Bachelor's Thesis

Update Filtering in BGP-based EVPN VXLAN Setups

by
Alexander Sohn

Supervisors
Prof. Dr. Holger Karl
Internet Technologies and Softwarization

Hasso Plattner Institute at University of Potsdam

August 28, 2023

Abstract

Layer 2 (L2) broadcast is still a problem in large Wi-Fi deployments because it takes up much airtime without benefitting the user [19, 32, 14]. To solve this problem, Schlitt et al. proposed a new Virtual eXtensible LAN (VXLAN)-based Wi-Fi architecture with Border Gateway Protocol Ethernet Virtual Private Network (BGP EVPN) as a control plane [25]. Real-world deployments of this architecture have a high roaming time with many connected devices. In this thesis, I explore how filtering in the BGP EVPN control plane can improve this. I use Route Target Constrained Route Distribution (RTCRD) as a filtering mechanism [17]. To evaluate the real-world impact, I built a prototype implementation of RTCRD based on Free Range Routing (FRR). Experiments with the prototype implementation show that RTCRD can decrease the roaming times from 30 s to 1 s in the Wi-Fi architecture.

Contents

1 Introduction **1**

2 Background **4**
2.1 WiMoVE - Wireless Mobility Through VXLAN EVPN 4
2.2 VXLAN - Virtual eXtensible LAN 4
2.3 VRF - Virtual Routing and Forwarding 5
2.4 BGP EVPN for WiMoVE 5
2.4.1 Border Gateway Protocol (BGP) Basics 5
2.4.2 Explaining the BGP EVPN Network Layer Reachability Information (NLRI) 6
2.4.3 RT - Route Target 7
2.4.4 BGP Topology of a WiMoVE System 7
2.4.5 RTCRD - Route Target Constrained Route Distribution 8
2.4.5.1 Configuring a VXLAN 10
2.4.5.2 Deleting a VXLAN 11

3 Approach **12**
3.1 Problem Analysis 12
3.1.1 Memory Exhaustion in WiMoVE Systems 12
3.1.2 Compute Exhaustion in WiMoVE Systems 12
3.1.3 Link Overload in WiMoVE Systems 13
3.2 Using RTCRD in a WiMoVE System 14
3.2.1 RTCRD Information in a WiMoVE System 14
3.2.2 Effects of RTCRD on Memory Usage 14
3.2.3 Roams With RTCRD Enabled 17
3.2.4 High-Load Scenario With RTCRD 17
3.3 Improvements to RTCRD in WiMoVE 19
3.3.1 Distributing the Load Among Multiple Route Reflectors (RRs) . . . 20
3.3.2 Origin-Based Filtering 20
3.3.3 Roaming Prediction 20

4 Implementation **23**
4.1 FRR as a Base Implementation 23
4.2 Design Decisions for the Implementation 23
4.2.1 Update Groups in FRR 23
4.2.2 Storing RTCRD Information in an Easy to Match Format 24
4.2.3 Challenges Processing RTCRD Updates 24

5 Evaluation **25**
5.1 Ethernet Virtual Private Network (EVPN) Load Generation 25

5.2 Effect of RTCRD on Memory Consumption . . . 25
5.3 Measuring Roaming Times in a WiMoVE System . . . 26
5.3.1 Load Scenarios . . . 26
5.3.2 Tracking Additional Metrics . . . 27
5.3.3 Measuring the Propagation Time for BGP EVPN Updates . . . 28
5.3.3.1 Experiment Setup . . . 28
5.3.3.2 Measurement Methodology . . . 28
5.3.3.3 Using Data Plane Learning . . . 29
5.3.3.4 Results . . . 30
5.3.4 Measuring the Setup Time . . . 31
5.3.4.1 Experiment Setup . . . 31
5.3.4.2 Measurement Methodology . . . 32
5.3.4.3 Results . . . 36

6 Related work **39**
6.1 Wi-Fi Architectures . . . 39
6.2 Other Filtering Mechanisms in BGP EVPN Setups . . . 39
6.3 Roaming Times in BGP EVPN Setups . . . 40

7 Conclusion **41**
7.1 Future Work . . . 42

References **43**

Acronyms

AFI Address Family Identifier

AP Access Point

AS Autonomous System

BGP Border Gateway Protocol

BGP EVPN Border Gateway Protocol Ethernet Virtual Private Network

DHCP Dynamic Host Configuration Protocol

eBGP external Border Gateway Protocol

EVPN Ethernet Virtual Private Network

FDB Forwarding Database

FRR Free Range Routing

iBGP internal Border Gateway Protocol

L2 Layer 2

L2VPN Layer 2 Virtual Private Network

L3 Layer 3

MPLS Multiprotocol Label Switching

NLRI Network Layer Reachability Information

ORF Outbound Route Filtering

RIB Routing Information Base

RR Route Reflector

RT Route Target

RTCRD Route Target Constrained Route Distribution

SAFI Subsequent Address Family Identifier

VM Virtual Machine

VNI Virtual Network Identifier

VPN Virtual Private Network

VRF Virtual Routing and Forwarding

VTEP Virtual Tunnel End Point

VXLAN Virtual eXtensible LAN

1 Introduction

L2 broadcast is still a problem in large Wi-Fi deployments because it takes up much airtime without benefitting the user [19, 32, 14]. For this reason, Schlitt et al. proposed a new Wi-Fi architecture called *WiMoVE*, [25] which is short for Wireless Mobility through VXLAN Ethernet Virtual Private Network (EVPN). WiMoVE applies the idea of overlay networks to the distribution system of Wi-Fi deployments to limit L2 broadcast traffic that has to be transmitted over the air. The difference between WiMoVE and other solutions is that it aims to provide users with fully functional L2 domains simultaneously. Therefore, WiMoVE partitions the connected devices into overlay networks, which are implemented with VXLAN as Layer 2 Virtual Private Network (L2VPN) technology and BGP EVPN as a control plane. The Access Points (APs) terminate the VXLANs, which means they have additional responsibilities compared to regular Wi-Fi deployments. The advantage of partitioning the distribution system is that every AP only has to send out the broadcast packets of VXLANs in which it has a connected device. Schlitt et al. showed that depending on the deployment architecture, this lowers the number of broadcast transmissions over the air by a factor of 500. Another difference is that WiMoVE allows the use of APs from different vendors since it builds upon OpenWrt. This open-source operating system offers images for many Wi-Fi APs [26].

For the L2VPNs to work, every AP needs to know to which AP a device is connected. This information is needed to forward packets to the correct AP for every device. In WiMoVE, this information is called *reachability information*. To distribute the reachability information, WiMoVE uses a control plane. In a WiMoVE system, the speed of the control plane matters when a device roams since it determines how long a device loses connectivity to the Internet and other devices. During a roam, two processes must be completed to restore the connectivity. The updated reachability information that the device is now reachable at another AP has to be distributed to all other APs, and the new AP the device is now connected to has to receive the reachability information for all devices in the VXLAN of the new device. The time until the first process is complete is called *propagation time*, and the time until the second process is complete is called *setup time*. The roaming time is the maximum of those two times. Because the scale of regular Wi-Fi roaming is between 10 ms and 100 ms [1], the control plane must not take longer to update the reachability information to avoid a negative impact on the roaming times.

Compared to conventional VXLAN BGP EVPN deployments, WiMoVE brings some challenges for the control plane. In contrast to Virtual Tunnel End Points (VTEPs) in other deployments, the APs are low-power devices, which means they have limited CPU and memory resources. Because in BGP EVPN, the information for all VXLANs is flooded to all VTEPs, every VTEP has to process all update messages and keep the whole reachability information in memory. Therefore, the load on the APs in WiMoVE scales with the number of all connected devices and the movement in the whole network. That is a problem for WiMoVE deployments with many connected devices. The resources of the APs get

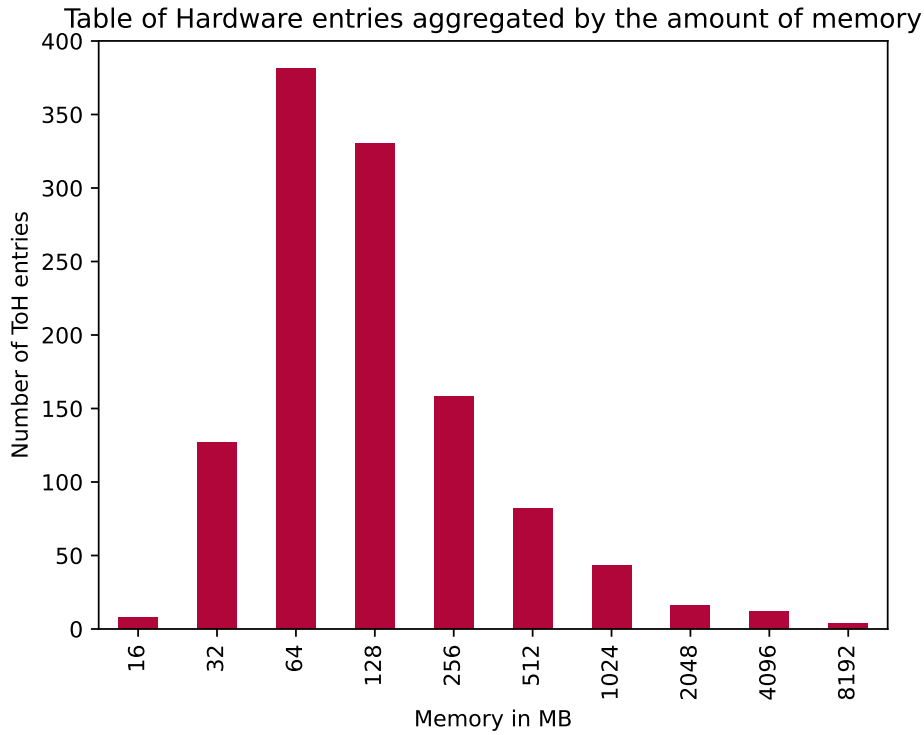


Figure 1.1: Number of devices aggregated by memory size in the OpenWrt Table of Hardware that support the 22.03.5 release. [29].

exhausted just from the load of the control plane. If the APs run out of memory because they have to carry too much reachability information, the Border Gateway Protocol (BGP) daemon crashes, resulting in a loss of connectivity for all devices connected to this access point. In Chapter 5, I show that this happens for an AP with 512 MB of memory at around 450,000 devices. However, most APs supported by OpenWrt have even less memory, as the data from the Table of Hardware in Fig. 1.1 shows [29]. With APs with less memory, the maximum number of concurrently connected devices would be even lower without improvements to the memory efficiency of the control plane. The limited CPU resources mean that an AP can not keep up with processing updates if the update frequency gets too high, which results in a processing delay. The processing delay affects the propagation time. The experiments in Chapter 5 show that the propagation time can increase to 30 s with an overloaded AP in the system. The setup time goes up to 3 s on an overloaded AP.

Schlitt et al. proposed using a filtering mechanism that takes into account the configured VXLANs on an AP and prevents flooding in the control plane. In this thesis, I investigate whether filtering in the BGP EVPN control plane can prevent overload situations and enable WiMoVE to accommodate more devices while keeping the roaming times low. With the filtering mechanism, I achieved that every AP only receives updates for VXLANs configured on the AP. Because every AP only has a few VXLANs configured at a time, this should significantly decrease the number of updates it has to process. In consequence, it should prevent out-of-memory situations and speed up roaming times. As a filtering

mechanism for BGP EVPN, I use Route Target Constrained Route Distribution (RTCRD), which RFC 4684 specifies [17].

In Chapter 2, I explain the technologies and concepts I used. After that, in Chapter 3, I analyze the problems in WiMoVE systems with many devices and high mobility. Then, I show how RTCRD can be used to lower the load on the APs. Because no open-source implementation of RTCRD was available, I describe in Chapter 4 how I built a prototype RTCRD building upon FRR. In Chapter 5, I evaluate how the roaming times in a WiMoVE system change when RTCRD is used. I use the *setup time* and the *propagation time* I introduced in this chapter as metrics for the experiments. The experiments show that using RTCRD decreases the propagation time and the setup time. In high-load scenarios, the propagation time decreased from 30 s to 220 ms, and the setup time decreased from 1.5 s to 1 s.

2 Background

This chapter introduces WiMoVE and the technologies upon which it is built. Additionally, I introduce how Route Target Constrained Route Distribution (RTCRD) works in BGP EVPN because this is the filtering mechanism I will evaluate later.

2.1 WiMoVE - Wireless Mobility Through VXLAN EVPN

WiMoVE is a new Wi-Fi architecture proposed by Schlitt et al. [25]. WiMoVE is designed to reduce the broadcast traffic the APs transmit over the air to save airtime. In regular Wi-Fi systems, the distribution system is one large L2 domain. An AP has to transmit all the broadcast traffic in this L2 domain. WiMoVE divides the distribution system into multiple L2VPNs that span across the APs to get smaller broadcast domains with less broadcast traffic.

L2VPNs are a technique to build isolated L2 networks over underlay networks. To achieve this, L2VPNs use encapsulation. The encapsulation and decapsulation of L2VPN packets happen at the L2VPN endpoints. The L2VPN endpoints might encapsulate packets for themselves and provide access for other devices to the L2VPN by bridging the L2VPN to a local L2 domain.

In WiMoVE, the APs are the L2VPN endpoints and perform the encapsulation and decapsulation for their connected devices. The connected devices can exchange L2 packets with all other devices in the same L2VPN via the AP. When a device connects, the WiMoVE daemon *WiMoVEd* assigns it to an L2VPN and bridges it to the corresponding L2VPN. When a device connects or disconnects, the L2VPNs grow and shrink due to the reconfiguration by WiMoVEd.

The devices also need connectivity to IP networks outside the WiMoVE system, for example, the Internet. For this purpose, there is a separate gateway. The gateway is part of the L2VPN and is the next hop for all Layer 3 (L3) packets leaving the WiMoVE system.

To reach other devices in the L2VPNs, the APs and the gateway need to know where a device is connected. Every participant of the L2VPNs has to maintain a mapping between the MAC address of a connected device and the address of the endpoint to which it is connected. In Chapter 1, I introduced this mapping as *reachability information*. Schlitt et al. propose a control plane to exchange this mapping. An implementation of this control plane that fulfills their requirements is BGP EVPN.

2.2 VXLAN - Virtual eXtensible LAN

RFC 7348 [15] specifies the L2 encapsulation technology Virtual eXtensible LAN (VXLAN) for L2VPNs, which WiMoVE uses. The Virtual Private Network (VPN) endpoints in a

VXLAN are called Virtual Tunnel End Points (VTEPs). VXLAN allows for multi-tenancy, meaning that a VTEP might not only be part of one VXLAN but can be part of multiple. To distinguish received packets, every VXLAN is identified by a Virtual Network Identifier (VNI), which is a 24 bit number. The multi-tenancy is important for WiMoVE since multiple devices with different L2VPNs might be connected to a single AP. Every VXLAN is a separate L2VPN.

2.3 VRF - Virtual Routing and Forwarding

In Section 2.2, I explained that one VTEP might be part of multiple VXLANs due to multi-tenancy. Those VXLANs share the same address space, meaning the same MAC address might exist in different VXLANs. For example, the L2 broadcast address exists in every VXLAN. Nevertheless, for different VXLANs, a packet with the broadcast address should be sent to different other VTEPs. The operating system on a VTEP needs a way to discern the stored reachability information for different VXLANs to support multi-tenancy. Linux uses Virtual Routing and Forwarding (VRF) for this purpose [8]. VRF means multiple routing and forwarding tables exist independently on a host. With VRF, it is possible to forward broadcast frames for different VXLANs to different VTEPs.

In WiMoVE, there is a 1:1 mapping between a VXLAN on a VTEP and a VRF. When configuring a new VXLAN on a VTEP, a new VRF gets created, and when the VXLAN gets deleted on a VTEP, the VRF gets deleted.

2.4 BGP EVPN for WiMoVE

In this section, I describe how BGP EVPN works as a control plane for VXLAN L2VPNs in WiMoVE systems. I will keep this introduction short and focus on the core concepts necessary to understand Route Target Constrained Route Distribution (RTCRD) later. I recommend reading the referenced RFCs to better understand BGP EVPN.

2.4.1 BGP Basics

RFC 4271 specifies Border Gateway Protocol (BGP) [21], which is a routing protocol initially designed to exchange IPv4 routing information between the Autonomous System (AS) boundary routers. In this context, an AS is an administrative domain that appears as one homogenous network to the outside. An interior routing protocol is needed to provide this homogenous network and forward traffic within an AS. This can be any routing protocol, including BGP. To differentiate between the usage of BGP as a routing protocol between ASes and within an AS, we talk about external Border Gateway Protocol (eBGP) and internal Border Gateway Protocol (iBGP). In this thesis, I focus on iBGP since WiMoVE only uses iBGP, and there is no need for multiple ASes since the whole network is supposed to be under our administrative control.

Every participant of a BGP network is called a BGP speaker; a connection between two BGP speakers is called peering. Two BGP speakers that have an active peering are called peers. In Fig. 2.1, the message flow of a BGP peering between two BGP speakers is shown. Every BGP peering starts with two *open* messages. In those messages, the BGP speakers

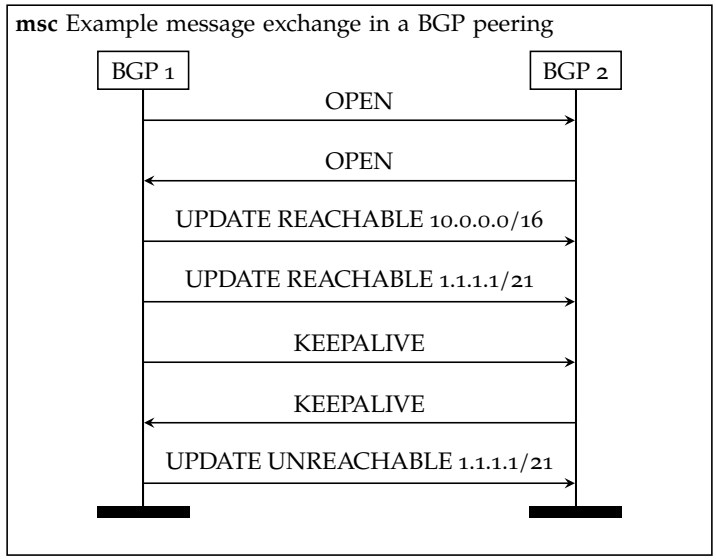


Figure 2.1: Example BGP peering with exchange of basic BGP message types between two peers

exchange information about their capabilities and negotiate the setup of the peering. After successfully establishing a connection, the two BGP speakers can start exchanging routing information. This is done via *update* messages. With an update message, a BGP speaker can advertise new routing information or withdraw old. The routing information in a BGP update message is encoded as Network Layer Reachability Information (NLRI). An NLRI consists of an address and the length. For IPv4, this corresponds to the CIDR notation. So, an NLRI might be 10.0.0.0/16. The address part is 10.0.0.0, and the length is 16. For a BGP speaker to discern between advertising and withdrawing, an NLRI has an attribute that determines if the NLRI is reachable or unreachable. When saying a route is advertised, an update message with a reach NLRI that encodes this route is sent out. When saying a route is withdrawn, an update message with an unreachable NLRI that encodes this route is sent out. In the context of BGP, an update message is often just called an update. The last message type is *keepalive* messages. BGP speakers exchange those periodically to know whether a peer is still reachable. If another BGP speaker is deemed unreachable via this mechanism, all routes received from this BGP speaker get withdrawn.

The routing information a BGP speaker receives is stored in the Routing Information Base (RIB). The RIB contains all the routing information contained in update messages. This routing information gets then installed into the routing and forwarding tables of the BGP speaker.

2.4.2 Explaining the BGP EVPN NLRI

BGP was not designed to exchange the mapping between the MAC address and the VTEP address that the control plane of an L2VPN needs. To adapt BGP to different address families, RFC 4760 introduces the Multiprotocol extensions [3]. An NLRI still carries an address and a length, but also an Address Family Identifier (AFI) and a Subsequent Address Family

Identifier (SAFI) to determine to which address family the NLRI belongs. Depending on the AFI and SAFI, the NLRI address has different semantics.

Building upon the multiprotocol extensions, BGP can be used to build a control plane for L2VPNs. RFC 7432 [9] defines the route types that are needed to build the control plane for L2VPNs with Multiprotocol Label Switching (MPLS). RFC 8365 describes the specific semantics to combine BGP with VXLAN instead of MPLS [23]. The important route type for WiMoVE is the MAC address advertisement route type. The NLRI for MAC address advertisement routes contains a MAC address and the VNI. The semantic is that the MAC address is reachable at the VTEP originating the route. With those routes, it is possible to use BGP EVPN as a control plane for WiMoVE. The APs and the gateway can now exchange the mapping between the MAC address and the VTEP address by exchanging BGP EVPN routes.

2.4.3 RT - Route Target

In Section 2.3, I introduced the concept of VRF and that there is a 1:1 mapping between VXLAN configured on a VTEP and a VRF. What is still missing is the link between routes in the RIB of a VTEP and how they get into the VRFs on the VTEP. The easiest approach would be to maintain a mapping between the VNI and the VRF, such that every route that carries the VNI gets imported into the VRF for that VNI. This limits what is possible with the BGP EVPN control plane. Two VXLANs that share the same VNI could not be isolated in the control plane because there is no way to control which VRFs import the route. For example, in the scenario shown in Fig. 2.2, devices connected to VTEP 3 and VTEP 4 should not be able to communicate with devices connected to VTEP 1 and VTEP 2 and vice versa, although the VXLANs use the same VNI. For this to work, the VXLANs must be kept separate in the control plane. To achieve this separation, BGP EVPN uses Route Targets (RTs) to define the import and export policy of a VRF. RTs are a BGP extended community defined in RFC 4360 [30]. A BGP extended community is an attribute that gets sent with a route. A route can carry many extended communities. The import and export policy of each VRF is a list of RTs to import and a list of RTs to export. A route exported from a VRF is sent out with the whole export list of the VRF. For every route a BGP speaker receives, it will check if the RT matches the list of imported RT of a VRF. If so, the BGP speaker will import the route into the VRF. In WiMoVE, a VNI is unique, and there is a 1:1 mapping between VNI and RT. Because this is common for BGP EVPN VXLAN setups, RFC 8365 defines a procedure to auto-derive RT [23]. WiMoVE uses RTs generated via this procedure.

2.4.4 BGP Topology of a WiMoVE System

Until now, we have focused on the communication between two BGP speakers and how they can exchange and store information. However, for WiMoVE, not only do two BGP speakers have to exchange routes, but all APs and the gateway need a way to exchange BGP messages to keep the RIBs in sync. Schlitt et al. use a topology with a Route Reflector (RR) for WiMoVE. RFC 4456 introduces RRs [4]. As the name implies, an RR reflects received routes to all route reflector clients. Figure 2.3 shows a BGP topology with three VTEP and one RR. The solid lines are the BGP peerings. Every BGP update message received by an RR gets sent to all other VTEPs. This means all BGP EVPN routes get flooded to all

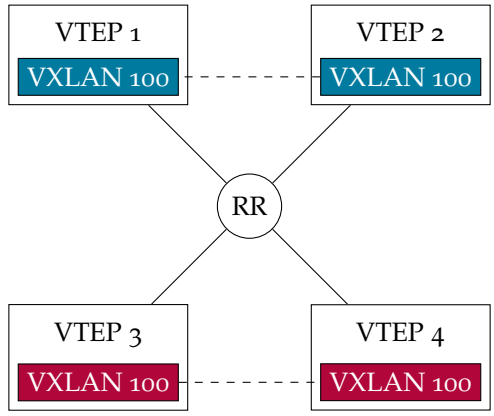


Figure 2.2: VXLAN setup that reuses the same VNI for two VXLANs that should be isolated from each other

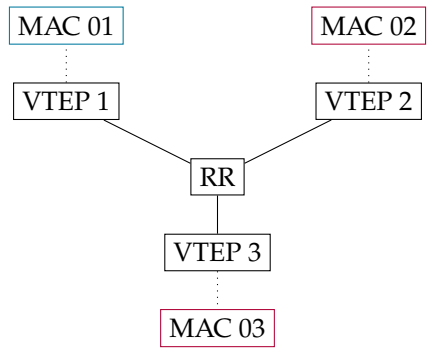


Figure 2.3: BGP EVPN setup with three VTEPs that all have a peering with the RR

VTEPs. This way, all VTEPs receive the same update messages and have the same RIB. An advantage of using an RR is that every AP only has one BGP peering. Additionally, it makes the configuration of new APs easier since only the address of the RR has to be known to establish the BGP peerings. Because this is the BGP topology used by WiMoVE, I will focus in this thesis on BGP topologies with one RR and multiple VTEPs.

2.4.5 RTCRD - Route Target Constrained Route Distribution

Marques et al. recognized that the VTEPs carry unnecessary routes in their RIB [17]. Let us look at the following example to show what Marques et al. mean with unnecessary routes. In Fig. 2.4, the RIB of the BGP speakers in a deployment with one RR and three VTEPs is shown. The rows in the RIB are colored depending on the RT. We remember from Section 2.4.3 that RTs are used to define import policies for VRFs. VTEP 1 only has a VRF importing routes with the blue RT but receives routes with the red RT nevertheless. VTEP 2 and VTEP 3 have routes with the blue RT in the RIB, although only a VRF that imports routes with a red RT exists. We define that a route is unnecessary at a VTEP if no VRF exists that imports routes with that RT. In Fig. 2.4, I highlighted the unnecessary routes in italic face. The blue routes do not have to be distributed to any other VTEP, but VTEP 2

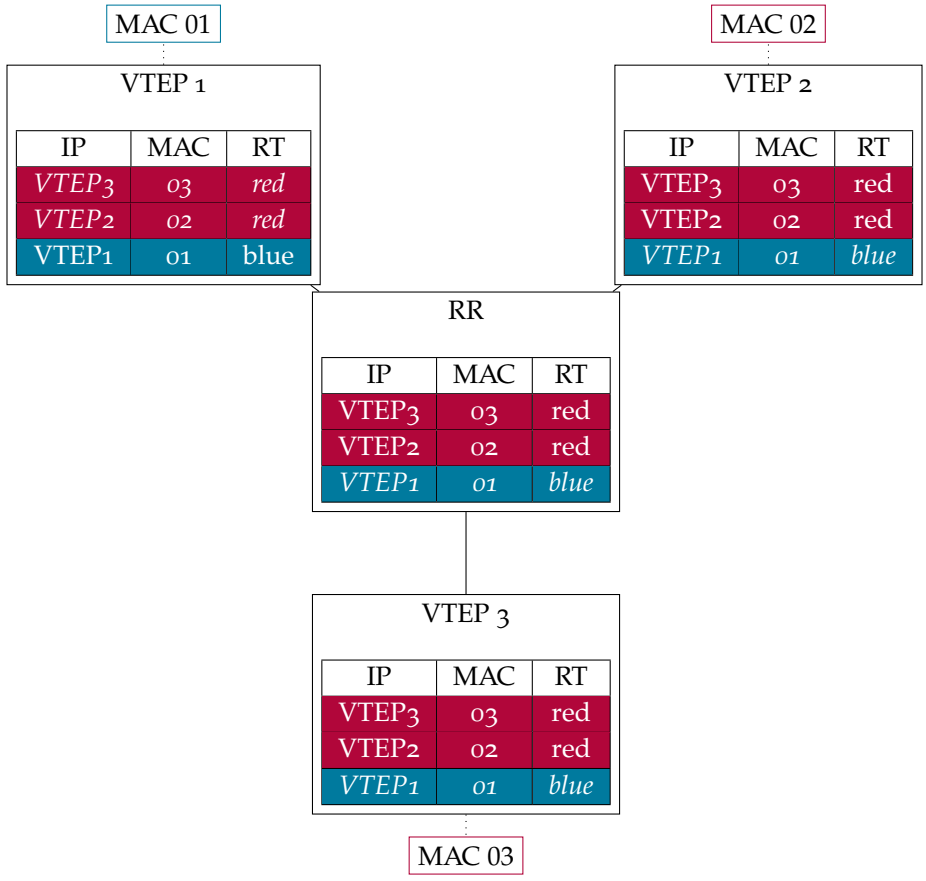


Figure 2.4: Visualization of the RIBs in a BGP setup without filtering

and VTEP 3 have to exchange red routes via the RR. It is possible to set up RT-based filters for the BGP EVPN routes on the VTEPs and the RR in a way that only necessary routes are advertised to the VTEPs and the RR when the import policy of every VTEP is known. However, the import policies of the VTEPs can change over time, for example, when a new VRF gets configured. Instead of manually updating the filters every time, Marques et al. propose in RFC 4684 to use a control plane to distribute the import policies of the VTEPs and use this information to build the RT-based filters [17].

Rather than using a separate control plane, the existing BGP peerings are used to exchange the import policies of the VTEPs. For this purpose, RFC 4684 defines a new address family with the SAFI 132 [17]. With this address family, the BGP speakers can advertise the RTs imported by a VRF. In our example, the VTEPs advertise that they import blue routes or red routes. Every RTCRD NLRI carries an RT and the length. The NLRI carries the length, so an NLRI can express a range of RTs. Because in BGP, the RT is a bit value and not a color, the length indicates how many leading bits of the RT must match. For example, the NLRI with length zero matches all other RTs and is called the default RT. Based on the received NLRIs, every BGP speaker can construct outgoing filters for BGP EVPN routes. The default policy for those filters is deny, and every RT that matches a received NLRI is allowed.

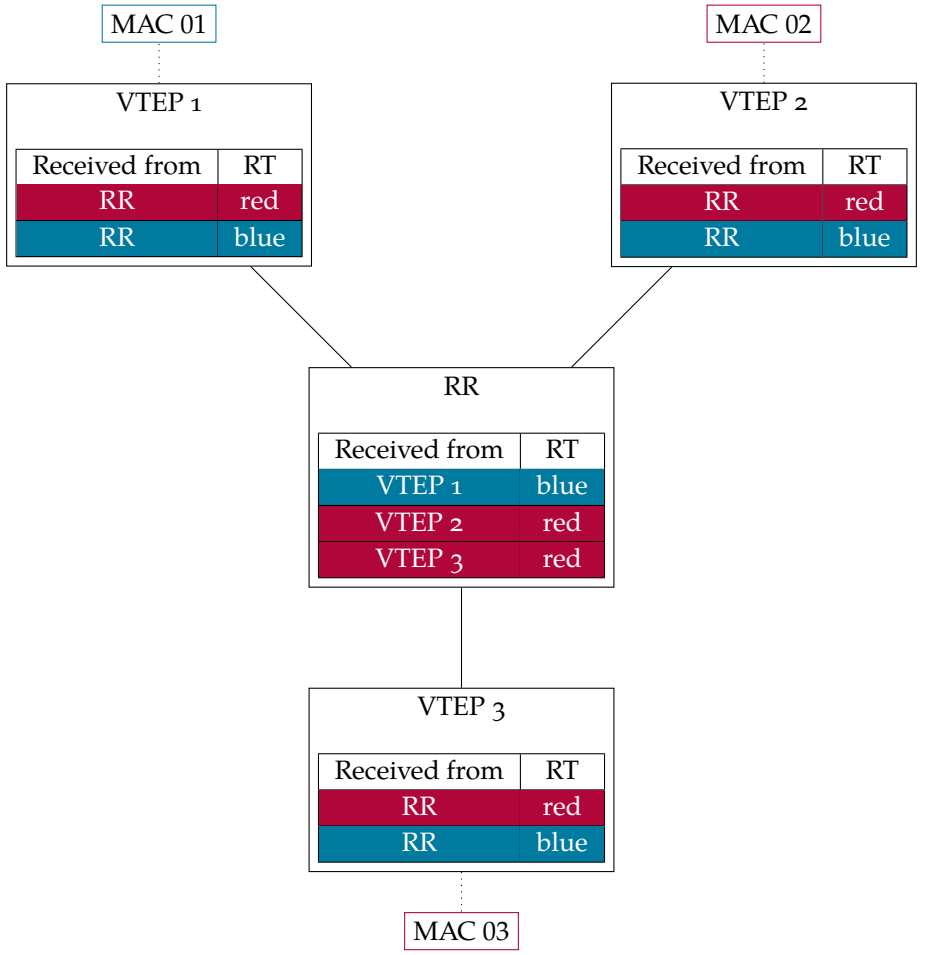


Figure 2.5: RIB entries for the RTCRD address family in the example setup.

Fig. 2.5 shows the distributed RTCRD routes for our example. What is apparent is that in topologies with multiple VTEPs connected to one RR, all filtering happens on the RR. This is because the RR reflects all RTCRD routes back to the VTEPs, so as soon as one VTEP has an RT in its import policy, all VTEP will receive an RTCRD route with this RT, including the originating VTEP itself. In our example, the RR has routes with the blue RT, although no other VTEP receives those routes.

2.4.5.1 Configuring a VXLAN

When the import policy of a VTEP changes, the BGP speakers exchange new RTCRD routes. The import policy changes when on a VTEP a new VXLAN and the corresponding VRF gets created. In this case, the AP sends an update that it is interested in the new RT. The RR reprocesses the RIB and sends out all routes with that RT that previously were not announced. Figure 2.6 shows the message exchange when VTEP 1 from our example in Fig. 2.4 configures a VXLAN with RT red. VTEP 1 sends out an RTCRD update message

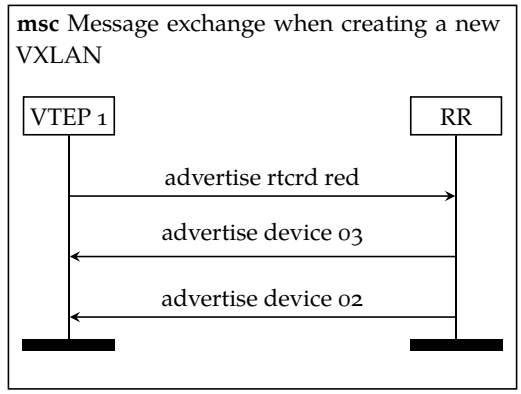


Figure 2.6: Message exchange when creating a new VXLAN with RTCRD enabled

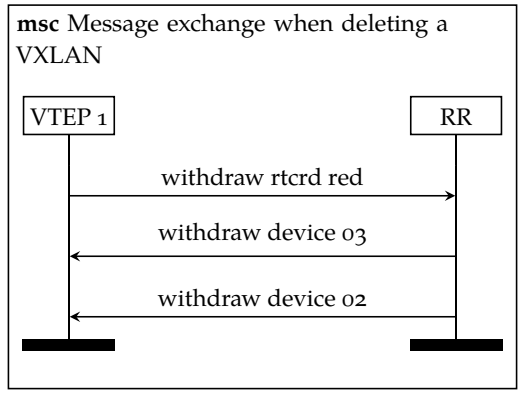


Figure 2.7: Message exchange when deleting a VXLAN with RTCRD enabled

with RT red. The RR will advertise the routes for devices 03 and 02 to VTEP 1 and reflect the updated membership information to VTEP 1.

2.4.5.2 Deleting a VXLAN

The import policy of a VTEP can not only change when a new VXLAN gets added but also when a VXLAN gets deleted on the VTEP. When the VXLAN gets deleted, the VTEP withdraws the RTCRD route with this RT. When the VTEP withdraws the route, the RR checks which routes it previously announced to this VTEP that are now filtered. Then, the RR withdraws the filtered routes, and the AP deletes the routes from its RIB to free up memory and avoid operating on outdated information if it configures a VXLAN with the same RT later on. Figure 2.7 shows the message exchange when VTEP 1 from our example deletes the VXLAN with RT red. VTEP 1 withdraws the RTCRD route with RT red. The RR withdraws the routes for MAC 02 and MAC 03 from VTEP 1.

3 Approach

In this chapter, I analyze the problems in WiMoVE systems with many devices and high mobility. Then, I explain how Route Target Constrained Route Distribution (RTCRD) works in WiMoVE systems and affects the roaming process. Lastly, I evaluate if RTCRD can mitigate the problems in high-load situations.

3.1 Problem Analysis

In Section 2.4.4, I explained why the BGP EVPN update messages get flooded between the VTEPs, and in Chapter 1, I showed that this could be problematic in a WiMoVE system because the APs can not handle the load of the control plane. In this section, I will analyze the effects of high-load situations in a WiMoVE system and the problems that arise in more detail.

3.1.1 Memory Exhaustion in WiMoVE Systems

Because BGP EVPN uses flooding for every connected device in the system, every AP has an entry in its RIB to store the received information. This means the memory consumption on the APs is proportional to the number of connected devices, so the memory resources get exhausted when the number of devices is too high. If the AP runs out of memory, all connected devices lose connectivity since their packets can not be forwarded anymore. When the BGP daemon restarts, the RR will try to send all routes to the AP once again. If there is still the same number of devices connected, the BGP daemon will crash again, and the service is still interrupted. This means a hard limit for the number of devices in a WiMoVE deployment that depends on the available memory on the APs. To improve this, the size of the RIB on every AP must not scale with the number of all connected devices.

3.1.2 Compute Exhaustion in WiMoVE Systems

Not only the memory resources might be exhausted, but the computing resources can be as well. The number of devices does not directly affect the usage of computing resources on the AP. However, the AP does need the computing resources to process updates. Roams in a WiMoVE system generate updates because after a roam, a device is connected to a different AP than before, and the reachability information has to be updated. Every roam generates two update messages: One when the device disconnects from an AP and one when the device connects to an AP. This means the usage of computing resources is proportional to the mobility in the WiMoVE system, which means many roams can lead to an exhaustion of computing resources on the APs. If this happens, the APs can not keep up with processing the updates. This results in a partial loss of connectivity because the reachability information

gets outdated, and connected devices can no longer send packets to devices with outdated reachability information.

Now, I introduce a scenario that I use to illustrate the problem. The scenario is a WiMoVE system with 20,000 APs and 400,000 devices. I chose 400,000 devices because this is close to the hard limit I got with the experiments I conducted in Section 5.2. Every AP has 20 devices connected. I chose 20 devices per AP because it is realistic that a commodity AP can handle 20 devices. For the size of the VXLANs, I chose two devices per VXLAN because Schlitt et al. calculated an expected value of 1.58 for the size of the VXLANs if the number of VXLANs equals the number of devices [25]. Every VXLAN has a gateway as an additional third device. For the analysis, I chose a scenario with high mobility to emphasize the effects. Every device roams every 25 s, and the roams are equally distributed. This results in 16,000 roams per second. Because every roam causes two updates, the APs send 32,000 update messages per second. The RR receives all those updates and has to reflect them to every AP and the gateway. For every received update, the RR has to send out $20,001$ update messages. One for every AP and one for the gateway. The RR has to send out $32,000 \cdot 20,001 = 640,032,000$ updates per second. Every AP has to process $32,000$ updates per second.

If the APs, the RR, and the gateway can keep up with processing these messages, this does not impact the propagation time. However, if the processing power of the APs is insufficient, the propagation time will increase. Suppose the processing rate of an AP is lower than 32,000 updates per second and, for example, 20,000 updates per second, and the processing rate of the RR is 100,000 updates per second since it has more computing resources. The RR receives less than the maximum processing rate and can forward all updates to the APs. The APs can not keep up with processing all the updates. Every second, an AP falls 12,000 updates behind. To handle spikes in update frequency, the AP has an input queue that holds up to 100,000 updates. This fills up within 8.3 s. The RR additionally has an output queue for every peer with up to 100,000 updates. This fills up in another 8.3 s. When both queues are full, it takes 10 s until an update that is enqueued at the RR is processed by the overloaded AP because all 200,000 updates that are already enqueued have to be processed. The 10 s are now a lower bound for the propagation time because an update issued by an AP can not be processed faster by all other APs. If the load on the system is lower again, the APs can catch up, and the propagation time will decrease.

Another problem is how the RR handles a full output queue. In the current implementation, the AP will be rescheduled for update generation from the RIB. This means the RR stores which routes have already been advertised to the AP. When the AP gets scheduled for update generation, the RR checks which routes that are currently in the RIB have not been advertised to the AP. This ensures that no updates get dropped, but additionally, to the time in the queue, it takes time until an update is generated from the difference between the RIB and the advertised routes. Also, the selection process in which updates are enqueued might be unfair. For example, the RR might go through the RIB in an arbitrary order, which might not be the order in which the RIB entries were received.

3.1.3 Link Overload in WiMoVE Systems

Not only the BGP speakers might experience an overload. The links between the BGP speakers can also be overloaded. The link that experiences the highest load is the one to

which the RR is connected. This is because the RR sends out every update an AP receives. Also, the amount of incoming traffic is multiplied by the number of RR clients. In the above scenario, the RR has to send out 640,032,000 updates per second. A reachability update is at least 21 B in size to accommodate the MAC address, the VNI, the RT, and the IP of the VTEP. This results in 107.525 376 Gbit s^{-1} of update traffic without the overhead of Ethernet and IP. If the link of the RR can not provide this data rate, the roaming times will also increase.

3.2 Using RTCRD in a WiMoVE System

In the previous section, I described the problems in WiMoVE systems when overload situations occur. One possible solution to those problems could be Route Target Constrained Route Distribution (RTCRD). I explained RTCRD in Section 2.4.5. RTCRD restricts the distribution of BGP EVPN update messages and could, therefore, reduce the load on the APs.

3.2.1 RTCRD Information in a WiMoVE System

In Section 2.4.5, I explained how RTCRD works. It uses BGP as a control plane to distribute the import policies of the VTEPs. Figure 3.1 shows what the distributed RTCRD information looks like in a WiMoVE system. What is special about the setup is that all VXLANs are configured on the gateway, even with no device connected. Schlitt et al. say the number of VXLANs should match the expected maximum number of devices to ensure that the VXLANs are small [25]. This means that the number of RTCRD RIB entries on the APs is the maximum expected number of devices. That is a problem because, with 400,000 RIB entries necessary in our example, the APs might already run out of memory. It would be advantageous to aggregate the RTCRD RIB entries into fewer routes to improve this.

I propose to advertise the default RT from the RR to the VTEPs. It is the NLRI of the RTCRD address family with length zero that matches all other RTs. As I explained in Section 2.4.5, the RR has to carry the whole RIB even with RTCRD enabled because the filters only affect the routes the RR sends out. So, allowing the VTEPs to send all routes to the RR does not increase the load on the RR. With the default RT advertised, the number of RTCRD RIB entries on the APs no longer scales linearly with the number of VXLANs. Figure 3.2 shows the RIBs in this scenario.

3.2.2 Effects of RTCRD on Memory Usage

In Section 3.1, I stated that the size of the RIB on an AP must not be dependent on the number of all connected devices. This is the case with RTCRD. The number of RTCRD RIB entries does not scale with the number of all connected devices, and neither does the number of BGP EVPN RIB entries. The AP only has RIB entries for the VXLANs it has connected devices in. This resolves the hard limit for the number of connected devices. The new limit is now for the number of connected devices in the VXLANs configured on the AP. In the worst case, an AP has a connected device in every VXLAN RTCRD does not help. But this is very unlikely in a WiMoVE system. Schlitt et al. calculated that the expected number of devices in a VXLAN is 1.58. So, at least half of all devices must be connected to

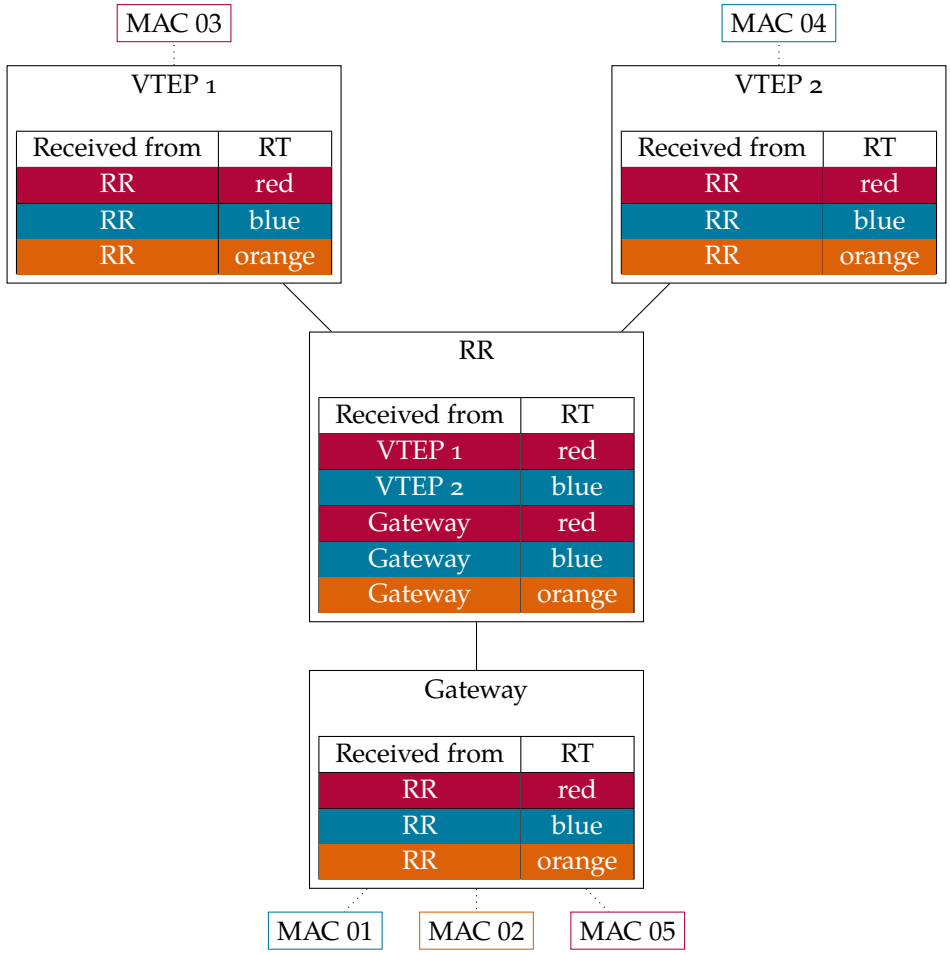


Figure 3.1: RIB entries for the RTCRD address family in a WiMoVE system

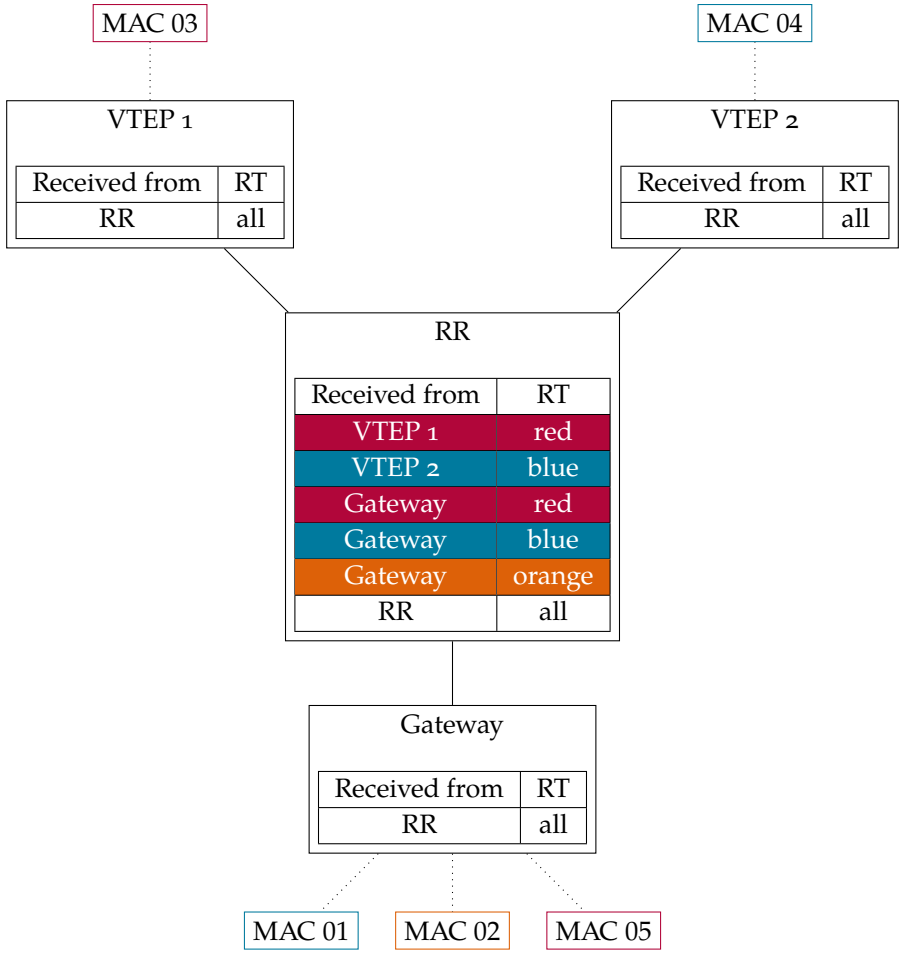


Figure 3.2: RIB entries for the RTCRD address family in a WiMoVE system when the RR uses the default RT

the AP to have one device in each VXLAN. Because an AP can handle a maximum of 2007 stations due to limitations in the 802.11 standard, the AP is unlikely to run out of memory [11].

3.2.3 Roams With RTCRD Enabled

Using RTCRD affects the message exchange when a roam happens. In Section 2.4.5.2 and Section 2.4.5.1, I explained how the message exchange works when a VXLAN gets deleted or created. In WiMoVE, VXLAN interfaces might get deleted or created when a device roams. A new VXLAN interface gets created when the device is the first device of that VXLAN connected to this AP. The VXLAN interface gets deleted when the device roams away and is the last device connected to this device in the VXLAN. In those cases, the AP has to update the RTCRD information and send update messages to the RR. Figure 3.3 shows the message exchange, including the BGP EVPN update messages.

In this scenario, device 2 roams from AP 1 to AP 2. AP 3 has a device in the same VXLAN called device 1. In this scenario, device 2 is the first device in the VXLAN to connect to AP 2 and the last device to disconnect from AP 1. The blue messages are there because device 2 was the last in the VXLAN connected to AP 1, and the red ones are there because device 1 is the first one connected to AP 2. Those messages are there because the RIBs of the APs need to be updated to comply with the new import policies. The blue messages withdraw the routes with an RT AP 1 no longer imports, and the red messages advertise the routes for the RT AP 2 now imports. For an individual roam, there is significant overhead on the number of messages. Six additional messages were sent between the RR and the participating APs. Without RTCRD, only the black messages would be necessary to update the RIBs. Furthermore, RTCRD also affects the roaming times. The blue messages are not critical for the roaming time of device 2, but the red messages have to be delivered for the roam to be complete. Until the red messages are delivered, device 2 can not reach other devices in the VXLAN. AP 2 has to wait until the BGP EVPN routes reach it. Without RTCRD, the BGP EVPN routes are already locally available on AP 2. Therefore, I expect the setup time to increase in a low-load scenario, which means the roaming time will increase.

3.2.4 High-Load Scenario With RTCRD

In the last section, I investigated the effects of RTCRD on a single roam and saw that using RTCRD is an additional effort and might worsen the roaming time. Now I will analyze the high-load scenario I introduced in Section 3.1 with RTCRD enabled. To do this, I want to examine how the load on the RR, the AP, and the links changes. The number of roams is the same: 16,000 per second. But the updates generated are not the same. Fig. 3.3 shows the message exchange that happens during one of those roams. It is the same message exchange I analyzed in the previous section. This shows the strength of RTCRD in high-load situations. It limits the participant of the message exchange to the BGP speakers with the VXLAN configured where the roam happens. Instead of all 20,000 APs, only five entities are part of the message exchange. The APs between the device roams, the AP the other device is connected to, and the gateway. This reduces the number of updates the RR has to send.

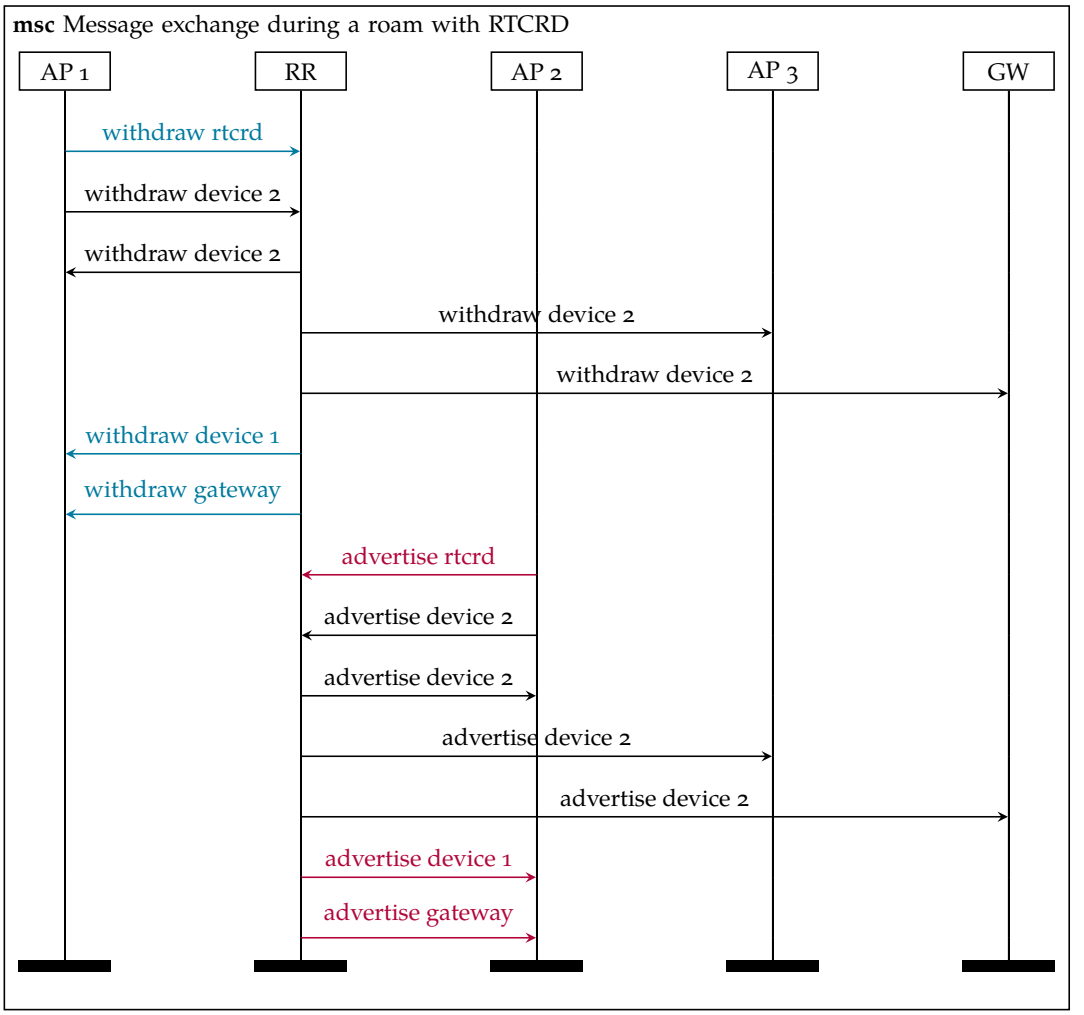


Figure 3.3: The message exchange during a roam with RTCRD enabled

In Fig. 3.3, we can see that the RR has to send out ten update messages for each roam. With 16,000 roams per second, this results in 160,000 update messages per second, which is less than the 640,032,000 update messages in the same scenario without filtering. But, the number of received messages on the RR is higher than without filtering. For every roam, the RR receives four messages instead of only two. This results in 64,000 update messages per second, twice as much as without filtering. Using RTCRD affects not only the RR but also the APs. The number of updates an AP has to process is lower. Because of the filtering, an AP only has to process update messages of VXLANs it has connected devices in and the update messages when a VXLAN gets created or deleted on the AP. In the scenario, every AP has 20 devices connected at any time. Because every device roams every 25 s, every 1.25 s, one of the connected devices roams to another AP, and another device roams to this AP. During one change of a connected device, the AP has to process six updates. It has to process the message that AP 1 and AP 2 receive in Fig. 3.3. This means an AP has to process, on average, $\frac{6}{1.25} = 4.8$ updates per second resulting from the roams the AP participates in. Additionally, an AP has to process the roams of the other devices in the VXLANs it has configured. These are the two updates AP 3 receives in Fig. 3.3. So an AP has to process an additional $\frac{2}{1.25} = 1.6$ updates per second. In total, an AP has to process 6.4 updates per second. This adds up because, with every roam, the RR sends eight update messages to the APs, and $8 \cdot 16,000 = 6.4 \cdot 20,000$. Compared to the 32,000 update messages per second that an AP had to process without filtering, 6.4 are far fewer messages to process.

Because the number of updates in general is lower, this should relieve the links in the WiMoVE system. Especially the link connecting the RR should be relieved because the RR has to send fewer messages.

In this example, the advantages of using RTCRD in high-load situations are imminent. It prevents the overload of the physical links between the BGP speakers and the overload of the APs. Because of the sparse membership that every AP has a maximum of 2007 connected devices and that the VXLANs have an average size of three, those benefits apply to all large WiMoVE systems [11, 25]. What I did not account for in this example are the additional costs of filtering. The RR has to filter the BGP EVPN routes and process the RTCRD messages. Those effects heavily depend on the implementation of RTCRD, and I will explain this trade-off in more detail in Chapter 4. Without those effects, RTCRD might worsen the roaming times in scenarios with no load, but I expect an improvement in high-load scenarios where the AP would be overloaded otherwise at the cost of a higher load on the RR.

3.3 Improvements to RTCRD in WiMoVE

I discussed how RTCRD affects WiMoVE systems in the last two sections. I formed the hypothesis that RTCRD will increase the roaming time in low-load scenarios because the information for new VXLANs is not locally available on an AP. Also, the load on the RR might increase because it has to process more incoming update messages. In this section, I discuss how those issues could be addressed.

3.3.1 Distributing the Load Among Multiple RRs

In the previous section, I showed that the number of incoming updates the RR has to process increases with RTCRD enabled. Also, the load might increase due to filtering, depending on the implementation. To counteract these effects, RTCRD could be used to distribute the load among multiple RRs. In Section 3.2.1, I discussed that advertising the default RT on the RR is advantageous because it lowers the number of RIB entries. A similar approach can be used to distribute the load among multiple RRs. Every RR is responsible for a subset of all RT in the WiMoVE system. Being responsible means that the APs exchange BGP EVPN routes for this RT only via this RR. This can be implemented by advertising special RTCRD routes. The RR advertises only RTCRD routes with the RTs it is responsible for to the APs and the gateway. The RR must not act as an RR for the RTCRD address family. If the RR acts as RR for the RTCRD address family, the split will not work because it would advertise all RTCRD routes from the gateway, for example, and each RR is responsible for all RTs. Without acting as an RR for the RTCRD address family, the RR only receives BGP EVPN routes for this subset of RT from the APs. The distributed RTCRD information for this scenario is shown in Fig. 3.4. Here, the load is distributed among two RRs. RR blue is responsible for RT blue and RR red for RT red. In our example, every RR would only have to process 32,000 incoming updates and send out only 80,000 updates. This split could also be done with other filtering mechanisms in BGP. The advantage of using RTCRD is that it is easy to configure this because the configuration happens on the RRs. With other filtering mechanisms, the configuration must happen on the APs. Having the configuration on the RR also facilitates designing a dynamic load-balancing approach. The RRs could communicate with each other and migrate the responsibility of RTs between them if necessary.

3.3.2 Origin-Based Filtering

In Section 3.2.3, I analyzed the effects of RTCRD on a single roam. I concluded that the setup time might increase because it takes longer until the information on the AP the device roamed to is available for a new VXLAN. This is because the APs do not have all the information, which is by design to prevent memory overload situations described in Section 3.1.1. To mitigate this effect, having information for important devices on all APs would be helpful. An important device could, for example, be the gateway. In many deployment scenarios, the connectivity to devices outside the VXLANs is important. Origin-based filters that supersede RTCRD could be used to discern between important and normal devices. Routes from specific origin would be distributed regardless of the received RTCRD routes. In a WiMoVE system, all routes originated by the gateway would be available on all APs all the time, which allows new devices to connect faster to the Internet. Wang et al. proposed a filtering mechanism that could be adapted to serve this purpose [31].

3.3.3 Roaming Prediction

Depending on the time it takes until an AP receives all information from the RR for a new VXLAN, roaming prediction can help to improve the setup time and make them equal to a scenario without filtering. When a roam is predicted, the AP the device roams to can

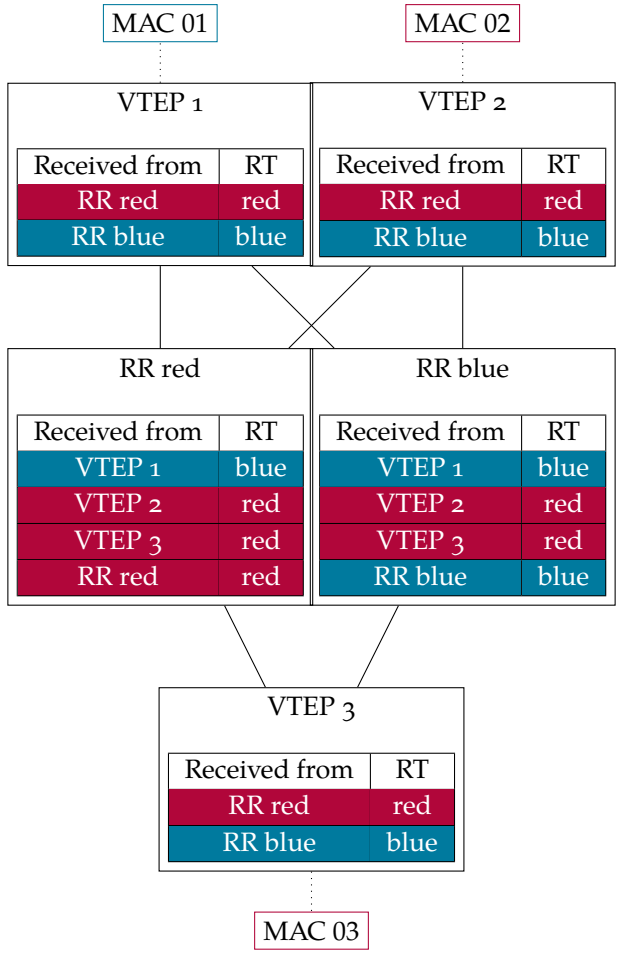


Figure 3.4: Configuration for the RTCRD address family to distribute the load among multiple RRs

already configure the VXLAN for that device. The red parts of the message exchange shown in Fig. 3.3 can happen before the device is connected. The designer of a roaming prediction must be aware that predicting a roam is associated with a cost. When a roam is predicted that does not happen, sending out the updated RTCRD routes and receiving the BGP EVPN routes was unnecessary. A simple approach for roaming prediction would be to scan the Wi-Fi environment for nearby devices connected to other APs and configure VXLANs for them if their received signal strength surpasses a threshold. The threshold controls that not too many false positive predictions happen, leading to unnecessary traffic in the control plane.

4 Implementation

In Chapter 3, I discussed how RTCRD affects roaming in a WiMoVE system. Although RTCRD is specified in RFC 4684 [17], no open-source implementation exists. I had to build a prototype implementation to evaluate the effectiveness of filtering in real-world scenarios. In this chapter, I will describe how I built the prototype implementation and the design decisions I took. The prototype implements all functionality specified in RFC 4684 within FRR. The code is open-source and available in the WiMoVE GitHub organization [28]. The goal is to integrate the code into the FRR mainline.

4.1 FRR as a Base Implementation

Free Range Routing (FRR) is an open-source routing daemon that implements BGP EVPN. It supports receiving and sending BGP EVPN routes, installs the received routes via *zebra* in the kernel forwarding tables, and generates updates from changes in the forwarding tables. I chose FRR because it has a large community and is actively developed.

4.2 Design Decisions for the Implementation

During the implementation, I had to make many decisions. In this section, I discuss those that affect the performance of the prototype implementation most. I will discuss why I decided to implement it that way and shed light on alternative approaches that might be implemented in the future.

4.2.1 Update Groups in FRR

BGP speakers with many peerings often have to distribute received update messages to multiple other BGP speakers. This is especially the case for RRs, which must distribute a received update message to all other RR clients. In BGP, for every route and every peer, the decision has to be taken if a route should be propagated. The decision process is specified in RFC 4271 [21] and involves multiple steps. In most cases, there are multiple peers with the same configuration, and therefore, the decision process is the same and just gets executed multiple times with the same parameters. Because of this, FRR uses update groups to improve the scaling properties of the BGP daemon. Donald Sharp introduced update groups in 2015.[1] If the peers' configuration is equivalent regarding filter-relevant configuration options, multiple peers can be aggregated into an update group. The computing power needed to distribute a received route now only scales with the number of update groups instead of the number of peers. In current BGP EVPN setups, all RR clients are aggregated

[1] https://github.com/FRRouting/frr/commit/3f9c7369f7112d87007b87a5faaa61cdd5e24c39

into one update group. With RTCRD, this is no longer possible. The received RTs affect the filtering and might be different for every peer. So, I decided to put every peer that has RTCRD activated into a different update group. This affects the computing power required on the RR to distribute received routes. So, implementing filtering is a trade-off on the RRs between the additional computation needed to perform the selection process and the saved computation power and data rate since fewer updates must be sent out. How this affects performance in real-life scenarios can be seen in Chapter 5. Countermeasures for this problem could be aggregating peers with the same RTCRD into update groups. Another idea would be to use the update groups for all the properties that are the same. Routes that get dropped there can be discarded, and only routes that do not get dropped there have to be evaluated by the RTCRD filter. If this is effective, it depends on the ratio of routes that get filtered before RTCRD filters the routes. Both proposed solutions require big code changes, and I did not implement them because they were out of scope for the prototype implementation.

4.2.2 Storing RTCRD Information in an Easy to Match Format

When receiving RTCRD, the information must be stored in the RIB to be distributed to other peers. One possibility to apply the filters to outgoing routes would be to traverse the RTCRD information stored in the RIB every time. But then, the time scales linearly to the number of RIB entries. Instead, I maintained a prefix tree for every peer that saves the received RTCRD information. Other filtering mechanisms in FRR already used this prefix tree implementation. The prefix tree implementation is based on hash maps and allows for a faster lookup. David Lamparter introduced the prefix tree structure in a commit in 2015 for different prefix-based filters.[2] The results of his measurements that he wrote down in the commit message show performance improvements for a high number of prefixes that have to be filtered.

4.2.3 Challenges Processing RTCRD Updates

Receiving new RTCRD routes from a peer changes the outgoing filters to that peer. The receiving peer has to evaluate which routes have to be advertised to or withdrawn from that peer. This is done by doing a full table scan of all BGP EVPN routes in the RIB and matching them to the new prefix tree. This means the effort of updating RTCRD information scales linear to the number of BGP EVPN entries in the RIB. This means processing RTCRD messages on the RR is expensive in a WiMoVE system. It has a BGP EVPN RIB entry for every connected device on the RR and receives, in the worst case, two RTCRD update messages per roam. To make the processing of the update messages less expensive, the receiving peer must quickly determine which routes must be withdrawn and advertised. This could be done by maintaining a mapping between RT and a list of all associated routes. This could speed up the processing significantly and make the processing time only dependent on the number of routes with the affected RTs. Implementing this would have been quite complex and is not essential for a working implementation. Because of this, I decided this was out of scope for my prototype. If there is a performance bottleneck on the RR, this could be a solution that solves the issue.

[2] https://github.com/FRRouting/frr/commit/7e111b6b0de44a9d351cdfeda4c674224a65ec3b

5 Evaluation

To evaluate the effectiveness of the filtering mechanism, I conducted two experiments with the prototype implementation I described in Chapter 4. In the first experiment, I measured the memory usage on the AP in a WiMoVE system in relation to the number of connected devices. In the second experiment, I measured the roaming times in different load scenarios with filtering and without filtering.

5.1 EVPN Load Generation

Since I could not build a WiMoVE system with multiple thousand devices, I needed a way to generate a comparable load on the control plane. The tool must support multiple thousand devices and must be able to simulate the control plane traffic resulting from multiple thousand roam events per second. No existing tool satisfies the need to generate load in BGP EVPN setups with RTCRD. For example, bgperf2 [20] focuses on evaluating the performance of processing IPv4 and IPv6 routes and has no support for BGP EVPN. exaBGP is another tool that can be used to generate BGP update messages [16]. It supports BGP EVPN, but there is no support for RTCRD. Because of this, I built a tool to generate load.

The load generator I designed consists of two components: VTEPs and an orchestrator. The VTEPs are docker containers with a FRR instance. The orchestrator coordinates the roams and decides which MAC address is reachable at which VTEP. Changing the timing of the roam events makes it easy to simulate different mobility patterns. The code is open-source and available on GitHub[24].

5.2 Effect of RTCRD on Memory Consumption

In Section 3.1, I stated that the memory consumption on an AP must not scale with the number of connected devices. The analysis showed that RTCRD should achieve this. To validate this in practice, I conducted this experiment. The setup is shown in Fig. 5.1 and consists of 21 VTEP and an RR. The dotted lines are BGP peerings, and the solid lines are links. One VTEP is an AP, and the others generate the load. The load generated by those VTEPs is distributed among 100 VNIs. On the AP, no VXLAN is configured. For the tests, I used the Linksys WRT 1900 ACSv2 AP, which is equipped with 512 MB of memory. In Fig. 5.2, we see the number of routes and the consumed memory on the RR and the AP without filtering. The AP can handle around 490,000 routes before running out of memory. The out-of-memory condition is visible because the number of routes drops to zero, as well as the consumed memory. The BGP daemon gets restarted two times but crashes shortly after. The RR can handle the number of routes without problem. In Fig. 5.3, we see the

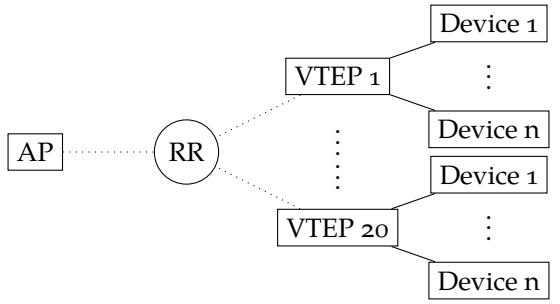


Figure 5.1: Setup for the memory measurements

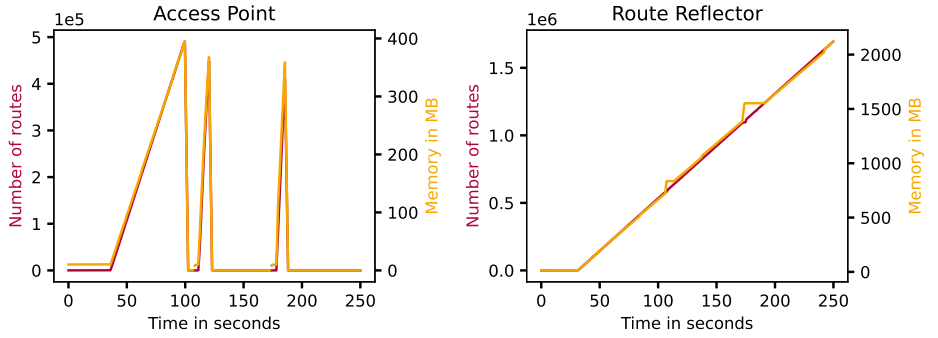


Figure 5.2: Memory usage in correlation to the number of routes without RTCRD

result for the same experiment but with filtering enabled. As expected, the AP does not receive any routes, and the memory usage is independent. This shows that the filtering is working and that the RIB size on an AP is now independent of the number of connected devices. This allows for larger WiMoVE deployments with more connected devices in total.

5.3 Measuring Roaming Times in a WiMoVE System

In Chapter 3, I explained that I expect the roaming times to improve in high-load scenarios and worsen in low-load scenarios when RTCRD is enabled. In Chapter 1, I described the two conditions that must be met to complete a roam in WiMoVE. The AP the device roamed to needs the forwarding information for the VXLAN of the device, and the other APs need the updated forwarding information. The timing of the first process is called *setup time*, and the timing of the second process is called *propagation time*. I measured the timing for both processes in separate experiments and with different load scenarios. The code for the measurements is on GitHub [27].

5.3.1 Load Scenarios

For the experiments, I designed three different load scenarios. The first one is called *idle*. In the idle scenario, there is no load generated. This scenario is used to get a baseline. I tried to replicate the scenario I introduced in Section 3.1 with the other two load scenarios. However,

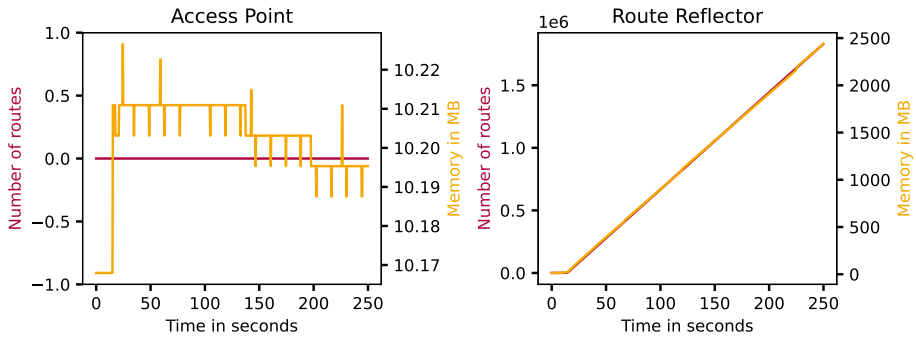


Figure 5.3: Memory usage in correlation to the number of routes with RTCRD enabled

it was impossible to simulate 20,000 VTEP and 400,000 VXLANs with the hardware I had available. Instead, I ended up with 20 VTEPs, 100 VNIs, and 400,000 devices because I could simulate this amount of VTEPs and VNIs with the available hardware and get reproducible results regarding the number of updates per second. To lower the load on the VTEPs, every VTEP only has a VRF for 20 VNIs configured.

The two scenarios only differ in the mobility pattern of the devices. A mobility pattern is characterized by the waiting time of a device between two roams. For the *normal* scenario, I used an exponential distribution with the parameter $\lambda = \frac{1}{25}$ to model the waiting time. Because of the properties of the exponential distribution, every device roams in expectation every 25 s. I chose an exponential distribution because it is cheap to compute and a general way to model waiting times. In the future, more sophisticated mobility models could be used that better represent the mobility pattern of WiMoVE users. For the *max* scenario, the orchestrator generated as many roam events as possible for every device. With the hardware I used, producing as many roams as possible resulted in 20,000 BGP update messages per second reaching the RR.

Using fewer VTEP affects the number of BGP update messages the RR has to send, but the number of incoming BGP EVPN update messages should be the same. What is different from the load scenario described in Section 3.1 is that the imported RTs do not change on the VTEPs. Every VTEP at any time has at least one device for every VNI. This means that no RTCRD update messages reach the RR. In Section 4.2.3, I explained that processing RTCRD updates is costly because it requires full table scans. For this reason, it would be interesting to repeat those experiments with many RTCRD updates instead of solely BGP EVPN updates.

5.3.2 Tracking Additional Metrics

I tracked additional metrics that shed light on the system's state to verify the generated load and explain the findings.

- The CPU load of the BGP daemon.
- The memory usage of the BGP daemon.
- The number of sent and received update messages for every connected peer.

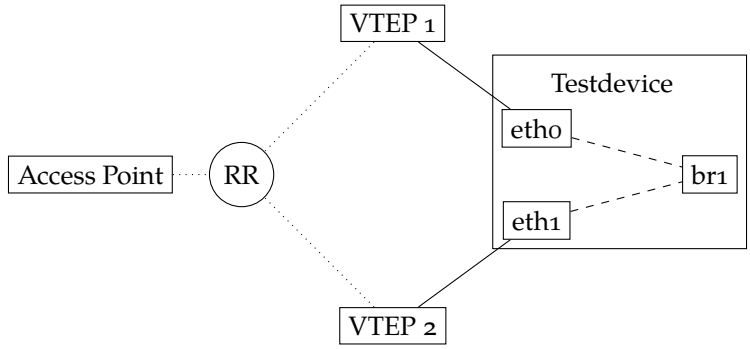


Figure 5.4: Visualization of the measurement setup for the propagation time

- The length of the in and out queue for messages in the BGP daemon for every connected peer.

I tracked those metrics for all the VTEPs not part of the load generation and the RR. The scripts I tracked those metrics with are in the measurements' repository [27]. The time resolution of those probes is best effort.

5.3.3 Measuring the Propagation Time for BGP EVPN Updates

For a roam to be complete, the update messages generated by the AP the device roamed to have to be processed by all other VTEPs. To measure this time, I designed a topology and wrote scripts.

5.3.3.1 Experiment Setup

To conduct this experiment, I needed a way to control the mobility of a device between two VTEPs. The topology I ended up with is shown in Fig. 5.4. The dotted lines are BGP peerings. The solid lines are links. The dashed lines are links that can be switched on and off. This topology allows simulating mobility of the testdevice. The testdevice can be connected to VTEP 1 or VTEP 2 by turning the links on and off. The VTEPs, the testdevice, the RR, and the AP are all physical devices. The network connection between the AP, the RR, and the VTEPs is a single L2 domain. What is important to note is that the VXLAN on the VTEPs was not deleted or created when the device roamed. I decided to measure the propagation time separately from the setup time. The AP is a Linksys WRT1900 ACSv2 running OpenWrt 22.03.2. The AP has 512 MB of memory and 2 Cores running with 1.6 GHz. The VTEPs and the RR are desktop machines running Ubuntu Server 22.10 LTS. They have Intel Core i5-6500 processors with 4 Cores, a base frequency of 3.2 GHz, and a boost frequency of 3.6 GHz. They have 8 GB of memory. All the links are 1 Gbit links.

5.3.3.2 Measurement Methodology

To measure the propagation time, I need to determine when the testdevice is reachable again from the AP after changing the VTEP.

One way would be to send probe packets between the testdevice and another device connected to the AP. As soon as packet loss occurs, the testdevice is no longer reachable, and as soon as packets are transmitted again, the testdevice is reachable again. I tried that approach and achieved a time resolution of 10 ms. Since the measured values were in the range of 100 ms, I needed an approach with higher time resolution.

The second approach I came up with is monitoring the Forwarding Database (FDB) on the AP. The FDB is the part of the VRF where the BGP routes get installed in the kernel, so the operating system knows about them and can forward packets accordingly. I came up with this approach because, with the packets, I probed if an FDB entry was updated. One could listen for this change directly instead of probing for this change with probe packets. When listening for the change, the time resolution is no longer limited by the probe packets' interval. There are multiple ways to interact with the kernel on a Linux system. Netlink is the most promising one for monitoring the FDB without polling because Netlink supports multicast groups that user space programs can subscribe to to get informed about events in the kernel.[12]. It is important to say that Netlink communication is asynchronous and has no guarantees regarding timing. No literature analyses the latency of netlink multicast messages. Nevertheless, I decided to go with Netlink because of two reasons. Other tools also use Netlink to monitor the FDB instead of polling. For example, FRR and the bridge command of iproute2 [10] use netlink to communicate with the kernel and to get notified about changes to the FDB. Also, the load on the Netlink system should be consistent during the measurements. There is only a tiny amount of Netlink messages during the experiments, so the latency of the Netlink messages should be consistent across the measurements and allow for a comparative analysis between the measurements. The netlink multicast group that is relevant for the measurements is called RTMGRP_NEIGH, which notifies about changes to the FDB in the kernel[13]. The generated netlink messages include detailed information about what changed in the FDB.

The script I used for the roaming measurement is shown in Listing 5.1. The code is executed on the AP. In the function triggerBridge, a message to the testdevice is sent so it changes the bridge state. A daemon listens on the testdevice that controls the bridge. After sending the message, the script starts listening to Netlink messages. For every netlink message, the script checks if the updated FDB entry contains the MAC address of the testdevice and sets the VTEP to the received value. With this setup, the transmission delay between the testdevice and the AP affects the measurements. The transmission delay is low and consistent across the measurements because the testdevice and the AP exchange those messages over a separate wired link. The exact code for the measurements can be found on GitHub [27].

5.3.3.3 Using Data Plane Learning

The VXLAN implementation of Linux allows to enable data plane learning for an interface in addition to the BGP EVPN control plane. With data plane learning enabled, the APs can learn the address of an AP for a device by packets sent out by a device that reaches the AP. This can not only be unicast packets but also broadcast packets. Many devices send out broadcast packets after a roam. For example, IPv6 router solicitations or Dynamic Host Configuration Protocol (DHCP) discover messages. Learning from those packets might

Listing 5.1: Measurement script for roaming between two VTEPs

```
def main()
    current_vtep = getCurrentVtep()
    while True:
        triggerBridge()
        start = time.now()
        old_vtep = current_vtep
        while old_vtep = current_vtep:
            netlink_message = recvNetlinkMessage()
            if netlink_message.Mac == "11:22:33:44:55:66":
                current_vtep = netlink_message.Remote
                end = time.now()
                break
        took = end - start
```

be faster than waiting for the control plane. For this reason, I conducted experiments with data plane learning enabled and disabled to make a comparison between those.

5.3.3.4 Results

In Fig. 5.5, the results of the roaming measurements are shown. In total, I conducted 12 runs with different parameters. For every run, I repeated the experiment 1000 times. The light red violin graphs show the results with filtering, and the light orange ones show the results without filtering. With data plane learning enabled, all scenarios have no visible difference in propagation time. They all vary between 20 ms and 80 ms. This meets my expectations because the VXLAN was not deleted or created when the testdevice roamed. This means that the VXLAN was always configured on the VTEPs, so broadcast packets originated by the testdevice could always be sent to the AP. Because of this, the load on the control plane does not matter for the data plane learning. The roaming time is consistent across all measurements. This means that if the guarantees a control plane gives are not strictly necessary, data plane learning can be used to keep the propagation delay equal regardless of the load on the control plane when a forwarding path between the VTEPs is already established.

With learning disabled, there are significant differences between the different scenarios. In the idle scenario, the times are between 120 ms and 180 ms, both with and without filtering. This meets my expectations because the message exchange with and without filtering is the same. There is no difference in the load on the RR, APs, or the VTEPs.

The plots in Fig. 5.6a and Fig. 5.7a show how the load scenarios differ regarding the number of updates per second received by the RR. In the normal load scenario, the RR receives approximately 10,000 updates per second, and in the max scenario, approximately 20,000 updates per second. Comparing Fig. 5.6a with Fig. 5.6b and Fig. 5.7a and Fig. 5.7b shows that the filtering is working because the RR has to send out way less updates with filtering enabled.

The propagation times with filtering are slightly worse for the normal load scenario. An unpaired one-sided t-test with a significance level of 95% shows that the mean of the times with filtering is higher. An explanation is that the BGP worker thread needs longer to

generate the updates for all connected peers because, as I explained in Section 4.2.1, with RTCRD enabled, the BGP speaker could no longer use the optimization of update groups. One indication of this is the length of the input queues shown in Fig. 5.6c and Fig. 5.6d. The input queues contain updates received from a peer but not processed by the BGP worker thread. On average, the queues on the RR for the peers in the load generation setup are 20 updates long when filtering is enabled and only 11 updates long with filtering disabled. This shows that the additional processing power needed for filtering impacts how fast the updates can be processed by the BGP worker thread. The difference in the length of the output queue on the RR shown in Fig. 5.6e and Fig. 5.6f is not enough to counteract this effect. The length of the output queue with filtering is always zero. Without filtering, the length goes up to 300. When the input queue is longer, it takes longer until an update can reach the AP. However, the spikes in the output queue length are seldom.

The max scenario shows a significant difference between the roaming times. With filtering, the times are between 100 ms and 300 ms. Without filtering, the roaming time is between 25 s and 35 s. The other metrics I tracked explain the rapid increase in the no filtering setting between normal and max load. If we compare the additional metrics for the max load scenario in Fig. 5.7 with the metrics for the normal load scenario in Fig. 5.6, we see that the AP can not keep up with the load in the max scenario without filtering. The AP output queue on the RR in Fig. 5.7e is almost always at the max value of 10,000. For the normal load scenario, the length of the output queue is at most 300. The same is true for the input queue of the AP in Fig. 5.8a and Fig. 5.9a. The input queue length in the normal load scenario peaks at around 1,400, whereas it is almost constant at around 10,000 in the max scenario. This means that the max scenario leads to an overload situation described in Section 3.1, and the AP can not keep up with processing the updates. The measurements show this results in a worse roaming time because the propagation time increases. RTCRD can prevent this overload. With filtering enabled, the AP only has to process the updates generated by the roams of the experiment and not the ones generated from the load scenario. Therefore, only the delay introduced by the additional load on the RR affects the propagation time.

The results show that RTCRD is a viable option for scenarios where the APs experience an overload situation because they can not keep up with the incoming update messages. However, using RTCRD in medium load scenarios might be disadvantageous because there are no update groups. I would need a measurement setup with more route reflector clients that receive the BGP EVPN updates to see the effect of not using update groups more dominantly. Testing this was not possible with the computing resources I had available.

5.3.4 Measuring the Setup Time

To communicate with other devices after a roam, the VRF of the new VXLAN needs the reachability information for all other devices in the VXLAN. The time it takes to complete this process is called *setup time*. In this experiment, I measure the setup time and how load in the WiMoVE system affects it.

5.3.4.1 Experiment Setup

To measure the setup time, I need a device connected to the system that does not roam between VTEPs but is always connected to one VTEP. So, to conduct this experiment, I

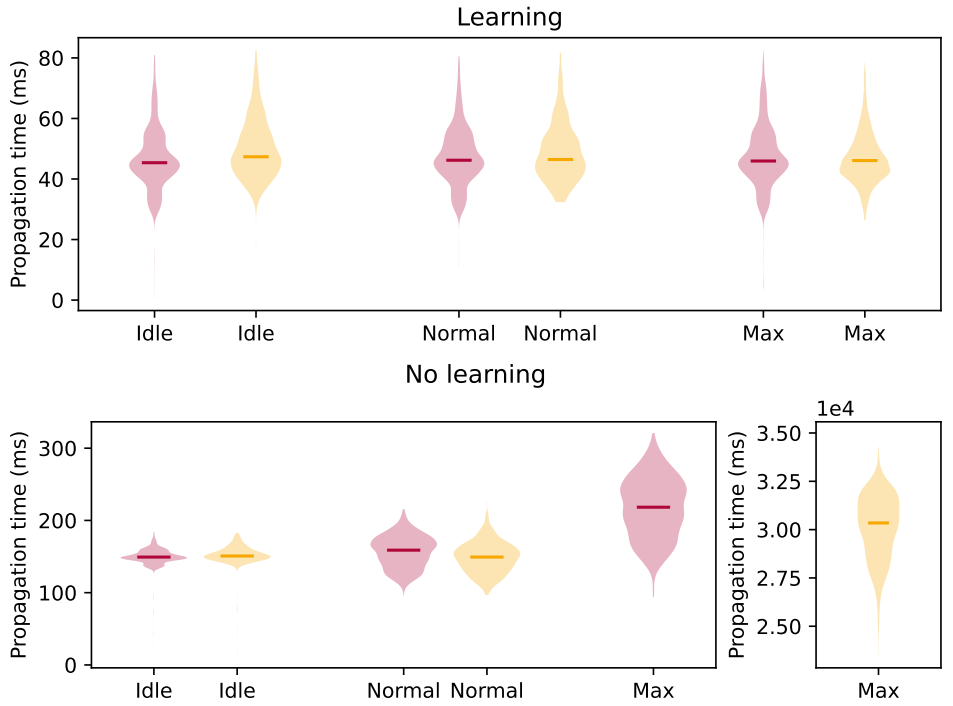
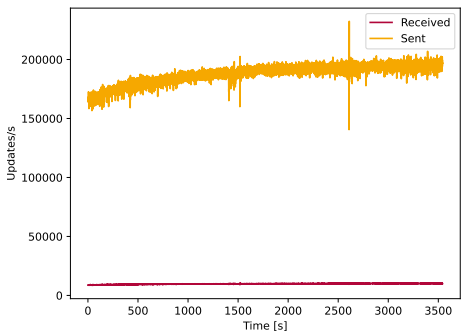


Figure 5.5: Results of propagation time measurements. Red with filtering and orange without filtering. Propagation time is in milliseconds. Every Measurement was repeated 1000 times

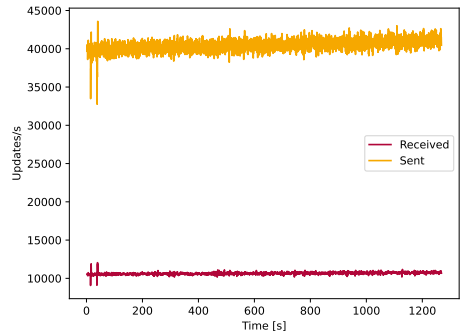
created the following setup shown in Fig. 5.10. Instead of only one device, I chose to have two devices connected to two different VTEPs, an RR, and the AP on which I conducted the measurements. The devices connected to the VTEPs are in the same VXLAN. I chose two devices because they can exchange packets so the VTEPs know the device is still connected and do not withdraw the routes because of time-outs. The AP is a Linksys WRT1900 ACSv2 running OpenWrt 22.03.2. The AP has 512 MB of memory and 2 Cores running with 1.6 GHz. The VTEPs and the RR are desktop machines running Ubuntu Server 22.10 LTS. They have Intel Core i5-6500 processors with 4 Cores, a base frequency of 3.2 GHz, and a boost frequency of 3.6 GHz. They have 8 GB of memory. All the links are 1 Gbit links.

5.3.4.2 Measurement Methodology

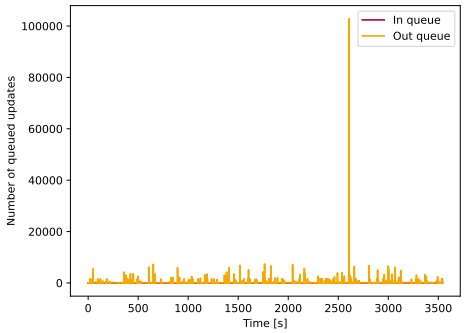
I used the topology shown in Fig. 5.10 for the measurements. To measure the setup time, I create and delete a VXLAN interface on the AP with the same VNI as the VXLAN device 1 and device 2 are connected. To determine when the setup is complete, I monitored the FDB directly using Netlink. I explained in Section 5.3.3.2 why I chose this approach for my measurements and the limitations of it. I define the setup time as the time delta between setting the VXLAN interface up and the MAC address of device 1 being installed in the FDB of the AP. How the whole measurement procedure works is shown in Listing 5.2. At first, the VXLAN interface gets deleted. Then, the script waits 2 seconds to give the control plane time to adjust to this change. When using filtering, the routes need some time to be



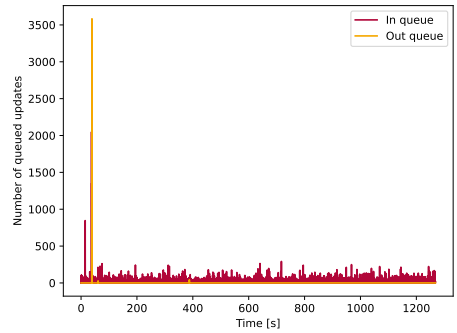
(a) Rolling average with n=10 for the updates sent and received from the load generation without filtering.



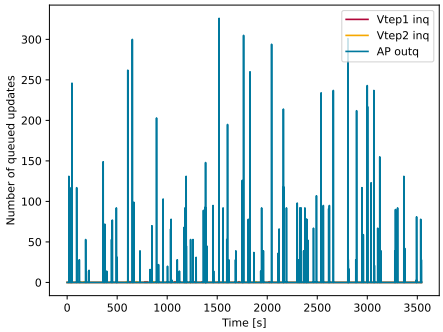
(b) Rolling average with n=10 for the updates sent and received from the load generation with filtering.



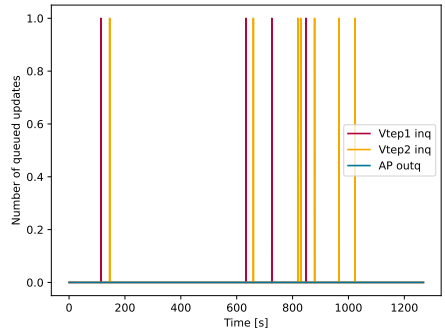
(c) Sum of the length of the input and output queue to the peers of the load system without filtering.



(d) Sum of the length of the input and output queue to the peers of the load system with filtering.

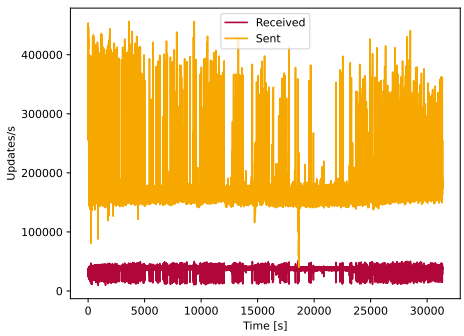


(e) Length of the input queue for VTEP 1 and VTEP 2 and for the output queue to the AP without filtering

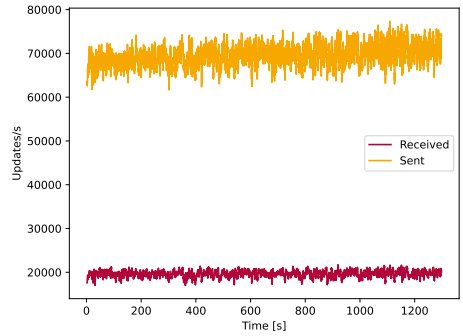


(f) Length of the input queue for VTEP 1 and VTEP 2 and for the output queue to the AP with filtering

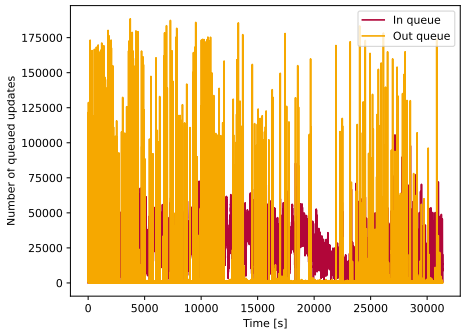
Figure 5.6: Additional metrics for the propagation time experiments with the normal load scenario and control plane learning disabled. On the left are the metrics for the runs without filtering, and on the right are the metrics for the runs with filtering.



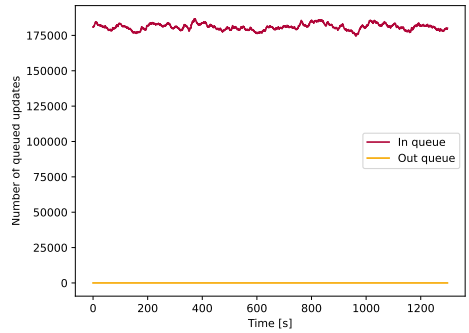
(a) Rolling average with n=10 for the updates sent and received from the load generation without filtering.



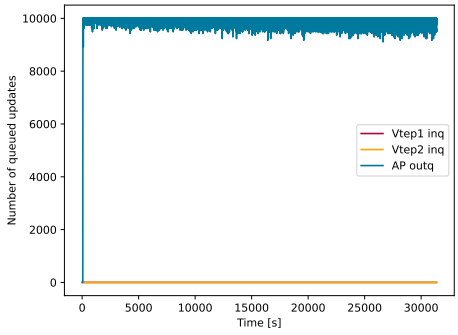
(b) Rolling average with n=10 for the updates sent and received from the load generation with filtering.



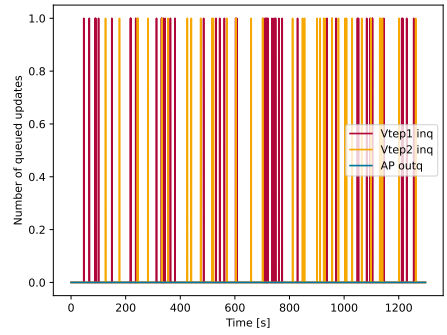
(c) Sum of the length of the input and output queue to the peers of the load system without filtering.



(d) Sum of the length of the input and output queue to the peers of the load system with filtering.

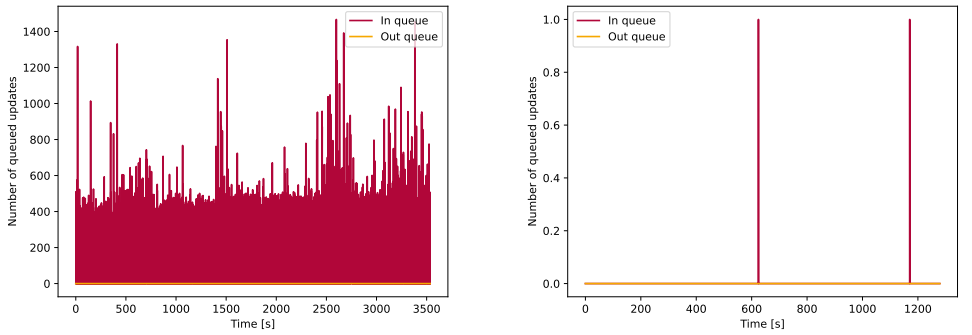


(e) Length of the input queue for VTEP 1 and VTEP 2 and for the output queue to the AP without filtering



(f) Length of the input queue for VTEP 1 and VTEP 2 and for the output queue to the AP with filtering

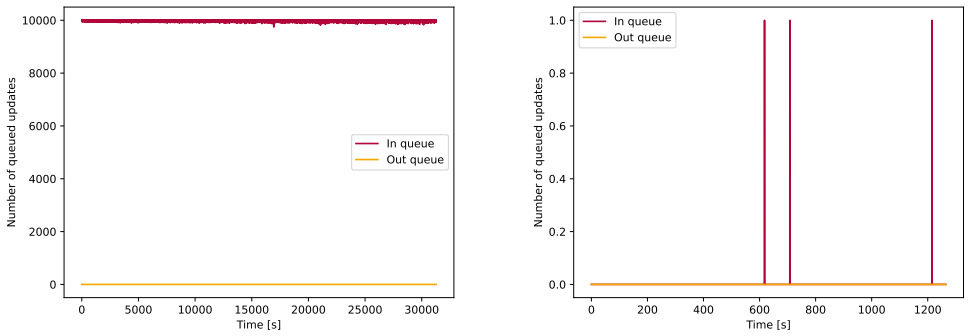
Figure 5.7: Additional metrics for the propagation time experiments with the max load scenario and control plane learning disabled. On the left are the metrics for the runs without filtering, and on the right are the metrics for the runs with filtering.



(a) Length of the input queue and the output queue on the AP with no filtering

(b) Length of the input queue and the output queue on the AP with filtering

Figure 5.8: Metrics on the AP for the propagation time experiments with the normal load scenario and control plane learning disabled. On the left are the metrics for the runs without filtering, and on the right are the metrics for the runs with filtering.



(a) Length of the input queue and the output queue on the AP with no filtering

(b) Length of the input queue and the output queue on the AP with filtering

Figure 5.9: Metrics on the AP for the propagation time experiments with the max load scenario and control plane learning disabled. On the left are the metrics for the runs without filtering, and on the right are the metrics for the runs with filtering.

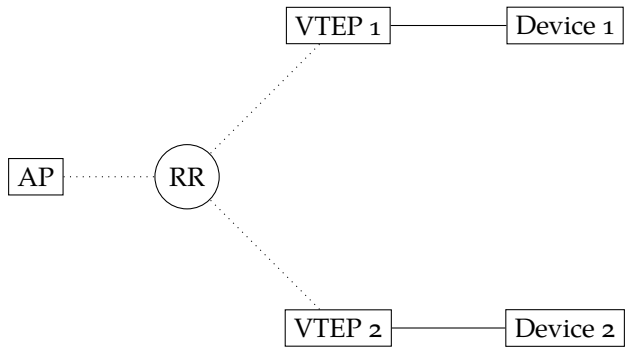


Figure 5.10: Visualization of the measurement setup for the setup time

withdrawn from the AP. Then, the script waits until the route for device 1 is installed and finishes the measurement. The exact code can be found in the GitHub repository [27].

Listing 5.2: Measurement for reconfiguring of a VRF

```
def main()
    while True:
        deleteVxlan()
        sleep(2)
        createVxlan()
        start = time.now()
        while True:
            netlink_message = recvNetlinkMessage()
            if netlink_message.Mac == "11:22:33:44:55:66":
                end = time.now()
                break
        took = end - start
```

5.3.4.3 Results

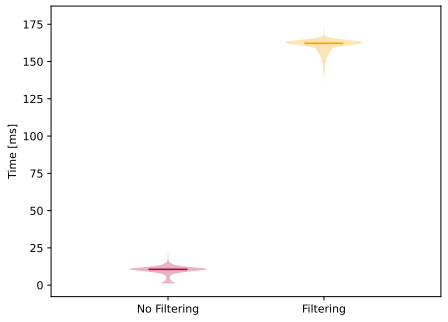
In Fig. 5.11, we see the results of all measurements. Let us first take a look at the idle scenario. As expected, the setup time increases when filtering is enabled because the routes have to be transmitted from the RR to the AP. This is an observation I already made in Section 3.2.3. However, it is by design that not all information is available on the APs because this is what leads to the out-of-memory conditions I showed in Section 5.2. However, in the normal Fig. 5.11b and max Fig. 5.11c scenario, the no-filtering experiment takes much longer. This is unexpected since the forwarding information is already available on the APs. I analyzed the FRR source code and found the root cause. The problem is related to the problem I explained in Section 4.2.3. The BGP daemon performs multiple full table scans on the AP to get the routes for the new VXLAN. It performs one full table scan for each BGP EVPN route type, in total, three full table scans. With RTCRD, there is no full table scan on the AP and only one on the RR to adapt to the new filter. The measurements in Fig. 5.11b and Fig. 5.11c show that the latency introduced by the three table scans is far more significant than the latency introduced by transmitting the routes over the network. This means RTCRD outperforms no filtering in those scenarios. A fix could be to store the route information differently and allow for easy access by a known RT, as described in Section 4.2.3.

It is important to note that the setup time depends on the transmission delay of the link between the RR and the AP. During the measurements, the RR and the AP were connected via a switch, which means the delay was in the range of 10 ms and, therefore, almost negligible. A higher transmission delay would affect the setup time in the filtering scenario but leave the time without filtering unaffected. So, RTCRD should not be used if a high transmission delay between the APs and the RR exists.

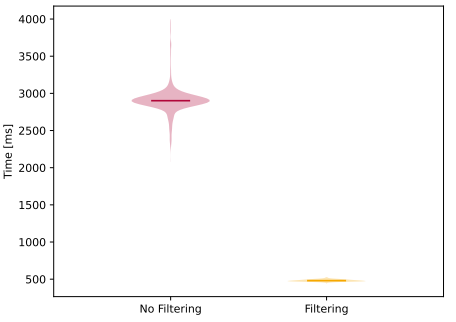
I compared the setup time between using filtering and not using filtering. Now, I want to compare the setup time when only the load changes and filtering or no filtering is constant. The results for filtering enabled are shown in Fig. 5.12b. As expected, the setup time grows from idle to normal to max. The higher the load on the RR gets, the longer it takes to process

the RTCRD message on the RR. The results in Fig. 5.12a are unexpected. The normal scenario performs far worse than the max load scenario. The number of RIB entries on the AP is equal for both scenarios, and the AP has to process more updates. With a static analysis of the FRR source code, I could hypothesize why the max scenario performs better. FRR uses an event-based architecture. Multiple event loops are running with a fetch execute cycle. The controlling entity of an event loop is the thread master. The thread master always fetches the next events and puts them in the ready queue. Then, all events in the ready queue get executed until the ready queue is empty again. For BGP, two event loops work together: The main working thread and an input/output thread. The input/output thread reads the packets from the network and puts them into the input queue. From there, the worker thread receives them and processes them accordingly. To synchronize those threads, FRR uses a mutex. The time the input/output thread is in its critical section depends on how many bytes it can read. The more data it reads, the longer it is in its critical section. In Fig. 5.9a, we can see that the input queue is always almost full in the max scenario. Therefore, the input/output thread is for a shorter amount of time in the critical section. The worker thread has to wait until the input/output thread leaves the critical section. Because of this, the event that a new VRF exists and that the routes should be installed needs longer until it is processed. I could not validate this hypothesis with the data I collected. One approach to validate this hypothesis is to use the built-in tracing feature. The tracing has to be enabled at compile time and allows fine-grained information about the scheduling and execution of events. This was not enabled in the FRR builds I used for my measurements, so I could not use it.

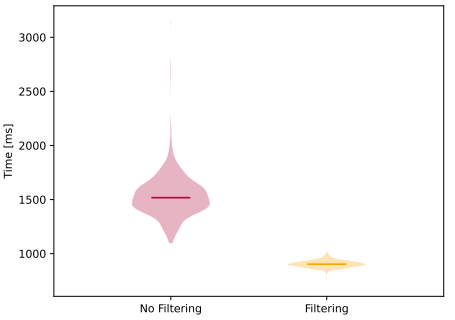
The results overall show that using RTCRD has benefits if the number of BGP EVPN RIB entries is high on the AP without it. The performance for filtering and no filtering is limited by the time it takes to perform the full-table scans. Improving the lookup times would benefit both scenarios.



(a) Setup time no load

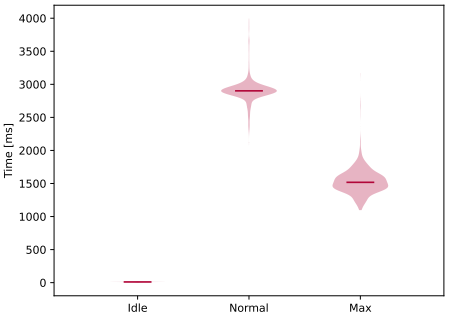


(b) Setup time normal load

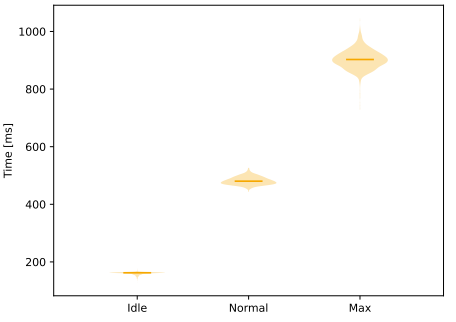


(c) Setup time max load

Figure 5.11: Setup time measurements grouped by load scenario



(a) Setup time no filtering



(b) Setup time filtering

Figure 5.12: Setup time measurements grouped by filtering and no filtering

6 Related work

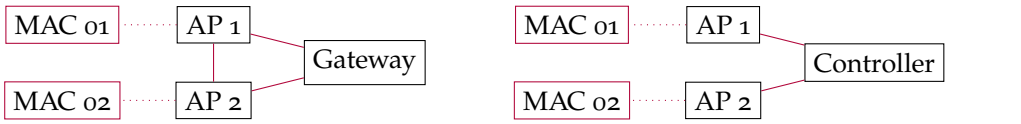
Since the WiMoVE architecture is new and not widely adopted, there exists no other work that analyzed roaming times in WiMoVE systems. Because of this, I evaluate if the approach applies to other Wi-Fi architectures, and I look at what the results mean for other BGP EVPN deployments.

6.1 Wi-Fi Architectures

Vendors for large Wi-Fi systems use different architectures. Instead of allowing the APs to exchange packets via tunnels directly, all traffic has to go through a central controller. WiMoVE also has the RR as a central node in the control plane, but the RR is not part of the data plane. In Fig. 6.1a, it is shown how the tunnels in a WiMoVE system look like. The device with MAC 01 and MAC 02 can communicate without using the gateway because there is a tunnel between AP 1 and AP 2. The same scenario in a Wi-Fi system with a central controller is shown in Fig. 6.1b. Packets exchanged between device 01 and device 02 must go through the controller because direct forwarding between AP 1 and AP 2 is impossible. Vendors that, for example, use this approach are Cisco and Ruckus [6] [7]. Because of the central controller in those systems, there is no need for a control plane that distributes the mapping between a device and the connected AP. Every AP forwards all traffic to the controller regardless of the destination. The reachability information only exists on the central controller. Therefore, the idea of filtering messages in the control plane to improve the roaming times does not apply to those Wi-Fi architectures.

6.2 Other Filtering Mechanisms in BGP EVPN Setups

Different filtering mechanisms in BGP allow an operator to enforce routing policies. The simplest ones are route maps. They get configured on a BGP speaker and filter incoming and outgoing routes. A route map consists of a match condition and an action. The match condition can be regarding most attributes of a route. The action is deny or accept. What route maps are missing to use them as a filtering mechanism for WiMoVE is the possibility to update them dynamically. However, if the import policy of all VTEPs is known and does



(a) Tunnels in a WiMoVE setup

(b) Tunnels in a Wi-Fi setup with a central controller

Figure 6.1: Difference between a Wi-Fi setup with a central controller and a WiMoVE system

not change, the operator can configure the route maps accordingly and does not need to set up RTCRD.

RFC 5291 specifies another filtering mechanism called dynamic Outbound Route Filtering (ORF) [5]. Two BGP speakers can exchange dynamic filters with this. The difference between RTCRD and dynamic ORF is that the filters do not propagate in the network and only affect the peering where they are exchanged. It would have been possible to use dynamic ORF for filtering in WiMoVE systems and achieve comparable results with the current topology. However, I chose RTCRD because it is more general and does not make assumptions regarding the topology. Also, ORF is not specified for BGP EVPN routes yet to allow filtering based on RTs.

6.3 Roaming Times in BGP EVPN Setups

VXLAN with BGP EVPN was developed as a network overlay solution for data centers [23]. WiMoVE is a different use case with different requirements for the speed of the control plane. For Data center networks, the mobility features of BGP EVPN and VXLAN are important to enable live migrations of Virtual Machines (VMs) throughout the network while keeping the network state [18]. The timescale for a VM migration is on the scale of seconds and not comparable to a Wi-Fi roam [22]. The experiments in Chapter 5 show that BGP EVPN can update the networking state in this time frame if the BGP speakers are not overloaded. Because there was no use case before WiMoVE with the same requirements, and the existing solution fulfilled the requirements, no research analyzed the speed of the control plane and possible optimizations.

7 Conclusion

In this work, I analyzed the effects of RTCRD on the roaming time in a WiMoVE system. I started by identifying the overload of the APs as the root cause for higher roaming times in WiMoVE deployments with many devices. Then, I explained how RTCRD prevents overload situations by filtering in the BGP EVPN control plane. Because RTCRD seemed promising, I built an open-source implementation based on FRR and conducted experiments to validate the findings. The experiments in Section 5.3 showed that RTCRD successfully prevents overload situations in high-load scenarios. The roaming times decreased from 30 s to 1 s with RTCRD enabled. One drawback of RTCRD is that in low-load situations, the setup time increases with RTCRD enabled from 10 ms to 150 ms. However, this does not impact the roaming time because, in low-load scenarios, the roaming time is limited by the propagation time, which is at around 150 ms with RTCRD enabled and disabled. Besides the advantages regarding the roaming times, RTCRD also removes the limitation for the maximum number of devices in a WiMoVE system.

The results are not only applicable to WiMoVE. RTCRD might also benefit other BGP EVPN setups. In Section 3.1, I explained the problems in high-load situations in a WiMoVE system. Together with the results of the experiments, I could establish some general guidelines that can help to decide if RTCRD should be used in a given BGP EVPN setup. The most significant prerequisite is that the membership of the VTEPs has to be sparse. Otherwise, RTCRD can not provide any benefit because no filtering can happen based on the RTs. If the membership is sparse, a clear indication that RTCRD should be used is when the BGP daemon on the VTEPs runs out of memory. When this happens, all devices connected to this VTEP lose connectivity. With RTCRD, fewer routes must be stored on the VTEPs because they get filtered before they reach the VTEPs. A second indicator is the length of the input queues on the VTEPs. The input queues will fill up if the VTEP can not keep up with processing the updates. Long input queues result in a partial loss of connectivity, affecting the devices whose updates have not been processed yet. RTCRD reduces the number of updates the VTEPs have to process, which leads to shorter input queues.

Not only the VTEPs can be overloaded. The links can also be overloaded. When this happens, RTCRD can reduce the number of updates that must be transmitted and relieve the links.

The downside of using RTCRD is that the load on the RR increases because it has to perform filtering. But, scaling the performance of the RR might be easier than scaling the performance of all VTEPs.

In this thesis, I analyzed the effects of RTCRD and built the first open-source implementation of RTCRD. This allows other researchers to explore how RTCRD affects their use case. Researchers in the past were already hindered by the fact that no open-source implementation of RTCRD existed. For example, Buob, Lambert, and Uhlig wanted to use RTCRD to build their new iBGP topology but could not do it due to the lack of an open-source implementation [2].

With the experiments, I not only evaluated how RTCRD affects WiMoVE, but I also tested the BGP EVPN implementation of FRR. During those experiments, I analyzed the implementation and developed some ideas on how to speed up the implementation. FRR would benefit from faster access to routes with a specific RT. The speed of the lookup is the limiting factor for the setup time.

7.1 Future Work

I built a load-generation setup for the experiments to create different load scenarios. This load-generation setup can be used to simulate different mobility patterns. In this thesis, I focused on simple mobility patterns. Future work could analyze the effect of RTCRD in more complex mobility patterns. A mobility pattern that results in many RTCRD messages is particularly interesting since those need much processing power on the RR.

Because RTCRD performed well for WiMoVE systems, it could be interesting to investigate how RTCRD affects other BGP EVPN use cases. One example would be the VM migration in data centers, which is the primary use case of BGP EVPN VXLAN [18]. One would have to analyze the mobility patterns of this use case. Simulating those mobility patterns would be possible with the load-generation setup.

To further improve the setup time with RTCRD, I made some proposals in Section 3.3 that were out of scope for this thesis. Because RTCRD successfully reduced the roaming time in high-load situations, it is worth investigating those to improve the performance in low-load scenarios.

Bibliography

[1] Sangeetha Bangolae, Carol Bell, and Emily Qi. "Performance Study of Fast BSS Transition Using IEEE 802.11r". In: *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*. IWCMC '06. New York, NY, USA: Association for Computing Machinery, 2006, pages 737–742. ISBN: 1-59593-306-9. DOI: 10.1145/1143549.1143696. URL: https://doi.org/10.1145/1143549.1143696.

[2] Marc-Olivier Buob, Anthony Lambert, and Steve Uhlig. "iBGP2: A Scalable iBGP Redistribution Mechanism Leading to Optimal Routing". In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. Apr. 2016, pages 1–9. DOI: 10.1109/INFOCOM.2016.7524409.

[3] Ravi Chandra, Tony J. Bates, Yakov Rekhter, and Dave Katz. *Multiprotocol Extensions for BGP-4*. RFC 4760. RFC Editor, Jan. 2007. DOI: 10.17487/RFC4760. URL: https://www.rfc-editor.org/info/rfc4760.

[4] Enke Chen, Tony J. Bates, and Ravi Chandra. *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*. RFC 4456. RFC Editor, Apr. 2006. DOI: 10.17487/RFC4456. URL: https://www.rfc-editor.org/info/rfc4456.

[5] Enke Chen and Yakov Rekhter. *Outbound Route Filtering Capability for BGP-4*. RFC 5291. RFC Editor, Aug. 2008. DOI: 10.17487/RFC5291. URL: https://www.rfc-editor.org/info/rfc5291.

[6] Inc. Cisco Systems. *Cisco Wireless Controller Configuration Guide, Release 8.10*. URL: https://www.cisco.com/c/en/us/td/docs/wireless/controller/8-10/config-guide/b_cg810.pdf.

[7] Inc CommScope, editor. *SmartZone 6.1.1 (LT-GA) Administration Guide (SZ100/vSZ-E)*. June 2023. URL: https://support.ruckuswireless.com/documents/4375-smartzone-6-1-1-lt-ga-administration-guide-sz100-vsz-e.

[8] The kernel development community. *Virtual Routing and Forwarding (VRF)*. URL: https://docs.kernel.org/networking/vrf.html.

[9] John Drake, Wim Henderickx, Ali Sajassi, Rahul Aggarwal, Dr. Nabil N. Bitar, Aldrin Isaac, and Jim Uttaro. *BGP MPLS-Based Ethernet VPN*. RFC 7432. RFC Editor, Feb. 2015. DOI: 10.17487/RFC7432. URL: https://www.rfc-editor.org/info/rfc7432.

[10] Stephen Hemminger, David Ahern, and Alexey Kuznetsov. *Iproute2 v6.4.0*.

[11] "IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks—Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), pages 1–3534. DOI: 10.1109/IEEESTD.2016.7786995.

[12] Michael Kerrisk and Andries Brouwer. *Netlink(7) — Linux Manual Page*. URL: https://man7.org/linux/man-pages/man7/netlink.7.html.

[13] Michael Kerrisk and Alejandro Colomar. *Rtnetlink(7) — Linux Manual Page*. URL: https://man7.org/linux/man-pages/man7/rtnetlink.7.html.

[14] Arjan Koopen. "The Importance of Broadcast/Multicast Filtering in High Density Wi-Fi Networks". 2016. URL: https://eventinfra.org/airtime.

[15] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. RFC Editor, Aug. 2014. DOI: 10.17487/RFC7348. URL: https://www.rfc-editor.org/info/rfc7348.

[16] Thomas Mangin, Thomas Morin, and Vincent Bernat. *exaBGP*. Version 4.2.21. Aug. 27, 2023. URL: https://github.com/Exa-Networks/exabgp (visited on July 27, 2023).

[17] Pedro R. Marques, Luyuan Fang, Jim Guichard, Luca Martini, Robert Raszuk, Keyur Patel, and Ron Bonica. *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*. RFC Editor, Nov. 2006. DOI: 10.17487/RFC4684. URL: https://www.rfc-editor.org/info/rfc4684.

[18] Dr. Thomas Narten, Eric Gray, David L. Black, Luyuan Fang, Larry Kreeger, and Maria Napierala. *Problem Statement: Overlays for Network Virtualization*. RFC 7364. RFC Editor, Oct. 2014. DOI: 10.17487/RFC7364. URL: https://www.rfc-editor.org/info/rfc7364.

[19] given-i=Inc family=Ubiquiti Networks given=Inc. *UniFi - Managing Broadcast Traffic*. 2020. URL: http://web.archive.org/web/20201019031102/https://help.ui.com/hc/en-us/articles/115001529267-UniFi-Managing-Broadcast-Traffic.

[20] Justin Pietsch, Pier Carlo Chiodi, Fujita Tomonori, Rafael Zalamena, Benedikt Rudolph, Maria Matejka, and Hiroshi Yokoi. *Bgperf2*. Version 2.0. July 27, 2023. URL: https://github.com/netenglabs/bgperf2 (visited on July 27, 2023).

[21] Yakov Rekhter, Susan Hares, and Tony Li. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. RFC Editor, Jan. 2006. DOI: 10.17487/RFC4271. URL: https://www.rfc-editor.org/info/rfc4271.

[22] Kateryna Rybina, Patni Abhinandan, and Alexander Schill. "Analysing the Migration Time of Live Migration of Multiple Virtual Machines:" in: *Proceedings of the 4th International Conference on Cloud Computing and Services Science*. 4th International Conference on Cloud Computing and Services Science. Barcelona, Spain: SCITEPRESS - Science and and Technology Publications, 2014, pages 590–597. ISBN: 978-989-758-019-2. DOI: 10.5220/0004951605900597. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004951605900597 (visited on Aug. 26, 2023).

[23] Ali Sajassi, John Drake, Nabil Bitar, Ravi Shekhar, Jim Uttaro, and Wim Henderickx. *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*. RFC 8365. RFC Editor, Mar. 2018. DOI: 10.17487/RFC8365. URL: https://www.rfc-editor.org/info/rfc8365.

[24] Aaron Schlitt and Alexander Sohn. *EVPN Load Generator*. Version 0.0.1. URL: https://github.com/WiMoVE-OSS/evpn-load-generator.

[25] Aaron Schlitt, Alexander Sohn, Lina Wilske, and Richard Wohlbold. "WiMoVE: An Architecture for Large Wi-Fi Systems". In: SKILL. Berlin.

[26] Adrian Schmutzler, Alexander Couzens, and Alvaro Fernández Rojas. *OpenWrt*. Version 22.0.3.5. URL: https://github.com/openwrt/openwrt.

[27] Alexander Sohn. *EVPN Measurements*. URL: https://github.com/Sohn123/evpn-measurments.

[28] Alexander Sohn. *FRR with RTCRD Support*. URL: https://github.com/wiMoVE-OSS/frr/tree/bgp-routing-target-constraints.

[29] *Table of Hardware OpenWrt*. CSV. https://openwrt.org/toh/views/start, July 23, 2023.

[30] Dan Tappan, Srihari R. Sangli, and Yakov Rekhter. *BGP Extended Communities Attribute*. RFC 4360. RFC Editor, Feb. 2006. DOI: 10.17487/RFC4360. URL: https://www.rfc-editor.org/info/rfc4360.

[31] Wei Wang, Aijun Wang, Huan Deng, Peng Wang, Huanan Li, Yue Wang, and Honglei Xu. "The Research on VPN Route Control Mechanism with Route Distinguisher Granularity in Intra-AS and Inter-AS Scenarios". In: *2022 IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2022, pages 1485–1488. DOI: 10.1109/IAEAC54830.2022.9929647.

[32] Bryan Ward. *Apples to Apples: An Analysis of the Effects of mDNS Traffic*. 2023. URL: https://wlanprofessionals.com/apples-to-apples-an-analysis-of-the-effects-of-mdns-traffic-on-a-campus-wlan/.