Analyzing the Impact of Malicious Network Traffic: Complete Implementation

ALEXANDER M. VALENTIN, Kennesaw State University, USA

CCS Concepts: • **Networks → Network simulations**; **Network performance modeling**; *Network performance analysis*; *Network experimentation.*

Additional Key Words and Phrases: Distributed Denial of Service (DDoS), Botnet, Load-Balancing, Distributed Adaptive Routing, Bandwidth Exhaustion, Resource Depletion, Queuing Theory, Network Topology

ACM Reference Format:
Alexander M. Valentin. 2025. Analyzing the Impact of Malicious Network Traffic: Complete Implementation. *ACM Trans. Access. Comput.* 1, 1, Article 1 (September 2025), 16 pages.

1 Implementation Summary

The current implementation of the malicious network traffic simulation includes all core models and algorithms outlined in the initial proposal, and delivers all of the expected functionality that was necessary to capture the system dynamics described in M1. Users can configure the simulation parameters via the provided configuration file and receive detailed, comprehensive data surrounding simulation outcomes. The implementation supports the following key features:

- Legitimate and Malicious Network Request Generation
- Modified Adaptive Routing to Facilitate Network Routing Decisions
- Bandwidth Exhaustion, Depletion of Resources, and Successful Attack Probability Tooling
- Modified Centralized C&C Topology for Issuing Malicious Client Instructions
- FIFO Network Server Request Processing Logic
- Comprehensive Data Collection and Management Tooling
- Network Representation Proportional to Real-World Enterprise Network Surface Area and Processing Capability
- Live Simulation Logging with State Data
- Data Visualization and Exporting Tooling
- Server Shutdown Mechanisms
- Comprehensive Calculation Tooling for Payloads and Simulation Outcomes

All of the implementation details supporting these key features are either supported by literature provided in the initial proposal, or are supported by additional research conducted over the course of development.

While there have been numerous architectural and design changes over the course of development, the scope from the original proposal has largely remained the same. It was imperative that the final simulation behavior accurately captures the system dynamics outlined in the initial proposal. As such, the only minor deviation in scope from the original proposal is with regard to returning results to the client that requested resources from the network. The aim of this malicious network simulation is to analyze how emergent behaviors arise over time as the system undergoes continued load from the attacking network, and how this component interaction impacts the server's ability to process legitimate requests. Since the focus of the simulation is request processing and not payload contents, I found this portion of the system dynamics to be irrelevant to the problem domain.

Author's Contact Information: Alexander M. Valentin, Kennesaw State University, Marietta, Georgia, USA.

2025. ACM 1936-7236/2025/9-ART1
https://doi.org/

As for architectural and design updates, there have been many changes made between the initial proposal and the final implementation. The biggest design change comes from the implementation of a Distributed Adaptive Routing algorithm, while a Distributed Adaptive Routing algorithm was initially described in M1, there are no network topology weights and a zero latency assumption for server communication. Since there are no metrics defining network communication latency, nor the "distance" by which requests will travel, I opted for a Health-Based Adaptive Load Balancing algorithm instead. The load balancing mechanism utilizes server health, an extrapolation of CPU utilization and queue utilization, in order to distribute traffic evenly among network servers. The aim of this simulation is to analyze network performance degradation under load, and it was explicitly stated that the only factors impacting request response time is the time taken to process the request itself, and any additional time the request may have sat in the processing queue and awaiting queue. As a result, the implementation currently provided is adequate for analyzing performance degradation without impacting simulation outcomes.

While the core system components described in M1 are still present in the final implementation, there have been many additions and refinements made to the simulation architecture in order to achieve the desired result. The first major change to the proposed architecture was the addition of the BaseNetworkModel class, a base class that both Botnet and LegitimateTrafficNetwork derive from. At their core, Botnet and LegitimateTrafficNetwork exhibit the same behavior. As a result, I chose to capture this behavior in an associated base class as opposed to writing two functions that will both generate requests to hit the target network. The rest of the key architectural changes are found in the utils and in the root folder, with the addition of ConfigValidator, DataPlotter, GenericEnums, TestEngine, and ProbabilityEngine. These classes were not outlined in the initial UML class diagram in M1. As opposed to supporting the core simulation pipeline, these classes are designed to capture supplemental functionality like testing, data visualization, configuration parameter validation, value storage, and outcome probability calculations. These classes are not imperative to the core dynamics of the simulation and were not captured in the original proposal as a result. Finally, DataCollector has been renamed to DataHandler in the final implementation. Over the course of development, DataCollector become an all encompassing class for data collection, calculation, manipulation, and exporting, as opposed to a single collection device.

2 Execution Documentation

2.1 Run Summary Table

Table 1. Run Summary Table

Run ID	Purpose	Key Parameters	Status	Data
1	Baseline	Default Values	Done	Download

2	Intense Flood	MALICIOUS_TRAFFIC_RATE = 40
		MALICIOUS_CLIENT_COUNT = 150
		REQUEST_TIMEOUT = 8
			Done	Download

3	High Legitimate Traffic	LEGITIMATE_CLIENT_COUNT = 1000
		LEGITIMATE_TRAFFIC_RATE = 4
			Done	Download

4	Mixed Workload	LEGITIMATE_CLIENT_COUNT = 700
		LEGITIMATE_TRAFFIC_RATE = 3
		MALICIOUS_CLIENT_COUNT = 120
		MALICIOUS_TRAFFIC_RATE = 25
			Done	Download

5	Balancing Pressure	NUM_SERVERS = 12
		PROCESSING_POWER = 300
		MAX_REQUESTS_CONCURRENT = 35
		MAX_REQUEST_QUEUE_LENGTH = 600
			Done	Download

6	High Server Outage	NUM_SERVERS = 5
		PROCESSING_POWER = 180
		MAX_REQUESTS_CONCURRENT = 20
		MAX_REQUEST_QUEUE_LENGTH = 300
			Done	Download

7	Moderate Server Downtime	NUM_SERVERS = 6
		PROCESSING_POWER = 200
		MAX_REQUESTS_CONCURRENT = 22
		MAX_REQUEST_QUEUE_LENGTH = 350
			Done	Download

8	Heavy Attack Payload	PROCESSING_POWER = 200
		MALICIOUS_LOAD_SIZE_LOWER = 25
		MALICIOUS_LOAD_SIZE_UPPER = 40
			Done	Download

9	Large Infected Client Assault	MALICIOUS_CLIENT_COUNT = 200
		MALICIOUS_TRAFFIC_RATE = 25
			Done	Download

10	Network Endurance	REQUEST_TIMEOUT = 5
		MALICIOUS_TRAFFIC_RATE = 18
		MALICIOUS_CLIENT_COUNT = 120
			Done	Download

2.2 Run Parameter Configurations

User configurable parameters such as the CPU utilization health weight (0.4), queue utilization health weight (0.6), increased utilization threshold (0.70), high utilization threshold (0.85), and critical utilization threshold (0.95) did not deviate from their original values over the course of simulation runs. This was done in order to preserve the validity of execution outcomes and ensure that differences in simulation outcomes were directly attributable to simulation pipeline changes, as opposed to variations in calculation weights. Furthermore, simulation duration (900), interval data polling (5), and interval output polling (30) also remained the same throughout the course of simulation runs to maintain the same level of data visualization granularity. The rest of the parameters for each run were configured as follows:

Run 1: Baseline Configuration

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 2: Intense HTTP Flood

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 8
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 40
- MALICIOUS_CLIENT_COUNT: 150
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 3: High Legitimate Traffic

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 4
- LEGITIMATE_CLIENT_COUNT: 1000
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 4: Mixed Workload Stress

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 3
- LEGITIMATE_CLIENT_COUNT: 700

- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 25
- MALICIOUS_CLIENT_COUNT: 120
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 5: Load Balancing Pressure

- NUM_SERVERS: 12
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 300
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 35
- MAX_REQUEST_QUEUE_LENGTH: 600
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 6: High Server Outage

- NUM_SERVERS: 5
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 180
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 20
- MAX_REQUEST_QUEUE_LENGTH: 300
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 7: Moderate Server Downtime

- NUM_SERVERS: 6
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 200
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 22
- MAX_REQUEST_QUEUE_LENGTH: 350
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 8: Heavy Attack Payload

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 200
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 15
- MALICIOUS_CLIENT_COUNT: 80

- MALICIOUS_LOAD_SIZE_LOWER: 25
- MALICIOUS_LOAD_SIZE_UPPER: 40

Run 9: Large Infected Client Assault

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 3
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 25
- MALICIOUS_CLIENT_COUNT: 200
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

Run 10: Analyzing Network Endurance

- NUM_SERVERS: 8
- SERVER_TIMEOUT: 2
- PROCESSING_POWER: 250
- REQUEST_TIMEOUT: 5
- MAX_REQUESTS_CONCURRENT: 25
- MAX_REQUEST_QUEUE_LENGTH: 400
- OFFLINE_CLEAR_THRESHOLD: 0.5
- HIGH_UTILIZATION_REJECTION_RATE: 0.15
- LEGITIMATE_TRAFFIC_RATE: 2
- LEGITIMATE_CLIENT_COUNT: 400
- LEGITIMATE_LOAD_SIZE_LOWER: 1
- LEGITIMATE_LOAD_SIZE_UPPER: 3
- MALICIOUS_TRAFFIC_RATE: 18
- MALICIOUS_CLIENT_COUNT: 120
- MALICIOUS_LOAD_SIZE_LOWER: 12
- MALICIOUS_LOAD_SIZE_UPPER: 20

2.3 Execution Environment Details

Simulation executions were performed and modified with the following machine and development environment specifications:

Machine Specifications

- Operating System: Windows 10
- CPU: Intel Core i5-9600K
- Mainboard: GIGABYTE B365M DS3H
- Memory: 16 GB Dual Channel DDR4
- GPU: AMD Radeon RX 6700 XT
- Storage: 222 GB SSD, 2TB HDD

Development Tooling

- Programming Language: Python Version 3.13
- Key Dependencies: SimPy Version 4.1.1, Matplotlib Version 3.10.6
- Development Environment: PyCharm
- Simulation Framework: SimPy Discrete-Event Simulation Engine

2.4 Environment Run Issues

The main issue when initially collecting simulation results was memory exhaustion and unexpected simulation outcomes. When first implementing research-driven configuration parameters, it became apparent that utilizing enterprise-scale infrastructure parameters would not be feasible, as these configuration values generated results that were far too large to store and accurately manage. In many cases, final values would appear as zero or negative numbers, indicating overflow and memory allocation issues.

As a result, extensive research was conducted in order to establish infrastructure ratios that accurately captured the real-world relationships of core simulation components. In the final implementation, many of these values have been scaled down roughly 100:1 while maintaining their relative sizes to one another. To illustrate, based on an industry report published by OneChassis, enterprise data centers typically house between 500 and 5,000 servers, while in the simulation implementation the number of servers can be between five and fifty.

By scaling configuration parameters to something more manageable, the simulation is able to preserve realistic relationships among attack ratios, network surface area, and processing capability whilst mitigating memory issues caused by the capabilities of the host machine.

3 Data Collection Overview

3.1 Description of Metrics Collected

The simulation comprehensively collects a multitude of different metrics across categories such as performance, system state, event counts and timings, resource consumption, quality and accuracy measures, and configuration parameters. The metrics collected are broken down as follows:

- **Performance**: Total Requests Received, Total Requests Processed, Success Rate, Acceptance Rate, Processing Rate, Drop Rate, Overall Success Rate, Legit Success Rate, Botnet Success Rate, Total Generated Requests, and Total Served Requests
- **System State**: Time Spent Offline, Offline Percentage, Simulation Duration, Active Workers, Health Score, Total Processing Capacity, and Capacity Threshold
- **Event Counts and Timings**: Drops Queue Full, Drops Timeout, Drops High Load, Total Drops, Legit No Response, and Botnet No Response
- **Resource Consumption**: CPU Utilization, Queue Utilization, and Queue Length
- **Quality and Accuracy Measures**: Bandwidth Exhaustion Probability, Resource Depletion Probability, and Successful Attack Probability
- **Configuration Parameters**: Number of Servers, Processing Power, Queue Size, Legitimate Clients, Malicious Clients, Legitimate Traffic Rate, Malicious Traffic Rate, Request Timeout, and Server Timeout

By collecting metrics at such a high level of granularity and detail across the network and at individual servers, users can perform detailed analysis of system-wide behaviors and can precisely identify the impact of focused network traffic over time.

3.2 Data Samples and Excerpts

Upon the completion of a simulation execution, a comprehensive report is generated capturing metrics such as periodic state snapshots at both the network and server level, individual event records, aggregate statistics and calculations, and parameters used. This is done in order to provide a high-granularity visualization of the simulation outcomes. To illustrate, provided are finalized data samples and excerpts from the baseline configuration. While this collection is not as granular and detailed as the data provided in the final execution report, these data samples do an excellent job displaying final outcomes and illustrating the level of depth that is achieved by the final simulation implementation.

Table 2. Parameters for Baseline Configuration

Parameter	**Value**
Timestamp	2025-10-23 13:30:30
Simulation Duration	900 seconds
Number of Servers	8
Processing Power	250 requests/second per server
Queue Size	400 requests
Legitimate Clients	400
Malicious Clients	80
Legitimate Traffic Rate	2 requests/second
Malicious Traffic Rate	15 requests/second
Request Timeout	3 seconds
Server Timeout	2 seconds

Table 3. Final Simulation Results for Baseline Configuration

Metric	**Value**
Total Generated Requests	1,799,520
Total Served Requests	1,593,651
Total Dropped - Queue Full	40
Total Dropped - Timeout	2,620
Total Dropped - High Load	190,969
Total Dropped - No Server	12,240
Overall Success Rate	**88.56%**
Overall Drop Rate	**11.44%**

Table 4. Probability Analysis for Baseline Configuration

Metric	**Value**
Bandwidth Exhaustion	60.00%
Resource Depletion	16.39%
Successful Attack	2.23%
Total Processing Capacity	2,000 req/sec
Capacity Threshold	1,600 req/sec

Table 5. Per-Server Final Performance Metrics for Baseline Configuration

Server	Succ %	Queue Drop %	Timeout Drop %	High Drop %	CPU %	Queue %	Off %
NetworkServer0	89.04%	0.00%	0.14%	10.81%	98.9%	83.4%	0.2%
NetworkServer1	89.07%	0.00%	0.14%	10.79%	98.9%	83.9%	0.4%
NetworkServer2	89.09%	0.00%	0.15%	10.75%	98.3%	82.8%	0.4%
NetworkServer3	89.18%	0.00%	0.15%	10.67%	99.4%	83.5%	0.4%
NetworkServer4	89.27%	0.00%	0.15%	10.58%	99.4%	83.6%	0.4%
NetworkServer5	89.18%	0.00%	0.14%	10.67%	98.3%	83.0%	0.4%
NetworkServer6	89.39%	0.00%	0.15%	10.45%	98.9%	83.5%	0.4%
NetworkServer7	89.11%	0.00%	0.15%	10.75%	99.4%	82.9%	0.2%

Table 6. Total Network Time-Series Performance Summary for Baseline Configuration

Time (s)	Success %	Drop %	Legit Success %	Botnet Success %	CPU %	Queue %	Avg Health Score
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.000
5	48.03%	38.60%	44.53%	50.15%	75.00%	35.09%	0.489
30	70.70%	24.35%	69.75%	71.33%	100.00%	86.00%	0.084
60	80.15%	17.47%	78.38%	81.31%	100.00%	82.88%	0.103
120	84.65%	14.11%	82.43%	86.13%	100.00%	86.31%	0.082
180	86.08%	13.09%	83.49%	87.80%	100.00%	86.53%	0.081
240	86.76%	12.63%	83.90%	88.66%	100.00%	85.78%	0.085
300	87.15%	12.36%	84.15%	89.15%	100.00%	86.06%	0.084
360	87.45%	12.14%	84.37%	89.49%	100.00%	86.22%	0.083
420	87.63%	12.02%	84.46%	89.74%	100.00%	86.69%	0.080
480	87.76%	11.93%	84.52%	89.92%	100.00%	86.47%	0.081
540	87.88%	11.85%	84.60%	90.06%	100.00%	86.50%	0.081
600	88.04%	11.72%	84.68%	90.23%	100.00%	83.00%	0.108
660	88.18%	11.60%	84.82%	90.46%	100.00%	83.97%	0.096
720	88.29%	11.52%	84.86%	90.61%	100.00%	82.09%	0.107
780	88.38%	11.44%	84.94%	90.76%	100.00%	80.72%	0.116
840	88.47%	11.36%	85.03%	90.90%	100.00%	83.91%	0.097
900	88.56%	11.44%	85.10%	90.85%	99.00%	83.30%	0.099

The full execution report for each run parameter configuration can be downloaded from the Run Summary Table in Section 2.1., or accessed at https://github.com/AMVGH/NetworkRunData.

3.3 Data Visualizations

Upon the completion of a simulation execution, multiple time-series graphs are generated, visualizing metrics such as: *Server Utilization* (CPU Utilization by Server Over Time, Queue Utilization by Server Over Time, Queue Length by Server Over Time), *Request Servicing* (Request Generation v. Service Rate Over Time), *Throughput and Drop Rate* (Legitimate Throughput Over Time, Legitimate Drops Over Time, Total Drops Over Time), and *Traffic Comparison and Impact* (Legitimate and Malicious Generation Over Time, Legitimate and Malicious Success Rate Over Time, Server Health Over Time). Attached are the time-series graphs for the baseline configuration.

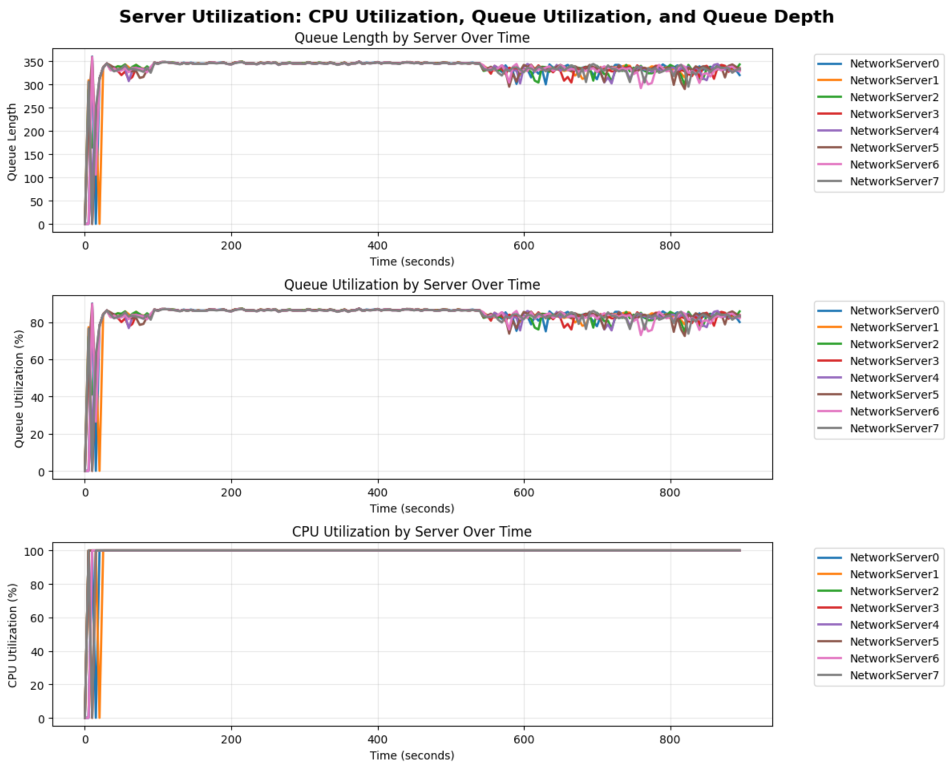

Fig. 1. Utilization metrics by server over time for baseline configuration.

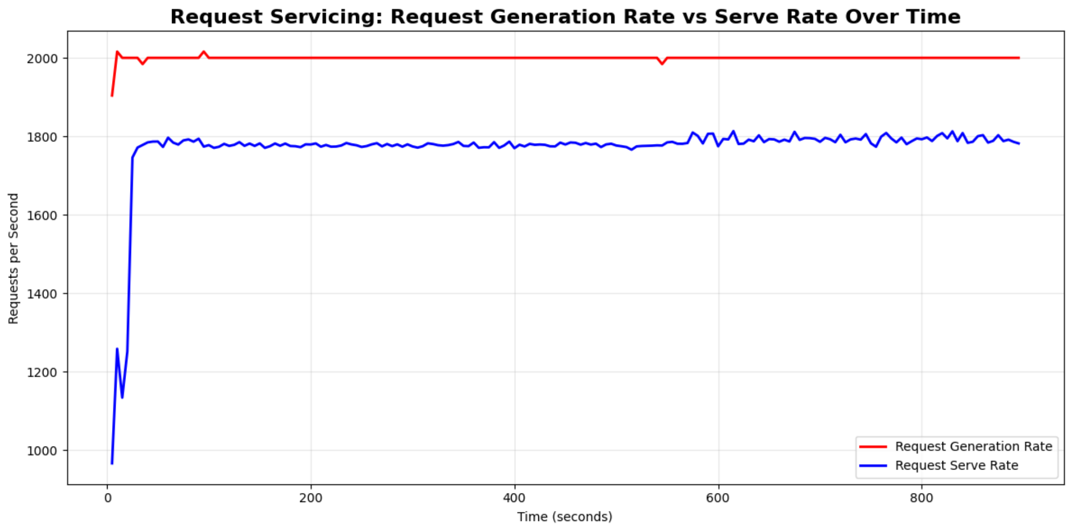

Fig. 2. Request generation rate and request service rate over time for baseline configuration.

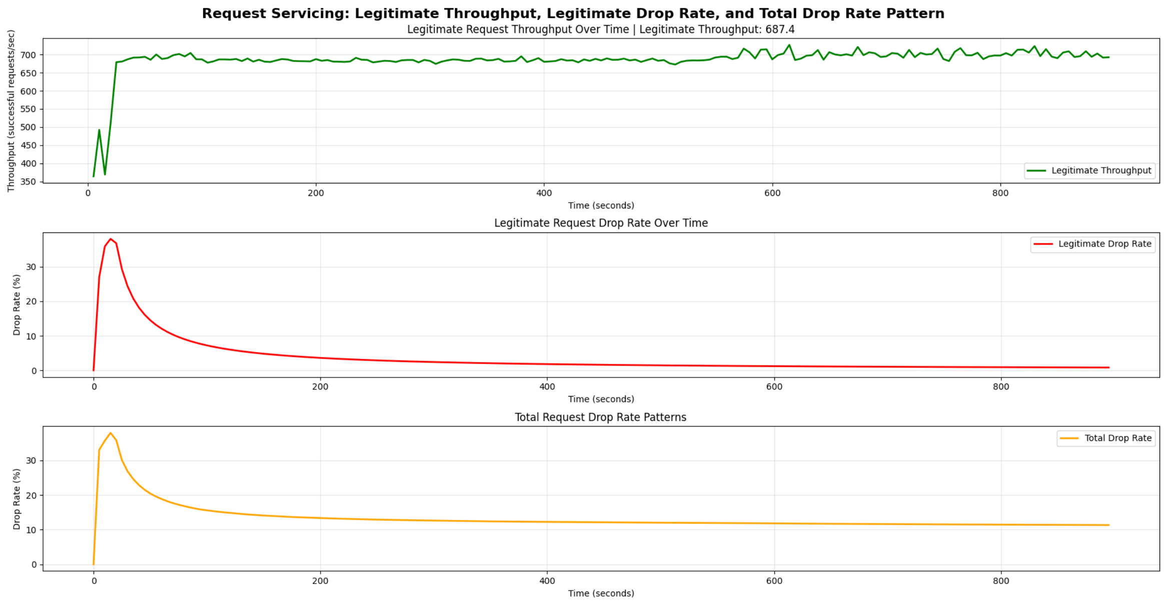

Fig. 3. Legitimate request throughput, legitimate request drop rate, and total request drop rate over time for baseline configuration.

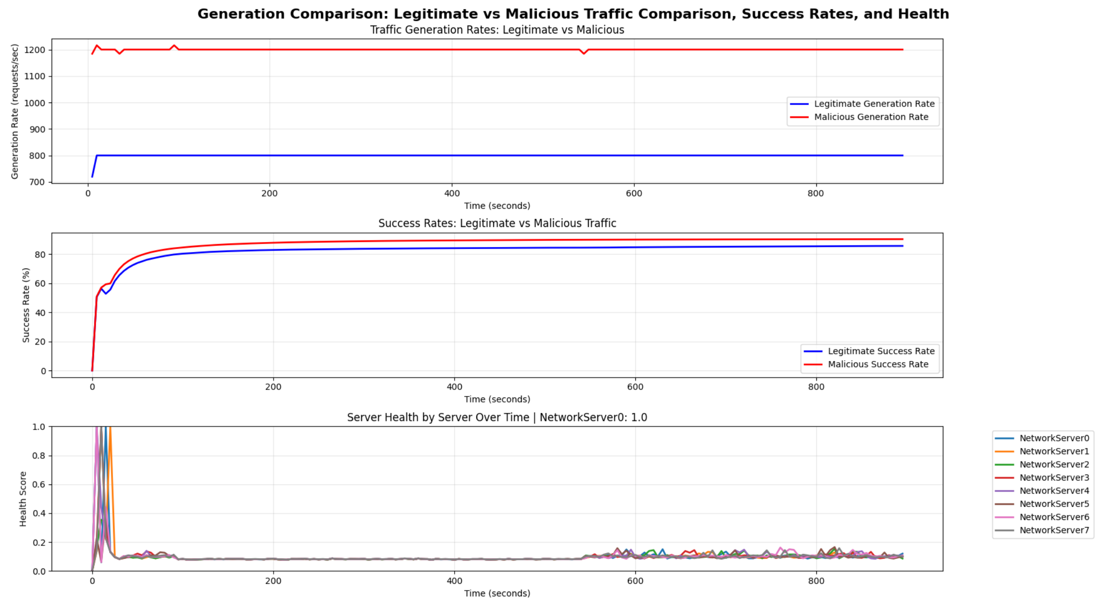

Fig. 4. Legitimate and malicious traffic rate, legitimate and malicious request success rate, and server health over time for baseline configuration.

3.4 Initial Data Observations

From observing the initial data as well as the tabular data for the baseline configuration outlined in Section 3.2, there are multiple key takeaways that can be observed from the baseline execution. It is clear that although the target network sustained stress from the attacking network, there was still a lot of overhead room to service incoming requests, as there was an observed success rate of 88.56% accompanied by an overall drop rate of 11.44%.

It is clear from the visual data that a majority of the dropped requests came from early request generation as opposed to a sustained drop throughout the execution's lifetime. As illustrated in the graphics, as time elapsed these drop rates tapered off and became more stable as the target network processed requests and the network router routed requests to network servers.

Across all eight servers, the CPU utilization remained around 98% while queue utilization hovered around 83%. These metrics alongside the data displayed in Figure 1 illustrates that the health-based load balancing algorithm is distributing traffic evenly and in a fashion that allows servers to push - but not exceed - their performance limitations. This sentiment is echoed when looking at the Offline % in Table 5, as no server exceeds more than 0.4% downtime across the 900 second period.

In all, the baseline configuration has served as a strong reference point for additional simulation executions, as it illustrates that system components are working as expected and the process interactions between simulation pipeline components - LegitimateTrafficNetwork, Botnet, Network, NetworkRouter, and NetworkServer - are behaving predictably. Since parameters in the baseline configuration are derived directly from relevant domain research, the execution outcomes for the baseline configuration have served as a strong validation tool for the simulation's stability and accuracy in modeling real-world HTTP flood Distributed Denial of Service (DDoS) attacks.

4 Preliminary Results

4.1 Basic Statistics from Simulation Runs

Table 7. Basic Statistics from Simulation Runs Across All Configuration Scenarios

Run	Scenario	Success %	Drop %	Attack Success	Total Generated	Total Served	Lead Drop Cause
1	Baseline	88.95%	11.05%	0.43%	1,799,520	1,600,682	High Load (193,723)
2	Intense Flood	15.76%	84.24%	88.27%	6,119,600	964,570	No Server (5,123,638)
3	High Legitimate	22.92%	77.08%	74.87%	4,678,920	1,072,344	No Server (3,576,292)
4	Mixed Workload	21.89%	78.11%	80.83%	4,589,300	1,004,774	No Server (3,554,066)
5	Load Balancing	99.99%	0.01%	0.00%	1,799,520	1,799,321	High Load (134)
6	High Outage	23.63%	76.37%	77.35%	1,799,520	425,314	No Server (1,339,182)
7	Moderate Outage	34.72%	65.28%	64.58%	1,799,520	624,777	No Server (1,140,011)
8	Heavy Payload	32.18%	67.82%	65.44%	1,799,520	579,163	No Server (920,977)
9	Large Botnet	18.59%	81.41%	85.45%	5,219,600	970,443	No Server (4,216,698)
10	Endurance	38.02%	61.98%	61.14%	2,663,600	1,012,601	No Server (1,617,785)

4.2 Simple Charts or Tables Showing Trends

Request Servicing

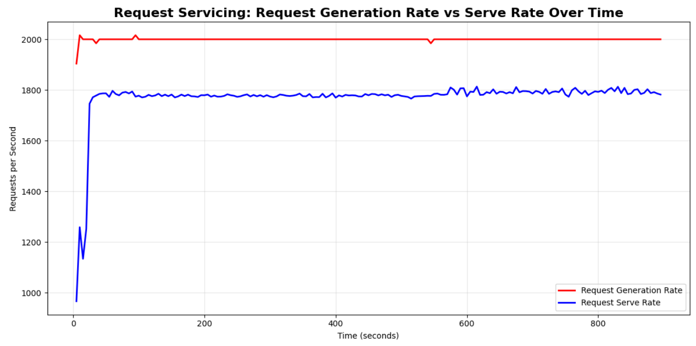

Fig. 5. Request generation rate vs service rate over time for baseline configuration.

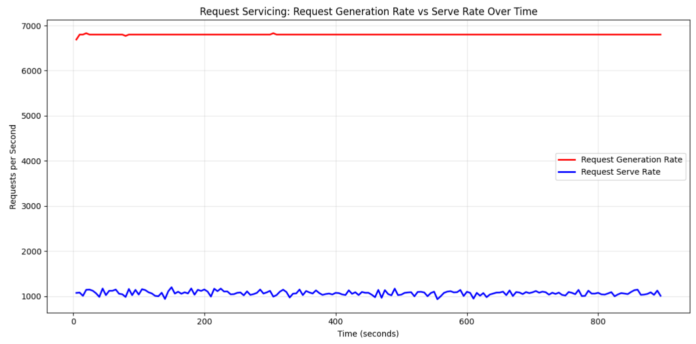

Fig. 6. Request generation rate vs service rate over time for intense HTTP flood configuration.

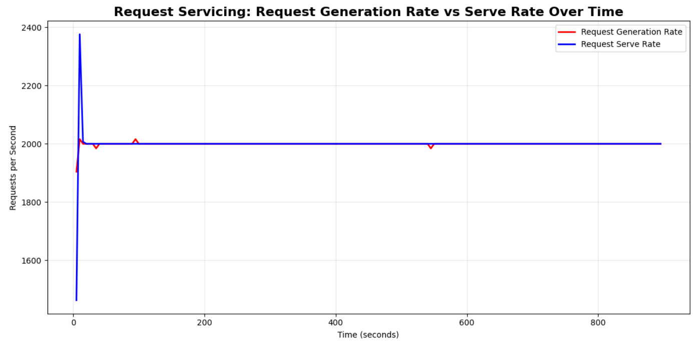

Fig. 7. Request generation rate vs service rate over time for load balancing pressure configuration.

Throughput and Drop Rate

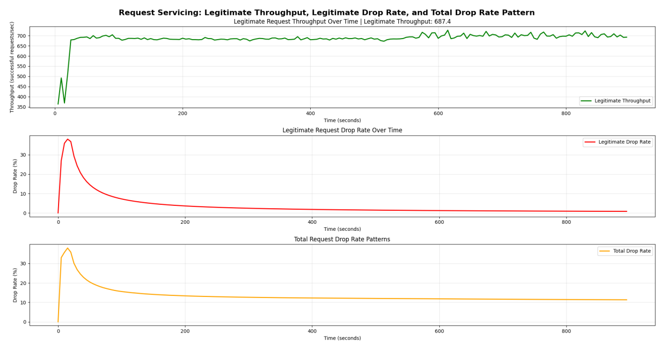

Fig. 8. Legitimate request throughput, legitimate request drop rate, and total request drop rate over time for baseline configuration.

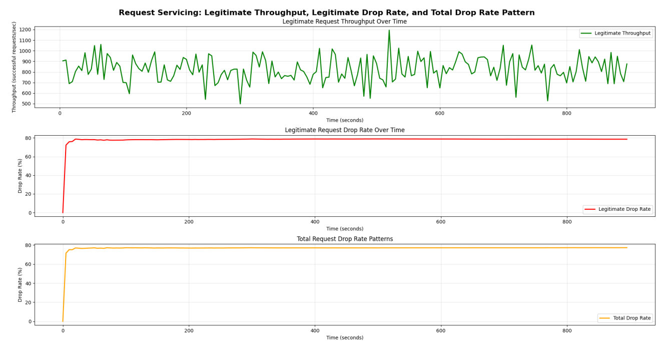

Fig. 9. Legitimate request throughput, legitimate request drop rate, and total request drop rate over time for intense HTTP flood configuration.

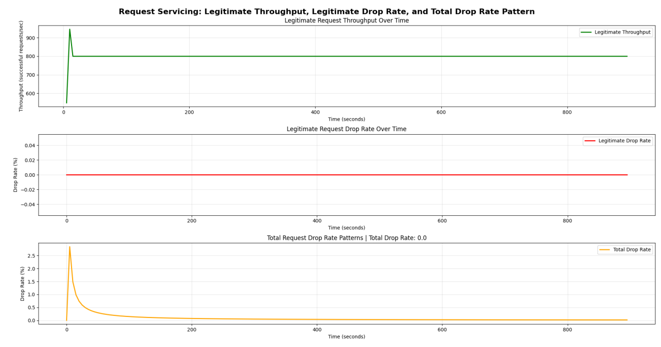

Fig. 10. Legitimate request throughput, legitimate request drop rate, and total request drop rate over time for intense load balancing pressure configuration.

Traffic Comparison and Impact

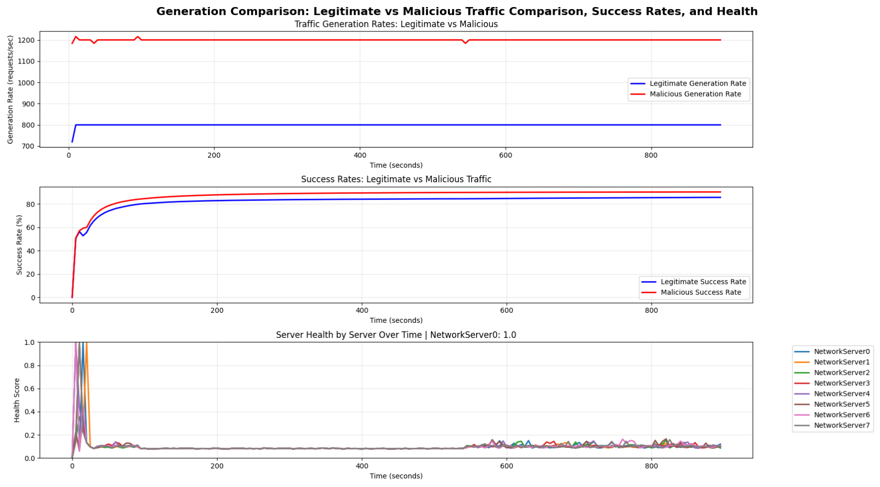

Fig. 11. Legitimate and malicious traffic rate, legitimate and malicious request success rate, and server health over time for baseline configuration.

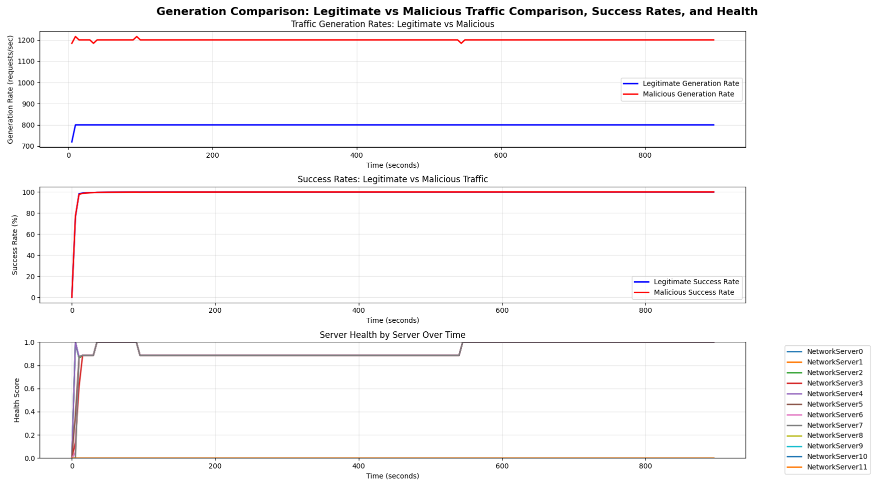

Fig. 12. Legitimate and malicious traffic rate, legitimate and malicious request success rate, and server health over time for load balancing pressure configuration.

4.3 Performance Observations

The simulation data demonstrates that there are clear request generation thresholds that impact system service as a whole. When the total generated requests exceeds two million - as in executions 2, 3, 4, and 9 - success rates consistently remain under 25%. However, there is reason to believe that the low success rate is directly a result of unsuitable network infrastructure, as the leading cause of network request drops in these executions is no server being available. This sentiment is echoed when observing execution 5, as the number of servers and their respective processing abilities have been increased in this execution's parameter configuration. Since execution 5 has a much stronger network infrastructure, the success rate remains close to 100% despite having a comparable number of requests generated.

This performance observation is illustrated in the opposite direction as well. When taking a look at executions 1, 6, and 7, the total requests generated is exactly 1,799,520 across all three simulation executions. However, there key difference between the three of these executions lies in the processing ability of the network. The baseline configuration - execution 1 - has eight network servers with a processing power of 250, a concurrent request maximum of twenty-five, and a request queue length of 400. On the other hand, execution 7 has six servers, with a processing power of 200, concurrent maximum of 22, and max queue length of 350. Finally, execution 6 had five servers, processing power of 300, concurrent maximum of 20, and max queue length of 300. Despite what would seem like small changes in the processing ability, there is a 54.23% success drop between executions 1 and 7, and additional 11.09% drop between executions 7 and 6. These outcomes are a stark indicator of the importance of the network processing ability and how distributing the traffic among servers allows for exponential gains in network throughput.

Finally, the simulation data indicates that although the mean load size for a malicious request is higher, high legitimate traffic can be just as - if not more - disruptive than malicious traffic if the network infrastructure does not permit. Execution 3 illustrates this, as the configuration value for legitimate client count was 1000, and the legitimate traffic rate was four. Under these conditions, execution 3 saw a 22.92% success rate and 77.08% drop rate across 4,678,920 requests, indicating catastrophic failure for a network receiving high rates of legitimate traffic.

4.4 Pattern Observation

As simulation executions occurred, two key patterns emerged reflecting key interactions among the simulation components: 1) Under all circumstance there will be early-stage performance degradation immediately followed by system stabilization 2) Network infrastructure is directly correlated to network stability over time. Across all simulation executions, a sharp spike can be seen at the beginning of the total request drop rate. While in some instances this is maintained, indicating a system under severe load, in execution 5 this drop rate quickly jumps and immediately drops following the beginning of the simulation execution. This suggests that while the health-based load balancing algorithm evenly distributes traffic among network servers, there is an initial period in which the network router has to respond to incoming server traffic and server-health weights have not been updated to reflect this influx.

Secondly, the simulation outcomes clearly demonstrate that network infrastructure is tightly coupled to network stability over time. While the tabular data and reasoning in Section 4.3 explain this, this is also demonstrated when analyzing the visual data for captured in Section 4.2. Configurations with higher server counts and increased processing capability tend to exhibit tightly coupled request generation and service rates. Execution 5 had a success rate of 99.99%, as such, there is little to no variation in the generation and service rate for the load balancing pressure configuration. Even in instances where the execution configuration saw a lower win-rate, this pattern is evident. With this, it is fair to reason that in execution configurations where there are less resources allocated to the target network, the generation and service rate graphic would be similar to that of the intense flood configuration.