A transparent load balancing algorithm for heterogeneous local area networks

Tom De Schepper, Steven Latré and Jeroen Famaey
Department of Mathematics and Computer Science, University of Antwerp - imec, Belgium

***Abstract*—Today's local area networks (LANs) consist of a plethora of heterogeneous consumer devices, equipped with the ability to connect to the Internet using a variety of different network technologies (e.g., Ethernet, Power-Line, 2.4 and 5GHz Wi-Fi). Nevertheless, devices generally opt to statically connect using a single technology, based on predefined priorities. This static behaviour does not allow the network to unlock its full potential, which becomes increasingly more important as the requirements of services, in terms of latency and throughput, grow. To address this issue, we present a load balancing algorithm that dynamically selects a suitable interface and path through the network for each flow, based on service requirements, bandwidth availability and current link quality. The goal of the algorithm is to find an optimal path configuration for all the flows across the network that maximizes the global throughput. It dynamically adapts to changing network conditions, the arrival and departure of flows and link failures. The problem is formulated as a Mixed Integer Linear Program (MILP) and its solution, as well as that of a faster heuristic algorithm, is extensively tested in a series of ns-3 simulations with different network topologies and flow configurations. We differentiate from existing work by estimating flow rates and dynamic network conditions using real-time monitoring information, taking into account the shared medium of wireless networks and the specific fairness behaviour of TCP. Results show an increase in throughput up to 70 % in heterogeneous LANs under dynamic network conditions.**

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

I. Introduction

In recent years, Local Area Networks (LANs) have evolved significantly. They are populated with an ever-growing set of diverse devices (e.g., computers, laptops, smart TVs, tablets, smartphones), connected to the Internet using a variety of wired and wireless networking technologies (e.g., Ethernet, power-line communications, different Wi-Fi standards) and consuming services with increasingly stringent requirements (e.g., Voice-over-IP, Video on Demand, IP-TV). 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, LANs are generally managed in a mostly static manner, unable to automatically react in a timely fashion to temporary disruptions that cause Quality of Experience (QoE) degradations.

Many modern consumer devices are equipped with multiple network interfaces (e.g., tablets and smartphones with 2.4 GHz and 5 GHz Wi-Fi, or laptops and smart TVs with Wi-Fi and Ethernet). Currently, such devices generally statically select one of the available technologies 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 technologies or using multiple ones simultaneously. Nevertheless, making network interface selection automatic and dynamic would enable optimizations such as multipath routing, load balancing and dynamic path reconfiguration, aiding to unlock the network's full potential. The efficiency of such a dynamic approach will only increase as more and more technologies are introduced to home and other LAN, such as sub-1 GHz Wi-Fi (i.e., 802.11ah), 60 GHz Wi-Fi (i.e., 802.11ad) and visible light communications (e.g., Li-Fi).

The IEEE 1905.1 standard has been defined precisely with this purpose in mind. It introduces an abstract data link layer (i.e., a hybrid or inter-MAC) that enables the transparent use of, and switching between, several wired and wireless network technologies.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. The standard provides the protocols and interfaces to direct flows to specific network interfaces based on data link layer packet header matching rules. A local or remote controller can dynamically adapt the rules to achieve dynamic flow redirection based on application requirements or network conditions. Since this standard is still in active development, and no products exist yet that support it, alternatives have been proposed to achieve the same effect, such as flow redirection based on software defined networking (SDN) [1], [2].

Although IEEE 1905.1 specifies the features to enable dynamic flow redirection, it does not define the algorithms for selecting suitable paths per flow. Therefore, this paper proposes a transparent load balancing and routing algorithm for LANs to fill this void. The algorithm 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. In line with the standard, path selection is performed at the granularity of data link layer flows. The algorithm aims to find a global optimal scheduling configuration for all the flows in the network, in order to achieve maximum global throughput. The proposed algorithm 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

978-3-901882-89-0 @2017 IFIP

medium instead. Third, existing algorithms directly assign rates to flows, ignoring TCP behaviour. Instead, our algorithm estimates the rate that TCP would assign to a flow based on its chosen path as well as that of other flows, and uses that information to find the optimal path for each flow within the rules imposed by TCP.

The contributions of this paper are threefold. First, we model the load balancing problem in heterogeneous LAN as a mixed integer linear program (MILP), which can be solved using linear programming solver (e.g., Gurobi). Second, we propose a first heuristic to provide a sub-optimal solution in a more feasible time. Third, we evaluate the resulting algorithm and a heuristic in a variety of scenarios, using different flow configurations, based on ns-3 simulation results. The evaluation aims to demonstrate the effectiveness of dynamic flow scheduling in LANs and compares this approach to the traditionally employed static solution.

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 present our algorithm in Section III. Section IV discusses the simulation results. Finally, conclusions and future research directions are provided in Section V.

II. Related Work

Today's LANs consist of a multitude of heterogeneous communication technologies (e.g., Ethernet, Power line, Wi-Fi) used by consumer devices (e.g., laptop, Smart TV, tablet or smartphone) to connect to the Internet. 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 network technology. Since more and more devices have the ability to use multiple network technologies interchangeably, the interest in this research topic is steadily increasing [3]. Optimizing the LAN to fully utilize and exploit these heterogeneous technologies can significantly increase overall network performance.

One of the main focal points so far has been the development of a unified high bandwidth environment. First, a convergent gigabit home network using heterogeneous transmission technologies was proposed [4]. This was followed by an Inter-MAC architecture for home network devices [5]. This architecture introduced an abstract or hybrid MAC layer on top of the current data link layer (OSI layer 2) to combine all the heterogeneous MAC interfaces in a transparent manner [6]. The defined abstract data link layer (i.e., a hybrid MAC) enables the transparent use of switching between several wired network technologies. For this purpose, all devices on the network have 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. 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. This standardization of the transparent usage of multiple technologies can increase the capacity of the network by load balancing different traffic flows over different paths. It also provides resilience against link degradation or failure by rerouting flows. While the standard provides the protocols and interfaces to direct flows to specific network interfaces, it still lacks the algorithms to exploit these features. Some load balancing algorithms have already been proposed, which will be discussed in the remainder of this section. Research on load balancing in other domains is also mentioned.

Sahaly and Christin define a framework for heterogeneous home networks that includes a per-flow decentralized load balancing technique [7]. 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 real-life results. Macone et al. proposed a per-packet load balancing algorithm [6]. 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 the in-order arrival guarantees of TCP. This results in severe throughput fluctuations in real-life systems when combined with standard TCP protocols. 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. [8]. 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, Olevera-Irigoyen has shown that determining the actual available bandwidth on the links, has a big impact on the results of distributing the flows [9].

Recent load balancing solutions focus heavily on energy optimization. Bouchet et al. proposed such an algorithm that aims to reduce energy consumption and use the most energy efficient link while still providing a good Quality of Service (QoS) [10], [11]. This is done by assuming the energy consumption model is known in advance, and not by real-time measurements on the devices. Similar work on energy efficient flow scheduling can be found within the field of data center management [12], [13]. Data center topologies typically consist of multi-rooted trees with many equal-cost paths between any given pair of hosts. Often per-flow static hashing is used by IP multipath protocols in these data center networks, which can cause long-term collisions. To cope with the resulting substantial bandwidth losses is a key challenge for load balancing in cloud computing environments [14], [15]. Other flow scheduling algorithms for data center networks have, for instance, as a goal the minimization of the flow completion time [16]. The proposed algorithm implements a Multiple Level Feedback Queue, in which a flow is gradually demoted from higher-priority to lower-priority queues based on bytes it has sent during its lifetime. Such queue scheduling algorithms are also found in home networks. Here, the focus lies on application-aware resource scheduling at the gateway (e.g., a YouTube video and a file download) [17], [18], [19].

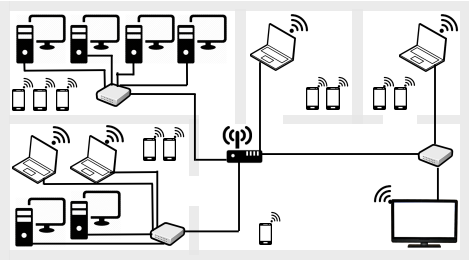


Fig. 1: Example heterogeneous LAN office scenario

This work is not directly related to load balancing of flows, but can be seen as complementary research. The same can be said about multipath TCP (MPTCP) [20], [21]. This TCP extension enables the transmission and reception of data concurrently on multiple interfaces. While MPTCP has similar goals as our presented work, mainly aiming to improve QoS and network resource utilization, it focuses on the alternative paths between two hosts and not on a network-wide scale.

In summary, 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 behaviour of TCP are also ignored. Similarly, existing work in the area of data center networks cannot be directly applied to LAN, as it assumes a tree topology network consisting of wired links and generally employs non-standard multipath TCP protocols. 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, it takes into account standard TCP protocol throughput behaviour, rather than assuming flow rates can be decided by the load balancing algorithm itself. Finally, the specific characteristics of wireless networks, such as mutual interference among senders and propagation loss based on distance, are also considered explicitly.

III. Heterogeneous load balancing algorithm

This section presents the proposed heterogeneous load balancing model and algorithm. In order to clarify the concept of heterogeneous LANs, an example scenario is depicted in Figure 1. The figure shows a variety of devices, each connected to the residential gateway via one or multiple interfaces. Connectivity can be either wired (e.g., Ethernet) or wireless (e.g., Wi-Fi).

A. Network Model

The architecture of the LAN is modeled as a multi-graph defined as a quadruple (N,T,L,G) where:

- N is the set of nodes $\{n_1, n_2, ..., n_n\}$. These nodes represent the different devices within the LAN.
- T is the set of technologies $\{t_1, t_2, ..., t_t\}$ and $c_t \in \mathbb{R}_{\geq 0}$ represents the the bandwidth capacity of $t \in T$.
- L is the set of links. A link is characterized by a triple $< s_l, d_l, t_l >$ with $s_l \in N$ the source node, $d_l \in N$ the destination node and $t_l \in T$ the technology.
- G is the set of all collision groups. A collision group $g \in G$ is defined as: $\{l_1, l_2, ..., l_i \in L \mid t_{l_1} = t_{l_2} = ... = t_{l_i} \wedge c_{t_{l_1}} = c_{t_{l_2}} = ... = c_{t_{l_i}}\}$. In other words, a collision group encapsulates all the links that share the capacity of a technology and where the transmissions can interfere with each other. For instance, if two devices (or nodes) are connected by Wi-Fi with an access point (AP), the links between the devices and the AP are in the same collision group, as their bandwidth capacity is shared.

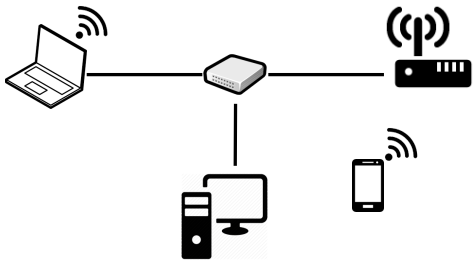
An example of a simple LAN topology is show in Figure 2a. This network consists of a gateway (that also serves as a Wi-Fi AP), a switch and two devices, a laptop and a (fixed) computer that are connected via the switch to the gateway. Furthermore, the laptop also has Wi-Fi connectivity. This network can be represented by means of the previously described model, as is shown in Figure 2b. Each of the devices is represented by a node $\{n_0, n_1, n_2, n_3, n_4\}$. There are also two different technologies $\{t_0, t_1\}$, representing the Ethernet cables (assuming that all cables have the same capacity) and the Wi-Fi connection. 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. For each of the connections between the nodes, there are two links present in the model. One link per direction. For instance, the Ethernet connection between the gateway and the switch is described by the links $< n_0, n_l, t_0 >$ and $< n_1, n_0, t_0 >$. In this example there are seven collision groups: there is one collision group that consists of the four links corresponding with the Wi-Fi connection $\{< n_0, n_3, t_1 >, < n_3, n_0, t_1 >, < n_0, n_4, t_1 >$ and $< n_4, n_0, t_1 >\}$. All other links have their own collisions groups, due to the fact that Ethernet is full-duplex so there are no collisions between the traffic in the two directions over the wired connection.

In addition to the network topology, traffic flows going through the network also need to be modelled. Let us define F as the set of all flows. A flow is a triple $< s_f, d_f, r_f >$ with $s_f \in N$ the source node, $d_f \in N$ the destination node and $r_f \in \mathbb{R}_{\geq 0}$ the desired rate of the flow. In our example, a flow of 1 Mbps from the gateway to the laptop will be represented as the triple $< n_0, n_3, 1 >$.

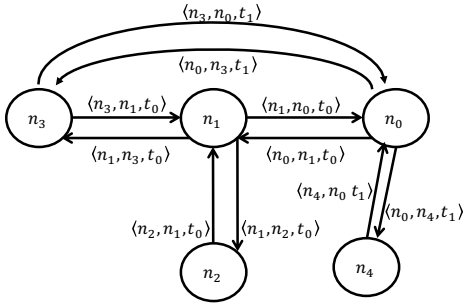
B. MILP formulation

The LAN flow scheduling problem considered in this paper is modelled 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 an optimal solution to the problem.

The inputs consist of the previously described network and flow model, as well as the following sets:



(a) Network topology



(b) Multi-graph representation

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

- S^{L_n} : $\{\forall l \in L \mid s_l = n\}$; all the links that have node $n \in N$ as source.
- D^{L_n} : $\{\forall l \in L \mid d_l = n\}$; all the links that have node $n \in N$ as destination.
- S^{F_n} : $\{\forall l \in L \mid s_l = s_f\}$; all the links that have the same source as flow $f \in F$.
- D^{F_n} : $\{\forall l \in L \mid d_l = d_f\}$; all the links that have the same destination as flow $f \in F$.
- X^{F_n} : $\{\forall l \in L \mid d_l = s_f\}$; all the links that have as destination the source of flow $f \in F$.
- Y^{F_n} : $\{\forall l \in L \mid s_l = d_f\}$; all the links that have as source the destination of flow $f \in F$.

We define the following decision variables:

- $\lambda_{l,f} \in \{0, 1\}$; this variable represents the path for every flow. If a flow $f \in F$ passes over a link $l \in L$ then $\lambda_{l,f} = 1$, otherwise it equals 0.
- $\tau_f \in [0, r_f]$; this variable defines the assigned rate (bandwidth) of a flow $f \in F$.

As an objective function, the model maximizes the total assigned rate (bandwidth) over all flows:

- $max \sum_{f \in F} \tau_f$

Finally, we define the following constrains:

- The capacity constraint makes sure that for each collision group, the total capacity of the technology is not exceeded: $\forall g \in G : \sum_{l \in g} \sum_{f \in F} \lambda_{l,f} \cdot \tau_f \leqslant c_{t_g}$
- The flow conservation constraints make sure that the links assigned to a flow form a loopless path:
  - Every path has exactly 1 link departing from the start node of the flow: $\forall f \in F : \sum_{l \in S^{F_n}} \lambda_{l,f} = 1$
  - Every path has exactly 1 link arriving at the destination node of the flow: $\forall f \in F : \sum_{l \in D^{F_n}} \lambda_{l,f} = 1$
  - Every path cannot have a link arriving at the start node of the flow: $\forall f \in F : \sum_{l \in X^{F_n}} \lambda_{l,f} = 0$
  - Every path cannot have a link departing from the destination node of the flow: $\forall f \in F : \sum_{l \in Y^{F_n}} \lambda_{l,f} = 0$
  - For every node on the path of the flow, except the start and destination, there can be at most one link departing from that node: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in S^{L_n}} \lambda_{l,f} \leqslant 1$
  - For every node on the path of the flow, except the start and destination, there can be at most one link arriving in every node of the path: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in D^{L_n}} \lambda_{l,f} \leqslant 1$
  - The total number of links departing from each node should be equal to the total number of links arriving at that node, except start and destination: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in S^{L_n}} \lambda_{l,f} = \sum_{l \in D^{L_n}} \lambda_{l,f}$
  - Of all the links, in both directions, between two nodes $n, m \in N$, there can be at most 1 part of each path: $\forall f \in F, \forall n, m \in N : \sum_{l \in S^{L_n} \cap S^{L_m}} \lambda_{l,f} \cdot \sum_{l \in S^{L_m} \cup S^{L_n}} \lambda_{l,f} = 0$

C. TCP fairness

Existing models and algorithms for load balancing in LANs assume the rate τ_f of each flow can be chosen by the algorithm. Although this offers an extra degree of freedom, and therefore often results in a better solution in terms of total throughput, it is unrealistic. In reality, the transport protocol (e.g., TCP) determines the rate of each flow, based on the chosen path and other coexisting flows. Since Internet traffic is hugely dominated by TCP, the model assumes all flows follow the TCP fairness rules. As such, a constraint is added that approximates TCP fairness behaviour.

Let O_t be the practical capacity correction factor, which represents the achievable throughput on a link as a fraction of its theoretical maximum throughput. Its value depends on the transmission technology used by the link, as well as protocol overhead and can be experimentally determined. The TCP fairness constraint is subsequently defined as follows:

$$\forall l \in L, \forall f \in F : \tau_f \cdot \lambda_{l,f} \cdot \sum_{g \in F} \lambda_{l,g} \leqslant O_t \cdot c_{t_l}$$

This constraint, however, is not linear as it consists of a multiplication of three decision variables. Nevertheless, it can be transformed into another set of constraints that can be solved by standard linear programming algorithms. First, a new decision variable needs to be defined:

- $\forall l \in L, \forall f, g \in F : \zeta_{l,f,g} = \lambda_{l,f} \cdot \lambda_{l,g}$

The original fairness constraint can then be transformed as follows:

- $\forall l \in L, \forall f, g \in F : \zeta_{l,f,g} \leqslant \lambda_{l,f}$
- $\forall l \in L, \forall f, g \in F : \zeta_{l,f,g} \leqslant \lambda_{l,g}$
- $\forall l \in L, \forall f, g \in F : \zeta_{l,f,g} \geqslant \lambda_{l,f} + \lambda_{l,g} - 1$
- $\forall l \in L, \forall f \in F : \tau_f \cdot \sum_{g \in F} \zeta_{l,f,g} \leqslant O_t \cdot c_{t_l}$

D. Estimating flow and network parameters

One of the novelties presented in this paper is the usage 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 algorithm. To enable this, a monitoring framework is implemented, based on the monitoring specifications of IEEE 1905.1 [22]. The IEEE 1905.1 standard defines link metric reporting and querying options based on the Link Layer Discovery Protocol (LLDP), which allows devices in a network to ask information regarding link quality to their neighbours. 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. Moreover, the flow fairness constraint requires the practical link capacity correction function as input. This function estimates how much of the theoretical capacity of a link can actually be achieved for the transmission of data. Management traffic, ACKs and overhead should thus be excluded. We have experimentally determined this factor and the results are discussed in the next section. 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 IEEE 1905 link metric reporting.

Because of the fact that we use an estimation (the measured MAC layer throughput) 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 algorithm 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 IV compares performance of the algorithm both when estimating and knowing the desired rate of flows.

E. Heuristic

Optimally solving the MILP model scales exponentially in terms of the number of devices and flow in the network. As such, heuristics solutions are needed in larger scenarios. We defined a heuristic solution where the objective function has been removed and was added as a fixed constraint to the model. The fixed constraint is formulated in the following way:

$$\forall f \in F : \tau_f = r_f$$

This allows a feasible solution to be found much faster.

IV. Results and discussion

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

A. Evaluation setup

The Gurobi Optimizer was used to solve the MILP formulation. We assume three technologies: Ethernet, 5 GHz Wi-Fi, and 2.4 GHz Wi-Fi. The Ethernet network is built out of full-duplex Ethernet cables with a theoretical throughput of 10 Mbps. The choice for 10 Mbps was made to limit the simulation time, since ns-3 execution time increases with throughput, and this limit allows a saturated state to be reached at lower total throughput. 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.

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 does not change over time and is chosen uniformly at random between an upper and lower bound, based on the involved device. We assign one flow per device and as such do not assume the concurrent usage of both Wi-Fi interfaces, as this is not supported by current hardware. The desired flow rate per flow and device type is also depicted in Table I. The flow rates were selected based on representative figures from literature [23] of existing applications in these three categories. 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 [24].

For every described scenario, results are averaged over different randomly generated flow and topology configurations. To this extend, each experiment was repeated 20 times. We will also report the standard error for each experiment. Moreover, every experiment has a simulation time of 60 seconds and the flow duration times are equal to the simulation time. 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

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	×			1–4 Mbps	1–2.4 Mbps	1–2 Mbps
Laptop	×	×	×	1–4 Mbps	1–2.4 Mbps	1–2 Mbps
HD Television	×	×	×	1–4 Mbps	2.5–8.5 Mbps	1–2 Mbps
4K Television	×	×	×	1–4 Mbps	12–19 Mbps	1–2 Mbps
Tablet		×	×	1–4 Mbps	0.6–4.5 Mbps	1–2 Mbps
Smartphone modern		×	×	1–4 Mbps	0.6–1.2 Mbps	1–2 Mbps
Smartphone old			×	1–4 Mbps	0.6–1.2 Mbps	1–2 Mbps

sent every second and the algorithm only takes into account the most recently received information. Our future work includes investigating the impact of including older (but still valid) monitoring information. Finally, unless noted otherwise, the algorithm is executed every 10 seconds in the simulation, starting from 15 seconds.

B. Estimation of practical correction factor

The constraint proposed in Section III-C requires the practical capacity correction function as input. This function estimates how much of the theoretical capacity of a links can actually be achieved for the transmission of data. Management traffic, ACKs and overhead should thus be excluded. We experimentally determined an average value for this factor, given the network settings outlined above, by running multiple experiments per technology over a link between two nodes. For Ethernet, 20 experiments were executed, where the desired rate and the number of flows were gradually increased. We started from one flow up to four and varied their rates between 2 Mbps and 12 Mbps. As such, both uncongested and saturated scenarios were evaluated. For the two different Wi-Fi technologies, the experiment was repeated 30 times per technology. Due to the higher theoretical rates (150 Mbps and 72.2 Mbps), the desired rates were increased by a factor of 10. In these experiments, not only flow count and rates were varied but also different station counts between 1 and 10 were evaluated. The obtained correction factors are 0.781, 0.795 and 0.718 for Ethernet, 2.4 GHz and 5 GHz Wi-Fi respectively.

C. Home and office scenarios

To demonstrate the impact of our algorithm in LANs, two basic scenarios were created to reflect two typical LAN environments: a home and an office network. For each setup, the exact number of devices per type and the possible flows are depicted in Table II. The results for the home and office scenario are respectively shown in Figure 3. The graphs compare the static baseline scenario and the proposed load balancing algorithm to the total sum of desired flow rates. Both scenarios show a strong improvement in the total throughput over the network. Moreover, the proposed algorithm achieves the desired rate of all flows in both scenarios. For the home scenario, throughput increases by 11.30 Mbps ($\pm$0.3460) or 59.86 % compared to the static baseline, even after executing the algorithm once. For the office scenario there is an increase of 13.73 Mbps ($\pm$0.5556) or 26.48 %.

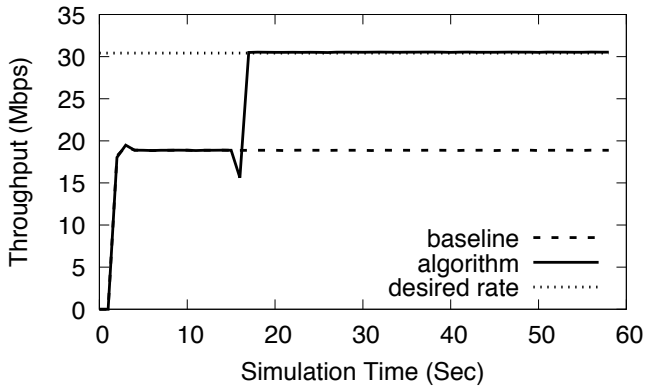
TABLE II: Topology parameters of the two scenarios

Device	Home	Office	Flows
Switches	1	2	N/A
Desktop PC	1	6	Download
Laptop	1	4	Download / Video conference
HD TV	2	0	Video stream
4k TV	1	1	Video stream
Tablet	2	2	All types of flows
Smartphone modern	2	6	All types of flows
Smartphone old	1	4	All types of flows
Total devices	11	25	

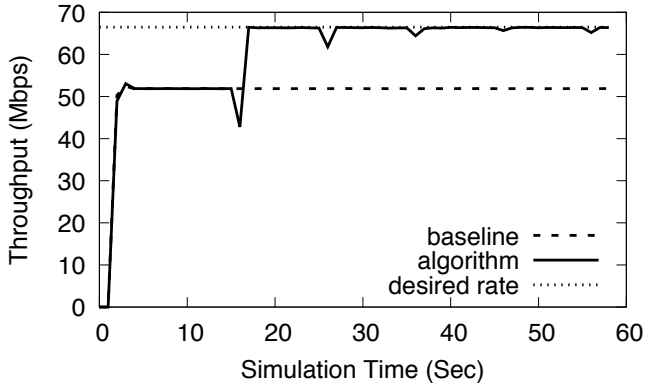
Taking a closer look at the results shows a temporarily decrease in throughput every time the algorithm is executed in the office scenario. This decrease is caused by flow path changes, which is documented in our previous work [1]. In the home scenario, there is only a decrease after the first execution of the algorithm, meaning that the most optimal solution has been found immediately. However, this is not the case for the more complex office setup, where a small decrease can be noticed after every execution of the algorithm. Nonetheless, the decrease becomes smaller over time, as the algorithm converges to the optimum. This can be explained by the fact that the algorithm only knows the measured rate of each flow. If multiple flows are going over the same link, they might not achieve their actual desired rate. However, when the flow paths change after the run of the algorithm, the flows are scheduled in a more optimal way and more flows will thus achieve their desired rate. In the next execution of the algorithm, these higher and more accurate rates are taken into account.

In order to further assess impact of the unknown the desired flow rate, each randomly generated scenario was executed a second time using the actual, rather than the estimated, desired flow rates. For the home scenario there was no difference at all, for every flow the same path was chosen. For the office scenario, there are however some small differences to be noted. The first time the algorithm is executed (i.e., at 15 s), an average of 6 out of 57 flows receive a different path when knowing the real desired rates. The standard error is 0.7557. When the algorithm is executed again (i.e., at 25 s), the difference is only 3 flows with a standard error of 0.6262, showing that the algorithm converges to the real desired rates. Overall, there is on average a slightly higher throughput of 0.2079 Mbps ($\pm$0.2342) when the algorithm knows the actual desired rates.

Finally, the accuracy of the MILP model, in terms of



(a) Home scenario



(b) Office scenario

Fig. 3: Throughput as a function of time for different scenarios, comparing our algorithm to default static interface selection

TABLE III: Average fair TCP flow rate calculation error (± standard error) over subsequent executions of the algorithm

Algorithm execution	Home scenario	Office scenario
1	1.2669 % (±4.8882)	0.6396 % (±3.2897)
2	0.7507 % (±1.8062)	0.3244 % (±1.2563)
3	0.7798 % (±1.8724)	0.3830 % (±1.6283)
4	0.7754 % (±1.8623)	0.3411 % (±1.2996)
5	0.0233 % (±0.0701)	0.0284 % (±0.1746)

estimating fair TCP flow rates, was assessed, as the fairness constraint of the model is a linear approximation of real TCP behaviour. Table III depicts, for subsequent executions of the algorithm, the average difference between the flow rate measured after executing the algorithm and the rate calculated by the MILP model. The table shows a clear decrease in calculation error and standard error, resulting in a negligible error after 5 executions.

D. Impact of network load

The previous section showed a difference in performance between the home and office scenario, which can be attributed to the load and network topology. To understand the full impact of the load, in terms of total desired rate, we conducted an experiment with varying network loads. This is achieved by

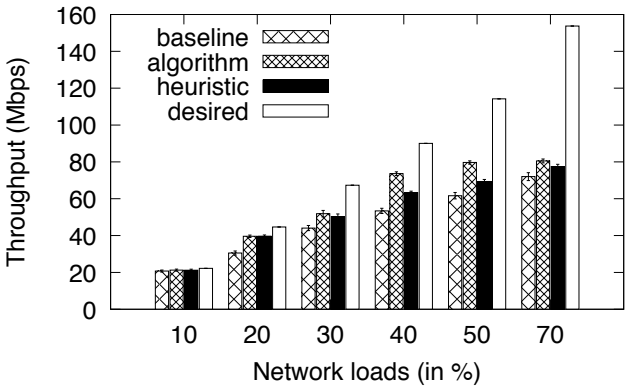


Fig. 4: Throughput as a function of network load, error bars depict the standard error

generating a random number of devices, each with one flow with uniformly at random selected device and flow type as well as desired rate (cf. Table I). The total desired rate of all flows was made sure to be equal to 10, 20, 30, 40, 50 or 70 % of the theoretical network capacity. As not all devices can use all network technologies, the real capacity of the network is obviously lower.

Figure 4 illustrates that the proposed algorithm offers a significant increase in throughput under all network loads. As the network load increases, so does the relative performance of our proposed algorithm compared to the baseline. From 30 % network load onward, neither approach achieves the desired rate. This is because the bottleneck caused by the Ethernet connection. Second, collisions in the wireless networks increased due to interference. If we look at the performance of the heuristic solution, we see that it consistently outperforms the baseline. However, in comparison with the algorithm, it falls of starting from 30 %. This is explained by the usage of non-optimal flow scheduling solutions.

Furthermore, the execution time of the MILP algorithm was also measured. These experiments were conducted using a single core of an Intel® Xeon® E5-2680 Processor running at 2.8 GHz and with 8 GB RAM memory. Due to the exponential time complexity for solving MILP formulations, there is a strong increase in execution time with increasing network loads, as depicted in Table IV. However, even in a network with on average 42 flows, which should cover any home network setting, the average execution time is low enough to run the algorithm every few seconds. The heuristic clearly needs less computational time than the MILP, but a similar increase in execution time can be noticed. We can conclude from these results, however, that in very large networks both the optimal algorithm and the heuristic cannot calculate a solution fast enough to immediately react to changes in network conditions. We aim to address this issue in two ways in our future work: (i) a distributed interface selection component running on the clients for fast adaptation in case of unexpected congestion, and (ii) further research into a heuristic that find a near-optimal solution for the centralized MILP in linear time.

TABLE IV: Execution time of MILP and heuristic

Load (%)	Flows (±SE)	MILP exec. (±SE)	Heur. exec. (±SE)
10	14 (±0.5916)	0.2367 s (±0.0597)	0.2161 s (±0.0537)
20	27 (±0.6649)	2.6650 s (±0.5412)	2.2907 s (±0.5412)
30	42 (±1.1562)	4.7969 s (±1.5207)	3.8202 s (±1.4314)
40	55 (±1.3465)	27.7596 s (±5.3880)	24.7209 s (±4.7007)
50	69 (±1.5560)	76.7836 s (±16.3218)	63.5008 s (±12.2703)
70	92 (±2.0976)	226.3498 s (±57.8981)	205.0288 s (±53.5296)

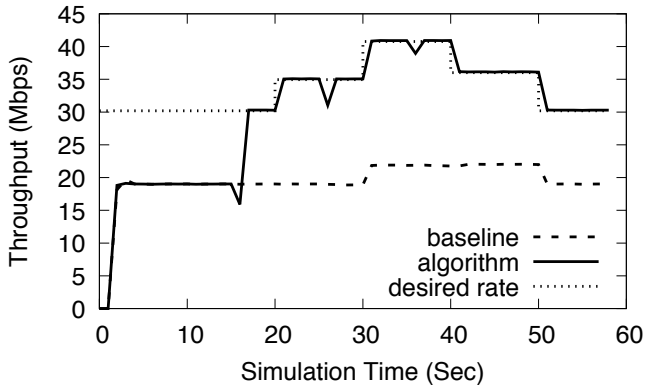


Fig. 5: Dynamic scenario where new devices join the network at time 20 s and 30 s, and leave at time 40 s and 50 s

E. Dynamic scenario

The previous scenarios considered a static number of flows. However, an important performance metric of the algorithm is its adaptability to dynamic conditions. In this section, we consider such a dynamic scenario, where new flows are gradually added to and then removed from the network. We started from the topology of the home scenario defined in Section IV-C and added four additional devices that join and also leave the network during the simulation: at simulation time 20 s a second laptop and third HD television join the network and will stay connected for 20 s. At simulation time 30 s another laptop and a modern smartphone become active until simulation time 50 s. The algorithm is again executed first at simulation time 15 s and every 10 s thereafter. In the results, as depicted in Figure 5, the load balancing algorithm's impact is clearly noticeable, as all newly arriving flows immediately receive their desired rates. Even if new flows use their static default prioritized interface when they initially arrive, the load balancing algorithm causes capacity to be available on all three technologies. As such, new flows already receive the capacity they require, even before the algorithm recalculates the optimal flow schedule. This stands in strong contrast with the baseline: as the two laptops and HD TV will by default use the already saturated Ethernet connection, no increase in throughput can be noticed. Only in the interval 30 to 50 s, an increase in traffic is visible, due to the fact that the smartphone will connect to the unsaturated 5 GHz Wi-Fi network. On average, the load balancing algorithm brings an increase of 14.4048 Mbps (±6.029E-7) or 70.77 % in the evaluated scenario.

As a final experiment, we assess the impact of different algorithm execution intervals. Above, a 10 s interval was used, while we also ran test for 1, 2, 3, 4, and 5 s intervals. The first execution of the algorithm still took place at the 15 s mark in the simulation. However, the results showed nearly no difference, with a maximum throughput variation among tests of 0.7 Mbps. Shorter intervals actually showed slightly worse results, as it is harder to obtain a stable throughput estimation at such short time intervals. In more dynamic scenarios, however, executing the algorithm more frequently would allow for easier adaptation to changing rates. As such, there is a trade-off to be considered, which is a topic for future work.

V. Conclusion

This paper presents a transparent load balancing and routing algorithm for LANs, which enables the utilization of the full potential of heterogeneous networks. The algorithm takes into account the different communication technologies present in modern devices, such as Ethernet, 5 GHz Wi-Fi, and 2.4 GHz Wi-Fi, and achieves automatic load balancing across these different technologies. The algorithm aims to discover a global optimal scheduling configuration for all the flows in the network, in order to achieve maximum global throughput. The problem was formulated as a MILP and its solution was extensively tested in a variety of scenarios using the ns-3 network simulator. The results show an increase in throughput in a heterogeneous LAN up to 70 % under dynamic network conditions, compared to static priority-based interface selection, which is the current de-facto standard. Moreover, the algorithm was shown to adapt near instantaneously to the arrival and departure for flows.

Future work includes the design of better fast near-optimal heuristics to solve the MILP and distributed client-side components for fast adaptation to sudden congestion and failures. More realistic and dynamic scenarios will also be considered with different flow duration distributions, the inclusion of other network technologies, more complex network topologies and different types of traffic. The planned development of a small scale-prototype enables the evaluation of the algorithm under real-life network conditions and the comparison with, for instance, MPTCP.

Acknowledgment

This work was partly funded by the SHIFT-TV project, co-funded by imec, a research institute founded by the Flemish Government. Project partners are Nokia, Technicolor, Proximus and OpenTelly, with project support from VLAIO.

References

[1] 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.

[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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[13] R. Liu, H. Gu, X. Yu, and X. Nian, "Distributed flow scheduling in energy-aware data center networks," *IEEE Communications Letters*, vol. 17, no. 4, pp. 801–804, 2013.

[14] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 10, 2010, pp. 19–19.

[15] W. Cui and C. Qian, "Difs: Distributed flow scheduling for adaptive routing in hierarchical data center networks," in *10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2014, pp. 53–64.

[16] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2015, pp. 455–468.

[17] F. Wamser, L. Iffländer, T. Zinner, and P. Tran-Gia, "Implementing application-aware resource allocation on a home gateway for the example of YouTube," in *Mobile Networks and Management*, 2014, pp. 301–312.

[18] F. Wamser, T. Zinner, L. Iffländer, and P. Tran-Gia, "Demonstrating the prospects of dynamic application-aware networking in a home environment," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 149–150.

[19] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer, "Dynamic application-aware resource management using software-defined networking: implementation prospects and challenges," in *IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–6.

[20] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "Tcp extensions for multipath operation with multiple addresses," Tech. Rep., 2013.

[21] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "A first analysis of multipath tcp on smartphones," in *International Conference on Passive and Active Network Measurement*. Springer, 2016, pp. 57–69.

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

[23] A. T. N. TN2224, "Best practices for creating and deploying http live streaming media for the iphone and ipad," 2012. [Online]. Available: https://developer.apple.com/library/ios/technotes/tn2224/index.html

[24] 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.