WiMoVE: A Scalable Wi-Fi System

Design Document

Aaron Schlitt
Alexander Sohn
Lina Wilske
Richard Wohlbold

Bachelor's Project 2022/23
Internet Technologies and Softwarization
Hasso-Plattner-Institute
Potsdam, Germany

Contents

List of Figures

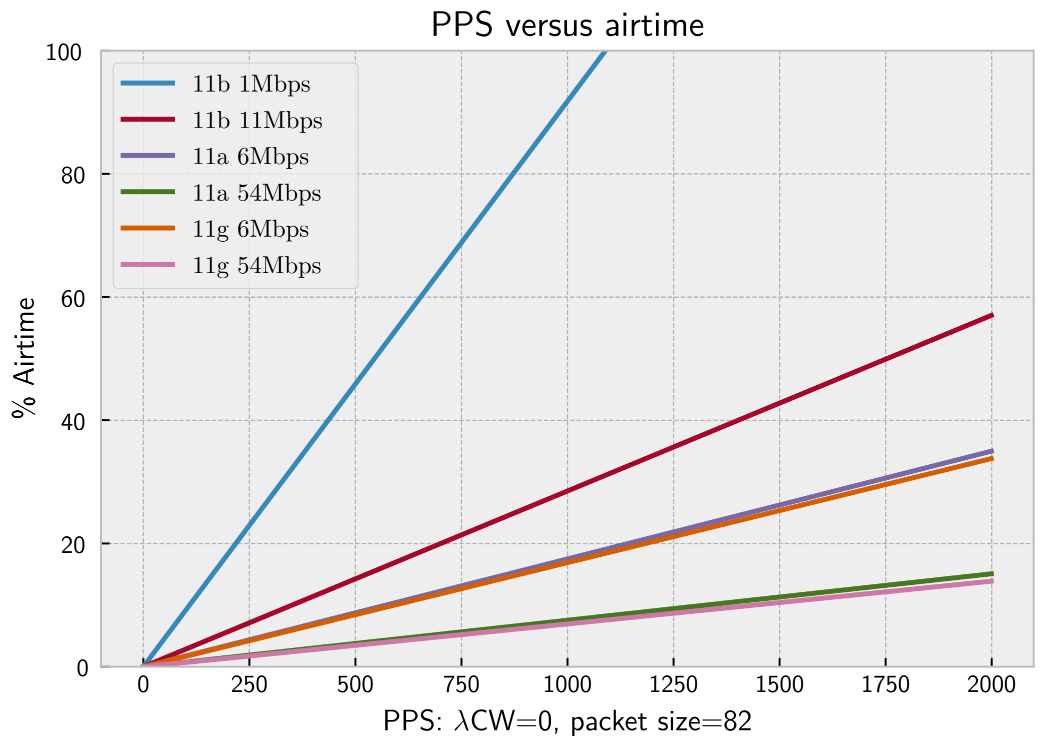

Figure 1: The percentage of airtime used in relation to packets per second for common data rates. All calculations were performed using a tool that Arjan Koopen described in a presentation at WLPC 2016 [1].

1 Introduction

In large Wi-Fi deployments, L2 broadcast traffic can become a problem. If all APs are in a single L2 domain, broadcast traffic is sent out to all stations, taking up lots of air time. This is the case because in comparison to unicast, it is sent at lower rates to be compatible with legacy devices, making sure that it reaches all connected stations.

Figure 1 shows a plot that shows the percentage of used airtime for common data rates that are used for broadcast transmissions. See Section 3.2 for a further example.

Blocking all broadcast traffic is infeasible since many real-world protocols, such as ARP and DHCP, rely on it. Broadcast issues could be solved by splitting up the L2 domain in multiple L2 domains that have L3 connectivity. In a wireless setting, splitting up the L2 domain affects handovers: Without adjusting routing tables throughout the L3 network, a station's flows would break on handover as can be seen in Figure 2.

Adjusting the routing tables throughout the network to counteract this also breaks basic assumptions on the relation between L2 and L3 networks. Typically, each L2 domain corresponds to a L3 subnet. When clients now move between the L2 domains of APs, their IP addresses are now in a different L2 domain than other clients in the same subnet.

Enterprise Wi-Fi solutions often use a single L2 domain with some non-standard features, such as Broadcast, unknown-unicast and multicast traffic (BUM) suppression and ARP proxying. These come with some downsides: L2 restrictions are not transparent to the users and affect

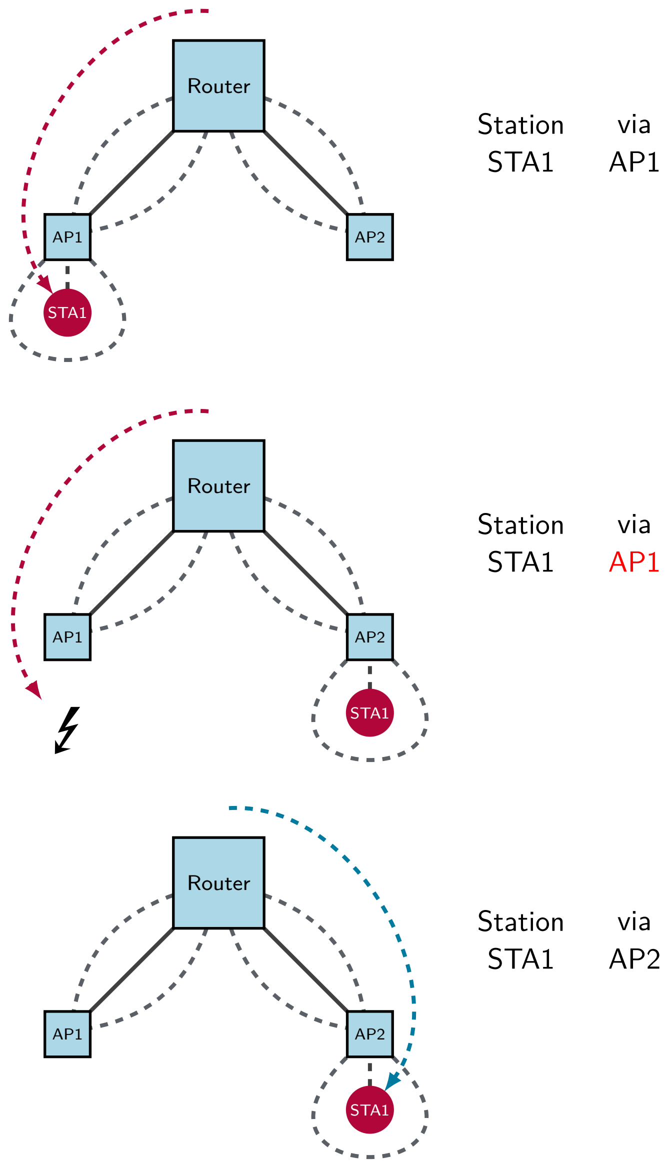

Figure 2: A basic structure of two APs with one station and one router with its routing table is shown. When STA roams from AP1 to AP2, we need to update the routing table of the routers in the L3 network since otherwise the traffic cannot be forwarded to the target station anymore

services that rely on L2 broadcast and multicast such as DHCP and MDNS. Additionally, these features are often only available on vendor-specific, proprietary hardware without being standardized. This results in vendor lock-in since solutions from different manufacturers cannot easily be combined.

Our goal is to build a Wi-Fi architecture that can handle large numbers of stations and APs while giving stations proper L2 domains with working handover.

2 Requirements

2.1 System assumptions

For our system design, we make some assumptions on the underlying infrastructure:

- There is L3 connectivity between every AP, so each AP can reach any other AP via L3 packets.
- Station MAC addresses are unique. We are also not concerned with MAC addresses being forged since we will not use them for authentication.
- We can modify the way the APs work: In practice, custom software can be run on them.
- We can deploy additional devices with custom software to the underlying L3 network.
- We cannot modify the underlying L3 infrastructure. This assumption ensures that our system can be easily deployed in existing network infrastructure.
- APs are able to isolate associated stations: At an AP, every connected station is put into a separate wireless L2 domain that can then be manually bridged to other wireless or wired L2 domains.

2.2 Functional Requirements

Our functional requirements are as follows:

- Every station is part of exactly one L2 domain with working L2 broadcast, multicast, and unicast.
- Multiple stations can be put into the same L2 domain.
- All stations have Internet access via the Wi-Fi network.
- Handover between any two APs maintains flows both to the Internet and inside the station's L2 domain.
- Wired devices can be bridged into the stations' L2 domains.
- Every station has IPv4 and IPv6 connectivity if the L3 infrastructure allows for it.
- APs can join and leave the ESS at any time.
- The solution must run on top of L3 infrastructure that is IPv4-only, IPv6-only, or dual-stack.

2.3 Non-functional Requirements

The system must additionally fulfill the following non-functional requirements:

- The system is transparent: No changes to the way stations operate are required.
- In scenarios with many APs and a small number of stations per AP, we want to lower the number of broadcast packets every AP has to send out.

3 Design Decisions

3.1 Overlay Networks as Data Plane

The core idea of our system is to partition all stations into small L2 domains. These are overlay networks, meaning that they are virtualized networks sitting on top of the existing L3 infrastructure, i.e. an L2 packet from an overlay network is encapsulated into a L3 packet. Inside each overlay network, there is working L2 unicast, multicast and broadcast. We isolate all overlay L2 networks such that two devices in different overlay networks cannot communicate with each other on L2. When a station connects to an AP, the overlay network of the station is extended to that AP. When no station in an overlay network is connected to that AP anymore, the overlay network shrinks. How a station is assigned to an overlay network is described in more detail in Section 3.3. A visualization of the overlay networks that exist in such a system can be seen in Figure 9. The approach of partitioning stations into multiple L2 networks results in smaller L2 domains, limiting the broadcast packets that have to be sent out by APs.

We can estimate the number of broadcast packets every AP has to send out for the following scenario: There are 500 APs and 20 stations are connected to each AP, resulting in a total of 10 000 connected stations. In order for the stations to have Internet access, ARP requests to resolve the L2 address of the gateway are necessary. A standard ARP timeout for connected stations is 60 seconds, resulting in 10 000 ARP requests per minute. The gateway should not send any broadcast traffic because it can learn the MAC address of the station via the ARP request of the station. Assuming they are uniformly distributed over time, there are 167 packets per second of ARP traffic. In a traditional setup without overlay networks, every AP has to send out all those ARP requests. Figure 1 shows that depending on the available rate, this can take up to 15% of the available airtime at a given AP.

This is a very conservative estimation of broadcast packets sent out by stations. In real-world examples, other protocols such as DHCP, mDNS, LLDP or NetBIOS can lead to far more broadcast traffic. Additionally, a temporarily higher fraction of airtime being used for broadcast traffic can be expected since we assume the best-case scenario with a uniform distribution over time in this calculation.

Now we analyze the situation when every station is in a different L2 domain, which consists of the station and a gateway. In this scenario, every AP only has to send out the broadcast packets originated by the gateway via the wireless interface. As discussed earlier, the gateway does not broadcast, so the APs don't have to send out any broadcast packets via their wireless interfaces.

In general, the number of broadcast packets an AP has to send out now only depends on the activity in the overlay networks it has stations in. Because the L2 domains of the overlay networks are smaller and an AP is generally not part of every overlay network, the number of broadcast packets it has to send out is smaller than in a traditional approach.

We define an *endpoint* as any networking device that performs encapsulation of overlay L2 packets into L3 packets and decapsulation of L3 packets into overlay L2 packets. We will discuss how to address L3 packets after encapsulation in Section 3.2. A *device* is any networking device that is a member of an overlay network and has a MAC address for sending and receiving packets in that network, e.g. a station. We say that a device is *attached* to an endpoint when the endpoint encapsulates L2 packets originated by the device. Note that endpoints and devices are roles within the ESS, a host can be both at the same time.

This terminology allows us to add non-wireless devices to the overlay networks; we will need this functionality in Section 3.1.1. For the purpose of this document, a device is attached to at most one endpoint, meaning that we do not consider multi-homing devices here. This has implications for the handover process, namely that make-before-break is not possible: endpoints cannot announce the presence of a device to prepare the handover while the device is still attached to another endpoint. This could become part of an extended feature set.

Each overlay network is identified by an ID from an ID namespace. The size of the ID namespace is an upper bound for the number of broadcast domains, meaning that with a namespace of $n \in \mathbb{N}$ possible IDs and a network of $e \in \mathbb{N}$ stations, there is at least one broadcast domain of size $\lceil \frac{e}{n} \rceil$. To only have broadcast domains of size $s \in \mathbb{N}$ or smaller, the namespace has to have a size of at least $n \geq \lceil \frac{e}{s} \rceil$.

3.1.1 Internet Access

For stations to connect to the Internet, overlay networks have to be terminated at a gateway.

One idea would be to use each AP as the L3 next hop for all its connected stations, forwarding packets between the stations and the underlying L3 infrastructure. The overlay networks would then only be used for communication between the stations. In order to reach a station after a handover, the routing tables of all routers in the underlying L3 infrastructure would have to be modified, requiring modifications to the underlying L3 infrastructure, which breaks our requirements. Additionally, this results in similar issues to the ones described in Section 1.

An alternative approach joins a gateway separate from the APs to each overlay network, serving as the L3 next hop. As an extended feature set, multiple of these gateways could be used for load distribution and failover. However, this is not part of the currently proposed system design. When a handover is performed, only the forwarding tables of the gateway and the APs in the overlay network have to be modified, leaving the other APs and the underlying L3 infrastructure untouched (see Figure 3).

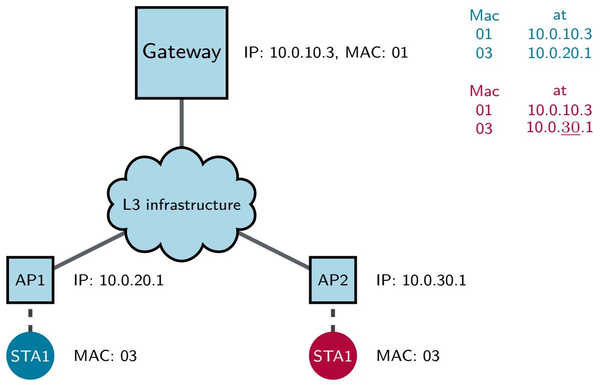

Figure 3: Forwarding tables of each overlay network endpoint (APs and gateway) before (blue) and after (red) the station has roamed between AP1 and AP2

Such a gateway can in theory be placed anywhere in the L3 network since L3 connectivity is a requirement we have imposed on the underlying infrastructure. However, in practice, since such a gateway processes all Internet-facing traffic, it may require a network connection with a high data rate, limiting possible deployment options.

We choose the option of using a gateway separate from the APs since it does not add any additional requirements to the L3 infrastructure while reducing control-plane complexity.

3.1.2 APs as Endpoints

To make the solution transparent to the stations, L2 packets have to be decapsulated before being sent to a station and encapsulated after being received from a station.

One option is to use a dedicated endpoint separate from the AP that performs encapsulation and decapsulation. With this approach, L2 isolation between the AP and the endpoint is still needed to meet the requirement of fully isolated environments for different stations. To distinguish packets at the AP, some form of encapsulation between endpoint and AP is therefore required.

Another option would be to perform encapsulation and decapsulation directly at the APs.

We observe that APs have to perform encapsulation and decapsulation to maintain L2 isolation either way. As a result, the second option has the advantage that fewer encapsulations and decapsulations are performed on overlay network packets. Therefore, we choose the APs to be the station-facing endpoints for the overlay networks.

3.2 Control Plane

To forward encapsulated L2 packets to the correct endpoint, endpoints need information on which MAC address is reachable at which endpoint in a given overlay network. We call this information *reachability information*, defined as the following partial function:

$$r : \text{MACAddresses} \rightarrow \text{OverlayNetworkIDs} \times \text{EndpointAddresses}$$

Endpoints should be able to assign reachability to a given MAC address. Since r is represented as a table in memory, an endpoint should be able to remove entries for devices that are no longer attached. We further describe these operations in Section 3.2.1.

Endpoints could learn r by performing backward learning. This approach is often called data-plane learning [2]. When an endpoint receives an encapsulated L2 packet with source address $m \in \text{MACAddresses}$ on overlay network $n \in \text{OverlayNetworkIDs}$ from endpoint $e \in \text{EndpointAddresses}$, it updates its reachability information r to r' by setting $r'(m) := (n, e)$. When encapsulating an L2 packet on overlay network $n \in \text{OverlayNetworkIDs}$ with destination address $m \in \text{MACAddresses}$, it looks up $r(m)$. If there is an $e \in \text{EndpointAddresses}$ such that $r(m) = (n, e) \neq \perp$, the L3 packet is addressed to e. Otherwise, the L3 packet is sent to all other endpoints. This approach has two downsides: Broadcasting L3 packets is expensive, and each endpoint needs to have a list of all other endpoints, presenting a discovery problem when endpoints can join the ESS at any time.

An alternative to data-plane learning is to distribute reachability information separately from the data plane. This is often called control plane learning [2]. There are two basic variants of control plane learning: push-based models and pull-based models.

In a pull-based model, the overlay network endpoint pulls the relevant reachability information from another device for every received packet to be forwarded. We call the device from which reachability information is pulled the *informant*. In this scenario, a lower bound for the handover duration is the propagation delay between informant and endpoint: When reachability information changes at the informant, it has to be propagated to the endpoint. For many packets, this procedure would be inefficient since endpoints do not change very often in comparison to the number of packets sent. These inefficiencies could be improved by caching reachability information with timeout t.

However, this would result in a worst-case handover time of at least t: When a station with address $m \in \text{MACAddresses}$ performs a handover right after the gateway pulled $r(m)$, the gateway would forward packets to the wrong endpoint until the timeout expires. Wi-Fi handovers are usually on the scale of 100 ms, t should be on the same scale or smaller to not significantly worsen handover time. In a scenario of 10 000 stations communicating with the gateway, a timeout of 100 ms would result in 200 000 reachability information pulls per second since the gateway and the stations' endpoints have to repeatedly pull reachability information after their timeout expires. Assuming one pull is 100 B in size, a data rate of $160\,\text{Mbit}\,\text{s}^{-1}$ would be used just for reachability information pulls. In comparison to a push-based model, this seems like unnecessary effort.

In a push-based model, each overlay network endpoint receives an update message from an informant when relevant reachability information changes. There are no timeouts in this scenario, so the propagation delay between informant and station is the only lower bound for the worst-case handover time. A potential problem with a push-based model is distributing reachability information to endpoints where it is not needed. One solution is to filter pushed reachability information by OverlayNetworkID before pushing it to each endpoint. We will discuss this in more detail in Section 3.2.2.

Overall, we felt like a push-based model is the better-suited approach here.

3.2.1 Consistency Model

As described in the previous section, endpoints need to read and write reachability information, serving as distributed storage locations for the forwarding tables they require for operation. We now want to look at different consistency models that could fulfill the system requirements, choosing the most practical among them.

The system allows each endpoint to read the OverlayNetworkID and EndpointAddress for a given MACAddress, denoted as $r(\text{MACAddress})$.

Two types of write operations are supported. The endpoint address in a message always corresponds to the endpoint from which the message originates. An endpoint should therefore only send out messages with its own endpoint address.

- REACH(MACAddress, OverlayNetworkID, EndpointAddress): A device is reachable in an overlay network at an endpoint. This information always overwrites the current reachability information in the state of the system.
- UNREACH(MACAddress, OverlayNetworkID, EndpointAddress): A device is no longer reachable at the endpoint. This information only changes the system state if

  $$r(\text{MACAddress}) = (\text{OverlayNetworkID}, \text{EndpointAddress}).$$

  Therefore, an UNREACH cannot overwrite reachability information from other endpoints.

If we used only one message type, it could occur that when an endpoint detects that a device is no longer attached, it would overwrite more current reachability information for that device written by another endpoint. By using separate REACH and UNREACH messages, we can make sure that removal of reachability information does not cause packet loss for the connected device.

One central aspect of the system is that events corresponding to write operations have an inherent order determined by physical time. This creates the problem of delayed updates: Let

m be the MAC address of a device, o an overlay network ID and let the device be attached to endpoint e_1. On connection, e_1 issues REACH(m, o, e_1). When the device roams to the endpoint with address e_2, e_2 issues REACH(m, o, e_2). Without a proper consistency model, the writes could now be read in opposite order on some endpoints, causing them to permanently send packets addressed to m to e_1 instead of e_2. An example is shown in Figure 4.

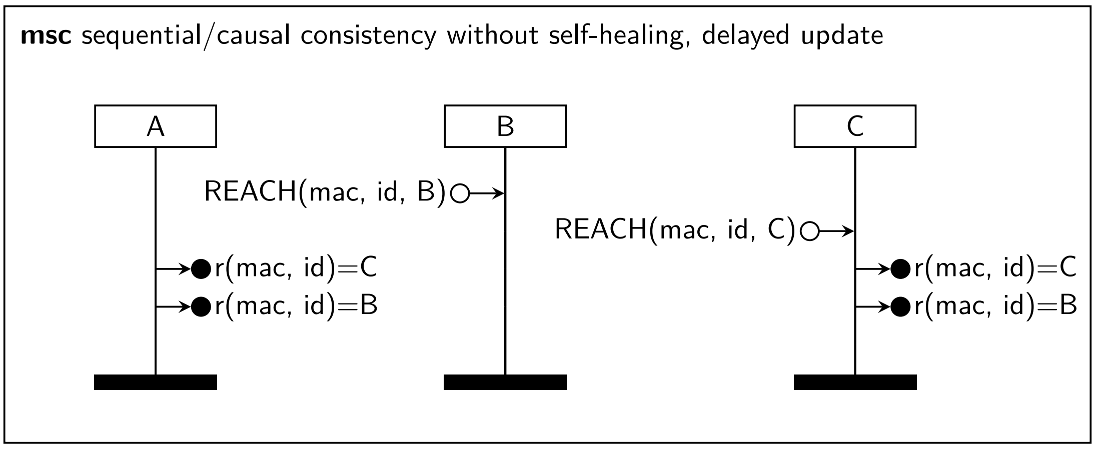

Figure 4: msc diagram of conflicting reachability writes: station moves from B to C. However, the resulting system state assumes that the station is still at B

Since the correct order of events is present on every client device, clients could be used as a point of serialization, i.e. by increasing a sequence number on the device. Since this is not a standard Wi-Fi feature, stations would have to be modified, violating our requirements.

We now look at the strongest possible consistency model, sequential consistency. The model itself does not fix the delayed update problem since sequential consistency allows for any interleaving of events, meaning that an earlier write may overwrite a later write.

This is why we propose a self-healing mechanism to fix inconsistencies in the system state, so that underlying sequential consistency is sufficient. It works as follows: Suppose an earlier write REACH(m, o, e_1) overwrites a later write REACH(m, o, e_2). This means that m is attached to e_2 but e_2 reads $r(m) = (o, e_1)$. e_2 can now detect the delayed write since m is attached to it. The endpoint now sends out an update message to correct the system state, which will then be delivered to all other endpoints after the delayed write. One example scenario where self-healing is not needed is shown in Figure 5, one scenario where self-healing is needed is shown in Figure 6.

The self-healing mechanism comes with a few problems, in particular attachment detection: In common wireless and wired scenarios, it is impossible for an endpoint to reliably detect whether a device is attached. Since most L2 protocols are packet-switched, an endpoint can only infer attachment from received device packets. These might be processed by an endpoint after the device changes its attachment, for example, when they are stored in a large queue. This can, at least in theory, cause an endpoint to send out wrong update messages. Assuming the device sends enough packets, the new endpoint will then correct the wrong reachability information. More sophisticated attachment detection schemes may be possible, reducing the number of wrong updates sent.

The self-healing mechanism makes sequential consistency suitable for the Wi-Fi architecture. However, standard approaches for sequential consistency, like transactions or token passing, are expensive to implement. Therefore, we want to look at other consistency models, which may be more practical.

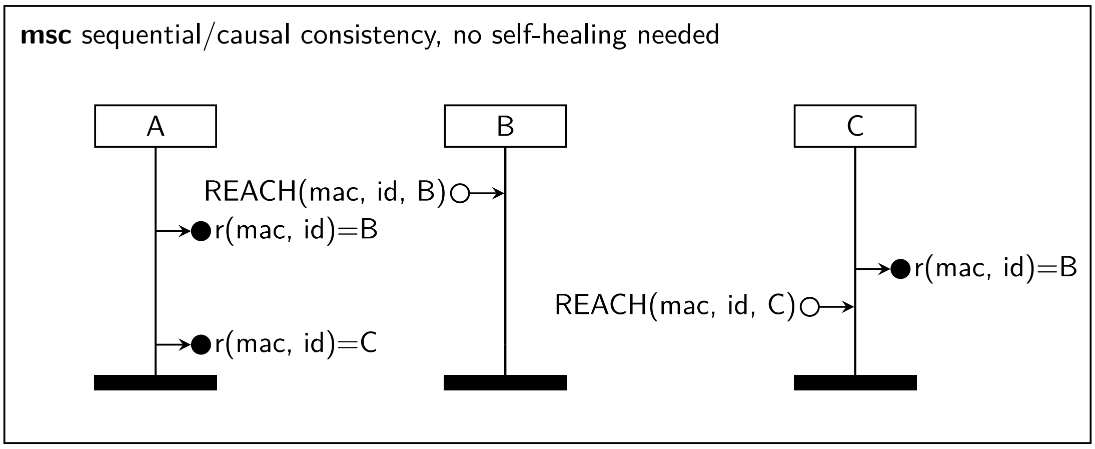

Figure 5: msc diagram of conflicting reachability information writes: station moves from B to C, no self-healing needed

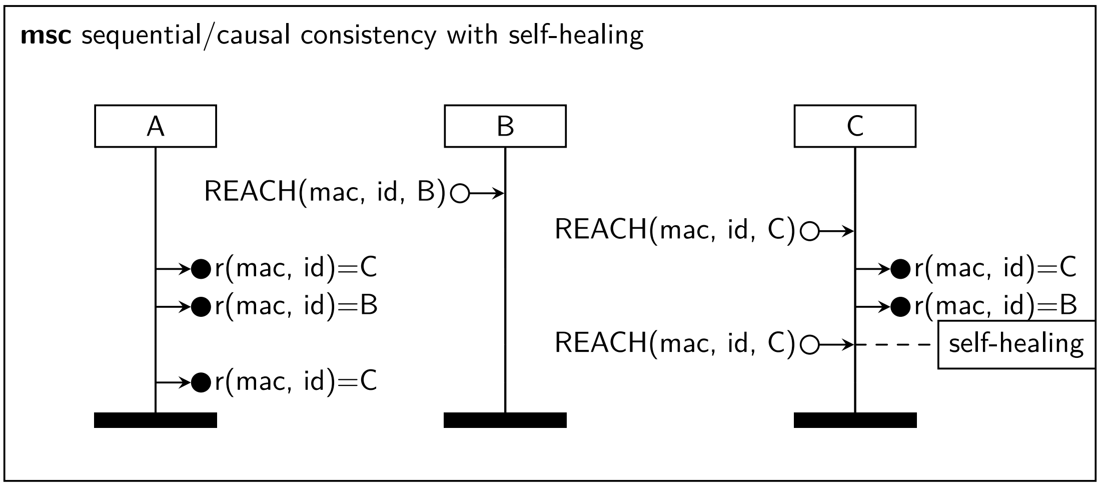

Figure 6: msc diagram of conflicting reachability writes: station moves from B to C, self-healing performed by C

Causal consistency with the same self-healing mechanism is equally sufficient as the sequentially consistent model since delayed update reads and their corrections are causally dependent, arriving in the same order on each endpoint. Since causal consistency is generally cheaper to implement than sequential consistency, it should be preferred when building a distributed system.

The next weaker model, FIFO consistency, is insufficient, even when paired with the self-healing mechanism, since the correcting writes are not dependent on the delayed read operations, as shown in Figure 7.

We can use the proposed self-healing mechanism combined with either sequential or causal consistency in our system. To make a final decision, we will first need to take a look at the underlying distribution system in the next section.

3.2.2 Reachability Information Distribution System

As described in the previous section, we want to build a distributed storage system where the endpoints replicate the state of the overlay networks they participate in. To achieve a practical

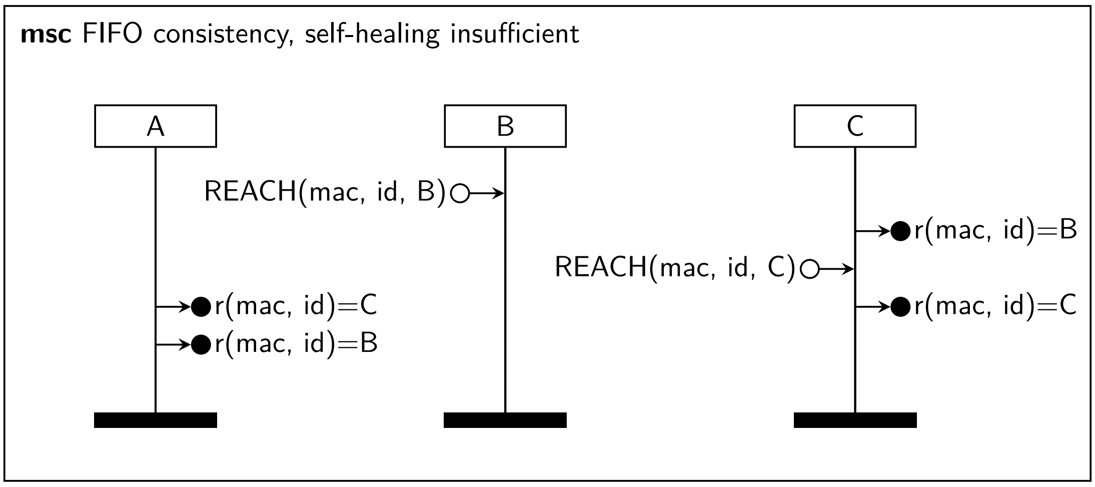

Figure 7: msc diagram of conflicting reachability writes: station moves from B to C. However, A assumes the station to be at B, undetectable by C

system architecture, we have to fulfill the following requirements:

- When joining the system, endpoints need to find other system participants in order to receive and send messages.
- When stations roam between APs, we need the state of the system to be updated. This has to happen according to one of the two possible consistency models (causal or sequential).
- Updates should be distributed while keeping the load on low-power APs small. At any point in time, every endpoint is only interested in update messages for overlay networks in which it has participating clients. To achieve this, we want to enable the system to only send relevant update messages to each endpoint.
- Endpoints must be able to join any overlay network at any time. This requires them to be able to get the current state of that overlay network.

In theory, these requirements could at least be partially fulfilled using only the endpoints as participants in the distributed storage. Causal consistency could, for example, be achieved using causal multicast.

However, we see three major challenges using this approach:

- A typical implementation of causal multicast uses vector clocks to determine the order of messages to deliver. These vector clocks each require one entry for every node participating in the distributed storage. With hundreds or even thousands of APs, this would result in the use of multiple kilobytes per update message just for the required vector clocks. The payload of these update messages only consists of the MAC address of a device and IP of an endpoint, resulting in a very high overhead.
- To enable filtering while reducing load on the APs, we would have to maintain a graph along which to distribute the multicast messages. This graph should be designed in a way that ensures that, where possible, only the endpoints interested in a certain overlay network will receive those messages. When stations roam between APs, this graph must change since the participants in an overlay network change. For this reason, we would need to generate such a graph automatically and potentially update it on every station movement to keep filtering effective.

- When querying the state upon joining a new overlay network, there is no single responsible node for getting that information. We may need to collect the data from multiple participating endpoints, implement conflict handling etc.

Such a system would thus be far more complex to design than the one we propose.

Our idea is to use a central entity to both replicate unicast as well as filter messages per endpoint. We call this central entity *Reachability Information Distribution System* (RIDS). The RIDS can also act as a serialization point for the messages. This way, we trivially achieve sequential consistency for the distributed storage. We still need the self-healing mechanism since update messages might still be reordered on their way from the endpoint to the RIDS.

This architecture allows us to add a distributed storage participant in the central entity that replicates the complete system state. This can simply be queried when an endpoint joins an overlay network and no conflict resolution is required.

Filtering can also be achieved since every endpoint has exactly one connection to a node from the central entity. This way, the central entity can store which connected endpoint is interested in which overlay networks and filter the messages when they are replicated. Changing the relevant overlay networks for an endpoint is also trivial this way.

A downside of such a central entity is that it is a single point of failure. However, in practice, we can replicate the system state on multiple nodes and enable failover to achieve high availability.

3.3 Assigning Devices to Overlay Networks

To reduce broadcast traffic, the goal is to assign devices to overlay networks in a way that results in small broadcast domains. Ideally, given enough virtual overlay networks, we would assign each device its own overlay network. However, to implement this guarantee, we would require an additional control plane that coordinates the assignment.

An alternative is to assign devices to the overlay networks by computing the OverlayNetworkID for a given device from its MAC address using a hash function. If we assume the hash function to be uniformly distributed, the assignment of devices to overlay networks becomes a version of the birthday problem with an arbitrary number of networks $b \in \mathbb{N}$ (days) and arbitrary number of devices assigned to one network $k \in \mathbb{N}$. Given a number of stations $n \in \mathbb{N}$ (people), we can now calculate the probability that at least k stations are assigned to the same overlay network anywhere in the system. Let $X = (X_1, \ldots, X_b)$ be the number of stations in each overlay network. X then follows a multinomial distribution:

$$X \sim \text{Mult}\left(n, b, \forall 1 \leq i \leq n : p_i = \frac{1}{b}\right).$$

We approximate $X_1, \ldots, X_b$ as independent Poisson random variables:

$$X_1, \ldots, X_b \sim_{iid} \text{Pois}\left(\frac{n}{b}\right).$$

We then have

$$P\left(\max_{1 \leq i \leq b} X_i \leq k\right) = F_{\text{Pois}\left(\frac{n}{b}\right)}(k)^b.$$

We calculated probabilities for the case that $n = b = 100\,000$ for different numbers of k, observing that probability falls off quickly. It is already very unlikely for $k \geq 10$ devices to be assigned to one overlay network. We confirmed the calculated probabilities with a simulation for $n = b = 100\,000$ with $100\,000$ iterations.

Therefore, using $n = b$ with random assignment of endpoints to overlay networks results in small overlay networks, even for large numbers of endpoints. Because of the reduction in complexity of the assignment, we choose this way of assigning devices to overlay networks. An extended feature set could include further modifications to the assignment, such as using a centralized service.

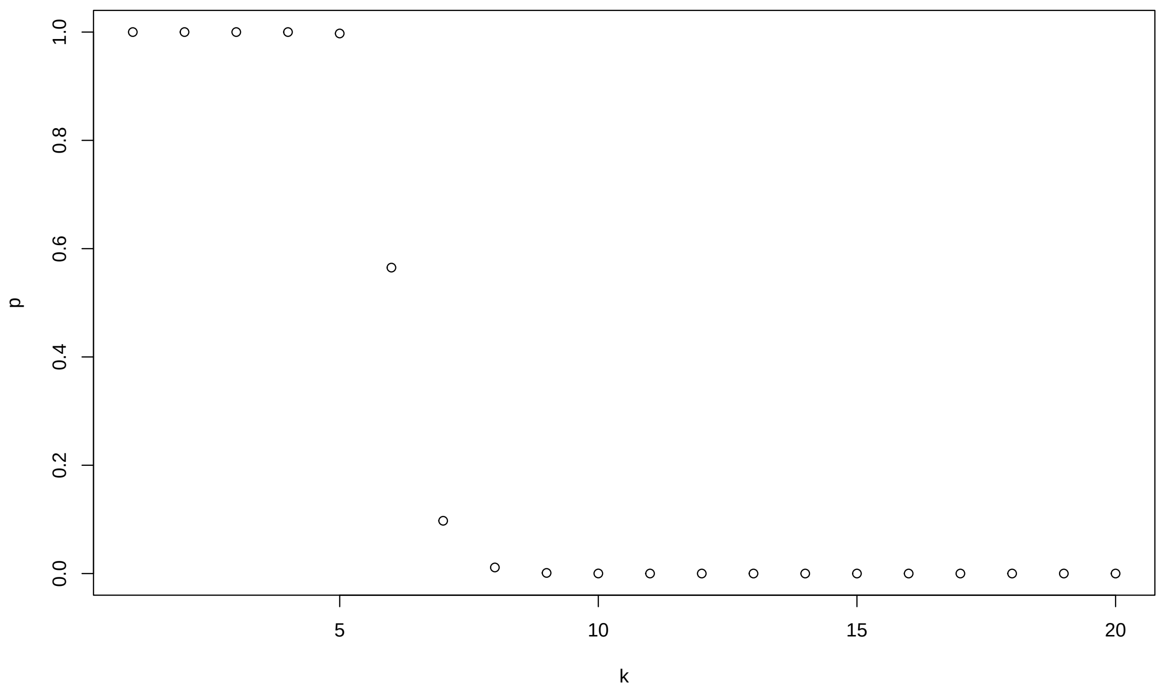

Figure 8: probabilities of assigning at least k stations in at least one overlay network with $n = b = 100\,000$ endpoints and networks

4 Architecture

In the previous section, we discussed the design decisions we made to build a scalable Wi-Fi solution. In this part of the document, we describe the resulting system architecture.

4.1 Components

4.1.1 Access Points

APs are the connection points for users with their stations, such as smartphones, tablets, or laptops. They are responsible for encapsulating and decapsulating L2 packets for the overlay networks, serving as overlay network endpoints. Additionally, they assign newly connected devices to their corresponding overlay networks based on a calculation from their MAC addresses. APs send reachability information based on currently connected stations to the RIDS. Lastly, they receive reachability information from the RIDS, updating their forwarding tables. The information provided is defined by the function r specified in Section 3.2.1.

4.1.2 Reachability Information Distribution System

Reachability information is received and distributed via the RIDS. The RIDS is a central component of the system. To reduce downtime, the RIDS consists of multiple nodes that replicate the system state. Every endpoint can connect to multiple nodes so that in the event of a failure of one of them, it can switch over.

4.1.3 Internet Access and Network Services

According to Section 2, users should have Internet access without special device configuration. We therefore require an IP next hop. For automatic IP configuration, we also need a DHCP server (for IPv4) or a service that responds to router solicitations (for IPv6) in each overlay network.

These services are provided by additional endpoints that do not have to be APs themselves. They can either be combined into one or split up into multiple endpoints.

4.2 Network Architecture

All endpoints are connected over an underlying L3 network. According to Section 2, there should not be any special requirements on this infrastructure. All endpoints and the RIDS need to be able to reach each other via IPv4 or IPv6 packets, depending on which protocols are used in the setup.

For these reasons, this section will focus on the structure of the overlay networks and the control plane.

4.2.1 Data Plane

The data plane consists of overlay networks spanning all APs where a station is connected to that overlay network. The overlay networks also include the endpoints required for services described in Section 4.1.3.

All L2 packets are encapsulated into L3 packets, which are then transmitted to the correct endpoint over the underlying L3 infrastructure. For a station, the overlay networks thus allow access to the Internet as well as other stations in the same overlay network, even when connected to a different AP. This is especially useful for an extended feature set when stations are manually assigned to overlay networks.

The constructed overlay networks are connectionless. Their state only consists of forwarding tables at each participating endpoint that are used to forward L2 packets received from stations to the correct endpoints. This allows direct communication between the APs, leaving out the gateway for packets to the local subnet. Broadcast packets sent by a station are forwarded to every endpoint where at least one station in the same overlay network is connected.

An example of different overlay networks can be observed in Figure 9. The red stations are assigned to the same overlay network but connected to different APs. To access the Internet, they both communicate with the gateway in their overlay network. When communicating with each other, the packets are directly sent between their APs. The third station shown in blue is part of a different overlay network and can thus not communicate with the other stations on L2.

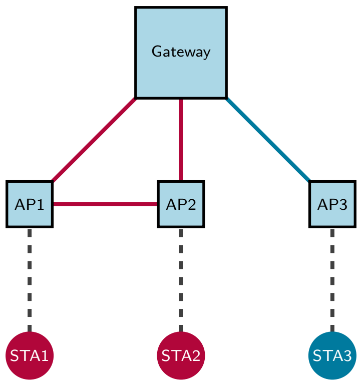

Figure 9: The virtual L2 structure when two stations are connected to the same overlay network on different APs, another station is connected to a third AP; colors represent virtual networks, dashed lines are wireless connections

Suppose STA1 wants to send a L2 packet to STA2 and both are connected to the overlay network with ID 1 (red). For this example, we assume STA1 has the MAC address 00:00:00:00:01 and STA2 has the MAC address 00:00:00:00:02. When each of them connected to the network, they used DHCP or NDP to get an IP address. That way, their APs received a packet from their MAC addresses and learned that the stations were connected to them. This information was then distributed over the RIDS, so that now both APs know at which APs with which corresponding IP addresses the two MACs are reachable. When STA1 now sends a packet to STA2 with MAC 00:00:00:00:02, AP1 will look up that MAC address in its forwarding table, encapsulate the packet, mark it with the overlay network ID 1, and send it to the IP of AP2. AP2 now decapsulates that packet and forwards it to the L2 domain of the overlay network with the ID 1 at that AP. Then, STA2 can ultimately receive the packet.

4.2.2 Control Plane

In our system, the control plane distributes reachability information for APs and other endpoints. It consists of the RIDS we proposed in Section 4.1.2 to which the overlay network endpoints connect.

When setting up a new AP, that AP is configured with a list of IP addresses for nodes in the RIDS. The AP then connects to one or multiple nodes of the RIDS on startup. Once a new station connects, it can then request the current system state for that overlay network from the RIDS.

4.3 Processes and Communication

In this section, we describe the central processes in our system and their corresponding message exchanges.

All communication on the control plane happens over reliable unicast messages, using, e.g. TCP or SCTP. Connection establishment and acknowledgements are not shown here. Every endpoint is only functional once at least one connection to the RIDS has been established.

These connections stay open until an endpoint is turned off. In case a connection fails, an endpoint attempts to reestablish the connection and then gathers the current system state from the RIDS, just like when it joined the system in the first place.

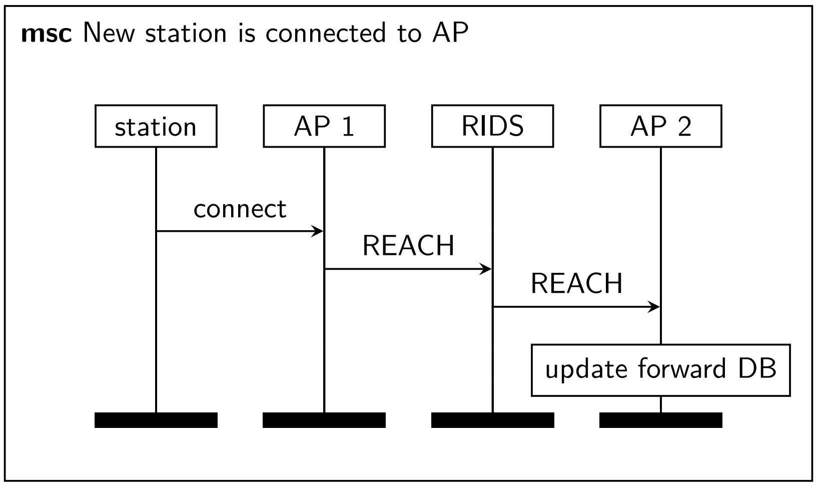

Figure 10: msc diagram of the reachability information of an new station that has connected to an AP being distributed. AP2 already has a station in the same overlay network the new station will be assigned to

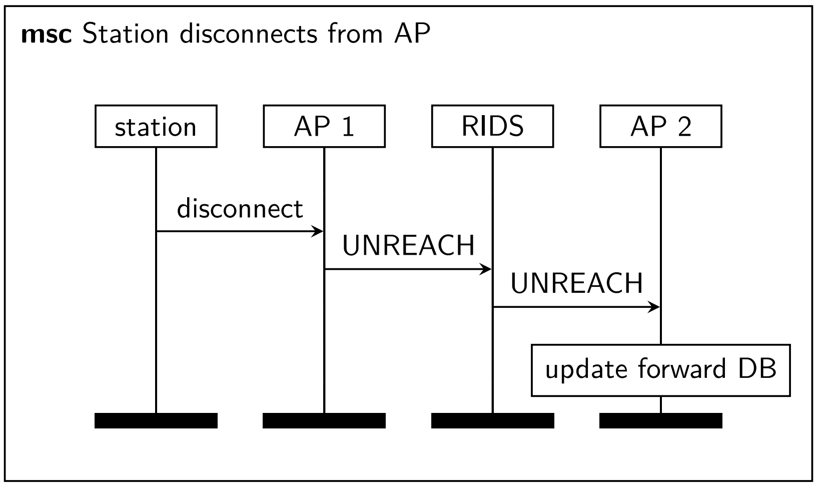

Figure 11: msc diagram of the communication happening when a station disconnects from the AP

4.3.1 Connecting a New Station to an Access Point

Assume that a new station connects to the Wi-Fi network. First, the OverlayNetworkID is calculated from its MAC address, and it is internally assigned to that overlay network. Once the first L2 packet from the device is received, the AP will detect that a device with that MAC address is available. This information is then sent out to the RIDS by our software, as can be seen in Figure 10. From there, the information is sent to all participating endpoints, which update their forwarding tables.

4.3.2 Disconnecting a Station From an Access Point

When a station disconnects from an AP, the operation is similar to when a station connects, as can be seen in Figure 11. However, instead of REACH messages, the AP now sends UNREACH messages.

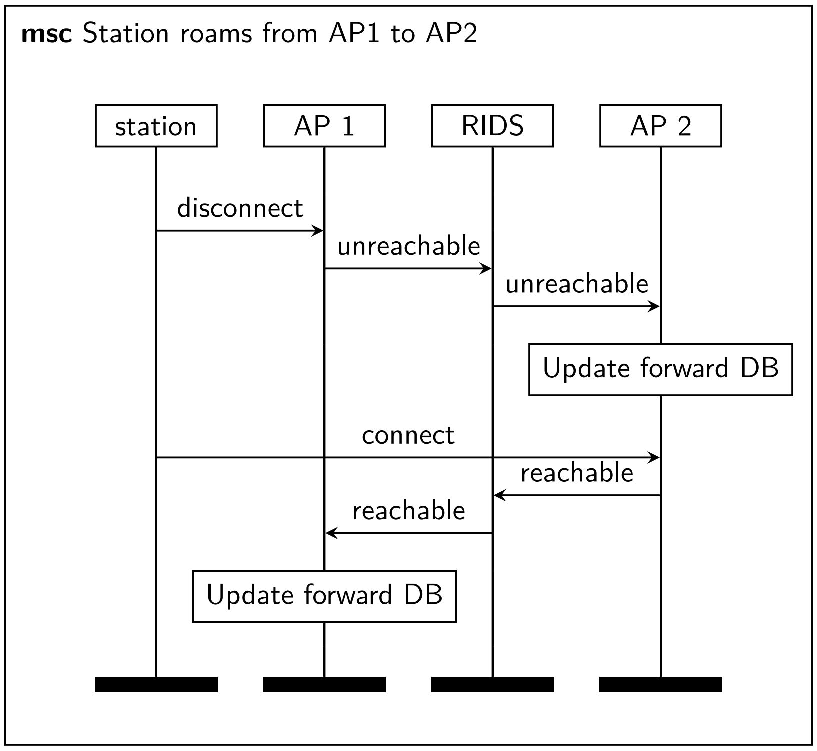

Figure 12: msc diagram of a station roaming from AP1 to AP2 with messages in ideal order

4.3.3 Roaming Between Access Points

Roaming between different APs is similar to disconnecting and reconnecting to another AP. The new AP sends out REACH messages to the system, overwriting the information from the old AP. Once the old AP detects that the station has disconnected because of a disconnect Wi-Fi event or a timeout, it will send out UNREACH messages.

The time frame of the roaming process might be shorter than the time needed to distribute reachability information throughout the system. This could result in the UNREACH messages or old REACH messages arriving at an AP after the new REACH messages have already been received.

However, the distributed storage system will always recover from such a situation thanks to both our chosen consistency model and the self-healing mechanism (see Section 3.2.1).

In the time frame after a roaming process but before the new reachability information has propagated to all participating endpoints, data packets sent to the roaming station will not reach their destination. This may especially be the case when a station quickly roams between access points.

Figure 12 shows a case in which messages are sent out and received in the correct order. The consistency model also properly handles the case where the disconnect event in the original AP is triggered after the connection event in the new AP.

Acronyms

ARP Address Resolution Protocol

BUM Broadcast, unknown-unicast and multicast traffic

DHCP Dynamic Host Configuration Protocol

ESS Extended Service Set

NDP Neighbor Discovery Protocol

References

[1] Arjan Koopen. “The importance of broadcast/multicast filtering in high density Wi-Fi networks”. WLPC. 2016. URL: `https://eventinfra.org/airtime`.

[2] M. Mahalingam et al. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.* RFC 7348. RFC Editor, Aug. 2014. URL: `https://www.rfc-editor.org/rfc/rfc7348.txt`.