

# Santa Clara University

Department of Computer Science and Engineering

Date: June 9, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED  
UNDER MY SUPERVISION BY

**Siena Hanna**

**Justin Lee**

**Tania Pham**

ENTITLED

**NetCon: Dynamic Resource Allocation to Containers Running on  
Network Switching Appliances**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING**

DocuSigned by:  
  
6ED175D6B91A4FF...  
Thesis Advisor  
Dr. Behnam Dezfouli

  
N. Ling (Jun 9, 2022 14:20 PDT)  
Chairman of Department  
Dr. Nam Ling

# **NetCon: Dynamic Resource Allocation to Containers Running on Network Switching Appliances**

By

Siena Hanna

Justin Lee

Tania Pham

Submitted in Partial Fulfillment of the Requirements  
for the Degree of Bachelor of Science  
in Computer Science and Engineering  
in the School of Engineering at  
Santa Clara University,

June 9, 2022  
Santa Clara, California

---

# Acknowledgments

We would like to thank our advisors Dr. Behnam Dezfouli and Mr. Ted Kummert for all of their help and guidance throughout the past year on this project. We would also like to thank Santa Clara University School of Engineering for providing us funding to complete our project. Finally, we would like to thank our family, friends, and professors for all of their support during our time spent at SCU.

# NetCon: Dynamic Resource Allocation to Containers Running on Network Switching Appliances

Siena Hanna

Justin Lee

Tania Pham

Department of Computer Science and Engineering  
Santa Clara University  
Santa Clara, California

June 9, 2022

## ABSTRACT

A WiFi Access Point (AP) is an important technology that switches data packets to transmit and receive wireless signals to create WiFi. These Access Points (AP) can have powerful processors, but they are idle most of the time [1]. Instead, these idle processing resources could be redirected to run containers for low-latency applications. However, it is essential to ensure that the AP does not suffer performance issues as a result of running containers. The main function of APs is to switch packets, and this function must be preserved while containers are running. In this thesis, we propose a method, *NetCon*, to measure packet switching delay and allow containers to run on APs only when there are available resources and low traffic. This would allow APs to be leveraged as edge devices for low-latency applications while also maintaining their primary function as packet switching devices by ensuring that packet switching delay is not adversely impacted by running containers.

---

# Table of Contents

<b>1</b>	<b>Introduction . . . . .</b>	<b>1</b>
1.1	Motivation and Background . . . . .	1
1.2	Related Work . . . . .	2
1.2.1	Containers . . . . .	2
1.2.2	Container Migration . . . . .	3
1.2.3	Resource Allocation Methods for Containers . . . . .	3
1.2.4	eBPF . . . . .	5
1.3	Existing Solutions . . . . .	5
1.4	Proposed Solution . . . . .	6
1.5	Thesis Organization . . . . .	7
<b>2</b>	<b>Requirements . . . . .</b>	<b>9</b>
2.1	Functional . . . . .	9
2.1.1	LAN Connectivity . . . . .	9
2.1.2	Applications and Containers . . . . .	9
2.1.3	Container Resource Allocation and Container Migration . . . . .	10
2.2	Non-Functional . . . . .	10
2.2.1	Performance . . . . .	10
2.2.2	Security . . . . .	10
2.2.3	Reliability . . . . .	11
2.3	Design Constraints . . . . .	11
2.3.1	Accurate and Dynamic Packet Switching Delay Measurements . . . . .	11
2.3.2	Limited Network Capability . . . . .	11
2.4	Summary . . . . .	12

<b>3</b>	<b>System Architecture</b>	<b>13</b>
3.1	Linux Machine	13
3.2	Hostapd/NetPlan: Using a Linux Machine as an AP	13
3.3	eBPF and Python: Measuring Packet Switching Delay	14
3.4	Load Balancing: Using Packet switching delay	16
3.5	LSTM: Predicting Traffic Trends	16
3.6	Docker: Allocating and Reallocating Container Resources	17
3.7	Kubernetes: Orchestrating Container Deployment and Migration	17
3.8	Iperf3: Testing	18
3.9	Summary	18
<b>4</b>	<b>Design and Implementation</b>	<b>19</b>
4.1	Design Rationale	19
4.1.1	Docker and Kubernetes	19
4.1.2	Packet Switching Delay	19
4.1.3	Programming	20
4.2	Implementation	20
4.3	Summary	22
<b>5</b>	<b>Evaluation</b>	<b>23</b>
5.1	Packet Size and Arrival Frequency	23
5.2	Packet Switching Delay vs. Container Resources	25
5.2.1	CPUs Assigned	26
5.2.2	Swap Memory and Memory	27
5.3	Summary	29
<b>6</b>	<b>Risk Analysis</b>	<b>30</b>
<b>7</b>	<b>Existing Issues</b>	<b>32</b>
7.1	Ethical Issues	32
7.2	Societal Issues	33
7.3	Political Issues	33

7.4	Economic Issues . . . . .	33
7.5	Environmental Issues . . . . .	34
7.6	Usability Issues . . . . .	34
7.7	Sustainability Issues . . . . .	34
7.8	Health and Safety Issues . . . . .	35
7.9	Manufacturing Issues . . . . .	35
7.10	Summary . . . . .	35
<b>8</b>	<b>Conclusion and Future Work . . . . .</b>	<b>37</b>
8.1	Conclusion . . . . .	37
8.2	Future Work . . . . .	37
8.2.1	Energy Efficiency . . . . .	38
8.2.2	LSTM . . . . .	38
8.2.3	Generalization . . . . .	38
	<b>Bibliography . . . . .</b>	<b>39</b>

---

# List of Figures

1.1	Conceptual model. . . . .	7
3.1	Measuring Packet switching delay. . . . .	14
4.1	Phases of Container Allocation. . . . .	20
4.2	Internal Activity Diagram. . . . .	21
5.1	Packet Switching Delay for Packets of Sizes 200-2k Bytes. . . . .	24
5.2	Packet Switching Delay vs. Number of Packets Delivered. . . . .	25
5.3	Packet Switching Delay vs CPUs Assigned to Containers. . . . .	26
5.4	Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=1 GB). . . . .	27
5.5	Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=2 GB). . . . .	28
5.6	Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=3 GB). . . . .	29



---

# List of Tables

6.1 Risk Analysis Table. . . . . 30

---

# List of Abbreviations

AP	WiFi Access Point
CPU	Central Processing Unit
CRIU	Checkpoint Restore in Userspace
eBPF	extended Berkeley Packet Filter
Hostapd	Host Access Point Daemon
LAN	Local Area Network
LSTM	Long Short-Term Memory
LXD	Linux Container Hypervisor
NAT	Network Address Translation
NIC	Network Interface Card
OS	Operating System
QoS	Quality of Service
VM	Virtual Machine
VNF	Virtual Network Functions
WLAN	Wireless Local Area Network
WPA	WiFi Protected Access

---

# CHAPTER 1

## Introduction

This chapter provides an introduction to the primary motivations behind this thesis and the problem that it is intended to solve, the related work in this topic and its limitations, and a description of the proposed solution.

### 1.1 Motivation and Background

WiFi Access Points (AP) are wireless devices that create a wireless local area network (WLAN) [2]. They are connected to a router that switches data packets to transmit and receive a wireless signal to create WiFi. However, this rarely requires all of their processing resources, and, outside of high-traffic times, the device is mostly left idle [2]. Since there is significant idle time on these devices, it is possible to utilize them to run containers for applications that require low latency. AP represent a significantly underutilized edge device for edge computing.

AP and network switches primarily perform packet switching. Packet switching delay is an important performance metric for AP and other network switching appliances, but it is not necessarily relevant to many other edge devices that might be running containers [3]. Packet switching, however, rarely requires all of an AP's processing resources, and there is idle time that could be filled with additional functionality [3]. In that time, it could be beneficial to run containers on the AP in order to decrease latency from the cloud, which could be particularly important

for applications that require consistent low latency. However, packet switching remains the essential operation of APs and network switches, and any additional functionality cannot be allowed to jeopardize the primary function of these devices [3]. Therefore, it may be beneficial to explicitly take packet switching performance into account when determining resource allocation to containers running on APs.

## **1.2 Related Work**

This section provides an introduction to current works on containers, container migration, resource allocation for containers, and extended Berkeley Packet Filtering (eBPF).

### **1.2.1 Containers**

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another [4]. A container helps solve the issue of software reliability when trying to move from one environment to another. Containers use a form of Operating System (OS) virtualization where the features of the OS can isolate processes and manage the amount of Central Processing Units (CPUs), memory, and disk that the processes need access to [4]. Containers run in isolated user environments which means that containers require less resources to maintain, are more efficient, faster to start up, are portable, and have less overhead [4].

### 1.2.2 Container Migration

Container migration is the process of moving applications between different physical machines or clouds without disconnecting the client [5]. In container migration, there is the source node which is the origin location of the container and the destination node which is the final desired location of the container [5]. Container migration is necessary because APs have a limited amount of resource, thus migration can open up space to allocate more resources [6]. NetCon will be using Docker and Kubernetes to deploy and migrate containers on AP.

### 1.2.3 Resource Allocation Methods for Containers

Overall, current dynamic resource allocation methods for containers rarely consider the specific challenges of running containers on AP or reallocating the processing resources of a single edge device.

In one method, containers are assigned to edge devices based on edge device availability [7]. This method of resource allocation treats the edge devices themselves as the resources being allocated; once the containers are running within those devices, the continued influence of the allocation scheme is limited, and it does not appear to dynamically re-allocate processing resources on those individual edge devices once a container has been assigned [7]. Instead, when resources become scarce, the container is moved to another available device based on the lowest migration cost. NetCon focuses specifically on the performance of the AP in terms of packet switching rather than simply making decisions based on the processing resources available.

There is also a method for network resource allocation for containers, even briefly addressing those that run on APs [8]. However, this method focuses on

managing the bandwidth usage of each container rather than the processor usage, and it does not appear to include the influence of non-container processes running on the specific edge device [8]. In contrast, the proposed solution of NetCon hinges upon utilizing the AP for containers while avoiding an overly negative impact on packet switching delay specifically, using this delay as an indicator of performance in order to determine the effect of containers on the AP's performance.

Specific resource allocation on containers have the issue of overhead to consider. Only under specific circumstances, containers may have more overhead than Virtual Machines (VM) [9]. However, generally, containers with a larger number of cores will have less overhead than containers with a smaller number of cores [9]. CPU-intensive applications are better used off with pinned container resources, where resources are specifically assigned to cores.

Energy-saving methods that have been used in VMs are also applicable to energy-saving methods with containers [10]. Excessive idle resources are checked and then reassigned to hosts with more VMs so that VMs who have little resources are shut down [10]. With the way containers work better with a larger number of cores, this can be checked and reallocated as well.

Furthermore, this thesis proposes to use packet switching delay as the measure of Quality of Service (QoS) for the AP. Since this is the primary responsibility of the AP itself, focusing on this measurement emphasizes the specificity of the proposed solution. The effects of containers on the AP and the availability of the AP to run containers will be determined not by the processing load on the device or the network conditions but by the packet switching delay and its change over time. When processing resources are low, or when network traffic increases, the AP may become slower at packet switching, and these times or packet switching delay increase should indicate that the AP is not available to run containers. When

packet switching delay is low, and when containers are sufficiently small to avoid adverse effects on packet switching, the AP should be available to run container and decrease idleness.

#### **1.2.4 eBPF**

Extended Berkeley Packet Filter (eBPF) allows user programs to be attached to events occurring in the kernel and is predominantly used for traffic control, network monitoring, load balancing, and monitoring the operating system [11].

Measuring packet switching delay is a less common use, though there are some preexisting methods to determine packet processing time. For example, one of these methods inserts a timestamp into the packet header when the packet arrives and when it is finished being processed, and the processing time is calculated accordingly [12]. However, as this method is specialized for Virtual Network Functions (VNF), it may not apply to use with a single AP.

In this case, eBPF will be used to measure packet switching delay (as the time elapsed between the packet's arrival to the wired interface and delivery to the wireless interface of the AP) in order to observe and react to changes in the QoS of the AP as the load on the AP increases when the resources allocated to containers is increased and when the traffic on the network changes.

### **1.3 Existing Solutions**

While there are several current solutions that use edge servers to run containers and decrease latency, there no widely known way of running containers and decreasing latency using APs. In particular, EdgeAP, a SCU senior design project from last

year, investigated the potential impact of running applications on APs [1]. EdgeAP specifically mentions that finding a way to migrate containers between APs will better support mobile devices [1]. However, EdgeAP did not develop a resource allocation algorithm to dynamically respond to an increase in packet switching activity and take into consideration the unique concerns with running containers on APs [1]. Without this resource allocation algorithm, EdgeAP’s system does not know how to deal with increasing packet switching delay and increasing latency.

Additionally, there exists a container migration algorithm called ShareOn which uses Linux Container Hypervisor (LXD) and Checkpoint Restore in Userspace (CRIU) to keep track of traffic and decides when to migrate containers based on the resources available at the current edge node [7]. ShareOn, however, only works in Mobile Edge Clouds and is not compatible for APs. Furthermore, their system results showed that it does not work well when encountering large system latency and higher server load [7]. Hence, ShareOn struggles to decrease latency during higher server load.

## 1.4 Proposed Solution

The proposed solution, NetCon, is built to run on APs, providing a way to monitor APs traffic and packet switching delay to determine when APs are able to run containers without interfering with their primary function. When the AP has excess idle time and is sufficiently free, the AP can run containers for different applications. When the AP has higher traffic, and packet switching delay is high, NetCon will recommend that the AP’s resources will be prioritized for packet switching. By allowing container to run on the APs when there is low traffic, NetCon seeks to decrease AP idleness while ensuring that packet switching delay remains low. NetCon



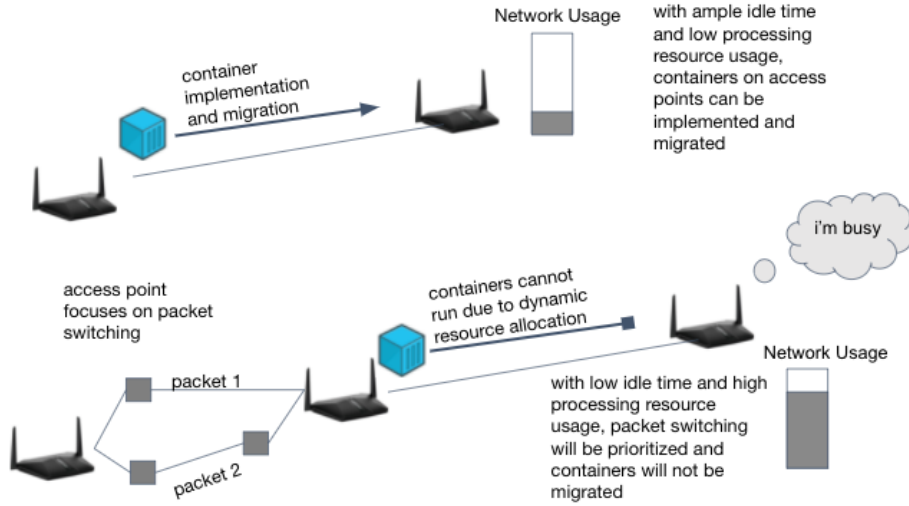


Fig. 1.1: Conceptual model.

dynamically analyzes traffic being switched by the AP and determines if containers can be run. When traffic is high, and running containers adversely affects packet switching delay, container resources should be decreased, and containers should be migrated to other edge devices when the available resources are too low for them to run.

Fig. 1.1 shows a high-level depiction of the concept behind NetCon. While packet switching delay (and traffic) is low, the AP will be available to run containers at will. However, when the packet switching delay is higher, the AP will prioritize packet switching, and containers will have their resources reduced, or they will be migrated to another device.

## 1.5 Thesis Organization

This section presents an overview of this thesis. Chapter 1 focused on the introduction to the problem, related works, and the proposed solution, NetCon. Chapter 2 will discuss the functional requirements, non-functional requirements, and design

constraints for implementation of NetCon. Chapter 3 will address NetCon's system architecture, including the underlying technologies used for its implementation. Chapter 4 will explain the design rationale and implementation strategies to build NetCon. Chapter 5 includes descriptions of testing strategies and the performance evaluation for NetCon. Chapter 6 addresses the risk analysis for this project and the impact that each risk had on the final deliverable. Chapter 7 analyzes the numerous non-technical issues and potential implications of NetCon. Finally, Chapter 8 will provide the conclusion and discussion of potential future work.

---

# CHAPTER 2

## Requirements

This chapter addresses the requirements for the implementation of NetCon. This includes functional requirements, non-functional requirements, and design constraints that needed to be considered for the creation of NetCon.

### 2.1 Functional

Functional requirements refer to functions within the system which specifies what the system needs to do when certain conditions are met [13]. Functional requirements describe what a system should do and its intended, specific behavior.

#### 2.1.1 LAN Connectivity

The AP must be built and configured correctly so that both wired and wireless connections to the AP are available. The AP must be able to successfully switch packets between the wired and wireless interfaces using a bridge.

#### 2.1.2 Applications and Containers

The AP must have the ability to run applications and containers. Specifically, it must be possible to run Docker containers on the AP itself.

### **2.1.3 Container Resource Allocation and Container Migration**

The AP must be able to allocate and re-allocate resources to and from containers. It must also be able to migrate containers to another edge device if the AP is experiencing high traffic.

## **2.2 Non-Functional**

Non-functional requirements describe how a system should behave and the limits that exist on its functionalities [13]. Non-functional requirements define the attributes and operations of a system.

### **2.2.1 Performance**

As more applications are added onto the AP, the AP's packet switching delay should be impacted as little as possible. Packet switching delay must be continuously monitored to determine if containers are having an adverse effect on the AP's performance of packet switching.

### **2.2.2 Security**

The AP and network must remain secure and not be compromised by NetCon. The system must be secure for containers to run and limit the container's access to only the resources required to function. Furthermore, NetCon should detect if containers are adversely impacting the AP's functionality and prevent the possibility of containers taking up resources to the extent that they cause denial of service.

### **2.2.3 Reliability**

The system must be reliable for containers to run. It is crucial that the network is reliable in order for the system to properly evaluate packet switching delays and system latency.

## **2.3 Design Constraints**

Design constraints refer to the conditions that needs to be satisfied in order for a system to function properly [13].

### **2.3.1 Accurate and Dynamic Packet Switching Delay Measurements**

The AP must be able to accurately and dynamically measure packet switching delay. If the measurements of packet switching delay are not accurate, or if they are not continuously measured, it would be difficult to determine when the AP is experiencing high traffic and when it is available to run containers. Furthermore, without comparing the change in packet switching delay over time, it would not be possible to determine a baseline performance and therefore comparatively determine what defines high and low traffic.

### **2.3.2 Limited Network Capability**

The AP must have the ability to run with limited network capability. Running with limited network capability would prioritize packet switching during high traffic times

and ensure that the original usage of these APs is kept. The AP must remain usable while adding, running, or removing containers.

## **2.4 Summary**

This chapter defined the functional requirements, non-functional requirements, and design constraints that were considered in the design of NetCon. In terms of functional requirements, an AP must be able to switch packets between the wired and wireless interfaces, run containers, and assign resources to those containers. Non-functional requirements require NetCon to ensure the AP performs well and continues to perform well when running containers and remains secure and cannot be denied service due to having resources over-allocated to containers. Finally, in terms of design constraints, NetCon must be able to accurately and dynamically measure packet switching delay measurements, as those measurements are a metric for the AP's performance in its main function.

---

## CHAPTER 3

# System Architecture

This chapter explores the hardware and software technologies used in the implementation of NetCon and how they all fit together.

### 3.1 Linux Machine

NetCon required the use of a Linux machine in order to create the AP. In this particular case, a Kingdel mini-PC with Intel i7 CPU, 8GB RAM, 256GB SSD was used [14]. A Linux-based OS, Ubuntu 20.04.4 [15], was installed on this machine, and then it was configured as an AP.

### 3.2 Hostapd/NetPlan: Using a Linux Machine as an AP

To set up the Linux machine as an AP, two main packages are used, hostapd and Netplan. Hostapd stands for Host Access Point Daemon, a user-space service that turns the network interface into an Access Point instead [16]. In this case, the Linux machine was connected to the Ethernet and configure the hostapd as an AP in order to access the internet. This turns the Linux machine into an AP and an authentication server by assigning IP subnets to link layer addresses. This is

done by configuration of the interface, driver, hardware mode, channel, and WiFi Protected Access (WPA). Netplan is used to create the network configuration with name server addresses, interfaces for bridges, IP addresses, and a gateway to connect to the Ethernet for functionalities [17].

### 3.3 eBPF and Python: Measuring Packet Switching Delay

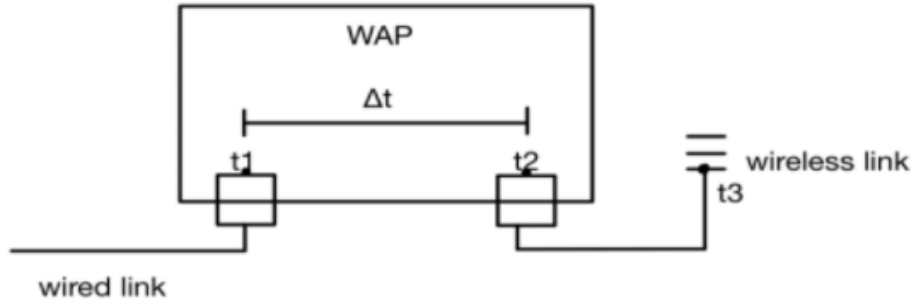


Fig. 3.1: Measuring Packet switching delay.

To monitor packet switching delay, eBPF was used. Using eBPF allowed the attachment of hooks to the kernel system calls associated with packet switching, in order to determine the time between when a packet arrives at the Ethernet interface and when it is ready to be transmitted over WiFi [18]. A Python program was written in conjunction with eBPF in order to measure the packet switching delay as shown in the figure below.

In Fig. 3.1, t1 refers to the time the packet arrives from the wired interface, t2 refers to the time the packet is transferred to the wireless interface for transmission, and t3 refers to the time the packet is actually transmitted. In this case, the



measurement for packet switching delay  $\Delta t$  will be the difference between  $t_2$  and  $t_1$  rather than the difference between  $t_3$  and  $t_1$ , since  $t_3$  depends on the number of devices connected or trying to connect and can be impacted by outside interference. In contrast, as the time elapsed between the packet's transfer from one interface to the other depends mainly on the processor of the AP,  $\Delta t$  is a better measurement of packet switching delay for the purpose of this thesis. Furthermore, this should the removal of the variable of transmission time (which is unlikely to be directly correlated to the resources used by containers running on the AP) in order to better measure the potential correlation between processor load and packet switching delay.

Measuring these times requires the use of eBPF. First, it is necessary to identity the system calls to the kernel being used when 1) the packet arrives from the wired interface and 2) the packet is delivered to the wireless interface for transmission [18]. Furthermore, as it is conceivable that more packets may arrive while earlier packets are being switched, it may also be necessary to keep track of the packets currently in transit in order to ensure that the time between their arrival and delivery is being accurately measured. These two timestamps are correlated between packets by the packet checksum. In order to ensure checksum was a reliable metric for correlating and identifying packets, it was necessary to disable Network Address Translation (NAT). Then, eBPF hooks must be attached to log these system calls with timestamps and compare the times between them [18].

The Network Interface Card (NIC), Qualcomm NIC: NETELY 802.11N Dual Band Mini-PCIE Interface, that was installed in the Linux machine used the ath9k driver [19]. Therefore, the system call for the packet's delivery to the wireless interface used was `ath-9k-tx-complete-buf`. For alternative NICs, it would be necessary to alter the eBPF program to target the proper wireless-interface function associated with that NIC.

### **3.4 Load Balancing: Using Packet switching delay**

Load balancing improves application responsiveness [20]. Through load balancing and resource management, containers can run safely on all network devices [20]. NetCon's load balancing is designed to allow applications to run on APs without impacting any of its functions. Packet switching delay was used as the main metric for AP performance. When times of high traffic and increased packet switching delay are predicted, containers are discontinued at a certain load threshold. However, it is also necessary to take into account the processing resources available on each device. If a container no longer has enough resources to thrive, it will be removed. If a container is doing well but packet switching delay is increasing, it will also be removed. At this point, NetCon does not have a predictive consideration of the impact of container migration before making the decision to migrate: it will be solely based on packet switching and available processing resources.

### **3.5 LSTM: Predicting Traffic Trends**

Originally, a Long Short-Term Memory (LSTM) algorithm was going to be used to predict traffic trends using the eBPF data. The purpose of this would be to use past traffic data to help predict future traffic times [21] so that resources could be allocated in advance of when they are needed. LSTM functions as a network that remembers previous states and can selectively choose past information in order to make predictions about future states [21]. Currently, however, it did not integrate well into the system, so the decisions are based solely off of current packet switching information, not predicted future information.

### **3.6 Docker: Allocating and Reallocating Container Resources**

The container technology that selected to allocate container resources on APs is Docker. Docker is a container run-time environment and is primarily used to create and build software inside containers [22]. Docker allows users to create, allocate, move, and delete containers. To create a container, Docker pulls images from the Docker registry and created a container using that image [22]. Additionally, Docker uses C-groups and user namespaces to allocate resources to containers. Docker also allows users to control how much memory and CPU share should be allocated to a container [22]. Through adjusting memory limit and CPU share, it allows us to reallocate resources that are idle to other containers that require more resources. All of the applications that run on NetCon uses Docker and all of the container images are stored in the Docker Hub registry.

### **3.7 Kubernetes: Orchestrating Container Deployment and Migration**

Kubernetes is a container orchestrator tool for scheduling and automating the deployment, management, and scaling of containerized applications [23]. Kubernetes acts as a complementary technology to Docker in order to complete the entire process of resource allocation and container migration [24]. The benefits of using container orchestration tools include: increased portability, efficiency, and security [23]. As an orchestrator, Kubernetes determines where containers should be located and manages the deployment and migration of containers. Kubernetes can also automat-

ically replicate containers and reschedule failing containers [23]. Within NetCon, Kubernetes is used to determine where containers should be located and to manage the deployment and migration of containers.

### **3.8 Iperf3: Testing**

Iperf3 is a command line tool that measures bandwidth with a variety of settings [25]. In the testing phase, iperf3 will be used to send packets of various sizes and amounts in a constant time frame in order to test the impacts of packet size, packet arrival frequency, and container resource allocation on packet switching delay.

### **3.9 Summary**

This chapter provided a discussion of the different types of technologies that were used to implement NetCon's system. First, NetCon uses a Linux machine like a Kingdel and the prerequisite software packages, such as Hostapd and NetPlan, to create the AP. Then, NetCon uses eBPF to measure packet switching delay. Packet switching delay is the metric that used to determine whether containers will remain in their current state, have resources removed from them, or be migrated elsewhere. Docker and Kubernetes were used to help with container deployment, resource allocation, and migration tasks. Lastly, iperf3 will be important during the testing and evaluation phase.

---

## CHAPTER 4

# Design and Implementation

This chapter presents an overview of NetCon’s design and how aspects of the design were implemented.

### 4.1 Design Rationale

This section details the reasoning behind certain aspects of NetCon’s design.

#### 4.1.1 Docker and Kubernetes

Docker and Kubernetes are commonly used for container images and orchestration respectively. It was important to use well-known container technologies to help maximize the portability and real-world implications of NetCon, so these technologies were selected to help implement NetCon.

#### 4.1.2 Packet Switching Delay

In order to determine the effects of running containers on the AP’s efficiency, packet switching delay was measured from the entry interface (wired) to the exit interface (wireless). In particular, this is something that the AP has control over and occurs internally using the AP’s processing resources. Therefore, it is an aspect of delay that is likely to be most effected by running high-load containers on the AP.

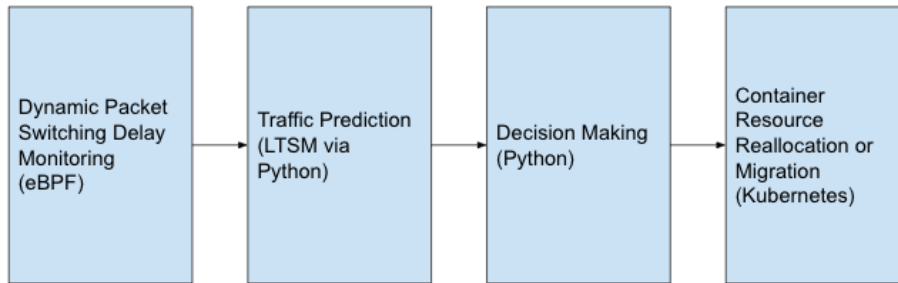


Fig. 4.1: Phases of Container Allocation.

### 4.1.3 Programming

The eBPF program is written in Python and the underlying, contained eBPF program with the hooks is written in C. C made it easier to implement the kernel probes required for the measurement to function, and Python was simpler to execute the eBPF program, process the data, calculate delays, and make predictions due to built-in data processing libraries such as pandas [26].

## 4.2 Implementation

This section is a general overview of the current implementation of NetCon. Fig. 1.1 depicts the main phases of the container resource allocation process. First, packet switching delay is continuously monitored and calculated using the eBPF program. Then, it should be fed into the LSTM model in order to predict future traffic trends. However, currently, LSTM is having implementation issues, so the delay data is instead directly used to make decisions based off of current trends rather than

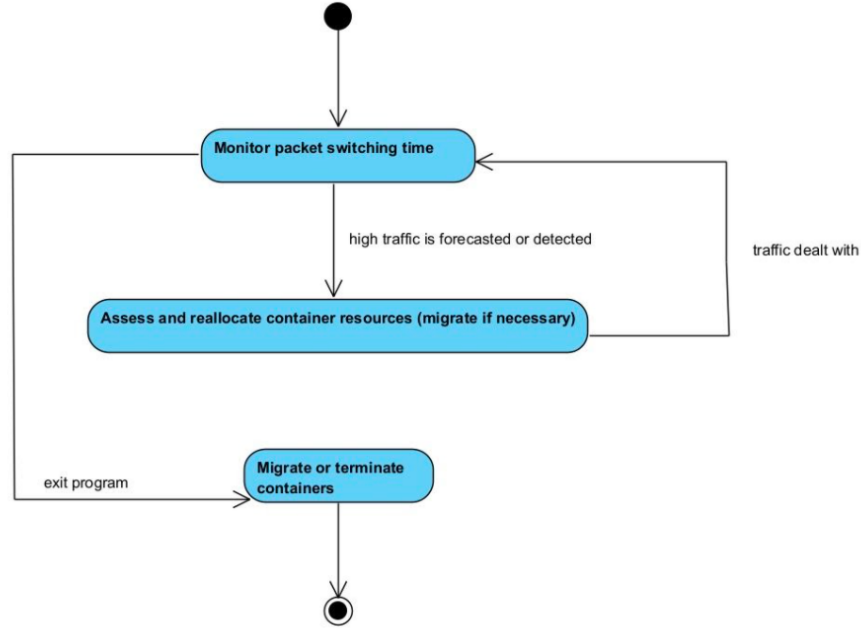


Fig. 4.2: Internal Activity Diagram.

future predicted trends. Based on that data, NetCon will make a recommendation for containers resources to be decreased. If the container is already at its usable resource minimum, it will be migrated away from the AP instead.

Fig. 4.2 shows the major stages that NetCon goes through. Once NetCon is started, it will begin to continuously and passively monitor the packet switching delay. If high traffic is forecasted (with implementation of LSTM) or detected (without implementation of LSTM), NetCon will assess current resources assigned to containers and reallocate resources as necessary to prompt a reduction in packet switching delay. If this is not sufficient to resolve the issue, or if the proposed reallocation of resources would harm a container’s ability to run properly, the container will be migrated instead. After this, NetCon will simply be passively monitoring the packet switching delay. If NetCon is exited, users can either migrate or terminate the containers currently running on the AP, since it would not be prudent to continue running them without considering their impact on packet switching delay.

## 4.3 Summary

NetCon depends on packet switching delay in order to determine whether the AP is available to run containers. Based on these packet switching delay measurements, NetCon can determine whether and when container resources should be decreased in order to ensure that packet switching delay does not suffer on account of resources assigned to containers. Therefore, containers can safely run on APs without compromising the performance of packet switching.



---

# CHAPTER 5

## Evaluation

This chapter provides an overview of the performance evaluation of Netcon’s packet switching delay measurement methods and the effects of packet size and container load on those measurements.

### 5.1 Packet Size and Arrival Frequency

This section displays the effects of packet size and amount of packets being sent on the packet switching delay within the AP. The initial hypothesis was that a greater number of packets being sent in the same timeframe would correlate to greater average packet switching delay. This was due to the assumption that high traffic and a higher frequency of packets arriving at the AP would imply then the packet switching delay is likely to increase due to queuing.

Packet switching delay was measured by conducting controlled network testing with iperf3. In each test, 1.25 Mb of data in 10 seconds were transferred from a public iperf-compatible server to a client device connected to the AP while the AP measured the packet switching delay. The packet size was adjusted for each test, starting at 200 bytes and increasing in increments of 200 bytes up to 2 kilobytes. During these initial tests, no containers were running. The purpose of these initial tests were merely to determine if the packet switching delay would show correspondence to different levels of traffic in terms of handling different sizes and amounts

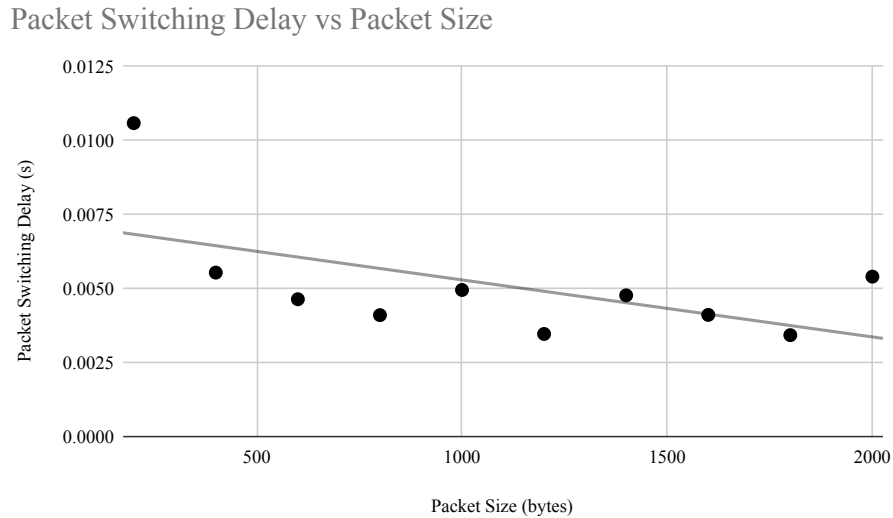


Fig. 5.1: Packet Switching Delay for Packets of Sizes 200-2k Bytes.

of packets in the same time interval.

Increasing packet size appeared to correlate weakly with a decreased packet switching time, with an R-squared value of 0.321 for the trendline in Fig. 5.1. However, upon review, packet size was an overly indirect metric for measuring the packet arrival frequency, though it was the main controllable iperf3 variable associated with packet arrival frequency. For example, increasing the packet size from 200 bytes to 400 bytes represents a doubling of the size and therefore halves the amount of packets being delivered. At greater sizes, an increase of 200 bytes becomes less meaningful, as the percent increase afforded by an increase of 200 bytes decreases as the packet size increases.

Evaluating the same data in terms of the amount of packets delivered in the 10-second test intervals against the packet switching delay, shown in Fig. 5.2, showed a stronger correlation between the amount of packets being delivered and packet switching delay than the size of the packets being delivered. Displaying the exact same test data with the explanatory variable of number of packets had a R-squared value of 0.828 for the positive correlation, compared to the R-squared value of 0.321

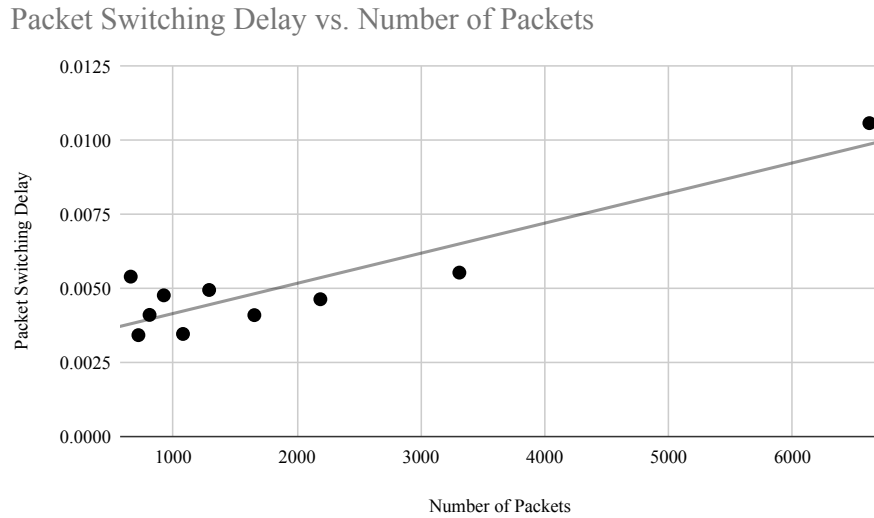


Fig. 5.2: Packet Switching Delay vs. Number of Packets Delivered.

for the original explanatory variable of packet size shown in Fig. 5.1.

Overall, this shows that the frequency at which packets are being delivered to the AP for switching from the network has a strong correlation with the packet switching delay. Since this delivery frequency varies with traffic, packet switching delay can likely be effective as an indicator of traffic and the AP’s performance.

## 5.2 Packet Switching Delay vs. Container Resources

This section displays the results of different container resources on packet switching delay. The main resources considered include assignment of CPUs, memory, and swap memory. Each of these are measured against packet switching delay averages for 10-second iperf3 tests using packets of fixed size at 200 bytes. In each comparison of tests, one resource variable is varied for the test and the others remain fixed.

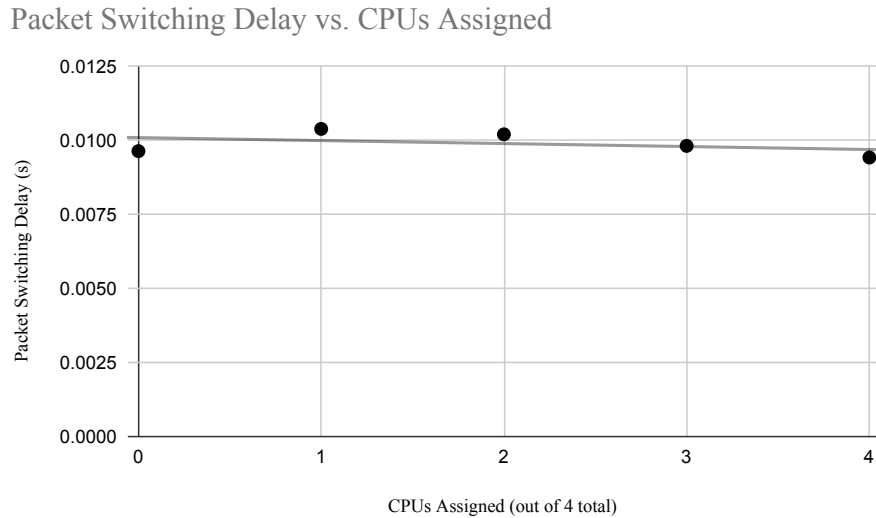


Fig. 5.3: Packet Switching Delay vs CPUs Assigned to Containers.

### 5.2.1 CPUs Assigned

This section shows the results of assigning different amounts of processors to running containers. In this scenario, containers are not pinned to specific processors except for when CPUs=1, but containers also do not necessarily take up all the processor’s resources, as other assigned resources, memory and swap memory in particular, are set to a constant 1 GB. The packet size is fixed to 200 bytes, and the amounts of packets arriving is similarly fixed.

The correlation between assigned CPUs and packet switching delay unexpectedly appears to be weakly negative. Overall, it may be possible that the numbers of CPUs assigned to containers does not have a significant impact unless all of the CPU’s resources are assigned to containers. In the case where the container load remains constant at less than all of the processing resources of the CPUs and the amount of CPUs changes, there does not appear to be a negative impact on packet switching delay as a result of assigning more CPUs to containers.

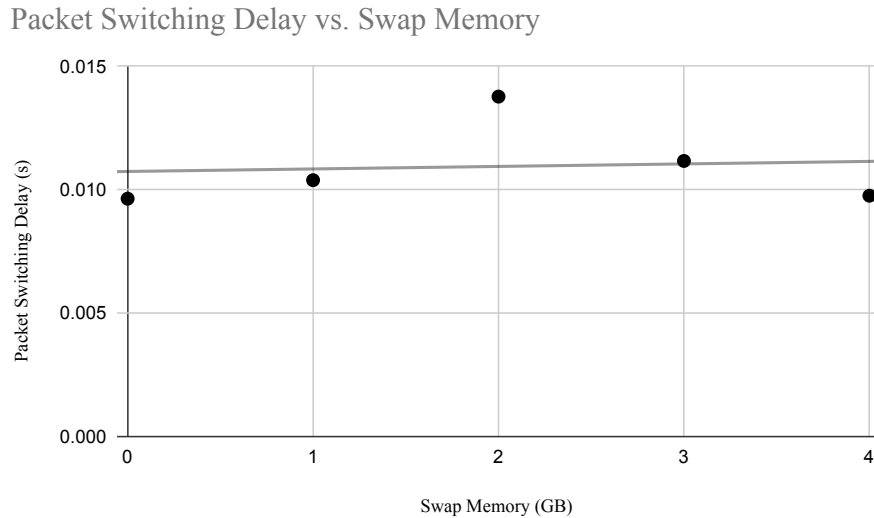


Fig. 5.4: Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=1 GB).

### 5.2.2 Swap Memory and Memory

This section shows the results of assigning different amounts of swap memory and memory to running containers. For each test measurement, packet size (and delivery frequency) are constant, the amount of CPUs assigned is constant at 1, and memory is set to a constant GB value between 1 and 3. Swap memory is the primary test variable for each set of test data displayed.

These results in Fig. 5.4 represent variations in the amount of swap memory assigned to containers for a constant CPU assignment of 1 and 1 GB of memory assigned to containers. Oddly, though packet switching delay increases on average according to the trendline, there is a weak correlation between delay and swap memory assigned to containers for this scenario. Though packet switching delay initially increases for 1 and 2 GB of swap memory, it decreases again for 3 and 4 GB. However, the average packet switching delay for 4 GB of swap memory remains slightly larger than the original value for the baseline measurement. There could potentially be confounding factors that are interfering with the results. In

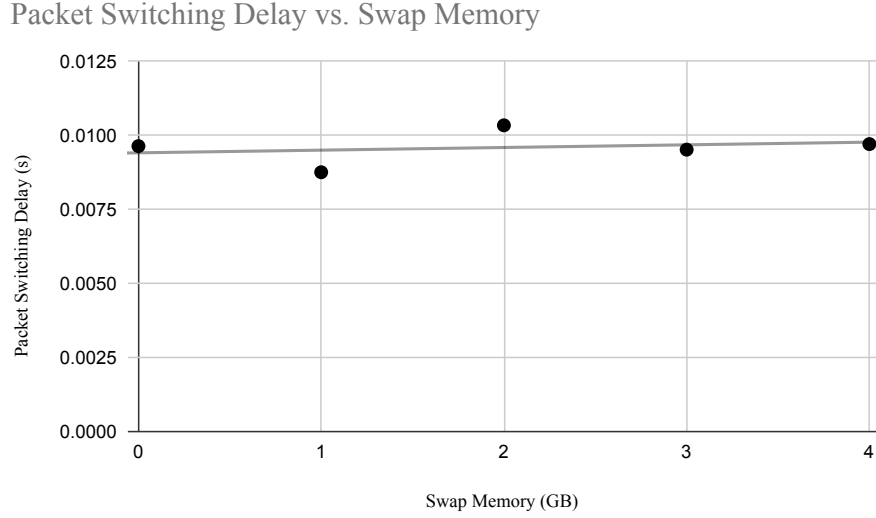


Fig. 5.5: Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=2 GB).

particular, the 3 and 4 GB tests were further removed from the baseline in terms of the time elapsed between tests than the 1 and 2 GB tests. If there was any change in the network itself during that time, that could interfere with the results. The range between the largest and smallest measurement is also larger than the subsequent tests, at 0.004139 seconds compared to 0.001585 and 0.001562.

For a set memory of 2 GB, depicted in Fig. 5.5, the packet switching delay average is more tightly clustered than for Fig. 5.4, with a range of 0.001585 seconds compared to 0.004129. Furthermore, the correlation between increasing swap memory and increasing average packet switching delay is stronger than the results for 1 GB of memory assigned to containers.

For a constant memory of 3 GB assigned to containers, depicted in Fig. 5.6, there is also an overall increase in packet switching delay with increased swap memory assigned to containers.

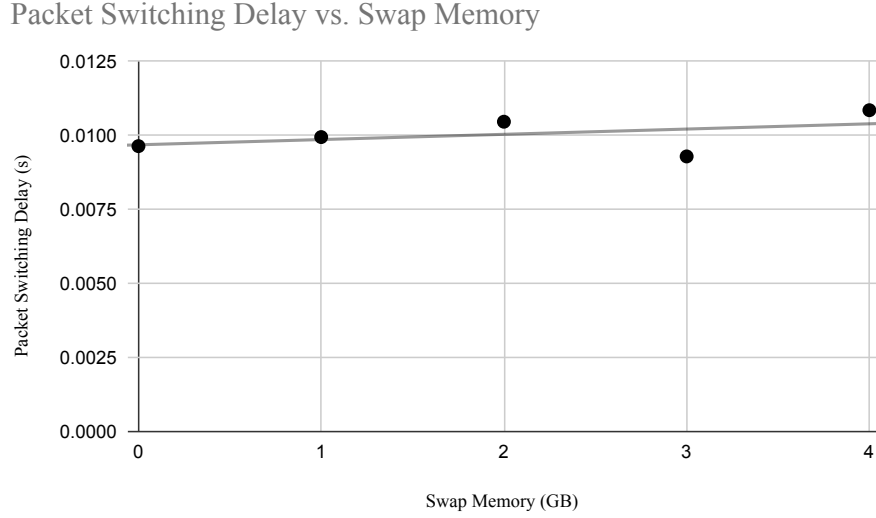


Fig. 5.6: Packet Switching Delay vs. Swap Memory Assigned to Containers (for Memory=3 GB).

### 5.3 Summary

Overall, as hypothesized, an increased frequency of packet arrival at the AP correlates strongly with a packet switching delay. Therefore, packet switching delay is correlated with the amount of traffic being switched through the AP, which means that it can be used as a reasonable indirect measurement of traffic and, in future developments, used to predict future traffic trends.

Furthermore, though the amount of CPUs assigned to containers does not appear to have a negative impact on packet switching delay if the container load is not high, other controllable container resources such as swap memory do appear to be positively correlated with packet switching delay, and increasing swap memory increases packet switching delay. This means that, along with standard load balancing techniques and processing capacity, it may indeed be beneficial to control the assignment of other resources to containers in order to prioritize packet switching delay.

---

## CHAPTER 6

# Risk Analysis

This chapter shows the risk analysis for NetCon. Risks have been sorted in descending order according to their calculated impact. The impact was calculated using the risk's estimated probability of occurring and its severity.

Risk	Consequences	Probability	Severity	Impact	Mitigation
Insufficient Time	Project completion is delayed	0.6	8	4.8	Iterative process model: provide basic functionality and additional features later
Software Issues	Will take additional time to fix these issues and reinstall	0.7	6	4.2	Periodically ensure software/OS is up-to-date
Hardware problems	Must repurchase and reinstall all software	0.2	6	3.6	Have multiple physical copies of our hardware up-to-date as backup
Insufficient Data	LSTM model is inaccurate and can create traffic prediction errors	0.5	7	3.5	Begin data collection process early so that we are able to play around with the data and see what works and what does not
Bugs	Project completion is delayed	0.9	3	2.7	Constantly test our code and project throughout our development
Loss of Data or Work Environment	Must restart from where we last saved in our backup system or from scratch	0.2	8	1.6	Constantly update our backup system whenever we complete a task

Table 6.1: Risk Analysis Table.



Overall, several of these risks impacted the development of NetCon, and some of them were more severe than initially calculated. Firstly, due to unexpected time consumption, the implementation of NetCon was much less ambitious than the original idea, focusing more on current packet switching delay than being able to predict future network trends. One issue that occurred was that the original Linux machines did not integrate well with the required eBPF programs, so it was necessary to purchase entirely new hardware and reinstall from scratch. On the new Linux machines, described in 3.1, there were different wired and wireless interfaces and an entirely different process for setting up the AP, so reinstalling these important prerequisites took more time than expected. Furthermore, since the LSTM model did not integrate well with the other components of the system, it had to be removed from consideration for this iteration of NetCon.

---

# CHAPTER 7

## Existing Issues

In addition to the technical aspects of NetCon, it is also important to talk about the non-technical ramifications that exists. These non-technical issues can have a significant impact on users if they are not taken into account during the product development. As such, it is important that NetCon considers and addresses these concerns.

### 7.1 Ethical Issues

NetCon uses the cloud for information configuration and transfer, specifically data packets as well as containers. Since NetCon can decrease latency in communication with container-based applications without jeopardizing any of the AP's functionalities, this ensures that other users of the AP are not negatively impacted by NetCon. Users can experience the positive effects that NetCon brings to APs without facing any negative repercussions. Furthermore, NetCon is also designed to prevent containers from over-utilizing APs and denying service to other users, so that should not be an issue. NetCon only runs if the system administrator or owners of the AP install it and the prerequisite software on the device, so it would be implemented with their full consent.

## **7.2 Societal Issues**

NetCon’s goal is to improve modern APs by decreasing processor idleness. NetCon should not adversely impact any members of society. In fact, NetCon is designed to be beneficial to individuals who have low-latency applications they would like to run as containers on edge devices. NetCon can be accessed and used by all populations regardless of their social status, demographic, or religion.

## **7.3 Political Issues**

A major political concern with internet networks is privacy. NetCon does not have any privacy concerns as NetCon primarily focuses on transferring containers and resources to decrease latency and packet switching delay, not transferring user private data or information. NetCon does not collect user-based data, only AP-based data measurements of packet switching delay and traffic. The data in the users’ containers is entirely private, but may be relocated based on available resources and impact on packet switching delay. NetCon does not store user’s private information or use it in any way.

## **7.4 Economic Issues**

NetCon can be used on any Linux APs, so customers do not need to specifically purchase a new device to use NetCon. Additionally, new features and functionalities can be added to NetCon in the future to further improve the model without needing to create an entirely new infrastructure.

## **7.5 Environmental Issues**

Since NetCon can work on all Linux APs, new hardware is not needed to be manufactured in order to support NetCon. NetCon continuously checks and reallocate containers when needed to avoid resources sitting in idle time. This method ensures that NetCon is efficient with the container resources and are continuously allocating them to perform tasks to their best ability. NetCon also helps mitigate the wasted idle processing resources of APs by using them to run container.

## **7.6 Usability Issues**

NetCon makes it easier to install, run, and delete containers without impacting AP functionalities. NetCon only requires the presence of a AP in order to be used. Since NetCon uses Docker containers, it is easy to implement any Docker containers on an AP using NetCon. NetCon so that any Docker container can run safely without adversely impacting the performance of the AP it is running on.

## **7.7 Sustainability Issues**

NetCon is a sustainable networking system as it can be use to upgrade current APs without requiring the purchase of new devices and technology. Furthermore, NetCon eliminates digital waste by constantly reusing containers and resources that would have been left idle. This allows for more devices to connect without encountering any significant delays. However, to fully assess the sustainability implications of NetCon, it may be necessary to run additional testing on the energy consumption of APs running container with and without NetCon.

## **7.8 Health and Safety Issues**

NetCon is not inherently a software that is likely to compromise the health and safety of its users. However, NetCon does deal with health and safety issues through its goal of decreasing latency from the cloud without compromising AP functionality. For example, security applications commonly require low latency [1]. If NetCon makes it easier to run these applications via containers on close edge devices like APs without negatively impacting network functionality, it may improve user's safety.

## **7.9 Manufacturing Issues**

NetCon does not have any manufacturing concerns as NetCon can run on existing APs. NetCon does not require any new hardware devices in order to function and the system can be installed onto modern APs that exists in many homes, school, and work, provided that they run on Linux.

## **7.10 Summary**

Overall, NetCon is unlikely to create any particular ethical, environmental, or implementation issues. In terms of ethical issues, NetCon does not collect user-based data, and its primary functionality are unlikely to harm users in terms of health and safety, and may, depending on its application, actually help with health and safety. In terms of environmental issues, NetCon is meant to use an underutilized resource that is commonly wasted, but it would be prudent to conduct further testing on the energy expenditure of NetCon in order to assess the full impact on the environment. Finally, since NetCon runs in a standard Linux environment, it is relatively portable

and will not create any manufacturing issues or sustainability issues through the production of new, specialized hardware.

---

## CHAPTER 8

# Conclusion and Future Work

This section presents the overall conclusion to this thesis as well as a brief note on potential future work that could be done to improve or expand the scope of NetCon.

### 8.1 Conclusion

NetCon is a proposed method make APs more efficient by running containers while still ensuring that packet switching delay remains low, and the AP can continue to perform its main function. The impact of different container resources on packet switching delay was evaluated and found to have a negative impact overall, which can be improved by keeping track of packet switching delay vs. container resources and ensuring that the former does not suffer.

Overall, APs exist across the world and are used almost every day by most of the world's population. Through these improvements with NetCon, users will have the option to run low-latency applications on APs, but they will not experience increased network congestion as a result.

### 8.2 Future Work

There are several potential future improvements and expansions of NetCon that are possible.

### **8.2.1 Energy Efficiency**

One major goal is to determine the level of energy expenditure of APs running containers with NetCon compared to standard container migration and load balancing algorithms and to APs that are only working on switching packets. This would allow for a more informed perspective on the sustainability impacts of NetCon, and if NetCon was found to be energy-inefficient, it would open up more avenues for improvement.

### **8.2.2 LSTM**

Secondly, the original intent was to implement LSTM in order to predict future increases and decreases in packet switching delays in order to be ready to remove or return containers to the AP as soon as possible. However, due to time difficulties, this was not fully implemented. Adding a predictive algorithm to NetCon instead of relying simply on current measurements would allow for quicker responses to any issues that show up, assuming the predictions are sufficiently accurate.

### **8.2.3 Generalization**

Finally, another potential improvement would be to generalize NetCon to work on any available network switching appliances, as the current implementation only works on APs. Furthermore, NetCon's packet switching delay measurement is currently specialized to work with a particular NIC. Another future improvement might be to make NetCon easily adjustable to mark the arrival and delivery timestamps of packets for multiple types of NIC that use different system calls.



---

# Bibliography

- [1] C. Desiniotis, J. Majors, and C. Miremadi, “Edgeap: Enabling edge computing on wireless access points,” in *Scholar Commons*, vol. 65. Santa Clara University, 2020.
- [2] SmartAerials. (2020) Wireless Access Points - What They Do and How They Work. [Online]. Available: <https://www.smartaerials.co.uk/blog/wireless-access-points-what-they-do-how-they-work>
- [3] K. Ross and J. Kurose. (2000) Delay and Loss in Packet-Switched Networks. [Online]. Available: [http://www2.ic.uff.br/~michael/kr1999/1-introduction/1\\_06-delay.htm](http://www2.ic.uff.br/~michael/kr1999/1-introduction/1_06-delay.htm)
- [4] IBM Cloud Education. (2021) What are containers? [Online]. Available: <https://www.ibm.com/cloud/learn/containers>
- [5] R. Synytsky. (2016) Containers Live Migration: Behind the Scenes. [Online]. Available: <https://www.infoq.com/articles/container-live-migration/>
- [6] L. Ma, S. Yi, and Q. Li, “Efficient service handoff across edge servers via docker container migration,” in *In Proceedings of SEC '17*. ACM, 2017, pp. 1–13.
- [7] S. Maheshwari, S. Choudhury, I. Seskar, and D. Raychaudhuri, “Traffic-aware dynamic container migration for real-time support in mobile edge clouds,” in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2018, pp. 1–6.

- [8] C.-H. Hong, K. Lee, M. Kang, and C. Yoo, “qCon: QoS-Aware Network Resource Management for Fog Computing,” *Sensors*, vol. 18, no. 10, p. 3444, 2018.
- [9] D. Samani, C. Denninnart, J. Bacik, and M. Salehi, “The art of cpu-pinning: Evaluating and improving the performance of virtualization and containerization platforms,” in *The 49th International Conference on Parallel Processing*, 2020.
- [10] C.-T. Yang, S.-T. Chen, and J.-C. Liu, “An energy-efficient cloud system with novel dynamic resource allocation methods,” *Springer*, vol. 75, pp. 4408–4429, 2019.
- [11] M. Vieira, M. Castanho, R. Pacifico, and E. Santos, “Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications,” *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–36, 2021.
- [12] N. Van Tu, J.-H. Yoo, and J. W.-K. Hong, “Real-time monitoring of packet processing time for virtual network functions,” in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 138–143.
- [13] ReQtest. (2012) Why is the difference between functional and Non-functional requirements important? [Online]. Available: <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>
- [14] Amazon. (2022) Kingdel Fanless Industrial PC, Windows 10 Home Mini Computer, with Intel i7 CPU, 8GB RAM, 512GB SSD, 2xNICs, 2xHD Ports, 4xUSB 3.0, 6xCOM RS232, Wi-Fi. [Online]. Available: <https://www.amazon.com/Kingdel-Industrail-Computer-i7-5550U-Broadwell/dp/B06XYXRZ1R>

- [15] Canonical Ltd. (2018) Ubuntu 20.04.4 LTS (Focal Fossa). [Online]. Available: <https://releases.ubuntu.com/20.04/>
- [16] J. Malinen. (2013) hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. [Online]. Available: <https://w1.fi/hostapd/>
- [17] E. Docile. (2020) Netplan network configuration tutorial for beginners. [Online]. Available: <https://linuxconfig.org/netplan-network-configuration-tutorial-for-beginners>
- [18] C. Cassagnes, L. Trestioreanu, C. Joly, and R. State, “The rise of eBPF for non-intrusive performance monitoring,” in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–7.
- [19] Amazon. (2022) NETELY 802.11N Dual Band Mini-PCIE Interface 600Mbps WiFi Bluetooth 4.0 Adapter for Laptop PCs-Mini-PCIE WiFi Card with Bluetooth 4.0(AR9462). [Online]. Available: <https://www.amazon.com/NETELY-Mini-PCIE-Interface-PCs-Mini-PCIE-AR9462/dp/B09NYCLBLB>
- [20] J.-B. Lee, T.-H. Yoo, E.-H. Lee, and B.-H. Hwang, “High-performance software load balancer for cloud-native architecture,” *IEEE Access*, vol. 9, pp. 123,704–123,716, 2021.
- [21] C. Maklin. (2019) Lstm recurrent neural network keras example. [Online]. Available: <https://tinyurl.com/long-short-term-memory-lstm>
- [22] Docker. (2021) Runtime options with Memory, CPUs, and GPUs. [Online]. Available: [https://docs.docker.com/config/containers/resource\\_constraints/](https://docs.docker.com/config/containers/resource_constraints/)
- [23] Avi Networks. (2021) What is Container Orchestration? [Online]. Available: <https://avinetworks.com/glossary/container-orchestration/>

- [24] C. Melandex. (2018) The Advantages of Using Kubernetes and Docker Together. [Online]. Available: <https://stackify.com/kubernetes-docker-deployments/>
- [25] ESnet. (2022) iperf3. [Online]. Available: <https://software.es.net/iperf/>
- [26] Pandas. (2019) About pandas. [Online]. Available: <https://pandas.pydata.org/about/>






# NetCon\_Dynamic\_Resource\_Allocation\_Containers\_Running\_Network\_Switching\_Publication

Final Audit Report

2022-06-09

Created:	2022-06-09
By:	Darcy Yaley (dyaley@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAA3muAaF4ye7-VKMXb_zKDLlcZtUs7Inkx

## "NetCon\_Dynamic\_Resource\_Allocation\_Containers\_Running\_Network\_Switching\_Publication" History

-  Document created by Darcy Yaley (dyaley@scu.edu)  
2022-06-09 - 4:43:25 PM GMT
-  Document emailed to N. Ling (nling@scu.edu) for signature  
2022-06-09 - 4:43:56 PM GMT
-  Email viewed by N. Ling (nling@scu.edu)  
2022-06-09 - 9:19:55 PM GMT
-  Document e-signed by N. Ling (nling@scu.edu)  
Signature Date: 2022-06-09 - 9:20:45 PM GMT - Time Source: server
-  Agreement completed.  
2022-06-09 - 9:20:45 PM GMT