Flow management and load balancing in dynamic heterogeneous LANs

Flow Management and Load Balancing in Dynamic Heterogeneous LANs

Tom De Schepper, Steven Latré and Jeroen Famaey
University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science, Belgium
firstname.lastname@uantwerpen.be

***Abstract*—Today's local area networks (LANs) consist of an ever-expanding number of heterogeneous consumer devices and communication technologies. Despite supporting multiple technologies, those devices tend to connect to the Internet using a single technology, based on predefined priorities. This static behavior does not allow the network to unlock its full potential, which becomes increasingly more important as the Quality of Service (QoS) requirements of services grow. Moreover, existing approaches make use of theoretical models that assume, unrealistically, full knowledge on the network. To this extent, we present a multi-technology flow-management load balancing framework that dynamically re-routes traffic through heterogeneous networks, in order to maximize the global throughput, based on changing network conditions and QoS demands. Along a problem formulation, we focus on the estimation of wireless and dynamic network characteristics and provide a thorough evaluation through simulations and a prototype implementation. We show that our framework is indeed capable of responding to dynamic network events in real-time and offers an increased overall throughput by optimally using the network's capacity. This results in a throughput increase of around 20 % on average.**

***Index Terms*—flow scheduling, load balancing, heterogeneous networks, local area networks.**

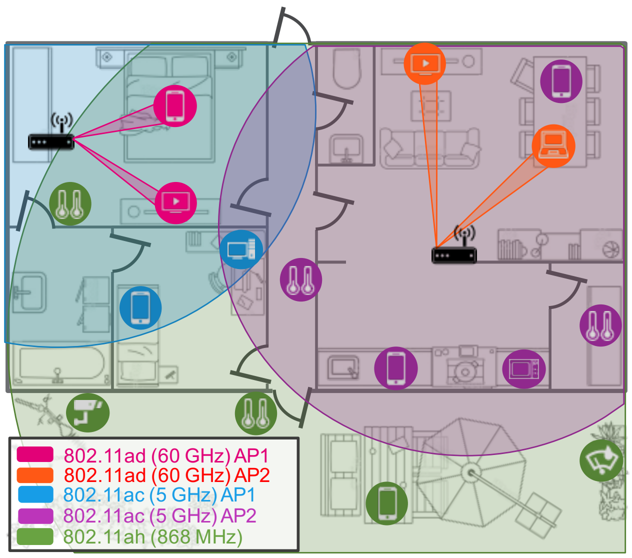


Fig. 1: Example of a future heterogeneous LAN, consisting of a variety of access points (APs) with heterogeneous network technologies and devices

I. Introduction

Over the past years, local area networks (LANs) have seen a large transformation and growth. This swift change has introduced an ever-expanding collection of heterogeneous consumer devices, such as smartphones, tablets, Internet of Things (IoT) devices, laptops and smart televisions (TVs). These devices are equipped with the ability to connect to the Internet using a variety of different network technologies (e.g., Ethernet, power-line communications, different Wi-Fi standards and LTE). Furthermore, these devices consume all kinds of different services with increasingly stringent requirements (e.g., Voice-over-IP, Video on Demand, IP-TV or even Virtual Reality applications). The previously described evolutions have raised a management puzzle: on one hand, modern multimedia services have stringent quality requirements and are very sensitive to network disruptions and degradations (e.g., high latency, congestion or link failures). On the other hand, current LANs are generally managed in a mostly static manner, unable to automatically react in a timely fashion to temporary disruptions that cause Quality of Service (QoS) or Quality of Experience (QoE) degradations. Moreover, the unique nature, characteristics and behavior of each network technology further increases overall management complexity.

Many modern consumer devices are equipped with multiple network interfaces (e.g., tablets and smartphones with IEEE 802.11n and IEEE 802.11ac, or laptops and smart TVs with Wi-Fi and Ethernet). Currently, such devices generally statically select one of the available technologies or intra-technology configurations (e.g., 2.4 GHz or 5 GHz Wi-Fi) based on predefined priorities (e.g., Ethernet, before 5 GHz Wi-Fi, before 2.4 GHz Wi-Fi). In some cases, the user can manually override these priorities, but no method exists for dynamically switching between interfaces or using multiple ones simultaneously. Making network interface selection automatic and dynamic would enable optimizations such as multipath routing, load balancing and dynamic path reconfiguration, aiding to unlock the current heterogeneous LAN environment's full potential. The efficiency of such a dynamic approach will only increase as more and more technologies are introduced to residential networks and other LAN environments. Examples include sub-1 GHz Wi-Fi (i.e., IEEE 802.11ah), 60 GHz Wi-Fi (i.e., IEEE 802.11ad or IEEE 802.11ay) and visible light communications (e.g., Li-Fi). Figure 1 illustrates the envisioned future LAN environment, consisting of a plurality of APs offering a variety of heterogeneous network technologies to a multitude of heterogeneous devices.

The implementation and usage of automatic and dynamic interface selection requires the introduction of an extensive level of control and monitoring in a LAN. For instance, it should be possible to handover devices and traffic flows between different technologies and to have accurate real-time information on the behavior of traffic flows (e.g., throughput, latency, packet loss) and link quality (e.g., Received Signal Strength Indicator (RSSI), theoretical and actual capacity).

Different types of solutions have been proposed for inter-technology handovers, such as (i) the use of a virtual Medium Access Control (MAC) layer, (ii) Software-Defined Networking (SDN) approaches, and (iii) band steering. The use of a virtual MAC (e.g., proposed by the IEEE 1905.1 standard) enables the transparent use of, and switching between, the available technologies on a device [1]. Traffic flows can be directed to certain network interfaces based on packet header matching rules, that can be dynamically adapted, for instance, by a centralized controller. The SDN approach has been proposed as an alternative that is easier to roll-out, as it is already more commonly used [2]. An SDN controller can change configurations on, for instance, virtualized switches or APs under it control, but is thus limited to the infrastructure side of the network. Finally, band steering does not require any changes to the device itself, as it is fully AP driven [3]. However, it can only be used to switch between different frequency bands of a single wireless technology family supported by a single AP (e.g., to switch from 2.4 GHz to 5 GHz Wi-Fi), incurs a longer disconnection delay, and breaks end-to-end connectivity, such as TCP sessions.

Although these above described solutions enable the available technologies and allow for dynamic flow redirection, they lack the required intelligence for, among others, selecting suitable traffic routes. Therefore, this paper proposes a transparent load balancing and routing framework for heterogeneous LANs that can be used on top of all the previously described approaches. The presented framework extends our previous work with strong improvements towards scalability and applicability in more realistic scenarios [4]. The framework takes into account the communication technologies available to each device and selects the most suitable one based on estimated application requirements and network conditions, which may vary over time. The framework aims to find a global optimal scheduling configuration for all the traffic flows in the network, in order to achieve maximum global throughput. Existing load balancing algorithms for LANs assume full knowledge on the network and are based on theoretical models that, amongst others, do not include the characteristics of the different technologies. In contrast, the proposed framework improves upon existing solutions in three ways. First, rather than assuming to know flow throughput requirements and dynamic network conditions, it estimates them using real-time monitoring information. Second, it takes into account the specific nature of wireless networks, where users do not have dedicated network resources but a shared medium instead. Third, the framework offers improvements in terms of scalability and computational efficiency, thereby allowing for more realistic use cases.

The contributions of this paper are threefold. First, we model the load balancing problem in heterogeneous LANs as a Mixed Integer Linear Programming (MILP), which can be solved using existing linear programming. Second, we present a real-time method for responding to dynamic network events and for estimating the capacity of wireless network technologies, taking into account the impact of neighboring stations. Third, we describe a real-life prototype implementation and demonstrate its impact and potential in a representative use case. We thoroughly evaluate the proposed framework under a variety of scenarios in both simulated and prototype environments. A comparison is provided to a static baseline, as well as state-of-the-art solutions [4].

The remainder of this paper is structured as follows. We start by giving an overview of the current state of the art in Section II. Next, we state the problem formulation and a network model in Section III and present our framework in Section IV. Section V discusses the simulation results, while the prototype implementation is discussed in VI. Finally, conclusions are provided in Section VII.

II. Related Work

Today's LANs consist of a multitude of heterogeneous communication technologies used by consumer devices to connect to the Internet. In this section we discuss the state-of-the art concerning management of challenging heterogeneous wireless networks.

A. IEEE 1905.1 standard

A key aspect in terms of user friendliness and QoE is the abstraction from network connectivity, as users do not want to struggle with the low-level specifics of each network technology. The main focal points so far has been the development of a unified high bandwidth environment that exploits the multiple interchangeable available network technologies on most devices [5]. Earlier work towards a converged gigabit home network and an Inter-Mac architecture, eventually resulted in the definition IEEE 1905.1 standard [1, 6, 7]. This architecture introduced a virtual or hybrid MAC layer on top of the current data link layer (i.e., OSI layer 2) to combine all the heterogeneous MAC interfaces in a transparent manner [8]. The defined abstract data link layer (i.e., a hybrid MAC) enables the transparent use of switching between several wired and wireless network technologies. For this purpose, all devices on the network require their own unique virtual MAC address. The implementation of the IEEE 1905.1 standard results in a simplified set-up, configuration and operation of network devices with heterogeneous technologies and allows for dynamic rerouting of flows. Currently, it supports Ethernet, Wi-Fi, power-line home plug, and Multimedia over Coax (MoCA), but a wider range of technologies could be supported in the future. Despite its potential, IEEE 1905 never really took off. Since its release in 2013, it has been under active development, without follow-up releases and no products exist yet that support it.

B. SDN-based approaches

An alternative for a virtual or hybrid MAC can be found in bringing SDN techniques into the LAN [2, 9]. In particular, the proposed solutions make use of the OF communication protocol to access and control the gateway and other infrastructure devices within the network [10, 11]. The installation of an Open vSwitch (OVS), a virtual multilayer switch that is controllable over the OF protocol, makes it possible to perform transparent handovers between the different available technologies, such as Ethernet, 2.4 GHz Wi-Fi, and 5 GHz Wi-Fi. An OF controller (e.g., Ryu) is used to query the OF-enabled devices for real-time monitoring information to identify different data flows and their requirements by the exchange of OF stats request and reply message. Link quality information can be acquired in a similar fashion. Furthermore, the controller can initiate the handover of a subset of flows to another technology by changing the outgoing OF port.

C. Load balancing in heterogeneous LANs

Although both IEEE 1905.1 and the SDN approach specify the features to enable dynamic flow redirection, they do not define the algorithms for selecting suitable paths per flow. Sahaly and Christin define a framework for heterogeneous home networks that includes a per-flow decentralized load balancing algorithm [12]. It is capable of reactively distributing incoming flows on the available links. However, the load balancing technique only takes local parameters per device into account and only a theoretical description is given without any realistic results. Macone et al. propose a per-packet load balancing algorithm [8]. Per-packet load balancing can better exploit the network resources and thus theoretically provides better results. However, per-packet load balancing in combination with TCP can result in unnecessary retransmissions, due to out-of-order packet arrival. The authors do not elaborate on how to solve this problem, or its effects on end-to-end performance. Furthermore, the algorithm runs centralized on the gateway and assumes full instantaneous knowledge of network resources and conditions.

A decentralized load balancing algorithm specifically for heterogeneous wireless access networks was proposed by Oddi et al. [13]. This algorithm relies on a multi-connection transport layer in order to cope with the drawbacks of per-packet load balancing in the case of TCP. The proposed algorithm is based on the Wardrop equilibrium and does not take into account the fact that users do not have dedicated network resources when using wireless technologies. In general, Olvera-Irigoyen et al. have shown that determining the actual available bandwidth on the links has a big impact on the results of distributing the flows [14].

Recent load balancing solutions focus also on energy optimization. Bouchet et al. proposed an algorithm that aims to reduce energy consumption and use the most energy efficient link while still providing a good QoS [15, 16]. This is done by assuming the energy consumption model is known in advance, and not by real-time measurements on the devices.

D. HetNets or 5G Networks

Finally, related research can also be found in the so-called heterogeneous networks (HetNets) or 5G networks, where load balancing, dividing connections across different technologies and handovers across LTE/UMTS, Wi-Fi, and WiMAX have been investigated. Most research proposes technology specific solutions that are capable of performing handovers or load balancing across only two of these technologies (e.g., LTE and Wi-Fi or Wi-Fi and WiMAX) [17]. The decision to perform a handover is made centrally by the base station and different decision strategies have been proposed using, among others, utility functions, multiple attributes decision making, Markov chains, game theory, and user location [17, 18]. Additionally, load balancing policies also look at the number of connected devices to a base station. These strategies take only a limited number of parameters into account, with RSSI and Signal to noise Ratio signal to noise ratio (SNR) being the most popular ones [19, 20]. Open issues include, for instance, the development of more generic solutions, better support for mobility, the use of multi-criteria decision functions, supporting different QoS classes and the increase of QoS during or after handovers [21]. Current solutions are technology specific and do not take actual application or QoS parameters and objectives into account, making them unsuitable for use with QoS-sensitive or mission critical services.

E. Summary

To summarize, current research on load balancing in LANs mostly focuses on the development of theoretical models that assume the detailed knowledge of flow throughput requirements and dynamic network conditions. The specific nature of wireless networks (e.g., interference, link quality variability) and the typical behavior of TCP are also ignored. Furthermore, approaches developed within the domain of 5G networks, are technology specific and not suitable for matching QoE requirements. In contrast, the work presented in this paper uses real-time distributed monitoring information, rather than assuming complete knowledge on the network and its flows. Moreover, the specific characteristics of wireless networks, such as mutual interference among senders, are also considered explicitly. To assure practical applicability and responsiveness to dynamical network changes, the computation time has been strongly reduced, in contrast to our previous work [4]. Finally, we assume that fairness is provided by the overlaying transport layer protocols, rather than trying to enforce our own flow rates.

III. Problem definition and network model

An example heterogeneous LANs was already shown in Figure 1. This examples shows how the network at your home might look like. The figure shows a variety of devices, each connected to the residential gateway via one or multiple technologies. Besides wireless connectivity, as depicted in the figure, connectivity can also still be wired (e.g., Ethernet). These topologies are expected to become more complex and diverse in the future with the addition of newer technologies (e.g., 60 GHz Wi-Fi) and more devices (e.g., for smart homes). This evolution will further increase the burden for the management of such heterogeneous networks, while also user expectations will continue to rise. A key feature, needed for providing reliable QoS, is the transparent handover of traffic between different technologies and automatic selection of paths for different traffic flows through the network. Hereby, the different requirements of applications and capabilities of technologies will need to be taken into account. We now present a model for such heterogeneous networks that takes into account the different technologies and devices and their capabilities.

A LAN is modeled as a multi-graph defined as a tuple (S,G) where:

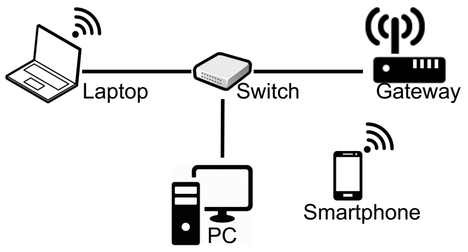
- S is a set of stations $\{s_1, s_2, ..., s_n\}$. These stations represent the different devices within the LAN, both consumer (e.g., a smartphone or laptop) and infrastructure devices (e.g. a switch or gateway).
- G is the set of all collision groups $\{g_1, g_2, ..., g_n\}$. A collision group $g \in G$ is defined as a set of stations $\{s_1, s_2, ..., s_n\}$ with $s_i \in S$ that share a common path to the gateway over a certain technology with a certain capacity. In other words, a collision group encapsulates all the stations that can interfere with each other since they share the capacity of a technology. For a wired technology, like Ethernet, there will be a collision group for each device that is directly connected to the gateway. For instance, all stations that are connected to a switch, with that switch having a single connection to the gateway, share one collision group. For wireless technologies, like Wi-Fi, there is a single collision group per technology per AP, due to the shared medium inherent to wireless technologies.

Furthermore, we define the following sets and elements to complete the network model:

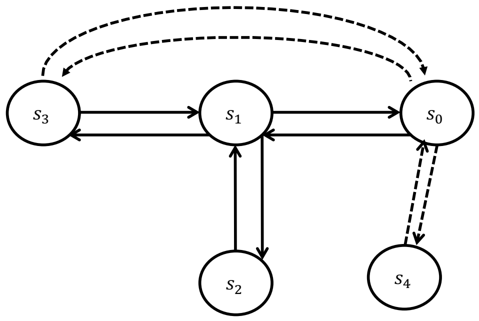
- $\forall g \in G : \exists c_g \in \mathbb{R}_0$ represents the (theoretical) capacity of that collision group $g \in G$.
- $\forall s \in S : G_s : \{\forall g \in G \mid s \in g\}$ defines for each station the set of all the collision groups to which it belongs (i.e., one for each technology it supports).
- Finally, we also define two subsets to distinguish between different characteristics of technologies:
  - $G_{fdup} \subseteq G$ is the set of full duplex groups;
  - $G_{hdup} \subseteq G$ is the set of half duplex groups;
  - with $G_{fdup} \cap G_{hdup} = \emptyset$.

  This distinction between full and half duplex collision groups (and the corresponding actual technologies) is required because this has a significant impact on the capacity and behavior of these technologies. Ethernet is probably the best known as a full duplex technology, while Wi-Fi is usually deployed as half-duplex, with up- and downlink sharing the medium.

An example of a simple LAN topology is shown in Figure 2a. This network consists of a gateway (that also serves as a Wi-Fi AP), a switch and three consumer devices: a smartphone, a laptop, and a (fixed) computer. The latter two are connected via the switch to the gateway over Ethernet. Furthermore, the laptop also has Wi-Fi connectivity, like the smartphone. This network can be represented by means of the previously described model, as is shown in Figure 2b. In an actual network representation there will likely also be a second (5 GHz) Wi-Fi network, but this was omitted from this example due to readability reasons. Each of the devices is represented by a station $\{s_0, s_1, s_2, s_3, s_4\}$. In this example there are two collision groups: the first one contains all the stations connected by Ethernet since they share a common link between the gateway and the switch. This first group consists of the following stations $\{s_0, s_1, s_2, s_3\}$ and is denoted by the full arrows in the figure. This group is a full duplex collision group. The second collision group denoted by the dashed arrows, represents the Wi-Fi connection between the stations $\{s_0, s_3, s_4\}$ and is thus half duplex. In contrast to the model we described in our previous work, we do not model explicitly the separate links and technologies [4]. This simplification helps in minimizing the overall model size and limiting the number of constraints, allowing for applicability in significantly larger scenarios, as we will demonstrate later on.



(a) Network topology



(b) Multi-graph representation

Fig. 2: Example simple heterogeneous network topology together with its multi-graph network model representation

In addition to the network topology, traffic flows going through the network also need to be modeled. Let us define F as the set of all flows. A flow $f \in F$ is a triple $< s_f, r_f^{in}, r_f^{out} >$ with $s_f \in N$ the station within the LAN that is the source or destination of the flow within the network, r_f^{in} the incoming desired rate of f $\in \mathbb{R}^+$ and r_f^{out} the outgoing desired rate of f $\in \mathbb{R}^+$. Note that we do assume that the gateway is always one of the two endpoints of the flow, while the other is denoted by s_f. Furthermore, we separate the desired rate of the flow between the incoming and outgoing rate. This allows us to more precisely schedule all flows across the different paths, and it is thus possible that incoming and outgoing packets of a flow are assigned a different route. To

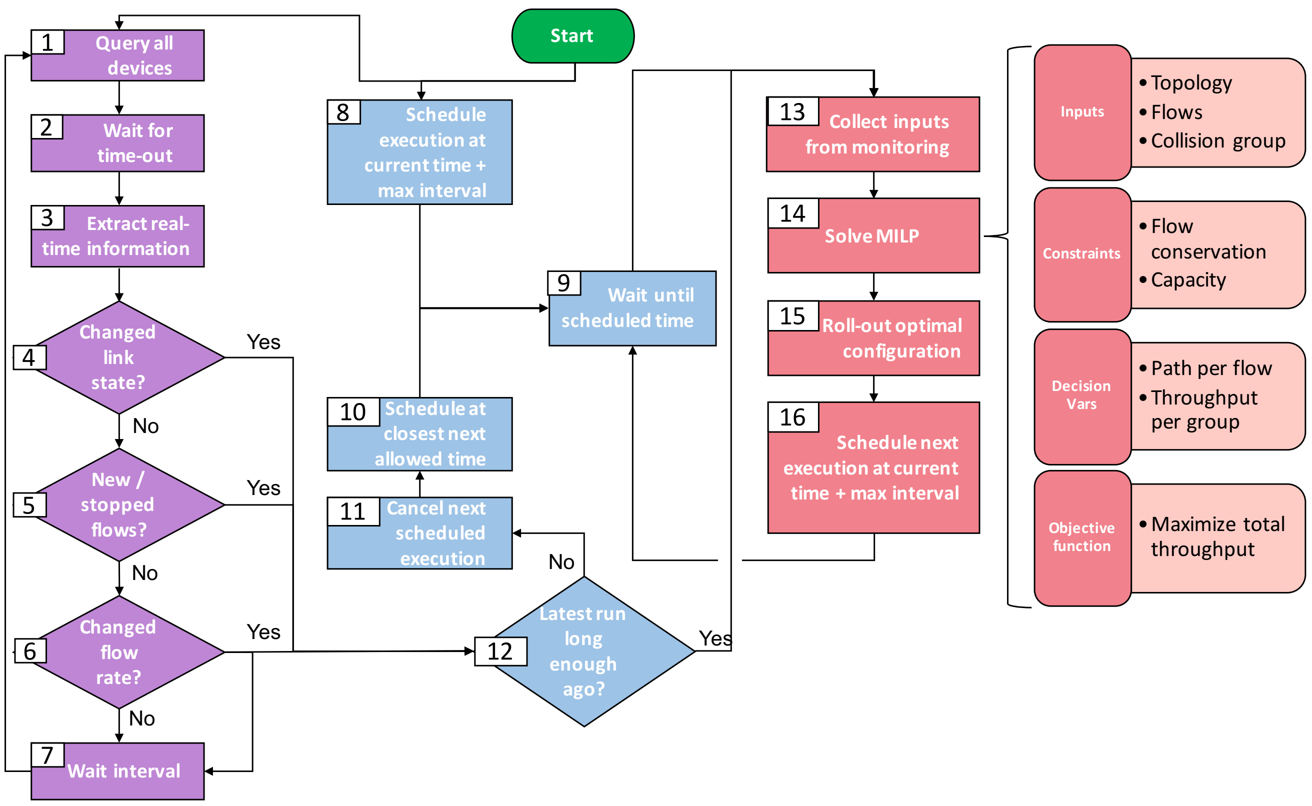

Start
1 Query all devices
2 Wait for time-out
3 Extract real-time information
4 Changed link state?
5 New / stopped flows?
6 Changed flow rate?
7 Wait interval
Yes
No
8 Schedule execution at current time + max interval
9 Wait until scheduled time
10 Schedule at closest next allowed time
11 Cancel next scheduled execution
12 Latest run long enough ago?
13 Collect inputs from monitoring
14 Solve MILP
15 Roll-out optimal configuration
16 Schedule next execution at current time + max interval
Inputs
• Topology
• Flows
• Collision group
Constraints
• Flow conservation
• Capacity
Decision Vars
• Path per flow
• Throughput per group
Objective function
• Maximize total throughput

Fig. 3: Flow chart illustrating the different steps of the proposed framework

clarify, for a TCP flow originating from some web server, the incoming rate is the rate of the data traffic, while the outgoing rate is the one of the ACKs. In our example in Figure 2, a flow from the gateway to the laptop, with, for instance, an incoming rate of 10 Mbps and an outgoing rate of 1 Mbps, will be represented as the triple $< s_3, 10, 1 >$.

IV. Framework Description

This section presents the proposed heterogeneous load balancing model and framework. We start with an overview of the entire framework and all its different components. Afterwards, we discuss the important components and parameters.

A. *Framework description*

The framework consists of a number of different components that interact with each other. This interaction is shown in Figure 3. There are three main components: the monitoring, the scheduling of the framework and the execution of the MILP formulation. On initialization, the monitor loop is started by sending for the first time a request message to all the devices in the network (step 1). Afterwards, in step 2, a time-out is started, indicating the time period in which the stations should respond. In parallel of step 1, upon initialization, the execution of the MILP is scheduled at the latest allowed time (denoted by *max interval*) in step 8. This value denoted the maximum interval between two consecutive runs of the MILP.

When the timeout for the receiving of monitoring information has passed, the received information is processed and compared with previous received information (if existing) in step 3. We check for three dynamic events in particular: a changed link state (i.e. a link or technology that has gone down or became available again), if new flows have arrived or existing ones have stopped, and if the rate of the flows have changed substantially (denoted by a threshold value). These checks are, respectively, denoted as steps 4 to 6 in Figure 3. If at least one such event is detected, the MILP formulation is scheduled to be executed if possible in step 12, in order to allow the framework to react to dynamic changes in the network. After checking for such changes, the monitoring loop waits a certain period of time in step 7, denoted by an interval value, before resending requests for monitoring information and restarting the whole process by going back to step 1.

The rescheduling of the execution of the MILP (step 12) is only allowed when enough time has passed since the last execution of the MILP, denoted by the *min interval*, to avoid unnecessary oscillations. If this interval has already passed, the MILP can be solved immediately (by moving on to step 13). Otherwise, the next execution is rescheduled to the earliest allowed time(step 10 and 11). Furthermore, upon the execution of the MILP, the necessary information is gathered from the stored monitoring data (step 13), before the allowance of a solver to calculate an optimal configuration in step 14. After calculation, the configuration is rolled-out across the entire network and the next execution is scheduled at the latest allowed time , in respective steps 15 and 16. At the right hand side of Figure 3 the different elements of the MILP formulation are depicted. In the next section, we focus on presenting and explaining this model into details. Afterwards, we also disclose information on the actual roll-out of the calculated configuration (step 15) and discuss which values to take for the execution parameters (e.g., time-out values or lower- and upper-bounds).

B. MILP formulation

The LAN flow scheduling problem considered in this paper is modeled as an MILP formulation, which consists of the necessary inputs, decision variables, an objective function, and a set of constraints. This model can be solved (i.e., to find the value of the decision variables, given the constraints and objective function) using a variety of optimization algorithms and heuristics. In this paper, the Gurobi solver is used, which finds the optimal solution to the problem. In contrast with our previous work, the goal is to maximize the throughput across all the collision groups, instead of calculating and maximizing the rate assigned to individual flows [4].

The inputs of the presented MILP consist of the previously described network and flow model. We have thus the following sets as input: S, G, G_s, G_{fdup}, G_{hdup}, and F.

Furthermore, we define the following decision variables:

- $\tau_g^{in} \in \mathbb{R}^+$; this variable defines the total incoming rate assigned to a collision group in $g \in G$. In other words, a summation over the rates of all incoming traffic flows that will be using the underlying technology of the collision group.
- $\tau_g^{out} \in \mathbb{R}^+$; this variable defines the total outgoing rate assigned to a collision group in $g \in G$. In other words, identical to the previous defined decision variable, but for outgoing traffic.
- $\lambda_{f,g}^{in} \in \{0, 1\}$; this variable represents the path for the incoming traffic of a flow. If the incoming traffic of flow $f \in F$ is scheduled over a collision group $g \in G$ then $\lambda_{f,g}$ = 1, otherwise it equals 0.
- $\lambda_{f,g}^{out} \in \{0, 1\}$; this variable represents the path for the outgoing traffic of a flow. If the outgoing traffic of flow $f \in F$ is scheduled over a collision group $g \in G$ then $\lambda_{f,g}$ = 1, otherwise it equals 0.

As an objective function, the model maximizes the total rate (bandwidth) of the traffic, both incoming and outgoing, across the entire network:

- $max\left(\sum_{g\in G} \tau_g^{in} + \tau_g^{out}\right)$

Finally, we define the following constraints:

- We first define three constraints that make sure the capacity of the individual collision groups and their underlying technologies is not exceeded:
  - For all full duplex collision groups, the total incoming rate across all flows assigned to the group should not exceed the capacity of the group:
    $\forall g \in G^{fdup} : \tau_g^{in} \leqslant c_g$
  - For all full duplex collision groups, the total outgoing rate across all flows assigned to the group should not exceed the capacity of the group:
    $\forall g \in G^{fdup} : \tau_g^{out} \leqslant c_g$
  - For all half duplex collision groups, the total of the incoming and outgoing rate across all flows assigned to the group should not exceed the group's capacity:
    $\forall g \in G^{hdup} : \tau_g^{in} + \tau_g^{out} \leqslant c_g$
- Next, we define two constraints that make sure that the rates assigned to a collision group do no exceed the total rate of the traffic that is assigned to the collision group:
  - The total incoming rate for a collision group should not exceed the total amount that is desired by the flows:
    $\forall g \in G : \tau_g^{in} \leqslant \sum_{f\in F} \lambda_{f,g}^{in} \cdot r_f^{in}$
  - The total outgoing rate for a collision group should not exceed the total amount that is desired by the flows:
    $\forall g \in G : \tau_g^{out} \leqslant \sum_{f\in F} \lambda_{f,g}^{out} \cdot r_f^{out}$
- The third group of constraints guarantees the conservation of flows in the network:
  - The incoming traffic for every flow needs to be assigned to exactly one collision group that contains its endpoint station:
    $\forall f \in F : \sum_{g\in G_{s_f}} \lambda_{f,g}^{in} = 1$
  - The outgoing traffic, for every flow, needs to be assigned to exactly one collision group that contains its endpoint station:
    $\forall f \in F : \sum_{g\in G_{s_f}} \lambda_{f,g}^{out} = 1$
  - The incoming traffic of each flow, should be assigned to exactly one collision group:
    $\forall f \in F : \sum_{g\in G} \lambda_{f,g}^{in} = 1$
  - The outgoing traffic of each flow, should be assigned to exactly one collision group:
    $\forall f \in F : \sum_{g\in G} \lambda_{f,g}^{out} = 1$

C. Complexity analysis

In order to have a responsive framework, it is important to be able to rapidly solve the MILP formulation. Typically, a branch-and-bound algorithm is used by a solver for solving an MILP. However, upfront different optimizations are performed to reduce the problem size (e.g. presolve or cutting planes). Since the number of constraints in a model has a strong impact on the complexity and solve time, we compare the number of constraints in the presented MILP to the number of constraints in our previous work [4].

This previous formulation consists out of 10 constraints: first, the capacity constraint depends on the number of collision groups. Second there are 8 flow conservation rules of which 4 depend on the number of flows, while 3 are depending on the number of flows and the numbers stations. Furthermore, there is a final conservation rule per a flow and a unique pair two stations. Finally, there is a TCP fairness constraint per combination of flow and collision group. If we assume 1 flow per station, the number of stations equals the number of flows, we get the following amount of constraints:

$$|G| + 4|F| + (3|F| \cdot 3|F|) + |F|^3 + (|G| \cdot |F|)$$

In contrast, our novel formulation contains only 9 constraints of which the first 5 only depend on the number of collision groups and the last 4 only on the number of flows (if we respect the order of definition in the Section IV-B). Thus, we have the following number of constraints:

$$4|G| + 5|F|$$

We can conclude that the presented model has a strongly reduced number of constraints, and thus a lower complexity.

D. Technology capacity estimation

In the previous section we defined our MILP and introduced the notion of collision groups. These are the encapsulations of all the stations that interfere with each other due to the sharing of the capacity of a certain technology. An important aspect that was not explicitly discussed is determining the capacity of the groups and their underlying technologies. Since the goal of the framework is to optimize the utilization of the network and schedule the flows across the different technologies, the capacity of these technologies should be determined as accurate as possible. The importance of this was already highlighted in our previous work where we experimentally determined a correction factor per technology [4]. When multiplying the theoretical capacity by this factor, a more realistic value for the capacity of a technology is obtained. While this method is sufficient for Ethernet, the practical capacity of wireless technologies is harder to determine, as it depends on the number of users, their traffic and location. Using the constant correction value in the framework showed to be inaccurate in dense wireless environment with many devices.

As such, we define a linear function that will approximate the actual capacity of the different technologies, taking into account the number of stations that are using that technology at a certain period in time. For a certain collision group $g \in G$, we define:

$$\gamma(g, \alpha, \beta) = \alpha \cdot (\sum_{f \in F} \lambda_{f,g}^{in} + \lambda_{f,g}^{out}) + \beta$$

The parameters α and β are technology specific and we will discuss their determination in the next Section IV-E.

In the three capacity constraints defined in the previous section, we can thus replace the theoretical capacity of the collision group by the function γ:

- $\forall g \in G^{fdup} : \tau_g^{in} \leqslant \gamma(g, \alpha, \beta)$
- $\forall g \in G^{fdup} : \tau_g^{out} \leqslant \gamma(g, \alpha, \beta)$
- $\forall g \in G^{hdup} : \tau_g^{in} + \tau_g^{out} \leqslant \gamma(g, \alpha, \beta)$

E. Dynamic determination of α and β parameters

The actual capacities of the different technologies, in particular for wireless ones, are dependent on several parameters (e.g., configuration of APs, interference of other devices within or outside the network, and the amount of traffic in the network). Estimating each of these parameters is very challenging and is in some cases a separate research problem (e.g., interference modeling). In order to take these parameters into account, without the need for complex models, we propose an experimental method that can be applied in real-time. For each technology, a series of experiments is conducted where the number of stations and the flow rates are varied, each taking values from a predefined set. For instance, the number of stations can be varied from one up to fifteen, while the traffic rate per station can be varied between the theoretical capacity of that technology and a relative small value like 1 Mbps to cover both saturated and unsaturated cases. For each number of stations, the achieved rate for all relevant scenarios is averaged and stored. Scenarios where the desired rate of all stations is achieved, are not taken into account, since we want to determine the actual capacity of the network. Afterwards the list of stored maximum capacities is interpolated as a function of the number of stations, leading to the function γ, as described above. Note that for Ethernet, due to its full duplex capabilities, it is sufficient to vary only the traffic as the number of connected nodes does not affect practical link capacity. Therefore, the parameter $\alpha = 0$ for Ethernet. This method can be applied for each heterogeneous environment to capture the specific characteristics and can be rapidly re-executed if needed.

F. Estimating flow and network parameters

One of the novelties presented in this paper is the use of real-time monitoring information to estimate the desired flow rates and dynamic network conditions, rather than assuming this information is fully known by the framework. To enable this, the framework relies on a monitoring component that collects the needed information. Link Layer Discovery Protocol (LLDP) is an example of an underlying protocol that can be used, as it allows devices in a network to ask information regarding link quality to their neighbors. Link metric reporting and querying options can thus be defined based on the LLDP, in a similar fashion as is done by the IEEE 1905.1 standard [1]. The queryable metrics are: number of packet errors, the amount of transmitted and received packets, MAC throughput, link availability and theoretical physical rate. Metrics are periodically requested and are valid for a certain amount of intervals. The designed model only requires information on MAC throughput and theoretical physical rate, so only those two metrics are currently supported. In our previous work, we only looked at the transmission of data [4], and did not take management traffic and ACKs into account. Since we now schedule both incoming and outgoing traffic, we consider ACKs as well, allowing for more precise scheduling. To summarize, all flow and link parameters (e.g., source, destination and rate) are estimated in real-time and not known upfront. The network topology is assumed to be known, as the discovery of devices and links is also provided by the link metric reporting mechanism.

Because of the fact that we use an estimation (the measured MAC layer rate during the last interval) of the desired rate of a flow, it is possible that the MILP provides a non-optimal solution as it does not know the actual desired rate of the flow (which is application layer information that cannot be known at the network or MAC layers). For instance, if a flow actually desires 12 Mbps while going over a link with a theoretical capacity of 10 Mbps, the framework will only know the measured throughput, which will be lower than the actual desired 12 Mbps. In such a situation, the MILP might decide not to change the path of the flow, while this would have been the case if the actual desired rate was known. Section V compares performance of the framework both when estimating and knowing the desired rate of the flows.

G. Handover methods for heterogeneous LANs

Our framework selects the optimal route for each traffic flow and thus provides the intelligence that is not present in the flow redirecting solutions themselves. Although the proposed framework can be run on top of all dynamic flow redirecting solutions, the different solutions each have their characteristics and features. Some of them might even influence the performance or impact of the framework. We therefore discuss briefly all three solutions and list their (dis)advantages:

a) Virtual MAC: a virtual MAC layer unifies all available technologies per device in a transparent manner to the upper layers, allowing for seamless inter-technology handovers. A centralized controller can modify the header matching rules to steer traffic to certain interfaces. This solutions is the most complete one, but requires the implementation of the new layer on all the devices, both client and infrastructure side. It allows for control over both up- and down-stream traffic and for distributed intelligence. We will use this approach in our simulations.

b) SDN-based: the second option is the use of our framework on top of existing SDN controllers. The main benefit of this approach is that is can easily be rolled-out and no changes to the client devices need to be made. However there is now no distributed intelligence possible and the functionality depends on the SDN framework. For instance, with an OF controller and OVS it is possible to re-route downstream traffic, but not upstream traffic (unless an OVS is installed on the client-side as well). Due its rapid deployment it is perfectly suited for our prototype evaluation.

c) Band steering: the final option, known as band steering, is to force (wireless) devices to a certain frequency or channel. This is usually done by stopping the transmission of Wi-Fi beacons to a certain device and not responding to association requests on a specific frequency. This way an inter-technology handover can be performed, however it performance strongly depends on the client (in particular the operating system and drivers) itself and no QoS guarantees can be given. Often the duration of the handover will be in the order of seconds or tens of seconds and thus counteracting with our dynamic and responsive framework. Therefore we do not believe this approach to be suited. Note that this option can be used together with the SDN approach.

V. Simulation results and discussion

This section evaluates the proposed framework using simulation results obtained from the ns-3 event-based network simulator. First, the evaluation setup and scenario are discussed. Second, the framework's performance, in terms of achieved throughput and execution time, is evaluated in a variety of static and dynamic scenarios. A comparison to a baseline, using pre-configured interface selection based on hard-coded priorities, and previous work is provided.

A. Evaluation setup

The Gurobi Optimizer (6.5.1) is used to solve the MILP formulation, while the ns-3 event-based network simulator is used for the simulations. In this simulator, we implemented a complete virtual MAC layer, as described in Section IV-G. This allows for the transparent switching of flows between network interfaces, without breaking the end-to-end connection [4, 10]. We assume three technologies: Ethernet, 5 GHz Wi-Fi, and 2.4 GHz Wi-Fi. The Ethernet network is built out of full-duplex UTP cables with a theoretical throughput of 100 Mbps. In every network topology there is at least one Ethernet switch present that is connected to the gateway of the LAN. Furthermore, every topology has exactly one 5 GHz Wi-Fi network and one 2.4 GHz Wi-Fi network and the gateway serves as AP for both of them. For both types, the IEEE 802.11n standard with mode MCS 7 and GI $= 400\,\mathrm{ns}$ (short interval) is used. For the 5 GHz Wi-Fi network a 40 MHz channel is assumed, in contrast to the 20 MHz channel for the 2.4 GHz Wi-Fi network. This allows for a theoretical data rate of respectively 150 Mbps and 72.2 Mbps. To avoid oscillations in the decision making we sometimes delay a decision: between two consecutive executions of the MILP formulation should be at least 2 s and at most 10 s. Moreover, the threshold value for indicating a changed flow rate is 25 %, while the timeout for receiving monitoring information is 0.25 s.

In order to generate representative network topologies and conditions, several types of consumer devices are defined, each with different types of interfaces and flows. The device types and their supported interfaces are depicted in Table I. The exact number of each of these devices is randomly chosen between lower and upper bounds and varies depending on the scenario. Furthermore, three different flow types are defined. We assume that the rate of each flow is once again chosen uniformly at random between an upper and lower bound, based on the involved device. Moreover, within the static scenarios the flow rates do not change over time, while in the other scenarios the download flows will consume as much bandwidth as possible (reflecting their actual behavior). Assuming a static flow rate for the first part of the evaluations, allows us to estimate the difference of using monitored information for flow rate estimation compared to knowing the desired rates. The size of the downloaded file is uniformly at random chosen between 10 MB and 10 GB. We assign one flow per device and as such do not assume the concurrent usage of both Wi-Fi interfaces, as this is generally not supported by current hardware. The desired rate per flow and device type is also depicted in Table I. The flow rates were selected based on representative figures from literature of existing applications in these three categories [22]. We decided to focus on cases with only TCP traffic, as current Internet traffic is dominated by TCP. For example, Lee et al. reported over 95 % of Internet traffic to be TCP in 2010 [23].

For every described scenario, results are averaged over different randomly generated flow and topology configurations. To this extent, each experiment was repeated 20 times. We also report the standard error for each experiment over

TABLE I: Overview of the devices used in the scenarios, including their supported network technologies and flow rates

Device type	Supported network technologies			Rate boundaries per flow type		
	Ethernet	5 GHz Wi-Fi	2.4 GHz Wi-Fi	Download	Video stream	Video conference
Desktop PC	×			10–30 Mbps	8–20 Mbps	4–10 Mbps
Laptop	×	×	×	10–30 Mbps	8–20 Mbps	4–10 Mbps
HD Television	×	×	×	5–25 Mbps	10–20 Mbps	5–10 Mbps
4K Television	×	×	×	5–25 Mbps	15–25 Mbps	7.5–12.5 Mbps
Tablet		×	×	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps
Smartphone modern		×	×	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps
Smartphone old			×	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps

TABLE II: Topology parameters for the two scenarios

Device	Home	Office	Flows
Switches	1	3	N/A
Desktop PC	1	11	Download
Laptop	3	7	Download / Video conference
4k TV	2	1	Video stream
Tablet	1	0	All types of flows
Smartphone modern	3	4	All types of flows
Total devices	10	23	

these 20 repetitions. As a baseline for comparison, a static configuration is used where all devices are connected to a technology according to the following priorities: Ethernet, before 5 GHz Wi-Fi, before 2.4 GHz Wi-Fi. In terms of monitoring, metric link requests are sent once per second and the framework only takes into account the most recently received information. The following α and β per technology were experimentally determined: for Ethernet we can report that α is zero (due to the full duplex character) and β is 0.99 multiplied by the theoretical capacity of the Ethernet link (in this case 100 Mbps). For Wi-Fi the values are, for α and β respectively: for 2.4 GHz Wi-Fi -1.74 and 57.58, and for 5 GHz Wi-Fi -3.21 and 112.99. Furthermore, we also compare the present framework in terms of throughput and execution time with our previous work [4]. To distinguish between the two formulations, we denote the current version as Group-Based Scheduling Formulation (GBSF) and our previous work as Flow-Based Scheduling Formulation (FBSF), with respect to the different optimization goals of the formulations. Finally, all simulation were done using a single core of an Intel® Xeon® E5-2680 Processor running at 2.8 GHz and with 8 GB RAM.

B. Home and office scenarios

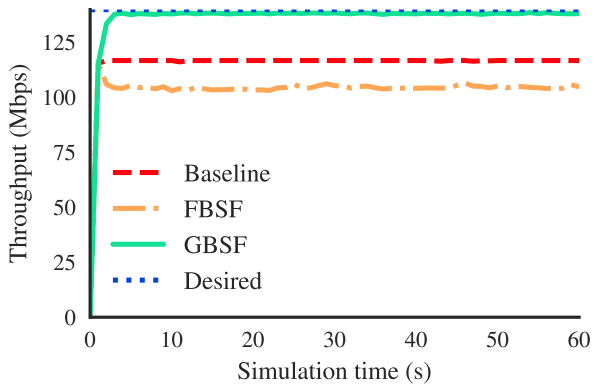
To demonstrate the impact of our framework in LANs, two basic scenarios were created to reflect two typical representative environments: a home and an office network. For each setup, the exact number of devices per type and the possible flows, with static flow rates, are depicted in Table II. The results for the home and office scenario are shown in Figure 4. The graphs compare the static baseline and the proposed load balancing framework GBSF to the total sum of the desired flow rates and the previous formulation FBSF [4]. Both scenarios show a strong improvement in the total throughput when using the proposed GBSF as compared to the baseline and FBSF.

For the home scenario, the average throughput for the baseline is 117.01 Mbps ($\pm$0.66), while the average throughput with GBSF increases to 137.62 Mbps ($\pm$3.16). This means that there is an increase of 20.62 Mbps or 17.62 %. Figure 4a shows that GBSF approximately reaches the desired traffic rate for all flows combined. The difference between the framework and the total desired rate is only 1.72 Mbps or 1.23 %. The figure also indicates that the older FBSF does not perform well and even performs worse than the baseline. The reason for this lays in the fact that FBSF does not take into account the characteristics of wireless spectrum as detailed as GBSF and ignores the impact of interference between different stations. Note that in previous work we showed that FBSF does provide an improvement, in comparison with the static baseline, for smaller scenarios where the capacity of the Ethernet links are limited to the 10 Mbps and the rates of the flows were more than half the size of the current values.

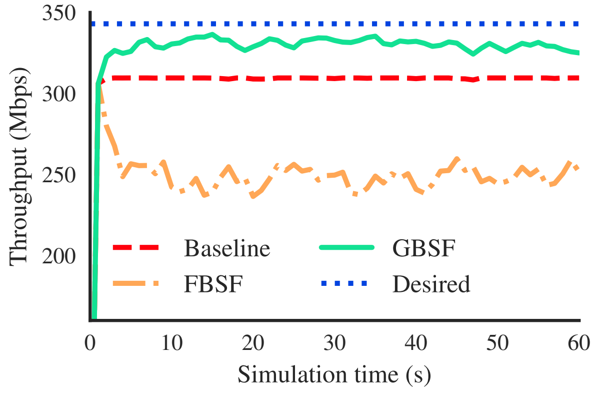
For the office scenario, the average throughput for the baseline is 310.39 Mbps ($\pm$2.67), while the average throughput with GBSF increases to 329.52 Mbps ($\pm$3.03). This means that there is an increase of 19.13 Mbps or 6.16 %. Despite this improvement, one can see on Figure 4b that GBSF does not reach the desired rates for all flows. The difference between the framework and the total desired rate is 12.40 Mbps or 3.63 %. However, the reasons for this can be found outside of our framework: first, this scenario is slightly oversaturated, as such the physical capacities of the network are being reached. Second, since an approximation of the capacities of the wireless technologies is used, a slight inaccuracy is possible. Third, because of the oversaturation it is also harder to estimate the actual desired flow rate, as discussed in Section IV-F. At the end of this section we discuss this into more detail. Figure 4b also indicates that the older FBSF fails to provide any improvement in comparison to both baseline and GBSF.

The impact of the implementation of a full virtual MAC layer implementation can clearly be noticed as the actual technology switching of flows can not be noticed in the figure. This is a key difference in comparison with previous work, where temporary drops in throughput could be noticed upon execution of a handover [4, 10]. The execution times for GBSF for the home and office scenario are, respectively, 8.16×10^{-4} s ($\pm2.62\times10^{-5}$) and 1.84×10^{-3} s ($\pm1.96\times10^{-3}$). For FBSF the execution times are higher, respectively, 0.11 s ($\pm3.99 \times 10^{-3}$) and 5.61 s ($\pm$1.79). This clear difference in execution time shows an improvement in performance of the GBSF, compared to the older FBSF. This is discussed into more detail in the next section.

In order to further assess the impact of the unknown desired flow rates, each randomly generated scenario was executed a second time using the actual, rather than the estimated, desired flow rates. For the home scenario nearly no difference can be identified, while a slightly higher throughput can be noticed

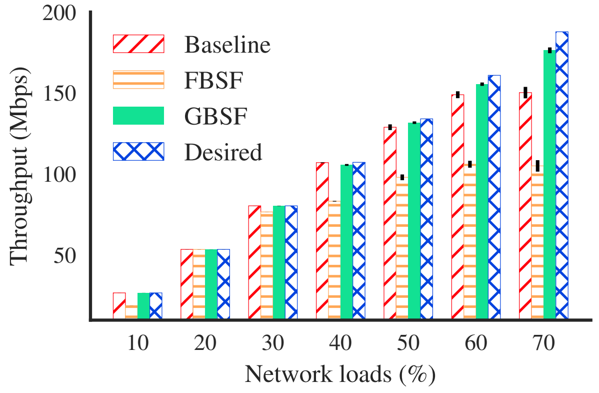


(a) Home scenario

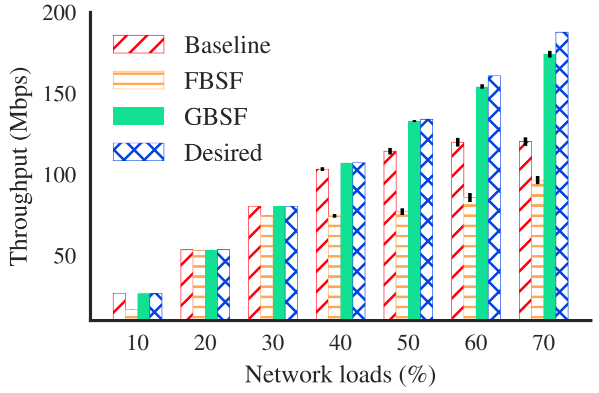


(b) Office scenario

Fig. 4: Throughput as a function of time for different scenarios, comparing our framework to static interface selection baseline



(a) Random types of devices



(b) Only Ethernet-enabled devices

Fig. 5: Throughput as a function of network load, error bars depict the standard error

for the office scenario: for the home scenario the average overall throughput equals to 137.64 Mbps (±3.15). This is only 0.02 Mbps higher than throughput achieved when using the measured rate for each flow. In the case of the office scenario, where the standard GBSF formulation does not reach the desired rate, the impact of using the actual desired rates of the flows is more clear as it results in an average throughput of 331.50 Mbps (±3.03), which is 1.98 Mbps higher than the standard GBSF. This means that knowing the desired flow rates would offer an additional improvement of 0.58 %.

C. Impact of network load and scalability

In order to further investigate the performance of the MILP formulation we conducted experiments with varying networks loads. This is achieved by generating a set of devices, with each a uniform randomly assigned flow with a randomly chosen type and rate. The total desired rate of all flows equals a certain percentage of total theoretical network capacity. Experiments were performed for loads of 10, 20, 30, 40, 50, 60 and 70 % of the theoretical network capacity. Furthermore, two scenarios were evaluated. First, using all types of devices, with every device selected at random from the set of seven devices listed in Table I. For the second scenario, only devices that at least have an Ethernet interface were selected. In other words this scenario does not contain smartphones and tablets. This way more traffic will initially pass over the Ethernet links, making it easier to reach a saturate state on that interface.

Figure 5 illustrates that GBSF offers an increase in throughput under the higher network loads. As the network load increases, so does the relative performance of our proposed framework compared to the baseline, for both scenarios. Furthermore, FBSF again fails to reach the level of the baseline.

For the first scenario, depicted in Figure 5a, from a network load of 50 % onwards, GBSF start having a noticeable impact. The best result is noticed in the case of a network load of 70 %. Here the baseline achieves an average total throughput of 150.33 Mbps (±3.53), while running GBSF leads to an throughput of 176.42 Mbps (±1.16). In other words, there is an increase of 26.09 Mbps or 17.35 %. The achieved rate of GBSF is 11.43 Mbps or 6.09 % below the desired rate. On the other hand, FBSF only achieves 105.64 Mbps (±2.88) or 70 % of the throughput of the baseline.

For the second scenario with only Ethernet-enabled devices, depicted in Figure 5b, GBSF provides an improvement from a network load of 40% onwards. The biggest improvement can again be seen with a network load of 70% : we can notice an

TABLE III: Execution time of the proposed framework (GBSF) compared to our previous work (FBSF)

Load (%)	**Flows**	**FBSF exec. (±SE) [4]**	**GBSF exec. (±SE)**
10	4	0.01 s (±0.01)	1.89 $\times 10^{-4}$ s (±1.92 $\times 10^{-4}$)
20	7	0.04 s (±0.01)	5.06 $\times 10^{-4}$ s (±1.11 $\times 10^{-3}$)
30	11	0.11 s (±0.03)	7.74 $\times 10^{-4}$ s (±1.63 $\times 10^{-3}$)
40	13	0.25 s (±0.14)	9.29 $\times 10^{-4}$ s (±6.84 $\times 10^{-3}$)
50	16	0.43 s (±0.26)	1.17 $\times 10^{-3}$ s (±1.01 $\times 10^{-2}$)
60	20	0.81 s (±0.50)	1.33 $\times 10^{-3}$ s (±8.75 $\times 10^{-3}$)
70	23	1.46 s (±0.71)	1.75 $\times 10^{-3}$ s (±1.33 $\times 10^{-3}$)

TABLE IV: Execution time of the proposed framework (GBSF) compared to FBSF) with increased number of flows

Load (%)	**Flows**	**FBSF exec. (±SE) [4]**	**GBSF exec. (±SE)**
10	7	0.03 s (±0.02)	2.88 $\times 10^{-4}$ s (±1.36 $\times 10^{-4}$)
20	12	0.18 s (±0.06)	6.63 $\times 10^{-4}$ s (±3.90 $\times 10^{-4}$)
30	19	0.80 s (±0.33)	1.06 $\times 10^{-3}$ s (±1.85 $\times 10^{-2}$)
40	24	1.80 s (±0.64)	1.28 $\times 10^{-3}$ s (±5.33 $\times 10^{-2}$)
50	31	4.10 s (±2.16)	1.58 $\times 10^{-3}$ s (±3.44 $\times 10^{-2}$)
60	38	8.68 s (±2.99)	3.07 $\times 10^{-3}$ s (±2.15 $\times 10^{-1}$)
70	44	17.49 s (±8.26)	2.88 $\times 10^{-2}$ s (±7.79 $\times 10^{-1}$)

improvement from 120.3770 Mbps (±2.5707) for the baseline to 174.2481 Mbps (±2.0366) for GBSF. This is an increase of 53.87 Mbps or 44.75 % and is only 13.56 Mbps or 7.22 % below the desired rate. FBSF is consistent in its underperformance as it only achieves a throughput of 91.86 Mbps (±4.64).

Similar to the previously discussed home and office scenarios, we also compared the results of the framework with and without knowing the desired flow rates upfront. A similar impact occurs as when evaluating the home and office scenarios. For the first scenario with all types of devices, the usage of the proposed framework (GBSF) with knowledge of the rates results in a throughput of 179.99 Mbps (±1.16). This is an increase of 3.57 Mbps compared to the normal execution. For the second scenario with only Ethernet-enabled devices, running GBSF with the rates of the flows known, results in a throughput of 177.05 Mbps (±2.17). This is an increase of 2.80 Mbps in respect to the standard execution.

Next, we evaluated the time it takes to execute the MILP and find the optimal configuration. Table III shows the averages of the measured values for the different runs for the first scenario. As a comparison we have added the execution times of the previous version of the formulation [4]. Note that these previous results come from a slightly differentiated setup as the capacity of the Ethernet links was only 10 Mbps, while the desired rates for the flows were on average lower. The results, as depicted in Table III, indicates a noticeable improvement in terms of execution time of GBSF, in comparison with previous FBSF. The difference is the highest for a network load of 70 % where it takes only 1.75 $\times 10^{-3}$ s to optimally solve the MILP formulation. This improvement, as explained in Section IV-B, is due to the change of objective function where the goal is to maximize the throughput across all the collision groups, instead of calculating and maximizing the rate assigned to individual flows. To show this even more, we re-run the first network load scenario (with randomized devices) but we half the lower and upper bounds for the rates of all flows. This way

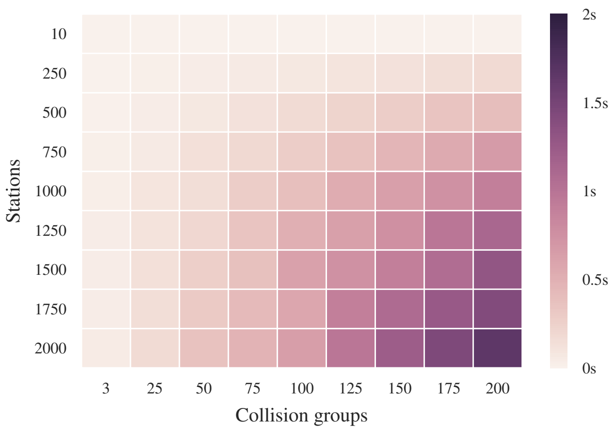


Fig. 6: Scalability of GBSF in terms of stations and collision groups

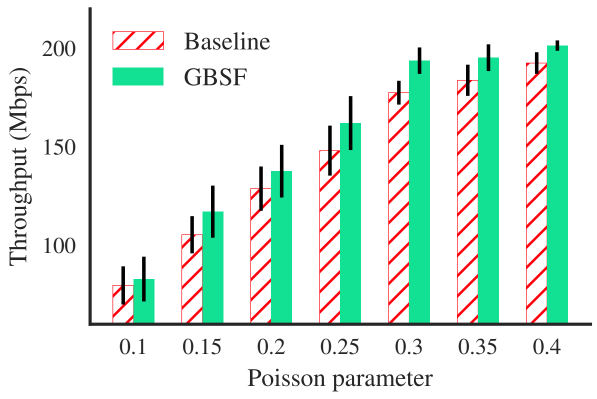
we artificially, approximately, double the number of flows, for each percentage of the network load. The results shown in Table IV more clearly show that increased execution of FBSF in terms of the number of flows. For a network load of 70 % it takes on average 17.47 s (±8.26) to execute FBSF This is in strong contrast with GBSF that only takes 0.03 s (±0.12).

To further investigate the scalability of the proposed solution, we conducted an experiment where we evaluated execution time in terms of number of stations and collision groups (i.e., technologies). Since the scalability of the ns-3 simulator is limited, as each packet is actually generated, we conducted a number of emulations on an Intel NUC. We artificially provide the necessary inputs to the framework, thereby varying the number of stations between 10 and 2000 and the number of collision groups between 3 (as in the simulations) and 200. Furthermore, the same assumptions as stated in Section V-A apply. For each pair of the number of stations and collision groups, we take the average across 20 executions, each with a randomly generated topology. Figure 6 shows the resulting heatmap, where every colored cell indicates the average time to solve the MILP for a specific pair. We can see that the execution time does increase when the number of stations and collision groups rise, but stays under the 2 s for all configurations. In particular, it takes on average 1.67 s (± 0.01) to solve the MILP for the largest configuration.

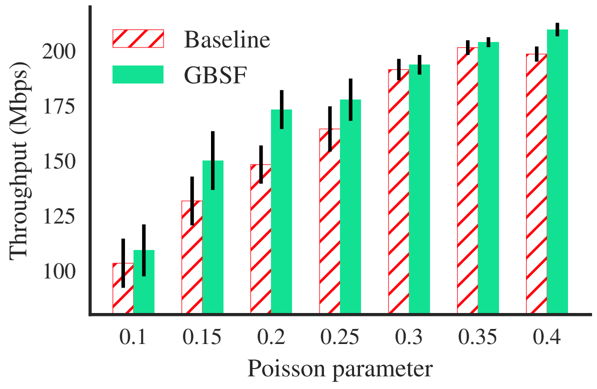
We summarize that the results confirm that the proposed GBSF improves in terms of scalability and performance. The formulation can thus be solved fast enough to react to dynamic network changes, even in large and dense networks.

D. Dynamic scenarios

So far we have only considered scenarios with static flow rates and arrival times, as we assumed the flows to be present throughout the entire simulation. While this helped in determining the performance of the framework, this is not always realistic. Furthermore, an important performance metric of the framework is its adaptability to dynamic conditions. To evaluate this aspect, we consider such a dynamic scenario in this section. All downloads act as in reality and consume as



(a) Short length of flow



(b) Long length of flow

Fig. 7: Throughput as a function of poisson parameters, error bars depict the standard error

much bandwidth as possible (or assigned to them by TCP flow control), until the desired amount of data has been downloaded (or the maximum flow length has been reached). Moreover, flows arrive according to a Poisson distribution and the flow length is uniformly at random chosen between bounds. The impact of different arrival rates is evaluated (Poisson parameter λ) for two scenarios with different flow lengths. They are randomly chosen between 5 and 15 s for the first scenario. For the second one, the bounds are doubled to 10 and 30 s.

Figure 7 shows the results for both scenarios. For all different arrival rates the framework outperforms the baseline significantly. For the scenario with the, on average, smaller flow lengths, depicted in Figure 7a the improvement is quite consistent, with a slightly smaller improvement for parameter values of 0.1 and 0.4. It is also clear that, in both scenarios, for the higher arrival rates, both the baseline and framework reach the physical limits of the network. For the second scenario, with longer flow lengths, the highest improvements by the framework can be noticed for the arrival rates of 0.15 en 0.2. For the arrival rate of 0.15, there is an increase from 131.71 Mbps (±11.09) to 150.10 Mbps (±13.37), or 13.96 %. For the parameter value of 0.25, there is an increase from 148.26 Mbps (±8.70) to 173.30 Mbps (±8.86), or 16.89 %.

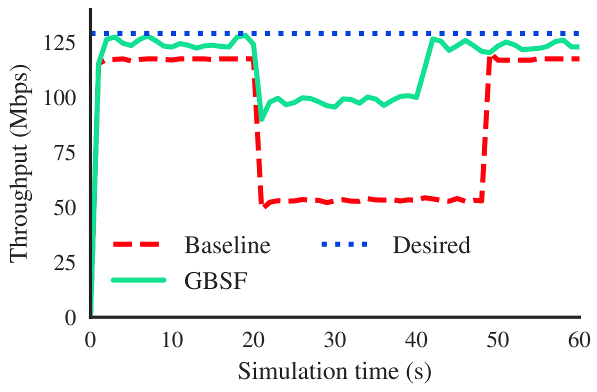


Fig. 8: Throughput as a function of time for a scenario with a link failure

It makes sense that these parameters results in the highest increase, because for the smallest values there is not enough traffic to benefit from the improvements of the framework. On the other hand, for the higher arrival rates with more overall traffic, the network capacity is a limiting factor. Finally, we note that the standard error is quite large, indicating large differences between the different individual experiments.

E. Impact of link failure

A final scenario we investigated is the impact of link failure, as this is a very disruptive and plausible network event. For the topology we use the one from the home scenario, defined in Section V-B but replaced one device (a TV-HD replaces the PC). In this scenario we assume that the Ethernet connection between the switch and gateway fails between the timestamps of 20 s and 40 s. In order to get a representative baseline we did a number of experiments to determine realistic behavior upon link failure. We disconnected the Ethernet connection and measured the time it takes to switch to Wi-Fi on a standard MacBook Air (13-inch, early 2015). Afterwards, when the Ethernet connection was up again, we measured how long it took the operating system to switch back to Ethernet. It turned out that the disconnection is recognized almost immediately (below a second) and no real disruption could be noticed. However, Upon Ethernet reconnection, it took on average at least 8 s before traffic was switched back from Wi-Fi to Ethernet. Our baseline will thus act correspondingly with an immediate handover to Wi-Fi upon link failure and a delayed handover (of 8 s) in the other direction.

The averaged results of this scenario are shown in Figure 8. For the baseline the impact of the link failure can clearly be noticed because of a drop in total throughput of more than 50 %. For the framework this drop is limited to less than 20 %. While the desired rate, before the link failure, is almost reached by the framework, this is not possible anymore after the handover because the capacity of the two Wi-Fi interfaces is simply not large enough. It is also clear that when the Ethernet interface returns, the framework almost immediately reschedules flows across all three interfaces. Thus

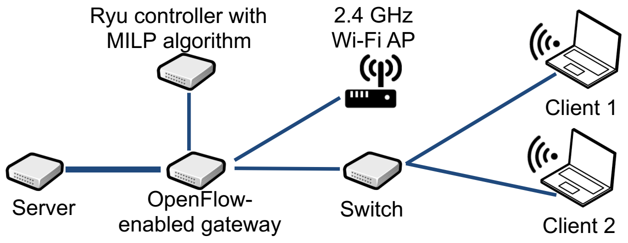


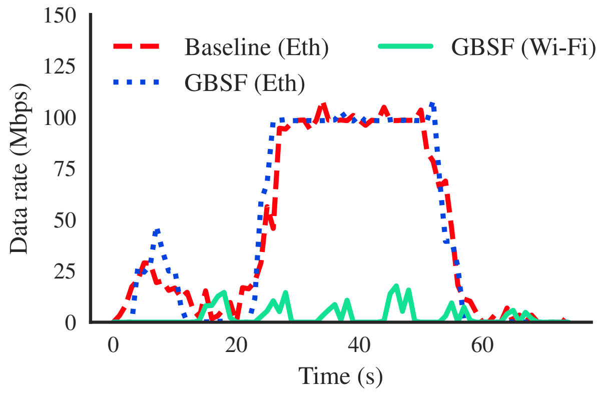
Fig. 9: Architecture of the prototype setup

reaching the same level of throughput as before the link failure. For the baseline, as discussed before, it takes 8 s before traffic is switched back. This behavior results in an average throughput of 82.12 Mbps (±0.60) for the baseline and an average throughput of 113.54 Mbps (±2.36), compared to a desired rate of 129.14 Mbps (±3.38). In other words, the framework does introduces an improvement of 38 %.

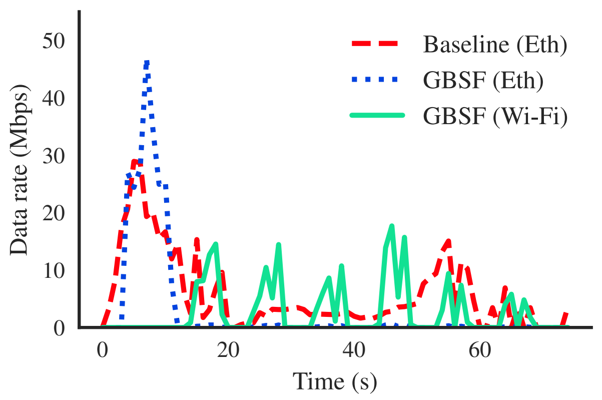
VI. Prototype

We have thus far evaluated the proposed framework under numerous scenarios in a simulation environment. To further investigate the performance of the framework, a real-life prototype was developed. The architecture of this prototype, as shown in Figure 9, consists of two client devices with two interfaces. First, each device has an Ethernet connection to the gateway over a single switch. In other words, the physical link between gateway and switch is shared. Second, both client devices have also an 2.4 GHz Wi-Fi connection. The rest of the architecture consists of a server and an OpenFlow-enabled gateway. On this gateway we have installed OVS, a virtual multilayer switch that is controllable by the OF protocol. The installation of an OVS allows us to perform transparent handovers between the different available technologies, in this case Ethernet and 2.4 GHz Wi-Fi. For the SDN controller we have opted for the Ryu implementation since it is open source and is being actively developed. On this controller we have implemented our framework that decides if and when a flow should be rerouted. For the client devices regular ASUS laptops are used. For the AP we used an off-the-shelf Netgear N750, while all other devices are Intel NUCs. All Ethernet connections are limited to 100 Mbps, except for the connection between the server and gateway that has a capacity of 1 Gbps.

We consider the following scenario: the first client is watching a real-time videostream, while after roughly 20 s the second client starts a download that lasts for approximate 40 s. The framework is compared to a static baseline that assumes that all traffic uses the Ethernet connections, as this is prioritized over the wireless one. For both baseline and framework, five different runs are conducted and results are averaged. Figure 10 indicates that the framework spreads the traffic across the two technologies, as the video traffic is switched to the Wi-Fi route, while the download flow remains on Ethernet. Figure 10a shows the impact for all monitored traffic on both interfaces in terms of data rate, while Figure 10b illustrates the impact on the video flow that is switched. The use of the framework results in an average overall data rate of



(a) All traffic



(b) Video traffic only

Fig. 10: Throughput over Ethernet and Wi-Fi interfaces as a function of time for illustrative scenario on real-life prototype

45.54 Mbps (±5.37), in respect to the baseline of 40.66 Mbps (±4.77). In other words, there is an increase of 4.88 Mbps or 12.01 %. This increase can clearly be seen in Figure 10b, where the video traffic on the Wi-Fi interface continues its burst behavior after the background traffic is introduced. In contrast, in the baseline scenario, the video traffic is not able to reach its desired rates, until the background traffic is removed (around 55 s). Even more important than the improvement in data rate, is the improvement in terms of user experience: in the case of the baseline, the videostream is not able to reach its required bandwidth causing the buffer of the client's video player to drain and thus introducing freezes to the video. This stands in strong contrast to the results of the framework, where at no point in time any impact on the video our its buffer can be noticed, not even when the handover between technologies is executed. Furthermore, we can also report that on average the Gurobi solver needed 0.0003 s ($\pm 1.7407 \times 10^{-5}$), which is in line with the reported execution times for the simulations.

VII. Conclusion

This article presents a transparent flow scheduling and load balancing framework for heterogeneous LANs, which enables to unlock the full potential of heterogeneous networks. The framework jointly manages multiple communication technologies present in the network, such as Ethernet and Wi-Fi, and dynamically reroutes traffic flows across all possible technologies, thereby optimizing network-wide throughput. The framework takes into account the specific nature of wireless networks, where users do not have dedicated network resources but a shared medium instead, and where contention among users may affect network capacity. A thorough evaluation, through a combination of extensive simulations and real-life prototype evaluations, shows that the presented framework can indeed react in real-time to dynamic network changes and offers a significant improvement in terms of throughput. Across different scenarios, an increase in throughput of around 20 % can be noticed compared to a static baseline. Higher increases of up to 40 % can be achieved in some scenarios, for instance, under the presence of link failures. The developed prototype shows the real-life applicability of our solutions and its significant improvement in terms of user experience.

References

[1] IEEE Std. 1905.1-2013, "IEEE standard for convergent digital home network for heterogeneous technologies," 2013.

[2] P. Gallo, K. Kosek-Szott, S. Szott, and I. Tinnirello, "SDN@home: A method for controlling future wireless home networks," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 123–131, 2016.

[3] Y. Yiakoumis, M. Bansal, A. Covington, J. van Reijendam, S. Katti, and N. McKeown, "BeHop: A testbed for dense WiFi networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 3, pp. 71–80, 2015.

[4] T. De Schepper, S. Latré, and J. Famaey, "A transparent load balancing algorithm for heterogeneous local area networks," in *International Symposium on Integrated Network Management (IM)*, 2017.

[5] A. Crabtree, R. Mortier, T. Rodden, and P. Tolmie, "Unremarkable networking: the home network as a part of everyday life," in *the Designing Interactive Systems Conference*, 2012, pp. 554–563.

[6] J.-P. Javaudin, M. Bellec, D. Varoutas, and V. Suraci, "OMEGA ICT project: Towards convergent Gigabit home networks," in *19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2008.

[7] T. Meyer, P. Langendörfer, M. Bahr, V. Suraci, S. Nowak, and R. Jennen, "An inter-mac architecture for heterogeneous gigabit home networks," in *20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2009.

[8] D. Macone, G. Oddi, A. Palo, and V. Suraci, "A dynamic load balancing algorithm for quality of service and mobility management in next generation home networks," *Telecommunication Systems*, vol. 53, no. 3, pp. 265–283, 2013.

[9] K. Xu, X. Wang, W. Wei, H. Song, and B. Mao, "Toward software defined smart home," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 116–122, 2016.

[10] N. Soetens, J. Famaey, M. Verstappen, and S. Latré, "SDN-based management of heterogeneous home networks," in *11th International Conference on Network and Service Management (CNSM)*, 2015, pp. 402–405.

[11] T. De Schepper, P. Bosch, E. Zeljkovic, K. De Schepper, C. Hawinkel, S. Latré, and J. Famaey, "Sdn-based transparent flow scheduling for heterogeneous wireless lans," 2017.

[12] S. Sahaly and P. Christin, "Inter-MAC forwarding and load balancing per flow," in *20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2009, pp. 1–4.

[13] G. Oddi, A. Pietrabissa, F. D. Priscoli, and V. Suraci, "A decentralized load balancing algorithm for heterogeneous wireless access networks," in *World Telecommunications Congress*, 2014, pp. 1–6.

[14] O. Olvera-Irigoyen, A. Kortebi, and L. Toutain, "Available bandwidth probing for path selection in heterogeneous home networks," in *IEEE Globecom Workshops (GC Wkshps)*, 2012, pp. 492–497.

[15] O. Bouchet, A. Kortebi, and M. Boucher, "Inter-MAC green path selection for heterogeneous networks," in *IEEE Globecom Workshops (GC Wkshps)*, 2012, pp. 487–491.

[16] A. Kortebi and O. Bouchet, "Performance evaluation of inter-mac green path selection protocol," in *12th Annual IEEE Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, 2013, pp. 42–48.

[17] M. Zekri, B. Jouaber, and D. Zeghlache, "A review on mobility management and vertical handover solutions over heterogeneous wireless networks," *Computer Communications*, vol. 35, no. 17, pp. 2055–2068, 2012.

[18] G. Gódor, Z. Jakó, Á. Knapp, and S. Imre, "A survey of handover management in lte-based multi-tier femtocell networks: Requirements, challenges and solutions," *Computer Networks*, vol. 76, pp. 17–41, 2015.

[19] J. G. Andrews, S. Singh, Q. Ye, X. Lin, and H. S. Dhillon, "An overview of load balancing in hetnets: Old myths and open problems," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 18–25, 2014.

[20] B. Ng, A. Deng, Y. Qu, and W. K. Seah, "Changeover prediction model for improving handover support in campus area wlan," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 265–272.

[21] S. Fernandes and A. Karmouch, "Vertical mobility management architectures in wireless networks: A comprehensive survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 45–63, 2012.

[22] A. T. N. TN2224, "Best practices for creating and deploying http live streaming media for the iphone and ipad," 2012.

[23] D. Lee, B. E. Carpenter, and N. Brownlee, "Media streaming observations: Trends in udp to tcp ratio," *International Journal on Advances in Systems and Measurements*, vol. 3, no. 3-4, 2010.