Xen Hypervisor Network Management System

Huseyin KARACALI[1*], Nevzat DONUM[2*], Efecan CEBEL[3*]

[1] TTTechAuto Turkey, Software Architect, huseyin.karacali@tttech-auto.com
[2] TTTechAuto Turkey, Embedded Software Engineer, nevzat.donum@tttech-auto.com
[3] TTTechAuto Turkey, Embedded Software Engineer, efecan.cebel@tttech-auto.com
[*] Correspondence: efecan.cebel@tttech-auto.com

(First received January 15, 2023 and in final form March 15, 2023)

Reference: Xen Hypervisor Network Management System. The European Journal of Research and Development,3(1), 71-85.

Abstract

On embedded devices, network traffic management is crucial. One of the most fundamental criteria in projects on which embedded devices are created is connecting devices to one another. Additionally, sharing the internet connection, if there is one, has evolved into a need in today's world. In this project, two different operating systems are running by using hypervisor on the same board. The aim of this study is to connect two operating systems running on Xen Hypervisor using a virtual bridge and share the internet connection. An Embedded Linux distribution created with the Yocto project was run on the host domain (Domain-0 in Xen Hypervisor terminology), and Android Automotive OS was run on the guest domain (DomU in Xen Hypervisor terminology). A special topology has been designed for two virtual domains to communicate. Network management is handled by Embedded Linux, as Embedded Linux runs on Domain-0. Moreover, some scripts have been developed on both domains to implement the designed topology. These scripts enable quick and automatic completion of the required actions. In conclusion, this study describes how to automatically establish a network that provides communication between two virtual domains on Xen Hypervisor running on embedded devices. This paper is applicable to any project, provided the Xen hypervisor framework is used. It offers a reliable and efficient solution to connection needs such as infotainment and cluster systems, especially in systems where communication and internet are becoming more widespread for embedded devices such as automotive.

Keywords: Network, Xen Hypervisor, Embedded Linux, Android Automotive OS, Wi-Fi, Ethernet, Network Topology

1. Introduction

In today's connected world, the ability to efficiently share internet connections between different systems and ensure that these systems can communicate with each other are becoming increasingly important. The integration of two domains on Xen hypervisor has led to a need for a reliable method to provide the required network connections between these systems. To achieve this, one solution is to create several virtual elements, such as network-bridges, interfaces in the host domain in conjunction with the Xen hypervisor technology. This paper outlines a method which meets all the network-related requirements for an embedded project, all on Xen hypervisor. The process involves configuring the network settings on both domains and enabling the sharing permissions on the devices if an internet connection is wanted to be shared. In this project, the host domain, Embedded Linux, system acts as the host, providing ethernet access to the guest domain, Android Automotive OS, device, while the AAOS device can also act as a Wi-Fi hotspot, sharing its internet connection over its virtual ethernet interface. This method is a cost-effective solution as it eliminates the need for additional hardware, has been tested with positive results, and provides a reliable and efficient way to share internet connections as a desirable example of network structure.

2. Materials

2.1. Xen Hypervisor

Xen Hypervisor is an open-source virtualization technology that enables multiple operating systems to run on a single physical server. It provides a layer of abstraction between the physical hardware and virtual machines, offering features such as virtual machine creation, resource management, network and storage connectivity, and live migration of virtual machines for high availability and scalability. Widely used in cloud computing and data center environments, the Xen Hypervisor is supported by a strong community of developers and users. By effectively managing virtualized environments, it allows organizations to make the most of their resources and reduce expenses.

According to Xen Hypervisor terminology, the main source is Domain-0 (Dom0) and all remaining virtual machines are called DomU. It was decided to use Xen Hypervisor with the support provided by the platform manufacturer we used in our project. Embedded Linux with Yocto project as Domain-0 and Android Automotive OS as DomU constitute our structure.

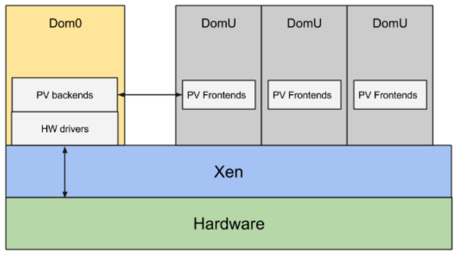

Figure 1:Xen Hypervisor Structure [1]

2.2. Embedded Linux

The use of the Linux operating system in embedded devices, such as Internet of Things (IoT) devices, smart devices, and industrial automation systems, is known as Embedded Linux.[2] The growing popularity of embedded Linux can be attributed to its open-source nature, flexibility, and reliability.

Embedded Linux has several advantages over other embedded operating systems. Firstly, it is equipped with a substantial collection of pre-built software components, libraries, and developer tools, which can be used for a variety of purposes. Second, Linux is highly customizable, allowing developers to configure the operating system to meet their specific needs.[3] Furthermore, the open-source nature of Linux makes the source code freely accessible and modifiable.

2.3. Yocto Project

The Yocto Project is an open-source initiative that aims to provide a framework for building custom Linux-based systems for embedded devices.[4] The project is designed to enable developers to easily create custom distributions that can be tailored to meet their specific needs and needs. The Yocto Project provides a set of tools, templates, and libraries essential for developing custom Linux systems, making it an ideal platform for developers looking to build embedded devices.[5]

The Yocto Project supports multiple hardware architectures, including x86, ARM, and PowerPC. It is therefore a versatile solution for different hardware platforms. In addition, Project Yocto provides support for a wide variety of software packages, including popular open-source software libraries such as the GTK+ UI toolkit, the Qt Framework, the OpenEmbedded build system, and the OpenCV computer vision library.[6]

The Yocto Project is also highly customizable, allowing developers to easily configure the software to meet their specific needs. Developers can choose to include or exclude certain software packages, software libraries, and can even modify the source code of software components to create custom versions. The Yocto Project also provides support for multiple development environments, including cross-development environments, allowing developers to build custom embedded systems on their desktops and then easily deploy them to target embedded devices.[7]

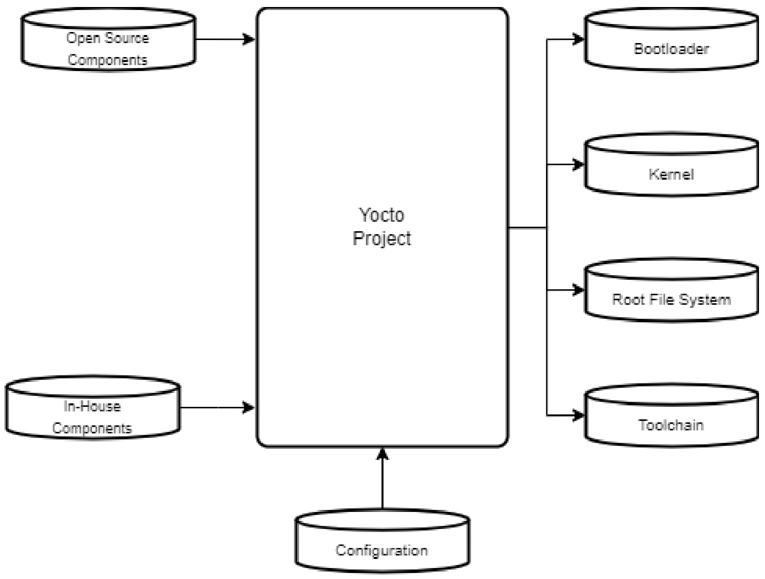

Figure 2:Yocto Project Working Diagram

2.4. Android Automotive Operating System

The Android Automotive Operating System (AAOS) is a specialized version of the Android platform that is specifically designed for use in vehicles. It provides a rich and intuitive interface for automotive infotainment and navigation systems, and allows for seamless integration of smartphone features and services into the vehicle.[8] AAOS includes a wide range of features, such as media playback, navigation, communication, and connected services. It also supports a variety of connectivity options, including Bluetooth, Wi-Fi, and cellular connectivity, allowing for seamless integration with other devices and services.[9] AAOS supports a flexible architecture that enables vehicle manufacturers to customize and personalize the user experience to meet their specific needs. Additionally, AAOS provides a platform for developers to create and distribute automotive-focused applications, further enhancing the capabilities of the system. Overall, the Android Automotive Operating System represents a powerful and versatile platform for automotive infotainment and navigation systems, providing a rich and engaging experience for drivers and passengers.

2.5. Python

Python, a high-level interpreted programming language first introduced in 1991, is well-regarded for its versatility, readability, and simplicity.[10] Its broad standard library, encompassing modules for web server connection, file reading and writing, and data manipulation, adds to its appeal. Furthermore, Python's support for object-oriented, functional, and procedural programming paradigms and its dynamically typed variables that make code testing and debugging more manageable, have made it a highly valued language in scientific computing, data analysis, and artificial intelligence. Its popularity amongst both novice and experienced programmers is also due to its thriving community that has developed a wealth of libraries, tools, and resources. All in all, the ease of use, versatility, and strong ecosystem of Python make it a valuable addition to any programmer's toolkit.

2.6. Network Topology

Network topology refers to the configuration of various elements within a computer network, including nodes, links, and devices. It encompasses both the physical and logical arrangement of these interconnected elements and their relationships with one another.[11] There are several distinct topologies that exist, including star, bus, ring, mesh, and tree topologies, each offering its own set of advantages and disadvantages in terms of ease of maintenance, scalability, and other factors. The selection of a specific topology is contingent upon the particular requirements of the network in question. The network topology exerts a substantial impact on the network's overall performance, including aspects such as reliability, efficiency, and security.[12] Therefore, a thorough comprehension of network topology is imperative for effectively designing and maintaining computer networks.

2.7. Ethernet

Ethernet is a local area network (LAN) technology that provides high-speed data transfer and communication between computers and other connected devices.[13] With its inception dating back to the 1970s, Ethernet emerged as the dominant LAN technology and is widely used today. The protocol that manages the flow of data between connected devices is known as the Media Access Control (MAC) protocol that runs within Ethernet. This technology has a range of data transfer rates from 10 Mbps to 100 Gbps, making it suitable for both small and large network configurations. Ethernet is characterized by its reliability, stability and scalability and is compatible with a wide variety of devices and operating systems. It also supports a variety of topologies, including Ethernet star, bus, and network, and is standardized with the 802.3 standard by the Institute of Electrical and Electronics Engineers (IEEE).[14]

2.8. Wi-Fi

Wi-Fi is a cutting-edge wireless networking technology that enables devices to communicate with one another and access the internet without the need for physical cables.[15] Utilizing radio waves in the 2.4 and 5 GHz frequency bands, Wi-Fi offers data transfer speeds ranging from a few megabits per second to several gigabits per second. The widespread adoption of Wi-Fi in homes, offices, public spaces, and mobile devices has made it a vital component of contemporary connectivity. Setting up Wi-Fi networks can be achieved through routers and access points, and the technology supports various security protocols, including WEP, WPA, and WPA2, for secure data transmission. The technology is standardized by the Institute of Electrical and Electronics Engineers (IEEE) through the 802.11 standards.[16] In essence, Wi-Fi represents a convenient and adaptable technology that affords seamless and fast wireless connectivity, thereby playing a central role in our interconnected world.

2.9. Bridge

A network bridge is a crucial component in the interconnection of networks, allowing devices on one network to communicate with those on another. This device operates at the data link layer of the OSI model and serves as a connection between separate networks.[17] Bridges forward packets between networks through the utilization of Media Access Control (MAC) addresses of network devices, preserving the security and segmentation of individual networks while enabling communication between them. Network bridges can be hardware- or software-based, and they can be applied in both wired and wireless networks.[18] In conclusion, network bridges play a vital role in facilitating communication between devices on different networks, improving network efficiency through the reduction of unnecessary network traffic.

3. Method

The Xen Hypervisor provides a suite of pre-fabricated network tools designed for facilitating communication between two operating systems hosted on its platform. These tools are intended to serve as an aid for developers in the installation of virtual interfaces. They are not equipped to provide a preconfigured network topology. Furthermore, these tools do not possess the capability to effectively manage network traffic and network interfaces.

The Xen Hypervisor Network Topology Management System was developed to manage network interfaces, to achieve port forwarding operations, and to adjust network traffic within a network topology. The system represents a critical solution for ensuring effective management of network resources, network stability and reliability.

In this study, the network communication between the embedded Linux and Android Automotive operating systems on the Xen hypervisor was investigated on Xen Hypervisor. Three distinct topologies were focused for interconnection and internet sharing.

3.1. Prepare Embedded Linux Distro

A custom Embedded Linux distribution was devised utilizing the Zeus version of the Yocto Project pursuant to the specifications. Then integration of the Xen hypervisor into the Embedded Linux distribution has been accomplished.

NXP iMX8QM system-on-chip was used as the test platform. For this reason, an operating system was compiled with Yocto for the iMX8QM SoC.

3.2. Prepare Embedded Linux Distro

Android Automotive OS image suitable for the test platform NXP iMX8QM system-on-chip was compiled using the Android source package. In the project, Android Automotive OS is included as a DomU.

3.3. Embedded Linux – Android Automotive OS Communication

The aim of this study is to establish inter-domain communication between Embedded Linux and Android Automotive OS operating systems on Xen Hypervisor. The Embedded Linux system, called Domain-0 in Xen terminology, serves as the primary domain, and has a physical Ethernet connection. However, the ethernet interface does not have internet connection. A virtual interface called "vif1.0" was created between Dom0 and DomU. This virtual interface was connected to a bridge which was created on Dom0. Communication between the two domains was ensured. Virtual eth0, which is the equivalent of vif1.0, was created on DomU. The configuration of the topology is shown in Figure 3.

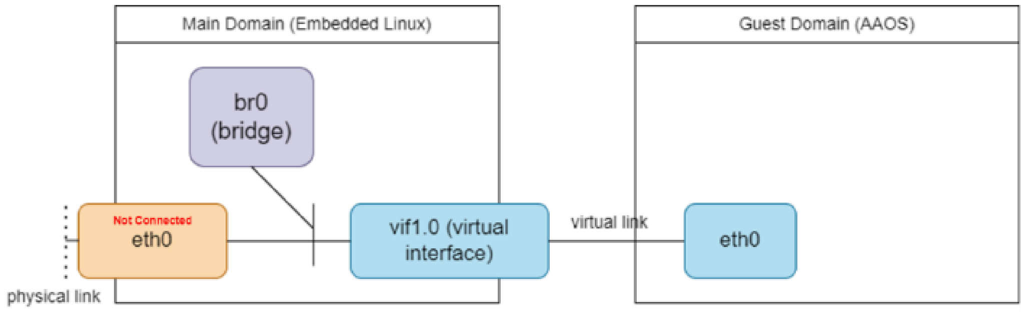

Figure 3:Domain-0 - DomU Communication Topology

IP address is allocated to the virtual bridge established over Dom0. Then, IPs are assigned to both the virtual interface and the virtual ethernet interface. All IP assignments must be within the same IP subnet for seamless communication.

3.4. Internet Sharing to Guest Domain Over Ethernet on Main Domain

Internet sharing over the physical Ethernet interface between Embedded Linux and Android Automotive operating systems running on Xen Hypervisor is carried out by creating virtual networks and interfaces. The internet sharing procedure was carried out over the topology we created in Figure 3. First of all, a virtual bridge is established in Xen Hypervisor, which acts as an intermediary between the two operating systems. Next, a physical Ethernet interface is added to the virtual bridge. A virtual interface referred to as "vif1.0" is established and connected to the bridge. This virtual interface acts as a communication channel between Embedded Linux and Android Automotive operating systems.

A virtual Ethernet interface is created on DomU and configured to set up internet connection. This virtual interface is added to the virtual bridge. It allows to communicate with the Embedded Linux operating system. The Embedded Linux operating system is configured to act as the gateway for the virtual bridge. This setup allows the DomU to access the internet through Dom0, which shares the internet connection over the Ethernet network. The block diagram of the applied topology is shown in Figure 4.

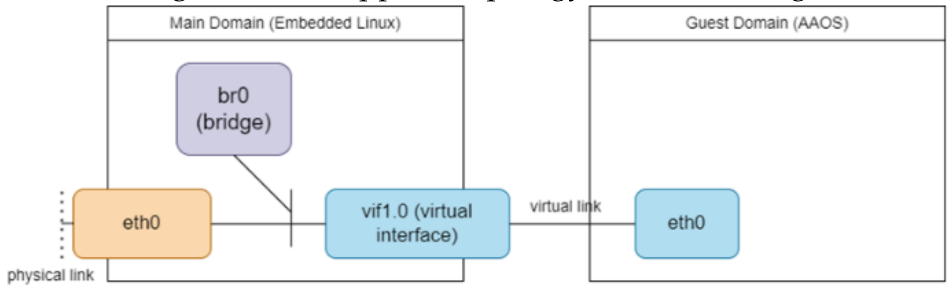

Figure 4:Ethernet Sharing Topology

3.5. Internet Sharing to Main Domain Over Wi-Fi on Guest Domain

The objective of this scenario is to share internet connectivity to Dom0 over the Wi-Fi interface from DomU using a network topology that is akin to the one depicted in Figure 4.

When the physical ethernet interface in Dom0 lacks internet access, a bridge (br0) and a virtual interface (vif1.0) are created on the Embedded Linux system. A virtual ethernet interface (eth0) is also created automatically to connect between Dom0 and DomU. Thereby enabling the sharing of the internet connection over the wlan interface in the DomU.

Port Forwarding process is implemented to achieve successful internet sharing. The gateway IP address of the bridge (br0) in Embedded Linux must be the same as the IP address of the virtual ethernet interface (eth0) in the Android Automotive OS. Therefore, IP address and gateway are assigned properly. Additionally, the "ip_forward" parameter must be enabled for the Port Forwarding process to be successfully executed.

Tethering is initiated over the virtual ethernet interface's IP address on DomU. The NAT process is initiated to allow multiple interfaces to access the internet through a single public address. Routing and port configurations are established using the "iptables" tool; so Wi-Fi sharing procedure is completed.

The block diagram of the applied topology is shown in Figure 5.

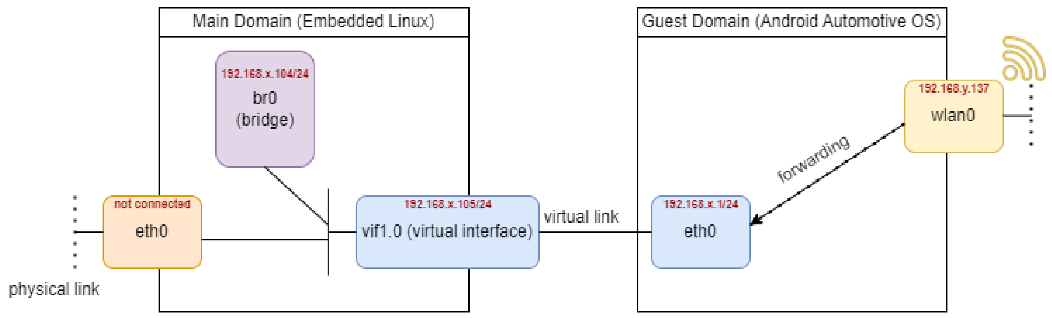

Figure 5:Wi-Fi Sharing Topology

3.6. Xen Hypervisor Network Topology Management System

Xen Hypervisor Network Topology Management System has been created to manage communication between domains. This system was developed via Python programming language. Thus, it executes on all operating systems that utilize the XEN Hypervisor. This innovative tool simplifies the entire process by allowing users to configure all necessary elements for topologies in a single operation. Moreover, the tool is designed to effectively manage network traffic and ensure that all connected domains maintain stable internet connections.

As a result, Xen Hypervisor Network Topology Management System represents a big step forward in creating and managing topologies, making the process more efficient, comfortable, and effective. Its innovative features and compatibility with various operating systems make it a valuable addition to any network infrastructure.

The flowcharts are explained in Figure 6 and Figure 7 how the Xen Hypervisor Network Topology Management System handles ethernet sharing and Wi-Fi sharing.

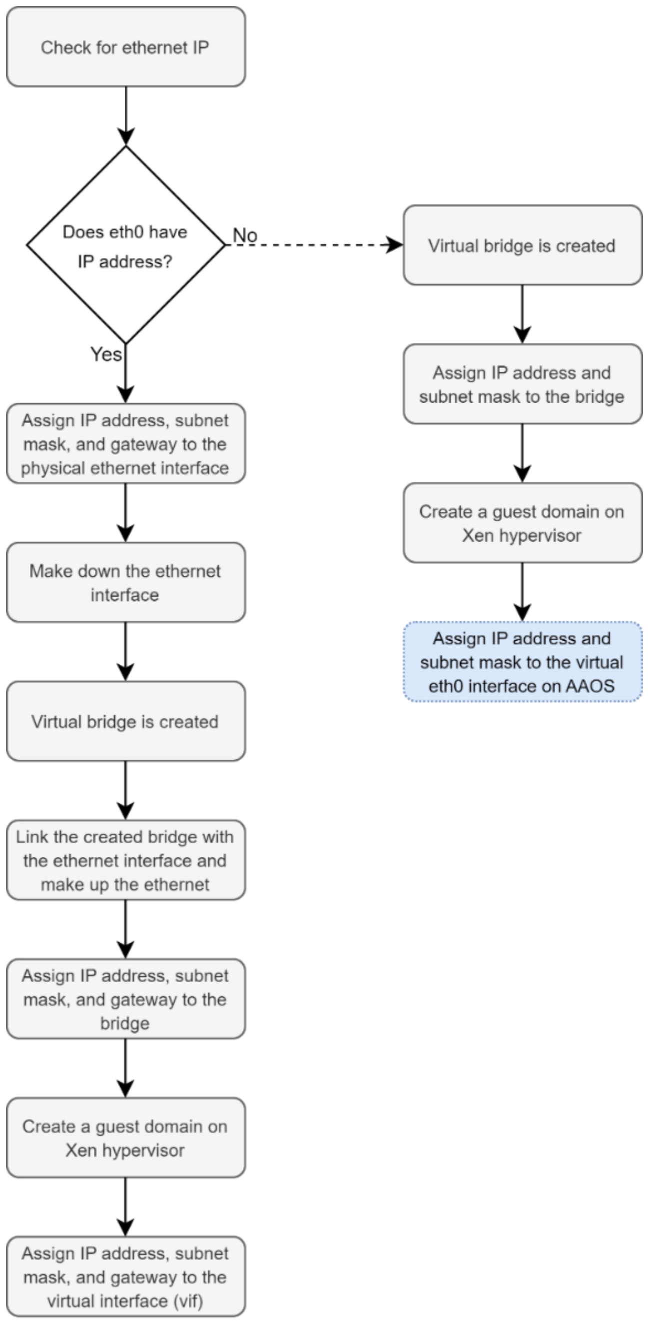
Check for ethernet IP
Does eth0 have IP address?
No
Yes
Virtual bridge is created
Assign IP address and subnet mask to the bridge
Create a guest domain on Xen hypervisor
Assign IP address and subnet mask to the virtual eth0 interface on AAOS
Assign IP address, subnet mask, and gateway to the physical ethernet interface
Make down the ethernet interface
Virtual bridge is created
Link the created bridge with the ethernet interface and make up the ethernet
Assign IP address, subnet mask, and gateway to the bridge
Create a guest domain on Xen hypervisor
Assign IP address, subnet mask, and gateway to the virtual interface (vif)

Figure 6:Ethernet Sharing Flowchart

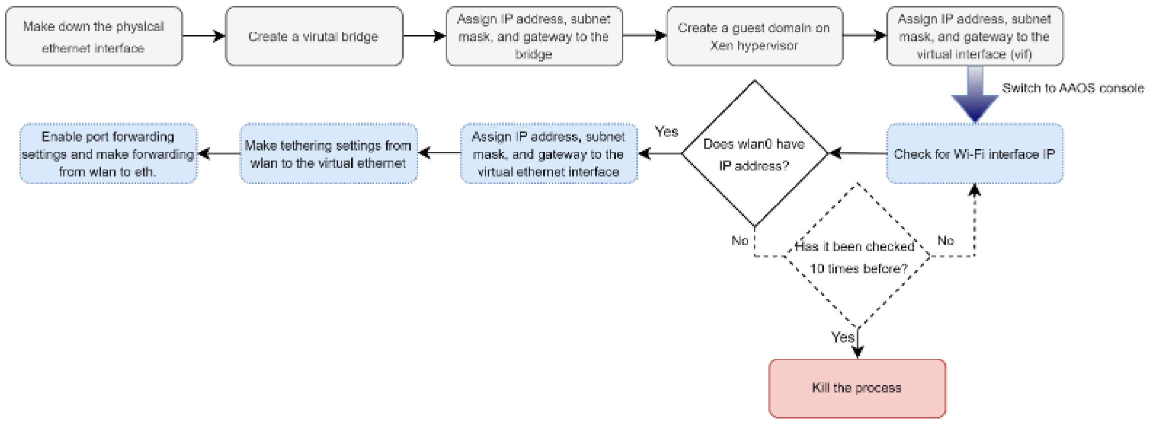
Make down the physical ethernet interface
Create a virutal bridge
Assign IP address, subnet mask, and gateway to the bridge
Create a guest domain on Xen hypervisor
Assign IP address, subnet mask, and gateway to the virtual interface (vif)
Switch to AAOS console
Check for Wi-Fi interface IP
Does wlan0 have IP address?
Yes
No
Assign IP address, subnet mask, and gateway to the virtual ethernet interface
Make tethering settings from wlan to the virtual ethernet
Enable port forwarding settings and make forwarding from wlan to eth.
Has it been checked 10 times before?
No
Yes
Kill the process

Figure 7:Wi-Fi Sharing Flowchart

4. Result

This application, which controls the network traffic between the machines, has been carried out using the above-mentioned materials and methods. It has become a solution to a needed issue by efficiently and reliably managing the network connections between different systems for any domains running on the Xen hypervisor.

Regardless of the platform, these tethering scripts are designed to run anywhere as long as the same operating systems are run on the Xen Hypervisor. As a result of the tests and studies, we can deduce that this method, which we have developed, is low-cost as it does not depend on any extra hardware, easy to implement thanks to its automation, and can be easily used in automotive and embedded systems projects today.

5. Discussion and Conclusion

In summary, Xen Hypervisor Network Topology Management System that manages the networks of domains on Xen Hypervisor has been developed to manage network traffic in embedded systems. At the beginning of this study, the targeted and desired results were fully achieved. If the Xen hypervisor structure is available, it has ability to work independently of the platform. It can take place in embedded and automotive systems projects thanks to these features.

On the other hand, in addition to these features, security is one of the aspects that can be improved for this project. A secure match can be ensured by the authentication of this process while the domains activate the connections between each other. Moreover, encrypting the packages to be sent in communication between operating systems can be an improvement in the scope of security.

Finally, the system can be trained with artificial intelligence. It can dynamically impose the most optimum topology on the system by analyzing the quality and speed of the connections currently in the system after training with artificial intelligence.

Acknowledge

We express our appreciation to our mentor, Hüseyin Karaçalı, the Software Architect, for his inspiration and expert advice. Additionally, we extend our gratitude to TTTech Auto Turkey for their assistance in the creation of this project.

References

[1] "Xen arm with virtualization extensions whitepaper," Xen. [Online]. Available: https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions_whitepaper. [Accessed: 01-Feb-2023].
[2] E. Barbieri, "What is embedded linux? part I," Ubuntu, 24-Nov-2021. [Online]. Available: https://ubuntu.com/blog/what-is-embedded-linux. [Accessed: 01-Feb-2023].
[3] J. Tan, "Embedded linux: A beginner's guide," Latest Open Tech From Seeed, 11-May-2022. [Online]. Available: https://www.seeedstudio.com/blog/2021/01/20/beginners-guide-to-embedded-linux/. [Accessed: 01-Feb-2023].
[4] "It's not an embedded linux distribution – it creates a custom one for you," Yocto Project. [Online]. Available: https://www.yoctoproject.org/. [Accessed: 01-Feb-2023].
[5] "What is the Yocto Project?," Wind River. [Online]. Available: https://www.windriver.com/solutions/learning/yocto. [Accessed: 01-Feb-2023].
[6] "Yocto Project," Toradex Developer Center. [Online]. Available: https://developer.toradex.com/linux-bsp/os-development/build-yocto/yocto-project/. [Accessed: 01-Feb-2023].
[7] E. Wu, "What is Yocto? why should you use Yocto for embedded linux applications," Latest Open Tech From Seeed, 28-Sep-2021. [Online]. Available: https://www.seeedstudio.com/blog/2021/09/23/what-is-yocto-why-shoud-you-use-yocto-for-embedded-linux-applications/. [Accessed: 01-Feb-2023].
[8] "What is Android Automotive? &nbsp;: &nbsp; Android Open Source Project," Android Open Source Project. [Online]. Available: https://source.android.com/docs/devices/automotive/start/what_automotive. [Accessed: 01-Feb-2023].
[9] "Design for driving &nbsp;|&nbsp; google developers," Google. [Online]. Available: https://developers.google.com/cars/design/automotive-os?hl=tr. [Accessed: 01-Feb-2023].

[10] "Welcome to Python.org," Python.org. [Online]. Available: https://www.python.org/. [Accessed: 01-Feb-2023].

[11] A. S. Gillis and T. Nolle, "What is network topology? - definition from searchnetworking," Networking, 23-Aug-2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/network-topology. [Accessed: 01-Feb-2023].

[12] "What is network topology? definition and faqs," What is Network Topology? Definition and FAQs | HEAVY.AI. [Online]. Available: https://www.heavy.ai/technical-glossary/network-topology. [Accessed: 01-Feb-2023].

[13] W. Chai, A. Irei, and J. Burke, "What is ethernet? definition from searchnetworking," Networking, 08-Oct-2021. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/Ethernet. [Accessed: 01-Feb-2023].

[14] C. BasuMallick, "What is ethernet? definition, types, and uses," Spiceworks, 12-Jan-2023. [Online]. Available: https://www.spiceworks.com/tech/networking/articles/what-is-ethernet/. [Accessed: 01-Feb-2023].

[15] "What is wi-fi?: Definition, meaning &amp; explanation," verizon.com. [Online]. Available: https://www.verizon.com/articles/internet-essentials/wifi-definiton/. [Accessed: 01-Feb-2023].

[16] "What is wi-fi? - definition and types," Cisco, 22-Dec-2021. [Online]. Available: https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html. [Accessed: 01-Feb-2023].

[17] Upravnik, "Network bridge explained," Study CCNA, 18-Dec-2022. [Online]. Available: https://study-ccna.com/network-bridge-explained/. [Accessed: 01-Feb-2023].

[18] "What is bridge in computer network - types, uses, functions &amp; differences," GeeksforGeeks, 27-Oct-2022. [Online]. Available: https://www.geeksforgeeks.org/what-is-bridge-in-computer-network-types-uses-functions-differences/. [Accessed: 01-Feb-2023].