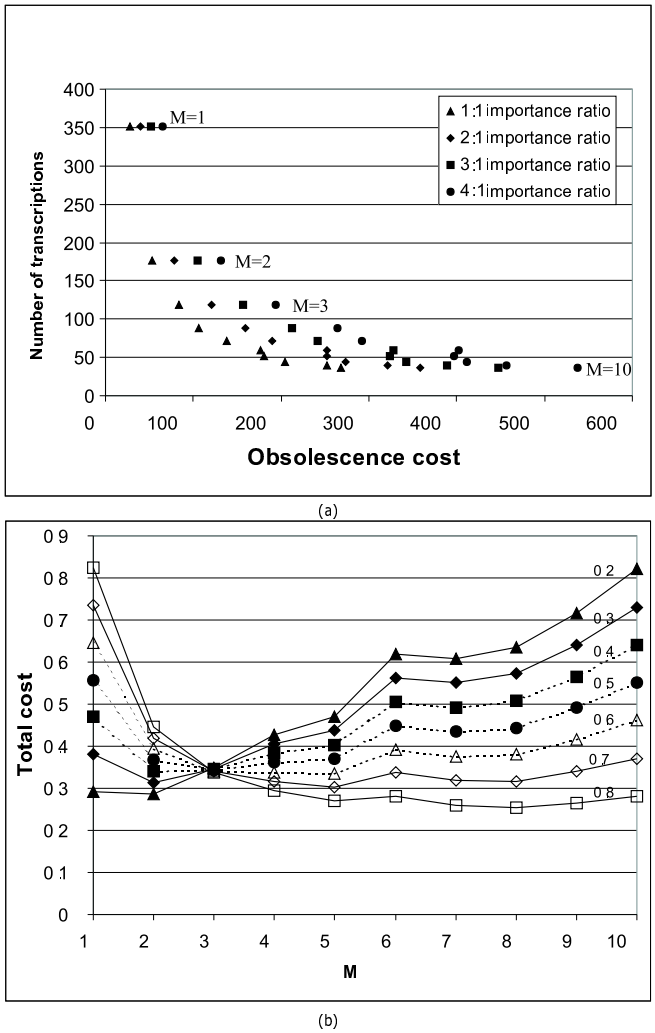


Figure 9: Uniform synchronization point policy.

normalized obsolescence and transcription costs for both insertion models and $\alpha \in \{0.6, 0.7, 0.8\}$ (assuming still a 3:1 preference ratio). Solid lines represent results from the homogeneous Poisson model, while dotted lines represent results related with the RPC Poisson model. Generally speaking, the RPC model performs better for small M values ($M < 7$), while the homogeneous model performs better for larger M values ($M \geq 8$).

To demonstrate the impact user preferences have on the obsolescence and transcription costs, consider Figure 11, which provides a comparison of costs for the homogeneous model and the RPC model using the threshold policy, for $M = 1$, and four different user preferences (as reflected by a varying ratio, from 1:1 to 4:1). The performance of the threshold policy using the homogeneous model is indicated by the dark diamonds at the top of the graph, while the performance of the threshold policy using the RPC model is indicated by the shaded squares at the bottom of the graph. For a 1:1 ratio, the RPC model performs far better in terms of the number of transcriptions, but is somewhat outperformed by the homogeneous model in terms of obsolescence cost. However, for the 2:1, 3:1, and 4:1 ratios, the RPC model is dominant, reducing both transcription and obsolescence

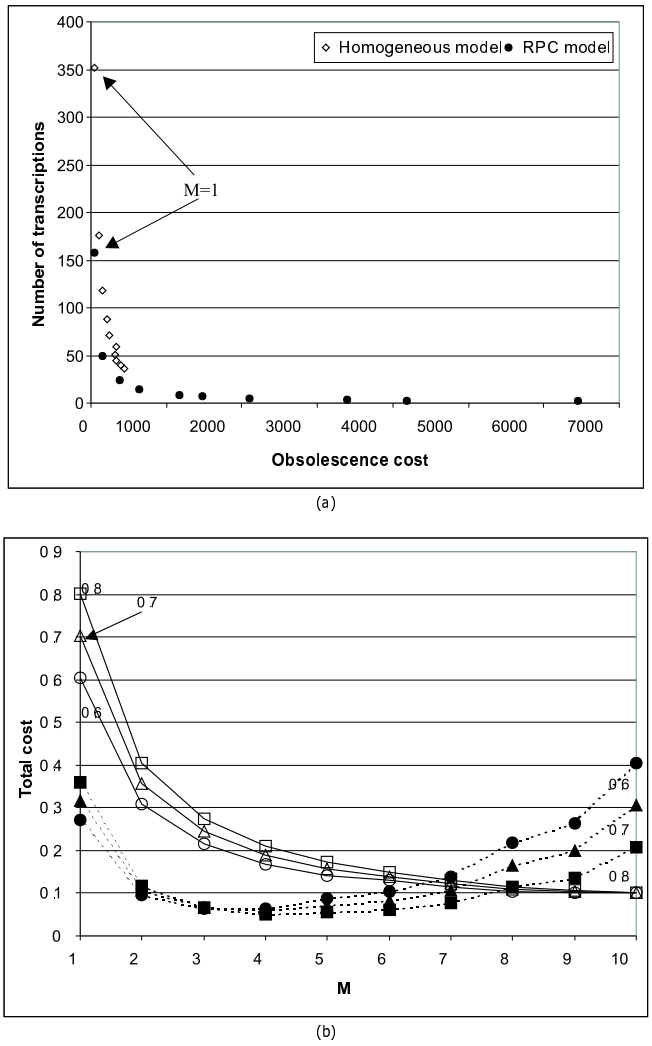


Figure 10: Threshold policy, homogeneous vs. RPC.

costs. Our conclusion is that the RPC model allows policies to cater better to users' needs, not only by tracking changing arrival intensities, but also by taking into account a user's preferences.

The FA policy We also experimented with the FA policy, comparing the performance of the homogeneous Poisson model and the RPC Poisson model. As can be seen from Figure 12(a), this policy is dominant for all $M \geq 3$. Figure 12(a) shows the results of our experiments for a user preference ratio of 3:1. Similar results were observed for other ratios. The results of the ρ-test for a 3:1 preference ratio are presented in Figure 12(b). Once again, solid lines represent the homogeneous Poisson model, while dotted lines represent the RPC Poisson model. Our conclusion is that for all M values, the RPC model is preferred over the homogeneous model.

Example overall comparison We conclude this section with a comparison of the USP policy, the threshold policy, and the FA policy for a given preference ratio (3:1) and ρ (0.6); see

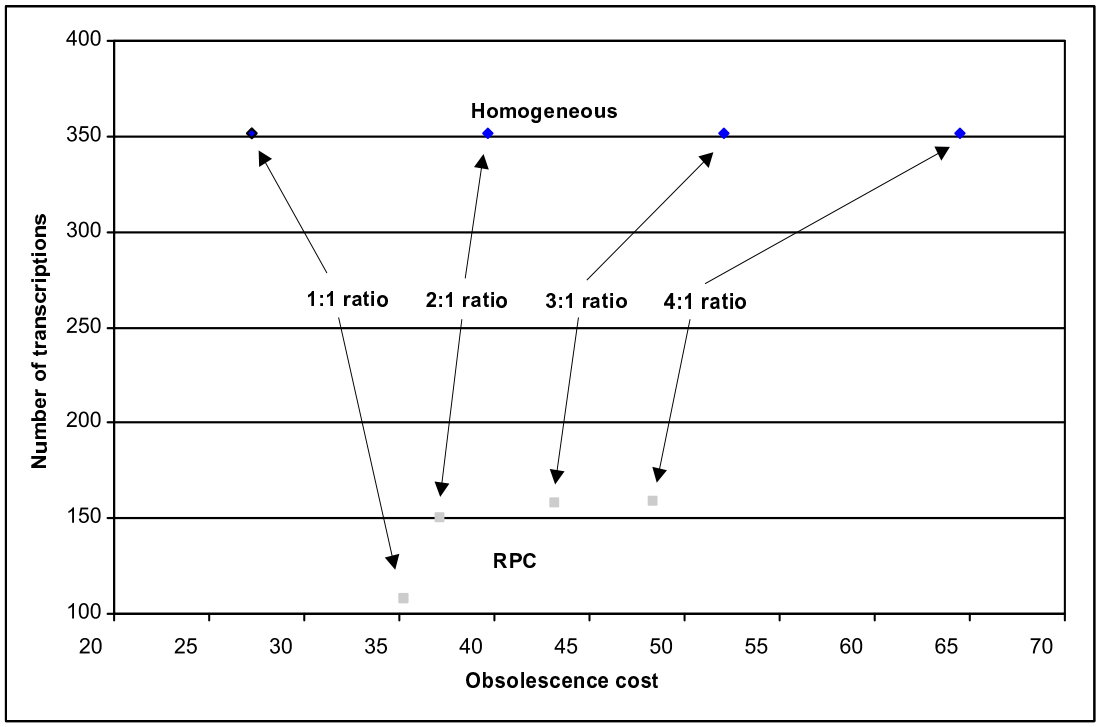


Figure 11: The impact of user preference on obsolescence and transcription costs, the homogeneous model versus the RPC model.

Figure 13. For $M \in \{1\} \cup \{5, \ldots, 10\}$, the FA policy is preferred, while the threshold policy is preferred for $M \in \{2, 3, 4\}$. The best policy for this choice of a_1/a_2 and ρ is the threshold policy with $M = 4$. We have conducted our experiments with various ρ values and our conclusion is that the RPC-based models hug the efficient frontier better than the homogeneous model.

5 Model applicability

This section focuses on the possible applications of the model. We envision four major areas in data management, where such a model can be applied. The common characteristic to all four areas is the network-centric environment, to which data sources connect according to some policy (may it be centralized or autonomous) to perform activities such as data posting and data transcription. In the process, data is transcribed from some primary data source to a secondary data source. Once this process is completed, the data in the primary data source may evolve independently of the transcribed data. Thus, between transcriptions, data in secondary sources may become obsolescent. Applications include:

Data warehouses: A data warehouse serves in consolidating data from throughout the organization. A data warehouse is typically updated during a night window by transcribing data from related transactional databases. The transcribed data is further processed to fit the data warehouse data structure. The transcription process in data warehouses is time-consuming and may be disruptive to the ordinary operation of the transaction system. The model

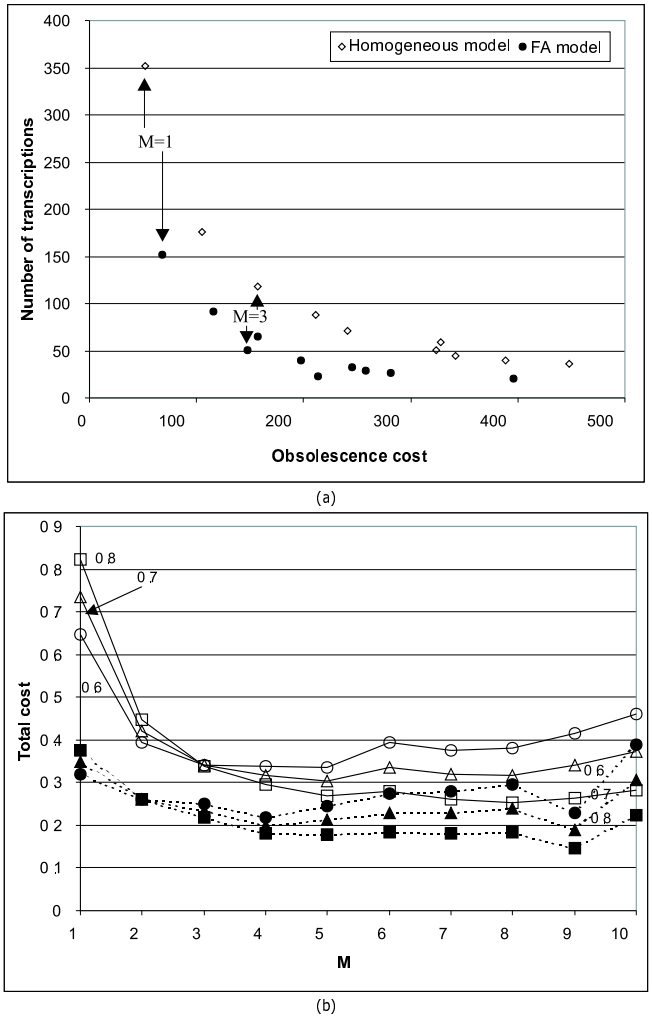


Figure 12: FA policy, homogeneous vs. RPC.

described in this paper can efficiently estimate how often the data warehouse needs to be refreshed. Moreover, consider a query submitted by a user who has some tolerance for obsolescent information. In building a query plan, the optimizer may have a choice between plans that use large amounts of network or computation resources to access the base data sources, and cheaper plans that use cached, possibly obsolescent data. We envision an extension of the proposed model to assist with query constructions.

Web crawlers and indexing engines: Web crawlers are automated processes, collecting information about the availability of resources on the Web on behalf of search engines and transcribing a concise version of their data for indexing purposes. Some data sources change rapidly, while some are more stable. Also, some tend to change at certain times of the day or week. Finally, some may take more time to access than others. With the assistance of the proposed model, the scheduling of Web crawling can be designed far more efficiently.

Periodically connected devices: In a pervasive environment [20], mobile devices periodically poll a central server to transcribe new relevant information. Mobile devices periodically poll

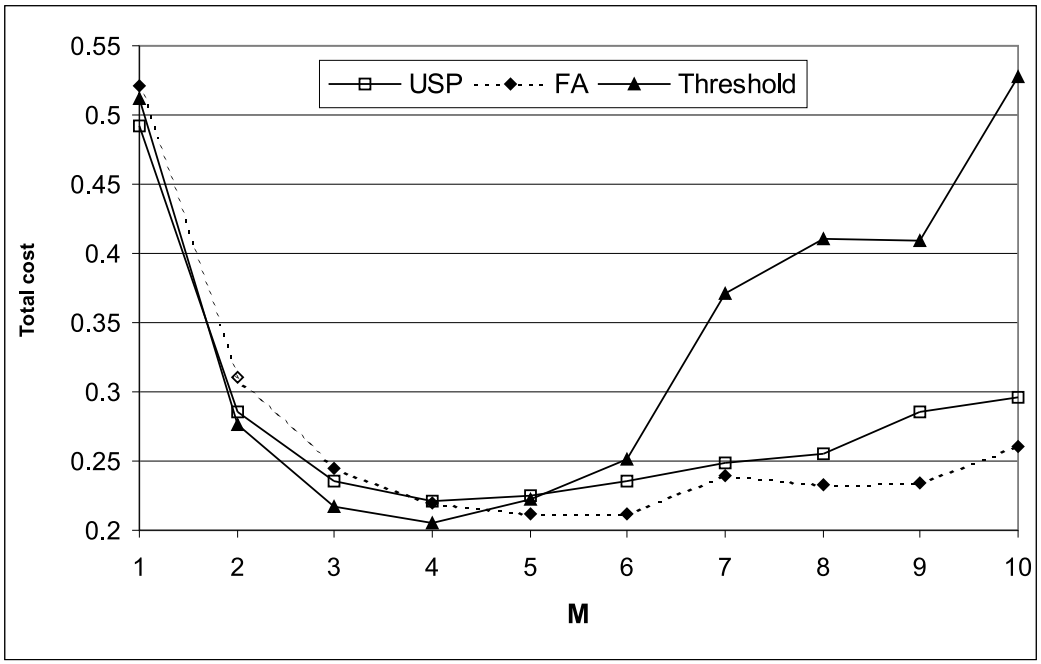


Figure 13: Comparison of three policies, for 3:1 preference ratio and ρ=0.6.

a central server for new information relevant to their owners. Communication resources (for example, a satellite-based network) are scarce and must be conserved. Using the proposed model, devices could efficiently time their polling of the server.

Web caches: The bottleneck of delivering Web pages involves primarily the network latency whenever distant information sources are involved. Caching solutions, where copies of previously requested pages are stored locally, were suggested to reduce delays and traffic and it has been shown that caching documents can improve Web performance significantly. The decision all cache managers face boils down to whether a cached document can be sent to the client or the request should be directed to the server. While the current solution involves heuristics such as Time-To-Live, the proposed model could enhance performance, by taking into account the rate of change of Web pages on the one hand and the client's tolerance towards data obsolescence on the other hand.

6 Conclusion

We have proposed a general model for data evolution in append-only environments, based on nonhomogeneous compound Poisson models. Two simple derivatives of the model, namely the homogenous Poisson Model and the recurrent piecewise-constant Poisson model were discussed and compared, both theoretically and using real data feeds. Our conclusion is that the recurrent piecewise-constant Poisson model can better model real-life applications with cyclic patterns of arrival behavior than its homogeneous counterpart. Such behavior exists in many applications, including call centers, reservations systems, and retailing. It is our belief that such models are also applicable in the general areas of data warehousing, Web crawling, and pervasive computing.

As discussed throughout the paper, obsolescence is a subjective metric, to be tailored to individual users. We provide a user with three complementary mechanisms in defining her preferences. First, ρ is a ratio of importance a user assigns with obsolescence versus transcription costs. Second, the function $a(\tau)$ allows a user to define a time-varying importance function to be assigned with the arriving data. Finally, for the FA policy, a user can set the probability of first arrival (π) as a threshold for transcription.

Several topics are left for future research. We are experimenting with fitting the arrival data with smoother forms of $\lambda(t)$, such as continuous piecewise-linear, or even second-order smooth cubic splines. These experiments will be discussed in a future paper. Also, the issue of autocorrelation or "burstiness" of arrivals needs to be investigated further, based on research results such as [28, 41], and a tractable model of autocorrelative effects should be designed.

Pragmatically, however, statistically invalid models can still be useful if they model reality better than the currently available alternatives. Initial experiments with the World Cup data, to be described in a future paper, compared a threshold policy based on the RPC model to other common heuristics for data caching (such as TTL [15]). The threshold-RPC approach appeared to yield better performance than the standard caching algorithms, although its underlying stochastic model can be statistically rejected with a high degree of confidence.

Another related topic has to do with the appropriate form of a cost function, taking into account real users' preferences. One could then attempt to construct policies designed to optimize this cost function. It may also be possible, given α and a_1/a_2, to use numerical optimization techniques to construct a weekly transcription schedule that (approximately) optimizes the cost function.

Another area for further research involves the monitoring of changes in the model's parameters and re-validating them when the need arises. Towards this end, one may consider similar problems of monitoring, filtering, and segmentation of process parameters, such as the arrival rate and its intensity levels, with the objective of developing effective algorithms for convenient off- and on-line practical implementation.

Several assumptions in this model should be further investigated. For example, we assume that at any connectivity period, data is refreshed in full. However, a connectivity period may terminate before the data is fully refreshed. Such a situation needs to be considered when scheduling the next transcription. Finally, in the broader context of data warehouses and transcription of more complex databases, we must attempt to model not only insertion, but also deletions and modifications. While a theoretical framework for these models was laid out in [17], we still need to verify its validity using real data feeds.

Acknowledgments We would like to thank Ben Melamed, Bob Vanderbei, and Themistoklis Palpanas for their help. Also, the assistance of Louiqa Raschid and Laura Bright with the analysis of the World Cup data is highly appreciated.

References

[1] S. Abiteboul. On views and XML. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 1–9, 1999.

[2] S. Abiteboul and O.M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 254–263, 1998.

[3] S. Acharya, M.J. Franklin, and S.B. Zdonik. Balancing push and pull for data broadcast. In J. Peckham, editor, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 183–194, Tucson, Arizona, May 1997. ACM Press.

[4] R. Alonso, D. Barbara, and H. Garcia-Molina. Data caching issues in information retrieval system. *ACM Transactions on Database Systems (TODS)*, 15(3):359–384, September 1990.

[5] T.A. Anderson, Y. Breitbart, H.F. Korth, and A. Wool. Replication, consistency, and practicality: Are these mutually exclusive? In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 484–495, Seattle, Washington, June 1998. ACM Press.

[6] L. Bright and L. Raschid. Using latency-recency profiles for data delivery on the web. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, 2002.

[7] P.J. Brockwell and R.A. Davis, editors. *Time Series: Theory and Methods*. Springer-Verlag, 1987.

[8] M.J. Carey, M.J. Franklin, M. Livny, and E.J. Shekita. Data caching tradeoffs in client-server DBMS architectures. In J. Clifford and R. King, editors, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 357–366, Denver, Colorado, May 1991. ACM Press.

[9] S. Chakravarthy. Just-in-time information: To push or not to push. In Brian Lings and Keith G. Jeffery, editors, *Advances in Databases, 17th British National Conferenc on Databases, BNCOD 17, Exeter, UK, July 3-5, 2000, Proceedings*, volume 1832 of *Lecture Notes in Computer Science*. Springer, 2000.

[10] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proceedings of the IEEE CS International Conference on Data Engineering*, pages 190–200, Taipei, Taiwan, 1995.

[11] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 117–128, Dallas, Texas, May 2000.

[12] L.S. Colby, T. Griffin, L. Libkin, I.S. Mumick, and H. Trickey. Algorithms for deferred view maintenance. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 469–480, Montreal, Quebec, Canada, June 1996. ACM Press.

[13] L.S. Colby, A. Kawaguchi, D.F. Lieuwen, I.S. Mumick, and K.A. Ross. Supporting multiple view maintenance policies. In Joan Peckham, editor, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 405–416, Tucson, Arizona, May 1997. ACM Press.

[14] A. Delis and N. Roussopoulos. Techniques for update handling in the enhanced client-server DBMS. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 10(3):458–476, 1998.

[15] R. Fielding, J. Gettys, J.C. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol - http/1.1. Technical Report RFC 2616, June 1999. http://www.rfc-editor.org/rfc/rfc2616.txt.

[16] A. Gal. Obsolescent materialized views in query processing of enterprise information systems. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 367–374, Kansas City, MI, 1999.

[17] A. Gal and J. Eckstein. Managing periodically updated data in relational databases: A stochastic modeling approach. *Journal of the ACM*, 48(6):1141–1183, 2001.

[18] S. Grumbach and L. Tininini. On the content of materialized aggregate views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 47–57, 2000.

[19] R.V. Hogg and E.A. Tanis. *Probability and Statistical Inference*. MacMillan, New York, second edition, 1983.

[20] A.C. Huang, B.C. Ling, and S. Ponnekanti. Pervasive computing: What is it good for? In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 84–91, Seattle, WA, August 1999.

[21] T.L. Lai. Sequential changepoint detection in quality control and dynamical systems (with discussion). *Journal of the Royal Statistical Society, Ser. B*, 57:613–658, 1995.

[22] T.L. Lai. Information theoretic bounds on quick detection of parameter changes in stochastic systems. *IEEE Transactions on Information Theory*, 44:2917–2929, 1998.

[23] T.L. Lai and J.Z. Shan. Efficient recursive algorithms for detection of abrupt changes in signal and control systems. *IEEE Transactions on Automatic Control*, 44:952–964, 1999.

[24] L.V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Transactions on Database Systems (TODS)*, 22(3):419–469, 1997.

[25] 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. Conference on Web Information Systems Engineering (WISE)*, 2002.

[26] A. Levy, A.O. Mendelzon, and Y. Sagiv. Answering queries using views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 95–104, San Jose, May 1995.

[27] B.G. Lindsay, L.M. Haas, C. Mohan, H. Pirahesh, and P.F. Wilms. A snapshot differential refresh algorithm. In Carlo Zaniolo, editor, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 53–60, Washington, D.C., May 1986. ACM Press.

[28] M. Livny, B. Melamed, and A.T. Tsiolis. The impact of autocorrelation on queuing systems. *Management Science*, 39(3):322–399, 1993.

[29] G. Lohman, C. Mohan, L. Haas, D. Daniels, B. Lindsay, P. Selinger, and P. Wilms. Query processing in R^*. In W. Kim, D. Reiner, and D. Batroy, editors, *Query Processing in Database Systems*. Springer-Verlag, NY, 1985.

[30] R.L. Mason, Y.-M. Chou, J.H. Sullivan, Z. G. Stoumbos, and J. C. Young. Systematic patterns in T^2 charts. *Journal of Quality Technology*, 35:47–58, 2003.

[31] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 1992.

[32] K. Nishina. A comparison of control charts from the viewpoint of change-point estimation. *Quality and Reliability Engineering International*, 8:537–541, 1992.

[33] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 144–155, Cairo, Egypt, September 2000. Morgan Kaufmann.

[34] M. Rabinovich, N.H. Gehani, and A. Kononov. Scalable update propagation in epidemic replicated databases. In P.M.G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 1996.

[35] M.R. Reynolds, Jr. and Z.G. Stoumbos. A general approach to modeling CUSUM charts for a proportion. *IIE Transactions on Quality and Reliability Engineering*, 32:515–535, 2000.

[36] S. Ross. *Introduction to Probability Models*. Academic Press, 1980.

[37] S. Ross. *Stochastic Processes*. Wiley, second edition, 1995.

[38] Z.G. Stoumbos. The detection and estimation of the change point in a discrete-time stochastic system. *Stochastic Analysis and Applications*, 17:637–649, 1999.

[39] Z.G. Stoumbos, M.R. Reynolds Jr., and W.H. Woodall. Control chart schemes for monitoring the mean and variance of processes subject to sustained shifts and drifts. *to appear in the Handbook of Statistics: Statistics in Industry*, 22, 2003. C.R. Rao and R. Khattree (eds.).

[40] Z.G. Stoumbos and M.R. Reynolds, Jr. Control charts applying a sequential test at fixed sampling intervals. *Journal of Quality Technology*, 29:21–40, 1997.

[41] Z.G. Stoumbos and M.R. Reynolds, Jr. Robustness to non-normality and autocorrelation of individuals control charts. *Journal of Statistical Computation and Simulation*, 66:145–187, 2000.

[42] Z.G. Stoumbos and M.R. Reynolds, Jr. Economic statistical design of adaptive individuals control schemes for monitoring the process mean and variance. *to appear in Nonlinear Analysis: Real World Applications*, 2003.

[43] Z.G. Stoumbos, M.R. Reynolds, Jr., T.P. Ryan, and W.H. Woodall. The state of statistical process control as we proceed into the 21st century. *Journal of the American Statistical Association*, 95:992–998, 2000.

[44] Z.G. Stoumbos and J.H. Sullivan. Robustness to non-normality of the multivariate EWMA control chart. *Journal of Quality Technology*, 34:260–276, 2002.

[45] H.M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1994.

[46] H.Z. Yang and P.A. Larson. Query transformation for PSJ-queries. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 245–254, 1987.

[47] E. Yashchin. Estimating the current mean of a process subject to abrupt changes. *Technometrics*, 37:311–323, 1995.

[48] E. Yashchin. Change-point models in industrial applications. *Nonlinear Analysis*, 30:3997–4006, 1997.