2013

Modeling an Adaptive System with Complex Queuing Networks and Simulation

Jesse C. Musgrove
West Virginia University

Modeling an Adaptive System with Complex Queuing Networks and Simulation

Jesse C. Musgrove

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science in Computer Science

Lane Department of
Computer Science and Electrical Engineering

Morgantown, West Virginia
2013

Keywords: Adaptive Systems, Queuing Networks, Security, Modeling, Simulation

ABSTRACT

Modeling an Adaptive System with Complex Queuing Networks and Simulation

Jesse Musgrove

An adaptive system differs from a non-adaptive system in that an adaptive system uses a specific process to identify and implement adaptations to system parameters during run time in an effort to increase system performance.

In order to develop an adaptive system one of the most important aspects is the ability to accurately predict and manage system behavior. If an unexpected event has occurred, accurate prediction of system behavior is needed in order to determine whether or not the system is able to continually meet expectations and/or requirements. When considering any possible adaptations to the system, one must be able to accurately predict their consequences as well.

In a feedback loop of an adaptive system performance analysis and prediction performance analysis and prediction may lead to a large number of different states. Therefore, the method of analyzing the system must be fast. For complex systems, however, the most popular method for predicting performance is simulation. Simulations, depending on the size of the system, are known for being slow.

In this thesis we develop a fast method for prediction of performance of a complex system. We use this method to allocate resources to the system initially, and then make decisions for system adaptation during runtime. Finally we test various modifications to the system in order to measure the robustness of our method.

ACKNOWLEDGEMENTS

I would like to thank my mentor, Professor Bojan Cukic, for his guidance while performing my duties at West Virginia University. I would also like to thank him for allowing me to work at my own pace and at my own discretion. If not for his understanding in this matter I would never have been able to complete my thesis.

I would like to thank my committee members, Dr. Donald Adjeroh and Dr. James D. Mooney for taking the time to read my thesis and listen to my defense.

I would like to thank Mayra Sacanamboy whose initial thesis work paved the way for my work. The queuing model studied herein is based entirely off the original queuing network developed by Mayra.

I would like to thank Professor Vittorio Cortellessa for his help while I worked on this project. Whenever I would get stuck or have questions on how to proceed he always gave great advice.

I would like to thank my fellow graduate student Sean Banerjee for his gracious assistance when I had trouble writing this document.

Lastly, I would like to thank my girlfriend, my friends, and my family, who regardless of how stressed out and unpleasant I became have always supported me through graduate school.

Jesse Musgrove
Morgantown, December 2013

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAS	Advanced Analytical Solver
AWT	Average Wait Time
CID	Call-In Delay
CT	Current Time
EL	Event Line
ESF	Extra Server First
EWT	Early Warning Time
FMR	False Match Rate
FNMR	False Non-Match Rate
LQNS	Layered Queuing Network Solver
MAT	Mean Arrival Time
NAS	Naïve Analytical Solver
OAAS	Optimized Advanced Analytical Solver
PO	Primary Officer
POQ	Primary Officer Queue
SAT	Scheduled Arrival Time
SO	Secondary Officer
SOQ	Secondary Officer Queue
TRF	Threshold Reduction First

1 Introduction

In her 2009 MS Thesis, Mayra Sacanamboy developed a queuing network to model border crossing for international flight arrivals [1]. In this queuing network, a biometric algorithm is used to evaluate traveler's identity claims. Passengers offering their true identity, as evaluated by the system, are allowed into the country. Others are placed in a second queue in which they are examined more closely. Most importantly, however, the thesis showed that under certain circumstances the parameters of the biometric algorithm could be adjusted in order to improve the throughput, while minimally affecting the accuracy of identification. It then follows that it may be possible to devise an adaptive system to react to unexpected changes in operating conditions of the system during its run time by making decisions that improve certain aspects of its performance.

This brings up a difficult challenge. The only way to identify intelligent decisions for the system would be to first accurately predict how they would affect performance. Therefore, when given a set of parameters for the system, we must be able to accurately measure whether or not the system performance will meet the requirements. Furthermore, we may need to examine hundreds of possible combinations of decisions during any iteration of the adaptive feedback loop. For that reason our method for performance prediction must be fast. The system we analyze is complex, therefore predicting the behavior of the system using traditional methods, such as simulation, would be slow and inefficient. Hence we explore alternative options.

1.1 Queuing Networks

Perhaps the most basic queuing network is what is referred to as an M/M/1 queue [7]. This notation (referred to as Kendall's notation [16]) describes the behavior of the queuing network. The first symbol, which in this case is M, signifies that the arrival rate is 'Markovian' – that is that the inter-arrival rate of arrivals is based on an exponential distribution. The second symbol which is also an M in this example signifies a Markovian service time. This means service time spent on each arrival that is processed by the server is also based on an exponential distribution. The third symbol refers to the number of servers in the queue. Figure 1 gives an example of this very basic queue.

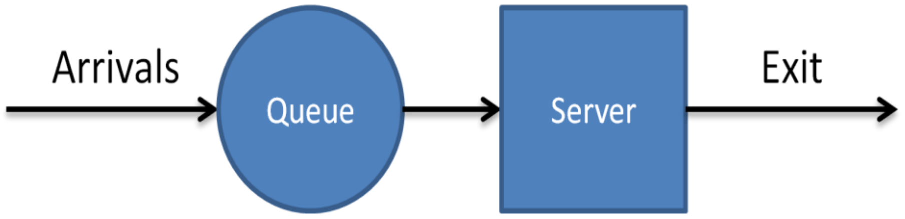

Figure 1 - A basic M/M/1 Queue

In 1917, Agner Krarup Erlang developed formulae for analyzing queues in which both the arrivals rate and service times are Markovian [8]. Very simple formulae may be used to find the average queue length, average waiting time, utilization of the server, etc. [16]. These formulae are commonly used to model call centers and are a basis for most analytical solvers for queuing networks [9]. As such, we will begin modeling using these formulae and use them as a baseline from which to improve.

In Figure 1, the events arrive into the queue following a Poisson process. Therefore, the inter-arrival time distribution is exponential. If the average arrival rate is λ, and the next arrival arrives in T, then:

$$P[T \leq t] = 1 - e^{-\lambda t}$$

The service time may be calculated in the same way. If the server is busy when a new arrival enters the system, then the new arrival must wait in the queue.

1.2 Software and System Specs

Initially we had planned on using a queuing network software package known as LQNS (Layered Queuing Network Solver) to run the experiments for this thesis. However, LQNS does not support queuing networks models in which the number of servers change over time. Since this was an extremely important part of our work, we opted to build our own queuing network modeling and simulation software instead.

In addition to the simulator, we needed to find a more efficient method of anticipating system behavior in order to perform resource allocation and adaptation analytically. Therefore, we developed two analytical solvers to do so. Both solvers as well as the simulator that were developed are described in detail in their respective subsections of Chapter 3.

The run time performance evaluations reported later in the thesis reflect the experimentation on a custom desktop computer with the following specifications:

- AMD Phenom II X4 940 Processor (3.01 Ghz)
- 8GB RAM
- Windows 7 Professional OS (SP1)

1.3 Thesis Contribution

In this Thesis we offer contributions to both adaptive systems and queuing networks as follows. We show that an efficient analytical solver can perform resource allocation in a complex network successfully. More specifically, when given a list of parameters, a list of possible biometric algorithm thresholds, and a set of performance and security requirements for the complex queuing network, our analytical solver will allocate resources in such a way that minimizes cost while creating a high confidence that requirements will be met.

We will show that such an analytical solver may be used to model a complex adaptive queuing network in real time, anticipate future problems with the system, and make adaptations to the system that will alleviate those problems. While there are many papers that show that analytical solvers may be used to verify the integrity of queuing systems, to the best of our knowledge, there are none that use analytical solver as a key tool for implementing an adaptive system.

1.4 Thesis Outline

In Chapter 2 we discuss work related to ours. It encompasses a detailed description of Queuing Networks, Adaptive Systems, and Biometric Systems. Each of these topics will be covered.

In Chapter 3 we discuss the two analytical solvers we developed in order to predict the behavior of the system we are modeling. The first solver we will discuss will be the so called naïve solver. This is the solver used most commonly in modeling simple queuing networks, which makes it suitable to serve as our performance baseline. The second solver we discuss is the advanced solver. We try to optimize it in order to accurately predict system behavior. Lastly, we discuss and attempt to validate the mechanics of the simulator we have developed in order to analyze the system.

In Chapter 4 we introduce capacity planning experiments. The purpose of these experiments is to compare and gauge the effectiveness of analytical solvers. A solver capable of predicting an appropriate resource allocation and the system’s reaction to that resource allocation is considered a success. We will also optimize the advanced solver in this Chapter.

In Chapter 5 we introduce biometric security concerns. The purpose of these security concerns is to add another analysis dimension– the tradeoff between performance and security. This will allow us to begin gauging the robustness of our analytical solvers.

In Chapter 6 we utilize our solver as part of an adaptive system. We discuss each step of the adaptive feedback loop: Collect, Analyze, Decide, and Act. After discussing the methodology for adaptation, we test various scenarios in which our solver attempts to adapt to the situation at hand while being given various system choices.

In Chapter 7 we discuss threats to validity of our experiments. We specifically address identified threats to construct, internal, and external validity.

In Chapter 8 and 9 we summarize our work and discuss how we may improve upon it in the future.

2 Related Work

This Chapter is divided into three sections: Queuing Networks, Adaptive Computing, and Biometric Security.

2.1 Queuing Networks

Queuing Networks have been a very important subject in computing for many years. One issue of ACM Computing Surveys was devoted entirely to queuing networks [10]. In the paper “Queuing Models of Computer Systems” [7], Dr. Allen describes various queuing networks, and more importantly, methods for their simulation. Much of our work has been based on the formulae in Dr. Allen’s paper as well as the Erlang formulas developed by Agner Erlang in his paper “Solution of some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges” [8]. While simulations have been the popular method for accurately gauging the performance of queuing networks, they can be very difficult and costly to construct [7]. In a survey paper by Dr. Spragins, “Approximate Techniques for Modeling the Performance of Complex Systems," he showed various cases in which simple analytical queuing models could be used in place of difficult simulations for complex systems successfully[11].

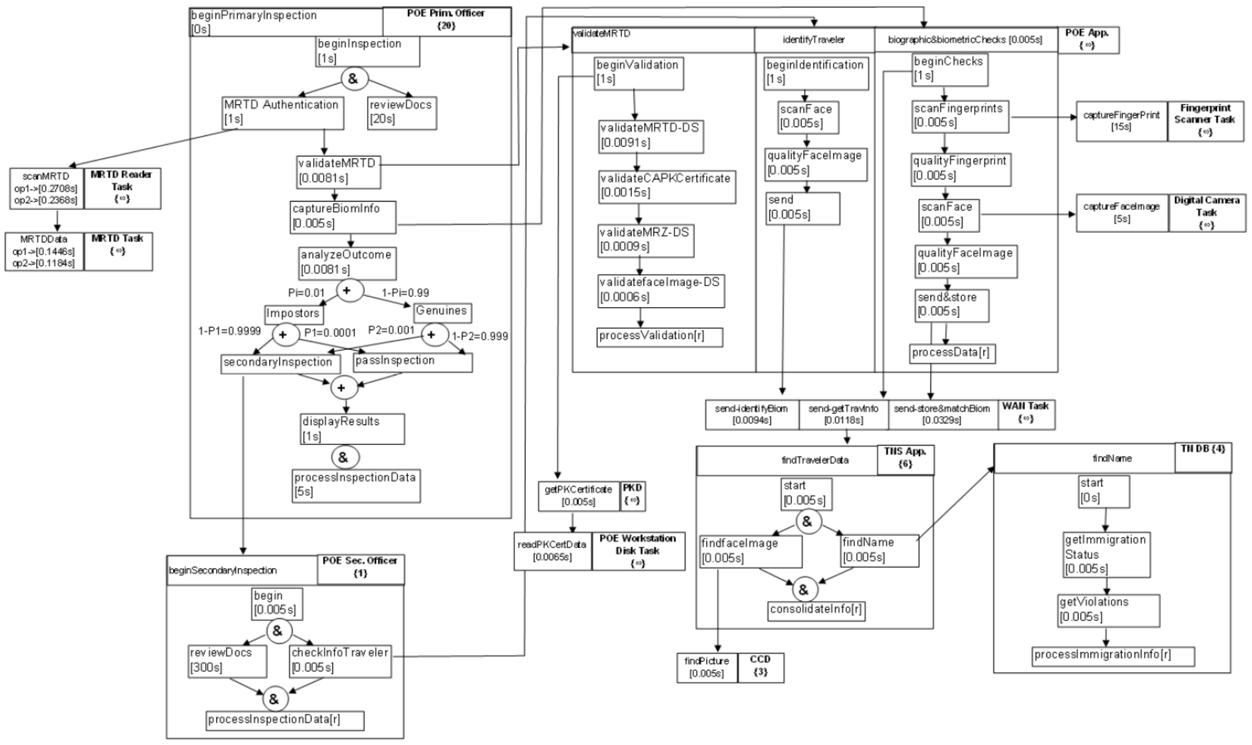

Figure 2 - Model for Airport Border Crossing [1]

The queuing model that serves as the starting point for our research was developed by Mayra Sacanamboy [1]. Her paper forms the basis of the idea that throughput at the border crossing could be increased if the system could adapt to the current situation at hand.

Figure 2 illustrates the queuing network we are modeling. This network describes the biometric inspection of passengers arriving from international flights. This network contains two passenger queues. Passengers who arrive at the airport first enter the primary officer queue (POQ) where they will wait for an available primary officer (PO) to assist them. If they pass inspection, they exit the system. If they fail to pass the inspection, they enter the secondary officer queue (SOQ) where they must wait for an available secondary officer (SO) before undergoing a more thorough inspection. Passengers who enter the SOQ are removed from the system after the secondary inspection is complete. A simpler representation of this process is illustrated in Figure 3.

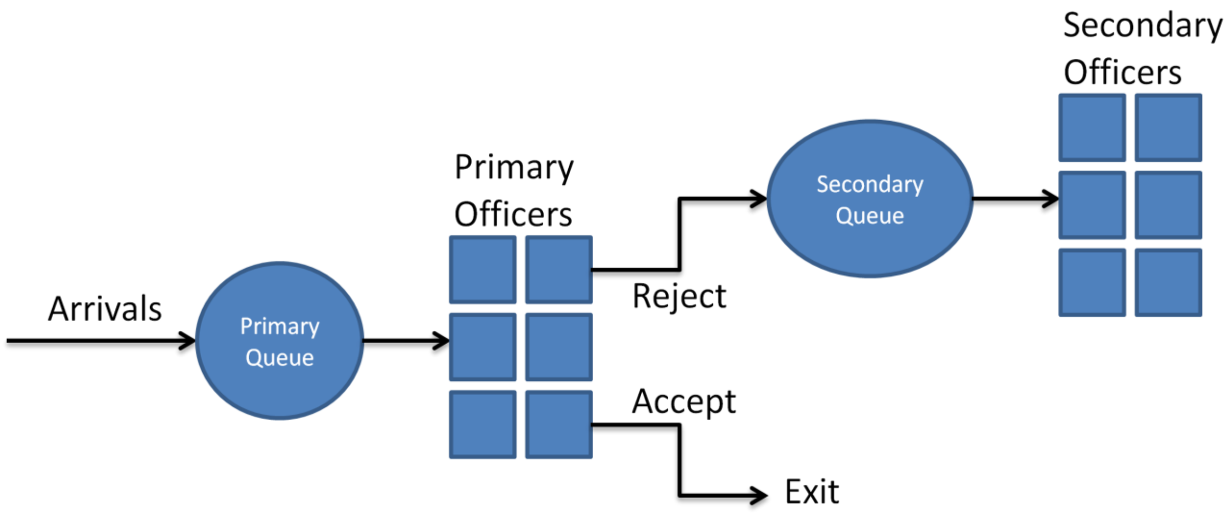

Figure 3 - Simplified Model for Airport Border Crossing

Arrival rates are typically described through a Poisson distribution. The inter-arrival rates of a Poisson distribution are acquired by pulling variates from a negative exponential distribution as follows. The time **t,** in seconds, until the next passenger arrives is $t = -\frac{\ln(u)}{r}$, where **u** is a uniformly distributed random value between 0-1, and **r** is the average arrival rate of passengers (in passengers/second).

Arrivals for the system described herein, however, are based on a schedule of plane arrivals. When a plane ‘arrives’ to the system, passengers quickly exit the plane and arrive at the queue following an exponential distribution as described above. This method of arrivals showing up in short bursts is quite different from a standard Markovian arrival scheme and thus creates issues when attempting to model analytically. Those issues are vetted in Section 4.

The network described here also differs from traditional queues in that the number of servers, represented by the primary and secondary officers, changes over time according to a server schedule. This makes our model much more difficult to solve analytically.

2.2 Adaptive Computing

Much of our research was based on the adaptive computing systems research road map [12] This paper describes various types of adaptation including their benefits and pitfalls.

While sparse, there have been previous papers in which analytical solvers for queuing networks are used alongside adaptive systems. One such paper is “Queuing Models for Analysis of Traffic Adaptive Signal Control” by Mirchandani [13]. This paper is very informative and has inspired ideas herein, the adaptive strategies were based on a paper by Newell, “The rolling horizon scheme of traffic signal control. [14]” Mirchandani’s paper closely resembles the work that we’ve done.

Cortellessa et al. predict adaptive behavior where performance risk is measured by the probability of the violation of performance requirements [20]. Like our work, this potentially gives an increase in the potential for speed in an adaptive system over simulation. This is inherit in any method for predicting system behavior given a discrete set of inputs.

In related papers, the authors explored the optimality of biometric modality and threshold selection through a statistical visual tool called cost curves [22].

2.3 Biometric Security

Biometric security itself was not a prime focus of this paper. However, the biometric algorithm chosen by the solver has a huge impact on the resulting model. The tradeoff between performance and security plays an important role in the system and thus it is important to understand how it works. The performance/security tradeoff is described in great detail in "Biometric security technology" by Faundez-Zanuy [15].

The correlation of biometric security in an adaptive system, and thus using tools to manipulate the biometric device for a performance increase was the prime focus of the paper by Cukic and Sacanamboy [21]. In this paper the authors show an enormous potential for performance increase by altering the biometric device. More information may be found in Sacanamboy's thesis in which she laid the groundwork for this document [1].

3 Modeling Techniques

The model we describe represents a 24-hour period beginning and ending at 4am. We chose to begin and end at 4am due to the fact that planes rarely arrive between the hours of 1am and 7am [6], and thus we would be less likely to have passengers in the system at the beginning or end of simulation. We use publicly available information about flight arrivals for Indianapolis International Airport [6]. For each hour of operation, our solvers determine a set schedule of primary and secondary officers. This schedule is determined by the solver's best estimate at the minimum resource cost for meeting performance requirements.

3.1 Naïve Model vs. Detailed Arrivals

We derive two analytical models of traveler inspection system, based on the arrival assumptions placed upon the primary queue (PQ). In a simple model (naïve model), passengers randomly arrive into the queue following Poisson distribution throughout the hour. This assumption makes calculating the average wait time of a passenger simple. However, this model ignores the fact that travelers arrive into the PQ in bursts, following plane landings. In order to increase modeling fidelity a second more advanced model acknowledges that each plane arrival creates its own burst of passenger arrivals.

3.2 Analytical Solvers

In this paper the purpose of the solvers is to take information about the system and generate a schedule of primary and secondary officers that meets performance requirements as shown in Figure 4.

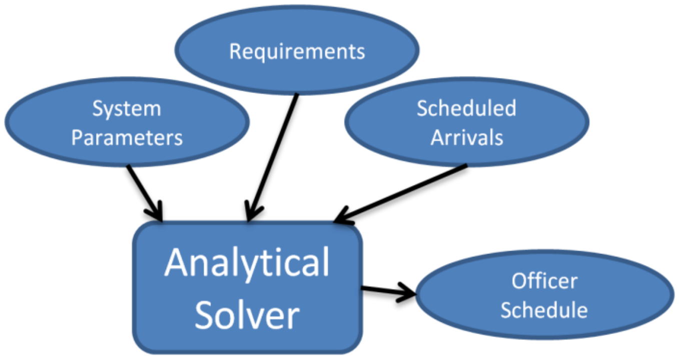

Figure 4 - Optimizing Resource Allocation

Each analytical solver must be capable of predicting both the average waiting time of passengers, as well as the percent of passengers that will be arriving in under X minutes. This will allow us to feed a schedule of plane arrivals to the analytical solvers and have them return a minimal schedule of officers that they predict will meet performance requirements.

3.2.1 Naïve Analytical Solver

The naïve analytical solver (NAS) is based on Erlang - C system description [8]. They are used to predict performance parameters, such as average waiting time, average server utilization, etc. Passengers who are unable to immediately get service must wait in a queue. The naïve model is described by the system of formulae in Figure 5.

$$AST = average\ service\ time$$

$$x = passengers\ per\ second * AST$$

$$n = number\ of\ servers$$

$$\mu = service\ rate\ of\ one\ server$$

$$B(n,x) = \left(\frac{x^n}{n!}\right) / \sum_{i=0}^{n} \frac{x^i}{i!}$$

$$C(n,x) = n * \frac{B(n,x)}{(n - x * (1 - B(n,x)))}$$

$$AWT = \frac{C(n,x)}{\mu * (n - x)}$$

$$P(t) = 1 - C(n,x) * e^{-(n-x)*t/AST}$$

Figure 5 - Erlang formulae

In Figure 5, B(n, x) is the *blockage* rate, which is a value strictly used in Erlang - B systems. It is used in the calculation of C(n, x), the probability of delay for an Erlang C system. This probability of delay is needed to calculate both the average wait time (AWT) and the probability that a passenger receives service in less than ***t*** seconds, P(t), assuming that *t* is the parameter specified in system requirements. Therefore this set of equations will be the predictive mechanism in which we check to see if an officer schedule meets performance requirements.

In addition to assuming a constant traveler arrival rate throughout the modeling period, this model suffers from one more simplification. Each modeling period, in our case one hour, is assumed to be independent of others. Therefore, there can be no hand-over of travelers who do not clear the inspection in the same time period they arrive into the next.

3.2.2 Advanced Analytical Solver

The advanced analytical solver (AAS) is built to monitor quick bursts of traveler arrivals and measure their impact on the system performance better than the NAS.

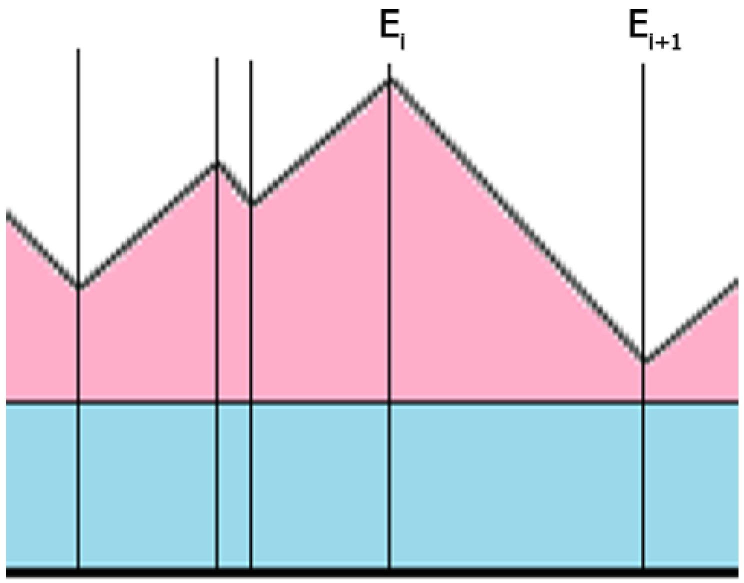

Figure 6 - Pseudo code for the AAS

The AAS works by breaking the entire day of plane arrivals into a list of events that happen throughout the day. At each event point you begin with the current number of passengers in queue. You then estimate the rate of change in passengers in the system until the next event (this is calculated by subtracting the rate that passengers are serviced and leave the system from the rate that passengers are entering the system). You then calculate the total waiting time of all passengers between the events by calculating the area under the event line (EL).

In the primary queue there are three types of events in which the net rate of change in passengers in the system changes. These are a change in the number of officers, a plane arrival, and a plane finishing the unloading of passengers. Consider the event marked E_i in Figure 6. This event marks that a plane has just finished unloading. The slope of the EL between E_i and E_{i+1} is determined by the number of primary officers and their rate of service, and also by the rate of passengers entering the system. Since no passengers are arriving into the system, the rate of passengers entering is zero, and the slope is negative. It is possible, if there are enough primary officers servicing passengers, for the slope to be negative even while one plane is unloading. However it is also possible for multiple planes to be unloading at one time if a second plane arrival event is analyzed before the first finishes unloading.

As the graph is being built, each event must measure its contribution in waiting time. This is easily calculated by measuring the area under the line. Since the height of the line represents the estimated size of the queue at that point, the area underneath represents waiting time in the queue. If you notice in Figure 6, there is a blue section under the waiting time section. This represents the utilization of the primary officers themselves. Passengers in these areas are being serviced and thus are not contributing to wait time. Only the areas above the blue sections, but below the ELs are calculated. The sum of all waiting times gives us a total waiting time for the day. We simply divide this by the total number of passengers in order to get our average waiting time for the day.

Calculating the percentage of passengers arriving in under X minutes is a more difficult problem. We do so by generating threshold lines (TL) on the graph that represent the point at which a passenger, based on the rate of service, will be waiting exactly X minutes in the queue.

Consider a situation in which there are ten officers, and working together they service ten passengers per minute. Also consider that we would prefer passengers be serviced in under 30 minutes. This means that if a passenger enters the queue and it increases the queue size to 300 passengers (which is a total of 310 in the system considering the utilization of officers), then it is estimated that passenger will wait exactly 30 minutes in the queue. Any passenger that arrives in position 300 or greater of the queue will wait longer than the threshold.

At the beginning of each hour the number of primary officers may change. This causes the TL that exists X minutes before the change to be sloped from the previous hour to the new one. However, there will only be at most forty-eight TLs, and only one TL at any given point in time.

Once the TLs are calculated, the problem becomes simple. For each EL you check to see whether it exists above the TL, below the TL, or intersects with the TL. If above, any passengers arriving during the EL (based on the rate of arrivals) will be added to the number of passengers that don't meet the threshold. If below, nothing happens. If they intersect, only the portion above the TL will contribute to the number of passengers not meeting the threshold. Since we want to ensure that 99% of all passengers make it in under the threshold, if greater than one percent of the total number of passengers falls on or above the TLs then the schedule is rejected as unable to meet performance requirements.

Performance for officer schedules generated for both the NAS and AAS are reported in Section 3. The drawbacks of AAS is the assumption that planes arrive exactly when they are scheduled and that passenger unloading rates and POs service times are constant.

3.3 Simulation Engine

The simulator has been designed to closely follow system requirements. Plane arrivals are processed in chronological order, creating passenger arrival events into the PQ. Twenty thousand passengers' inter-arrival times were created from a negative exponential distribution, shown in Figure 7.

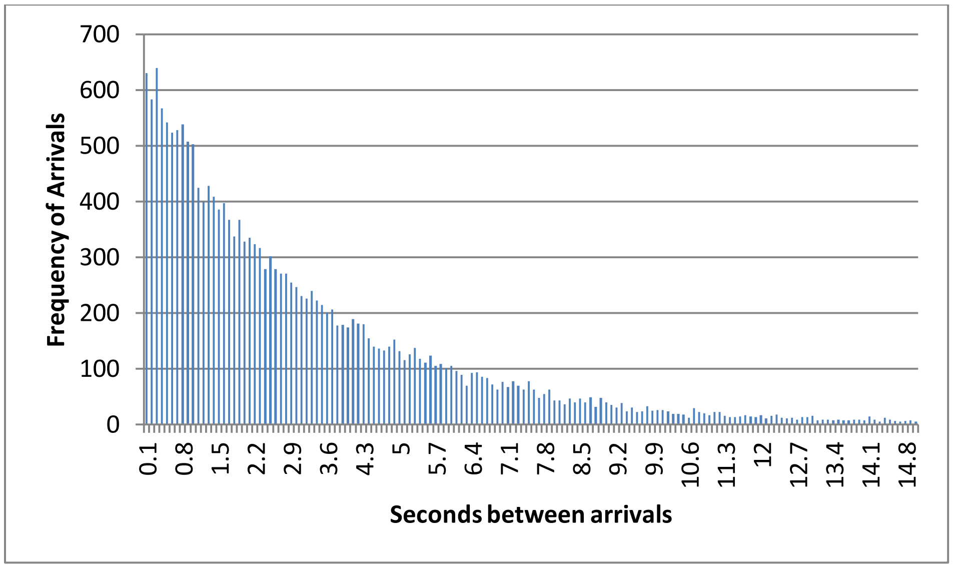

Figure 7 - Inter-Arrival Rate of Passengers

The assumptions regarding the inter-arrival rates of passengers were made in discussion with the experts. We were told that it takes about five minutes to unload a 96-passenger plane. Thus the average inter-arrival rate for passengers arriving to the primary queue is estimated at 3.125 seconds per passenger.

Simulating service time travelers experience when they meet an immigration officer, however, creates a complicated technical problem. Given a set of actual service times by the POs, a best-fit distribution could be estimated. However, since such information is not available to general public, we assume an Erlang distribution of service times. This assumption will make it easier to internally validate the results. We are aware that external validation of this parameter is lacking, although we had a chance to discuss the model with domain experts and have used their suggestions in the selection of parameters [26], [27].

Once a passenger is processed by the primary officer, whose service time is given from an Erlang distribution, the passenger is either removed from the system or placed into the secondary queue. The inspection outcome follows a random value from a uniform distribution between zero and one. If this value is less than the false non-match rate (FNMR) for a genuine passenger, or (1 – false match rate (FMR)) for an imposter, they are placed into the secondary queue.

For the purpose of modeling the system as accurately as possible, the simulator has the option of adding delays to the scheduled arrival times of planes. These delays follow a Normal distribution with mean equal to *-163.8* seconds and variance equal to *825* seconds [2].

3.3.1 Simulator Validation

In order to make sure that our simulation engine is producing reasonable results, we chose to create an Erlang system which can be compared to the results of an Erlang calculator, as described in Section 3.2.1. For this model we included a single plane at time zero with 21,000 passengers. We assume no imposters and zero FNMR, thus no passenger ends up in the secondary queue. We set the number of servers to 14, the average inter-arrival rate to 3.125 seconds, and the average service time for the primary officer to 42 seconds.

Using Erlang equations we've determined that the average waiting time of a passenger should be approximately 62.48 seconds. Since this value is a measure of the system in a steady state and the first few passengers would be entering the system with zero wait time and utilization, we omitted the first thousand passengers from each simulation in our results.

To test the system we developed the following null and alternative hypotheses:

- *Null hypothesis (H_0):* The simulator cannot accurately measure the average waiting time of passengers in an Erlang system.
- *Alternative hypothesis (H_1):* The simulator can accurately measure the average waiting time of passengers in an Erlang system.

We repeated the simulation a thousand times, each time measuring the average waiting time of the last twenty thousand passengers. The overall average wait time of all one thousand simulations was 61.62 seconds with a variance of 517.1 seconds squared. Figure 8 shows a histogram of the frequency of average waiting times of all one thousand simulations.

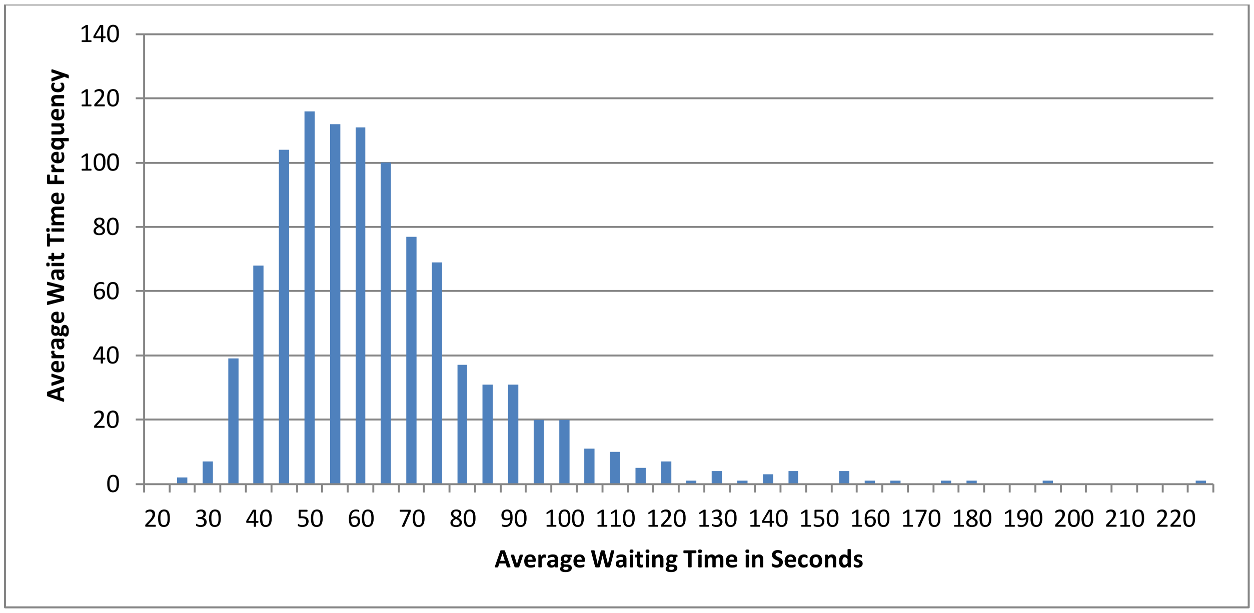

Figure 8 - Average Waiting Times Obtained from Simulation

We performed a one-sample t-test on the data comparing our expected value of 62.48 with simulation outcomes using $\alpha = 0.05$. We calculated a t-value of -1.201, and a two-tail critical value of 1.962. Since the absolute value of t is less than our two-tail critical value, we can reject the null hypothesis with 95% confidence and conclude that our simulator is producing the expected values.

3.4 Solvers and Simulation Run Time

In order to generate the following table, we ran 1000 iterations of the simulator (without adaptation as discussed in Section 6), 1000 iterations of the NAS, and 1000 generations of the AAS. These tests were done using parameters as discussed later in Section 5.3 after integrating security concerns, thus measuring the time requirement of our tools in a full practical application of the system.

The tests were performed on the computer discussed in Section 1.2. The tests were timed using the computer's system clock that initiated directly before entering the testing module and finished directly following each testing module. Four significant figures were considered.

Tool	Time (seconds)
NAS	0.021
AAS	3.811
Simulator	791.772

Table 1 - Time required for 1000 iterations of various tools

4 Capacity Planning Experiments

In this section we analyze the methods we have developed for predicting performance using a series of capacity planning experiments. Capacity planning will estimate the number of POs needed at each hour in order to meet performance requirements. The performance we are predicting with these experiments is whether or not performance requirements will be met with the proposed schedule of POs.

4.1 Description of Variables

We have been given two system-level performance requirements. The first states that the average wait time for passengers is less than fifteen minutes (including wait time in both the primary and secondary queue). The second requirement is that 99% of all passengers spend less than 30 minutes in the primary queue.

In these experiments, our biometric subsystem utilized Viisage (V-Norm) facial recognition algorithm with uncontrolled lighting, implying the FNMR of *0.146* when the FMR is 0.001 [3]. We assume an average passenger inter-arrival rate of *3.125* seconds and an imposter probability of *0.0001* (one in ten thousand). We set the average service time of the PO to *42* seconds, and the average service time of the SO to *300* seconds. These values were recommended through our interactions with domain experts either in private conversations or in the relevant technical reports [24], [25].

Our dependent variables include the estimated waiting times of the analytical solvers, the average waiting times from simulation. We calculate the percentage of all simulations that meet the stated performance requirements.

We analyze the NAS versus the simulator with both scheduled arrival times and delayed arrival times. We also test the AAS against the simulator with scheduled arrival times and delayed arrival times. Each simulation is performed one thousand times.

4.2 Capacity Planning Results and Discussion

Since analytical solvers can only work with scheduled arrival times, the capacity planning was the same for both the scenario in which planes arrive on time and when planes arrive with delays. While both solvers created similar schedules, their performance was very different. Figure 9 shows the number of primary officers scheduled for each hour of the day along with the estimated number of passengers.

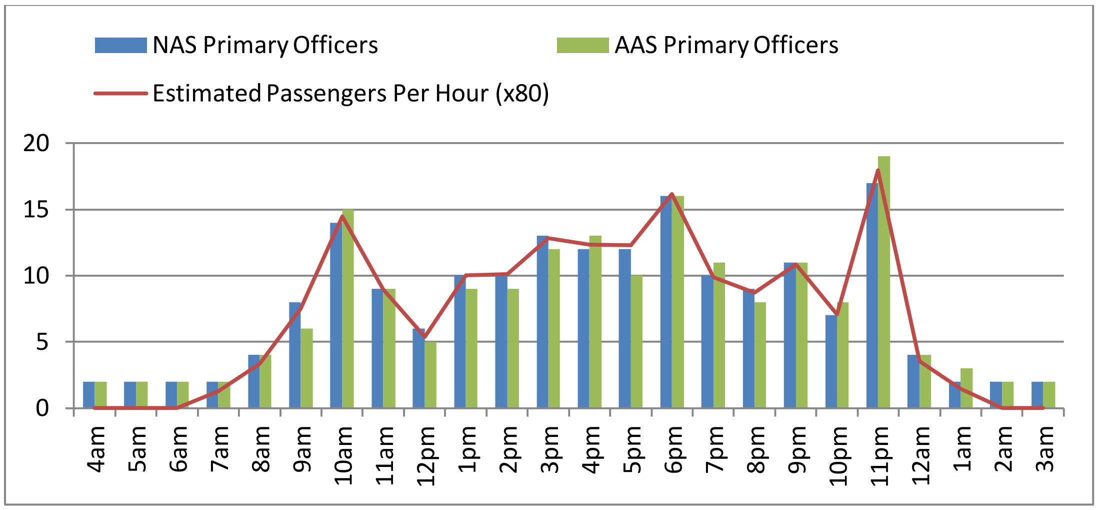

Figure 9 - Primary Officers Scheduled by Analytical Solvers

The results from the analytical solvers using the scheduled arrival times are shown in Table 2. Table 3 shows results when planes are given random delays.

	Sched. Hours	Est. Ave. Wait Time	Actual Ave. Wait Time	% Meeting Perf. Reqs.
Naïve	370	514.0s	597.2s	0%
Advanced	373	527.6s	531.1s	50.9%

Table 2 - Comparison of Analytical Solvers assuming planes arrive exactly as scheduled

	Sched. Hours	Est. Ave. Wait Time	Actual Ave. Wait Time	% Meeting Perf. Reqs.
Naïve	370	514.0s	706.1s	24.3%
Advanced	373	527.6s	698.7s	45.1%

Table 3 - Comparison of Analytical Solvers when delays are simulated for planes

Tables 2 and 3 suggest that the AAS is outperforming the NAS by a great deal, albeit with a slightly higher number of scheduled hours. However, to truly gauge their predictive capabilities, we must compare their estimated waiting time to actual waiting time by the hour. Figure 10 compares what the NAS believed would be the wait time per hour against the median and average wait times over *1,000* simulations.

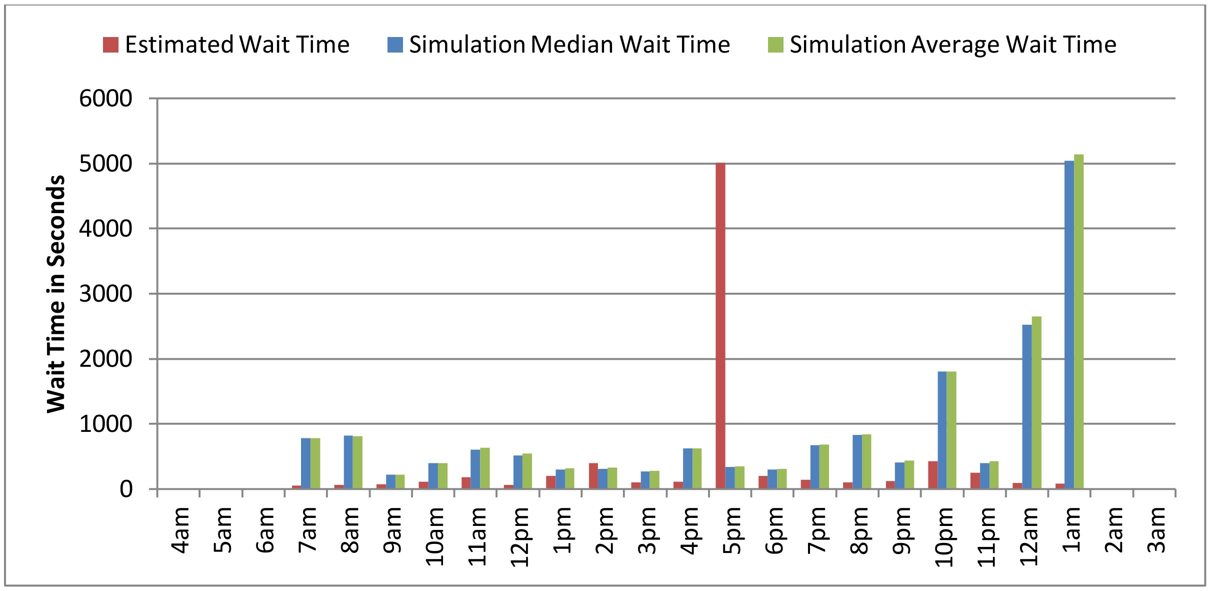

Figure 10 - NAS estimated wait time per hour versus wait times in simulation

Figure 10 shows that NAS does a very poor job at predicting passenger wait time. By assuming passenger arrivals are distributed evenly over each hour, the NAS vastly underestimates the wait times. The large spike in estimated average wait time at 5pm is a result of the NAS deciding that it has leeway in meeting the performance requirement due to underestimating most waiting times. It just so happens that passengers entering the secondary queue at 5pm could wait hours for service and overall daily performance requirements would still be met. However the NAS assumes each hour is independent of subsequent hours, thus when the number of secondary officers increases by four in the following hour any accumulated queue is quickly dissipated.

Where NAS proves the least reliable, however, are the last few hours of arrivals. The largest spike in arrivals occurs between 11pm and 12am, and then dissipates greatly for the following two hours. Due to the underestimation of waiting times, the NAS under-schedules the number of primary officers between 11pm and 12am. As a result, a large queue is left over when the number of POs drops to four and then two. Thus passengers arriving between 12am and 2am are placed in the back of a larger than expected queue. This increase in primary queue waiting times causes the NAS to fail to meet the performance requirement that 99% of passengers in the primary queue are serviced in under 30 minutes.

Figure 11 compares AAS wait time predictions against the median and average wait times over *1,000* simulations.

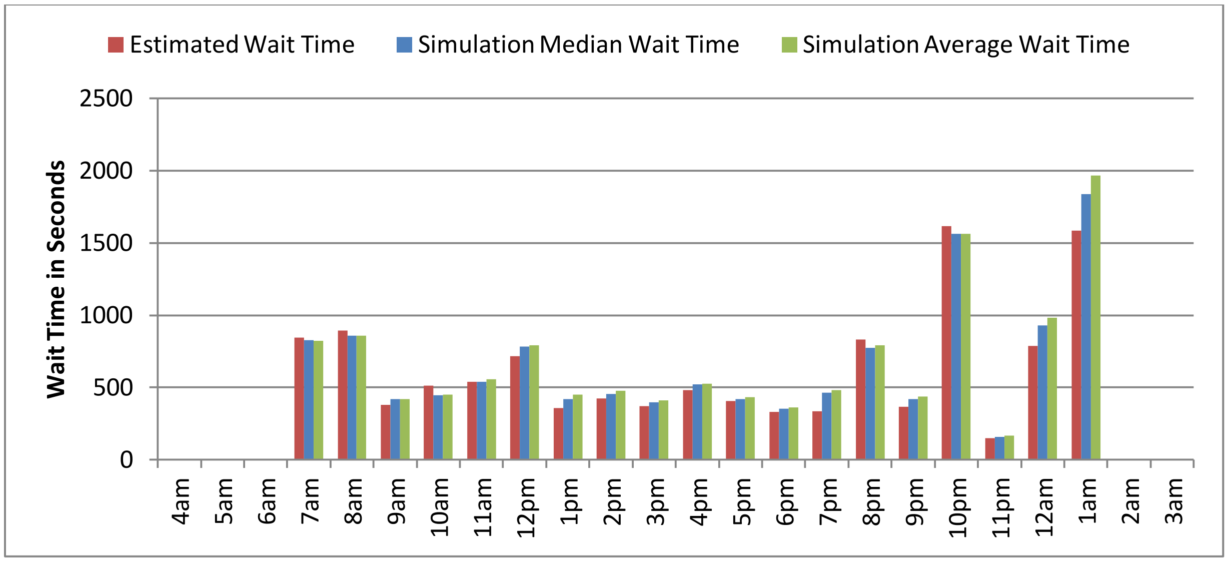

Figure 11 - AAS estimated wait time per hour versus wait times in simulation

The AAS is far superior to the NAS in estimating passenger wait time. In some cases it seems to underestimate or overestimate the waiting time, but at a small degree.

4.3 Optimization

While the AAS performs well, only 509 of the 1,000 simulations runs have managed to meet performance requirements. The 491 that failed did so because less than *99%* of passengers in the primary queue received service in less than 30 minutes. This result is due to the fact that the analytical solver attempts to find a point in which exactly 99% of all travelers are inspected in less than thirty minutes. Therefore, the randomness of the simulator causes approximately half of the results to fall below that threshold. The distribution of passengers serviced below the thirty minute threshold for the simulation of AAS vs. scheduled arrivals can be found in the histogram in Figure 12.

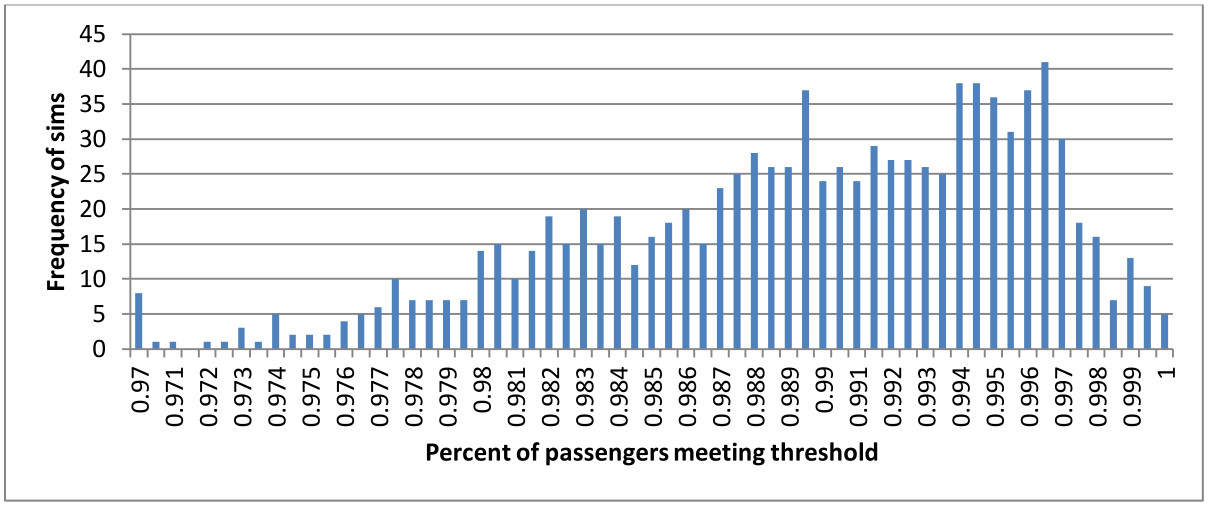

Figure 12 - Distribution of percentages of passengers being serviced in under the required threshold

Let's assume that we want to increase this value to a 95% confidence. In order to do so, we introduce a new value into the analytical solver called the leeway value, depicted herein by **£**. This value gives us some leeway in ensuring that the simulation has a significant confidence that performance requirements are met.

In order to find this value, it's important for us to know which thresholds were met in our previous simulation. We ran another series of distributions using the parameters in Section 3.2 using the AAS and the scheduled arrival times in the simulator. We then plot the distributions of thresholds met in a histogram as shown in Figure 13.

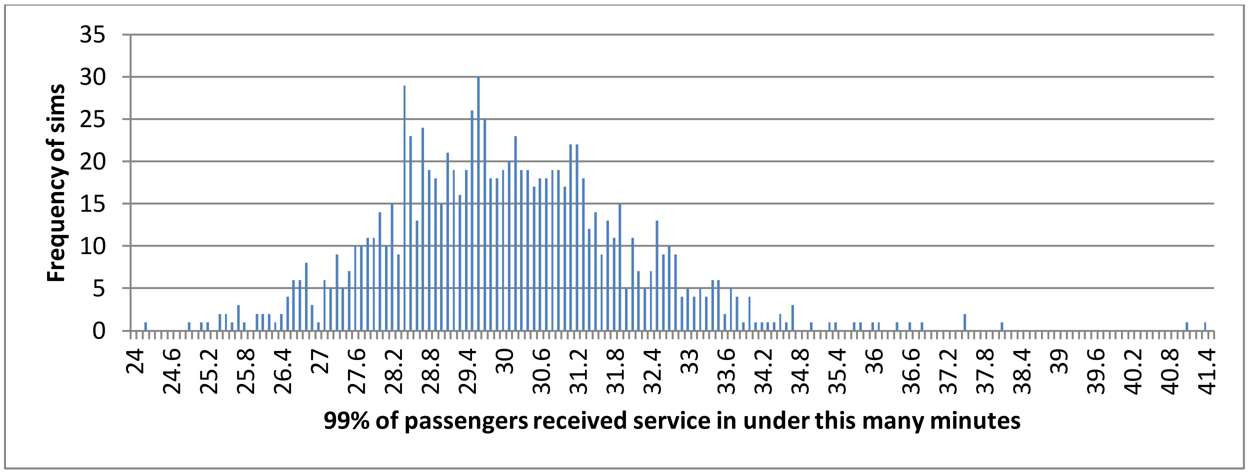

Figure 13 - Distribution of thresholds

As we can see, the distribution of thresholds is very similar to a Normal distribution. Given the standard deviation, we can determine for which value 95% of our simulations were below. We may calculate the value at which 95% of simulations successfully met the threshold, **v**, and ultimately the leeway value, **£**.

$$\mu = average(A)$$

$$z = 1.645$$

$$\sigma = stddev(A)$$

$$v = \mu + z * \sigma$$

$$£ = \frac{\mu}{v}$$

The value **z** is a confidence interval ensuring 90% confidence that a value exists between two points equally distant from the mean. However, since we're fine with values appearing in the left tail, we get a 95% confidence that a particular result doesn't appear in the right tail of the distribution.

The way we ultimately find the leeway value is by first finding the value, **v**, satisfied by the original set of parameters. This tells us that, in order to have 95% confidence that an officer schedule meets the security requirement that 99% of all passengers get service in the primary queue in under **v** minutes, we must set the AAS to create a schedule given the performance requirement that 99% of passengers get service in the primary queue in under **v * £** minutes. It then goes to imply that in order to have a 95% confidence that 99% of passengers in the primary queue are serviced in under 30 minutes, that we must assume a 30 *** £** requirement when running the AAS.

In the case of the simulation above, we found a **£** value of 0.900839. By multiplying the requirement as it is read in to the analytical solver by our leeway value, we end up with new results for our now 'optimized' advanced analytical solver (OAAS). The results of the OAAS as compared to the AAS using the same parameters from 4.2 while the simulator uses the scheduled arrivals are in Table 4.

	Sched. Hours	Est. Ave. Wait Time	Actual Ave. Wait Time	% Meeting Perf. Reqs.
AAS	373	527.6s	531.1s	50.9%
OAAS	375	512.0s	510.9s	96.6%

Table 4 - Comparing AAS and OAAS against the simulator while using the scheduled arrival times

The OAAS is now able to meet performance requirements with a significant level of confidence.

In order to determine whether or not the leeway value remains consistent over varying thresholds, we repeated the experiment finding new leeway values for simulations with thresholds ranging from 15 minutes to 60 minutes in five minute increments. The findings are in Figure 14.

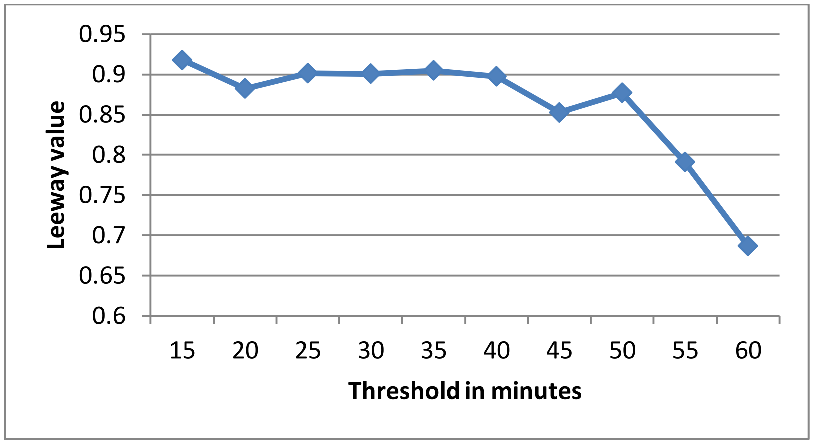

Figure 14 - Leeway values derived from performance requirement thresholds

The leeway values remain consistent for the first 40-45 minutes and then begin to drop off as the standard deviation of our results grows faster than the increase in threshold.

5 Integration of Security Concerns

Up until this point we have been focusing solely on performance requirements. In this section we introduce security requirements and examine how our system can adapt to varying system parameters from day to day.

5.1 Performance vs. Security

As discussed earlier, any given biometric algorithm has both an FMR (false match rate) and an FNMR (false non-match) rate. However, almost all biometric devices may vary their FMR and FNMR based on a configurable threshold.

If a lower, more lenient threshold is chosen then passengers are less likely to be rejected by the system. This means that the FNMR decreases. As a result, less passengers need to enter the secondary queue and performance improves. However, FMR increases and thus imposters are also more likely to be accepted. So this increase in performance is accompanied by a decrease in security.

If a higher, less lenient threshold is chosen then the exact opposite situation occurs: more passengers are rejected. Thus more passengers enter the secondary queue and performance suffers. However, imposters are less likely to be accepted and security improves. Figure 15 illustrates this trade-off with various facial recognition algorithms (along with human performance).

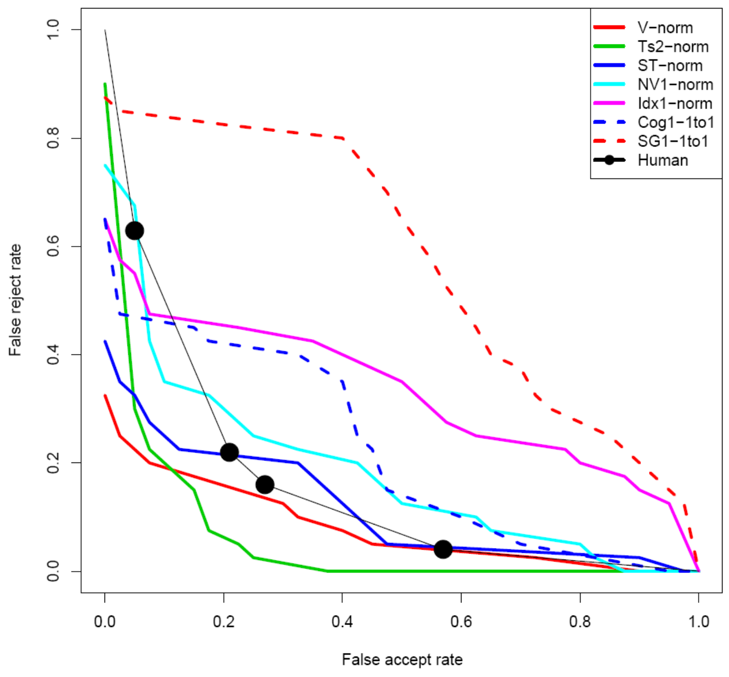

Figure 15 - ROC curves of various facial recognition algorithms [3]

5.2 Identifying Biometric Threshold

Given that we are using a static security requirement (as opposed to a cost heuristic), the requirement and parameters of the system alone dictate which thresholds are available to use for a given day. If the static requirement is written as "*Less than or equal to x imposters may successfully pass the primary officer in a given day*," then the maximum acceptable FMR can be calculated as follows:

$$FMR_{max} = \frac{x}{(p_{imp} * \#pass)}$$

Where p_{imp} is the probability of an imposter and #pass is the total number of passengers arriving on that day.

While it's obvious that the number of passengers varies from day to day, the security requirement and probability of imposter may also vary due to changes in foreign affairs or other external events. Thus the ability to identify a new maximum FMR each day is important.

Once a maximum FMR has been established, we must maximize performance and minimize cost. This is done by implementing the threshold with the lowest FNMR out of the remaining thresholds available.

Table 5 shows some of the thresholds of an undisclosed NEC algorithm extracted from a 2010 NIST report on facial recognition accuracy [5].

Threshold	FMR	FNMR
1	0.01	0.024
2	0.001	0.037
3	0.0001	0.055
4	0.00001	0.065

Table 5 - NEC algorithm thresholds

If, for example, we were to model the day we've been exploring throughout this paper in which there are 13,928 total passengers, and we assumed an imposter probability of 0.00001 (one in a hundred thousand) with a security requirement that the probability of an imposter getting through in a day must be less than or equal to one in ten thousand (0.0001), then our maximum FMR would be equal to 0.000718. Thresholds 3 and 4 both meet this requirement. Since threshold 3 has a lower FNMR it would be chosen for performance purposes.

5.3 Testing and Discussion

In this section we experiment with various levels of security requirements and see how they affect our performance and capacity planning using our OAAS.

For the following experiments we are using the same performance requirements, service times, and passenger inter-arrival rates that we used in Section 3.1. We use the OAAS solver for capacity planning. The probability of an imposter and the security requirement (and in turn, the algorithm) vary between experiments as listed.

We run three experiments, each chosen to simulate varying states of security. They are listed in Table 6.

	Probability of Imposter	Security Requirement	FMR_{max}	Threshold
Test 1	0.000001	0.001	0.07	1
Test 2	0.00001	0.0001	0.0007	3
Test 3	0.00001	0.00001	0.00007	4

Table 6 - Test parameters

Test 1 represents a low-security test. In this scenario it is estimated that only one in a million international arrivals are imposters. Furthermore, security requirements only dictate that the probability of an imposter entering the country on a given date must be less than one in a thousand. Given that we have 13,928 passengers arriving, we need a FMR of less than 0.07 (7%) to meet the security requirement. Threshold 1 fits this requirement and has the lowest FNMR of all available thresholds.

Test 2 represents a medium-security test. Test 3 represents a very high security test. They require the use of thresholds 3 and 4 respectively.

The results of these tests are shown in Table 7. Please note that only secondary servers are listed since chances in security have no impact on the primary queue or officers.

	Secondary Servers Scheduled	Average Waiting Time	Percent Meeting Performance Requirements
Test 1	39	553.07s	95.9%
Test 2	75	534.47s	98.9%
Test 3	87	581.82s	93.8%

Table 7 - Test Results

Figure 16 shows the number of secondary servers schedules to meet the demand caused by variations in FNMR throughout the day. A lower the level of security result in less servers being scheduled.

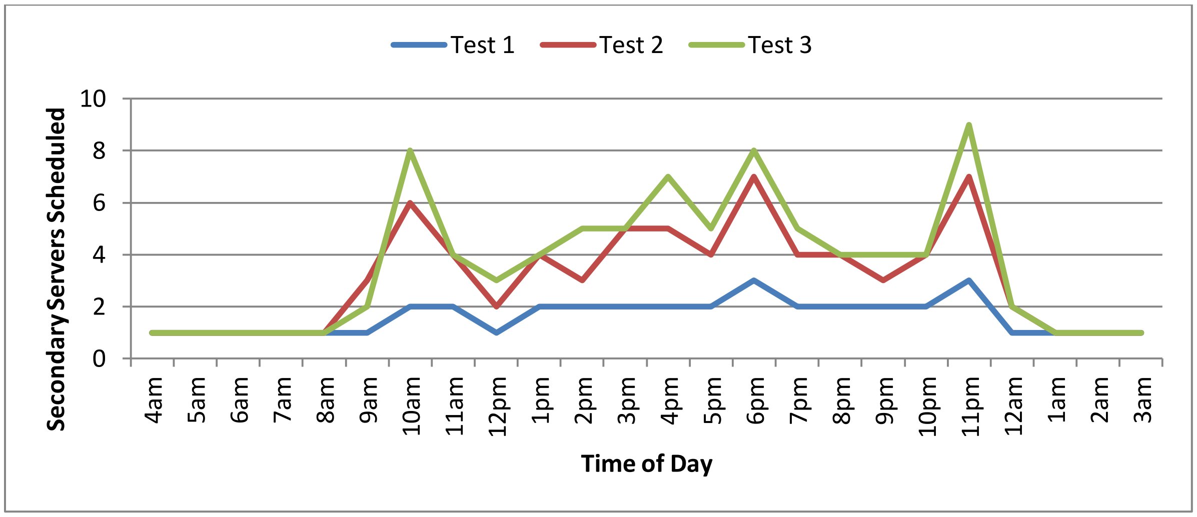

Figure 16 - Secondary Servers Scheduled

6 Adaptation

In order to adapt to unexpected events in system's environment at run time, we must first discuss the method in which our simulator gives feedback to our analytical solver. Figure 17, taken from [12] gives a basic idea of the information flow in a feedback loop of an adaptive system.

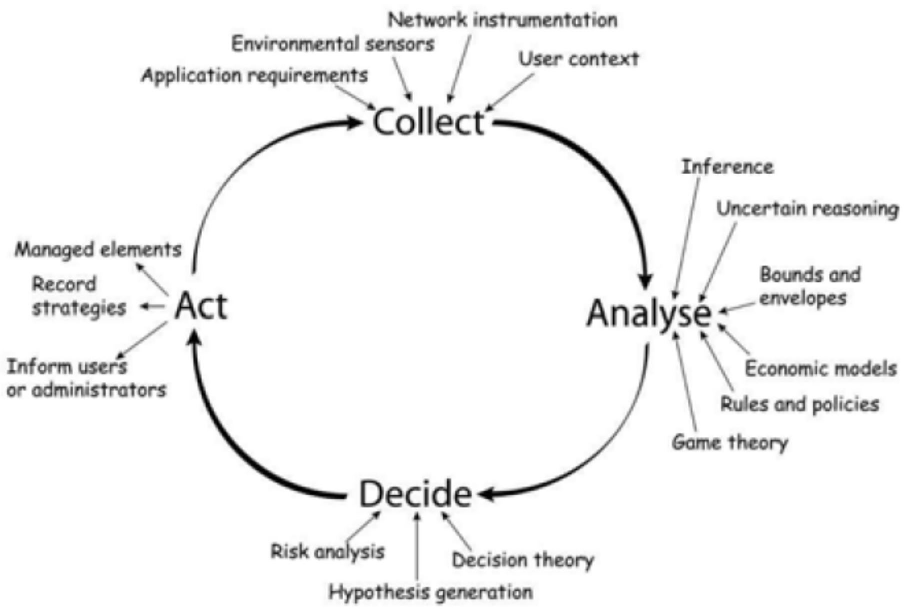

Figure 17 - Feedback loop for Adaptive Systems [12]

The overreaching goal of system adaptation is to meet performance requirements. Based on information received, if the solver decides as though the current configuration of servers does not meet performance requirements then it has the option of making decisions that changes the way the system behaves.

For the purposes of this experiment we have decided that the feedback loop initiates once every five minutes beginning at 4:05am and ending at 2:55am the following day. The reason it ends early is due to the fact that we have set the limitation that adaptations may only be implemented by the hour, and we may only change system performance in time periods that begin an hour after the feedback loop is run.

Furthermore, we only use the delayed plane schedule when testing the adaptive model. This is due to the fact that the important feedback is based on the 'actual' arrival times. Our system adapts to planes that arrive delayed.

6.1 Collect

The first step in the feedback loop of our adaptive system is to collect information about the system and its environment for analysis. This means that we need to determine what information our analytical solver has access to.

Since the goal is to maintain performance requirements, the feedback we need is that in reference to the waiting times of passengers. While current queue size would be very useful to this end, it would be impractical to try and accurately gauge the queue size at a given point in time. Since we have shown in Section 4.2, and more specifically in Figure 11, that the AAS is capable of accurately estimating waiting time using only the plane schedule, this is the data we analyze with the analytical solver.

Other possible choices such as server utilization, PO accept and reject rates, or actual service times would be difficult and impractical to obtain in a real system.

6.2 Analyze

When analyzing the data it's important to keep four pieces of information in mind:

- The original plane arrival schedule
- The actual arrival times of every plane that has landed thus far
- The original server schedule and algorithm threshold
- A list of all adaptations implemented in previous iterations of the feedback loop

The analytical solver is designed to take a schedule of primary and secondary officers along with a plane arrival schedule, and a list of which algorithmic thresholds are being utilized each hour, and to determine if performance requirements are maintained. The schedule of primary and secondary officers may be ascertained using the original schedule along with the list of adaptations made by previous iterations of the feedback loop. The schedule of plane arrivals, however, must be reacquired at each feedback loop iteration, i.e., every 5 minutes.

When determining the most accurate plane schedule there are two situations to consider for each plane:

- All planes that have already arrived are listed as arriving at their 'actual arrival time'.

- All planes that have yet to arrive have their arrival times estimated based on a truncated Normal distribution using their scheduled arrival time, the current time (of the feedback iteration), and the average distribution of delays.

Since the second point may not be immediately clear, we consider the following example:

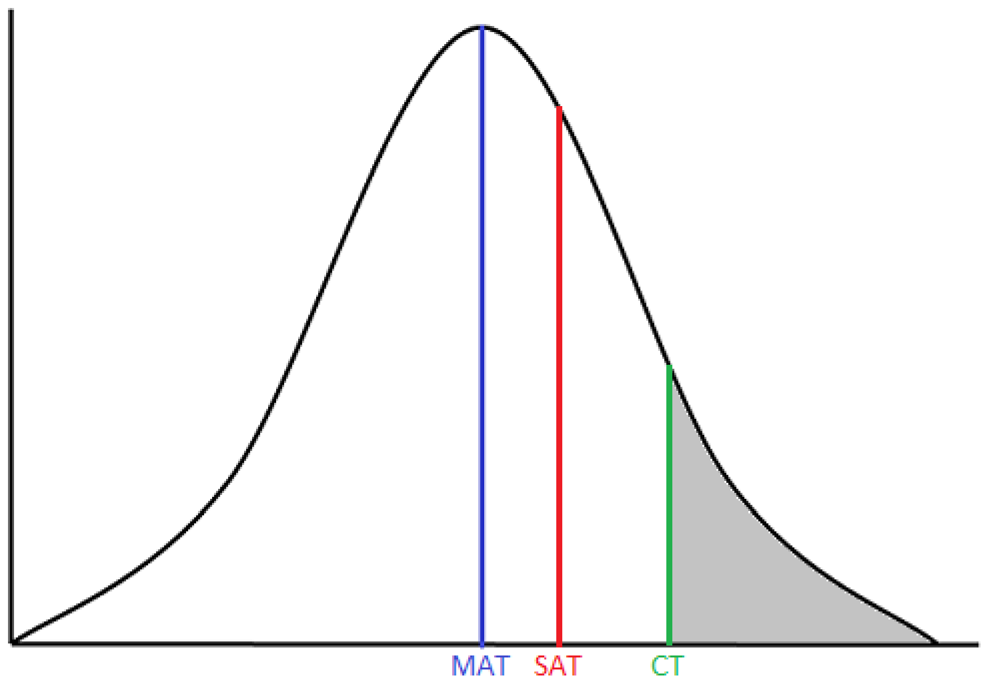

Figure 18 - Truncated Normal Distribution

Figure 18 demonstrates a case in which a plane was scheduled to arrive before the current time and hasn't. The MAT is the mean arrival time which takes place 163.8 seconds before the scheduled arrival time as discussed in Section 3.3 [2]. SAT signifies the scheduled arrival time. Finally, CT signifies the current time (the time at which the feedback loop was initiated). The only thing that system can know about this plane is that it must arrive sometime in the shaded area of the distribution.

The arrival time of the plane, X, given that X must be greater than the current time, CT, is given as follows [18]:

$$E[X \mid X > CT] = \mu + \sigma * \lambda(\alpha)$$

$$\alpha = \frac{(CT - \mu)}{\sigma}$$

$$\lambda(\alpha) = \frac{\phi(\alpha)}{[1 - \Phi(\alpha)]}$$

where $\phi(\alpha)$ is the probability density function and $\Phi(\alpha)$ is the cumulative distribution function [18].

It is important to note that this method for estimating future arrival times is useful for all planes which have not arrived, not only the ones that are known to be late. The analysis step determines, given the modified server schedule and plane arrival schedule, if the system is still on track to meet performance and security requirements.

6.3 Decide

Once analysis is complete it is time to consider making adaptations to the system in order to maintain performance requirements. There are five possible adaptations to our system and we have put them into two categories, simple and drastic.

Simple adaptations are those that are always taken as long as they result in a performance increase. Most importantly, these are the possible adaptations the system considers regardless of whether it was determined that the system will still meet performance requirements. The two simple adaptations are as follows:

- Move a secondary officer to the primary officer position
- Move a primary officer to the secondary officer position

In either of these cases it creates no added cost. If it is determined that moving any number of officers from one position to the other potentially increases performance than the change is made.

Drastic adaptations are those which increase resource cost or threaten to trip the security requirements. These adaptations are only considered if analysis reveals that performance requirements will most likely not be met. Drastic adaptations are as follows:

- Add a secondary officer
- Add a primary officer
- Temporarily reduce the biometric algorithm threshold

Adding an officer undoubtedly increases performance but at the cost of employing an extra server.

Reducing the biometric algorithm threshold also undoubtedly increases performance, but this may cause the simulation to break security requirements. Please note that temporarily reducing the threshold does not always trip the security requirements. However, since planes are randomly delayed and service times are randomly distributed, in the worst case scenario every remaining passenger that has not yet been serviced may be processed during the hour in which the threshold is reduced. However unlikely this may be, the system always treats this as a potential risk.

The only design decision left is to decide whether to prioritize adding additional officers or reducing the algorithm threshold. It really depends on what is most important to the vendor. Since either method could be tested with a simple reordering of the code, we opted to test out both methods. These methods are referred to as Extra Server First (ESF) and Threshold Reduction First (TRF). An alternative would be to allow the addition of additional servers up until a point, then prioritize reducing the threshold, and then finally allow the addition of the remaining available servers in the worst case scenario. We did not test this method due to time constraints.

One last added note is that the system does not implement a set of drastic adaptations unless it believes that by doing so the system is once again be in a position to meet security requirements. This was done to prevent a situation in which the performance requirements fail early and the system responds by downgrading the algorithm for every remaining hour to the lowest threshold while also calling in every possible additional server in vain. If the system is unable to meet performance requirements in this situation then it reverts to only making simple adaptations.

6.4 Act

Once any number of decisions have been made, those decisions are then fed back to the simulator and the alterations are added to the event list at the appropriate times.

6.5 Testing

Testing the adaptive system was done using the same parameters that were used for the testing done in Section 5.3. The security level chosen stipulates that the probability of an imposter getting into the country must be less than or equal to one in ten thousand with the assumed probability of imposters equal to one in one hundred thousand.

The system choices for adaptations were given a few limits to make them more practical. First of all, in no case may all officers be moved from one queue to another. Secondly, only twelve servers may be 'added' to the system (that is twelve server hours total). Lastly, no more than three servers may be added at any given hour. This was done because it seems unlikely that there would be more than three additional servers willing to be 'on call.'

Three experiments were conducted. One thousand simulations were run without adaptation, one thousand simulations were run with adaptation prioritizing the increase in servers, and one thousand simulations were done with adaptation prioritizing lowering the threshold.

6.6 Results and Discussion

The results for the tests from Section 6.5 are shown in Table 8.

Solver	Adaptation	Average Wait	Percent meeting performance Requirements
OAAS	No	682.71s	62.8%
OAAS	Yes, prioritizing more servers	606.68s	73.2%
OAAS	Yes, prioritizing downgrading algorithm	598.25s	75.2%

Table 8 - Adaptive Computing Results

Before discussing these results it is important to make note of some interesting behavior from adaptation. The following points are in reference to the adaptation in which downgrading the algorithm had preference unless noted otherwise:

- In 412 of the simulations, no adaptations were made. All 412 of these simulations met performance requirements.
- In an additional 42 simulations, only simple adaptations were made. The only simple adaptation for these simulations was to remove officers from primary to secondary (and only during the busiest hours).
- When a drastic change was made to reduce the algorithm threshold, it was always accompanied by transferring secondary officers to primary. This was most likely due to the fact that a decrease in algorithm threshold would mean less passengers would move on to the secondary queue.
- In all 248 simulations in which the performance requirements were not met, the OAAS correctly identified that performance requirements would not be met during a feedback iteration. The

OAAS also incorrectly stated that performance requirements would not be met during 27 simulations in which they were.

- In almost all cases in which the OAAS identified that performance requirements would not be met, it did so just before or during the largest spikes in traffic (6pm and 11pm).

It is the last point which helped us identify the reason as to why the adaptive system wasn't creating a larger increase in performance.

Consider what was shown in Figure 11 about the AAS in respect to estimating the average hourly wait time. Compare that to Figure 19 below.

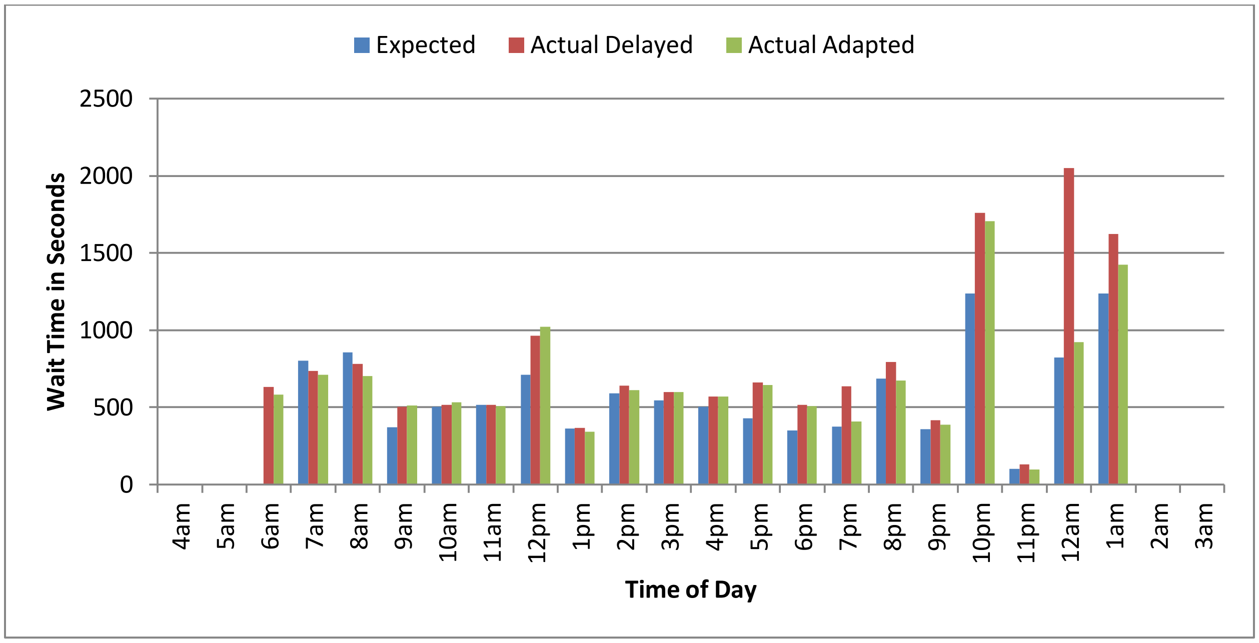

Figure 19 - Waiting Times: Expected versus Actual

Figure 19 shows the comparison between expected waiting times and the actual waiting times from both the normal delayed simulation and the adapted delayed simulation.

In the case of the normal delayed simulation (marked in red) the times in which the solver makes the poorest estimations of waiting times are immediately before and after the 'spike' hours (6pm and 11pm). This makes sense due to the fact that the hours before and after spikes have the greatest potential for unexpected plane arrivals due to planes in the spike arriving early or late.

In the case of the adapted simulation, however, the only significantly poor estimates are during hours that precede the spikes. The hours following spikes do fairly well. This is because the system is adaptive and responds to feedback. It knows before a spike ends whether or not all the scheduled planes have arrived. If not, then it knows those planes will most likely land in the following hour and the system takes the necessary precautions to avoid a drop in performance. However it has no way of knowing that planes will arrive early before the spike, and is thus unable to prepare for them. This is why the OAAS typically reported failure to meet performance requirements nearing the ends of hours proceeding spikes.

The adaptive system we have developed for the border crossing works well for late arrivals, but is still vulnerable to early arrivals with no mechanism for anticipating potential threats. Extensions of the work which help to mitigate these issues are discussed in the next section as well as the Future Work section.

6.7 Early Warning and Extended Delays

In order to further test the robustness of our solution given a complex system, we have devised a final set of conditions and battery of tests to use in conjunction with the analytical solver.

6.7.1 Early Warning Time

Given that our biggest obstacle with the adaptive system is that we are unable to react to early arrivals, we wanted to explore the idea that the airport would most likely have an advanced warning of the actual arrival times of the aircrafts. If early enough, this advanced warning that we call the Early Warning Time (EWT) gives the analytical solver an advanced warning of early arrivals while there is still time to make adaptations in the system.

In order to implement this we need to maintain a working schedule during each run of the simulator that maintains a list of the best-known arrival times for aircrafts. Consider for example that one particular aircraft was scheduled to arrive at 5:05pm but ended up arriving at 4:35pm instead. Without an early warning the solver would be unaware of the early arrival until 4:35, and therefore would be unable to make adaptations in a timely manner. If, however, there were an EWT of one hour, then the solver would become aware of the actual arrival time of the aircraft at 3:35pm. This would allow for the system to make adaptations that could greatly reduce the performance cost.

6.7.2 Call-In Delays

One issue that we have ignored thus far in testing was the delay in the arrivals of officers called in for backup. In Sections 6.5 and 6.6 a new officer could be scheduled to come in with as little as a five minute delay. This is impractical in a real world setting. However, if we set a Call-In Delay (CID) of one hour for example then it becomes much more difficult to react to changes in the schedule.

Given that the EWT and CID are closely related, we explore them both together.

6.7.3 Early Warning Time versus Call-In Delay

The system parameters for the following testing were done using the same parameters from Section 5.3. The analytical solver prioritized downgrading the security setting before adding additional officers when faced with the inability to meet performance requirements. Each combination was simulated one thousand times.

		CID			
		None	One hour	Two Hours	Three Hours
EWT	None	74.8%	64.5%	62.3%	63.7%
	One hour	87.3%	83.4%	71.8%	70.9%
	Two hours	89.2%	87.5%	84.1%	71.5%
	Three hours	89.2%	87.8%	86.8%	83.9%

Table 9 - Percentage of simulations meeting Performance Requirements when considering CID vs. EWT

Any significant CID that isn't offset by an equal or greater EWT reduces performance by a great deal. One possible explanation for this is the fact that it limits the solver to trying to allocate for resources during times in which it has no information of delays.

However, when the EWT is one hour or greater with an equal CID, a significant performance increase is seen. This results from reduction of the security level and the ability to move officers from secondary to primary without any delay. Therefore, any EWT greater than zero helps to fix the problems introduced by early arrivals.

Finally, when given a significantly higher EWT than CID the performance increases by a great deal. This performance increase is almost as high as when working with an exact schedule without the introduction of delays.

6.7.4 Storm Delays

The last scenario that we wanted to explore was a situation in which a storm delayed all arrival times by a significant delay. For this test we chose to delay all plane arrivals between the hours of 10:00pm and 12:00am. Beginning at 12:00am any plane scheduled to land would do so in chronological order with a three minute delay between landings. Thus the first plane whose actual landing was between 10-12 would land at 12:00am, the second at 12:03am, ect. The solver receives word about this delay two hours in advance at 8:00pm.

Initial testing showed that no simulations were able to meet performance requirements under these conditions. This was in large part due to the limitation that only three additional server hours could be scheduled for any given hour. Therefore the solver was given a new potential decision in the simple change category:

- Primary and secondary officers may be delayed until a later time. For example, if five officers were scheduled to work from 4-5 and four officers were scheduled to work from 5-6, then an officer could be removed from the 4-5 interval and added to the 5-6 interval at no cost.

Given this new option we performed one thousand simulations with the delay using the same parameters as in 7.7.3 using an EWT of two hours and a CID of one hour. The resulting simulations showed that 845/1000 or 84.5% met the performance requirements. This was barely any worse than the performance without the delay. The solver simply indentified the problem and shifted the officers into a more appropriate configuration.

6.7.5 Discussion

What we can conclude from our latest round of testing is that our analytical solver remains resilient when confronted with radical changes in the system’s operational environment. We have also seen how better information increases the performance of our adaptive system.

7 Threats to Validity

7.1 Construct Validity

The validity of the results is highly dependent on the accuracy of the simulator. In this paper we used the simulator to represent the real-world queuing network, so any discrepancies in our simulator mean that our conclusions are less likely to carry over to a real system.

Due to the difficulties in obtaining information about international flight arrivals, we had to randomly generate the number of passengers on each plane as well as only ran experiments on the data collected from a single day. As a result there may have been a risk of ‘overfitting’ during optimization for that specific day which would not give as strong results if given new data.

7.2 Internal Validity

Most variables were controlled very carefully throughout the experiments. However, we never tested the effect of a given FAR/FRR pair on the accuracy of the analytical solvers, thus algorithm choice may be a potential confounding variable.

7.3 External Validity

The queuing network described herein is a very specific type of queuing network. The reason the AAS was developed was due to the fact that the arrival rates were similar to that of an MMPP queue. As such, we believe the methods described herein would be extensible to MMPP queuing networks as well as batch arrival networks. Simple Markovian arrival networks, however, would most likely need to develop and test alternative solvers.

8 Summary

Developing an adaptive solution for a complex system can difficult. Deciding about how to implement a process for predicting the behavior of the system is very important.

The study has been motivated in part by the requests from some of our industry partners to develop a low cost alternative to extensive and complex simulations developed as a part of conceptual analysis of modern border crossings [23]. While the design of airport immigration systems if largely known and well studied, the design of exit procedures for passengers leaving the United States and many other countries through airports is still an open issue. We believe that our techniques offer a good starting point for the early, design - time analysis of architectural alternatives. The organization of pedestrian, vehicle and other type of BIMS also remains open to innovation. For this reason, the lessons learned from the application and experimentation of rather mature performance analysis tools and techniques are relevant and are likely to be consequential for the domain practice.

Consider if we had tried to use the simulator instead of the AAS to predict the system's behavior. At each instance of the feedback loop, we may need to test 100 different options. One simulation per test wouldn't give us any significant confidence that we are correctly predicting the outcome of the system following the given choice. Therefore we would need to repeat the simulation at least 15-20 times. For any given instance of the feedback loop this may take thirty minutes for the simulator to determine the best course of action using a mid-to-high-end computer like the one utilized for our tests. This is infeasible without a large investment of resources.

The advanced analytical solver however not only runs over two hundred times fast than the simulator, it only needs a single run to reach its optimal predictive behavior. Thus it predicts the best course of action for the same feedback loop in less than a second.

Not only have we shown that efficient solvers may be developed to accurately predict the behavior of complex systems, we have tested our solver against a simulation of one such real-world system. We have verified the robustness of our solution by introducing multiple 'unexpected events' into the simulator.

There are many applications of adaptive systems [12], many of which are too complex for simple solvers. Given the work shown herein, we are confident similar techniques may be employed to increase the efficiency, or even feasibility, of many real-world systems in use today.

9 Future Work

We would like to be able to extend this work onto a different complex queuing network in order to see how extensible our optimization techniques are. There are potentially thousands of different outlets for this work.

One thing we would like to consider is the possibility of testing different analytical solvers. Initial brainstorming rejected the idea of an optimal solver using Bayesian analysis as being too computationally complex, but we haven't ruled out the idea all together. Furthermore, there may be dozens of potential suboptimal solvers such as AAS which may obtain even better performance.

While the adaptive system is able to predict and expect some future events, it was unable to react to early arrivals without a greater than zero EWT due to their being entirely unexpected. This problem may be solved by implementing a learning element to the analytical solver which develops the initial schedule using theory from pattern matching to learn to identify potential threats to performance and take action beforehand. Adding an evolutionary element to the adaptive system could potentially create a substantial increase in performance.

Possibly the most obvious extension of this work, however, would be to apply it to a real-world system. In doing so we could better validate the results and therefore further our capabilities of dealing with complex queuing models.

10 References

[1] Sacanamboy, Mayra. “Risk Analysis in Biometric-Based Border Inspection System", MS Thesis, West Virginia University, 2009.

[2] Mueller, Eric; Chatterji, Gano. “Analysis of Aircraft Arrival and Departure Delay Characteristics”, AIAA’s Aircraft Technology, Integration, and Operations (ATIO), 2002 Technical 1-3, , Los Angeles, California, October 2002.

[3] Phillips P. J., Scruggs W. T., O’Toole A. J., Flynn P. J., Bowyer K. W., Schott C. L., and Sharpe M., “FRVT 2006 and ICE 2006 Large-Scale Results”, NIST, March 2007.

[4] Wilson C., Hicklin R. A., Korves H., Ulery B., Zoepfl M., Bone M., Grother P., Micheals R., Otto S., and Watson C. “Fingerprint Vendor Technology Evaluation 2003 Analysis Report”, NIST, 2003.

[5] Grother P., Quinn G., and Philips P. “Report on the Evaluation of 2D Still-Image Face Recognition Algorithms”, NIST, 2010.

[6] Flight Arrivals: http://www.flightarrivals.com

[7] Allen, A. O. “Queueing Models of Computer Systems,” IEEE Computer, Vol. 13, No. 4, April, 1980, pp. 13-24.

[8] "Solution of some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges", Elektrotkeknikeren, vol 13, 1917.

[9] Robbins, T.R.; Medeiros, D. J.; Harrison, T.P., "Does the Erlang C model fit in real call centers?," *Simulation Conference (WSC), Proceedings of the 2010 Winter* , vol., no., pp.2853,2864, 5-8 Dec. 2010 doi: 10.1109/WSC.2010.5678980

[10] ACM Computing Surveys, VoL 10, No. 3, Sept. 1978.

[11] J. Spragins, "Approximate Techniques for Modeling the Performance of Complex Systems," Computer Languages, Vol. 4, No. 2, 1979, pp. 99-129.

[12] B. H. C. Cheng, H. Giese, P. Inverardi, J. Magee, and R. de Lemos. “Software engineering for self-adaptive systems: A research road map”, Dagstuhl-seminar on software engineering for self-adaptive systems. 2008.

[13] Mirchandani, P.B.; Zou, N., "Queuing Models for Analysis of Traffic Adaptive Signal Control," *Intelligent Transportation Systems, IEEE Transactions on* , vol.8, no.1, pp.50,59, March 2007 doi: 10.1109/TITS.2006.888619

[14] Newell, “The rolling horizon scheme of traffic signal control,” Transp. Res. Part A, vol. 32, no. 1, pp. 39–43, Jan. 1998.

[15] Faundez-Zanuy, M., "Biometric security technology," *Aerospace and Electronic Systems Magazine, IEEE* , vol.21, no.6, pp.15,26, June 2006 doi: 10.1109/MAES.2006.1662038

[16] Kendall, D. G. (1953). "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain". *The Annals of Mathematical Statistics* **24** (3): 338. doi:10.1214/aoms/1177728975.JSTOR 2236285

[17] L. Bruer, D. Baum. (2005). Markovian Arrival Processes. In *An Introduction to Queueing Theory and Matrix-Analytic Methods* (pp. 185-196). Springer Netherlands.

[18] Yu, J., Tian, G. "Efficient Algorithms for Generating Truncated Multivariate Normal Distributions," Acta Mathematicae Applicatae Sinica, English Series, Volume 27, Issue 4, October, 2011, pp 601-612

[19] Greg Franks, Tariq Omari, C. Murray Woodside, Olivia Das, Salem Derisavi, "Enhanced Modeling and Solution of Layered Queueing Networks", IEEE Trans. Software Eng. 35(2): 148-161 (2009).

[20] Cortellessa V., Goseva-Popstojanova K., Appukkutty K., Guedem A., Hassan A., Elnaggar R., Abdelmoez W., Ammar H., "Model-based Performance Risk Analysis", IEEE Transactions on Software Engineering, Vol. 31, No. 1, January 2005, pp. 3-20.

[21] Cukic B., Sacanamboy M., "Combined Performance and Risk Analysis for Border Management Applications", 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN) (2010).

[22] Drummond C., and Holte R., "Cost curves: An improved method for visualizing classifier performance", Machine Learning, Vol. 65(1), pp. 95- 130,2006.

[23] Edmunds T., Sholl P., Yao Y., Gansemer J., Cantwell E., Prosnitz D., Rosenberg P., and Norton G., "Simulation Analysis of Inspections of International Travelers at Los Angeles International Airport for USVISIT", Technical Report, Lawrence Livermore National Laboratory, 2004.

[24] PKI digital signatures for machine readable travel documents, version 4, technical report, ICAO, 2003.

[25] ICAO requirements for epassports interoperability, annex k, version 1, Technical Report, ICAO, 2004.

[26] PKI for machine readable travel documents offering ICC read-only access, version 1. Technical Report, ICAO,2004

[27] Machine Readable Travel Documents (MRTDs): History, Interoperability, and Implementation, ICAO, TAG-MRTD/17-WP/16,2007.