Modeling TTL-based Internet Caches

Jaeyeon Jung, Arthur W. Berger and Hari Balakrishnan
MIT Laboratory for Computer Science
Cambridge, MA 01239, USA
E-mail:{jyjung,awberger,hari}@lcs.mit.edu

***Abstract*— This paper presents a way of modeling the hit rates of caches that use a time-to-live (TTL)-based consistency policy. TTL-based consistency, as exemplified by DNS and Web caches, is a policy in which a data item, once retrieved, remains valid for a period known as the "time-to-live". Cache systems using large TTL periods are known to have high hit rates and scale well, but the effects of using shorter TTL periods are not well understood. We model hit rate as a function of request arrival times and the choice of TTL, enabling us to better understand cache behavior for shorter TTL periods. Our formula for the hit rate is closed form and relies upon a simplifying assumption about the inter-arrival times of requests for the data item in question: that these requests can be modeled as a sequence of independent and identically distributed random variables. Analyzing extensive DNS traces, we find that the results of the formula match observed statistics surprisingly well; in particular, the analysis is able to adequately explain the somewhat counterintuitive empirical finding of Jung *et al.* [1] that the cache hit rate for DNS accesses rapidly increases as a function of TTL, exceeding 80% for a TTL of 15 minutes.**

I. Introduction

Caching is one of the oldest techniques for improving performance in computer systems; by storing information locally, caches typically enhance performance by reducing access latency to data source and by reducing bandwidth requirements at the data source. Internet systems employ caching in abundance—the Web and the Domain Name System (DNS) [2] are two important examples of systems that use caching for these reasons, and they derive significant benefits from doing so.

Caching mechanisms in Internet systems are designed to scale to large numbers of caches. A common way of achieving scalable caching is to use *time-to-live (TTL)-based caches*, which work as follows: for any data item D, the site that maintains the current, authoritative version of D is called the *origin*. If the origin receives a request for D at time t it returns the current version along with a TTL period, T. The *requestor*, which is a cache used by one or more clients or client caches, is allowed to cache D. Any subsequent requests at the requestor in the time interval $(t, t+T)$ can instead be served from the requestor's cache *without* contacting the origin site. However, the first request after time $t+T$ *must* go to the origin site, since the TTL has expired for D in the requestor's cache.

TTL-based caching scales well because origin sites neither have to maintain any per-requestor (*i.e.*, per-cache) state, nor even have to know of the existence of caches. This also enables "opportunistic caching" by caches across the Internet, since they don't need to inform the origin that they are caching data. The trade-off in TTL-based caching is between consistency and scalability—because the origin does not invalidate cached content, clients may occasionally receive outdated data until the previously advertised TTL expires.

A significant number of cache misses in Internet systems that employ TTL-based caching occur because of TTL expiration. For DNS caches, essentially all misses occur because of TTL expiration or first access to a domain name. Capacity misses are negligible, as storage is ample given the size and number of DNS cache entries. This is also true for Web data that change frequently with time (e.g., news headlines, sports scores, tickers, etc.). We only consider read-only data as this reflects common workload patterns typical in Web and DNS accesses. Note that these caches don't exhibit "conflict" misses caused by multiple concurrent writes.

In previous work [1], we conducted an extensive trace-driven study of DNS cache performance driven by large client-side TCP and DNS traces. We found a surprising result: that the cache hit rate was over 80% for a TTL of 15 minutes and that the hit rate improved by only 17% (to 97%) when the TTL was 24 hours as opposed to 15 minutes! The small difference in hit rates for the large increase in TTL is rather counterintuitive since it seems reasonable to expect many more accesses to any given origin site in 24 hours than in 15 minutes, and since (only) the first access after the TTL expires actually causes a cache miss. Other researchers have similarly observed dramatically diminishing marginal returns from increasing TTLs in a separate but related domain: Web caching [3], [4], [5], [6], [7].

We speculated that this incremental improvement in cache hit rates reported in [1] had to do with the nature of accesses to the origin site from the clients sharing a cache. Motivated by this observation, we seek to answer the following fundamental question in this paper: *How does the cache hit rate for TTL-based Internet caches depend on the statistics of data accesses and TTL of data items?*

Somewhat surprisingly, we find little previous research (except recent work by Cohen *et al.* [6], [7], discussed in Section II) devoted to answering this question, despite its importance in understanding how well Web and DNS caches work. Building on the observation that the first access after TTL expiration incurs a cache miss, while all subsequent accesses until the next expirations hit in the cache, we develop an analytic model for the cache hit rate based on a renewal process that answers the above question. We then describe

a computationally tractable way to come up with numerical solutions for the model and show that our model predicts hit rates remarkably well. An important component of our model is the inter-arrival distribution of accesses to the cache for a given, arbitrary data item; we find that among the several models we considered, a Pareto distribution with a point mass is a good fit for DNS queries.

In Section II we survey related work. Section III describes our analytic model and Section IV presents numerical simulations that show that our model predicts observed cache hit rates for three different datasets accurately. We conclude with a summary of our findings and suggestions for future work in Section V.

II. Related Work

A. Modeling Cache Hit Rates

There has been a good deal of research attention on determining cache hit rates as a function of various cache replacement policies, such as *least recently used* (LRU) and *first in first out* (FIFO) [8], [9].

In contrast, there has been relatively little work on hit rates as a function of the time-to-live period and the distribution of request arrival times. Focusing on the impact of aging on caching performance, Cohen and Kaplan developed a simple cache hit model for three specific inter-request time distributions — fixed-frequency, Poisson, and Pareto, and derived the miss rates under different cache configurations. In their work, a cache may retrieve a data item from other caches as well as from the origin site. The *age* of a cached copy is defined as the elapsed time since fetched from the origin and it is deducted from its origin TTL when the cached copy is passed over to other caches [6].

In the subsequent work [7], Cohen *et al.* further explored the aging issue and demonstrated the relation of the miss rate at a client cache to the distribution of TTL across data sources. They presented analytic results of the miss rate for a Poisson and fixed-frequency inter-request time distribution.

We do not consider a multi-level cache structure in this paper. Rather we consider a single cache in which a data item is always fetched from the origin, and focus on analytic models. In comparison with Cohen *et al.*, our model provides a cache hit probability for any arbitrary inter-request time distribution.

B. Modeling TCP Connection Inter-Arrival Times

In this study, we infer the presence of a DNS lookup from the presence of the opening of a TCP connection. This approach assumes that the vast majority of connections are made by machine name, and thus the opening of each connection is preceded by a DNS lookup. This approach was first introduced in [1] in trace-driven simulations of DNS cache behavior based on the observation that TCP was a major driving application of DNS lookups. This method allows us to estimate DNS cache hit rate when varying the TTL and the degree of cache sharing using empirical distributions for the inter-query times. The possible causes of errors that may result from using TCP connections instead of real DNS lookup traces are also addressed in [1]. In the same paper, the authors made an initial attempt to explain an asymptotic behavior of DNS hit rate with a renewal assumption and showed that a heavy-tailed Pareto distribution closely fitted the distribution for TCP connection inter-arrivals. Extending that work, we include a Weibull distribution to seek a good fit, as suggested by Feldmann [10]. Feldmann studied the TCP connection arrival process for an aggregation across all destinations and calculated maximum likelihood estimators for the parameters of fitted distributions. In contrast, we focus on a TCP connection arrival process for a given, arbitrary destination. We use the `fminsearch` function in `matlab` [11], which finds a good fit using an unconstrained nonlinear optimization [12].

III. Analytic Models

In this section we present a simple analytic model for the cache query process in order to obtain a closed-form formula for the hit rate as a function of the TTL period. We then show how the equation can be transformed to use the available trace data. The resulting formula is then used to compute the hit rate from those traces and from a model of request arrivals derived from trace data. This section concludes with a discussion of the numerical computation of the hit rate from the trace-driven simulations.

A. Cache Assumptions

We develop a model for the cache hit rates that uses *time-to-live* (TTL) to control consistency as discussed in Section I. Specifically, we assume that for a given data item, the TTL value is always the same independent of where and when the data item is fetched by the cache. In other words, our model doesn't apply to a cache in which the TTL is dynamically assigned to a data item and hence could have a different value at different instances when the data item is fetched by the cache. However, if the TTL varies over some range, our model provides an upper bound on the cache hit rate, assuming the maximum value of the TTL is used. Our model also assumes that data items are only purged from the cache after the TTL expires.

B. Renewal Model for Query Process

The model considers a given, arbitrary data item (e.g., a DNS name or a Web object) and the sequence of queries to a cache for that data item. We make the simplifying assumption that the sequence of inter-arrival times of queries for the given data item can be modeled as a sequence of independent and identically distributed (i.i.d.) random variables. The i.i.d. assumption is equivalent to assuming that the inter-query times are a renewal process (i.i.d. assumption and renewal assumption may be used interchangeably herein). Reality is more complicated. One might expect that the process of inter-query times is bursty, and has positive autocorrelation. Thus, heuristically one would expect that the true hit rate should be higher than that predicted by a model based on an i.i.d. assumption. That is, one would expect that the i.i.d.

assumption is conservative, possibly very conservative, in predicting the hit rate. However, we will see in Section IV-C, that the resulting model predicts the hit rate quite well, despite of the simplification. In particular, Paxson and Floyd pointed out that i.i.d. Pareto interarrivals were useful as approximations to particular finite processes arising in wide-area network traffic [13]. Feldmann also observed that the i.i.d. Weibull model captured well the burstiness of Web connection interarrivals [10]. Both cases show that the i.i.d. assumption for the inter-query times closely approximates the underlying query arrival process of Internet caches, even if it is not statistically exact.

Let X_i be the time interval between the $i-1^{th}$ query for the given data item and the i^{th} query. Assume X_i is a *proper* random variable: X_i has a distribution function $F(x) \equiv \Pr(X_i \leq x)$, where $\lim_{x\to\infty} F(x) = 1$ (as opposed to a value less than 1). Let time $t = 0$ be chosen at the arrival of a query to the cache for which the data item is not cached, or the TTL has expired (*i.e.* the time of a cache miss).

$X_0 = 0$ and X_i's are proper, non-negative, i.i.d. random variables, X_i may have infinite mean.

Let $N(t)$ equal the number of queries for the given data item in the interval $(0, t]$. $N(t)$ is called the renewal counting process. Note that the event at $t = 0$ is excluded. Let $S_n = X_1 + X_2 + \cdots + X_n$ ($S_0 = 0$), and

$$\begin{aligned} \Pr(S_n \leq t) &= \Pr(X_1 + X_2 + \cdots + X_n \leq t) \\ &= F^{(n)}(t), \end{aligned} \quad (1)$$

where $F^{(n)}(t)$ denotes the n^{th}-fold convolution of the distribution function $F(x)$ with itself.

Let the value of the time-to-live (TTL) parameter be T.

Key observation: The renewal counting process evaluated at time equal to the TTL, $N(t)\,|_{t=T}$, models the number of cache hits per cache miss, for the given data item.

Figure 1 illustrates the idea. At time $t = 0$, there is a cache miss. Subsequently, three queries occur, at times S_1, S_2, S_3, before the TTL expires at time $t = T$. These three queries are cache hits. The subsequent, fourth query at time S_4 occurs after $t = T$ and is a cache miss. Thus in Figure 1, $N(T) = 3$ and the number of cache hits per miss is 3. Note that one could reset the time origin to S_4 and the resulting process would be stochastically equivalent to the first.

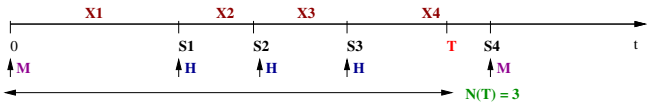


Fig. 1. Time line diagram of queries to a given data item, and associated model random variables.

In summary, using a renewal assumption, we obtain a renewal counting process, which when evaluated over the time-to-live models the number of cache hits per miss. This random variable can be used to compute other quantities of interest, such as hit and miss rates.

Selection of inter-query time distribution: Given the renewal assumption, the remaining free attribute of the model is the distribution of the inter-query time, $F(x)$. One natural choice is the empirical distribution obtained from the collected data. As the model is based on the viewpoint of picking a given, randomly selected data item, conceptually, one computes the empirical distribution for each data item of interest (say each of the data items in the data set) and then takes a weighted average of the distributions, where the weights are the fraction of queries for a given data item. An equivalent and more straightforward, calculation is to compute the relative frequency of the inter-query times for all of the data items of interest. Another natural choice for the distribution is an analytic one, such as Pareto or Weibull. Section IV considers both empirical and analytic distributions.

C. Formula for Hit and Miss Rates

In this section we derive an expression for the long-term hit and miss rates as a function of the TTL, T. Define a *cycle* as the sequence of a cache miss followed by cache hits (if any) before the TTL, T, expires, for a given data item. The cycle starts at the cache miss. The length of a cycle is defined as the time interval from the start of the cycle until the start of the next cycle. Figure 1 illustrates a cycle in which there are three cache hits, and whose length is S_4. Over the time interval $(0, u]$, one can calculate the hit rate, $H(u : T)$ as the summation of the number of hits in each cycle divided by that of the number of queries, for a given data item. Let $H(T)$ denote the limiting hit rate as $u \to \infty$ and given the TTL is T. The limiting miss rate, denoted $M(T)$, is analogous, where "cache hit" is replaced with "cache miss."

Theorem 1: If the inter-query times X_i's to a given data item are proper, non-negative, independent and identically distributed random variables, whose mean may be infinite, then

$$\begin{aligned} H(T) &= \frac{E[N(T)]}{E[N(T)]+1} \text{ and} \\ M(T) &= \frac{1}{E[N(T)]+1} \text{ with probability one.} \end{aligned} \quad (2)$$

Remark: For any finite time u, the hit and miss rates have a complicated distribution. However, due to the strong law of large numbers, in the limit as $u \to \infty$, the distribution simplifies to a single point mass.

Equation (2) is heuristically appealing: the hit rate equals the mean number of hits per miss divided by the same quantity plus the one cache miss. Equivalently, if one thinks of an episode, or a cycle, then Equation (2) says that the hit rate equals the mean number of hits in a cycle divided by the mean number of queries (one miss plus hits).

The proof of the theorem is standard, with a few subtle points, and is given in the appendix.

D. Calculation of Hit Rates

The following two sections show how to calculate the hit rate in the renewal model and the trace-driven simulation, respectively. The equations derived herein are used to compute the numerical results in Section IV-C.

1) Calculation of Hit Rate in Renewal Model: The computation of the hit rate over the interval $(0, T]$ in the renewal model, Equation (2), requires the computation of the mean number of queries, $E[N(T)]$. The expectation of the renewal counting process is a common entity of interest in renewal theory and is called the renewal function: $m(t) \equiv E[N(t)]$.

$m(t)$ can be expressed as

$$m(t) = \sum_{n=0}^{\infty} F^{(n)}(t),\ t \geq 0$$

As a result, the hit rate, $H(T)$ from Equation (2), can be written as a function of $F^{(n)}(t)$:

$$H(T) = \frac{\sum_{n=0}^{\infty} F^n(T)}{\sum_{n=0}^{\infty} F^n(T) + 1} \tag{3}$$

Although Equation (3) is of conceptual interest as it expresses the hit rate in terms of the inter-arrival distribution, $F(x)$, it is not a convenient form for numerical computation.

The renewal function, $m(t)$, satisfies the *renewal equation*:

$$m(t) = F(t) + \int_0^t m(t-x)dF(x) \tag{4}$$

Discretizing Equation (4) yields a numerically convenient iteration for $m(t)$; see [14] for details.

2) Calculation of Hit Rate in a Trace-Driven Simulation: In a trace-driven simulation, the computation of the hit rate is straightforward: simply count the number of queries and the number that are cache hits. However, this requires repeating the counting process for all data items in the trace, which is not computationally efficient, especially when it takes a long time to scan through a data trace. To avoid this, one can also compute the hit rate in a way that mirrors the renewal model and provides sample path realizations for the entities that the stochastic model attempts to describe. We describe this latter calculation in this section.

With the same definition of cycle in the Section III-C, let C be the number of such cycles in the simulation run[1]. Every query is in exactly one of these cycles.

Let f_n be the number of cycles in the simulation run in which there were n cache hits (n = 0, 1, 2, ...). Since every cycle contains one cache miss, f_n also equals the number of cycles in the simulation run in which there were $n+1$ queries. Thus, the total number of cache hits in the simulation can be expressed as $\sum_{n=0}^{\infty} n \cdot f_n$ and the total number of cache queries (hits + misses) can be expressed as $\sum_{n=0}^{\infty}(n+1) \cdot f_n$.

Note that there is an edge effect at the end of the simulation: multiple cycles are likely to be in progress. One could view this as negligible noise if the simulation runs over a period of time much longer than the TTLs. Alternatively, one could omit the in-progress cycles at the end of the simulation from the counters, f_ns and C.

The hit rate is defined to be the number of cache hits divided by the number of cache queries in the simulation run. So, the hit rate can be expressed as:

$$H = \frac{\sum_{n=0}^{\infty} n \cdot f_n}{\sum_{n=0}^{\infty}(n+1) \cdot f_n} \tag{5}$$

From the definitions,

$$\begin{aligned} \frac{f_n}{C} &= \text{fraction of cycles in which there are } n \text{ cache hits} \\ &= \text{fraction of cycles in which there are } n+1 \text{ queries} \\ \sum_{n=0}^{\infty} \frac{f_n}{C} &= 1. \end{aligned}$$

The above quantities provide sample path estimates for the modeled renewal counting process, where we evaluate $N(t)$ at $t = T$. Recall that $N(T)$ is a random variable representing the number of cache hits in a randomly chosen cycle for a random data item, given the TTL is T. Based on the trace-driven simulation:

$$\Pr(N(T) = n) = \tfrac{f_n}{C}$$

The sample mean of $N(T)$, denoted $\overline{N(T)}$, is

$$\overline{N(T)} = \sum_{n=0}^{\infty} n \cdot \frac{f_n}{C}. \tag{6}$$

The hit rate, Equation (5), can be expressed in terms of $\overline{N(T)}$. Dividing the numerator and denominator of (5) by C and substituting in (6) yields:

$$H = \frac{\sum_{n=0}^{\infty} n \cdot \frac{f_n}{C}}{\sum_{n=0}^{\infty}\left(n \cdot \frac{f_n}{C} + \frac{f_n}{C}\right)} = \frac{\overline{N(T)}}{\overline{N(T)} + 1} \tag{7}$$

In summary, for computing the hit rate in the trace-driven simulation one can directly count the number of queries and the number of cache hits. A numerically equivalent calculation is to use Equation (5), which in turn is equivalent to (7). None of these calculations makes any i.i.d. assumptions about random variables.

One can then compare these results with a calculation that does make the i.i.d. assumption: calculate the empirical distribution for the inter-query time $F(x)$ and apply the iteration of Equation (4) and substitute into Equation (2). One can also use Equations (4) and (2) to compute the hit rate for candidate analytic inter-query time distributions. The next section discusses these calculations.

IV. Numerical Results

The previous section described an analytic model of the hit rates for TTL-based Internet caches. This section begins with a discussion on the analytic models of a TCP connection arrival process which generates DNS queries, thus providing a useful estimator for $F(x)$ in Section III-B. Using DNS as an example system, we then present the hit rates calculated in three different ways: using trace-driven simulation, using the renewal assumption with the empirical distribution of $F(x)$

[1]Note at a given point in time, multiple cycles can be in progress

obtained from the data set, and using the renewal assumption with an analytic distribution. Finally, we evaluate our analytic model developed in Section III by comparing the above hit rates and exploring the gap resulting from the renewal assumption and an approximate model of $F(x)$.

A. The Data

We use three separate traces collected at the border gateway of MIT's Laboratory for Computer Science (LCS) and Artificial Intelligence Laboratory (AI) and at a link that connects Korea Advanced Institute of Science and Technology (KAIST) to the rest of the Internet. The first trace, `mit-jan00` was collected from 3 January to 10 January 2000; the second, `mit-dec00` was collected from 4 December to 11 December 2000. Both were collected at the same point in MIT. The third set, `kaist-may01` was collected at KAIST from 18 May to 24 May 2001. Each data trace recorded over 3 million TCP outgoing connections generated from over 900 clients for over 30 thousand different destinations. A detailed description of the traces is available in [1].

With the same data sets, Jung *et al.* [1] estimated the DNS cache hit rates inside the traced networks by trace-driven simulations. Assuming that TCP was a major application that drove DNS lookup sequences, they used TCP connections to model cache references, and conducted simulations showing the impact of a time-to-live parameter on the DNS cache hit rate. In this section, we evaluate our renewal model by calculating a hit rate using the methodology discussed in Section III-D.1 and compare it with the one from the trace-driven simulations where the hit rate is simply the number of cache hits divided by the number of queries. Computing the hit rate via Equations (4) and (2) requires the inter-query time distribution, $F(x)$, which can be obtained either empirically or analytically from a given data trace.

To deduce the distribution of the inter-query time at a cache from the real traffic, we calculate the time difference between two consecutive connection arrivals for a given destination IP address, denoted as x, and count the frequency of x across all pairs of such occurrences. For all three traces, x spans 9 orders of magnitude, ranging from 10^{-3} to 10^{6} seconds. It is also noticeable that there is a jump at time 1 ms which suggests that there are a number of connections back-to-back in very close succession.

Table I lists the statistics of each data set including the median, the mean, $E[x]$, the 95^{th}-percentile, and the standard deviation, σ_x. Due to a heavy tail, $E[x]$ and σ_x are skewed by large values. For instance, $E[x]$ is more than 400 times larger than the median. To reduce the scale, we transform data using a natural log, and the corresponding mean, $E[\ln x]$ and standard deviation, $\sigma_{\ln x}$, are listed in Table I.[2]

TABLE I
STATISTICS FOR TCP CONNECTION INTER-START TIMES FOR A GIVEN DESTINATION

	`mit-jan00`	`mit-dec00`	`kaist-may01`
Median	4 (sec)	7 (sec)	1 (sec)
$E[x]$	1977 (sec)	2814 (sec)	325 (sec)
95^{th}-percentile	3913 (sec)	5932 (sec)	173 (sec)
σ_x	14348 (sec)	19587 (sec)	5024 (sec)
$E[\ln x]$	9.0 (msec)	9.1 (msec)	6.7 (msec)
$\sigma_{\ln x}$	3.9 (msec)	3.8 (msec)	3.3 (msec)

B. Analytic Models of TCP Connection Inter-start Time

In this section we derive analytic models describing the distributions for TCP connection inter-start time for three data sets, which is a good approximation for and can substitute for the analytic inter-query time distribution, $F(x)$, defined in Section III-B.

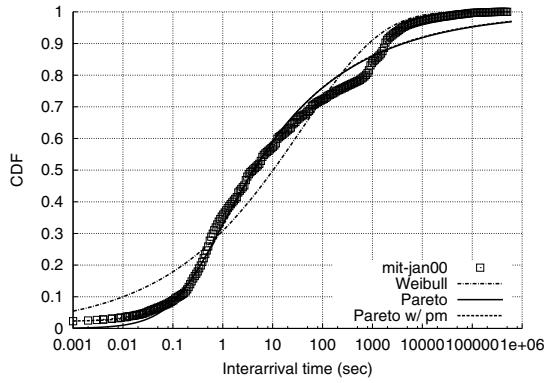
A number of well-known probability distributions were considered as candidates for the distribution function $F(x)$ for each data trace, including an exponential, a Normal, a Pareto, a Weibull, a log Normal, a log Pareto, a Pareto with a point mass, and a Weibull with a point mass. For the sake of brevity, we report the results for the three best performing choices — a Weibull, a Pareto, and a Pareto distribution with a point mass, all of which capture a heavy tail feature of the TCP connection inter-start time distribution.

Parameter estimation is done with `matlab` using an unconstrained nonlinear optimization [11]. Figure 2 illustrates the fitted distributions and estimated parameters along with the empirical cumulative distribution, shown as square dots.

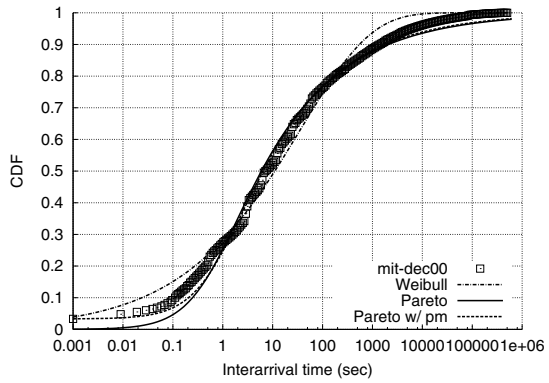
The fitted Weibull, $W(x)$, captures a spike at $t = 0.001$ (sec) while the fitted Pareto, $P(x)$, fits well starting from $t = 0.1$ (sec). The fitted Pareto, however, estimates a heavy tail better than the Weibull, whose distribution has a decreasing exponential term. As a result, the Weibull distribution approaches 1 faster than the empirical distribution. With a point mass at $t = 0.001$ (sec), the second fitted Pareto, $\widetilde{P(x)}$, results in a better fit *both* for the beginning and the tail as shown in Figure 2.

This observation can be confirmed using the discrepancy measure, $\hat{\lambda^2}$ [15] [16]. $\hat{\lambda^2}$ is a modified chi-squared test that enables us to compare discrepancies for different values of the number of bins. Paxson [16] uses the result of Scott [17] in modeling wide-area connection inter-arrival. With fixed-sized bins the mean-square error is minimized with a bin width given by $w = 3.49\sigma_x n^{-1/3}$, where n is the number of samples. Both n and the corresponding w are listed in Table II.

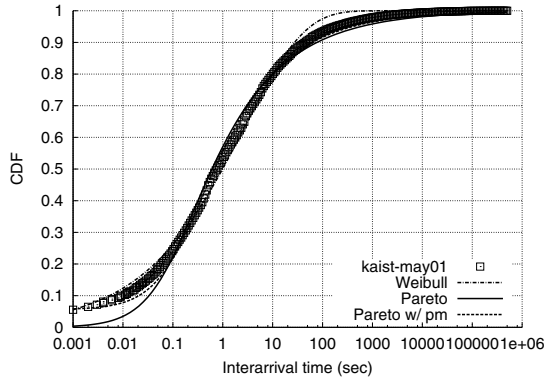
[2] x is measured in a granularity of millisecond and the smallest value is 1 (ms) for the log transformation.



(a) mit-jan00 (Weibull: d = 39227, c = 0.27; Pareto: a = 0.23, k = 223; Pareto w/ pm: a = 0.24, k = 274, w = 0.023)



(b) mit-dec00 (Weibull: d = 36306, c = 0.31; Pareto: a = 0.28, k = 540; Pareto w/ pm: a = 0.29, k = 735, w = 0.033)



(c) kaist-may01 (Weibull: d = 2729, c = 0.36; Pareto: a = 0.34, k = 92; Pareto w/ pm: a = 0.37, k = 153, w = 0.056)

Fig. 2. Fitted Weibull $W(x) = (1 - e^{-\frac{x}{d}^c})$, Pareto $P(x) = \left(1 - \left(\frac{k}{x+k}\right)^a\right)$, and Pareto distribution with a point mass $\widetilde{P(x)} = w + (1-w) \times \left(1 - \left(\frac{k}{x+k}\right)^a\right)$ for TCP connections inter-arrivals

TABLE II

BIN-WIDTH w TO MINIMIZE ERROR IN APPROXIMATING A DISTRIBUTION USING FIXED-SIZED BINS

	mit-jan00	mit-dec00	kaist-may01
n	3571746	4483468	6304553
w	0.089	0.080	0.062

Table III shows the goodness-of-fits for three different fitted distributions with fixed-size bins where a bin size is determined as shown in Table II. Lower values indicate a better fit.[3] For all data sets, the fitted Pareto with a point mass yields a better match to the empirical distribution than the other two distributions.

Complementing the plots of the distribution functions and the numerical goodness-of-fit results, Figure 3 provides a visual comparison via a histogram of the data sets along with a histogram calculated from the three fitted analytic distributions. First, we transform the x value into a log scale and count the number of samples falling into each range of size w from the empirical data. For an interval $(x_1, x_2]$, a histogram is calculated using the cumulative distribution's values at x_2 and x_1. Frequent spikes and the shape of humps existing in the empirical data make it hard to yield a good fit over the entire range.

TABLE III

INTERVAL $[(\hat{\lambda^2} - \sigma_\lambda), (\hat{\lambda^2} + \sigma_\lambda)]$ FOR THE FITTED MODELS WHERE w IS THE WIDTH OF THE BINS

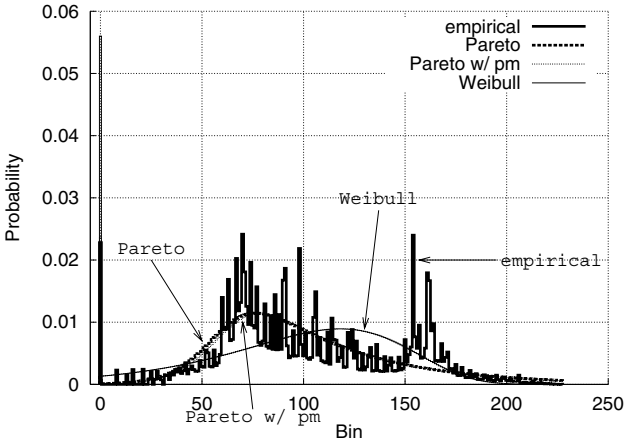
	mit-jan00	mit-dec00	kaist-may01
Weibull	0.945 - 0.952	33.142 - 34.674	364569724 - 393066504
Pareto	1.088 - 1.096	2.968 - 2.991	1.262 - 1.269
Pareto w/ point mass	0.689 - 0.693	1.283 - 1.295	0.790 - 0.796

TABLE IV

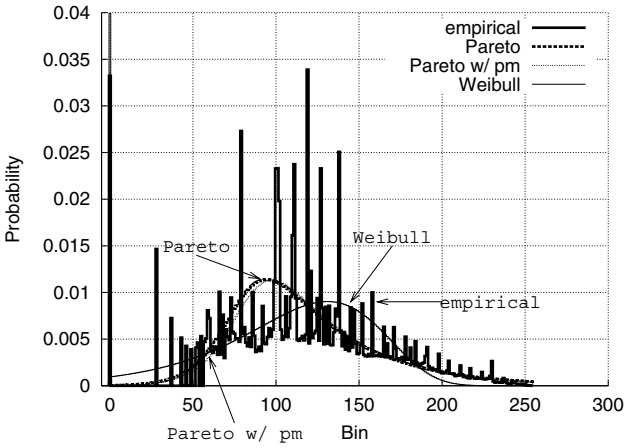
INTERVAL $[(\hat{\lambda^2} - \sigma_\lambda), (\hat{\lambda^2} + \sigma_\lambda)]$ FOR THE FITTED MODELS WHERE BIN SIZE IS DETERMINED BY FIXED Y INTERVAL (WIDTH = 0.005)

	mit-jan00	mit-dec00	kaist-may01
Weibull	0.742 - 0.746	6.250 - 6.366	14248- 14568
Pareto	0.928 - 0.936	2.396 - 2.418	0.842 - 0.847
Pareto w/ point mass	0.498 - 0.494	0.326 - 0.328	0.162 - 0.163

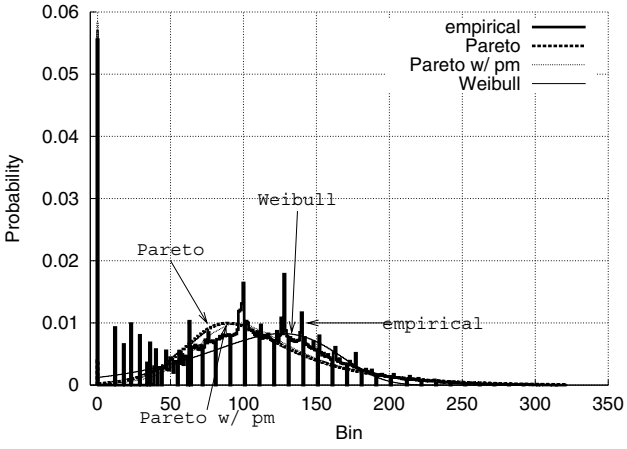
[3]We use $<_\sigma$ operator as described in [16]



(a) mit-jan00

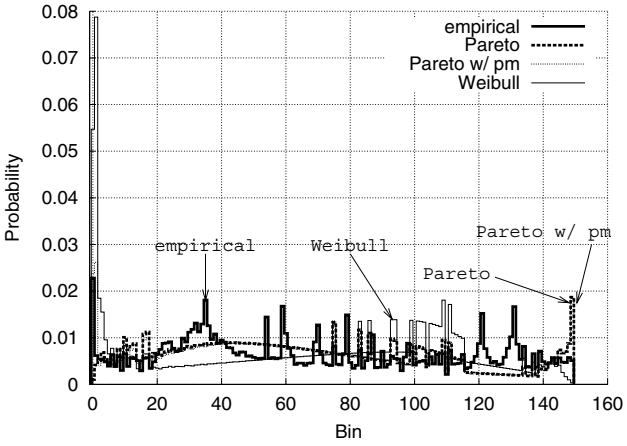


(b) mit-dec00

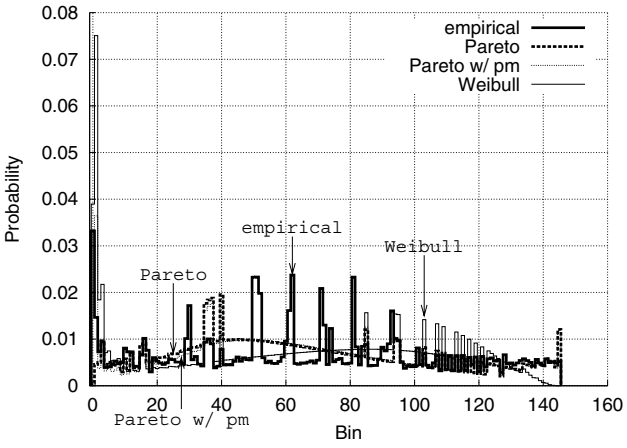


(c) kaist-may01

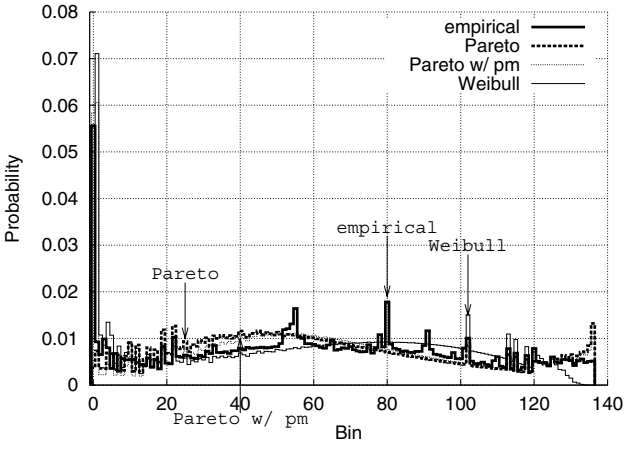
Fig. 3. A histogram of the data sets vs. a histogram of the numerical goodness-of-fit results (width = w)



(a) mit-jan00



(b) mit-dec00



(c) kaist-may01

Fig. 4. A histogram of the data sets vs. a histogram of the numerical goodness-of-fit results (fixed y width = 0.005). Note that the number of bin with non-zero sample points is less than $1/0.005 = 200$ for all three cases.

In particular, as the fitted Weibull distribution approaches 1 faster than the empirical distribution, for the larger values of x the difference in the cumulative distribution's values is small; and consequently a very small denominator of the $\hat{\lambda^2}$ results in a huge discrepancy for `kaist-may01` trace.

Instead of dividing the x range into equal-sized bins, we use an alternate approach that uses an equal-sized bin over the cumulative distribution (CDF) range. From the empirical distribution, we seek the largest CDF value of each bin and keep track of the corresponding x values.

Note that at jumps in the empirical distribution, a bin may have no sample points, and therefore can't contribute the associated x value; in this case, the bin is merged with the next adjacent one until there appears a bin with some sample points. The histogram, is then the difference of the CDF values calculated from the two adjacent x values picked from the previous scanning process.

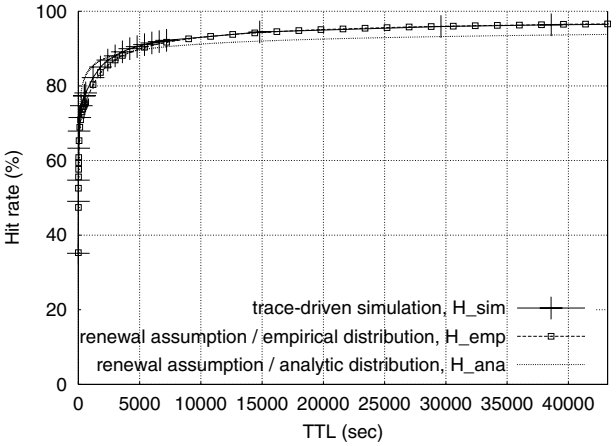
Figure 4 compares the histograms of the empirical and analytic distributions when the CDF range is partitioned into 200 equal-sized bins. With bins having roughly equal probability mass, the goodness-of-fit gets smaller for all cases (See Table III and Table IV for comparison). Figure 4 also confirms that the fitted Pareto distribution with a point mass yields the best fit of all three distributions. However, note that the parameters of the fitted Pareto differ markedly for the different data sets (Figure 2). Thus, there is no Pareto with fixed, selected parameters that uniformly fits well across our data sets.

C. Comparison of Hit Rates

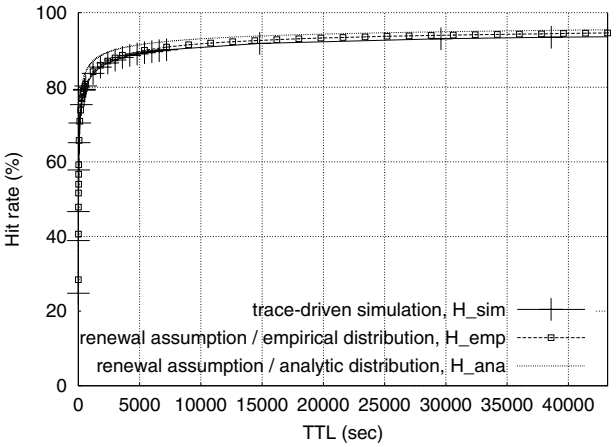
Figure 5 compares cache hit rates as a function of TTL obtained by three different methodologies. A solid line plots the hit rate, H_{sim}, from trace-driven simulations, and H_{sim} is calculated exactly in the same way described in [1]. A dotted line with a square represents the hit rate, H_{emp}, in Equation (2) calculated using the empirical distribution of the inter-query time, $F(x)$, from each trace. Lastly, a dotted line shows the hit rate, H_{ana} in Equation (2) calculated using the fitted Pareto distribution with a point mass, which describes an inter-query process with the smallest discrepancy of three candidate models, as shown in Tables III and IV.

Comparing H_{sim} and H_{emp}, we found that the renewal model worked surprisingly well. The renewal model closely follows the simulation results and the difference is less than 2% for TTL values up to 86400 (sec). This suggests the inaccuracy of the i.i.d. simplifying assumption is remarkably small with regard to the DNS query process itself. The following example shows how the renewal model explains the empirical findings observed in [1].

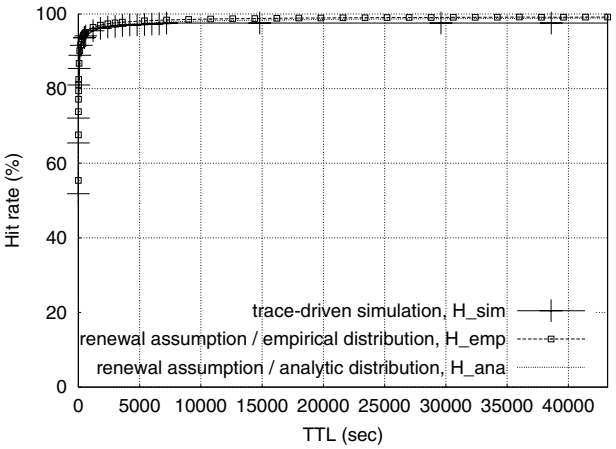
Note that for a TTL of 15 minutes (900 seconds), the hit rate is relatively high — over 80% for `mit-jan00` and `mit-dec00`, and over 94% for `kaist-may01`. This leaves room for only modest increase in the hit rate for larger TTLs. For example, for a TTL of 24 hours (86400 seconds), the hit rates are 97% for `mit-jan00`, 94% for `mit-dec00`, 98% for `kaist-may01`, respectively.



(a) `mit-jan00`



(b) `mit-dec00`



(c) `kaist-may01`

Fig. 5. Comparing DNS hit rates obtained by three different methods

From Equation (2), the hit rate will be 80% when the mean

number of cache hits per miss, $E[N(T)]$, is 4. A nominal sample path would be 4 inter-query times before the TTL expires. However, note that the mean for just one inter-query time is greater than 15 minutes, for the MIT traces (Table I). It is the high variability of the inter-query times that enables $E[N(T)]$ to be as big as 4 (and the hit rate as high at 80%).

If, in contrast, the inter-query times were rather regular (they would be deterministic in the limit), but, had the same mean, then for $T = 15$ minutes, $E[N(T)]$ would be less than 1, and the corresponding hit rate would be less than 50%, and decreasing to 0% for the case of deterministic inter-query times. For instance, if the inter-query times were exponentially distributed (i.e., a Poisson process) with mean of 2,000 seconds (corresponding to the MIT traces), then for a TTL of 15 minutes, $E[N(T)] = 900/2000 = 0.45$, and the hit rate would be only 31%.

However, when an approximate analytic model of $F(x)$ replaces the empirical distribution of the inter-query time, the resulting hit rate, H_{ana}, is less accurate. The inaccuracies are due to the complicated structure of the real inter-query time distribution, which cannot be described by the fitted model (Figures 3 and 4). But, for all three data sets, H_{ana} by the fitted Pareto with a point mass predicts the hit rate better than the two other distributions discussed in Section IV-B.

V. Conclusion

This paper has presented analytic models for the hit rate for TTL-based Internet caches in terms of the statistics of data accesses and the TTL value set by the data origin. We developed a closed-form formula for the hit rate based on a renewal assumption wherein the sequence of inter-arrival times of the queries for a given data item can be modeled as a sequence of independent and identically distributed random variables.

Analyzing extensive DNS traces, we find that our model predicts observed statistics remarkably well. In particular, it's able to adequately explain the somewhat surprising empirical finding of Jung *et al.* [1] that for DNS accesses, the cache hit rate rapidly increases as a function of TTL, quickly reaches over 80% for the 15 minute TTL, leaving room for only a modest increase in the hit rate for larger TTLs. The prediction is best when we use the observed, empirical distribution of the inter-arrival times of DNS queries. We also find that if the inter-arrival times are modeled according to an analytic distribution, a Pareto distribution is a better fit than a Weibull distribution.

In this paper, we considered a single cache shared by entire clients but our model can be easily extended to the multi-level cache structure in which the TTL of a data item is drawn from a certain distribution. In ongoing work, we are investigating the renewal-model prediction of hit rates varying the degree of client sharing as opposed to all sharing a cache. Also, in light of the good fit of the hit rate function, we are examining the inaccuracy of the i.i.d. simplifying assumption with regard to the query process itself.

Acknowledgments

The authors thank Nick Feamster, Stuart Schechter, Michael Walfish and Emre Koksal for their useful comments that improved the paper.

References

[1] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS Performance and the Effectiveness of Caching," *IEEE/ACM Transactions on Networking*, Oct. 2002.

[2] P. Mockapetris and K. Dunlap, "Development of the Domain Name System," in *Proc. ACM SIGCOMM*, Stanford, CA, 1988, pp. 123–133.

[3] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal Policies in Network Caches for World-Wide Web Documents," in *Procedings of the ACM SIGCOMM '96 Conference*, Stanford University, CA, 1996. [Online]. Available: citeseer.nj.nec.com/williams96removal.html

[4] M. Arlitt, R. Friedrich, and T. Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies," in *Performance Tools '98*, 1998. [Online]. Available: citeseer.nj.nec.com/arlitt98performance.html

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *Proceedings of the INFOCOM '99 conference*, Mar. 1999. [Online]. Available: http://www.cs.wisc.edu/~cao/papers/zipf-like.ps.gz

[6] E. Cohen and H. Kaplan, "Aging Through Cascaded Caches: Performance Issues in the Distribution of Web Content," in *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, ser. Computer Communication Review, R. Guerin, Ed., vol. 31, 4. New York: ACM Press, Aug. 27–31 2001, pp. 41–54.

[7] E. Cohen, E. Halperin, and H. Kaplan, "Performance Aspects of Distributed Caches Using TTL-Based Consistency," in *Proceedings of ICALP'01 conference*, 2001. [Online]. Available: citeseer.nj.nec.com/cohen00performance.html

[8] W. King, "Analysis of Demand Paging Algorithms," *Information Processing*, pp. 485–490, 1971.

[9] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," in *Proceedings of the ACM SIGMETRICS*, Denver, CO, 1990.

[10] A. Feldmann, "Characteristics of TCP Connection Arrivals," 1998, technical report, AT&T Labs Research. [Online]. Available: citeseer.nj.nec.com/feldmann98characteristics.html

[11] "`fminsearch` (matlab function reference)," `http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fminsearch.shtml`, 2001.

[12] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex algorithm in low dimensions," *SIAM Journal on Optimization*, vol. 9, pp. 112–147, 1998. [Online]. Available: citeseer.nj.nec.com/lagarias96convergence.html

[13] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995. [Online]. Available: citeseer.nj.nec.com/paxson95widearea.html

[14] D. Heyman and M. Sobel, *Stochastic Models in Operation Research*. McGraw Hill, 1983, vol. 1.

[15] S. Pederson and M. Johnson, "Estimating Model Discrepancy," *Technometrics*, vol. 32, no. 3, pp. 305–314, 1990.

[16] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking*, vol. 2, no. 4, pp. 316–336, 1994. [Online]. Available: citeseer.nj.nec.com/article/paxson94empiricallyderived.html

[17] D. Scott, "On Optimal and Data-based Histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[18] S. Ross, *Stochastic Processes*. New York: John Wiley & Sons, 1996, vol. 1.

APPENDIX

This section provides a proof of the theorem in Section III-C.

Let $u = 0$ be the start time of the first cycle, and index the cycles $i = 1, 2, 3, ...$ Let $C(u;T)$ denoted the number of cycles to have completed over the interval $(0, u]$, given that the TTL is T. Since the inter-query times, X_i's, are assumed to be a proper random variable, and the property of "proper" is preserved under convolution, the inter-start times of cycles are a proper random variable. Hence the associated renewal counting process, $C(u;T)$ for any finite T has the limit [18]:

$$\text{As } u \to \infty, C(u:T) \to \infty \text{ with probability one.} \quad (8)$$

Equation (8) holds even when the mean inter-query time, $E[X_i]$, is infinite.

Let $N_i(T)$ denote the number of cache hits in cycle i, given that the TTL is T. Note that $N_i(T)$ has the same distribution as the number of cache hits in cycle 1, which in the notation of the inter-query renewal process is $N(t) \mid_{t=T}$, In particular, the expectations are equal:

$$E[N_i(T)] = E[N(T)] \quad (9)$$

From the definitions,

The number of queries (hits plus misses) in the cycle i

$= \quad N_i(T) + 1$

The number of queries in cycles completed over $(0, u]$

$$= \quad \sum_{i=1}^{C(u;T)} (N_i(T) + 1)$$

$$= \quad \sum_{i=1}^{C(u;T)} N_i(T) + C(u;T)$$

Let

$H(u;T)$ = hit rate for cycles completed over the interval $(0, u]$, given the TTL is T

$M(u;T)$ = miss rate for cycles completed over the interval $(0, u]$, given the TTL is T

From the definitions,

$$H(u;T) = \frac{\sum_{i=1}^{C(u;T)} N_i(T)}{\sum_{i=1}^{C(u;T)} N_i(T) + C(u;T)} \quad (10)$$

$$M(u;T) = \frac{C(u;T)}{\sum_{i=1}^{C(u;T)} N_i(T) + C(u;T)} \quad (11)$$

$$H(u;T) + M(u;T) = 1 \quad (12)$$

Let

$H(T)$ = limiting hit rate, $\lim_{u\to\infty} H(u;T)$

$M(T)$ = limiting miss rate, $\lim_{u\to\infty} M(u;T)$

From Equation (10)

$$H(T) \equiv \lim_{u\to\infty} H(u;T) \quad (13)$$

$$= \lim_{u\to\infty} \frac{\sum_{i=1}^{C(u;T)} N_i(T)}{\sum_{i=1}^{C(u;T)} N_i(T) + C(u;T)}$$

$$= \lim_{u\to\infty} \frac{\frac{1}{C(u;T)} \sum_{i=1}^{C(u;T)} N_i(T)}{\frac{1}{C(u;T)} \sum_{i=1}^{C(u;T)} N_i(T) + 1}$$

$$= \frac{E[N(T)]}{E[N(T)] + 1} \text{with probability one,} \quad (14)$$

where the last equality is the result reported in the theorem and follows from applying the strong law of large numbers to $\frac{1}{n}\sum_{i=1}^{n} N_i(T)$, and recalling Equations (8) and (9) and noting that the function $f(x) \equiv \frac{x}{x+1}$ is continuous for $x > 0$.

Note that for any finite u, $H(u;T)$ is a random variable (that is a function of various random variables), with a not-easily-determined distribution, while the limit is a constant. All probability is in a single point mass.

The derivation for the limiting miss rate $M(T)$ is analogous.