

Harrisburg University of Science and Technology

## Digital Commons at Harrisburg University

---

Other Student Works

Computer and Information Sciences, Graduate  
(CSMS)

---

Spring 4-15-2020

### A Study over Registration Server System Simulation

Maolin Hang

Follow this and additional works at: [https://digitalcommons.harrisburgu.edu/csms\\_student-coursework](https://digitalcommons.harrisburgu.edu/csms_student-coursework)



Part of the [Computer Sciences Commons](#)

---

#### Recommended Citation

Hang, M. (2020). *A Study over Registration Server System Simulation*. Retrieved from [https://digitalcommons.harrisburgu.edu/csms\\_student-coursework/6](https://digitalcommons.harrisburgu.edu/csms_student-coursework/6)

This Dissertation is brought to you for free and open access by the Computer and Information Sciences, Graduate (CSMS) at Digital Commons at Harrisburg University. It has been accepted for inclusion in Other Student Works by an authorized administrator of Digital Commons at Harrisburg University. For more information, please contact [library@harrisburgu.edu](mailto:library@harrisburgu.edu).

**A Study over Registration Server System Simulation**

- Continuous study with the Redesigned Architecture and Performance Comparison

Maolin Hang  
Harrisburg University of Science and Technology  
mhang@my.harrisburgu.edu

Thesis submitted to the Faculty of the Graduate School of the



*In fulfillment of the requirements for CISC699, Spring 2020 of the Computer Science and Information Program*

*Harrisburg University*

Supervised by: Abrar Qureshi, Ph.D.  
[Spring 2020]

## Abstract

This paper is a continuous study of the registration server system using a previous created real-time simulation application for my working product- T-Mobile Digits' registration server system - an Enterprise-level solution ensembles Skype for Business, but with a sizable testing user pool.

As a standard system design normally includes the hardware infrastructure, computational logics and its own assigned rules/configures, and as all the complex system, a well-set server structure is the kernel for no matter testing or commercial purpose. The challenges are real and crucial for both business success besides the concerns of access capability and security. It will begin with the discussion of the server-side architecture and the current functional workflows. However, the problematic project is facing stalling issues of the registration system whenever the automation tests deploys, or the pressure tests are happening.

The project norms are based on my previous study, current study after architecture refactor and enterprise server function reporting tool: Splunk.

I will create a new hypothesis of the mathematical model/formula towards the new architecture and will retrieve the most of simulation skeleton formed from last semester by introducing new variables and new model for the performance comparisons.

This project will finalize the study from the last semester and evaluate the server performance under the new architecture. Also, I will try to explore and compare the performances before and after the structure level refactors in the server architecture design, which is in achieving to provide comparison to the system architects or other stakeholders and help them to explore the possible improvements of the current registration server system.

The ultimate goal of the study remains the same: I am seeking opportunities to analyze over current problematic flows and achieving making betterments to the product and I expect to make theoretical suggestions to better for the current workflow and logic structure of the current registration server system so that the server would be more durable for automation tests and malicious attacks.

**Keyword -- Simulation; Server Simulation; Registration Server System; System Design and Architecture; Architecture Refactor; Performance Comparison; Data Analysis.**

## 1. Introduction

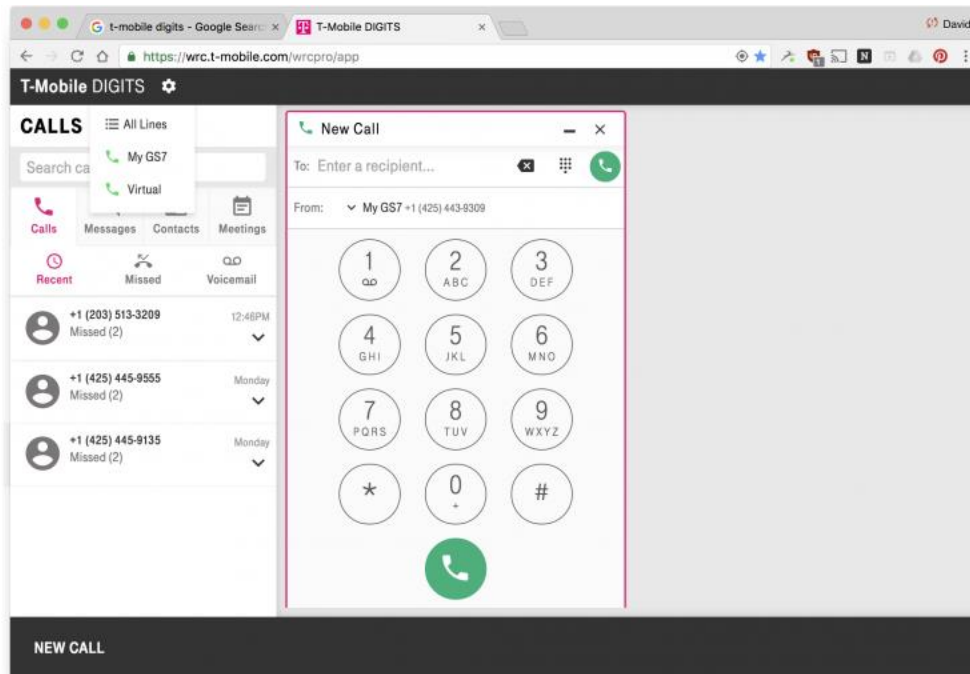
As the third U.S. largest wireless carrier and provider, T-Mobile released its DIGITS product on May 25, 2017 as an additional service to current customers with valid consumer level voice lines. The product helps users to get access to their phone lines while the devices are absent or other special circumstances.



I've been working as a software engineer for the last 3 years in T-Mobile DIGITS team, in where we are seeking for the ultimate business solutions for individual, small business and enterprise level customers.

As a matter of fact, our team is fundamentally supported by the telecommunication industry and T-Mobile's infrastructures. In addition to that, the browser and desktop client team I am in is nested to a gigantic network with multiple gateways.

Backend teams are held accountable for providing the relentless telecommunication services, and the client ends for the users are extending the backend services from the cloud, and the users will be able to reach their contacts, messages, calls, voicemails and other information both from the via their cell phones or the client applications with the web or cell data connection.



We initiated DIGITS enterprise (E-Digits) solution in 2017 soon after the T-Mobile DIGITS' release. The enterprise project is much more scalable in the size and much more complicated in functionalities and backends' services. The project is now with a gigantic skeleton with many kinds of backend servers, which provides all kinds services to the user client. During the last three years, I witnessed the server upgrading from one single small project becoming a multiple-servers system.

The registration service should be the initial and the most important service for the all the feasible services lined up afterwards in this project.

I believe no user would like to use any product having hard time to sign-in, which is an utterly disappointing user experience. However, we are still facing countless registration failures and stalling during the tests and real-life practices.

This behavior not only prevents the users from using the application, but also blocks the developers to function smoothly on all the nodes.

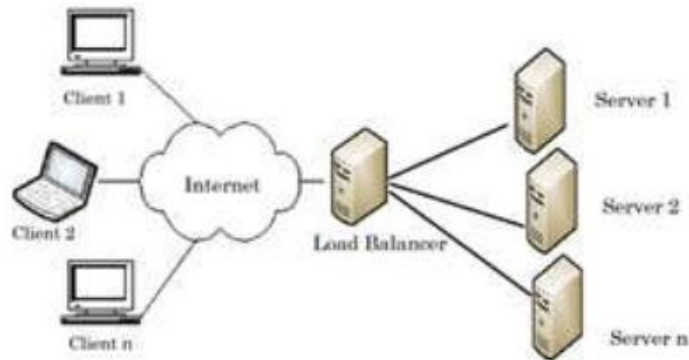
I'm expecting to use the built simulation of registration server system from the other course to run multiple variables to study, analyze and understand the registration system better, thus, to make meaningful recommendations and updates to the registration system.

Therefore, I decide to choose to study the registration server system for my GRAD695 class.

## 2. Backgrounds

### 2.1 General Server Information

The registration server system is a multiple-servers system with a load balance with the architecture as shown below:



*Sample of Multiple Servers with a Load Balancer*

The current system includes 1 load balancer and 5 processing servers, which are all connected by the gateway service like WebRTC gateways. The system is designed to deal with the maximum capacity of 30,000 users with 5 instances of signed-in devices each.

A normal healthy registration session will be kept alive for two hours, and upon each session expiration, the client will either resend another registration request to renew the active user instance of registration on the same server or log out the user if any node is not responding. Exception applies when the system got stuck, in idle or other error. The user session will be still retrievable exceeds 2 hours in rare chances.

When the user tried to register to our application, the request will be sent to the load balancer before sorting and being redirected to the assigned processing servers. The load balancer will load the user information and analyze if the user is a primary user or a tenant-free user and deciding to send the user's request to the pre-designated servers or not.

If the first processing server accepting the request, the registration process will start being piled up at queue on the certain server, other wisely, the request will be sent back to load balancer and seek another viable servers until the request being processed or finalized with response like: bad request or server not available.

The load balancer collects responses from the processing servers to compute the remaining capacity and to log the location of certain user the registration information from signed-in server while sending the requests.

Normally, in a WebRTC system, after the registration request got successful response, the channel subscription or WebSocket notification channel will be established for the registered users between the clients and the registration server end.

The client will keep pinging POST request to the system bridge to make sure the notification channels is open and in health condition while making sure the registered user is successfully registered to the registration system.

And we don't have to worry about send the responses all the way back client and will normally form a success response {status: "200 ok"}.

Each of the processing server has the capacity of dealing and holding about 100,000 access nodes/requests individually, which means in the flawless condition, the system should be able to provide services to 41,666 users at the same time.

This is calculated by 5 instances of signed-in, the nodes will be doubled if the session get renewed, and with the 6<sup>th</sup> instance signed-in of retiring the first idle session or directly turns the request down since the user already has 5 active sessions.

## 2.2 Current Workflow

The current problematic request flow is showing as following:

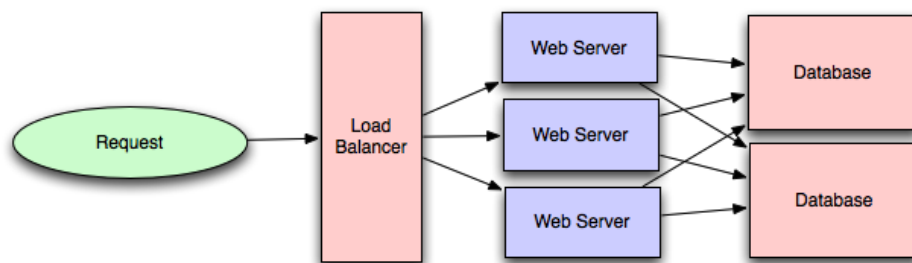


Figure 1. DFD Level 0 Chart for Registration Server System

In the database level, all the users are identified by the primary key namely as TUID, and most of the enterprise-level users should be bound with a tenant domain, which makes them the primary user of the system.

Ideally, the primary users' requests will only be sent to the designated fast-queued or prioritized servers, the random/tenant-free/guest users' requests will be sent to the random available servers. However, since of the pre-assignment of the servers, the primary used may end up waiting longer than the tenant-free users since the piled queue time is longer than the random servers.

Theoretically, each HTTP request can take from 20ms up to 7000ms to be transmitted to the load-balance server, and the requests will be queued in the load balancer before assigned to the processing servers.

Assumingly, the transmission inside the registration system is about 40ms for each request, and the registration processing time without queues could be within 1 second.

In the worst scenario, the user is experiencing 7000ms (7 seconds) each way of sending the request and receiving the request response and was turned down four times since all the previous four servers were with full capacity ( $40\text{ms} * 10 + 1000\text{ms} * 5$ ) = 5400ms, which is 5.4 seconds.

Excluding the extreme conditions like complete internet outage, the server should be able to make response to the user within 20 seconds; however, the traffic could be piled up if 30,000 users are trying to renew their session spontaneously, which is always happening, when the quality analysts are making automation tests or if the malicious users are trying to overflow the registration server.

Besides all the things mentioned above, the automation tests are normally performed every other day on the weekday afternoon starting 1PM PST. And so far, we have encountered uncountable occasions for all the developers hopelessly waiting the whole afternoon and wished for the registration system to be working properly, which means all the users are trying to register 5 instances and tried to do additional registrations all the time to test the durability of the servers.

On the normal days, we have about 97% the initial registration successful rate; however, we only have around 72% passing rate of renew registration if high volume of requests were sent simultaneously.

In the fine weather, the registration process will take less than 10 seconds to redirect the user to the client application, since the above-mentioned reasons, the client still sets up the logics for tolerating 30 seconds as the maximum waiting time for the registration.

### **2.3 Proposed Workflow and Architecture**

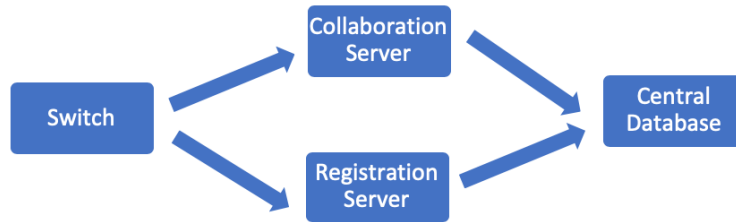
From the two model study result of Fall 2019, the Model 1 suggested that the non-primary servers are always in critical condition to process requests for the non-primary (non-enterprise) users, and there were no major differences if we terminate the servers designations since the Primary users are always in prior queues, which means that the Model 1 in the below content is no longer a suggestion model for checking the registration server health condition, and the performance issue may due to an upper level structure and request dispense logic.

Model 2 did reflect the actual result of the Processing Queue Time using a meaningful method; however, the result is not really ideal since according to the current workflow, most requests are bouncing between registration servers to the load balancer internally which delays the final response to the users. At the same time, the registration services will consume several external requests, which also adds the latency for the server response queue time.

Therefore, according to those results, I suggested the team to form a developing/testing pool with 300 user accounts and started a new virtual registration server system using the new skeleton of the registration server architecture design which split the functions on the load



balance server by adding switch nodes and a database over the old structure before the load balance and after the performance of the registration servers.



Graph 1. New Proposed Request Model

The new registration server system architecture is showed as following:

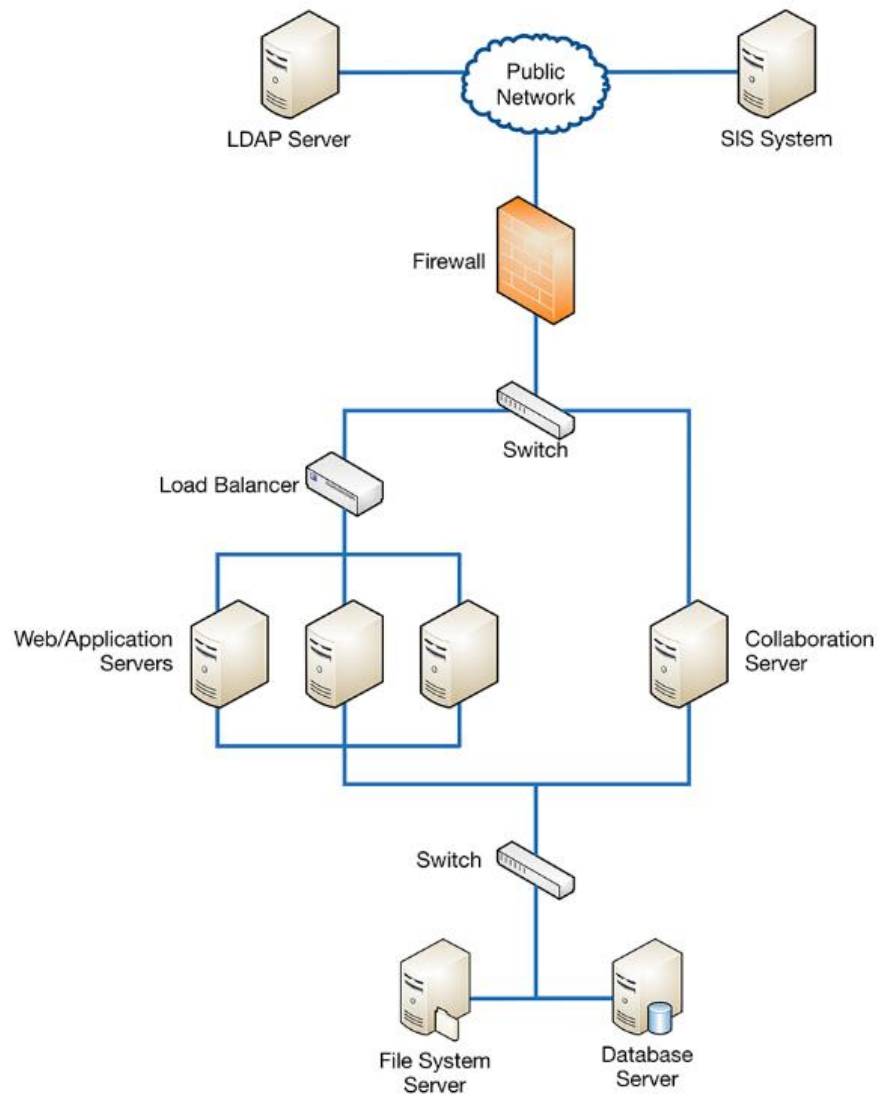


Figure 2. Experimental Registration Server DFD Level 0 chart

By using this new added switch server, the requests are quickly sorted by a direct request to collaboration server for a self-check and accessing to the file system interface which accesses to the Central Database to validate the user's registration information and also validate requests to achieve a quick response to the user.

In this new architecture, a new flag is added as a part of the user registration information object, which will increase the count of the registered devices (registered or in-progress) to enable the multi-threads. This flag will also work with client session and client friendly name to remove the duplication of the requests, which will decrease the actual requests going to the Load Balancer, who dispenses the tasks to the registration servers.

Since the Load Balancer server is no longer filtering the requests for duplication requests and only deals the bouncing requests to find the server availabilities, which simplifies and promotes its performance of internal requests between the Load Balancer and the Registration Servers as all the incoming requests are validated and there are no longer need for filtering the requests which exceed the limit per accounts.

The process of dealing each request is expected to be consumed more internally instead of externally.

Theoretically, any new request will need a pair of external processing for incoming request and outgoing response while accessing the Switch server. With less frequent external requests exchange between the client to the registration server, the user would expect to know if they are able to register faster even with a much slower tethering network than the ideal network speed.

Afterwards, the logic split at the switch server:

If the requests are **invalid**, the user would almost immediately know the result with no stalling by simply query through the Collaboration Server to the Central Database, which is about 1 second or 2 in the worst scenario. (First Logic Split End)

If the requests are **valid**, then, it will be forwarded to the Load Balancer, which will lately dispense the requests to the available server. Similar to the process of the current existing workflow, the requests will be queued in the Load Balancer and bounce between the Load Balancer and the Registration Servers until they are processed. Since all the requests are handled internally in a much simpler version and no longer required to response to the user directly, the processing time will be shortened as well.

Finally, the registration servers are done with the request, and final result of registration or timeout cut response will be updated to the switch and updated to the Central Database and the users. (Second Logic Split End)

The estimated processing time for the Second Logic Split will at least including two internal processing, two external processing, two switch processing and two data query processing with or without entering the Load Balancer fore registration.

After entering the Load Balancer, the requests are going through the maximum failure allowance for failure retries or exit with preliminary timeout.

### 3. Models

There are three models formed and used for simulating the registration server system, however, only Model 2 and Model 3 will be used for the current study.

The first model is for estimating whether the processing server has met its ultimate capacity (Model 1), this model will perform a logic computing and produce a Boolean value for responding the load balance whether the registration request can be processed eventually in the server.

The second model is an mathematical model for calculating the queued time to the load balance accessing to the processing servers to tell whether it is still needs to be waiting that long to let the client know the expected waiting time instead of timed-out the requests too early (Model 2).

The third model

#### 3.1 Model 1: Processing Server Capacity and Health Condition Test (Deprecated Model)

*This model has been deprecated since the previous study result shows that the random servers for non-primary users are always in critical health condition as a result of the primary users are always lifted to a prior queue.*

*Therefore, cancelling the server designation will not improve the current registration server performance and the server designation actually guarantee the stable services provide to the primary users which is the major paid users of the whole user pool.*

*The part of study is still presented in this article for the trace of previous study and explains the necessary of introducing the new system architecture.*

As mentioned in the Backgrounds, all the users can sign-in up to 5 devices, and several designated servers are prioritizing for the primary users, while the rest have no preferences of the user types.

The attention-grabbing part of the Model 1 is the pre-assigned partitioned percentage for the primary users when the servers is run as the designated servers for the Primary Users.

The Primary Users can take up the whole designated servers if needed or requested. In this condition, the Tenant-Free Users shouldn't be registered to the pre-designated server at all. But normally, the Tenant-Free Users will occupy only the nodes if the Pre-assigned Partitioned Percentage for Tenant Free Users was not fully met. (100% minus Pre-assigned Partitioned Percentage for Primary Users).

In the extreme cases, if a user signed-in five devices, the renewed requests are all processing at the same time, and the same account is trying on the 6th device, which means the user are taking up to

$$(5*2) + 1 = 11 \text{ accessing nodes}$$

Therefore, a formula can be formed as:  $(2*r+1)$

In the Model 1, I assume that if one user has been registered on one certain processing server, all his/her signed-in instances should be registered on the same registration server.

Therefore, the actual maximum capacity for tenant-free users are restricted by smaller value between the pre-assigned percentages of the tenant-free users or the actual leftovers from the primary users.

The formula could be described as:

$$\text{MIN} ((\text{MAX} - \text{PRU} * (2* r+1)), (\text{MAX}*(1-\text{pp})))$$

There are 2 backup servers running as non-designated servers for all the users, furthermore, I am also suspecting the righteousness of having the pre-assigned servers at all.

The formula is easily formed as:  $(\{\# \text{current user numbers}\} + 1) * (2*r+1)$

The formulas are comparing if the nodes are available

The above described model structure is shown as below:

**Model 1:**

**PRU:** The number of Registered Primary Users

**TFU:** The number of Tenant Free Users

**AU:** The number of All the Users

**r:** The max failure number allowed for each request

**PP:** Pre-assigned partitioned Percentage for Primary User

**Max:** The maximum capacity

<p><i>Designated Servers:</i></p> <p>Registered Primary User: <math>(\text{PRU} + \text{TFU} + 1) * (2*r+1) &lt; \text{MAX}</math></p> <p>Tenant-Free Users: <math>(\text{TFU} + 1) * (2* r+1) &lt; \text{MIN} ((\text{MAX} - \text{PRU} * (2* r+1)), (\text{MAX}*(1-\text{pp})))</math></p>
<p><i>Non-Designated Servers :</i></p> <p><math>(\text{AU} + 1) * (2*r+1) &lt; \text{MAX}</math></p>

**3.2 Model 2: Estimated Queued Time for the Request**

Also referred in the section II, the client timed-out period will be a regulated 30 seconds. The client times out the initial request and immediately start another possibly-fail-again retry registration request without knowing the promising anticipating time for retry.

The noteworthy part of the Model 2 is the variable FTN: estimated Failure Trials on Approaching Processing Servers. If the architecture designed better, one user registration request should go for more than two processing servers. And in the extreme condition, when there were no piled-up queues, but somehow all the first trials on 5 processing servers failed and EQT is much smaller than 30 seconds. Should I allow the load balance to recheck the freed-up servers before handing out a failure response?

The Model 2 is shown as below:

**Model 2:**

**ProcTime:** estimated Processing Time of each Request

**ATT:** estimated Alien Transmission Time of each Request

**ITT:** estimated Internal Transmission Time of each Request

**request:** the number of the Requests

**FTN:** estimated Failure Trial on Approaching Processing Servers

**EQT:** estimated Queued Time for the Request

$$EQT = [(FTN + 1) * (2 * ITT + ProcTime) + 2 * ATT] * request$$

### 3.3 Model 3: Improved Estimated Queued Time for the Request using the new Architecture

All the background information for Model 3 is similar to Model 2. I'm trying to generate an Estimated Queue Time for the Request Responding Process while introducing of the switch explore to the Central base in the new architecture as showed in Figure 2 in the section II.

Both Model 2 and Model 3 is supposed to be less than 30 seconds according to the non-functional requirements.

The Model 3 is shown as below:

**Model 3:**

**maxUserNumber:** A user manual input figure or default as 300 users

**ProcTime:** estimated Processing Time of each Request

**SPT:** estimated Processing Time of each Request inside Switch Server

**DQT:** estimated Processing Time of each Query to the Central Database

**ATT:** estimated Alien Transmission Time of each Request

**ITT:** estimated Internal Transmission Time of each Request

**LBTT:** estimated Internal Transmission Time of each Request inside Load Balancer

**request:** the number of the Requests

**FTN:** estimated Failure Trial on Approaching Processing Servers

**EQT:** estimated Queued Time for the Request

$$EQT = [(FTN + 1) * (2 * LBTT + 4 * ITT) + ProcTime + SPT + DQT + ATT] * request + (maxUserNumber - 1) * (SPT + DQT)$$

## 4. Design Decision

According to the models and system's exclusivity described above, in order to stimulate a registration server system. I will need to implement the simulation using my own interface.

### 4.1 Programming language

The software of the implemented simulation is going to be based on a node project using JavaScript and React Framework to form a web application.

React Framework bundles with a lot of chart and graph illustration libraries, when the variable changes, the data change could be easily illustrated and updated in the real time.

The web application will be all platform friendly, which means anyone can access the User interface without concerning if they are running on the Windows, Linux, MACOS and all other React friendly portable devices.

As the failure result of previous study, I find out that using React/Chart and other graphic libraries are not currently supported in the React/TypeScript project since the strict data type check. Therefore, I browsed more online libraries for Custom Data Illustration and choose Google Graph API for this semester, as it is free for all the non-commercial users and actually dose greater illustration for the data to show my simulation result as a robust tool.

### 4.2 Constants and Options

Most constant variables are preset from the Splunk running result of the server performance study:

#### Model 1:

1. **MAX**: the maximum capacity, since the limitation for the infrastructure.

#### Model 2:

1. **ProcTime**: the processing time and the transmission time in the worst scenarios will be taking as constants from Splunk Result
2. **ATT**: estimated Alien Transmission Time of each Request from Splunk Result
3. **ITT**: estimated Internal Transmission Time of each Request from Splunk Result

#### Model 3:

1. **ProcTime**: the processing time and the transmission time in the worst scenarios will be taking as constants from Splunk Result
2. **ATT**: estimated Alien Transmission Time of each Request from Splunk Result
3. **ITT**: estimated Internal Transmission Time of each Request from Splunk Result
4. **LBTT**: estimated Internal Transmission Time of each Request inside Load Balancer
5. **SPT**: estimated Processing Time of each Request inside Switch Server
6. **DQT**: estimated Processing Time of each Query to the Central Database

Here are the listed of options in both models could be edited by the users:

**Model 1:**

- **PRU:** The number of Registered Primary Users should be able to switch from 1 to 30,000;
- **TFU:** From 29,999 to 0 as a reverse from the primary users, may all the enterprise users in the system are primary users;
- **AU:** The number of All the Users on un-designated servers can be any number from 0 to 30,000;
- **r:** currently the maximum registered devices is 5, there were a point that the number was 20 since people always forgetting to logout and session retiring mechanism was not implemented, I would suggest the number at least from 2 devices and above.
- **PP:** Pre-assigned partitioned Percentage for Primary User could be anywhere from 0% to 100%. In the extreme cases, all the pre-designated servers are only for primary users.

**Model 2:**

- **request:** can be pretty random from user behaviors
- **FTN:** estimated Failure Trial on Approaching Processing Servers, the user may exhaust all the servers or in loop-detection mode before hitting any carriable servers.

**Model 3:**

- **request:** can be pretty random from user behaviors
- **FTN:** estimated Failure Trial on Approaching Processing Servers, the user may exhaust all the servers or in loop-detection mode before hitting any carriable servers.
- **maxUserNumber:** mannully input by the maximum user number for the server, with a default value 300, since the current testing user pool is 300.

### 4.3 User Interface and functions

The participated simulation User Interface (or UI) will be displayed in a web page as in a web application, including an interactive form for user to type down some randomized input. Then I will show dedicated illustrations for the servers' capacity and processing animation for request processing.





# Registration Server Simulation

Maolin Hang, #234374

## Test Result

Primary User Percentage: 64 %  
Primary Server Percentage: 8 / 20

### Estimated queuing Time in Worst Scenario in (mm:ss):

Model2 - Old Architecture: 797:30   
Model3 - New Architecture: 0:20 

Server health check (Deprecated Model 1): Primary Servers: ok, Tenant-Free Servers: fail

### Chart

Total User Number: (optional, but default as 300)   
Primary User Number:   
Total Server Number:   
Primary Server Number:   
Primary User Percentage for Primary Server:  %  
Request Limit per User:

**Please click Download to save all the previous test results**

*Figure 3. Project Interface Illustration*

There will be a simple analytical description on the top of the application for browsing Model 1, Model 2 and Model 3 simulations.

On each of the stimulation page, I would prefer having the user input form on the top of the page and the simulation to be shown on the same web page; however, if the illustration goes too out of the scale, I will have to open a new window for the graphic illustration. To be more specifically:

On Model 1 Simulation, I will use text to show the designated servers and regular servers when holding two types of the users and the health condition of the server.

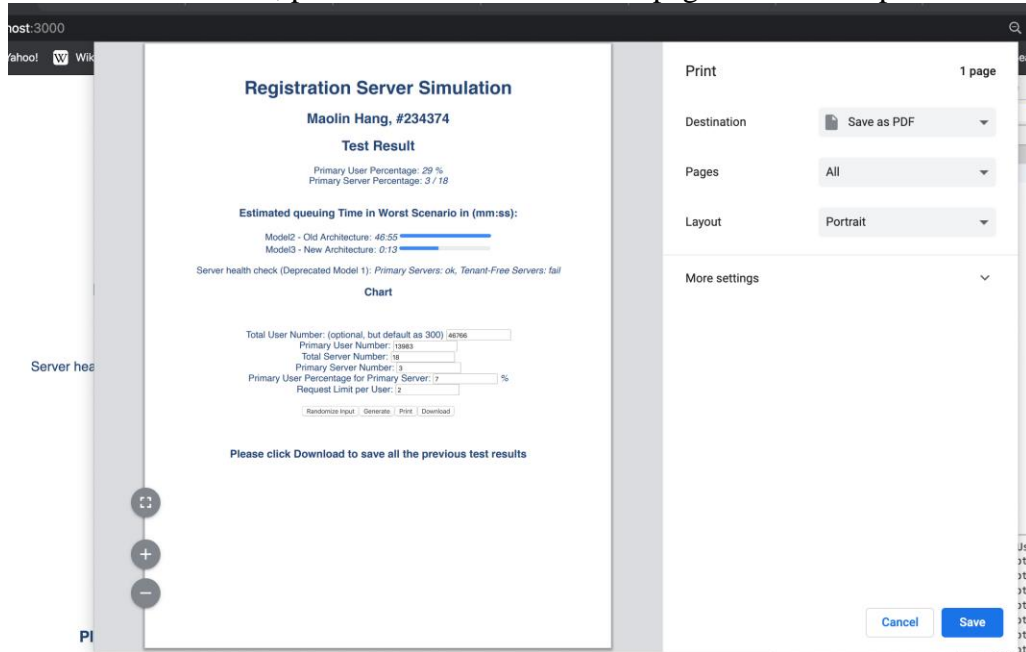
On Model 2 Simulation, I will use an animated progress bar for the estimated Queued Time for the Request.

On Model 3 Simulation, I will also use an animated progress calculation for the new estimation of the Request Queue Time using the new Architecture.

Both Progress rendering will be using Material-UI.

As an additional improvement to my project from last semester, this simulation now supports both Print and Download functions for further data analysis.

When the user clicks, print the information on the page could be output as below:



When the user clicks Download, all the previous auto-generated or manually inputted tests will be print out in a txt format file documented in JSON format showed as following:



All the tested figures from all models will be logged on the bottom of the application page, so that the user can click buttons to revive the tested stimulations and /or save the

stimulation results to local file folders or send those results to the preferred email addresses with proper internet connection.

## 5. Conclusion

Last semester, I studied the registration server system using a real-time simulation I built previously.

From the final result of the last project, I learned that the registration server system is complicated and challenging, since the current access nodes of all the server are very limited and always causing registration stalling experience for non-primary users because of the primary users have higher queue order in all the queue lists of the system, which suggests that we need more processing servers at the end.

Therefore, no matter if the company eliminate the logic differences between the Primary Designated Servers and the Random Servers, the problem will remain.

The partition of different kinds of servers actually some-what guarantee the services for all the paid enterprise-users besides its economic and publicity meanings.

Therefore, I choose a different route to reduce the requests load directly comes to the Load Balancer- Registration Server component by introducing a Switch server for filtering and querying shallowing for the user registration information.

The new architecture is tested in a user pool of 300 in a new virtual registration server system with the same rate of access nodes and I can see a clear improvement of the process time since the benefits of splitting the functions of the older Load Balancer and enabling the multi-threads.

I've already seen the betterments and improvements during the practice, with the Real-time simulation estimation

In this semester, I am still allowing both manual input and randomized tests case in the simulation, while providing comparison illustration of the system performances in two different architecture.

The simulation results are all with the extreme conditions or in the worst scenarios, so that the architectures and all related stakeholders would be able to know the superior advantages of the new architecture.

Meanwhile, I fully understand that introducing the Switch Server and the Collaboration Server will impose an economic concern which may be very costly to the enterprise, which changed my original proposal of *“a more reasonable partitioned percentage should be assigned or a better load balance structure to be adopted or a simple favor of just buying more processing servers”* to where to use the new servers in the most economic ways and find new purposes to the old servers.

Overall The study of the registration server system targets to improve ultimate user experiences as before.

Since I'm using a data size much smaller than the enterprise-level data size. Therefore, I would face much less complications or aspects of the enterprise-level problem and my limit study will be theoretical than practical. However, the limitation of my study actually triggers more enthusiasms for me to study the registration server system.

I would like to bring more possibilities of the new architecture and logics to this registration server system, which I will explore in the further study in the near future.

## Annotated Bibliography

WM Newman, MG Lamming, M Lamming. (2003) Chapter 1 Interactive system design, *Interactive system design*. Retrieved from:  
<http://web4.cs.ucl.ac.uk/ucllc/people/w.newman/ch1.doc>

The author, William M Newman was a Visiting Professor in UCLIC, closely involved in both research and teaching, 2004-2009. In particular, he developed and taught a module, jointly with John Dowell, on "Perspectives on Design", supervised various MSc projects and also contributed to two EPSRC projects ("User Centered Interactive Search" and "Making Sense of Information").

This article is relevant to my paper because it showed the traditional interaction system design that is practiced everyday now.

Michel K. Bowman-Amuah. (1999) System, method and article of manufacture for cross-location registration in a communication system architecture.  
*US6707812/ US Patent 6,081,518, 2000*. Retrieved from:  
<https://patents.google.com/patent/US6081518A/en>

Michel Bowman is a seasoned business innovation pioneer and entrepreneur with a diverse background in Technology development, telecommunications, management consulting and financial Services. Over his 30-year career, Michel has created, incubated and birthed five successful businesses, raising capital for each venture and eventually taking two of them public. He also created highly profitable and groundbreaking Technology Management Consulting practices for Accenture and prior to that Renaissance Worldwide as well as AT&T, Time Warner, and MCI-WorldCom. As a creative and prolific inventor, Michel Bowman-Amuah currently holds over 100 worldwide patents with several dozen pending. Mr. Bowman has been in the forefront of technology and standards development, being the co-founder of the standards organization that birthed Internet Protocol Data Records (IPDR), a standardized format for rendering billing information for Services and Applications delivered over Internet Protocol. Mr. Bowman-Amuah was educated as an Electronics Systems Engineer & Information Systems Architect in the UK and received a Master's in Business Administration and Organization Management the USA.

This patent is relevant to my paper because it shared the same condition of my current project and illustrated the similar design solution the registration system.

Jerry Banks, John S Carson II, Barry L Nelson, David M Nicol. (2010) Chapter 1 Introduction to Simulation, Introduction to Discrete-Event System Simulation.

*University of Illinois, Urbana-Champaign, Pearson Education 2010.* Retrieved from:  
[http://ce.sharif.edu/courses/95-96/2/ce634-1/resources/root/Books/Discrete%20Event%20System%20Simulation%20\(Fifth%20Edition\)%20.pdf](http://ce.sharif.edu/courses/95-96/2/ce634-1/resources/root/Books/Discrete%20Event%20System%20Simulation%20(Fifth%20Edition)%20.pdf)

This text is for Junior & Senior level simulation courses in engineering, business, or computer science and provides a basic treatment of discrete-event simulation, including the proper collection and analysis of data, the use of analytic techniques, verification and validation of models, and designing simulation experiments. It offers an up-to-date treatment of simulation of manufacturing and material handling systems, computer systems, and computer networks.

Chapter 1 of this text is relevant to my paper since it provides the roadmap of how to make the simulation and help me figure out the relevant events in the all the discrete events, thus form the effective models.

Rosario G Garroppo, Stefano Giordano, Stella Spagna, Saverio Niccolini. (2009) Queueing Strategies for Local Overload Control in SIP.

*GLOBECOM 2009 -2009 IEEE Global Telecommunication Conference.* Retrieved from:  
<http://ieeexplore.ieee.org/abstract/document/5425591/authors#authors>

The paper presents a simulation analysis, aimed at evaluating the impact on system performance of different queueing structures, service disciplines, and buffer sizes. Simulation results clearly show that the proposed queueing discipline produces good system performance with a low complexity increase. Finally, the simulation results point out the weakness of the {503: service unavailable} message mechanism, which does not introduce significant improvement when combined with the proposed solution.

The leading author Rosario Giuseppe Garroppo received the M.S. (Laurea) degree (cum laude) in telecommunications engineering and the Ph.D. (Dottorato di Ricerca) degree in information engineering (ingegneria dell'informazione) from the University of Pisa, Italy in 1995 and 1999, respectively. He is currently an Assistant Professor with the Dipartimento di Ingegneria dell'Informazione, His expertise is on networking, and his main research activities are focused on experimental measurements and traffic modeling in broadband and wireless networks, MoIP systems, traffic control techniques for multimedia services in wireless networks, network optimization, and green networking.

This article is relevant to my paper is because it provides the general idea of Queueing Strategies for the Local Overload Control, which is the same study content of my Model A in the paper.

Charles Shen, Henning Schulzrinne, Erich Nahum. (2008) Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation.

*IPTComm 2008: Principles, Systems and Applications of IP Communications, Services and Security for Next Generation Networks*. Retrieved from:

[https://link.springer.com/chapter/10.1007/978-3-540-89054-6\\_8](https://link.springer.com/chapter/10.1007/978-3-540-89054-6_8)

The leading author, Dr. Charles Shen is a Research Scientist in Civil Engineering and Engineering Mechanics and the Co-Director of Advanced Construction and Information Technology (ACTION) Laboratory at Columbia University. He is also a Senior Member of IEEE. Prior to assuming his current role, he worked as a Senior Member of Technical Staff at AT&T. Before AT&T, he conducted research at Columbia University's Department of Computer Science, IBM Watson Research Center, Telcordia Technologies (now part of Ericsson), Samsung Advanced Institute of Technology and Singapore's Institute for InfoComm Research (A\*STAR).

Dr. Shen's current research interest involves the application of advanced Information Technologies such as Deep Learning, Blockchain, Internet of Things (IoT) and Big Data in the sustainable urbanization domains. His past experience includes extensive research on mobile computer networks and applications, such as scalability of IP telecommunications networks, architecture, security and privacy of cloud based mobile IoT services.

This article is relevant to my paper because it provides a different aspect and algorithm of dealing the server overload control, which is also a concern of my Model A.

Daiki Min, Yuehwern Yih. (2007) A simulation Study of registration queue disciplines in an outpatient clinic: a two-stage flow model.

*European Journal of Industrial Engineering*, 2009. Retrieved from:

[https://www.researchgate.net/profile/Yuehwern\\_Yih/publication/46513858\\_A\\_simulation\\_study\\_of\\_registration\\_queue\\_disciplines\\_in\\_an\\_outpatient\\_clinic\\_A\\_two-stage\\_patient\\_flow\\_model/links/555ce60508ae6f4dcc8bd107.pdf](https://www.researchgate.net/profile/Yuehwern_Yih/publication/46513858_A_simulation_study_of_registration_queue_disciplines_in_an_outpatient_clinic_A_two-stage_patient_flow_model/links/555ce60508ae6f4dcc8bd107.pdf)

Dr. Yuehwern Yih is a Professor of Industrial Engineering at Purdue University, USA, and the Director of the Smart Systems and Operations Laboratory. Her research focuses on the dynamic scheduling and control of complex production systems, which incorporates dynamic and multiple production requirements and changing system conditions into an online controller.

The applications of her research results include wafer scheduling in semiconductor fabrication facilities, the synchronization of the production line in the elevator industry, warehousing operations in e-business, machine failure diagnosis and prediction, water network security, advanced life support systems for Mars missions and healthcare



systems. She received her PhD in Industrial Engineering from the University of Wisconsin-Madison in 1988. She is a member of INFORMS.

This article is relevant to my paper because it provides me a simulation study plan with controlled variables, and we share the similar two-stage flow model of the registration system.

Sudheer Sirivara, Jeffrey S McVeigh, Robert J Reese, Gianni G Ferrise. (2001) Server-side measurement of client-perceived quality of service.

*US7185084B2/US Patent 7,185,084, 2007.* Retrieved from:

<https://patents.google.com/patent/US7185084B2/en>

Leading inventor; Sudheer is a proven product and engineering leader with more than 18 years' experience across two Fortune 20 companies having incubated & launched both Enterprise and Consumer grade products. Recognized leader within Media industry having built & launched multiple at scale cloud services spanning PaaS and SaaS enabling global events including Super Bowl, Olympics, and Enterprise SaaS products. Empowered & developed leaders across a globally distributed engineering team. Drove partner and product strategy for Azure in media and accelerated growth of media business. Pioneering development of real-time video analytics (AI) for multiple industries and data-led insights driving enhanced discovery and personalization of content. Incubated, built and launched Microsoft Stream, Azure Media Services, Azure Video Indexer, and Azure CDN.

This invention is relevant to my paper is because the content is relates to determining quality of service (QOS) for network delivery of content, which is the guide to improve the user experience.

Satoshi Kamiya. (2008) Load balancer, network system, load balancing method and program.

*US20090245113A1/US Patent App 12/410,830, 2009.* Retrieved from:

<https://patents.google.com/patent/US7185084B2/en>

Leading inventor; Satoshi Kamiya, CISSP is currently employed by NEC corporation as a Senior Engineer works toward Research and Development, application-level network device, network security, DPI (deep packet inspection), high-performance network processing engines and OpenFlow network.

This invention is relevant to my paper is because it provides a load balancer and a load balancing method, if load on a server becomes higher and the server enters a high-load state, processing is transferred from the server to a second server.

Kazuma Yumoto, Eri Kawai, Masihiro Yoshizawa. (2006) Load balancing server and system.

*US8095681B2@/US Patent 8,095,681, 2012*. Retrieved from:  
<https://patents.google.com/patent/US8095681B2/en>

This invent suggests when load balancing (LB) for SIP message communication is performed across multiple SIP servers, the number of messages that must be processed by an LB server is reduced. The SIP LB server includes an LB management means which, if the previous hop source of a received request message is a terminal in the local domain, determines which SIP server to serve the terminal, and a redirect function which notifies the message source terminal in the local domain of the address of the serving SIP server. The SIP LB server further includes an LB management table which, if the previous hop source of a received request message is other than a terminal in the local domain, is searched to find a SIP server serving communication with a destination terminal in the local domain, and a stateless forwarding function which stateless-ly forwards the received request message to the resolved SIP server address.

This invention is relevant to my paper is because it provides a load balancing for SIP message communication across multiple servers, which is the ultimate solution to promote the efficiency of the load balancer.

Roman Chertov & Sonia Fahmy. (2006) Optimistic load balancing in a distributed virtual environment.

*Article No 13, Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video, 2006*. Retrieved from:  
<https://dl.acm.org/citation.cfm?id=1378208>

The authors:

Sonia Fahmy is a professor at the Department of Computer Science at Purdue University, a member of CERIAS, with current research interests lie in the areas of network architectures and protocols, specifically Network Functions Virtualization, Internet measurement and tomography, network simulation and testbeds, network security, and wireless and sensor networks.

Roman Chertov is a seasoned professional that has participated in academic research, military research, and enterprise networking software development. He has also opined as an expert witness in PTAB IPR proceedings.

This article is relevant to my paper is because it supports the optimistic assumption of my model B when the processing norms are fixed and the article also sets an example of load balancing distribution logics for me to follow.

## References

1. Wikipedia, System Design, [Online] Retrieved from:  
[https://en.wikipedia.org/wiki/Systems\\_design](https://en.wikipedia.org/wiki/Systems_design)
2. Vasanthk, System Design CheetSheet, [Online] Retrieved from:  
<https://gist.github.com/vasanthk/485d1c25737e8e72759f>
3. ScienceDirect, Server Architecture, [Online]. Retrieved from:  
<https://www.sciencedirect.com/topics/computer-science/server-architecture>
4. David Pogue, T-mobile Digits frees your phone number from your phone, Jan 27, 2017 [Online]. Retrieved from:  
<https://finance.yahoo.com/news/david-pogue-t-mobile-digits-review-165437873.html>