CMPE 492

COMIC-RAN: Container-based Migration in C-RAN

Ömer Şükrü Uyduran

Advisor:

Tuna Tuğcu

November 11, 2024

TABLE OF CONTENTS

1. INTRODUCTION

In recent years, cellular network technologies have seen significant advancements, driven by the growing demands for higher data rates, lower latency, and improved energy efficiency. The evolution from 4G to 5G has paved the way for new architectures like Cloud Radio Access Networks (Cloud-RAN), which offer promising solutions for the increasing challenges faced by mobile network providers. With the rapid expansion of network coverage and the rise of mobile devices, the necessity for a more efficient approach to infrastructure and resource management is apparent. This project focuses on container-based migration within Cloud-RAN, utilizing virtualization techniques to migrate running server processes between virtual machines (VMs) with minimal disruption, as part of an effort to contribute to the emerging technological landscape of 5G.

The traditional approach to Radio Access Networks (RAN) involves distributed Baseband Units (BBUs) and Remote Radio Heads (RRHs), where each BBU is dedicated to its respective RRH. This setup leads to significant challenges, such as increased operational costs and inflexible resource management, which become particularly problematic as coverage areas shrink with newer generations of cellular networks. The move to 5G has prompted the introduction of Cloud-RAN, where BBUs are virtualized and centralized, allowing for more dynamic allocation of resources and reducing the burden of hardware maintenance. Unlike earlier work focused on virtual machine-based BBUs, our project leverages containerization for increased efficiency and scalability.

Our project aims to achieve live migration of server processes running inside containers between two virtual machines in a post-copy manner, minimizing downtime and ensuring high availability—critical aspects of achieving compliance with stringent 5G latency requirements. Instead of relying on third-party containerization tools, we have developed our own sandbox environment to ensure simplicity and tailor the functionality to our needs. In this midterm report, we present the initial progress made

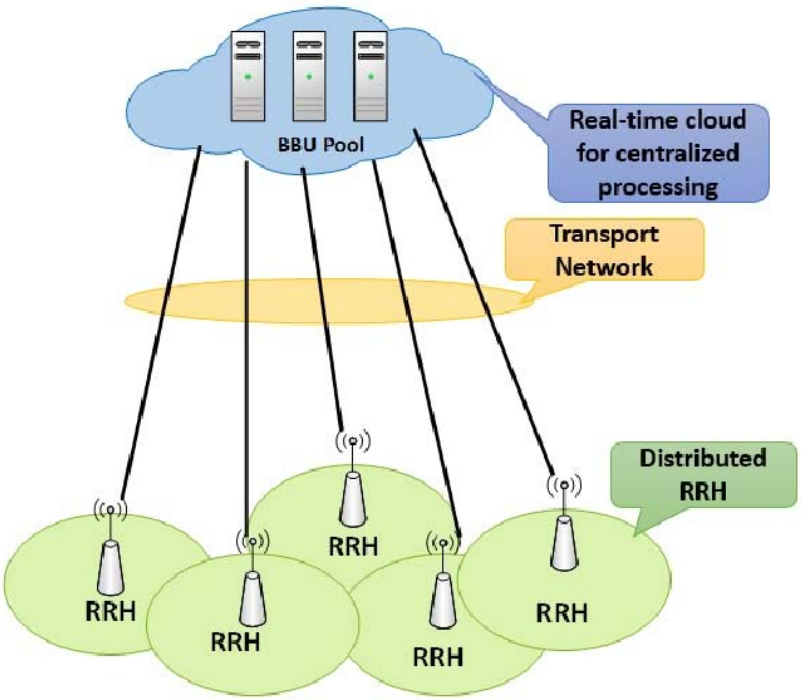

Figure 1.1. Traditional C-RAN Architecture (taken from [3]).

in implementing the sandbox environment, setting up UDP servers, and preparing the necessary infrastructure for live migration.

1.1. Broad Impact

The impact of this project extends beyond the academic exploration of container migration. The telecommunications industry is currently undergoing a major transformation, with a push toward more flexible and cost-efficient solutions to meet the increasing demands of mobile data users. By focusing on container-based migration in Cloud-RAN, our project directly contributes to addressing some of the major challenges faced in rolling out 5G infrastructure, specifically by reducing costs, improving resource utilization, and minimizing network latency. These improvements are critical in delivering the high-speed, reliable connectivity promised by 5G and ensuring that network operators can meet user expectations without incurring prohibitive capital and operational expenditures.

The methodology of using lightweight containers instead of traditional VMs also

offers significant potential benefits for the environment. With reduced resource overhead and improved efficiency, container-based Cloud-RAN can contribute to lower energy consumption in cellular networks, thereby supporting sustainability initiatives in the telecommunications industry. Additionally, the implementation of our own sandbox environment demonstrates the feasibility of customized containerization solutions that are specifically tailored to the needs of 5G and beyond, offering a pathway for future research and development in more targeted virtualization techniques.

Another important impact is on network reliability. By employing post-copy migration, we hope that our approach minimizes the downtime experienced by users during the migration of active network components. This directly translates into better user experience and satisfaction, as well as a more robust network capable of handling the demands of emerging applications such as augmented reality, virtual reality, and autonomous systems, which all require consistent, low-latency connectivity.

1.2. Ethical Considerations

1.2.1. Access

All of the source code, documentation, and the guides are published as open to everyone's access in the project's GitHub repository.

1.2.2. Transparency and Accountability

Methodologies and important aspects of the project will be reported to ensure interested ones can reproduce the outcomes and evaluate the results.

2. PROJECT DEFINITION AND PLANNING

2.1. Project Definition

This project aims to implement and evaluate a container-based live migration mechanism within Cloud-RAN, focusing on migrating server processes between virtual machines with minimal disruption. By utilizing our own sandbox environment, we seek to achieve efficient migration with reduced complexity compared to traditional virtualization solutions. The goal is to minimize downtime and maintain high availability during the migration process, meeting the stringent requirements of 5G in terms of latency and reliability.

2.2. Project Planning

2.2.1. Remaining Work and Future Planning

For the remainder of this semester, our primary focus will be on implementing the migration process in a post-copy manner, ensuring that server processes can be seamlessly transferred from one virtual machine to another. We will also work on designing and conducting test scenarios that simulate various real-world conditions, such as large memory changes and heavy server loads, to evaluate the robustness of our migration approach. These test scenarios will help us identify potential bottlenecks and fine-tune our solution to ensure optimal performance.

2.2.2. Project Time and Resource Estimation

The project is planned to be completed by January 2025, which marks the end of the current academic semester. So far, all milestones have been met according to the planned schedule, and the progress is on track to meet the final deadline. We expect to complete the migration implementation and testing phases in time for the

final evaluation.

2.2.3. Success Criteria

The success of this project will be measured based on two primary criteria: achieving zero downtime during the live migration process and maintaining response delays within the limits defined by 5G standards (i.e., less than 1 millisecond). These criteria are essential to ensure that the migration process is seamless and that users experience no noticeable disruption in service.

2.2.4. Risk Analysis

Performing live post-copy migration poses several challenges, particularly in synchronizing memory changes between the source and destination machines. The frequent updates to memory during the migration process make it difficult to achieve zero downtime, as any discrepancies between the two instances can lead to inconsistencies and service interruptions. It is a considerable possibility that we may not achieve complete zero downtime, given the complexity of managing synchronization in real time. However, our efforts will focus on minimizing downtime as much as possible, aiming for an acceptable level of service continuity.

2.2.5. Team Work

As a student, I am working on this project alone. However, my project advisor Tuna Tuğcu supports me during our weekly meetings. He brainstorms with me on the challenges I encounter, suggests me critical techniques for the problems, and conveys his vision to me.

3. RELATED WORK

3.1. 5G/CloudRAN [1]

The 5G/CloudRAN project by Ahmet Bugrahan Tasdan and Salih Sevgican aimed to implement a Cloud-RAN architecture for 5G networks, focusing on the virtualization of BBUs using virtual machines. Their work primarily utilized OpenStack to create virtual BBUs, which were then migrated between different physical servers. This project demonstrated the feasibility of using virtual machines for live migration within a Cloud-RAN environment, successfully achieving migration without significant downtime. However, the use of virtual machines posed challenges in terms of resource overhead and memory footprint, which impacted the efficiency of the migration process. By using container-based virtualization, our project aims to address these limitations by providing a more lightweight and scalable solution.

Tasdan and Sevgican's work also included the use of Software Defined Networking (SDN) to manage network traffic during migration, ensuring that the network remained stable and consistent throughout the process. Their approach to SDN integration serves as a foundation for our own use of SDN in managing container migrations, particularly in ensuring minimal packet loss and maintaining service quality during the transition.

3.2. Container-Founded 5G vRAN Network [2]

The Container-Founded 5G vRAN Network project by Ömer Cihan Benzer and Yunus Emre Topal focused on implementing a virtual Radio Access Network (vRAN) using containerization instead of traditional virtual machines. Their work highlighted the advantages of using containers for vRAN, including reduced resource overhead and increased scalability. The project utilized Docker and Kubernetes to manage and orchestrate the containerized BBUs, demonstrating that container-based vRAN could achieve comparable performance to VM-based approaches while offering significant

improvements in resource utilization.

Benzer and Topal also explored the challenges of live container migration, particularly in maintaining low latency and minimizing packet loss. They utilized Kubernetes' built-in tools to manage container orchestration and ensure that the migration process was as seamless as possible. Their project serves as a direct precursor to our own work, providing valuable insights into the benefits and challenges of container-based migration in a 5G context. By building on their findings, we aim to further optimize the migration process, focusing specifically on achieving zero downtime and meeting 5G latency requirements.

Additionally, Benzer and Topal's project referenced the work of Tasdan and Sevgican, noting the differences between VM-based and container-based approaches in terms of efficiency and scalability. Our project continues this line of investigation by directly comparing the performance of our custom sandbox tool with existing containerization solutions, aiming to provide a more tailored and efficient approach for Cloud-RAN migration.

4. METHODOLOGY

Our approach to container-based migration within Cloud-RAN is designed to address the unique challenges of achieving seamless, low-latency live migration that complies with 5G requirements. The key methodology revolves around post-copy migration, which allows server processes to continue running while memory pages are transferred to the destination virtual machine. This approach minimizes downtime and ensures a smooth transition, crucial for maintaining the stringent latency and reliability standards demanded by 5G.

4.1. Custom Sandbox Environment: Jailor

One of the foundational elements of our methodology is the development of a custom sandbox tool called Jailor. Unlike traditional containerization tools such as Docker or Kubernetes, Jailor is designed specifically to meet the requirements of our project while avoiding unnecessary complexity. Jailor reads configuration files that define the container's parameters, including the root directory, executables, network configurations, and any dependencies. Using Linux features like chroot, unshare, and network namespaces, Jailor isolates server processes in a lightweight, secure environment. This custom approach allows us to focus on core functionalities relevant to Cloud-RAN without the overhead associated with generic containerization tools.

4.2. Migration Process

Our migration process involves incrementally copying the memory of a running server process from the source VM to the destination VM while maintaining the server's availability. The post-copy migration technique helps ensure that the server process continues to respond to requests even during the migration. During the process, both instances (source and destination) are kept in sync by directing incoming UDP requests to both VMs, while only accepting responses from the source until the migration is com-

plete. This ensures that no requests are lost and that the system maintains consistent service quality.

4.3. Communication Between Components

The architecture of our solution comprises several components that work together to achieve live migration. The main components include:

4.3.1. Jailor

Responsible for setting up and managing the sandbox environment, including setting up network interfaces and isolating namespaces. Jailor ensures that each server process runs in a secure, isolated environment with the necessary configurations for migration.

4.3.2. SDN Controller

Our Software Defined Networking (SDN) controller is responsible for managing network traffic, ensuring that incoming requests are routed to the correct instance (source or destination) during migration. The SDN controller helps us minimize packet loss and ensures consistent connectivity throughout the migration.

4.3.3. Server and Client

The server is a simple UDP-based server that receives numerical requests and returns their squared values. The client sends requests to the server, simulating real-world network traffic that the system must handle during migration. The server runs inside the sandbox, and its migration is the focal point of our project.

4.3.4. Sdeamon and Smanager

The sdeamon is responsible for managing sandboxes within the virtual machines, listening on a TCP port for commands to start or terminate containers. Smanager acts as an interface for the user to manage these sandboxes, sending commands to sdeamon to control server instances as needed.

We hope that the combination of these components allows us to create a flexible and efficient migration system that minimizes downtime and ensures that network performance is not compromised during migration. By developing custom tools and leveraging SDN for dynamic routing, we are able to address the specific needs of 5G Cloud-RAN, providing a tailored solution that is both efficient and scalable.

4.4. Testing and Evaluation

To validate our methodology, we will design and conduct a series of tests that simulate real-world scenarios, including changing memory utilization and heavy server loads. We will also use a monitoring tool like Wireshark to measure network traffic statistics, such as packet loss, latency, and throughput, to ensure that our solution meets the required latency and reliability standards. The success of these tests will be determined based on metrics such as downtime, packet loss, and response latency, with the goal of achieving zero downtime and sub-millisecond response times.

Overall, our methodology combines a custom sandbox environment, post-copy migration, SDN-based network management, and comprehensive testing to achieve efficient and reliable container migration within Cloud-RAN. This approach not only ensures compliance with 5G standards but also offers a scalable solution for future network deployments.

5. REQUIREMENTS SPECIFICATION

5.1. Functional Requirements

- **Container Management**: The system must provide a mechanism to set up and manage sandbox environments for running isolated server processes. It should automate configuration, dependency resolution, and management of container lifecycles.
- **Live Migration**: The system must support live migration of server processes between virtual machines with 0 downtime. The migration must maintain service availability with latency under 1 millisecond.
- **Network Traffic Routing**: An SDN-based approach should manage and route incoming requests to appropriate server instances during the migration, ensuring seamless service without packet loss.
- **User Control**: The system must provide a command line interface for managing sandboxes, allowing users to start, terminate, and migrate server processes across virtual machines.
- **Monitoring Capabilities**: The system must include monitoring tools to track network performance metrics, such as packet loss and latency, to evaluate the efficiency of the migration process.

5.2. Non-functional Requirements

- **Performance**: The system must ensure 0 downtime and meet 5G latency requirements of less than 1 millisecond during migration. Scalability should allow for managing multiple sandboxes across multiple VMs.
- **Security**: Processes running inside sandboxes must be securely isolated from the host system.
- **Usability**: The system must simplify container management through automation of configurations and provide an intuitive interface for end-users to control the

migration and sandbox processes.

- **Reliability**: The system must guarantee continuous availability of server processes during migration.

5.3. System Requirements

- **Hardware**: The implementation requires at least two virtual machines with sufficient resources (CPU, memory, storage) to support multiple sandbox instances.
- **Software**: The system must run on a Linux-based OS with support for namespaces, chroot, and other isolation features. A compatible C compiler (e.g., GCC) and network monitoring tools like Wireshark are also required.

6. DESIGN

You can find diagrams for system design and information flow below. In the diagrams, Server-2 within Jailor-2 is illustrated as migrating processes from VM-1 to VM-2. Also, the black arrows represents data flow between the components whereas the blue ones represents actions.

In the Figure 6.1 you can see how SDN manages the network traffic between the Servers and the Client. In this figure, Server-2 is currently being migrated from VM-1 to VM-2. See how SDN handles the traffic whose destination is a migrating server.

In the Figure 6.2 you can see how Smanager communicates with the Sdeamons in the VMs and how Sdeamons manage the sandboxes (Jailors) and Servers inside their VM.

Lastly, in the Figure 6.3, you can see the current overall design of the system.

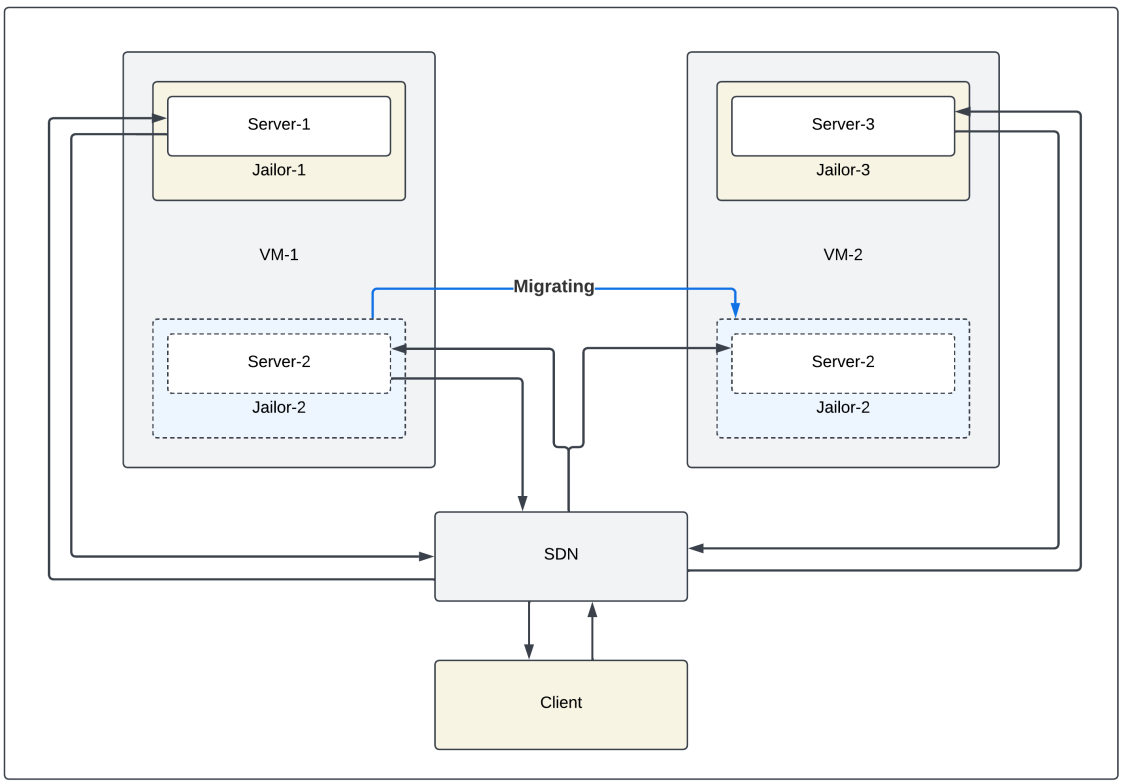

Figure 6.1. Data flow and system design for Server - Client communication.

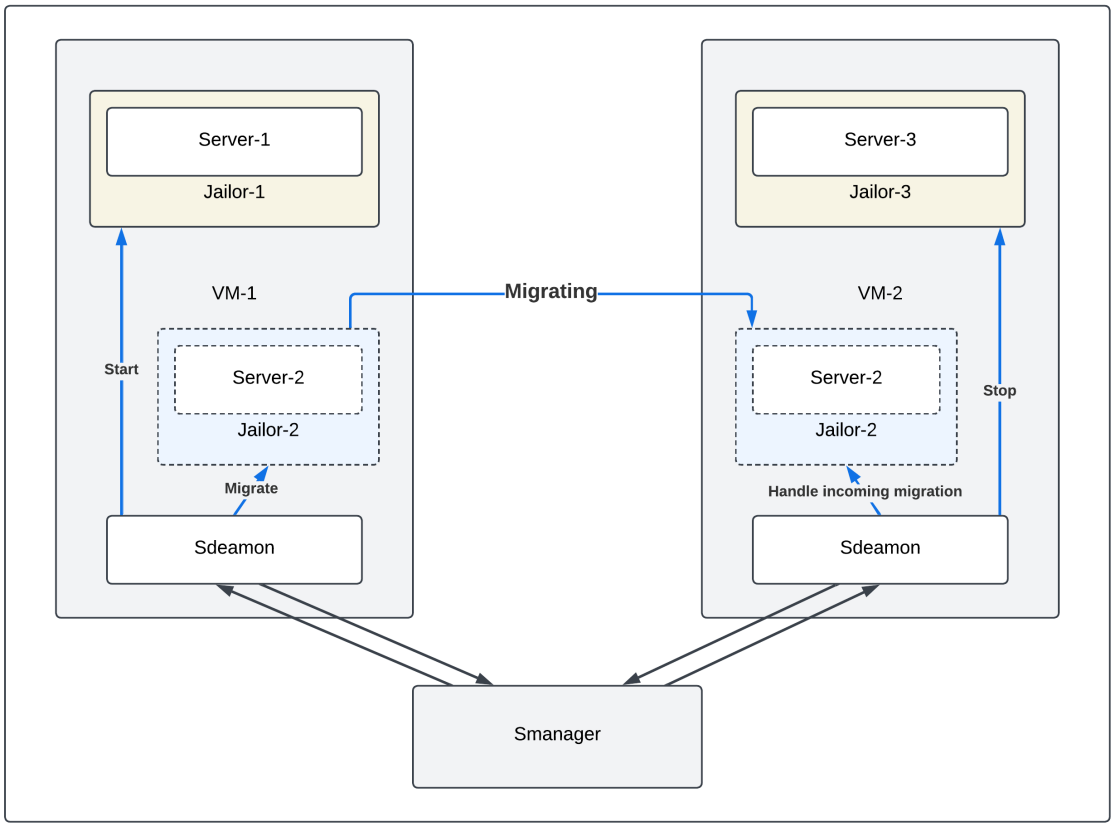

Figure 6.2. Data flow and system design for Smanager - Sdeamon communication.

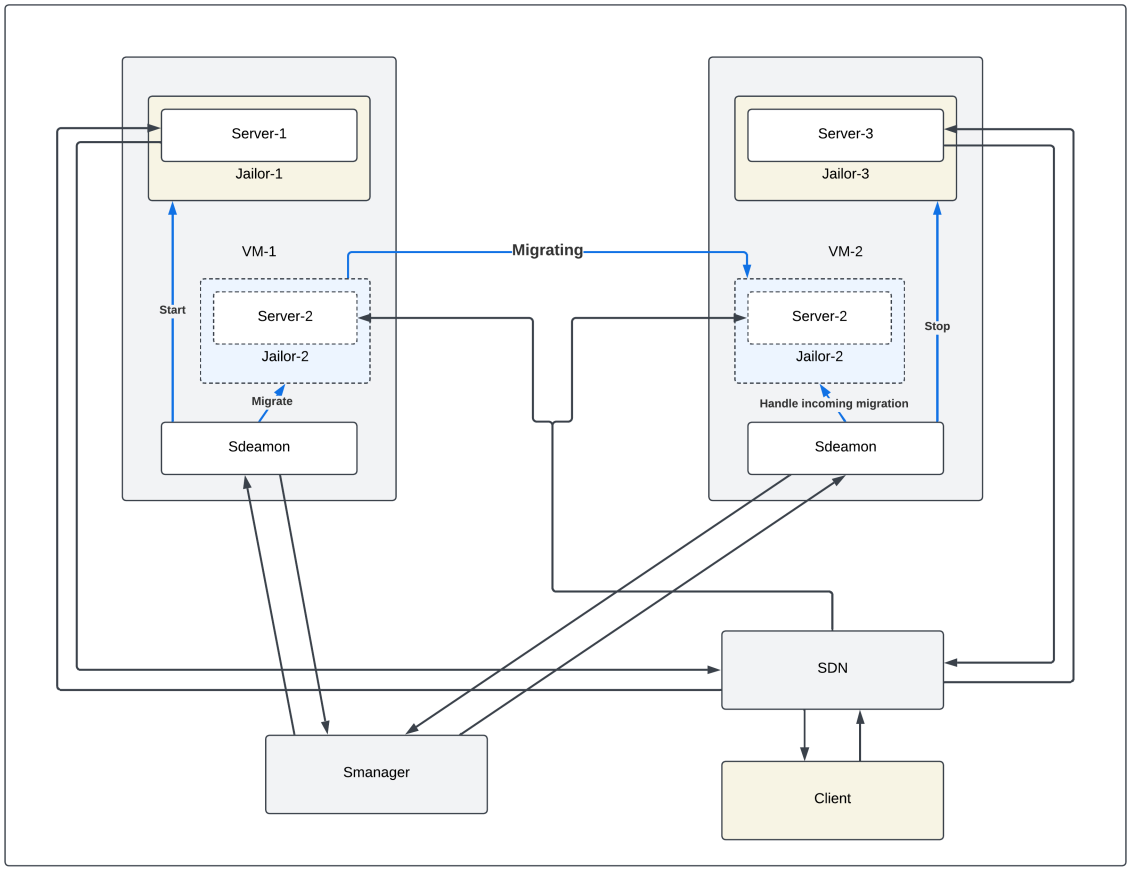

Figure 6.3. Overall system design.

7. IMPLEMENTATION

The implementation of our container-based migration system involves a suite of custom programs that facilitate the setup, management, and migration of server processes across virtual machines. All of these components have been implemented in C programming language, providing a lightweight and efficient solution tailored to the specific requirements of our project. The following sections describe each of these components in detail.

7.1. Configer

The configer program plays a critical role in automating the setup of sandbox environments. It takes a base configuration file as input, which includes details such as the sandbox ID, the path to the root directory, the path of the executable for the root process, command-line arguments for the root process, IP configurations for connectivity, a list of additional executables, required files, and symlinks. Configer is responsible for:

- **Dependency Resolution**: It reads the base configuration file to determine all the dependencies that the root process and additional executables will need within the sandbox. This includes resolving symbolic links to find their actual paths and ensuring all necessary files are listed for inclusion.
- **Output Config Generation**: It generates an output configuration file that includes a comprehensive list of files, executables, and symlinks required for the sandbox. This is especially helpful in eliminating the need for manual dependency management, significantly reducing the time and effort involved in setting up sandbox environments.

The output generated by configer serves as the input for the jailor program, ensuring that the sandbox has all the resources it needs to function correctly.

7.2. Jailor

The jailor program is responsible for creating the sandbox environments based on the configuration file provided by configer. It performs the following tasks:

- **Root Directory Setup**: Jailor creates the root directory for the sandbox and copies all the required files listed in the configuration file to this directory.
- **Network Setup**: It creates a virtual Ethernet (veth) pair to establish connectivity between the sandbox and the outside network. It also configures the network settings to enable communication.
- **Namespace Isolation**: The program clones (forks) a new process while separating it from the host system by using Linux namespaces. This includes isolating the network, process, and mount namespaces to create a fully isolated environment.
- **Chroot and Execution**: The child process is assigned the root directory specified in the configuration file using chroot, ensuring that it operates within the sandbox environment. Necessary symlinks are then created inside the sandbox, and the root process specified in the configuration file is executed with the given command-line arguments.
- **Cleanup**: Once the root process completes execution, the parent process ensures that the sandbox's root directory is cleaned up, removing all files and directories recursively.

Jailor is essential for ensuring that the server processes run in a controlled and isolated environment.

7.3. Sdeamon

The sdeamon program is designed to manage sandboxes on virtual machines. It runs as a daemon process and listens on a specified TCP port for incoming commands. Its main responsibilities include:

- **Container Management**: Sdeamon can start and terminate sandboxes based on commands received from the smanager program.
- **Port Forwarding**: It sets up necessary port forwarding rules to ensure that the sandboxes can communicate with the external network, enabling the UDP servers inside the sandboxes to receive requests.
- **External Control**: By running as a daemon, sdeamon allows sandboxes to be managed remotely, making it possible to initiate or stop server processes as needed during the migration process.

Sdeamon serves as the interface between the virtual machines and the sandbox management commands, providing the necessary control for managing server processes within each VM.

7.4. SDN

The sdn component is responsible for managing incoming client requests and directing them to the appropriate server instances. It plays a vital role in ensuring seamless communication during the migration process. The key functions of sdn include:

- **Request Management**: It accepts UDP requests from clients and uses a lookup table to determine which virtual machine and container should handle each request.
- **Threaded Request Handling**: When a request is received, sdn creates a new thread to handle it, allowing multiple requests to be processed concurrently without blocking.
- **Dynamic Routing**: During migration, sdn will update its lookup table to direct client requests to the appropriate instance, ensuring that responses are returned to the clients correctly and without interruption.

The sdn component is crucial for managing network traffic during the migration,

ensuring that requests are consistently routed to the correct server instance, even as the server process is being migrated.

7.5. Smanager

The smanager program acts as a user interface in the host system for managing the sandboxes within the virtual machines. It interacts with sdeamon to initiate or terminate server processes based on user input. The key features of smanager include:

- **User Interaction**: It prompts the user for actions such as starting or stopping server processes and sends the appropriate commands to sdeamon running inside the virtual machines.
- **Centralized Control**: By providing a centralized way to manage sandboxes, smanager simplifies the process of controlling multiple server instances across different VMs.

7.6. Server and Client

- **Server**: The server is a simple UDP server that listens for incoming numerical requests from clients. When it receives a number, it calculates the square of the number and sends the result back to the client. The server runs inside the sandbox environment, which is managed by jailor.
- **Client**: The client is a basic UDP client that sends numerical requests to the server and waits for the response. The client simulates real-world network traffic, providing a way to test the migration process and ensure that the server remains responsive throughout.

7.7. Component Interaction and Communication

The components described above work together to achieve seamless container-based migration within Cloud-RAN. The configer program sets up the necessary con-

figuration files, which are then used by jailor to create isolated sandbox environments. Sdeamon manages these sandboxes within each virtual machine, while smanager provides centralized user control. The sdn component ensures that network requests are routed correctly during migration, and the server and client programs simulate network traffic to test the system's performance.

The communication between these components is designed to be efficient and reliable, with sdeamon and smanager enabling remote management, and sdn ensuring that the migration process does not disrupt ongoing communications. Together, these components form a cohesive system that allows for the live migration of server processes with minimal downtime, meeting the stringent requirements of 5G networks.

8. CONCLUSION

In this midterm report, we have outlined the progress made in the implementation of a container-based migration system for Cloud-RAN, designed to achieve seamless migration of server processes between virtual machines. The system leverages custom-developed tools to manage sandbox environments, facilitate live post-copy migration, and ensure that network traffic is consistently routed during the migration process, all while complying with the stringent requirements of 5G networks.

The primary goal of the project is to minimize downtime and maintain high availability during migration. To achieve this, we implemented components such as jailor, configer, sdeamon, sdn, and smanager, which work in concert to set up and manage sandbox environments, automate configuration processes, route network requests efficiently, and provide user-friendly management CLIs. These components have been designed to simplify container management and enhance system scalability, while ensuring process isolation and security.

The methodology we employed centers around post-copy migration, which allows server processes to continue running while memory pages are transferred to the destination virtual machine. This approach, combined with Software Defined Networking (SDN) for request routing, ensures minimal disruption to the end users. Additionally, monitoring tools such as Wireshark will be used to evaluate network performance metrics during migration, ensuring compliance with 5G latency standards.

The remaining work for the project includes implementing the migration process, designing test scenarios, and conducting thorough testing to validate the efficiency and reliability of our solution. The primary success criteria are achieving zero downtime and maintaining response latency within the limits defined by 5G standards.

In conclusion, the progress made so far lays a solid foundation for the successful

completion of the project. We have implemented critical components that provide the necessary infrastructure for container-based migration, and we have a clear roadmap for the next steps. Moving forward, we will focus on testing and fine-tuning the migration process to ensure that it meets the high standards expected in 5G networks, ultimately contributing to the development of efficient, scalable, and flexible Cloud-RAN solutions.

REFERENCES

1. Taşdan, A. B. and S. Sevgican, "5G/CloudRAN", *Boğaziçi University, Computer Engineering Senior Projects*, 2018.

2. Ömer Cihan Benzer and Y. E. Topal, "Container-Founded 5G vRAN Network", *Boğaziçi University, Computer Engineering Senior Projects*, 2022, `https://github.com/ocebenzer/UDP-Simulation`.

3. Chabbouh, O., S. B. Rejeb, Z. Choukair and N. Agoulmine, "A novel cloud RAN architecture for 5G HetNets and QoS evaluation", *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, 2016, `https://api.semanticscholar.org/CorpusID:21304415`.

APPENDIX A: DATA AVAILABILITY STATEMENT

Source code of the COMIC-RAN can be found in the repository:

https://github.com/Simurgan/comicran