Fog Computing for Smart Environments

Ulrich Huber, B.Sc.
Advisor: Jan Seeger, M.Sc.
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: ulrich.huber@tum.de

ABSTRACT

Fog Computing or edge computing is a new approach to perform services and applications at the edge of the network, which keeps latency and bandwidth usage to a minimum. Therefore fog computing is a way to enable fast responding applications in smart buildings. This paper evaluates the suitability of multiple frameworks for fog computing and from different usage areas, for use in building automation tasks. Especially compatibility with low-powered hardware, strategies for failures of components or infrastructure as well as soft-realtime capability are necessities to such frameworks. We use these three formulated requirements, to evaluate the basic architecture of six frameworks, which are from different areas of research and usage. Throughout the paper we find that there is currently no suitable candidate for deployment in smart environments. But the results of the evaluation can be used to develop a suitable framework.

Keywords

IoT, smart building, fog computing, edge computing, frameworks, EHOPES, FRODO, Gabriel, MACaaS, NetFATE, ParaDrop

1. INTRODUCTION

With the introduction of the Internet of Things (IoT) we find ourselves confronted with not only new possibilities, but also new problems to overcome. As the number of sensors in our environment rises and therefore the amount of data that needs to be transmitted, analyzed and stored, the currently used paradigm of cloud computing is unsuited for the development of the full potential of the IoT [13]. Especially if we do not only want to gain information by analyzing the data, but want to act on it in a timely manner, the round-trip-time is too high for many scenarios like health care or smart home/city. For example when sensor data represents user-interactions, which need to be analyzed to influence actuators near the vicinity of the user, we have very strict delay requirements [5]. Additionally conglomerating all sensor data to centralized data centers is infeasible, since the network bandwidth would saturate and not be scalable. Finally moving processing of gathered data to the cloud raises security concerns, since there is no direct influence on how privacy is handled by the provider of the cloud [3].
To circumvent these restrictions, the new approach of fog computing was introduced [13]. Thus fog computing enables low bandwidth and latency by using devices (fog nodes) in the vicinity of the user to store and analyze data. These fog nodes can be any type of device in the network, including devices that already have their own purposes, such as routers. To enable a multitude of services on devices with very different hardware specifications, fog computing relies very heavily on virtualization.

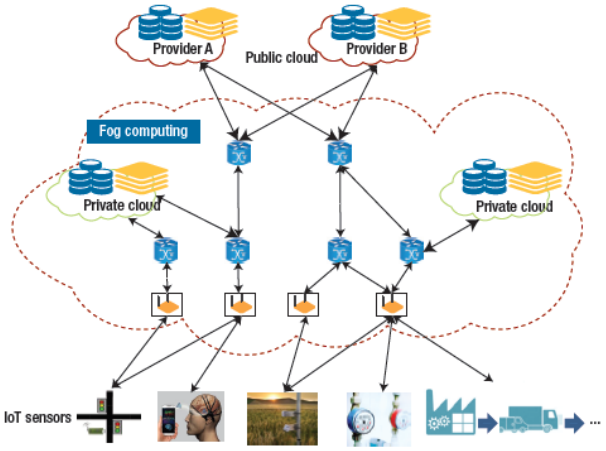

Figure 1: Distributed data processing in a fog-computing environment. Collected sensor data is dynamically procesed by nearby fog devices, including gateways and private clouds. Taken from [3].

By using fog nodes, which physically reside near the smart objects as can be seen in Figure 1, it is possible to keep latency to a minimum and perform most analysis of the sensor data without reaching out to the public cloud and thereby saving bandwidth and keeping data secure.
The previous advantages of fog computing are very important for the IoT, but most applications still require globalization as provided by the cloud. Therefore fog computing is no replacement but an expansion to the cloud, where services can be transparently moved from a centralized cloud to the very edge of the network. With the localization of services some difficulties are introduced, like the need to move the services on a fog node along with the user while he moves and the need for interoperability between different nodes, which has to be solved by frameworks using fog computing [1].
In the context of smart buildings most applications and services are latency-sensitive, for example light switches and blinds. Fog computing can be used to act upon user interac-

doi: 10.2313/NET-2018-11-1_05

tions or events within a timely manner. Since fog computing uses a distributed approach, even with increasing amounts of sensors and therefore generated data, existing infrastructure can be used to enable smart buildings, since bandwidth requirements remain stable.

In this paper we evaluate a variety of frameworks, which rely on fog computing, for their suitability for use in building automation tasks. Reliability is an important requirement for this task, since especially in cases of emergency the system needs to function to a certain degree. The easiest way to accomplish this is by building the framework with no hard dependencies on connectivity or distant resources, like the cloud. To keep deployment in buildings inexpensive and make use of "leftover" computation power of smart objects, such frameworks should have low hardware requirements. Finally many tasks, like switching lights on and off or streaming content, have soft-realtime character, which frameworks need to support. Thus strategies to instruct powerful enough devices to handle such tasks, or handle them in between multiple devices in cooperation are required. The evaluation will cover these three requirements, namely reliability, hardware requirements and soft-realtime capability.

In Section 2 we give some background on fog computing and describe the most common components of frameworks incorporating this approach. The frameworks to be evaluated, are introduced in Section 3. In Section 4 we formulate multiple evaluation criteria based on the basic requirements to frameworks using fog computation within smart buildings, and evaluate the previously introduced frameworks with them. Finally a conclusion is drawn, which approaches are especially noteworthy and should be included in frameworks for smart buildings.

2. FOG COMPUTING

In scientific literature the terms of fog computing and edge computing are interchanged quite often, ignoring the differences in meaning of the two terms. While edge computing targets the locality where services are instantiated, the edge of the network, fog computing implies the distribution of computing power, communication and data storage to devices close to the users requiring them [11]. Therefore the terms of edge computing and fog computing have a certain overlap, but while hardware for edge computing is bound to a specified service, fog computing is about creating the possibility to run any service at the edge of the network.
Thus hardware for fog computing has to provide a common interface that is suitable for many different tasks, which is accomplished by virtualization of the hardware. This has the added advantage, that there are no restrictions to the devices, which are called fog nodes, that are used in fog computing. That means, that fog computing can make use of any device from a smart light-bulb or a router with "leftover" computing power, to whole clusters of computers or even data centers, as long as a device can run the agent software needed for virtualization and communication between the fog nodes.
Not specified in the definition of fog computing but used by many frameworks that embody fog computing, is the so-called fog server. While fog nodes are the workers of such a framework, the fog server is the boss and janitor. It takes care of organizational tasks, like instantiation, deployment, migration and termination of tasks, which are then executed on fog nodes, chosen by the fog server. Additionally the fog server handles maintenance tasks, which include logging of events and redistributing work, when nodes or parts of the infrastructure fail.

3. FRAMEWORKS

In the recent years multiple groups of scientists incorporated fog computing into their projects, thus creating a huge variety of frameworks with very diverse approaches to the topic. Most frameworks are tailored to the requirements of their projects and are therefore not necessarily suited to other uses. Still there are not yet enough research projects into fog computing for smart buildings, to enable an evaluation solely using such frameworks. Therefore the frameworks, to be introduced, are not necessarily developed for the purpose of enabling smart buildings. They were selected for their unique approaches to one or more of the requirements listed in Section 1, not their compatibility with the actual automation of building tasks.

3.1 EHOPES

EHOPES is designed for the topic of smart living, which is separated into multiple subtopics for modularization. The subtopics, further on called applications, are Smart Energy, Smart Healthcare, Smart Office, Smart Protection, Smart Entertainment and Smart Surroundings. Each application brings up its own sub-network of smart objects, which provide the necessary data and actions to perform its work. EHOPES has the goal of minimizing transmission latency that smart living frameworks utilizing cloud computing show and is developed by Li et al.. The framework utilizes a star topology for its framework, where a fog server is the central node and fog nodes are the outer nodes. Additionally fog nodes can be connected to other fog nodes, to establish direct communication between them, but each fog node has to be connected to the fog server with a one-hop distance. Unfortunately Li et al. do not mention any reason for this restriction [6].

Fog Node A fog node (FN) in EHOPES is adjacent to smart objects and provides processing power as well as storage and communication abilities. As long as the agent software of EHOPES, can run on a device, it can be used as a FN. Additionally they have various interfaces to connect to smart objects, like Wi-Fi or Bluetooth. Their main purpose is to filter repetitive data from smart objects and make decisions, based on which they can take actions. FNs support collaboration between themselves and other FN and can have different capabilities depending on the requirements in a network. Last but not least, FNs are self-configuring and provide routing, security and QoS.

Fog Server The Fog Server (FS) of EHOPES is the bridge between FNs and the cloud. It hosts , not further described applications, and stores data acquired by the FNs. The applications and data are used to support FNs, additional to leasing processing power on request. With an on-demand connection to the cloud, the FS

doi: 10.2313/NET-2018-11-1_05

can work jointly and independent with/from it. Between the Cloud and the FS exists a high speed internet connection, to provide fast access to data stored in the cloud. EHOPES supports multiple FSs within a single fog network, and one physical device can take the roles of FN and FS simultaneously.
The tasks of a FS include advanced routing and switching between FNs, storing of data and applications, configuration of FNs, QoS, security and remote access for management by a maintainer. The remote access includes application deployment, data offloading, network configuration, optimization, billing and other services. How this remote access is designed, is not described by Li et al.

3.2 FRODO

Developed by Seitz et al., FRODO [10] is a fog computing extension to MIBO [9]. MIBO is a framework for multimodal intuitive building controls intended for smart buildings, that utilizes the cloud approach for a centralized decision-making and action-taking core. FRODO moves part of the tasks of the core to the edge of the network, by providing all the functionalities of the FS at each fog node (FN). Thus the FS does only handle decision-making and action-taking on the inter-FN level, while a FN can handle everything on intra-FN level.

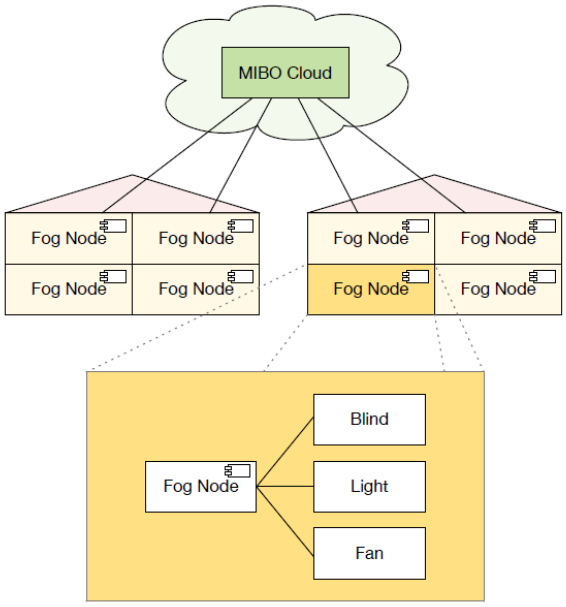

Figure 2: Architecture of FRODO, with FNs distributed on a per-room basis, taken from [10]

FRODO makes decisions based on many rules and definitions, which are used to infer the actions to take. The rules and definitions are stored in such a way, that both FS and FN can access those that are required by them, while ignoring all others. This means that a FN is a tailored clone of the FS with partial knowledge of all rules and definitions.
Rules in MIBO and FRODO are created by privileged users, for example by requesting that "All blinds are closed after 6 pm". Definitions can be created by users, for example by stating: "Open the blinds". If the previous definition is stated after 6 pm, it will contradict the previously specified rule, thus creating a conflict.
MIBO is able to negotiate such conflicts and infer the action that should be taken. FRODO moves this process to the FNs, which has the additional advantage of introducing the context of the location. Thereby decisions can also be easily tailored to specific peculiarities of the location of deployment, which is a huge improvement compared to MIBO.
Additionally FRODO is robust to network outages, since each FN can work independently from the FS. Seitz et al. propose the distribution of FNs based on rooms, as shown in Figure 2, to make rules and decisions as independent and straight forward as possible. The fog node in a room then handles all smart devices within this room. The resulting network between the FS and the FNs resembles a star topology with the FS in the center.

3.3 Gabriel

Gabriel [5] is developed by Ha et al., as an assistive system for wearable devices like Google Glass. It uses fog computing to provide crisp low-latency interaction, by offloading resource intensive processing tasks to nearby Cloudlets (fog nodes). To support mobility of the user, Gabriel supports hand-offs between Cloudlets and has multiple offload strategies implemented. Those strategies include the use of **a)** a nearby Cloudlet **b)** the distant Cloud and **c)** on-body devices like laptops or smartphones. If the Wearable device loses connection to a nearby Cloudlet it queries a Cloud-hosted list of Cloudlets for a new one nearby, that can be used to offload processing tasks. If no one is found, it offloads tasks to the Cloud and if that is also impossible, for example when no connection to the internet can be established, to the on-body device. Gabriel is additionally built to take the different latencies of the offload strategies into account and adjusts the user interface accordingly.

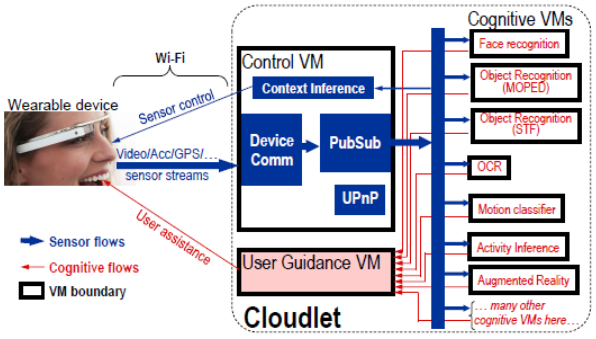

Figure 3: Back-end Processing on Cloudlet (fog node) of Gabriel, taken from [5]

As seen in Figure 3 Gabriel uses independent virtual machines (VM), with specialized cognitive engines running on them, in its Cloudlets, to provide different aspects of image processing. The cognitive engines are thinly wrapped by Gabriel to access the required sensor streams and to publish their results in a defined way. Additional to the virtual machines holding the cognitive engines (Cognitive VMs), a Cloudlet consists of two other VMs.

doi: 10.2313/NET-2018-11-1_05

The Control VM takes care of the the transportation of all required sensor data to the Cognitive VMs via an UPnP server and a publish-subscribe service. The incoming sensor data is additionally preprocessed by the Control VM, to reduce redundant tasks within the Cognitive VMs, like decoding.
The outputs of the Cognitive VMs are handled by the User Guidance VM, which integrates all of them and performs higher level cognitive processing to generate the actual user assistance that is sent to the Google Glass device, which can vary from visual aid to speech guidance.
Gabriel is a highly power saving approach, that goes as far as to switch of sensors when they are not needed and keeps processing on the wearable device very small, by offloading as much work as possible.

3.4 MACaaS

Monitoring, Analyzing and Controlling as a Serivce, in short MACaaS, developed by Byeon et al., is a platform utilizing fog computing to provide the aforementioned services for IoT devices [2]. In contrast to all other introduced frameworks, MACaaS is data-driven and utilizes a learning engine to infer new rules. Since MACaas does not include communication between fog nodes, it has no need for a fog server.

The IoT devices that can be connected to a fog node of MACaaS are separated into two groups, depending on their capabilities. If an IoT device is intelligent enough, to directly connect and communicate with MACaaS, it is classified as High-end IoT Device. If it has not the capabilities to do so (e.g. Low-end IoT Device), it will be first connected to an Edge Gateway, which is connected with the fog node instead of the device itself. The Edge Gateway can be understood as a wrapper for the IoT device, which handles communication with the framework.

A fog node of MACaaS provides all services of the platform namely Monitoring (MaaS), Analyzing (AaaS) and Controlling (CaaS). MaaS handles authentication, saving of generated data, classification of said data and monitoring of IoT device nodes and their status. AaaS tracks changes in users, IoT devices and environmental data and extracts important features of the tracked data through a learning engine, to generate new knowledge. CaaS updates event triggers and modules with the analyzed data and controls the IoT device nodes. For data backup and restoration a fog node can have more than one fog storage.

3.5 NetFATE

A solution tailored to providers of telecommunication services (TELCO) is NetFATE, which is developed by Lombardo et al. [7]. NetFATE incorporates the paradigms of Software Defined Networks (SDN) and Network Functions Virtualization (NFV), additional to fog computing to bring services to the edge of a TELCO network. Noteworthy is that NetFATE only relies on infrastructure components that are already existent in the current infrastructure of a TELCO network. Thus it is easy to deploy, as it does not need any changes to the network.
The aim of NetFATE is to provide the three service delivery models **a)** Infrastructure as a Service, **b)** Platform as a Service and **c)** Network as a Service.

NetFATE separates all its hardware components into three types, that take care of different aspects of the framework. The three logical components required for NetFATE are:

CPE/PE Node The CPE nodes and PE nodes are the fog nodes of NetFATE, since they are equipped with a virtualization framework and provide resources for the execution of VMs with network functions (NF). Physically there are two different types of nodes, the Consumer Premises Equipment (CPE) node, that resides at the location of the consumer and acts as gateway to the TELCO network. Similar to the current setups, each customer has one such CPE node, which is used exclusively by him.
The second type of fog nodes is the Premises Equipment (PE) node, which resides within the TELCO core network and has the task of aggregating many CPE nodes and is therefore used by many customers.
The fog nodes of NetFATE are general purpose computers with CentOS as a base operating system. The framework uses para-virtualization with NFs encapsulated within light network oriented OSs which can be executed on CPE/PE nodes. The nodes use the Xen Hypervisor for para-virtualization and OpenvSwitch to handle the internal network traffic between the VMs and the base OS. Deployed VMs can be migrated to other nodes on request, to support mobility of customers.

Data Centers While data centers are listed as required components, neither in the paper of Lombardo et al. nor in its cited sources exists a definite description for what they are used for.
The according ETSI Group Specification [4] lists them as environment for application virtualization and support for other nodes. Therefore it can be assumed, that the data centers support CPE/PE nodes on request and provide additional resources for processing and storage of data, as well as acting as fog nodes themselves within NetFATE. While this somehow describes their purpose, it does not describe why they are listed as required components, as they do no work that could not be done by other components.

Orchestrator The orchestrator is fog server of the framework. It handles the tasks, mentioned in Section 2 and controls the SDN by realizing routes and configuring the network interfaces of the nodes on the route. The orchestrator is a dedicated server within the telco core network and communicates with the VM hypervisor of the fog nodes via the IP network of the TELCO.
The orchestrator can be separated in three separate entities, **a)** the SDN Controller realized with POX, **b)** the NFV Coordinator, handling all tasks concerning VMs and **c)** the Orchestration Engine, which gathers information on the network and decides how the resources of the devices should be managed.
The information gathered by the Orchestration Engine includes among other things the topology of the network, the number of connected clients and the required network services. The policies used for the management of the resources can range from energy consumption to the number of hops on a route in the SDN.

doi: 10.2313/NET-2018-11-1_05

Framework	supports application migration	can handle loss of			
		cloud connection	device failure	fog node failure	fog server failure
EHOPES	✓	✓	✗	partial	✓
FRODO	✗	✓	✗	✗	partial
Gabriel	✓	✓	not applicable	✓	not applicable
MACaaS	✗	not applicable	✓	✗	not applicable
NetFATE	✓	✗	not applicable	partial	✗
ParaDrop	✓	partial	✗	✗	✗

Table 1: Results of the evaluation of reliability

Since NetFATE is intended for big TELCO networks, it is understandably oversized for smart buildings. While this is the case, the infrastructure does suite buildings very well. When looking at the framework as a hierarchy, the orchestrator with the data center stands at the top, followed by the PE nodes which finally aggregate the CPE nodes.
Comparable to the hierarchy of NetFATE, buildings can also be separated in the same way. As CPE nodes are the outermost nodes of the network in NetFATE, it is intelligent to distribute them room-wise in a building, as services will most likely be used on a per-room basis as well. The next bigger node in the hierarchy of a building would be the floor, which is reflected by PE nodes in NetFATE. Like each floor contains multiple rooms, each PE node aggregates multiple CPE nodes. Additionally when users move from room to room, a service has to be migrated to the PE node only. And only when a user changes floors it has to be migrated to the datacenter and back to the PE node of the new floor. Finally the orchestrator and data center of NetFATE can be placed anywhere in the building, while a uniform distance to all PE nodes is best for latency. As one can see, NetFATE reflects the hierarchy of buildings and thus suits the topic of smart buildings even if it is intended for distinctly bigger use-cases.

3.6 ParaDrop

ParaDrop [12] makes use of the gateway that exists in every home with internet access. Developed by Willis et al. from the University of Wisconsin, ParaDrop uses the gateways as fog nodes and positions the orchestrator in the cloud. The orchestrator of ParaDrop manages deployment and migration of VMs on fog nodes and provides APIs for companies to enable communication between their applications on the fog nodes and their services in the cloud. ParaDrop uses the widely acknowledged RESTful paradigm and a JSON-backend, for the communication between the single components, thereby simplifying development of chutes.
The virtualization in ParaDrop is realized with Linux Containers (LXC), which provides fully, virtually isolated compute containers, called chutes. In comparison to OS-level-virtualization (e.g. Lguest) LXC has the advantage of better performance and less overhead in network operations as well as fairer distribution of resources. As Willis et al. have described in their paper, LXC should be favored for I/O intensive applications. While the advantages of LXC are very interesting for devices with limited resources, the disadvantage is that applications need to be runnable on the base OS, since deployment of fully private OSs is not supported. In essence this means, that the application must be supported by the kernel used by the base OS of the fog node. The developers still claim, that porting of applications to ParaDrop containers is very easy and goes often without the need to rewrite code.
ParaDrop uses OpenFlow for effective networking between the gateway and the chutes, to distribute network resources fairly. Additionally the framework enforces very strict resource policies on CPU, RAM and the network, which can be manually edited via a management interface provided by the orchestrator. Unfortunately Willis et al. present no details on restrictions to the usage of storage space, which is most times just as limited.
Like Gabriel and NetFATE, ParaDrop supports migration of containers from one fog node to another, thus removing restrictions to the mobility of users. Even when being migrated, containers retain the user state and can resume processing of data seamlessly.

4. EVALUATION OF FRAMEWORKS

In this section we evaluate the Frameworks for their suitability for smart buildings. Since the previously described frameworks have mostly other purposes than enabling smart environments, we restrict this evaluation to the basic underlying architecture of them. To enable the evaluation of the frameworks, we use three major criteria, which are described below. The findings for each criteria are presented in a condensed form in Tables 1, 2 and 3.

4.1 Reliability

Power outage, network faults, sensor faults and loss of internet connection are only some of the problems that need to be gracefully handled by frameworks for smart buildings. In the best case, the maintainer is notified instantly and the problem is circumnavigated internally without the normal user being any the wiser, until it is solved, by switching resources or using other algorithms. But in some cases it is impossible to remain in full operational mode, when components fail. Like loss of internet connection, when a user demands a resource that is not locally available. Here frameworks should notify the user but remain operative as far as possible. Lastly smart buildings have to take the safety of inhabitants into account. For example switching on lights in case of a fire, so people can leave safely. All this needs intelligent fallback strategies that imbue all aspects of a framework.

Of the introduced frameworks, Gabriel provides the best strategies for loss of connection to certain components. It gracefully falls back to the next best offload device reliantly, and returns to better options when they are available again. Additionally it can handle faults of Cognitive VMs since it

doi: 10.2313/NET-2018-11-1_05

Framework	hardware independence	virtualization technique	requirements to fog node	requirements to fog server	requirements to infrastructure
EHOPES	✓	unknown	claimed low	unknown	high
FRODO	unknown	not applicable	unknown	unknown	unknown
Gabriel	✓	XEN	medium	not applicable	low
MACaaS	not applicable	not applicable	medium	not applicable	not applicable
NetFATE	✓	XEN	medium	high	low
ParaDrop	✓	LXC	low	not applicable	none

Table 2: Results of the evaluation of hardware requirements

gathers all data within the User Guidance VM which integrates them to the user interface [5]. FRODO has another interesting approach, in deploying nodes that work independently where possible and only rely on other resources when required. Thus providing most functionality even when there are sever problems like network faults in critical connections. The main problem of FRODO is the single fog server that has no backup available [10]. If this server fails, no connection between FENs is possible. EHOPES handles this problem by allowing more than one fog server which can work completely independent from the Cloud if necessary. Unfortunately the paper does not go into detail, how the migration from one server to the other works [6]. MACaaS is a special case when looking at the fog server. While fog nodes contain a subcomponent that is called fog server, one must not confuse this with the similar named physical component in other frameworks. MACaaS is a framework that relies on one single fog node, that is directly connected to IoT devices, with no communication between fog nodes. Thus it has the fog node as single point of failure. MACaaS can handle failing IoT devices very gracefully, with logging, notifying the maintainer and vendor and even updating drivers transparently [2]. Which stands in contrast to all other frameworks, which have absolutely no strategy to handle misbehaving IoT devices.
While most of the frameworks can handle outage of the fog server more or less gracefully, many frameworks have problems when fog nodes fail. The exceptions being Gabriel, EHOPES which can at least redistribute applications to other nodes and only loose the connection to the IoT devices connected to the failing node [7] [6]. NetFATE can redistribute work when a PE node fails, but failing CPE nodes can not be compensated, since no other connection to the customer exists.
The framework with the worst reliability is ParaDrop, since the orchestrator is positioned in the cloud. Therefore it requires internet connection for installing and migrating applications and maintanance of nodes by the network maintainer, and fails at doing these tasks when the node looses internet connection. Fortunately ParaDrop can handle loss of connection to the cloud when the previously mentioned tasks are not required by the fog node [12].

4.2 Hardware Requirements

For the practical use of a framework in a smart building not only the features it provides are of interest, but also the requirements in hardware and infrastructure. For example deploying expensive specialized hardware as fog nodes is not viable for big buildings, since the cost would be immense. Thus frameworks with low hardware requirements, that can use cheap nodes like a Raspberry Pi and are platform independent, will be favored against ones with high restrictions to hardware. Additionally frameworks with low hardware requirements make it possible to use "leftover" computational power of smart devices, that are deployed in a building and are not powerful enough to support demanding frameworks. Finally the hardware should also be power efficient to minimize maintenance cost.

With the exception of EHOPES and FRODO all frameworks provide at least some data, with which it is possible to infer their actual hardware requirements. ParaDrop requires the least amount of resources, with the orchestrator being maintained by the developers of ParaDrop itself. Thus it does not count to the actual nodes required for deployment. Additionally the fog node runs on consumer routers, comparable to ones currently distributed by TELCOs and is available as VM that can be deployed to a virtualisation environment [8]. In the paper of Willis et al. they are able to provide two additional applications to the standard router and gateway functionality on a router with an AMD Geode 500MHz CPU and 256MB of RAM as the fog node. This is possible because ParaDrop uses LXC for virtualization and not Lguest or a comparable method, which is more resource intensive [12].
While ParaDrop shines in terms of hardware requirements, MACaaS is able to work with little more resources. Byeon et al. provide no direct data on the hardware of the fog node used in MACaaS, but the applications installed on it, being Nginx, OpenMediaVault and MySQL running on Linux, point to very low requirements. To simulate the IoT device nodes, Byeon et al. use smartphones in the mid-range price segment and Raspberry Pi 2[2].
Gabriel uses for its Cloudlets clusters of four desktop machines with an Intel® Core™ i7-3770 and 32GB of RAM, calling these machines modest [5]. While this amount of computing power and RAM is partially required for the resource intensive computations the Cognitive VMs need to handle, this is also owed to the use of para-virtualization in contrast to application virtualization, since it has higher overhead, as tested by Willis et al. in their paper [12]. Therefore it can be assumed, that fog nodes of Gabriel have higher resource demands compared to ParaDrop.
Still Gabriel supports fallback strategies to less powerful devices, like smart phones, and should therefore have scalable hardware requirements. But the paper does not include if the decrease in functionality, when migrating to a fog node with less power, is only in terms of Cognitive VMs or the overall system.
In contrast to the prior frameworks NetFATE has signifi-

Framework	soft-realtime capable	chosen approach
EHOPES	✓	cooperation of fog nodes
FRODO	✓	reduction of latency by local decision-making
Gabriel	✓	choosing of most powerful fog node with lowest latency
MACaaS	unknown	
NetFATE	✓	choosing of capable node
ParaDrop	✓	use of low latency virtualization method

Table 3: Results of the evaluation of soft-realtime capability

cantly higher requirements. Intended for huge networks of TELCOs, it is understandably completely oversized when looking at smart buildings. Mainly the orchestration engine on each fog node, building on the Xen Hypervisor for para-virtualization of the NFs, requires at least a general-purpose computer, which makes deployment in smart buildings costly and possibly not viable. It is up for debate, how far NetFATE could be trimmed down to be in a competitive position to ParaDrop or MACaaS.
While EHOPES has no data on the actual hardware requirements, the paper of Li et al. mentions independence from hardware provided by Foglet and claims low requirements to the fog edge nodes. The main disadvantage of EHOPES is the restriction that fog server and fog nodes must be in one-hop proximity. Whether this refers to the term "hop" used in networking, thus disallowing switches or routers inbetween nodes, or means that no FENs are stringed together is not described in the paper [6]. If it is meant in the former we postulate that deployment in larger buildings will be very hard to realize with this restriction. Thus making EHOPES only viable for smaller buildings like small homes. If Li et al. mean stringing together FENs, it has no detrimental impact on deployment in large buildings.

4.3 Soft-Realtime Capability

The inhabitants of a smart building will evaluate the framework on reliability, evaluated in Section 4.1, and latency, since those are the two aspects which influence the user interface the most. In other terms the soft-realtime capability of a framework is as important as reliability. For example switching the light on and the framework actually switching the intelligent light bulbs on, should be two events with at the most a few seconds in between. While this scenario is not very resource intensive, there are other scenarios like streaming digital content on-demand that are harder to accomplish in nearly realtime. Therefore it is important that frameworks can handle tasks in soft-realtime, no matter how resource intensive they are.

The most obvious approach to support soft-realtime, ist to minimize latency wherever possible. FRODO does this by making decisions, concerning IoT-devices connected to a single fog node, internally in this node and thereby saving the hop to the fog server [10]. Willis et al. decided against OS-level-virtualization and for containerization to minimize latency and overhead and distribute resources more fairly in fog nodes of ParaDrop [12]. Gabriel implements the elaborate fallback strategies, additional to the restriction of communication between Cloudlet and cloud to non-time-critical parts of the code [5]. Finally Li et al. do also claim that EHOPES has very little overhead, but do not go into further detail [6].
While all those approaches are essential to enable realtime capability, they assume that one fog node will be able to handle the task on its own. This may be true for most tasks, but sometimes the physical capabilities of a fog node are inadequate for a certain task. Only NetFATE and EHOPES have implemented additional measures, to support these cases. EHOPES enables fog edge nodes to collaborate via Foglet and share resources, thereby working jointly to handle the workload. NetFATE does not allow such cooperation, but the orchestrator selects a node that is capable of handling the task on its own [7]. While this still assumes that there is a node within the network that can handle the task, it does not require every single fog node to provide the resources needed to handle every possible task. Thus allowing lightweight CPE/PE nodes and a heavy-duty node in the form of a data center. Of course this comes with additional latency, since the data center might not be in the vicinity of the user.

5. CONCLUSION

Fog computing plays a central part in enabling IoT. Especially for latency sensitive applications, like smart buildings, it can be of great advantage. None of the introduced frameworks include all necessary parts to be a perfect fit for smart buildings. Still each framework has its unique approach, providing different interesting advantages.
EHOPES brings the best reliability concerning fails of major components, with the possibility to duplicate the fog server. Additionally it handles resource intensive tasks gracefully by collaboration between multiple FENs. This approach is in some way mirrored by FRODO, where fog nodes are able to make decisions independently when they only concern smart devices attached to themselves, which ensures valuable functionality in emergency situations. MACaaS is the framework of choice, if many devices should be connected that are not able to communicate with the IoT on their own. But with the missing cooperation between multiple fog nodes it is not viable for large buildings. Finally ParaDrop shows an efficient framework for small lightweight nodes, that is easily distributed in buildings, but does not have the necessary reliability for smart buildings.
As described in Section 3.5, NetFATE is in terms of infrastructure a perfect fit for large smart buildings. Therefore it is a great start to create a framework for such buildings. To be really suitable for this scenario, it needs down-scaling of especially the CPE nodes, which can be done by utilizing the approach of ParaDrop for example. To circumvent the single point of failure the orchestrator presents, EHOPES and FRODO show great potential. When inhabitants of the building are on the move, Gabriel presents a nice strategy,

which nodes should be used as the current offload device. And finally MACaaS can be incorporated, to support even devices that existed prior to deploying the framework in the building.
Thus a suitable framework for smart buildings is a conglomeration of all introduced frameworks.

6. REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[2] G. Byeon, M. Shon, H. Lee, and J. Hong. Macaas platform for fog computing. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '17, pages 287–292, New York, NY, USA, 2017. ACM.

[3] A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116, Aug 2016.

[4] ETSI GS NFV 002. Standard V1.1.1, ETSI, 650 Route des Lucioles, F-06921 Sopia Antipolis Cedex, FR, 10 2013.

[5] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 68–81, New York, NY, USA, 2014. ACM.

[6] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner. Ehopes: Data-centered fog platform for smart living. In *Proceedings of the 2015 International Telecommunication Networks and Applications Conference (ITNAC)*, ITNAC '15, pages 308–313, Washington, DC, USA, 2015. IEEE Computer Society.

[7] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene. An open framework to enable netfate (network functions at the edge). In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015.

[8] ParaDrop. Paradrop v0.10.3 documentation, 2017. Retrieved March 31, 2018 from http://paradrop.readthedocs.io/en/latest/device/.

[9] S. M. Peters. *MIBO - A Framework for the Integration of Multimodal Intuitive Controls in Smart Buildings*. PhD thesis, Technische Universität München, 2016.

[10] A. Seitz, J. O. Johanssen, B. Bruegge, V. Loftness, V. Hartkopf, and M. Sturm. A fog architecture for decentralized decision making in smart buildings. In *Proceedings of the 2Nd International Workshop on Science of Smart City Operations and Platforms Engineering*, SCOPE '17, pages 34–39, New York, NY, USA, 2017. ACM.

[11] P. O. Östberg, J. Byrne, P. Casari, P. Eardley, A. F. Anta, J. Forsman, J. Kennedy, T. L. Duc, M. N. Mariño, R. Loomba, M. . L. Peña, J. L. Veiga, T. Lynn, V. Mancuso, S. Svorobej, A. Torneus, S. Wesner, P. Willis, and J. Domaschka. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, June 2017.

[12] D. Willis, A. Dasgupta, and S. Banerjee. Paradrop: A multi-tenant platform to dynamically install third party services on wireless gateways. In *Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '14, pages 43–48, New York, NY, USA, 2014. ACM.

[13] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42, New York, NY, USA, 2015. ACM.