INFORMATION RETRIEVAL ON THE INTERNET: A REALISTIC QUEUING MODEL FOR SOUJOURN TIME ANALYSIS

A. B. WOLPERT, J. R. KENEVAN
School of Computer Science and Telecommunications, Roosevelt University, Chicago, IL
{jkenevan,awolpert}@roosevelt.edu

Abstract: In this paper we design a model of a complex process known as the WEB information retrieval. An open queuing network represents the Internet. The TCP channel delivery and multi-facility requirements to serve an information retrieval request are incorporated into the model. We further use this model to determine analytical solutions for mean values of sojourn times needed to respond to an information retrieval request.

Keywords: Modeling and Simulation of the Internet, Communication Networks, Client-Server Systems, and Queuing Networks

1. INTRODUCTION

The main purpose of this paper is to design realistic model of a complex process known as WEB information retrieval. This model must allow evaluating the mean time that it takes the Internet to serve a request. In this paper it is assumed that all requests are directed to one server. Due to limited processing capabilities, the server can run a predefined maximum number of threads (processes) simultaneously. Besides the limitation on the total number of threads is also limited in the number of threads of each service application that can run but on the server simultaneously. Each WEB request requires to run simultaneously a predefined number of threads of different service applications (i.e., WEB database server, http server, mail server, address translation server, etc.) Therefore, if a request arrives that requires more threads of certain type that are currently available then the service is denied and the request rejected. Consequently we consider the system to be a multi-facility propagation network with blocking service.

Here we assume that all requests from the same source require the same facilities. Apparently, a reasonable model of the service process should incorporate the model of underlying TCP sessions with a subsequent evaluation of a TCP sojourn time and a given probability of connection establishment. The system under consideration in this paper is given on Fig. 1.1, i.e., a number of sources $s_1,...,s_n$ generate requests for an access to a Threaded Server TS. Each source $s_i$ generates requests according to a Poisson process with the rate $\lambda_i$. Each generated request must travel through the network cloud to reach a Server TS where it will be served. A request may be split into a number of different Transport Protocol Data Units (TPDU’s) that could travel separately over different routes. The number of routes available for a source $s_i$ depends only on a source. TPDU’s are recombined into a request before delivering it to the TS server. Therefore, recombined TPDU’s that form the request must be delivered in the order they are transmitted. Apparently each route has its own propagation time that is assumed to be exponentially distributed with the mean service time of $k$th route being $T_k=1/\mu_k$. A TPDU is directed into a $k$th route with a probability $p_k$. In this paper we assume that every source request is delivered to the destination with a probability $p_d$ that is the same for all sources and routes. Maximum of number of request retransmissions $N_{max}$ is allowed. After maximum number of retries is reached TCP connection is dropped.

The server TS is a multi-facility server that can run simultaneously $c_1$ threads of application $A_1$, $c_2$ threads of application $A_2$, ..., $c_L$ threads of application $A_L$. Denote a thread capacity vector of the server TS by $c=(c_1, c_2, ..., c_L)$. Each request from a source $s_i$ requires certain number of threads of applications from the set $\{1,...,L\}$. This demand can be represented by a vector $d_i=(d_{i1},...,d_{iL})$ where $d_{ij}$ is the number of servers of type $j$ (threads of application $A_j$) required to serve the request from the source $s_i$. Obviously $D=[d_i]_{1\leq i\leq n}$ is a demand matrix of the mix.
When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]

When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]

When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]

When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]

When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]

When a request from the source \( s_i \) finally arrives at the multi-facility server \( TS \) it generates \( d_1, \ldots, d_k \) to servers of types \( A_1, \ldots, A_k \) (threads of corresponding applications) if the corresponding number of servers is available. If the required number of servers (threads) is not available for at least one the 

\[ p \]

directed into a route with a probability \( p \). The service discipline for each queue is FCFS. A single server (route) \( R_i \) has an exponential service time distribution with the mean \( 1/\mu_i \). Upon the arrival from route \( R_i \), a call from the source \( s_i \) joins the buffer \( RB_i \) to await for a service completion of all calls that entered the system before it. With a probability \( p_d \) a call is delivered to the plain

\[ \text{Fig. 2.1} \]
source they are re-sequenced, instantaneously recombined and delivered to a source.

3. Mean reaction time: retrieved information delivery.

Since thinning of a Poisson process by a Bernoulli process results in a set of independent Poisson processes (see [1]), the set of queues between planes A and B Fig 2.1 behave as a set of independent M/M/1 systems. Therefore, if we denote set of queues between planes A and B fig 2.1 behave as a set of independent Poisson processes (see [1]), the since thinning of a Poisson process by a Bernoulli process results in a set of independent Poisson processes (see [1]), the

\[ E[t_s] = \frac{1}{sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right)} \]

\[ sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right) \]

\[ sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right) \]

Consider now the multi-unit multi-facility server TS as a separate entity. Let us describe the state of the TS by a vector \( \mathbf{n} = (n_1, \ldots, n_l) \) where a \( n_i \) is a number of busy servers of type \( i \) (i.e., the number of active threads of service application \( i \)). Then the state space of the system is defined by \( \text{State}(\mathbf{c}) = \{ \mathbf{n} | \mathbf{n} \leq \mathbf{c} \} \). Here the \( \mathbf{0} \) denote all 0's vector. For such a system under the following assumptions:

- The service time for each type of server is exponentially distributed with a mean service time \( t_s \).
- The request inter-arrival time is distributed according to a general distribution with the mean time between arrivals of calls of the type \( r \) being \( T^{\text{arr}}_r \), the equilibrium probability distribution is given by a well-known product form (see, e.g., [4]):

\[ \pi(n) = \frac{1}{n!} \Pi_{i=1}^{n_i} \left( \frac{\rho_i}{n_i!} \right) \]

\[ \sum_{n \in \text{State}(\mathbf{c})} \Pi_{i=1}^{\mathbf{L}_i} \left( \frac{\rho_i}{n_i!} \right) \]

\[ \sum_{n \in \text{State}(\mathbf{c})} \Pi_{i=1}^{\mathbf{L}_i} \left( \frac{\rho_i}{n_i!} \right) \]

At this point TPDU is either delivered or dropped. If a delivery probability for request is the same and is equal \( p_d \) then with a probability \( 1-p_d \) retry is needed.

The probability of exactly \( K \) retries for delivery of the request is

\[ P(K) = (1-p_d)^K \cdot p_d \]

Therefore the mean number of retries is

\[ E[N_{\text{rt}}] = \Sigma_{i=1}^{n_{\text{max}}} \pi(i) = \Sigma_{i=1}^{n_{\text{max}}} (1-p_d)^i \cdot p_d \]

We assume that the travel of rejection notification takes one call and travels through the cloud the same time. Therefore the mean time to reach plain C Fig 2.1 is computed as

\[ E[t_d] = E[N_{\text{rt}}] \times 2 E[T_s] = \]

\[ \sum_{i=1}^{n_{\text{max}}} (1-p_d)^i \cdot p_d \times 2 \left( \frac{1}{\mu_i - p_i \cdot \lambda_i} \right) \]

Finally, when a request is recombined from its calls at the plain C it's mean delivery time to the server TS becomes

\[ E[t_s] = \frac{1}{sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right)} \]

\[ sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right) \]

\[ sp \left( \sum_{i=1}^{N_{\text{max}}} kE[t_d] \right) \]
Therefore, formula 3.9. Apply.

As in circuit switched systems the rejected request can be repeated with persistence probability \( P_j(j) \) or it can be dropped with a probability \( 1 - P_j(j) \). Let \( P_j(j) = 1 - P_0(j) \) be the probability that a request was after all delivered. Obviously the probability that the request was served in exactly \( m \) tries is

\[
P_m(j) = (P_0(j)P_j(j))^{m-1}(1 - P_0(j))
\]

Therefore the mean number of attempts due to the fact that server is busy is computed as

\[
EN_{RTL}(j) = \sum_{i=1}^{\infty} (P_0(j)P_j(j))^{i-1}(1 - P_0(j)) =
\]

\[
= (1 - P_0(j)) \sum_{i=0}^{\infty} (i - 1) P_0(j)P_j(j)^i + \sum_{i=0}^{\infty} \{ P_0(j)P_j(j)^i \} =
\]

\[
= (1 - P_0(j)) \sum_{i=0}^{\infty} i P_0(j)P_j(j)^i + \sum_{i=0}^{\infty} \{ P_0(j)P_j(j)^i \} =
\]

\[
= \frac{1 - P_0(j)}{(1 - P_0(j)P_j(j))}
\]

Therefore, the mean time \( T_j^{\text{comp}} \) between the request of type \( j \) arrival in the system and completion of service of this request in the server TS can be computed as follows:

\[
T_j^{\text{comp}} = E(N_{RT}(j))E(t_j) + \max_{j \in \{1d_u \neq 0\}} \frac{1}{\mu_j} =
\]

\[
= \frac{1 - P_0(j)}{(1 - P_0(j)P_j(j))} E(t_j) + \max_{j \in \{1d_u \neq 0\}} \frac{1}{\mu_j} \tag{3.12}
\]

Here \( \max_{j \in \{1d_u \neq 0\}} [1/\mu_j] \) is a maximal mean service time of all services that are required \( \text{i.e., have nonzero } d_j \).

At this point the retrieved information is traveling back to a source along one of the same routes it traveled to the TS server. The only difference is that the inter-arrival time between responses to server \( s_j \) requests is no longer exponentially distributed. The sojourn time distribution problem for a system of \( n \) parallel GI/M/1 queues with resequencing buffer was solved in [6] where it was shown that for such system has mean time computed as

\[
E(T^j) = \sum_{i=1}^{n} \frac{p_j}{2 - \rho_j(1 - \sigma_j \mu_j^2)}
\]

Therefore, after recombination the mean delivery time from Ts to a source \( s_i \) is

\[
E(t_i^{\text{rec}}) = \frac{1}{asp_i \sum_{j=a}^{asp_i} j E(T^j)}
\]

\[
= \frac{1}{asp_i \sum_{j=a}^{asp_i} j \sum_{i=1}^{n} p_j \frac{2 - \rho_j(1 - 4\mu_j^4)}{2(\mu_j - p_j \lambda_j)}}
\]

\[
\tag{3.15}
\]

To complete our investigation we need to take into account both (3.12) and (3.15) to receive mean sojourn time of a request from the source \( s_j \) in the system

\[
E(T_s^{\text{system}}) = \left( \frac{1 - P_0(j)}{(1 - P_0(j)P_j(j))} \right) \times
\]

\[
\left( \frac{1}{asp_i \sum_{j=a}^{asp_i} j \sum_{i=1}^{n} p_j \frac{2 - \rho_j(1 - 4\mu_j^4)}{2(\mu_j - p_j \lambda_j)}} \right) +
\]

\[
\max_{j \in \{1d_u \neq 0\}} \frac{1}{\mu_j} + \sum_{i=1}^{n} p_j \frac{2 - \rho_j(1 - 4\mu_j^4)}{2(\mu_j - p_j \lambda_j)}
\]

\[
\tag{3.16}
\]
4. Conclusions

As stated in the introduction we in this paper we obtained mean sojourn times for an Internet information retrieval. The importance of this result comes from the fact that to the best of our knowledge this is the first attempt to model the system in its entirety. However, in conclusion we'd like to mention a number simplifying assumptions that are fundamental for this model but do not always hold in a system under consideration:

- All distributions that are model parameters are exponential which is not always or even predominantly the case in reality;
- A source is expected to generate more requests before receiving a retrieved information. The latter assumption allows us to use open queuing network as a system model as opposed to more realistic model of closed queuing network;
- The assumption of the negative confirmation TPDU travel time from a re-sequencing buffer to a source being the same as a original call travel time is a simplification that only hold if a server re-sequencing policy is FCFS which is usually not the case.

Therefore, further work is needed to remove the aforementioned limitations.

References