Enhancing Network Simulation Accuracy through Calibration and Model Optimization*

Maede Davoudzade[1,*,†], Chris Barber[2,†], Babak Esfandiari[3,†] and Thomas Kunz[4,†]

[1]Systems and Computer Engineering, Carleton University, Ottawa, Canada
[2]Blue Planet, Ciena, Ottawa, Canada
[3]Systems and Computer Engineering, Carleton University, Ottawa, Canada
[4]Systems and Computer Engineering, Carleton University, Ottawa, Canada

Abstract

Accurate network simulations are critical for replicating real-world performance and guiding design decisions. Parameter calibration is needed to improve simulation accuracy, particularly with regards to End-to-End delay. Network simulation tools typically simplify or ignore processing delay, resulting in inaccurate results when considering this factor. We use a linear optimization approach to enhance simulation accuracy by incorporating processing delay as a key parameter. This adjustment reduces the discrepancy between simulation outcomes and real-world network behavior. However, even after calibration, certain models may still fail to fully capture the complexity of real-world networking devices. In such cases, developing a new model is necessary to better reflect device-specific characteristics. To demonstrate this, we present a case study highlighting the limitations of existing simulation models and validate the effectiveness of our proposed model in accurately replicating real network conditions, particularly in terms of End-to-End delay.

Keywords

Network Simulation Accuracy, Simulation Calibration, Linear Optimization, End-to-end Delay, Processing Delay

1. Introduction

1.1. Context

Network simulations are essential tools for researchers and engineers to test, analyze, and validate network protocols and designs before real-world implementation. The construction and testing of real networks is costly and time-consuming, while simulations provide a cost-effective alternative. Accurate simulations are crucial for reliable, real-world-like results, helping fine-tune network parameters, diagnose issues, and plan upgrades by predicting performance impacts of changes. [1, 2]

Inaccurate simulations can lead to flawed predictions of network performance, particularly for metrics like delay, bandwidth utilization, jitter, and packet loss. Underestimating or overestimating these metrics can cause network designers to make poor decisions [3]. Real-world networks are complex due to traffic patterns, protocol interactions, and hardware constraints, which inaccurate simulations may fail to capture, especially in critical areas like congestion control and packet loss [4]. This can result in protocols or configurations that perform well in simulations but fail in real deployments [3, 5]. Inaccurate simulations also undermine the credibility of research, slowing progress in networking studies [5]. Simplified models may overlook real-world effects, leading to overly optimistic results. One such overlooked factor is processing delay, which can have a significant impact on network performance but is frequently simplified in simulations [6].

In simulation accuracy validation, comparing key metrics such as End-to-End delay is essential for capturing a comprehensive understanding of the network's behavior. End-to-End delay is a critical metric for ensuring that latency-sensitive applications meet quality of service (QoS) requirements [7, 8].

Companion Proceedings of the 17th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling Forum, M4S, FACETE, AEM, Tools and Demos co-located with PoEM 2024, Stockholm, Sweden, December 3-5, 2024
*

maededavoudzade@sce.carleton.ca (M. Davoudzade); cbarber@ciena.com (C. Barber); babak@sce.carleton.ca (B. Esfandiari); tkunz@sce.carleton.ca (T. Kunz)

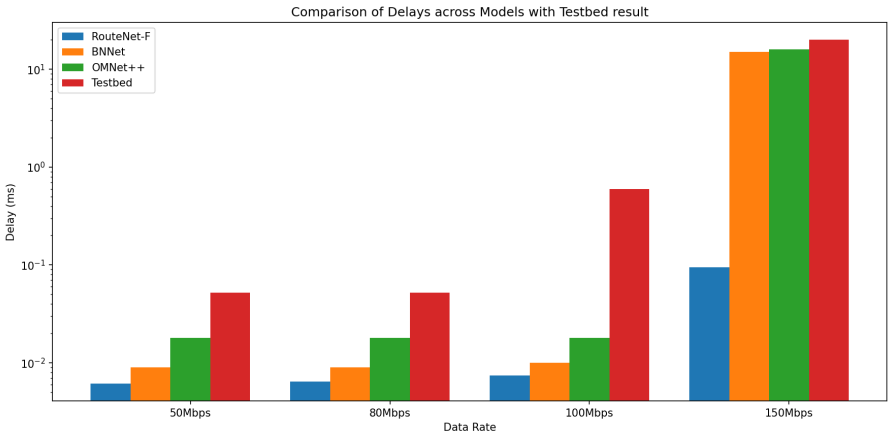

Figure 1: Comparison of End-to-End delays across RouteNet Fermi, BNNet, and OMNeT++ simulations against the testbed results by varying data rates

1.2. Motivation

End-to-End delay is the sum of four distinct delays: transmission delay, queuing delay, propagation delay and processing delay, summed up over all routers and links a packet traverses. Transmission delay is the time it takes to push all the bits of a packet onto the link. Queueing delay is the time a packet spends waiting in a queue before it can be transmitted. Propagation delay is the time it takes for the signal to travel from the sender to the receiver. Processing delay refers to the time a router or network device spends handling a packet. Processing delay is influenced by factors such as the hardware of the router, the complexity of routing decisions, and the type of operations required, such as encryption or traffic filtering [9, 10, 11].

In many network simulations, processing delay is often ignored or simplified because it was historically considered negligible[12]. It is essential for realistic network simulations to incorporate processing delay, especially in modern high-speed networks where such delays are becoming more pronounced [13].

We conducted a case study using a real network provided by Ciena, with the objective of building a simulation model that accurately replicates the real network. We only take two routers that are directly connected together. End-to-End delay was measured across multiple simulation tools, including RouteNet Fermi [14], BNNet [15], and OMNeT++ [16]. Consistent configurations were maintained, such as a 100 Mbps link bandwidth, 1,600,000-bit buffer size, and a uniform traffic setup with 200-byte message lengths. Data rate variations are illustrated in Figure 1. RouteNet Fermi, which requires training on real network data, exhibited notable performance gaps compared to the testbed, particularly under high-load scenarios. In contrast, BNNet and OMNeT++ produced more accurate delay predictions. These results are shown in Figure 1.

Ultimately, OMNeT++/INET was selected as the primary simulation tool for its flexibility in handling packet sizes and its reliability without the need for additional training. The remainder of this paper focuses on improving the accuracy of OMNeT++/INET simulations to minimize the gap between model and real network performance.

1.3. Contribution

In this paper, we improve the accuracy of network simulations, specifically with respect to End-to-End latency, by incorporating processing delay as a key factor in the calibration process. While calibration helps reduce discrepancies, we recognize that sometimes simulation models may still fall short in capturing the full complexity of real-world networking devices, highlighting the need for developing new models. To address this, we introduce a refined model that better represents device-specific characteristics and validate its effectiveness through a case study.

In other words, we answer the following question in this paper:

1. Can the simulation model closely replicate the real network?
2. What is the effect of integrating processing delay on simulation accuracy?

2. Related Work

In network simulation evaluation, many studies focus on the accuracy assessment of simulation models. For instance, [5] compares the results of two widely used simulators, OPNET Modeler and NS-2, against data from a real network testbed to validate their accuracy. Similarly, [8] examines the performance and scalability of several network simulators, including NS-2, OMNeT++, NS-3, SimPy, and JiST/SWANS, by implementing identical simulation settings across these tools. Both studies highlight the importance of ensuring that simulation results align closely with real-world network behavior.

Model calibration ensures the correct representation of a system the simulation model is supposed to simulate. Paper [17] presents model calibration as tuning model parameters to match the input-output behavior of the reference system and highlights the inherent complexity of model calibration. The paper shows model calibration can be NP-complete and therefore, heuristic or approximate solutions are recommended. Paper [18] presents an optimization method for tuning parameters of middleware services in wireless sensor networks to improve Quality of Service (QoS) metrics like accuracy, response time, and power consumption. Also, paper [19] investigates the automated calibration methods for parallel and distributed computing simulators and proves that algorithms like Grid Search, Random Search, and Gradient Descent are efficient in improving accuracy.

In all the referred papers, the authors relied on the simulation models to replicate the behavior of real networking devices. However, due to specific configurations, available simulations may be unable to accurately reflect the behaviour of such devices like routers, and therefore calibration can be insufficient, see[20, 21]. The proposed model aims to enhance these tools by integrating dynamic routing and performance analysis during congestion to improve simulation accuracy.

While these studies provide valuable insights into simulation validation and calibration, they do not consider the role of processing delay as a key parameter. Our work directly addresses this gap by integrating processing delay into simulation models through a calibration process. However, we recognize that even with this improved calibration, sometimes, simulation models may still fail to capture the complexities of real-world networking devices. To address this limitation, we propose a refined model that better reflects device-specific characteristics and validate its effectiveness through a case study.

3. Model Evaluation and Calibration

Building on the insights from previous studies and addressing the gaps identified in the Related Work section, we propose an approach to improve the accuracy of network simulations by incorporating processing delay. Our approach is structured around two main tasks: Validating the Simulation Model and Calibrating the Simulation Model.

We begin by evaluating the accuracy of the simulation model. If a significant gap is identified between the simulation results and the real network data, we proceed with calibrating the model by incorporating processing delay as a key parameter. After the calibration, we re-evaluate the model to determine if the discrepancies have been minimized. However, if a significant gap persists, we question whether the simulation model accurately replicates the behavior of the actual network device. In such cases, a new model may need to be developed, tailored specifically to the device's behavior to improve simulation accuracy.

3.1. Validating the Simulation Model

Our evaluation process begins with hypothesis testing to identify any statistically significant differences between the simulation model and real-world data. We set the null hypothesis to state that no significant difference exists between the data sets; if this cannot be rejected, we conclude that the results are *good enough*. Additionally, we calculate an error function to quantify the discrepancy between observed and simulated values, which is essential for calibrating and optimizing the simulation.

3.2. Calibrating the Simulation Model

To enhance the accuracy of our simulation, we introduce the simulation model M, defined as a triple (X_M, P_M, Y_M), where X_M is the input, P_M is the set of parameters, and Y_M denotes the output metrics. Our reference system, the Testbed, has specific input values X_S and corresponding output values Y_S. The objective is to adjust the model parameters P_M so that the model's output Y_M matches the reference system's output Y_S for given inputs.

The inputs, X_M, include detailed network configuration, such as nodes, links, queues, and flows, while the outputs Y_M measure key network performance metrics, including loss, throughput, and End-to-End delay. If the model outputs Y_M differ significantly from Y_S, particularly in terms of End-to-End delay, we perform a calibration by adjusting P_M, specifically incorporating processing delay as a critical factor.

3.2.1. Input Definition X_M

The inputs, X_k, include the network configuration, such as a set of nodes N_k, a set of links L_k, a set of queues Q_k, and a set of flows F_k. These inputs collectively define the environment in which the network operates and manage traffic, determining performance metrics. Further, each l_i in L can be defined as a tuple:

$$l_i = (src_i, dest_i, b_i, \textit{length}_i) \; ; \forall l_i \in L \tag{1}$$

Where,

- src_i is the source node of the link l_i.
- $dest_i$ is the destination node of the link l_i.
- b_i is the bandwidth of the link l_i.
- $\textit{length}_i$ is the length of the link l_i.

Similarly, we have a set of output queues:

$$Q = \{q_i : i \in (1, \ldots, |L|)\} \tag{2}$$

where each queue can be defined as:

$$q_i = (l_i, s_{q_i}, p_{q_i}) \tag{3}$$

where

- l_i is the link associated with this outgoing queue.
- s_{q_i} is the size of the i-th queue.
- p_{q_i} is the policy of the i-th queue such as FIFO, WFQ, RED.

Flows follow a source-destination path. Hence, we define flows as sequences with tuples of the queues and links they traverse along with the message length and packet rate of the flow:

$$f_i = \{(q_{F_q(f_i,0)}, l_{F_l(f_i,0)}, m_i, s_i, t_i), \ldots, (q_{F_q(f_i,n_q)}, l_{F_l(f_i,n_l)}, m_i, s_i, t_i)\} \tag{4}$$

To summarize, we can define f_i as below:

$$f_i = (Q_{f_i}, L_{f_i}, m_i, s_i, t_i) \tag{5}$$

This can be further expanded as:

$$F = \{(Q_{f_i}, L_{f_i}, m_i, s_i, t_i) : i \in (1, \ldots, n_f)\} \tag{6}$$

Where:

- $F_q(f_i, j)$ returns the index of the j-th queue in the network that is part of the path taken by flow f_i .
- $F_l(f_i, j)$ returns the index of the j-th link in the network that is part of the path taken by flow f_i .
- L_{f_i} represents all the links that the i-th flow is taking.
- Q_{f_i} represents all the queues that the i-th flow is taking.
- m_i represents the message length of the i-th flow.
- s_i is the packet rate of the i-th flow.
- t_i is the shape of the traffic for the i-th flow.
- n_q is the number of queues in the path.
- n_l is the number of links in the path.
- n_f is the number of flows in X_i.

Therefore, our input X is a set of X_k:

$$X = \{X_k : k \in (0, \ldots, n)\} \tag{7}$$

$$X_k = (N_k, L_k, Q_k, F_k)$$

3.2.2. Output Metrics Y_M

Our output Y_M is the set of metrics for all flows that can be represented as:

$$Y = \{Y_k : k \in (1, \ldots, n)\}$$

Where each output Y_k includes metrics such as loss, throughput, and End-to-End delay:

$$Y_k = \{(f_{j_l}, f_{j_d}, f_{j_t}) : \forall f_j \, , j \in (1, \ldots, n_f)\}$$

where,

- f_{j_l} is the packet loss for flow f_j
- f_{j_t} is the throughput for flow f_j
- f_{j_d} is the End-to-End delay for flow f_j
- n_f is the number of flows.

Each metric in our output captures different aspects of network performance. Throughput measures the efficiency of data transfer, indicating the network's capacity to handle traffic. Loss measures the reliability of data transfer, showing how often packets are lost due to congestion or errors. Together, they provide a complete picture of network behavior making our simulation reliable and robust. The End-to-End delay is influenced by the intricate relationship between traffic demand, network topology, and network performance [22]. The total End-to-End delay is given by:

$$D_{total} = D_{trans_j} + D_{prop_j} + D_{proc_j} + D_{queue_j}$$

where,

- D_{total} represents the total delay for a flow, which is the End-to-End delay.
- D_{trans_j} is the total transmission delay over each link on the path of flow j.
- D_{prop_j} is the total propagation delay over each link on the path of flow j.
- D_{proc_j} is the total processing delay for each node on the path of flow j.
- D_{queue_j} is the total queuing delay for each outgoing interface on the path of flow j.

Transmission delay for a flow refers to the time it takes to push all the bits of a packet onto the links that the flow traverses. For flow j, the total transmission delay is the sum of the transmission delays for all links l_i on the path of flow j. The transmission delay for each link is a function of the message length m_j and the transmission rate b_i of the link.

We define the total transmission delay for flow j as:

$$D_{trans_j} = \sum_{i \in \text{path}(j)} \frac{m_j}{b_i}$$

Where:

- D_{trans_j} is the total transmission delay for flow j.
- m_j is the message length for flow j.
- b_i is the transmission rate of link l_i on the path of flow j.

Propagation delay refers to the time it takes for a signal to travel from the sender to the receiver along each link in the path of the flow. For flow j, the total propagation delay is the sum of the propagation delays for all links l_i in the path of the flow. Propagation delay depends on the distance, which we previously defined as $length_i$ in Equation 1, between the sender and receiver on link l_i and the speed of the signal s_i, which is typically close to the speed of light (commonly approximated for propagation delay as 2×10^8 m/s).

We define the total propagation delay for flow j as:

$$D_{prop_j} = \sum_{i \in \text{path}(j)} \frac{length_i}{s_i}$$

Where:

- D_{prop_j} is the total propagation delay for flow j.
- $length_i$ is the distance between the sender and receiver on link l_i.
- s_i is the signal speed on link l_i.

3.2.3. Calibration Parameters P_M

P_M plays a fundamental role in aligning the simulation model with the real-world testbed. Our goal is to minimize the gap between the simulation and the testbed by considering processing delay for each router. Although transmission and propagation delays can be easily measured or estimated, and queuing delay is typically calculated by simulation tools, processing delay is often overlooked in the overall delay calculation within these tools. However, it plays a significant role in the behavior of modern routers, especially under high-load traffic conditions. Therefore, we define P_M as the set of processing delays per node:

$$P = \{d_{proc_i} : i \in (0, \ldots, |N|)\}$$

3.2.4. Objective Function

Our objective function is to minimize the delay gap between simulation results and the Testbed by calibrating the processing delay. So, we define our function as below for every input X_i:

$$\min(\sum_{j=1}^{|F|}(d_{testbed_{f_j}} - (d_{simulation_{f_j}} + \sum_{n=1}^{|N|}\omega_{f_j}(d_{proc_n})))), \tag{8}$$

$$\forall f_j \in F_i$$

where:

- F_i is the set of flows defined as part of the input X_i.
- f_j is a flow from F_i.
- $d_{testbed}$ is the End-to-End delay per flow in the testbed.
- $d_{simulation}$ is the End-to-End delay per flow in the simulation.
- ω is the selection function which represents the existence of node n in the flow f_j. It returns 0 if node i is not in the path taken by flow j, otherwise it returns 1.
- d_{proc_n} is the processing delay of node n.

Our constraints are:

- processing delay must be non-negative: $0 \leq d_{proc}$.
- If there is no flow passing by node k, d_{proc_k} is 0.
- We define the maximum gap between the testbed and the simulation as:

$$\Theta = \max(d_{\text{testbed}} - d_{\text{simulation}}).$$

  This defines an upper bound on the total processing delay across all nodes along the path, constrained by:

$$0 \leq \sum d_{\text{proc}} \leq \Theta.$$

  In other words, the cumulative processing delay assigned to each node in the path of a flow should not exceed the maximum observed delay gap between the testbed and the simulation.

Building on the work of [10, 11, 12], a linear model for processing delay was proposed to account for both a constant base cost per packet and an additional cost that scales with packet size. This approach reflects the observation that larger packets require more processing time due to tasks such as inspection, memory management, and checksum computations. The processing delay for packets belonging to a specific flow f_k at router i, denoted as $d_{\text{proc}_i}(f_k)$, is formalized as:

$$d_{pr_i}(f_k) = \alpha_i + \beta_i \cdot l_k \tag{9}$$

Where:

- d_{pr} represents the processing delay for a single packet in the router.
- α is the constant processing delay per packet. This cost accounts for operations that are required for every packet, such as reading the header, performing routing table lookups, and basic error checking. These operations are fixed and apply equally to all packets, irrespective of their length [10, 11, 12].
- β is the delay per byte. This component reflects the increased complexity and time required for processing larger packets, which involve tasks such as memory handling, payload inspection, encryption, and checksum calculations. Since these tasks become more resource-intensive as packet size increases, β is a proportional factor that adjusts the processing delay based on the message length [10, 11, 12].

- l_k represents the size of packets in flow f_k.

In the linear problem define by the objective function in Equation 8 and the constraints listed above, we establish the total processing delay for each router in the network. Our goal is to introduce processing delays that accurately reflect the behavior of each router. Using the model in Equation 9, we identify the optimal values of α and β for each router. Therefore, in Equation 10, the total processing delay for a router is calculated as the sum of the processing times for each packet in every flow passing through the router.

$$d_{proc_n} = \sum_{j=1}^{|F|} \sum_{i=1}^{P_j} (\alpha + \beta \cdot l_{j,i}) \tag{10}$$

Where:

- $|F|$ represents the total number of flows passing through the router.
- P_j is the number of packets in flow j.
- $l_{j,i}$ represents the length of the i-th packet in flow j.

4. Experiment

To validate our approach and minimize the gap between the testbed and simulation results, we conducted a series of experiments using a physical testbed provided by Ciena. The testbed, represented in Figure 2, consists of four routers from three different series mentioned in the figure, with a traffic generator (*IXIA*) to create data flows across the network. We aim to calibrate processing delay by comparing the testbed results with our OMNeT++/INET simulations. The primary objective of the experiments is to adjust the simulation to more accurately reflect real-world behavior by tuning processing delay parameters.

4.1. Setup and Configuration

For our experiments, we used Cisco routers with 100 Mbps links across the network, except for the links connected to *IXIA*, which were set to 1 Gbps. This configuration allowed us to introduce various traffic scenarios and examine the impact of processing delay under different conditions. The overall topology of the testbed is illustrated in Figure 2. The run time for each experiment in the testbed and OMNeT++/INET environments is 12 seconds, and we repeat each experiment multiple times to ensure the reliability of the results. Using varying data rates, the network is tested under different loads—before, during, and after the queue reaches its capacity. For experiments with varying message lengths, the data rate is high enough that, by 12 seconds, the network reaches a steady state.

We set up flows with varying message lengths to observe the behavior of each router under different traffic conditions. Our focus was on *Router R2* to measure End-to-End delay and identify potential discrepancies between testbed and simulation results.

4.2. Processing Delay Calibration

The main objective of the calibration process is to determine the optimal values for the processing delay parameters, α and β, for each router. We began by evaluating *router R2* through a series of experiments involving a single traffic flow with a message length of 300 bytes, adjusting the data rate to between 50 Mbps and 150 Mbps to simulate different network loads. The queue capacity was set to 500 packets, and each experiment was run for 12 seconds in both the testbed and simulation environments.

We conducted 14 experiments on this router to determine its optimal parameters. The mean absolute error (MAE) between the End-to-End delay in the testbed and OMNeT++/INET, shown in Figure 3 (highlighted in blue), indicates that, as expected, there is no significant difference in delay between the testbed and the simulation when traffic levels are low, as confirmed by our hypothesis test.

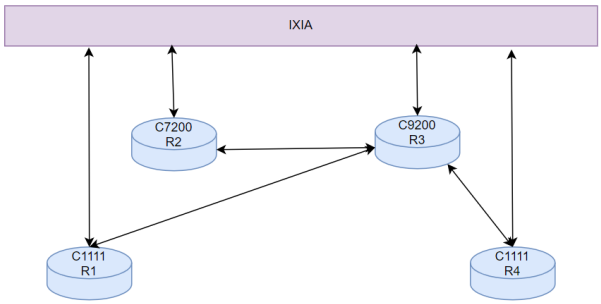

Figure 2: Network topology of the physical testbed consisting of four Cisco routers and an IXIA traffic generator.

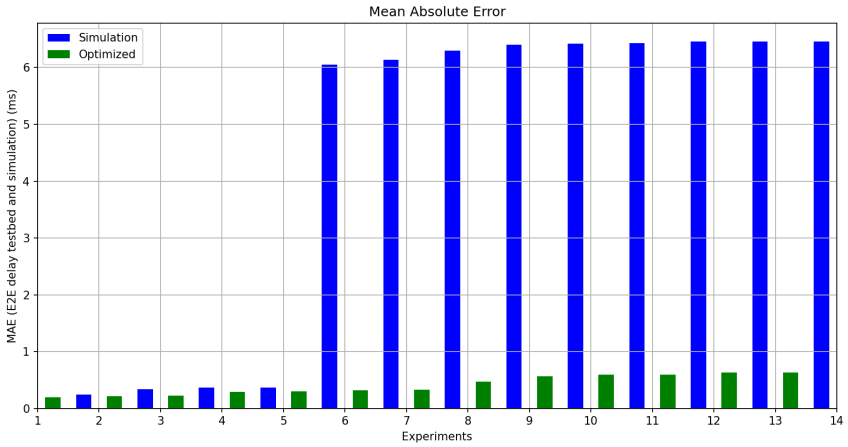

Figure 3: MAE comparison between the testbed, the initial OMNeT++/INET simulation, and the optimized OMNeT++/INET model for *Router R2*

Table 1
Calibrated values of processing delay parameters α and β for Router R2 and R3.

Parameter	α	β
R2 Values	5 (ns)	0.85 (ns / byte)
R3 Values	10 (ns)	1.3 (ns / byte)

However, as traffic load increases and *router R2* becomes congested, we observe a noticeable delay gap between the testbed and simulation results. In Figure 3, this delay gap widens with higher traffic loads, leading us to reject the null hypothesis, indicating a significant difference in delay under congestion. This behavior aligns with findings in previous studies [9, 23], which emphasize the impact of congestion on processing delay. When the queue reaches its capacity, the value of End-to-End delay and the delay gap become stable because once the queue is full, packets are dropped while the remaining packets take almost the same processing delay.

The calibrated values for α and β derived from this experiment are listed in Table 1. Figure 3 (highlighted in green) demonstrates a significant reduction in the delay gap for the calibrated model. According to our statistical tests, we fail to reject the null hypothesis, concluding that these values produce results that are *good enough*.

To validate the calibration and verify the values obtained in the previous step, we conducted further experiments using a range of message lengths from 100 to 1500 bytes. Each experiment ran for 12 seconds in both the simulation and real testbed environments, with a fixed data rate of 150 Mbps to evaluate the model's performance under high traffic conditions. Figure 4 illustrates the End-to-End

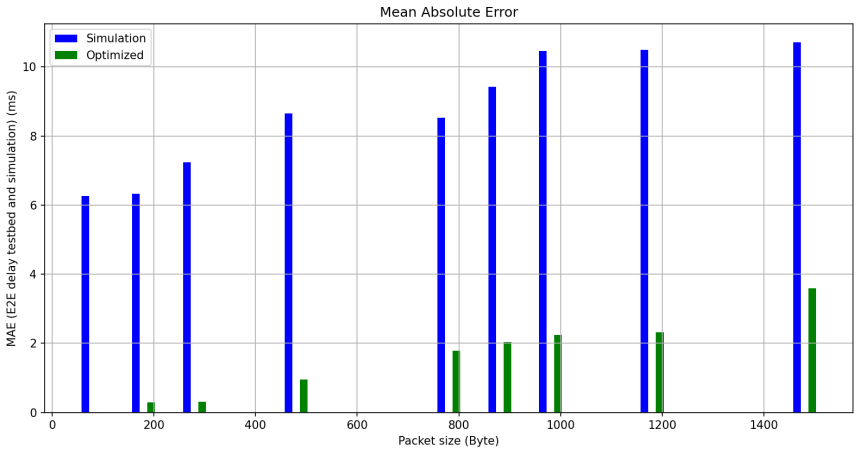

Figure 4: Comparison of End-to-End delays for various message lengths between testbed, OMNeT++, and the calibrated OMNeT++ model for Router R2.

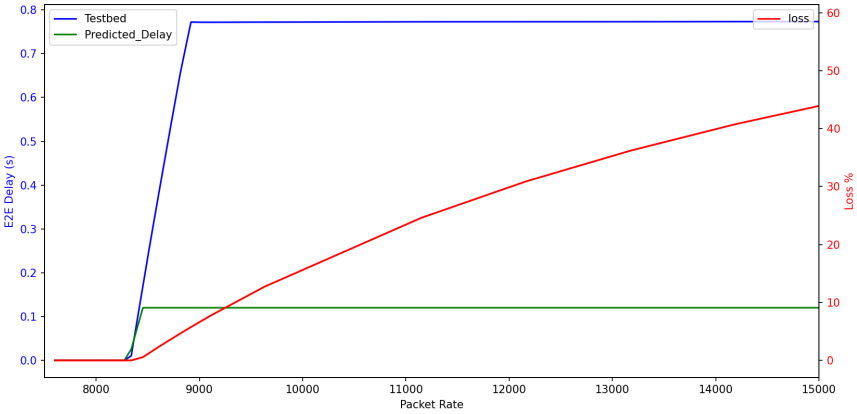

Figure 5: End-to-End delay and packet loss behavior of *Router R1* under a single traffic flow with 1500-byte packets and varying packet rates, compared against OMNeT++ simulation results.

delay results, showing that the calibrated model (*optimized*) significantly reduced the delay gap between OMNeT++ and the testbed and the End-to-End delay from OMNeT++/INET closely matches the testbed results, showing no significant differences.

We then conducted the same experiments on *Router R3*. The results for this router showed slight variations compared to the previous experiments, which is consistent with findings from prior studies [12, 10, 11]. These studies highlight that processing delay is influenced by the specific design and architecture of each router, meaning different types of routers can exhibit varying processing delays. The values obtained for *Router R3* are provided in Table 1. Additionally, we ran the same experiments conducted on *Router R2* to validate the values obtained for this router. Again, our calibrated model effectively reduces the delay gap, similar to *Router R2* (results not shown due to space limitation).

Up to this point, we have successfully calibrated the simulation model for routers in the *C9200* and *C7200* series. Now, the focus shifts to calibrating *Router R1* and *Router R4* from the *C1111* family. We sent traffic with 1500-byte messages, and the packet rates are shown on the x-axis in Figure 5 and each experiment is run for 12s. With a 100 Mbps link, the network can handle 8333 packets per second. The queue size was set to 12,000,000 bits, resulting in an expected queuing delay of 0.12 seconds. The *green* line is representing the expected behaviour once the queue is full with End-to-End delay 0.12s. The *red* line shows packet loss corresponding to OMNeT++/INET.

The measured delay of this router is shown by the *blue* line. For packet rates below 8333 packets

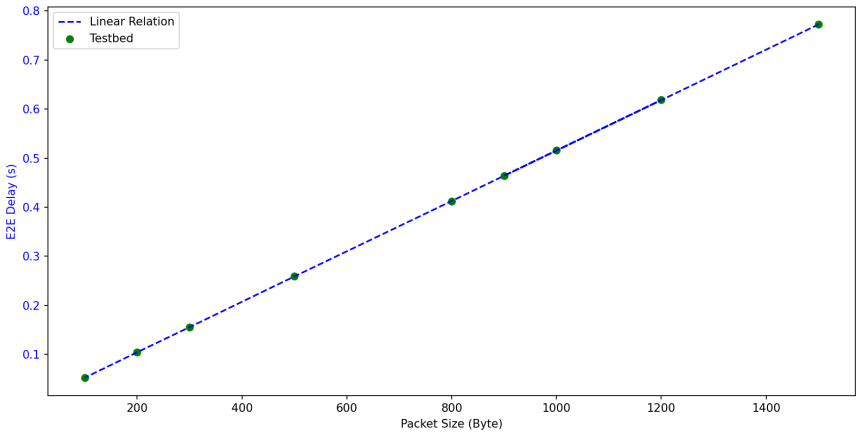

Figure 6: Observed End-to-End delay behavior of *Router R1* for different message lengths, showing a linear relationship between packet size and delay.

per second, the delay matched simulation results. However, when the rate exceeded 8333 packets per second, the queue starts to fill up, causing higher delays and packet loss. Contrary to expectations, the delay rose linearly, stabilizing at 0.78 seconds, indicating potential issues with the router under congestion. Clearly this router was not operating as expected, i.e., as modelled in the OMNeT++/INET router model, given the configuration parameters., So we needed a modified delay model, as discussed below, to ensure that our simulation closely matches the results from the testbed.

4.3. A New Model for Replicating Router Behavior

To accurately model the router's behavior, we need a refined model of the delay introduced by that router. For example, Figure 6 illustrates how delay changes with varying packet sizes. The data rate was set to a high value to observe the delay behavior when the queue reaches full capacity. The figure demonstrates an almost linear relationship between packet size and delay, which can be formalized by the equation below.

$$E2EDelay(\mu s) = \kappa * MessageLength(bytes) + \delta \tag{11}$$

We can replace the numbers based on the router behaviour, for example for router R1 we have:

$$E2EDelay(\mu s) = 510(\mu s/bytes) * MessageLength(bytes) + 8850(\mu s)$$

To accurately model the linearity observed before reaching the maximum delay, it is important to understand how the slope changes based on different packet sizes. We introduce the term *packet rate threshold* (PRT) to represent the maximum number of packets of a given size that can be sent through the link without experiencing any loss.

To identify the appropriate range of packet rates where the delay begins to increase linearly until it reaches its maximum value, we conducted a series of experiments for each packet size. From these experiments, we observed that for nearly all packet sizes, the delay starts to grow in a linear fashion as the packet rate increases. This linear behavior occurs within a specific range of packet rates, defined by $[PRT, PRT + \gamma]$. The *Packet Rate Threshold* (PRT) represents the point at which this linear increase begins, and γ denotes the range where the delay continues to rise linearly until it stabilizes. We refer to this range as the *linear threshold*, captured by γ.

Our objective is to develop a model that accurately captures the behavior of the router, particularly in terms of delay. The model represents a list of flows, each characterized by a pair consisting of a packet rate and message length. If the total traffic load exceeds the available bandwidth of the link, the network will experience congestion, resulting in packet loss and delays. Based on our observations

from Router R1, we noted that the total packet loss is distributed equally among all flows, meaning each flow experiences the same percentage of loss. For example, with two flows defined by (message length, packet rate) pairs, [(1500 bytes, 5000 packets/second), (1000 bytes, 8000 packets/second)], the combined traffic rate amounts to 15,500,000 bytes per second, exceeding the link's capacity by 3,000,000 bytes per second. As a result, we would expect an overall packet loss of approximately 19.3%, and each flow would experience this same percentage of loss. This behavior has been consistently observed across multiple experiments, confirming that it is not a scenario-specific occurrence.

To model the delay, we introduce Algorithm 1. This algorithm replicates the behavior of *Router R1* by predicting the End-to-End delay for any combination of flows. Our observations indicate that the delay is influenced by the flows traversing the same link, so the algorithm focuses on calculating the delay based on these flow characteristics.

The first step is to determine if the given set of flows causes congestion in the link. If congestion is detected (line 1), we then need to calculate the actual packet rate for each packet, i.e., the number of packets successfully transmitted per second for each flow. This is done by computing the total packet loss and loss portion (lines 4 and 5), as described in the previous section on loss behavior. Once the total packet loss is known, we calculate the number of packets lost per second for each flow (line 7). By subtracting this value from the original packet rate, we can determine the number of packets successfully transmitted per second for each flow (line 8).

If the packet rate for a flow lies within the range [*PRT*, *PRT* + γ], or if the experiment duration is too short for the delay to fully stabilize (lines 8, 9, and 10), we calculate the delay by determining how much of the input packet rate falls within the linear threshold. This value is then multiplied by Equation 11 to compute the delay (line 11).

If the packet rate exceeds the linear threshold (LT), or if the experiment runs long enough for the delay to stabilize, the maximum delay is reached.The delay remains stable and does not exhibit a sharp increase until the Packet Rate Threshold (PRT) for each flow is reached. For example, if only the first flow [(1500, 5000)] is present, there would be no delay or packet loss since the total traffic rate remains below the available bandwidth.

However, once a second flow is introduced, delays arise. One limitation of Equation 11 is that it assumes the entire link bandwidth is dedicated to a single flow. In reality, each flow uses only a portion of the bandwidth, so the actual delay for a flow is less than what would occur if the flow were utilizing the full bandwidth. Therefore, we adjust the delay by calculating the fraction of bandwidth used by each flow (lines 12, 14). Given that each experiment lasts for 12 seconds, we divide the calculated delay by the total experiment duration (θ) to account for the running time of the traffic in the network.

Here:

- PRT is *Packet Rate Threshold*(packet/second).
- BW is bandwidth (Mbps).
- ml is message length(Bytes).
- pr is the original packet rate(packet/second).
- Delay is the End-to-End delay.
- t is the total run time(s).
- γ is the packet rate range during which the delay diagram has a linear shape, reaching its maximum value beyond that range.
- κ and δ are the slope and intercept in Equation 11

To validate our proposed delay model for router R1, we conducted experiments. We began with a single flow to ensure the model's accuracy in simple scenarios before progressing to more complex ones. The link bandwidth is 100 Mbps, and the message size is 1500 bytes. The model accurately predicts the delay in this router. Before the packet rate reaches 8333 packets per second, there is no congestion, so the delay remains low. However, during congestion, our linear model calculates the delay for each packet rate, and once it reaches the PRT, the delay remains constant at its maximum value.

Algorithm 1

procedure ModelDelay
  flows $= [(pr_0, ml_0), ..., (pr_i, ml_i)]$
  if $sum(pr_i * ml_i) > BW$: **then**
    $totalByteLoss = sum(pr_i * ml_i) - BW$
    $totalByteLossPortion = totalByteLoss / sum(pr_i * ml_i)$
    for i in flows **do**
      $perFlowPacketLoss = totalByteLossPortion * pr_i$
      $PRT = pr_i - perFlowPacketLoss$
      $tThreshold = \gamma / perFlowPacketLoss$
      if $pr_i < PRT + \gamma$ and $t < tThreshold$ **then**
        $Delay = [(pr_i - PRT)/\gamma * [\beta * ml_i + \alpha]]$
        $Delay = Delay * [PRT * ml_i / BW]$
      else
        $Delay = [\kappa * ml_i + \delta] * PRT * ml_i / BW$
      end if
      $Delay = Delay * t / \theta$
    end for
  end if
end procedure

Table 2
Details of five experiments with varying flow configurations used to validate the delay model for Router R1. The pairs represents as (Message Length (bytes), Packet Rate (packet / second)).

experiment	$flow_1$	$flow_2$
1	(1500,5000)	(1000, 9000)
2	(1000,9000)	(500,12000)
3	(1300,6000)	(900,9000)
4	(1200,7000)	(400,13000)
5	(300,20000)	(200,40000)

Next, we need to validate our model with more complex scenarios involving flows with varying packet sizes and packet rates. To advance this validation, we introduce scenarios that are slightly more complex by including two flows with different packet rates and packet sizes. We conducted five different experiments, with the details of each provided in Table 2. The source of all flows is R1, and the destinations are R2 and R3. Individually, each flow is insufficient to cause link congestion, so the observed delay and loss result from the combined effect of these two flows. The delay model is represented in Fig 7. The y-axis is the delay we observed from router R1 and the x-axis is the delay we calculate based on our model. As the diagram depicts, the linear relation between y-axis and x-axis indicates that our model can perfectly observe the behaviour of this router.

We conducted the same experiments on *Router R4*, which belongs to the same family as *Router R1*. The results were almost identical to those of *Router R1*, further confirming the efficiency and accuracy of the introduced model for routers in this family. These findings demonstrate that the model is capable of accurately predicting End-to-End delay and congestion behavior across multiple routers from the *C1111* series.

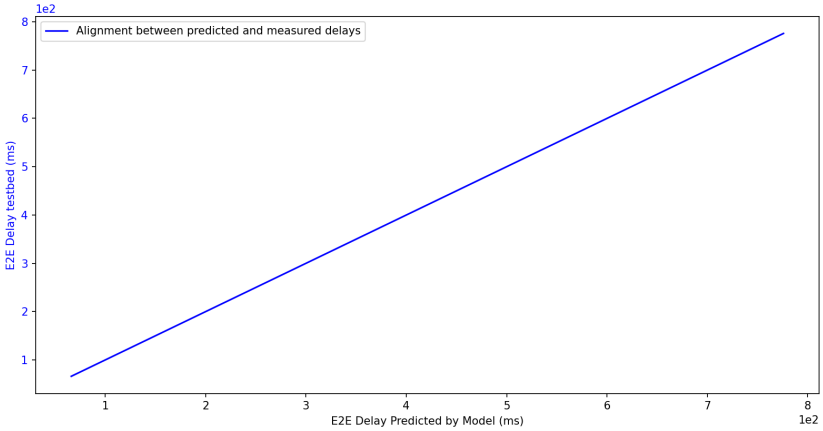

Figure 7: End-to-End delay comparison between model predictions and testbed measurements for Router R1. The line represents how closely the model's predicted delays align with the actual delays measured in the testbed under different packet sizes and rates.

5. Conclusion and Future Work

In this paper, we introduced a novel approach for improving the accuracy of network simulations by incorporating processing delay as a key factor in the calibration process. By integrating this parameter into the simulation model, we were able to better align the simulated performance with real-world behavior, particularly in terms of End-to-End delay. Our experiments demonstrated that the processing delay can significantly impact network performance and that neglecting it in simulations can lead to inaccurate predictions, especially for modern, high-speed networks.

We successfully calibrated our model for routers from the C9200 and C7200 series, showing improved accuracy in capturing their behavior. However, the case study involving the C1111 series highlighted the limitations of existing simulation models when dealing with specific router behaviors. To address this, we proposed a new delay and loss model that more accurately reflects the behavior of routers of that family under complex traffic conditions.

6. Acknowledgments

We acknowledge the support of Ciena and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] A. J. N. Ronit L. Patel, Maharshi J. Pathak, Survey on network simulators, International Journal of Computer Applications 182 (2018) 23–30. doi:10.5120/ijca2018917974.

[2] A. Zarrad, I. Alsmadi, Evaluating network test scenarios for network simulators systems, International Journal of Distributed Sensor Networks 13 (2017). doi:10.1177/1550147717738216.

[3] J. Heidemann, K. Mills, S. Kumar, Expanding confidence in network simulations, IEEE Network 15 (2001) 58–63. doi:10.1109/65.953234.

[4] K. Pawlikowski, H.-D. Jeong, J.-S. Lee, On credibility of simulation studies of telecommunication networks, IEEE Communications Magazine 40 (2002) 132–139. doi:10.1109/35.978060.

[5] G. Flores, M. Paredes-Farrera, E. Jammeh, M. Fleury, M. Reed, Opnet modeler and ns-2: comparing the accuracy of network simulators for packet-level analysis using a network testbed, WSEAS Transactions on Computers 2 (2003).

[6] S. Khan, B. Aziz, S. Najeeb, A. Ahmed, M. Usman, S. Ullah, Reliability of network simulators and simulation based research, in: 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, 2013, pp. 180–185.

[7] Y. Zhou, H. Sethu, A simulation study of relationships between delay jitter and properties of a switching network, SIMULATION SERIES 32 (2000) 227–231.

[8] E. Weingartner, H. vom Lehn, K. Wehrle, A performance comparison of recent network simulators, in: 2009 IEEE International Conference on Communications, IEEE, 2009, p. 1–5. doi:10.1109/icc.2009.5198657.

[9] L. Angrisani, G. Ventre, L. Peluso, A. Tedesco, Measurement of processing and queuing delays introduced by an open-source router in a single-hop network, IEEE Transactions on Instrumentation and Measurement 55 (2006) 1065–1076. doi:10.1109/tim.2006.876542.

[10] P. Carlsson, D. Constantinescu, A. Popescu, M. Fiedler, A. A. Nilsson, Delay performance in ip routers, in: 2nd International Working Conference (HET-NETs' 04), 2004.

[11] N. Hohn, K. Papagiannaki, D. Veitch, Capturing router congestion and delay, IEEE/ACM Transactions on Networking 17 (2009) 789–802. doi:10.1109/tnet.2008.927258.

[12] R. Ramaswamy, N. Weng, T. Wolf, Characterizing network processing delay, in: IEEE Global Telecommunications Conference, 2004. GLOBECOM '04., volume 3, IEEE, 2004, p. 1629–1634. doi:10.1109/glocom.2004.1378257.

[13] R.-H. Hwang, J. Kurose, D. Towsley, The effect of processing delay and qos requirements in high speed networks, in: [Proceedings] IEEE INFOCOM '92: The Conference on Computer Communications, IEEE, 1992, p. 160–169 vol.1. doi:10.1109/infcom.1992.263550.

[14] M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, A. Cabellos-Aparicio, RouteNet-Fermi: Network Modeling with Graph Neural Networks, arXiv preprint arXiv:2212.12070 (2022). doi:10.48550/arXiv.2212.12070.

[15] M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, A. Cabellos-Aparicio, Routenet-fermi: Network modeling with graph neural networks (2022). doi:10.48550/ARXIV.2212.12070.

[16] A. Varga, The OMNeT++ discrete event simulation system, in: Proceedings of the European Simulation Multiconference (ESM'2001), SCS - European Publishing House, 2001, pp. 319–324. URL: https://omnetpp.org.

[17] M. Hofmann, On the complexity of parameter calibration in simulation models, The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 2 (2005) 217–226. doi:10.1177/154851290500200405.

[18] G. Simon, P. Volgyesi, M. Maroti, A. Ledeczi, Simulation-based optimization of communication protocols for large-scale wireless sensor networks, in: 2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652), volume 3, Big Sky, MT, USA, 2003, pp. 3_1339–3_1346. doi:10.1109/AERO.2003.1235250.

[19] J. McDonald, M. Horzela, F. Suter, H. Casanova, Automated calibration of parallel and distributed computing simulators: A case study, in: 2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), volume 2, IEEE, 2024, p. 1026–1035. doi:10.1109/ipdpsw63119.2024.00173.

[20] E. E. Ogheneovo, Modeling network router, switches and security using cisco and opnet simulation software, IOSR Journal of Engineering 4 (2014) 44–50. doi:10.9790/3021-04734450.

[21] M. Beshley, N. Kryvinska, H. Beshley, O. Yaremko, J. Pyrih, Virtual router design and modeling for future networks with qos guarantees, Electronics 10 (2021) 1139. doi:10.3390/electronics10101139.

[22] T. Bonald, J. W. Roberts, Internet and the erlang formula, ACM SIGCOMM Computer Communication Review 42 (2012) 23–30. doi:10.1145/2096149.2096153.

[23] A. Flammini, E. Sisinni, F. Tramarin, Ieee 802.11s performance assessment: From simulations to real-world experiments, in: 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), volume 14, IEEE, 2017, p. 1–6. doi:10.1109/i2mtc.2017.7969752.