AN INTELLIGENT TOKEN BUCKET-BASED QUEUE MANAGEMENT STRATEGY FOR QOS OF NAMED-DATA NETWORKING

by

Eric Binnendyk

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Computer Science

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

New Mexico Institute of Mining and Technology
Socorro, New Mexico
May 2022

This thesis is dedicated to my advisor, professor Jun Zheng.

Eric Binnendyk
New Mexico Institute of Mining and Technology
May 2022

ACKNOWLEDGMENTS

This dissertation was typeset with LaTeX[1] by the author.

[1]The LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for the New Mexico Institute of Mining and Technology dissertation format was written by John W. Shipman.

ABSTRACT

Named data networking (NDN) is a representative architecture for information-centric networking (ICN), which has been shown as a promising solution for supporting the communication needs of a variety of smart systems such as smart grids, smart cities, and smart homes. Compared with traditional host-centric IP-based networking, NDN is based on the content-centric communication model where the data objects are retrieved by names instead of delivered to a destination address. One of the major challenges faced by NDN is how to provide differential quality of service (QoS) to network traffics with different priority levels. In this report, we introduce a new intelligent queuing strategy for QoS based on a token bucket strategy. The token bucket rates for queues with different priority levels are adaptively adjusted with a piecewise linear loss function according to the traffic condition. Simulations are performed by using the ndnSIM simulator. The results show that the proposed strategy has better performance than the strategies based on fixed token bucket rates.

Keywords: Named data networking (NDN), Quality of service (QoS), Delay, Throughput, Token bucket, Queue management

CONTENTS

LIST OF TABLES

LIST OF FIGURES

This dissertation is accepted on behalf of the faculty of the Institute by the following committee:

Jun Zheng

Academic and Research Advisor

Sihua Shao

Hamdy Soliman

I release this document to the New Mexico Institute of Mining and Technology.

Eric Binnendyk May 12, 2022

CHAPTER 1

INTRODUCTION

The Internet was originally built atop phone networks, and as such the networking protocols were **host-centric** instead of **data-centric**: requests consisted of *where* data was located rather than *what* it was. In the early days of the Internet, all links were assumed to be permanent, or at least last long enough that it could be trusted to be secure. In modern times, these are no longer valid assumptions, as a lot of computation is done over mobile networks, and network links are easier to set up. However, the underlying design of IP and TCP protocols have remained the same since the early 90s. Because the Internet was not designed for mobile or insecure data links, the base protocols have proved inadequate as we have had to emulate security and data-centrism at the application layer using URLs and secure protocols such as HTTPS.

The Internet has an **hourglass structure** [1] with TCP/IP at the waist. Nearly all data transferred over the Internet is transferred via TCP/IP. This structure has been very effective for compatibility of devices, and it also allows future Internet architectures to be backwards-compatible (by building them on top of TCP, even if this is not an optimal implementation). [2]

One problem with IP is the fixed number of possible addresses. Recently, the space of 2^{32} IPv4 addresses has proven insufficient, prompting a switch to IPv6. A network architecture that uses *hierarchical* names can fix this problem.

Information-centric networking (ICN) is a recent networking paradigm that has gained much interest as an alternative to conventional IP-based networking, with potential for supporting the communication needs of systems like smart cities and smart grids. For example, a smart grid is an emerging network of internet-connected electronic devices, defined by the DOE as a technology that provides bidirectional communication between power entities, rather than the traditional model of power flow being managed by a central source [3]. “Smart” electrical devices are finding their way into more and more homes, and these devices often have to negotiate prices with the power grid and communicate with each other. Conventional networking paradigms suffer from multiple problems, including overhead, latency, and simplicity at the forwarding level leading to congestion. On the other hand, ICN-based architectures like iCenS [3] and iCASM [4] have shown promising performance for smart grid communications.

One of the important goals of current ICN research is to provide **Quality of Service** (QoS), which is the practice of providing differential treatment to data with different priority levels. QoS is normally measured in terms of packet forwarding delay, round-trip delay, and packet loss rate. One major factor that affects QoS is the behavior of queues on each node: how the queue drops packets as it fills up. The effect of queuing strategies on the overall quality of network performance can be quite complex. In this study, we develop an intelligent token bucket-based queue management strategy for the QoS of a representative ICN architecture called named data networking (NDN). The strategy involves two "token buckets" with token rates that limit the average number of packets of a particular priority that reach a router. The network performance is determined as a function of the number of packets dropped, which in turn can be modeled as a piecewise linear function of the token rates. The strategy then checks the token rates at the boundaries of each linear component to find the ones with the maximum reward.

CHAPTER 2

BACKGROUND AND RELATED WORKS

2.1 Information-centric networking

Information-centric networking (ICN) is a network paradigm defined by using recursive, hierarchical names to identify data, rather than numerical addresses corresponding to the endpoint ID. This sets ICN, a **data-centric** paradigm, apart from traditional **host-centric** systems including IP. One benefit of the ICN model over IP is that IP routing protocols can only use one path (the best path) to deliver data between a given host and client, and as a result the "best" path may become overly congested and delay network performance [4].

The fact that the meaning or context of a piece of data can be inferred by the forwarders has deep implications, for one allowing for "smart" forwarding strategies or more secure transfer of data. One of the challenges in ICN is to use "smart" forwarding and queuing strategies to provide differential QoS to different packet priorities. They can also be used to solve the IP congestion problem mentioned above. ICN also allows content to be cached closer to the users compared to conventional networks [3]. As a result, ICN-based architectures are promising solutions for large scale smart systems like smart cities and smart grids. Some recent architectures that have been developed under the ICN paradigm include DONA [5], PURSUIT [6], and NDN, to be described below.

2.1.1 Named-data networking

Named-data networking (NDN) is a representative ICN architecture, first introduced by Zhang et al. in 2014 [1]. The goal of NDN is to make the Internet easier to use as a *data-centric network*, its most common use today, rather than a *host-centric network* as it was originally designed [1].

In NDN, communication happens via the exchange of two types of packets: *Interests* (requests) and *Data* (responses). There are three types of nodes: *consumers* (producing Interests), *producers* (producing Data), and *routers*. A consumer node sends out an Interest requesting the name of a piece of data, and obtains a Data packet in response. In some sense, NDN is a "blank slate" networking paradigm, because it is new enough that standard network protocols have not yet been developed for it.

Each router contains a **Forwarding Information Base** (FIB), a **Content Store** (CS), and a **Pending Interest Table** (PIT).

- The FIB contains prefixes of Interests and the outgoing interfaces to send the Interests. Each Interest will be sent through the FIB if it doesn't get matched by the CS or PIT.
- The CS contains cached pieces of Data matched to their corresponding Interests. When an Interest arrives, it is first checked whether it can be matched with Data from the CS before it is forwarded outward.
- The PIT contains a list of Interests that are waiting for responses: their name, the incoming interface, and the outgoing interface, on which to send back each incoming piece of Data, so that it satisfies the Interests that requested it. If a new Interest already matches a pending Interest in the PIT, the PIT entry is just updated with the new data.

Names Each Data packet in NDN is identified by a *name*. This name is a *uniform resource identifier* (URI), a hierarchical structure similar to a URL (example: /NMT/registrar/catalogs/2021-2022). The recursive structure of the URI has the benefit of grouping together similar content under the name prefix. Because the semantics of the name are opaque to the routers, a service can choose any available name it wants [1]. At the same time, the names are more human-readable than IP addresses and can be given meanings in context by custom "smart routing" applications. Also, because the names can be arbitrarily long, they are more scalable than the flat names of IP.

NDN doesn't have a transport layer separate from the routing layer, because all of the transportation information (the equivalent of port number) is contained within the hierarchical names.

Interests can have *selectors*, which are patterns determining how the requested data name matches the Interest name. An example of a pattern is *leftmost child matching*. [1]

Applications of NDN Named-data networking is feasible for webpage loading, real-time video streaming, chatrooms, conferencing [7], smart meter data, sensor data, and other things. Protocols built atop NDN specify standard formats for Interests and Data. Protocols also require extra software for constructing names, forwarding according to a specific strategy, and routing. NDN is compatible with common routing algorithms such as link state and distance vector [1].

2.2 Active queue management

Active queue management (AQM) is a term for intelligent strategies of queue management — deciding whether the queue on a forwarder should queue a packet or drop it. This problem is challenging because it involves balancing the goals of low latency, low drop rate, and reliability. Traditional queue management algorithms drop large sets of packets at once when the queue is full, leading to bad reliability and inconsistency in latency times. In contrast, AQM provides a more gradual droppage of packets if the network is congested [8].

One of the first widely used AQM strategies was Random Early Detection (RED) [9]. RED involves two thresholds th_{min} and th_{max}, and a probability p. If the queue size is between th_{min} and th_{max}, the packet is either marked or dropped with probability p, depending on the transport protocol. If the queue size is less than th_{min}, the packet is queued, and if it is greater than th_{max}, the packet is dropped.

Another AQM strategy is Controlled Delay (CoDel) introduced by Kathleen Nichols and Van Jacobson [10]. CoDel focuses on determining the length of time after a packet is enqueued before it should be dropped. This calculation involves the packet-sojourn time, which is the average amount of time a packet waits before it is processed. The drop time is set using the following equation:

$$t_{new} = t_{curr} + \frac{T_{int}}{\sqrt{N_{drops}}}$$

These strategies both use simple equations to determine whether to drop packets.

2.2.1 Weighted fair queuing

In many scenarios it is desirable to split different packets from a queue according to different priorities. A useful algorithm for this is *weighted fair queuing* (WFQ), which is based on the stride scheduling algorithm for process scheduling. In WFQ, each packet is assigned a priority w. A running total is kept for each priority, representing the total amount of traffic of that type scaled by the priority level. Each time a packet is dequeued, $1/w$ is added to the running total. The packet type with the lowest running total (that has a nonempty queue) is chosen on each time step.

2.2.2 Token bucket strategy

The token bucket strategy for fair queuing was first introduced by Kidambi et al. in 1999 to solve the problem of fair network allocation [11]. It is used by

James et al. in the iCAAP network architecture [12]. It provides QoS for three priorities of traffic. There are three “token buckets,” one for each priority, and tokens drip into each bucket at a constant rate until the bucket is full. Furthermore, if multiple priority queues are occupied, a weighted fair queuing (WFQ) strategy is used to select one queue to be dequeued. The parameters of the strategy are the rate, the size of each bucket, and the weight associated with each priority. This allows for the long term rate of packets of each priority to be limited, while allowing for short term bursts of high packet delivery. Our work is based upon the token bucket strategy, but instead of having fixed token rates, we allow the rates to change over time for better performance.

Similar token bucket strategies have also been used for detecting and mitigating network attacks. One of the four NDN strategies presented by Afanasyev et al. uses a token bucket to mitigate interest flooding attacks (IFAs) [13].

CHAPTER 3

PROPOSED METHOD

3.1 Intelligent Token Bucket-Based Queue Management Strategy

In the proposed strategy, we consider the traffics in the NDN have two priority levels: high and low. Similar to [12], our token bucket strategy uses WFQ to select a packet from one of the priority queues.

Our strategy simply seeks out the optimal performance by mathematically modeling the reward as a function of token rates and finding the maximum of the function. Table 3.1 shows the symbols and their corresponding descriptions used in this report.

Symbol	Description
TR	sum of two token rates
HPP	high priority packet arrival rate
LPP	low priority packet arrival rate
HPD	high priority drop
LPD	low priority drop
rew	array of four candidates for maximum reward
HPR	array of four candidates for high priority token rate
$TR1$	high priority token rate
$TR2$	low priority token rate

Table 3.1: Symbols and their descriptions used in this report

The rationale behind this method is that a reward function which is a linear combination of high and low priority loss rates is a *piecewise linear* function of token rates, meaning it can be defined as a partial function where each piece is linear. This means that the optimal value must occur at the boundary of a linear region.

We used the following linear combination as our reward function:

$$reward(TR1, TR2) = 2/3 \cdot \min(1, TR1/HPP) + 1/3 \cdot \min(1, TR2/LPP) \quad (3.1)$$

To approximate the reward for the next time step, we use the token rates and packet rates from the previous time step. This works based on two assumptions: the traffic rate is almost constant between two observation periods, and the sum $TR = TR1 + TR2$ is constant even when the optimal token rates change. The first one of these assumptions is satisfied if the time between observations of traffic rate is small. The second assumption is justified by the fact that setting $TR1 + TR2$ to *less* than the link bandwidth is inefficient, while setting it to *more* than the link bandwidth causes the token bucket to fail to mitigate packets when the network is congested.

Assuming the packet rates are constant, this is a piecewise linear function of $TR1$ and $TR2$ because the *min* function is piecewise linear. Furthermore, assuming the sum $TR1 + TR2$ is constant, this can be expressed as a function of $TR1$ only, so there is only one degree of freedom for the values.

As mentioned above, the optimal value must occur at an endpoint of one of the linear regions. The four values of $TR1$ corresponding to the endpoints are 0, HPP, $TR - LPP$, and TR.

Each priority token rate produces a long-term cap for the number of interests per second of that priority. The value TR is a cap for the total bandwidth passing through the priority queues, and so it makes sense to set it to the maximum rate of the link. This gives us a rough idea of the correct values for the token rates if we know the traffic conditions.

3.1.1 Queue Management Algorithm

The queuing strategy is installed on all router nodes. An overview of the algorithm is shown in Figure 3.1.

First, the values HPR and LPR are measured using the traffic rates from the last second. All four endpoint values of $TR1$ (and corresponding values of $TR2$ satisfying $TR1 + TR2 = TR$) are measured, and the values that lead to maximum reward become the new default token rates.

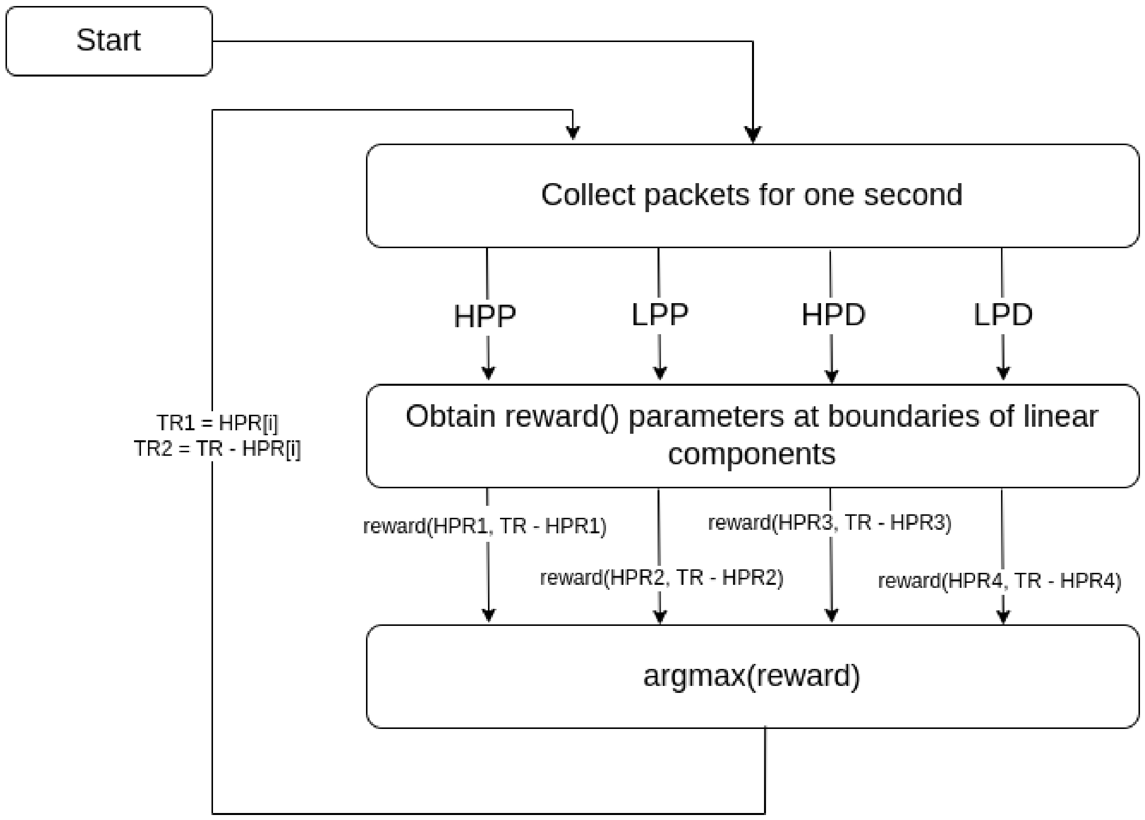

Figure 3.1: Overview of queue management algorithm

Algorithm 1 Function to find optimal token rates based on linear optimization

```
function OPTIMIZE(TR, HPP, LPP, HPD, LPD)
    $HPR \leftarrow [0, HPP, TR - LPP, TR]$
    $rew \leftarrow [0, 0, 0, 0]$
    for $rew[i]$ in $rew$ do
        $rew[i] \leftarrow reward(HPR[i], TR - HPR[i])$
    end for
    $m \leftarrow max(rew)$
    for $HPR[i]$ in $HPR$ do
        if $rew[i] = m$ then return $HPR[i]$
        end if
    end for
end function
```

Algorithm 2 Main algorithm to set token rates

```
$TR \leftarrow TR1 + TR2$
while running do
    Find $HPP$: total high priority packets arriving in previous second
    Find $LPP$: total low priority packets arriving in previous second
    Find $HPD$: total high priority packets dropped in previous second
    Find $LPD$: total low priority packets dropped in previous second
    $new\_TR1 \leftarrow optimize(TR, HPP, LPP, HPD, LPD)$
    $TR1 \leftarrow new\_TR1$
    $TR2 \leftarrow TR - new\_TR1$
end while
```

CHAPTER 4

PERFORMANCE EVALUATION

We tested our queuing strategy using the ndnSIM simulator for NDN [14], which is built atop the ns-3 network simulator. Our code was developed based on the codebase of the token bucket strategy and simulations (`https://github.com/nsol-nmsu/ndnQoS`). We tested the strategy on a network topology consisting of one high-priority sender, one low-priority sender, one receiver, and two routers, as shown in figure 4.1. Each token bucket initially consisted of 2000 tokens. Each link had a delay of 10 ms and a throughput of 10 Mbps, except the middle one which had a throughput of 1 Mbps. The base ns-3 queue had a capacity of 10 and the other queues had a capacity of 20. The optimal value of TR was estimated to be 110 based on the size of data packets and the bottleneck link rate.

The metrics we calculated were round-trip time and throughput rate. We compared the performance of the proposed scheme to those of the strategies based on fixed token rates.

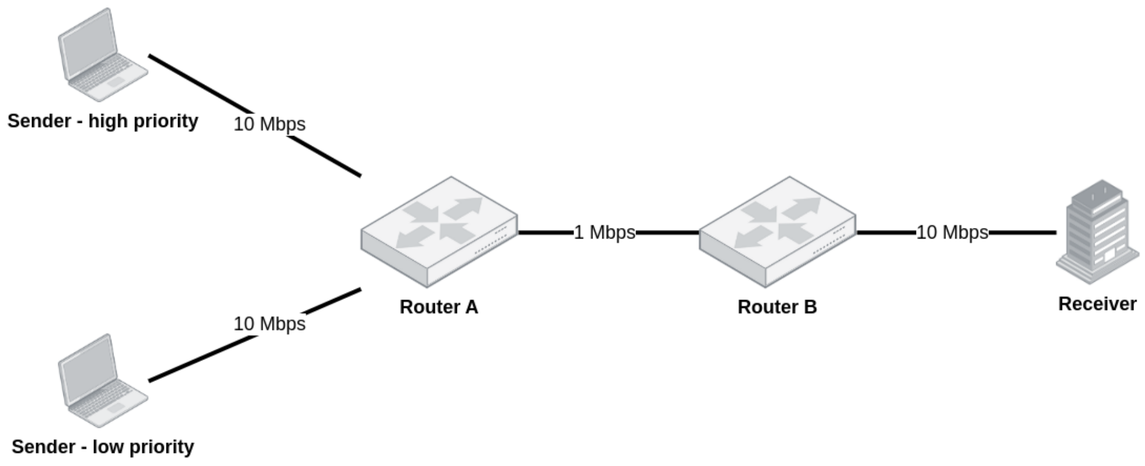

Figure 4.1: Network topology

4.1 Tests with constant traffic rates

4.1.1 Test 1

In this simulation we have the following traffic conditions:

- 70 high priority packets per second on sender 1
- 80 low priority packets per second on sender 2

The results of this simulation are shown in Figures 4.2 and 4.3. These results show that the proposed strategy achieves better QoS compared with the strategies with fixed token rates. Compared with the strategies with fixed token rate (50, 60) and (20, 90), the proposed strategy obtains a significantly lower round-trip delay for high priority traffic with a reasonable round-trip delay for low priority traffic. Although the round-trip delay for high priority traffic of the proposed strategy is slightly higher than that of the strategy with fixed token rate (80, 30), the proposed strategy has a significantly lower round-trip delay for low priority traffic. Similar trends can be observed from the throughput results.

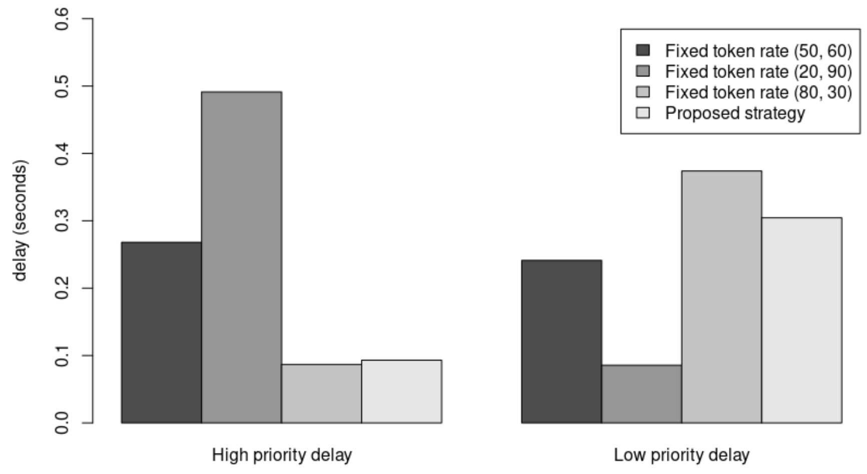

Figure 4.2: Round-trip delays for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under constant traffic rates (70, 80).

4.1.2 Test 2

In this simulation we have the following traffic conditions:

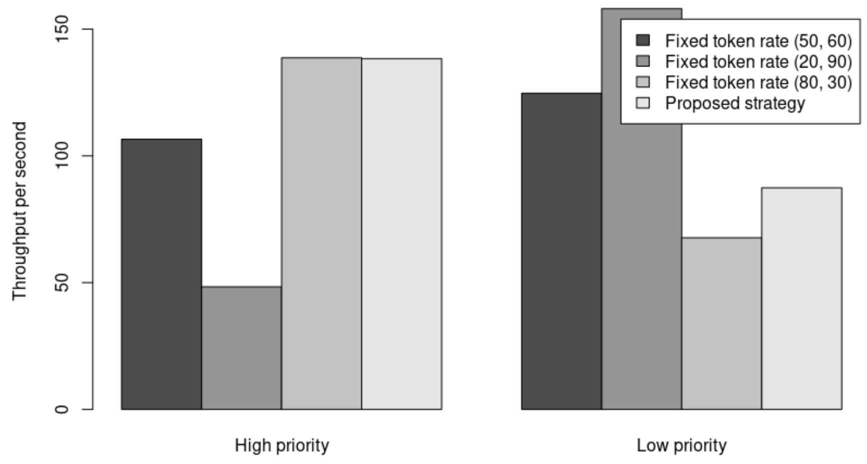

Figure 4.3: Throughputs of priority queues on router A for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under constant traffic rates (70, 80).

- 30 high priority packets per second on sender 1
- 120 low priority packets per second on sender 2

The results of this simulation are shown in Figures 4.4 and 4.5. It can be observed that the results are similar to those of test 1. Compared with the strategies with fixed token rates, the proposed strategy has the lowest round-trip delay for low priority traffic with a slightly higher round-trip delay for high priority traffic than the two strategies with fixed token rates (50, 60) and (80, 30), both setting a high token rate for high priority traffic. Figure 4.5 show that the proposed strategy significantly improves the throughput of low priority traffic without compromising the throughput of high priority traffic significantly.

4.2 Tests with variable traffic rates

4.2.1 Test 3

In this simulation we have the following traffic conditions:

- 30 high priority packets per second up to 100 seconds
- 70 high priority packets per second from 100 seconds onwards

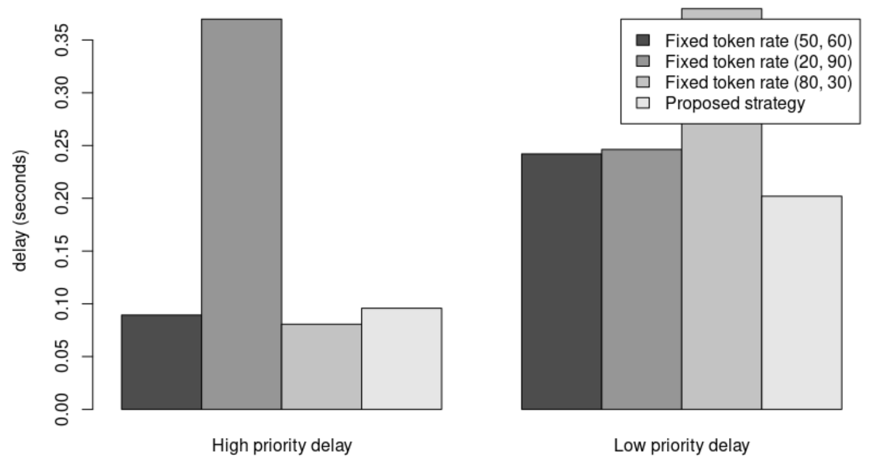

Figure 4.4: Round-trip delays for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under constant traffic rates (30, 120).

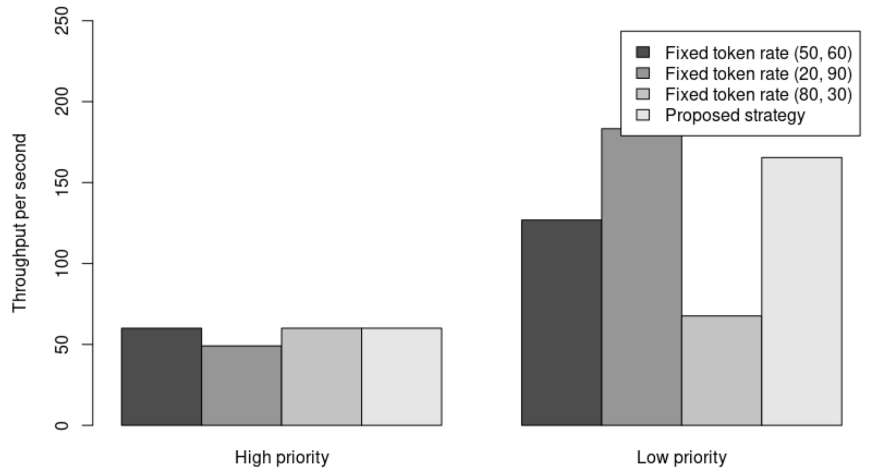

Figure 4.5: Throughputs of priority queues on router A for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under constant traffic rates (30, 120).

- 100 low priority packets per second

The results of this simulation are shown in figures 4.6 and 4.7. This test shows that the proposed strategy obviously achieves better QoS in terms of delay and throughput for traffics of both priorities than the strategies with fixed token rates. Figure 4.7 show that the proposed strategy gives a close-to highest throughput for the high priority traffic with a good throughput for the low priority traffic.

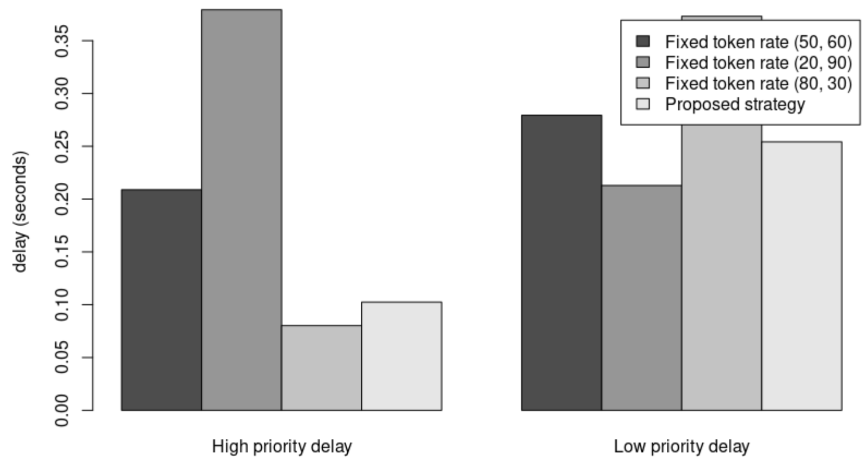

Figure 4.6: Round-trip delays for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under jumping traffic rates $(30 \rightarrow 70, 100)$.

4.2.2 Test 4

In this simulation we have the following traffic conditions:

- 70 high priority packets per second up to 100 seconds
- 30 high priority packets per second from 100 seconds onwards
- 100 low priority packets per second

The results of this simulation are shown in figures 4.8 and 4.9. We can see that the results are similar to test 3 showing that the proposed strategy achieves a better QoS than the strategies with fixed token rates.

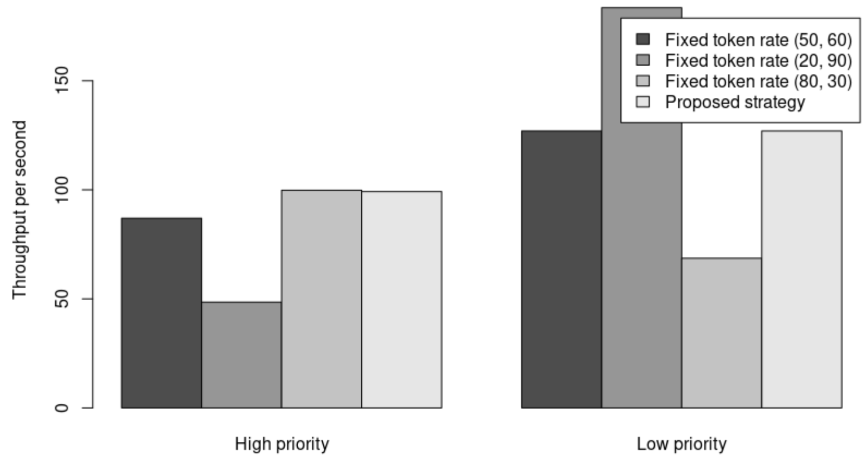

Figure 4.7: Throughputs of priority queues on router A for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under jumping traffic rates $(30 \to 70, 100)$.

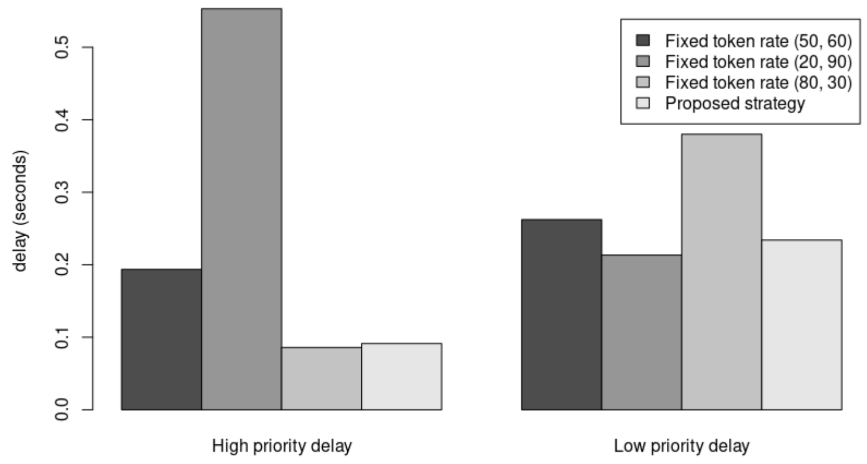

Figure 4.8: Round-trip delays for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under jumping traffic rates $(70 \to 30, 100)$.

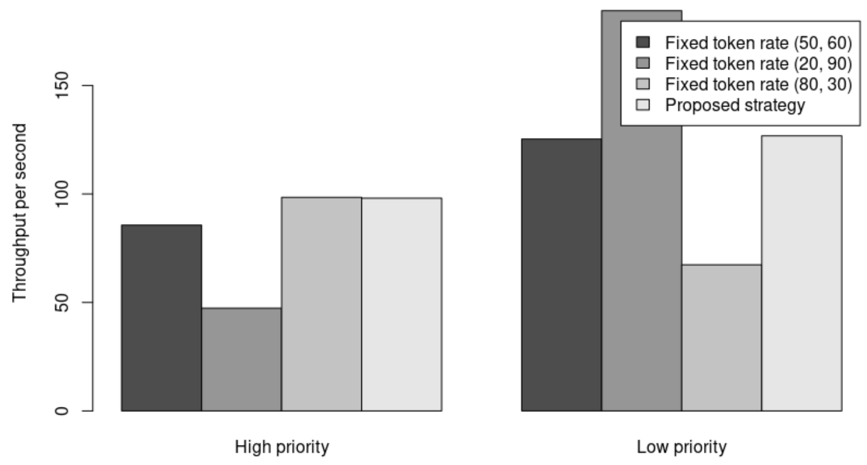

Figure 4.9: Total throughputs of priority queues on router A for strategies with fixed token rates (50, 60), (20, 90), and (80, 30) vs. proposed strategy under jumping traffic rates $(70 \rightarrow 30, 100)$.

CHAPTER 5

CONCLUSION

In this report, we propose an intelligent token bucket-based queue management strategy for QoS of NDN. Through the simulations performed with the ndnSIM simulator, we demonstrate the proposed strategy can achieve a better QoS for traffics with different priorities in terms of delay and throughput than the strategies with fixed token rates.

There are several future research directions to further improve the proposed strategy. One direction is to use reinforcement learning to come up with a strategy that optimizes both packet loss and delay. Alternatively, we could use a hybrid model that uses the analytical strategy to come up with a "starting guess" and uses reinforcement learning to get better token rates.

It would also be worthwhile to research what happens when the token bucket is set to be empty initially. In the simulations in our tests, the token bucket started out full. If the network was congested, this caused an initial period of time when packets were dropped only by the base ns-3 queue as the token buckets were emptying.

Another future research direction is for better packet classification. Currently, packets are classified based on their name identifier. Packets can also be classified using A.I. into a hierarchy of priorities, with feedback from endpoints to determine if the priority is correct. This improvement could help mitigate network attacks, including interest flooding attacks (IFAs), the type of attacks against which NDN is the most vulnerable.

REFERENCES

[1] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, jul 2014.

[2] Eirik Ketilsønn Kjevik. Active queue management for window based applications in named data networking. Master's thesis, Univ. of Oslo, 2019.

[3] Reza Tourani, Satyajayant Misra, Travis Mick, Sukumar Brahma, Milan Biswal, and Dan Ameme. icens: An information-centric smart grid network architecture. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 417–422, 2016.

[4] Gelli Ravikumar, Dan Ameme, Satyajayant Misra, Sukumar Brahma, and Reza Tourani. icasm: An information-centric network architecture for wide area measurement systems. *IEEE Transactions on Smart Grid*, 11(4):3418–3427, 2020.

[5] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, page 181–192, New York, NY, USA, 2007. Association for Computing Machinery.

[6] Dirk Trossen, Arjuna Sathiaseelan, and Jörg Ott. Towards an information centric network architecture for universal internet access. *SIGCOMM Comput. Commun. Rev.*, 46(1):44–49, jan 2016.

[7] Zhenkai Zhu, Jeffery Burke, Lixia Zhang, Paolo Gasti, Yanbin Lu, and Van Jacobson. A new approach to securing audio conference tools. pages 120–123, 11 2011.

[8] Minsu Kim. *Deep Reinforcement Learning based Active Queue Management for IoT Networks*. PhD thesis, 01 2019.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Commun. ACM*, 55(7):42–50, jul 2012.

[11] J. Kidambi, D. Ghosal, and B. Mukherjee. Dynamic token bucket (dtb): a fair bandwidth allocation algorithm for high-speed networks. In *Proceedings Eight International Conference on Computer Communications and Networks (Cat. No.99EX370)*, pages 24–29, 1999.

[12] Anju K. James, George Torres, Sharad Shrestha, Reza Tourani, and Satyajayant Misra. icaap: information-centric network architecture for application-specific prioritization in smart grid. In *2021 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, 2021.

[13] Alexander Afanasyev, P. Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in named data networking. pages 1–9, 01 2013.

[14] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. On the evolution of ndnsim: an open-source simulator for ndn experimentation. *ACM SIGCOMM Computer Communication Review*, 47:19–33, 09 2017.

AN INTELLIGENT TOKEN BUCKET-BASED QUEUE MANAGEMENT STRATEGY FOR QOS OF NAMED-DATA NETWORKING

by

Eric Binnendyk

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY