Analyzing Response Time of Batch Signing*

Turgay Korkmaz†
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, USA
korkmaz@cs.utsa.edu

Suleyman Tek
Department of Mathematics
University of the Incarnate Word
San Antonio, TX, USA
tek@uiwtx.edu

Abstract

Digital signatures are mainly used to make the receiver believe that a document is actually sent by the claimed sender. However, since generating digital signatures requires intensive computations, researchers proposed batch signing systems to sign multiple documents at once while having almost the same cost of signing one document. In this paper, we analyze how the batch formation strategies and batch sizes impact the response time. Using simulations, we verify our analytical results obtained under the assumption of non-bursty arrivals. We also consider bursty arrivals in our simulations. In general, we observe that using appropriate batch sizes and strategies minimizes the response time in all cases. The improvements are specifically significant when the arrival rate is bursty and dynamic bach sizes are used.

Keywords: Digital signatures; Batch signing; Performance modeling

1 Introduction

In electronic commerce and many other distributed applications, it is necessary to verify that a received message or document is actually sent by a claimed sender. This can be done by using *digital signatures* [11, 6, 12]. For a given document D_i, the sender first computes a hash value H_i and then cryptographically signs this hash to generate a digital signature $DS(H_i)$. The sender then attaches this digital signature to the document and send it. The receiver similarly computes H_i and then checks the digital signature if the receiver has the right key. In general, computing a hash (e.g, SHA1 or MD5) is a computationally cheap task [3]. However, computing a digital signature for each document is a computationally intensive task due to complex mathematical operations in digital signature schemes. So, as illustrated in Figure 1, if a dedicated server is in the charge of generating a digital signature for each document individually, then there will be significant computational overhead. This will result in extra delays and inefficient use of resources, particularly when the document arrival rate increases.

To cope with the intensive computations, researchers proposed batch signing systems to sign multiple documents at once while having almost the same cost of signing one document [10, 5, 13]. Specifically, a simple batch signing server first collects m requests/documents. As before, the server computes a hash H_i for each document and concatenate these hashes $H = H_1|H_2|...|H_m$. Finally, the server uses the concatenated hash to compute a digital signature $DS(H)$. Figure 2 illustrates a server that performs simple batch signing when $m = 3$. Clearly, computing one digital signature instead of computing m digital signatures will significantly reduce the computational load on the server. However, to let the receiver verify the digital signature, we need to also attach the concatenated hash to each document, causing extra bandwidth overhead in the order of $m*$sizeof(hash).

Journal of Internet Services and Information Security, volume: 1, number: 1, pp. 70-85

*An abridged version of this paper was presented in IEEE ICCCN 2009 Workshop on Security, Privacy and Trust of Computer and Cyber-Physical Networks (SecureCPN). This research is supported by DoD Infrastructure Support Program for HBCU/MI, Grant: 54477-CI-ISP (UNCLASSIFIED).

†Corresponding author. Phone: +1-210-458-7346, fax: +1-210-458-4437.

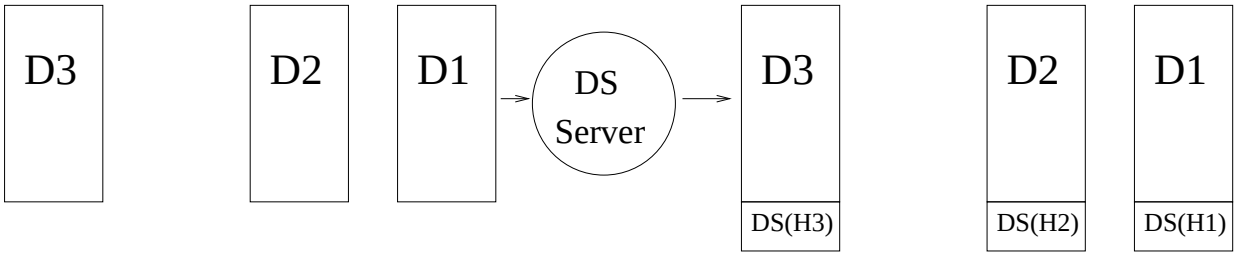

Figure 1: Digital signing of each document independently.

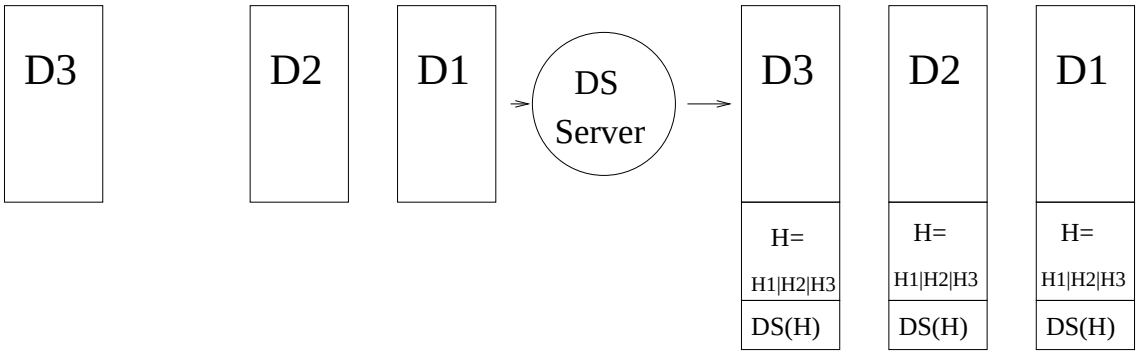

Figure 2: Simple batch signing when $m = 3$.

To minimize the bandwidth overhead, researchers have proposed tree-based batch signing [10]. The basic idea is to first compute hashes for each document as above and then create a binary tree of hashes as illustrated in Figure 3. This means that the server needs to compute total $2m$ hashes. Due to efficiency

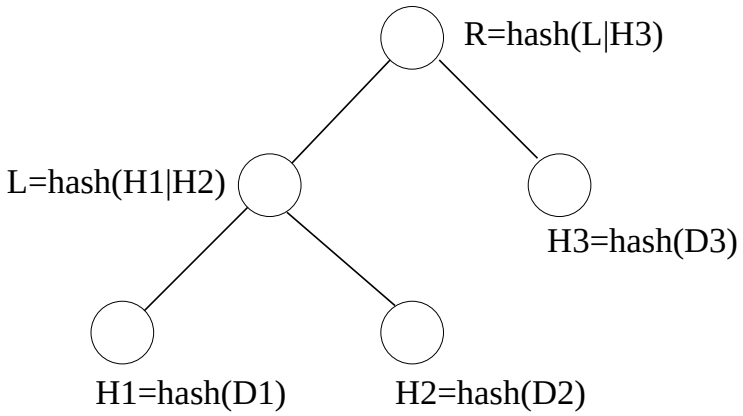

Figure 3: Formation of batch tree for $m = 3$.

of hash computations, this would not create a significant burden on the server; thus, it can be traded to improve bandwidth efficiency. After batch signing tree is constructed, the server only signs the root node and attaches it to each document along with the intermediate hashes that are the sibling nodes on the unique path from the node representing the document to the root. For example, we need to add

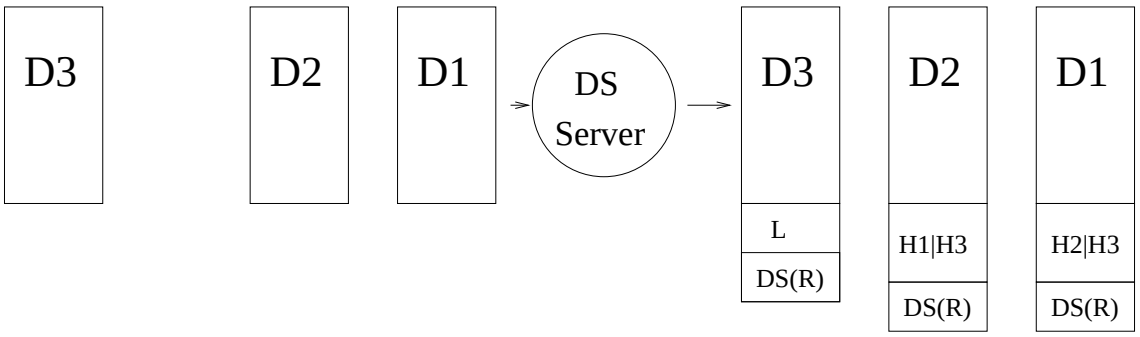

Figure 4: Tree-based batch signing when $m = 3$.

H_2 and H_3 for D_1 while adding only L for D_3. Figure 4 illustrates a server that performs tree-based batch signing when $m = 3$. As proven in [10], the number of hashes added to the document increases as $O(\log_2 m)$. So, the bandwidth overhead of tree-based signing is in the order of $\log_2 m$*sizeof(hash). Table 1 summarizes the computation and bandwidth overhead costs of generating and verifying digital signatures for m documents under different signing schemes.

	No Batch Signing	Simple Batch Signing	Tree-based Batch Signing
Number of hashes computed	m	m	$2m$
Number of digital signatures computed	m	1	1
Number of hashes checked	1	m	$\log_2 m$
Number of digital signatures verified	1	1	1
Bandwidth overhead per message	sizeof(hash)	m*sizeof(hash)	$\log_2 m$*sizeof(hash)

Table 1: Computation cost and bandwidth overhead for generating and verifying digital signatures for m documents under different batch signing schemes.

In this paper our goal is to analyze the response time of batch signing, particularly we consider the cost functions of tree-based batch signing. Since processing a batch of requests takes significantly less amount of time than processing requests one at a time, it would be better to use *larger* size batches. On the other hand, since a request has to wait until a batch is formed, it would be better to use *smaller* size batches to minimize response time. Given this trade-off, it is necessary to carefully analyze the batch formation strategies and determine the batch size so that the average response time for a signing request is minimized. Accordingly, we model batch signing as a queueing system and analyze its response time behavior based on the system parameters and document arrival rate. In any queuing system, the service rate (μ) must be greater than the arrival rate (λ); otherwise, the system will be in an instable condition (i.e., queue will be built up and the delay will go to infinity) [1]. If $\lambda < \mu$, then requests can be signed one at a time to minimize the response time. However, when $\lambda > \mu$, then we need to increase the batch size and re-consider the underlying queuing model and other components of the response time, as done in this paper.

In the rest of this paper, we first study the problem in case of fixed-size batches and try to analytically determine the optimal batch size to minimize response time under the assumption that document arrival rate is Poisson (non-bursty). Using simulations, we verify our analytical results under non-bursty arrivals. We also consider bursty arrivals in our simulations. In addition to fixed size batch sizing, we investigate dynamic batch sizing strategies and evaluate them using simulations under both non-bursty and bursty

arrivals. In general, we observed similar trends regarding how batch sizing improves the response time. The improvements are particularly significant when the arrival rate is bursty and dynamic batch signing is used. We also study how much increase in document arrival rate and/or degradation in service rate can be handled by batch signing. This analysis might be useful during the provisioning of the underlying system. Finally, we conclude this paper.

2 Fixed-size Batching (FSB)

For a given batch size $m \geq 1$, we can define the average response time for a request as follow:

$$RT(m) = f(m) + w_q(m) + s(m) + v(m) \tag{1}$$

where

- $f(m)$ is the cost function denoting the average delay for a request to wait until a m-size batch of signing requests is *formed*,
- $w_q(m)$ is the cost function denoting how much time an already formed m-size batch (and thus each request in that batch) *waits in queue* to be processed,
- $s(m)$ is the cost function denoting how much time the server spends to *sign* an already formed m-size batch,
- $v(m)$ is the cost function denoting how much time a user/client spends to *verify* a signature bundled in an m-size batch.

In general, the computation required for signing and verifying a batch of m requests is mainly the same as that of signing and verifying a single request. However, since we specifically focus on the tree-based batching discussed in previous section, we have take into account some additional costs such as generating $2m$ hashes when signing, and checking $\log_2(m)$ hashes when verifying. Based on the cost functions in Table 1, we model the cost for $s(m)$ and $v(m)$ as the functions of m in the forms of

$$s(m) = t_s + 2ma_s$$

and

$$v(m) = t_v + \log_2(m)a_v,$$

where t_s and t_v are the times required to generate and verify a digital signature for a single request, respectively; and a_s and a_v are the times for computing and checking a hash value during signing and verification, respectively. In general, $t_s > t_v >> a_s > a_v$. Given t_s, t_v, a_s, and a_v, we can easily compute $s(m)$ and $v(m)$.

Therefore, we next focus on $f(m)$ and $w_q(m)$ in a batching system. To analytically compute these components of response time, we first assume that incoming requests follow the Poisson process with mean arrival rate λ (i.e., inter-arrival time is exponentially distributed with mean $1/\lambda$) while the processing time is deterministically given with the service rate μ (when processed one at a time). Accordingly, we obtain an approximate closed form expression for $RT(m)$ and check its validity using simulations under Poisson (i.e., non-bursty) arrivals. We also evaluate the performance of batch signing using simulations under busty arrivals generated by a two-state Markov Modulated Poisson Process (MMPP).

2.1 Stability condition

For a stable batch signing system, $\lambda_{batch} = \frac{\lambda}{m}$ must be smaller than $\mu_{batch} = \frac{1}{s(m)} = \frac{1}{t_s + 2ma_s}$. Accordingly, we can determine that

$$m > \frac{\lambda t_s}{1 - 2\lambda a_s}$$

for a stable system. In addition, denumerator $1 - 2\lambda a_s$ must be greater than zero so that batch size will not reach to infinity. To satisfy this,

$$a_s < \frac{1}{2\lambda} \text{ or } \lambda < \frac{1}{2a_s}.$$

2.2 Computing $f(m)$ under Non-Bursty Arrivals

Suppose the first request of a batch is arrived at time t and $t = 0$ without loss of generality. As illustrated in Figure 5, the second request arrives with inter-arrival time T_1, the third request arrives with inter-arrival time T_2, and so on until the m^{th} request arrives to complete the formation of a batch. Note that

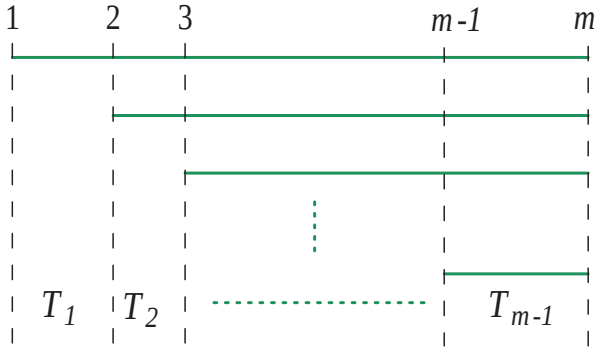

Figure 5: Formation of a fixed-size batch.

$T_1, T_2, \cdots, T_{m-1}$ are independent and identically distributed (iid) random variables corresponding to inter-arrival times in Poisson process. As a result, we can compute the average delay for a request during the formation of a batch as follows:

$$f(m) = \frac{E[T_1] + 2E[T_2] + \cdots + (m-1)E[T_{m-1}]}{m}.$$

Since inter-arrival times are iid and exponentially distributed random variables with mean $1/\lambda$, we have

$$f(m) = \frac{1/\lambda \sum_{i=1}^{m-1} i}{m} = \frac{m-1}{2\lambda}.$$

2.3 Computing $w_q(m)$ under Non-Bursty Arrivals

First we need to determine the inter-arrival times between batches and the distribution of these inter-arrival times. Suppose a batch has been formed at time t and $t = 0$ without loss of generality. The next batch will be formed when m requests arrive. In other words, the inter-arrival time between batches is the sum of m inter-arrival times between requests, as illustrated in Figure 6. Since inter-arrival times are independent exponential random variables with mean $1/\lambda$ and the sum of m exponential random variables has gamma distribution (more specifically m-Erlang distribution) with parameters of $\alpha = m$

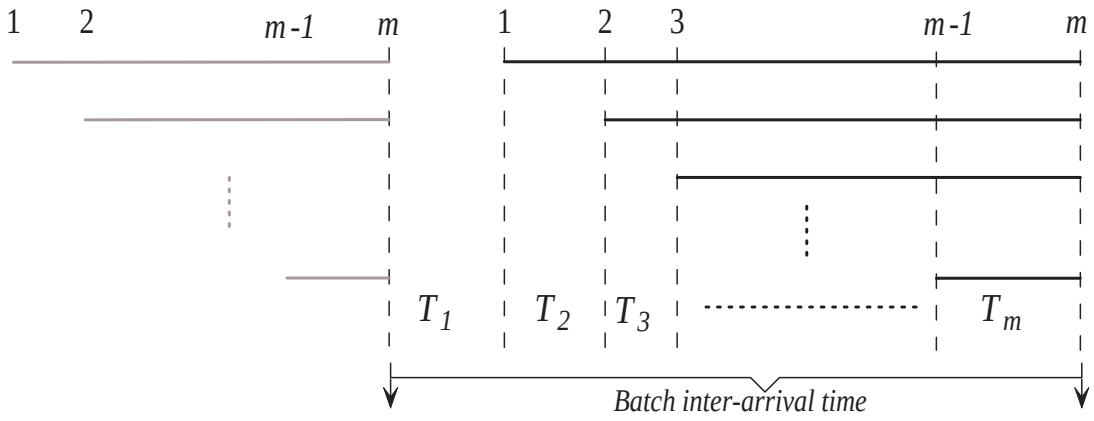

Figure 6: Inter-arrival time of fixed-size batches.

and $\beta = \lambda$ [1, 9], we conclude that the batch inter-arrival time is an m-Erlang distributed random variable with mean of $\alpha/\beta = m/\lambda$ and variance of $\alpha/\beta^2 = m/\lambda^2$.

In summary, the arrival of batches has a general distribution with the rate of $\lambda_{batch} = \lambda/m$ and the deterministic batch processing with the rate of $\mu_{batch} = 1/s(m)$. To determine the average delay ($w_q(m)$) that a batch encounters in the queue, we consider the G/G/1 queuing system. In the literature (e.g., [7]), the researchers have provided good approximations for the average queuing delay of G/G/1 systems. With our notation, the average queuing delay can be computed as follows:

$$w_q(m) \simeq \frac{\rho}{1-\rho}\frac{C_a^2 + C_s^2}{2}\frac{1}{\mu_{batch}},$$

where $\rho = \lambda_{batch}/\mu_{batch} = \frac{\lambda s(m)}{m}$, C_a^2 and C_s^2 are the coefficients of variations of batch inter-arrival and service times, respectively. Since the inter-arrival time has m-Erlang distribution, $C_a^2 = 1/m$. Since the service time is deterministic, $C_s^2 = 0$. As a result, the average queuing delay can be approximately computed by:

$$w_q(m) \simeq \frac{\lambda s(m)}{m - \lambda s(m)}\frac{1}{2m}s(m).$$

By substituting $f(m)$, and $w_q(m)$ into (1), we obtain

$$RT(m) \simeq \frac{m-1}{2\lambda} + \frac{\lambda s(m)^2}{2m(m-\lambda s(m))} + s(m) + v(m) \tag{2}$$

Using this approximate formula of $RT(m)$, we can also analytically determine the optimal batch size (say m^*) by taking the derivative of $RT(m)$ with respect to m and then solving $RT'(m) = 0$. For this, first let

$$u = \lambda s(m)^2 = (4\lambda a_s^2)m^2 + (4\lambda a_s t_s)m + (\lambda t_s^2)$$

and

$$v = 2m(m - \lambda s(m)) = (2(1 - 2\lambda a_s))m^2 - (2\lambda t_s)m.$$

Then

$$RT'(m) = \frac{1}{2\lambda} + \frac{vu' - uv'}{v^2} + 2a_s + a_v\frac{1}{m\ln(2)},$$

where

$$u' = (8\lambda a_s^2)m + (4\lambda a_s t_s)$$

and

$$v' = (4(1-2\lambda a_s))m - (2\lambda t_s).$$

After substituting u, u' v, and v' and doing some algebraic operations, it is easy to see that we will obtain a fourth degree polynomial for $RT'(m) = 0$. Unfortunately, finding roots of such a polynomial will not be an easy task. Instead, we can use Newton's method [4]. To solve $RT'(m) = 0$, Newton's method has the following formula

$$m_{i+1} = m_i - \frac{RT'(m_i)}{RT''(m_i)} \quad i = 0, 1, 2, \ldots,$$

where m_0 is given as an initial guess. As seen in the above formula, Newton's method requires us to also take the second derivative of $RT(m)$. This can easily be done as follows.

$$RT''(m) = \frac{v^2(vu' - uv')' - (vu' - uv')(v^2)'}{v^4} - \frac{a_v}{m^2 \ln(2)}$$

$$= \frac{v^2(v'u' + vu'' - u'v' - uv'') - (vu' - uv')2vv'}{v^4} - \frac{a_v}{m^2 \ln(2)}$$

where $u'' = 8\lambda a_s^2$ and $v'' = 4(1 - \lambda a_s)$.

In our case, Newton's method is easier than solving a forth degree polynomial and it converges quickly since we are able to make a good guess for m_0 as follows. As discussed above, for a stable batch signing system, $m > \frac{\lambda t_s}{1-2\lambda a_s}$. Accordingly, we set $m_0 = \frac{\lambda t_s}{1-2\lambda a_s} + 1$ and in a few iterations, we determine m_n as the root of $RT'(m) = 0$. Note that m_n might be a real number. In this case, we check if $RT(\lceil m_n \rceil) < RT(\lfloor m_n \rfloor)$, then set $m^* = \lceil m_n \rceil$; otherwise, $m^* = \lfloor m_n \rfloor$. For example, we obtained the optimal batch sizes of $m^* = 7$ within 6 iterations, and $m^* = 18$ within 7 iterations for the two cases that we consider during our simulations in the next subsection (see Figure 7).

2.4 Simulation Results under Non-Bursty Arrivals

To check the validity of the analytical approximation in Equation (2), we conduct several simulations. We implemented our simulator in C language using CSIM [8] library. It simply generates signing requests according to Poisson process with the given parameters, forms batches based on the given strategy, and then process batches using a queuing system. We run our simulations for 100,000 requests and take averages of their response times.

Since we observed the similar trends under many runs with different parameters, we just report two cases as shown in Figure 7. Clearly, the figure shows that the analytical formula in Equation (2) captures the response time behavior of a batch signing system with a slight overestimation of RT. Both simulation and analytical results give the same optimal batch size. Using the Newton's method described in previous subsection, we determined that the optimal batch size for Case I is $m^* = 7$ within 6 iterations, and that for Case II is $m^* = 18$ within 7 iterations.

2.5 Simulation Results under Bursty Arrivals

We extended our simulator to also generate bursty arrivals. For this we used a two-state Markov Modulated Poisson Process (MMPP) with parameters λ_1, λ_1, r_1, r_2 [2]. As a doubly stochastic Poisson process, a two-state MMPP controls the arrival rate by a continuous-time Markov chain that has two states, namely S_1 and S_2. Transition intensities from S_1 to S_2 and from S_2 to S_1 are r_1 and r_2, respectively. In state S_1,

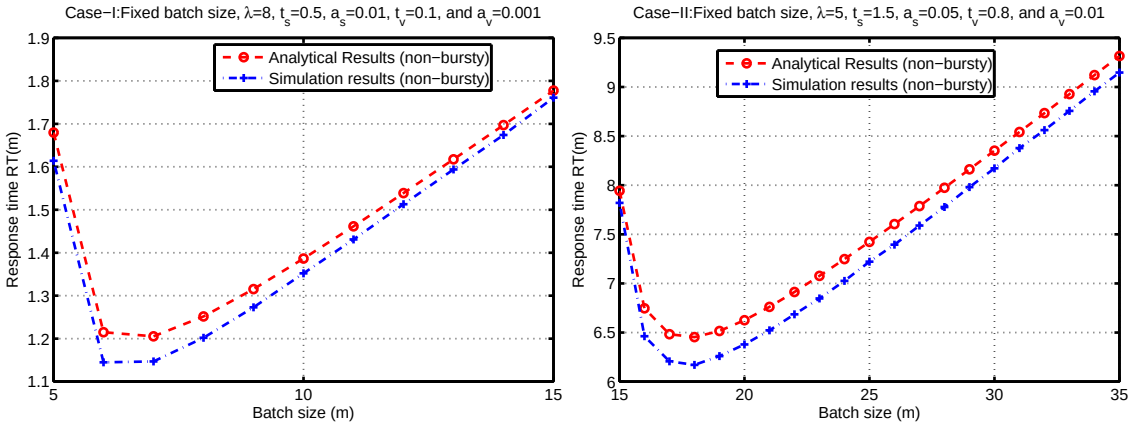

Figure 7: Comparison of analytical and simulation results in case of fixed-size batches under non-bursty (Poisson) arrivals.

the arrival process follows a Poisson process with rate λ_1. In state S_2, the arrival process again follows a Poisson process but with rate λ_2. The mean arrival rate λ is given as

$$\lambda = \frac{\lambda_1 r_2 + \lambda_2 r_1}{r_1 + r_2} \tag{3}$$

In our simulations, we use (3) to determine the parameters of MMPP such that the same mean arrival rate λ will be used for both non-bursty and bursty arrivals. Specifically, we use the same parameters and constants that were used to generate bursty web traffic in [2]. Accordingly, we set r_1 and r_2 to 0.05 and 0.95, respectively, while setting λ_1 to $\lambda_1 = 0.75\lambda$. Then from (3), we determine λ_2 as

$$\lambda_2 = \frac{(r_1 + r2)\lambda - \lambda_1 r_2}{r_1}$$

In summary, λ_1 is the low arrival rate and will be used 95% of the time while λ_2 is the high arrival rate and will be used 5% of the time to generate sudden bursty arrivals.

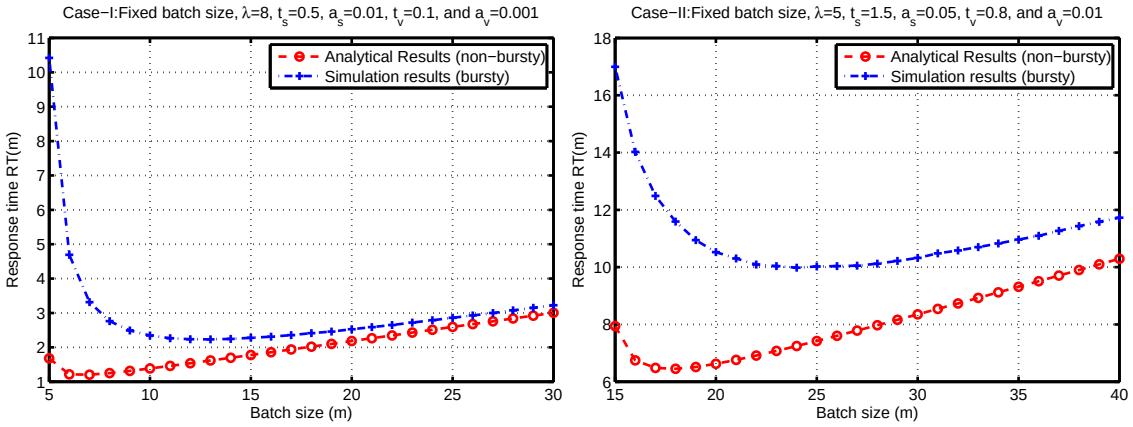

Figure 8: Comparison of analytical and simulation results in case of fixed-size batches under bursty (MMPP) arrivals.

Under the same two cases we considered before, we just changed the arrival process and obtained the results shown in Figure 8. Clearly, the analytical formula derived based on Poisson process cannot capture the response time behavior of a batch signing system when arrivals are bursty. However, the general trend shows that when an appropriate batch size is selected the response time will be reduced significantly. Also we observed that since the very large batch sizes smooths the burstiness, the analytical formula derived based on Poisson process may closely approximate the performance behavior under bursty arrivals.

3 Varying-size Batching (VSB)

Instead of fixed-size batching (FSB) described above, we may also use other strategies that use batches with different sizes. We now consider what would be the best strategy in case of varying-size batches. We mainly consider the following three varying-size batching strategies:

- **Lower-bounded batching (LBB)**: This strategy requires a minimum batch size (e.g., MINB=3). In other words, the server will wait at least 3 requests to come in. It will then process them. While processing these 3 requests, 10 more requests might come in. When the server is done with these 3 requests, it can process all of the 10 requests in the next batch and so on. In general, the batch size will vary in the range [MINB, ∞).
- **Upper-bounded batching (UBB)**: This strategy puts a maximum limit on the batch size (e.g., MAXB=20) and processes any batch that is smaller than MAXB. For example, suppose the server founds 10 requests when it gets idle, then it starts processing these 10. While processing these 10, the server may receive 30 requests. In this case, the server takes the first 20 in a batch and process them while the other 10 is waiting in queue. Thus, the batch size in this case will vary in the range [1,MAXB].
- **Lower and Upper-bounded batching (LUBB)**: This strategy is the combination of the previous two strategies. It simply maintains MINB and MAXB, and process any batch that has the size in the range [MINB, MAXB].

Unfortunately, it is very difficult if not impossible to obtain analytical models in case of varying batch sizes. Therefore, we mainly use simulations for analyzing and comparing these strategies under both non-bursty and bursty arrivals.

3.1 Simulation Results

We extended our simulator to handle varying batch sizes. We again consider the same two cases under the non-bursty and bursty arrivals as described before. Figures 9 and 10 show the simulation results for varying size batches under **non-bursty arrivals** for Case-I and Case-II, respectively. Similarly, Figures 11 and 12 show the simulation results for varying size batches under **bursty arrivals** for Case-I and Case-II, respectively.

In all these figures, part (a) presents the average response times for fixed-size batching (FSB), LBB, and UBB. In all cases, both LBB and UBB give better response time (RT) than FSB as long as the MINB in LBB is less than the optimal batch size found for FSB, and MAXB in UBB is greater than the optimal batch size found for FSB.

Part (b) presents the average batch size for FSB, LBB, and UBB. For FSB, the average batch size naturally increases as the fixed batch size (m) increases. **LBB** uses the batch sizes within the range [MINB= m, ∞). As long as MINB is less than the optimal batch size in FSB, LBB strategy is able to

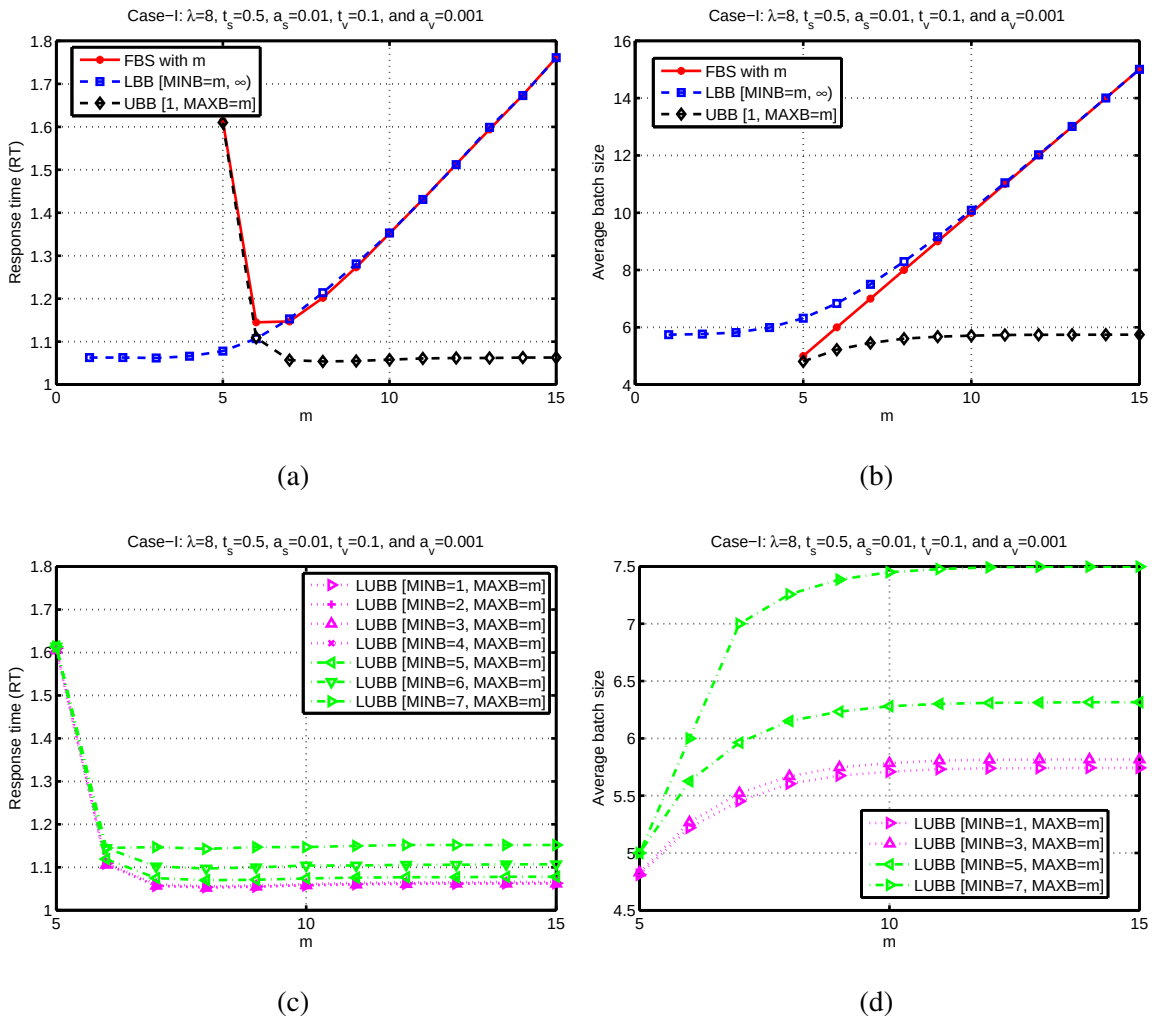

Figure 9: Case-I: Comparison of strategies using varying size batches under non-bursty (Poisson) arrivals.

adjust the batch size and operate around the optimal point in terms of both average batch size as well as the average response time as seen in part (a). When MINB is increased beyond the optimal batch size in FSB, the average batch size as well as the average response time as seen in part (a) keep increasing. **UBB** uses the batch sizes within the range [1, MAXB= m]. So UBB is also able to adjust the batch size and operate around the optimal point as long as the MAXB is greater than the optimal batch size in FSB. If MAXB is less than the optimal batch size in FSB, then the system cannot be stable and the average response time will be significant as seen in part (a).

Part (c) and (d) present the average response time and average batch size for LUBB, which uses the batch sizes within the range [MINB, MAXB] as the combination of both LBB and UBB. So when MINB is 1, LUBB is the same as UBB. Increasing MINB does not cause any significant change in response time. Actually, when we zoom in the figures in part (c), we see that a small value for MINB (e.g., 2 or 3) may give better results. This is due to fact that the batch signing system is now able to better deal with the random fluctuations in the arrival rate of signing requests by jointly minimizing the batch formation delay and queuing delay. However, increasing MINB beyond 3 usually causes slight increase in response time. Moreover, increasing MINB naturally increases the average batch size as seen in part (d). As a

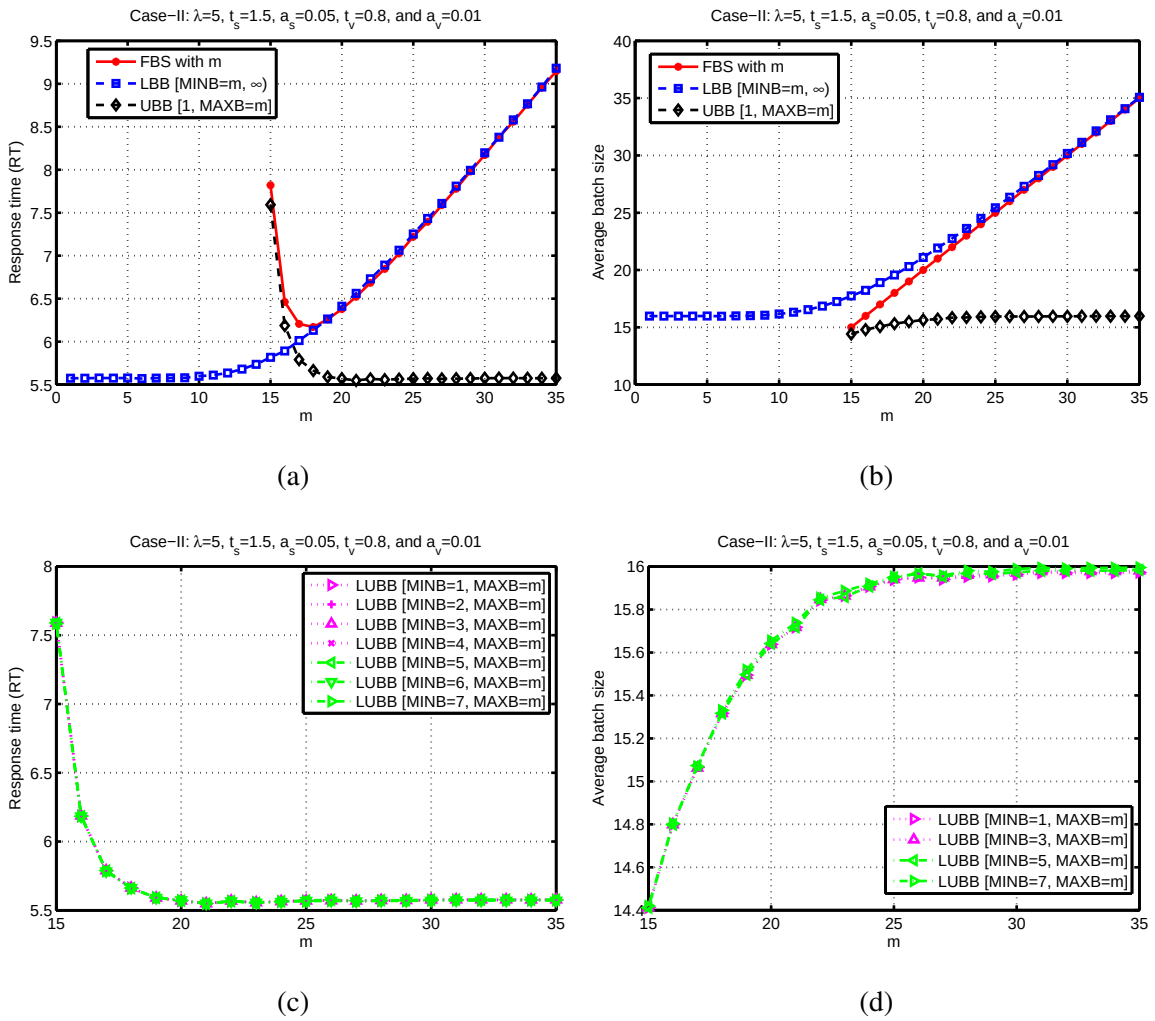

Figure 10: Case-II: Comparison of strategies using varying size batches under non-bursty (Poisson) arrivals.

result, to get the best performance, we need to use LUBB with smaller values of MINB along with an appropriate MAXB that should be greater than the optimal batch size in FSB.

In summary, the varying-size batching techniques allow the server to adjust the batch size and operate around the optimal point in terms of both the average response time and average batch size. For example, we can simply use LBB with a small value of MINB. Even though LBB provides good performance on average, it has no upper limit on the maximum batch size. In some case, it might be necessary to have an upper limit on the maximum batch size (e.g., to limit bandwidth overhead for each document). In that case, we can use LUBB with small MINB and an appropriate MAXB. As the above simulation results show that limiting MAXB will not significantly affect the performance as long as MAXB is greater than the optimal batch size in FSB. In this regard, our analysis in previous section may help to determine appropriate values for MAXB based on the expected load and system parameters. LUBB then adjust the actual batch size on the fly and give the best performance.

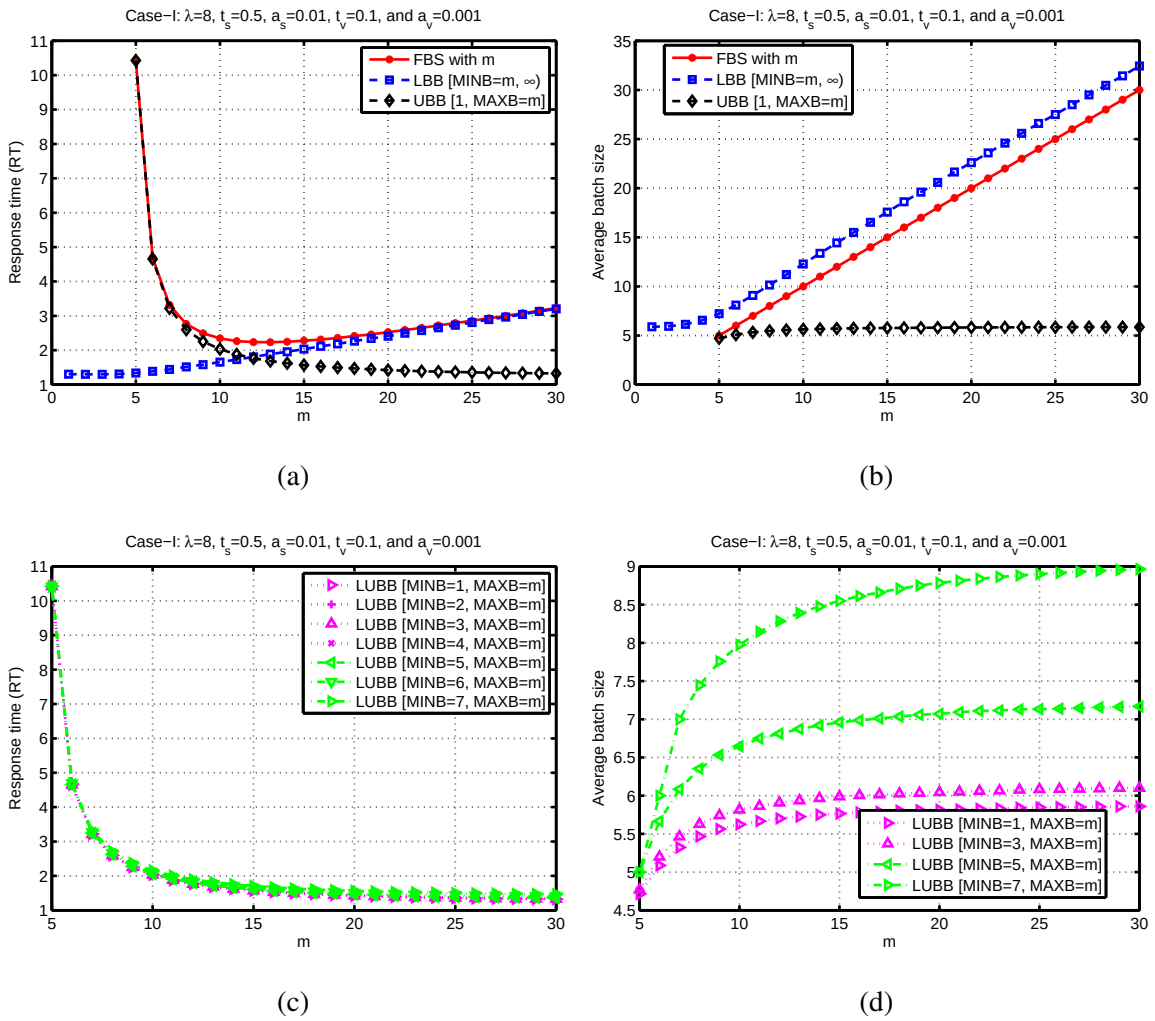

Figure 11: Case-I: Comparison of strategies using varying size batches under bursty (MMPP) arrivals.

4 Impact of Arrival Rate Increase

As discussed and analyzed in previous sections, batch signing is necessary when arrival rate (λ) is greater than the service rate (μ). We are now interested in how much increase in arrival rate can be tolerated by the batch signing.

Suppose the arrival rate increases from λ to $\lambda_{new} = a\lambda$, $a > 1$. Clearly, we can use our previous analysis with λ_{new} and determine the new optimal batch size that minimizes the new average response time (RT), which naturally increases as a increases. Here we are specifically interested in how much increase the signing system can tolerate. The answer depends on wether we want to have (i) a stable system with arbitrary (but finite) response time, or (ii) a stable system with some upper bound on response time. For the first design option, we need to determine the maximum value of a while making sure that the system is still stable (e.g., average response delay might be large but it will be finite).

From the discussions in previous sections, we know that the condition for a stable system is

$$\lambda_{batch} = \frac{a\lambda}{m} \quad < \quad \mu_{batch} = \frac{1}{s(m)}.$$

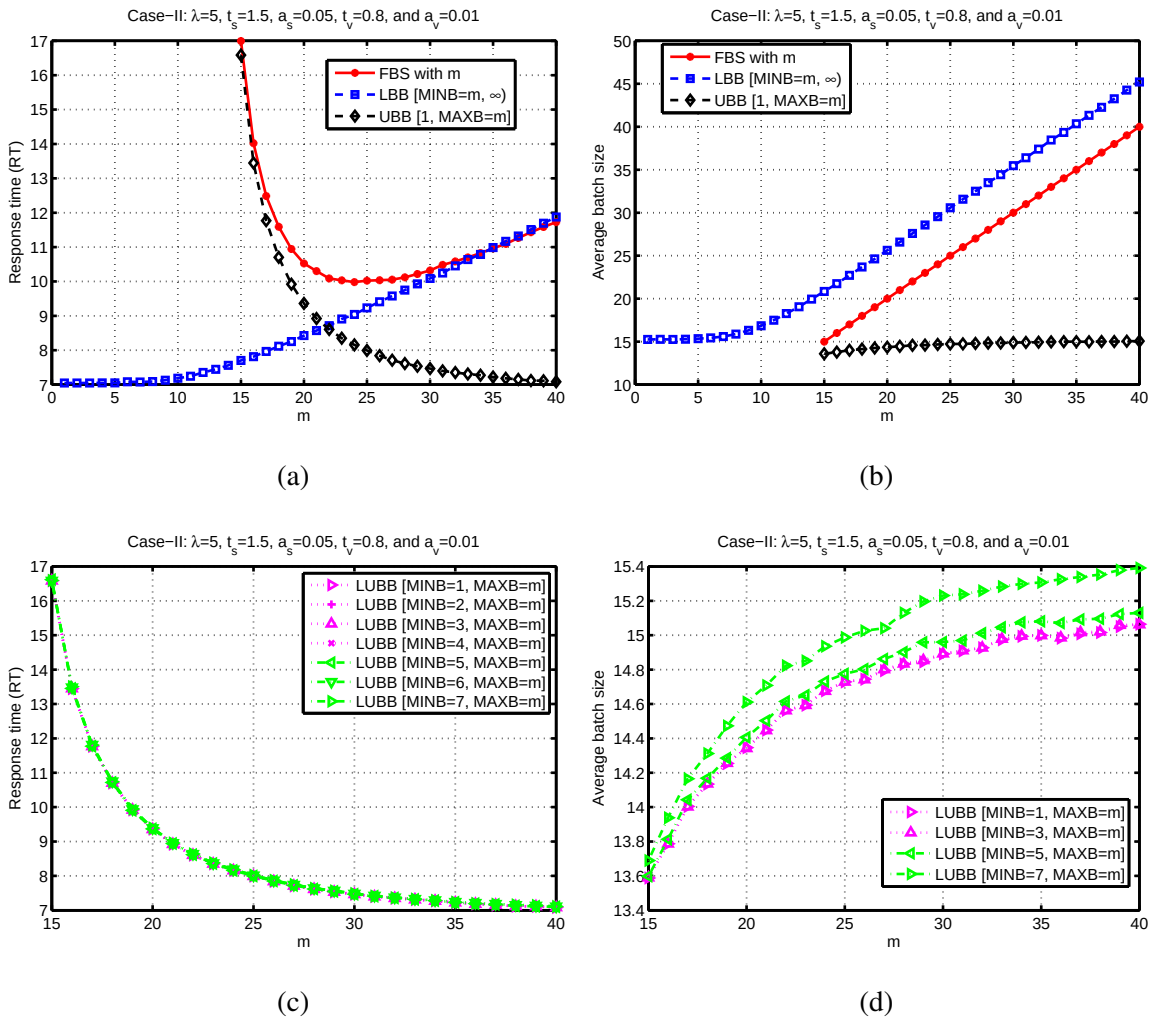

Figure 12: Case-II: Comparison of strategies using varying size batches under bursty (MMPP) arrivals.

From which, we can determine that

$$a < \frac{m}{\lambda t_s + 2\lambda a_s m}.$$

As m goes to ∞, we can determine the upper bound on a by taking the limit as follows:

$$a < \lim_{m\to\infty} \frac{m}{\lambda t_s + 2\lambda a_s m}.$$

Using the L'Hospital rule, we have

$$a < \frac{1}{2\lambda a_s}.$$

For the two cases that we considered in Figure 7, the upper bounds on a will be 6.25 and 2, respectively. As long as the new arrival rate is less than 6.25 times the original arrival rate in Case-I and 2 times in Case-II, then the batch signing system will be stable by selecting an appropriate batch size by using our analysis in previous sections. For example, Figure 13 shows the response time behavior of the batch signing system and the optimal batch size under different values of a for the Case-I and Case-II we considered before.

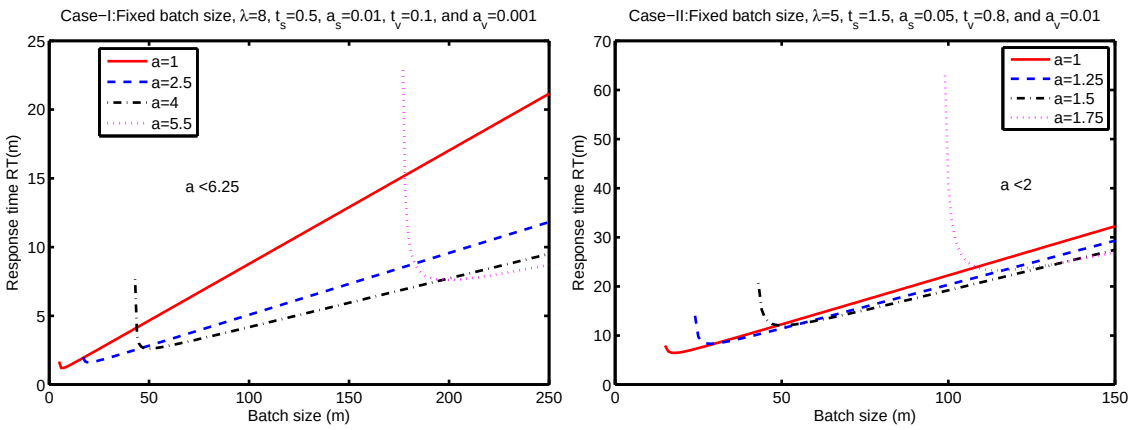

Figure 13: Performance of batch signing when arrival rate increases.

For the second design option (i.e., a stable system with some upper bound on response time), we need to (*i*) replace λ with $\lambda_{new} = a\lambda$ in our previous $RT(m)$ formula and (*ii*) determine the maximum value of a and the value of m^* such that $RT(m^*)$ is less than the given upper bound denoted by $MAXrt$ on the response time. This will require us to first symbolically solve $RT'(m) = 0$ and obtain m^* as a function of a, and then find maximum a that satisfies $RT(m^*) < MAXrt$. As we discussed before, since there is no closed form solution for $RT'(m) = 0$, it will be difficult to compute an analytical upper bound on a for a given $MAXrt$. However, we can numerically determine the upper bound on a by using the binary search in the range $[1, \frac{1}{2\lambda a_s}]$.

5 Impact of Service Rate Degradation

Suppose that the server experiences some performance degradation and thus spends $s_{new} = \gamma s(m)$ unit of time to process signing requests, where $\gamma > 1$. Clearly, we can use our previous analysis with s_{new} instead of $s(m)$ and determine the new optimal batch size that minimizes the average response time (RT), which naturally increases as γ increases.

In Figure 14 we illustrate how much performance degradation at the server can be tolerated by increasing the batch size for the same two cases we studied before. Specifically, we consider 20% and 40% degradation and thus set γ to 1.2 and γ to 1.4, respectively. From Figure 14, it is easy to see that when γ increases, we need to increase the batch size so that the system will be in stable condition. Depending on the value of γ, the optimal batch size can be analytically determined by simply modifying the formulas we provided before.

6 Conclusions

In this paper we analyzed the performance characteristics of batch signing. We first described how to analytically determine the optimal batch size to minimize response time under non-bursty (Poisson) arrivals. We then verified our analytical results via simulations. We also considered bursty arrivals in our simulations. In general, we observed that using very large or small batch sizes increase the average response time and cause instabilities under both non-bursty and bursty arrivals. So we need to carefully select the batch size based on the system parameters and load. However, since the load may not be known in advance, it will be really a challenge to pick an optimal fixed batch size.

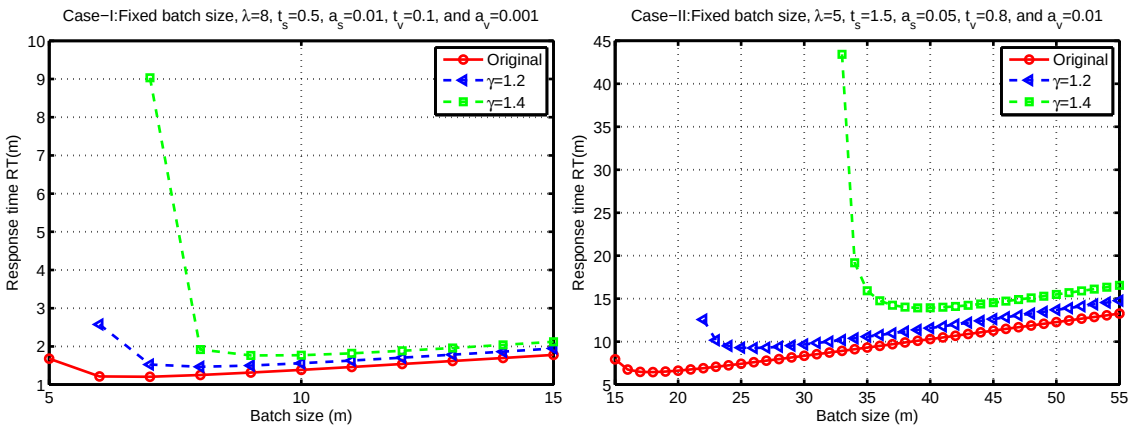

Figure 14: Performance of batch signing under performance degradation at the server.

Accordingly, we considered varying-size batch signing strategies that can dynamically adjust the batch size. Fortunately, our results showed that varying-size batch signing strategies can adjust the batch size on the fly and provide the best performance. Specifically, we recommend to use LBB with a small value of MINB. Even though LBB provides good performance on average, it has no upper limit on the maximum batch size. If needed, we can limit the maximum batch size by using LUBB with a small MINB and an appropriate MAXB. We need to make sure that MAXB for LUBB is set to a value that is at least greater than the optimal fixed batch size found using our analysis. So the analytical results help us to determine certain parameters of varying-size batch signing strategies based on the expected system parameters and load.

Finally, we also analyzed how much increase in arrival rate or degradation in service rate can be tolerated by a batch signing system. This analysis would be useful when provisioning a batch signing systems with the desired level of robustness under various parameters, loads and attacks.

Acknowledgment

I would like to thank Shouhuai Xu for the discussions and comments on the problem formulation.

References

[1] Arnold O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications (2nd Edition)*. Academic Press, 1990.

[2] Mikael Andersson, Jianhua Cao, Maria Kihl, and Christian Nyberg. Performance modeling of an Apache web server with bursty arrival traffic. In *Proc. of the 2003 International Conference on Internet Computing (IC'03), Las Vegas, Nevada, USA*, volume 2, pages 558–514. CSREA Press, June 2003.

[3] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Proc. of the 16th Annual International Cryptology Conference (CRYPTO'96), Santa Barbara, California, USA , LNCS*, volume 1109, pages 1–15. Springer-Verlag, August 1996.

[4] I.N. Bronshtein, K.A. Semendyayev, G. Musiol, and H Muehlig. *Handbook of Mathematics (4th Edition)*. Springer-Verlag, 2004.

[5] William C. Cheng, Cheng fu Chou, and Leana Golubchik. Performance of batch-based digital signatures. In *Proc. of the 10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'02), Fort Worth, Texas, USA*, pages 291–299. IEEE, October 2002.

[6] Charles Kaufman, Radia Perlman, and Michael Speciner. *Network Security, Private Communication in a Public World (2nd Edition)*. Prentice Hall, 2002.

[7] Paul J. Kuehn. Approximate analysis of general queueing networks by decomposition. *IEEE Transactions on Communications*, 27(1):113–126, January 1979.

[8] Inc. Mesquite Software. CSIM 19 Simulation Engine (c version). http://www-users.cselabs.umn.edu/classes/Fall-2009/csci5104/CSIM/UsersGuide-CVersion.pdf, 2004.

[9] Athanasios Papoulis. *Probability, random variables, stochastic processes (3rd Edition)*. McGraw-Hill, 1991.

[10] Christopher J. Pavlovski and Colin Boyd. Efficient batch signature generation using tree structures. In *Proc. of International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99), City University of Hong Kong, Hong Kong*, pages 70–77, July 1999.

[11] William Stallings. *Cryptography and Network Security (3rd Edition)*. Prentice Hall, 2003.

[12] Wikipedia. Digital signature. `http://en.wikipedia.org/wiki/Digital_signature`, 2008.

[13] Taek-Young Youn, Young-Ho Park, Taekyoung Kwon, Soonhak Kwon, and Jongin Lim. Efficient flexible batch signing techniques for imbalanced communication applications. *IEICE transactions on Information and Systems*, E91-D(5):1481–1484, May 2008.

Turgay Korkmaz received the B.Sc. degree with the first ranking from Computer Science and Engineering at Hacettepe University, Ankara, Turkey, in 1994, and two M.Sc. degrees from Computer Engineering at Bilkent University, Ankara, and Computer and Information Science at Syracuse University, Syracuse, NY, in 1996 and 1997, respectively. In Dec 2001, Dr. Korkmaz received his PhD degree from Elec. and Computer Eng. at University of Arizona, under the supervision of Dr. Marwan Krunz. In January 2002, he joined the University of Texas at San Antonio as an Assistant Professor of Computer Science Department. Dr. Korkmaz received his tenure in September 2008, and he is currently an Associate Professor of Computer Science Department. Dr. Korkmaz works in the area of computer networks and network security and Internet related technologies.

Suleyman Tek received the B.Sc. degree with cum laude in Mathematics in 2001 from Department of Mathematics, Dokuz Eylul University, Izmir, Turkey. He received the M.S. and PhD degree in Mathematics under the supervision of Dr. Metin Gurses in 2003 and 2007, respectively, from Department of Mathematics, Bilkent University, Ankara, Turkey. Dr. Tek worked on a project funded by US Department of Defense and University of Arkansas at Little Rock (UALR) as a posdoctoral research Associate between August 2007 and September 2008. He worked as an Adjunct faculty at UALR between September 2008 and August 2009. He joined the University of the Incarnate Word, San Antonio, TX in August 2009 and currently serving as an Assistant Professor of Mathematics. Dr. Tek's research area are differential geometry of integrable partial differential equations, Solution surfaces, geometry of biomembranes, mathematical modeling of biological systems, cryptography.