Fog computing with P2P: enhancing fog computing bandwidth for IoT scenarios

Ahmad Rabay'a
Department of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
ahmad.rabaya@hhu.de

Eduard Schleicher
Department of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
eduard.schleicher@uni-duesseldorf.de

Kalman Graffi
Department of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
graffi@hhu.de

***Abstract*—Cloud computing is an architecture which several Internet-of-Things (IoT) devices relay on to outsource their computation tasks. The numbers and needs of IoT devices have grown massively, to which cloud computing could not fulfill these needs. Cloud computing has its shortcomings in terms of bandwidth, latency, and real-time response. To overcome these shortcomings fog computing paradigm was introduced, which can fulfill requests from IoT devices and forward requests to the cloud if necessary. However, there are still requests that need to go to the cloud and might suffer from its shortcomings. Our proposed peer-to-peer (p2p) fog model enhances fog computing by adding p2p mechanism into the fog layer, which allow the fog nodes to collaborate in order to meet the clients needs. This way we minimize the requests that go to the cloud and fulfill most of the requests by the fog nodes at the proximity of the user. In this research, we evaluate the proposed p2p fog model and show that it has better outcomes in term of bandwidth throughput compared to cloud computing and fog computing models. Therefore we have simulated cloud computing, fog computing, and p2p fog computing scenarios, then compared and evaluated their outcome results. Further research into fog computing could build upon this proposed model and extend existing fog computing architectures.**

Keywords: Cloud computing, Fog computing, Internet-of-Things, Peer-to-peer, Bandwidth, Simulation

I. Introduction

The number of IoT devices increases rapidly. As reported by Cisco, there will be over 50 billion devices connected to the Internet by 2020, and there will be over 500 zettabytes of data produced by 2019 [1]. However, because IoT devices are suffering from a lack of computation power and storage, many IoT computations are outsourced to strong server ends. These strong server ends are mostly deployed in the cloud, which is considered a solution to deliver services to end users and provide applications with elastic resources at low cost [2]. As we can see, cloud computing is becoming the over-arching Internet approach for information storage, retrieval and management, and IoT devices become the major outlets of service applications. This means that the key task for the next networks generation is the successful integration of cloud computing and IoT devices. However, with cloud computing, this faces several fundamental challenges:

- Network Bandwith Constraints: The vast and rapidly growing number of connected devices is creating data at an exponential rate. It is estimated that an autonomous vehicle generates about one gigabyte per second. [3] To support such resource hungry applications, it is important to keep a high-rate data exchange between cloud and IoT devices. But with the long-thin connection between cloud and mobile users, the network bandwidth becomes a bottleneck for cloud computing [4].
- Latency Requirement: many IoT applications, such as vehicle-to-vehicle communications, require latencies below a few tens of milliseconds. Mainstream cloud services cannot guarantee such latencies [4].
- Real-time response: many IoT devices rely on real-time response, e.g. wearable devices like Google glasses. [5] which is too latency-sensitive to deploy on cloud [2]

To overcome these challenges a new computing model called "fog computing" has been proposed. Fog computing extends the traditional cloud computing by placing light-weight cloud-like facility at the proximity of IoT devices. Therefore, Fog computing can serve IoT devices with a direct short-fat connection compared to the long-thin cloud connection [4]. Most of the proposed architectures of fog computing are derived from a fundamental three-layer structure. This means we have an intermediate fog layer between IoT devices and cloud [4], as illustrated in figure 1.

Compared to cloud computing, fog computing can provide enhanced service quality with much increased data rate and reduced service bandwidth cost [5]. However, like IoT devices, fog nodes are resource constraints as well. Therefore they need to contact the cloud often as some computations may require an amount of resources that cannot be single handled by the fog node. When we could fulfill as many as possible requests at the proximity of end users we would have much less requests to be sent to the cloud and much bandwidth cost saved. To achieve that, we propose a new architecture model referred to as peer-to-peer fog(p2pfog). This model enhances the three-layered client-fog-cloud architecture by introducing peer-to-peer network mechanism into the fog layer to become client-p2pfog-cloud. This way in case the fog node do not have sufficient computing and storage capacity, it can delegate some of the computations to the peered fog nodes in order to fulfill the IoT devices requests before contacting the cloud.

The goal of this research is to show that our proposed

p2pfog model can further reduce the bandwidth throughput compared to fog computing. Our approach in achieving that is by evaluating the simulation outcomes of the proposed p2pfog model compared to the cloud computing and fog computing. The simulations have been done using a P2P simulation software called "PeerfactSim.KOM". This research focuses on file-sharing application as a use case to measure and evaluate the bandwidth throughput of the three architectures models.

Overall, we want to corroborate that further research into peer-to-peer fog computing is worth the effort and large-scale networks should adapt peer-to-peer fog computing architectures to extend and improve traditional fog computing. The p2pfog model proposed in this research is motivated by IoT service requirements, user demands, and technology progress, which is to the best of our knowledge, has not been addresses before in a coordinated fashion.

The rest of the paper is divided into the following sections. Section 2 addresses the theoretical background. Section 3 describes the Use case and configuration. The evaluation of the different scenarios is introduced in section 4, followed by results discussion in section 5. Finally, relevant conclusions based on the findings are made in Section 6.

II. Theoretical Background

This section provides background information about the concepts introduced in this paper. It starts by introducing the definition and architecture of cloud computing and fog computing. Then introduces edge computing, as it is often mixed up with fog computing. After that the concept of peer-to-peer is introduced, followed by a background about the simulator used through the evaluation process.

A. Cloud Computing

Cloud computing is defined as a "model for enabling convenient, on-demand access to a virtualized shared pool of computing resources (e.g., servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort" [6]. The 2010 Davos World Economic Forum highlighted the benefits of cloud technologies compared to traditional systems, such as: 1) fast and easy deployment with no need for extensive IT experiences, or build large IT backbone infrastructure, 2) scalability where organizations can scale up or down their IT infrastructure according to their consumption needs, and 3) distribution of cost through multi-tenancy and pay-as-you-go model [7] [8].

B. Fog Computing

The term Fog Computing was first introduced by Cisco in 2012 [9]. As mention before, the idea of fog computing is placing light-weight cloud-like facility at the proximity of mobile users; the fog, therefore, can serve mobile users with a direct short-fat connection as compared to the long-thin mobile cloud connection [5]. Thus the Fog nodes are located near the user and dedicate on local service applications [5]. Being deployed near the users, fog computing can provide customized and engaged location-aware services [5]. This is one of the main characteristics of fog computing, but there are also some other advantages and characteristics [4]:

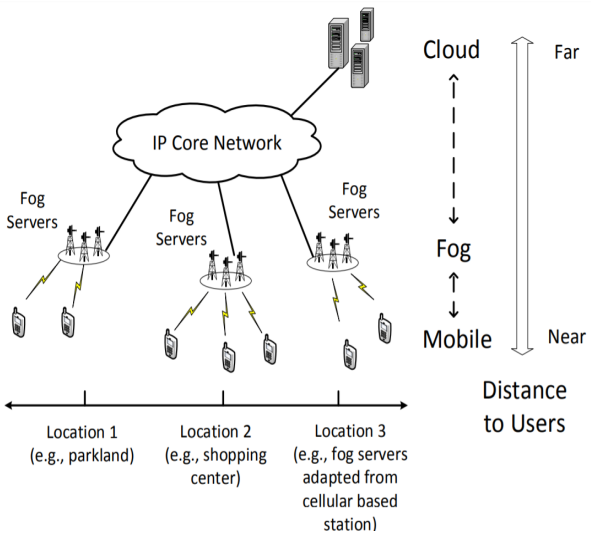


Fig. 1: Architecture of fog computing [5]

- Low latency and real-time interactions: Fog nodes process and store data by edge devices in local area network. This provides speedy high-quality localized services and reduces data movement across the Internet. Therefore it meets the demand for real-time interaction and latency-sensitive applications of IoT devices.
- Save bandwidth: Many computational tasks are done locally by the fog nodes. Only part of the data is transmitted to the cloud which can save bandwidth effectively.
- Support for mobility: IoT devices can be mobile or static devices. With the help of routing, communication or addressing protocol, fog nodes can directly interact with mobile devices and users.
- Geographical distribution: Fog computing advocate widely geographical distributed nodes, which have the ability to track and derive the locations of end devices and ensures the proximity of data analytics near the user.

Most of the proposed fog computing architectures are derived from the fundamental three-layer structure, which extends cloud computing, by introducing a fog layer between end devices and the cloud [4], as illustrated in figure 1. The fog server is a generic virtualized equipment with on-board storage, computing and, communication capability, which can deliver more intelligent and adaptive services to users [5].

C. Edge Computing

Edge computing allows computation and storage to be performed on edge devices. Edge devices perform computation tasks to reduce the network latency and bandwidth between end devices and the cloud. End devices not only act as data consumers but also as data producers, so these devices not only request services from the cloud, they also provide services to the cloud [10]. The term edge computing is often mixed with fog computing. [11] They both have the same prime objective,

TABLE I: Differences between Edge and Fog Computing

Parameter	Edge computing	Fog computing
Location of data	Edge devices	Network edge devices
Multiple IoT applications	Unsupported	Supported
Focus	Things level	Infrastructures level
Proximity to end devices	In End devices	Near End Device
Resource	More limited	Limited

as they bring computing and storage capabilities to the edge of the network to reduce service latency and save network bandwidth [12]. However, Edge computing focuses more on IoT devices side as the computing should happen at the edge, while fog computing focuses on the infrastructure side [10]. More differences are listed in Table 1 [4]

D. Peer to Peer (P2P)

In the scope of this work peer-to-peer(P2P) is necessary for routing the communication between the fog nodes, which allows to share computing resources through direct exchange between systems. P2P systems can be classified as centralized/decentralized systems, further, the decentralized systems can be classified as structured/unstructured systems [13].

Centralized peer-to-peer systems mix features of centralized (e.g., client-server) and decentralized network architectures. There are one or more central servers, which help peers to locate their desired resources or coordinate actions among them. In a decentralized peer-to-peer system, every peer has equal rights and responsibilities. Peers only have a partial view of the P2P network and offer data/services that may be only relevant for some queries/peers. The difference between structured and unstructured systems relays in how queries are being forwarded. In an unstructured P2P system, each peer is responsible for its own data. Peers keep track of a set of neighbors that it may forward queries to. Structured P2P systems use a mapping between data and peers [13].

For this research, we use Chord, which is a structured, decentralized peer-to-peer system [14]. Chord supports one operation: given a key, it maps the key onto a node. Chord uses a variant of consistent hashing to assign keys to Chord nodes [15]. In Chord every node has a unique overlay identifier, which are arranged in a ring structure. Every node has a successor and predecessor. The predecessor is the next node in the circle in the counter-clockwise direction. The successor, in contrary, is the next node in the clockwise direction. Moreover, each Chord node knows some short-cuts to nodes further away, are called fingers. Every lookup is sent to the closest finger, which is still preceding the queried identifier. Each data is assigned to a key, and the key/data item pair is stored at the node whose corresponding responsibility interval contains the key. To lookup data, the node queries the corresponding key, this results in a query that is routed towards the peer, whose responsibility interval contains the key. [16]

E. Basics of PeerfactSim.KOM

In order to evaluate the three computing scenarios, we used PeerfactSim.KOM as a simulator for this research. PeerfactSim.KOM is a discrete event-based simulator written in Java and focuses on the evaluation of multi-layered peer-to-peer systems. PeerfactSim.KOM allows us to do largescale, multi-parameter comparative evaluations of peer-to-peer systems [14]. The multi-layered architecture of PeerfactSim.KOM is shown in figure 2. Each layer provides well-defined interfaces to express its individual functionalities, operations, and provided services [16]. Remarkable for this thesis is the overlay layer, application layer, and network layer. The overlay layer implements structured, unstructured, and information dissemination overlays. It operates on top of the network and transport layer. The workload for a specific application can be modeled with the application layer. The network layer is used for modeling delay, bandwidth, jitter, and loss between two nodes in the simulation. [14]. General setup and configurations of the PeerfactSim.KOM can be found in the following section.

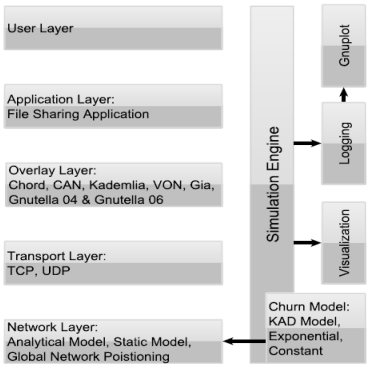


Fig. 2: Layered Architecture of PeerfactSim.KOM [16]

III. Use case and simulator setup

Fog computing can be used for various computational tasks, however we need to specify one use case that we will use to evaluate and compare the behavior of the intended scenarios. This use case is a read-only file-sharing application, because it is a first attempt to demonstrate a peer-to-peer fog scenario, and it is a simple use case with no much complications.

A. Use Case

For this research we used a read-only file-sharing application to evaluate and compare the behavior of the three scenarios. Which means that users can not update or change the files they receive, also they cannot upload them. In summary, a file request goes from a client to a server as lookup, and the response publishes the file to the client. Fog servers behave like storage units that pre-cache files, deliver files to clients, and request files from the cloud. Cloud node retrieves lookup requests, and responds with the requested files.

The file-sharing mechanism is already implemented in PeerfactSim.KOM, which allows us to define large sets of files and the distribution of their lookup and publish probability. For storing and publishing files, the file-sharing application relies on the offered functionality of the Chord overlay. Based on the published files, peers can randomly seek a lookup.

Peers can start a single lookup for a randomly chosen file based on the provided keys. When simulating a file-sharing application we need to define a file-set. In this research we use FlatDocumentSet, which is a simple file-set where every file has the same probability to be stored and looked up.

The file-sharing application provides different lookup and publish operations, where the publish operation publishes a random amount of resources from a given set and distributes them exponentially. The lookup operation looks up a random file from a file-set, but only takes files that were published before. The lookup period is exponentially distributed with a mean value that can be configurated. [16]

B. Setup and configurations of PeerfactSim.KOM

Table 2 shows the parameters we used for configuring the three scenarios in PeerfactSim.KOM. We ran multiple simulations for each scenario during this research. The number of nodes was increased gradually starting from 100, 200, and 1000 nodes. These nodes are divided into 1 cloud node, 10% fog nodes and around 90% client nodes. For the simple fog and p2p fog scenarios, the fog nodes implement a caching mechanism that can store up to 90% of the file-set. The file-set we have defined is a FlatDocumentSet of 100, 500, and 1000 files, to check the impact of number of files on the simulations outcomes, with each file size is 10 kilo bytes. The network layer we used is simple network layer as it addresses high memory efficiency, which we need for our high scaled simulations [16]. We did not apply any churn model, which means that nodes will always be online [14]. We used the existing file-sharing application in order to implement our file-sharing mechanism.

For simulating the fact that fog nodes are at the proximity of end users, we place fog and client nodes in Germany. On the other hand, the cloud is placed in America. The simulation time with 100 min allows the scenario to simulate enough load to obtain reasonable results [14]. The used overlays are already implemented in PeerfactSim.KOM but we customized them to fit our scenarios. In this research we use Chord protocol for the peer-to-peer implementation. Three key features that distinguish Chord from many other peer-to-peer overlays are its simplicity, provable correctness and provable performance, these features are the reasons for our decision [15].

In general, a simulation begins with 1 min interval where all nodes join the overlay. After that, the cloud node has about 1 min to store and prepare the files to be published. When storing is done, the actual simulation part begins as files are ready and all nodes have already joined the overlay. Now client nodes start to lookup for random files in uniform distributed intervals [14]. The lookup distribution reflects how often a client sends a new file request. We start with 5 seconds, then gradually reduce it to 2 seconds, and 1 seconds in further simulations. We have chosen these values to see the impact of the lookup distribution in the evaluation of the scenarios, by getting a varied amount of traffic. With a decreased lookup distribution value we have more lookup requests from the clients.

TABLE II: Constant parameter

Parameter	Value
Number of nodes	100 / 200 / 1000
Simulation time	100 min
Network layer	Simple(static delay, no packet loss)
Churn model	none
Overlays	Centralized Storage and Chord
Applications	File Sharing
File set distribution	Flat(100 / 500 / 1000 files)
File size	10 kilo bytes
Lookup distribution	5s / 2s / 1s Interval

The configurations and parameters values are chosen to be good enough to illustrate our goal and purpose of the evaluation, and bring our scenarios closer to realistic IoT scenarios. However, the values of these parameters could be further changed to illustrate different configurations.

To run the simulations, we have used our personal computers for the scenarios with small parameters values, while we used the high-performance computer at the Heinrich Heine University Düsseldorf "HILBERT" to run the scenarios with larger parameters values that our PCs were not able to handle.

IV. Scenarios and Evaluation

In this section we discuss the three scenarios with their different architectures. Then we evaluate and compare the scenarios separately and against each other with the help of the data provided by PeerfactSim.KOM, which provides many output metrics. Because not every metric is relevant to our use case, we have chosen the metrics that are directly related to our research goal, as we want to show the enhanced bandwidth throughput of our proposed model compared to simple fog computing and cloud computing. We have only chosen the in/received metrics because the out/send metrics are similar for all scenarios. The chosen metrics are:

- Net Bandwith in: Shows the overall bandwidth throughput in bytes per second for all the peers in the whole scenario during the simulation time in minutes.
- NetPeers Bandwith in: Shows the bandwidth throughput in bytes per second of every single peer starting from the first peer which is the cloud, until the last peer.

As we have previously detailed, during this research we have used different setup values for the simulator parameters to get more insight into the impact of each of the parameters in the simulation outputs. However, due to the limited space of the publication, only a part of the simulations will be illustrated, but it will be enough to show the results of the evaluation. The configuration parameters we have considered are: 100 nodes, 100 minutes, 100 files with both 2 and 1 second lookup distribution period, the rest of the parameters did not change. We have chosen those values, even though with bigger configuration values we got better comparison results, because the output figures will look less complicated to illustrate our comparison, and it would be easier to spot the timely enhancement per peer.

Based on the simulations of different configuration parameters values, it was clear that the lookup distribution value has

the most impact on the simulator outcomes. The lower lookup distribution value is, the closer our scenario gets to an IoT scenario. Moreover, the number of nodes and size of the file set has similar impact, as they both when increased, our proposed p2pfog scenario gives better bandwidth throughput compared to the other two scenario. On the other hand, changing the simulation time was insignificant, so we kept it the same.

The caching system at the fog nodes saves up to 90% of the files, however, when the cache is full, a random cached file will be replaced by the requested file. This way we can ensure that most but not all of the lookup requests will be handled by the fog nodes directly over a long runtime.

A. *Cloud Computing*

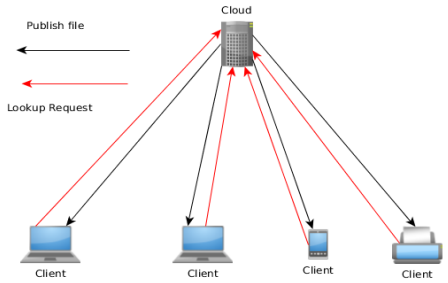
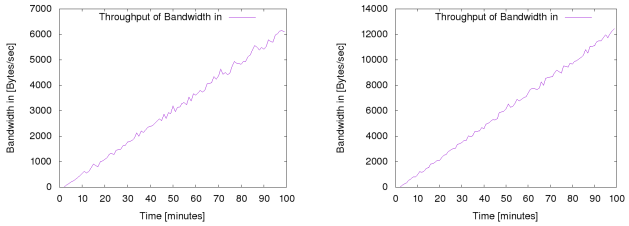


Fig. 3: Architecture of cloud computing scenario

In this scenario, client nodes are directly connected to the cloud. Client nodes send file requests to the cloud, and the cloud responds with the requested files. To simulate the scenario, we have used 100 nodes, which are divided into 1 cloud node and 99 client nodes. We have evaluated the results under 2 lookup distributions; 2 and 1 seconds. Figures 4 and 5 illustrate the bandwidth throughput of the cloud computing scenario for the whole system and for the peers respectively.

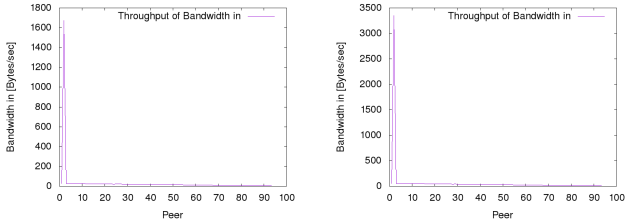
Figure 4 with both parts 4(a) and 4(b) shows that the bandwidth throughput value is raising steadily through the simulation time, as more nodes are joining and starting new requests for files. However, the cloud node is responsible for handling all of that bandwidth, as all lookup requests go to the cloud, who then publishes the files to the clients. This is clearly illustrated in figures 5(a) and 5(b), as the cloud node ”first node” has most of the bandwidth throughput compared the other nodes.

Moreover, it is clear the impact of lookup distribution on the bandwidth throughput value, as illustrated in figures 4(a) and 4(b). With 1 sec lookup distribution as in figure 4(b), the bandwidth throughput is almost double the value when the lookup distribution is 2 sec as in figure 4(a). Because with a decreased lookup distribution value we have more overall lookup requests. Furthermore, figures 5(a) and 5(b) show that changing the lookup distribution value from 2 sec to 1 sec, the cloud node bandwidth throughput has almost doubled. In figure 5(a) with lookup distribution value is 2 sec, the bandwidth throughput of the cloud node around 1700



(a) 2s distribution (b) 1s distribution

Fig. 4: Cloud Computing Scenario - Net Bandwidth



(a) 2s distribution (b) 1s distribution

Fig. 5: Cloud Computing Scenario - Net Peers bandwidth

bytes/second, while it was doubled to 3400 bytes/second with lookup distribution value of 1 sec as in figure 5(b).

B. *Fog Computing*

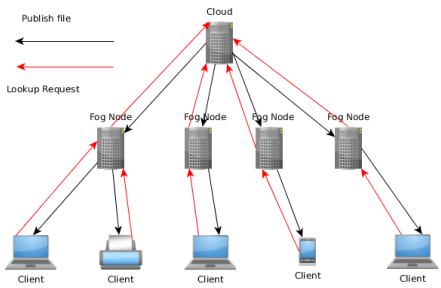
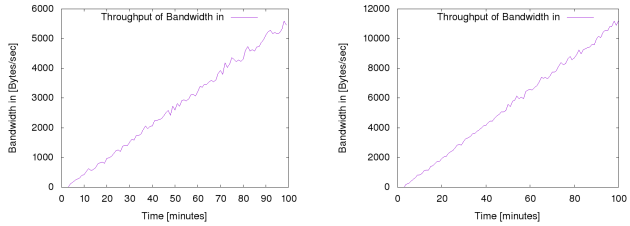


Fig. 6: Architecture of fog computing scenario

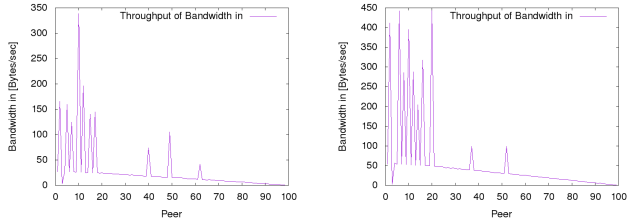
The fog computing scenario is a basic three-layer fog computing architecture, which extends the cloud architecture by adding an intermediate fog layer [4]. This fog layer responds to clients requests and sends requests to the cloud if needed. In this scenario, fog nodes are pre-cache units that are empty at the beginning. Every time a fog node receives a file request from a client, it first checks if it contains the requested file. If the fog node contains the file, it responds to the client node with the requested file. Otherwise, the fog node starts a lookup request for the file from the cloud. The cloud then responds to the fog node with the requested file. Now, as the fog node receives the file, it caches the file and send it to the client.

The different diagrams in this section show the simulation results for 100 nodes, which are divided as 89 clients, 10 fog nodes, and 1 cloud node. We have evaluated the results under 2 lookup distribution values; 2 and 1 seconds. The bandwidth



(a) 2s distribution (b) 1s distribution

Fig. 7: Fog Computing Scenario - Net Bandwidth



(a) 2s distribution (b) 1s distribution

Fig. 8: Fog Computing Scenario - Net Peers bandwidth

throughput of the fog computing scenario is illustrated in figure 7 for the whole system and figure 8 for the peers.

Figures 7 with both parts 7(a) and 7(b) show that the bandwidth throughput is raising linearly through the simulation time, as more nodes are joining and starting new requests for files. All lookup requests go from the clients to the fog nodes, and the fog nodes either respond with the file, in case it has the file, or send a lookup request to the cloud to get the file, then send the file back to the client. As the fog nodes are the first point of contact, they are handling most of the traffic as illustrated in figures 8(a) and 8(b). We can see from the figures that the fog nodes has the highest the bandwidth throughput, while the other nodes including the cloud have less bandwidth throughput value. Moreover, the bandwidth throughput will start to drop gradually, with some exceptions, after last the fog node (the 10th node). This is clearly the goal of fog computing is reducing the bandwidth throughput of the cloud.

On the other hand, it is clear the impact of the lookup distribution on the bandwidth throughput. As illustrated in figure 7, with 1 second lookup distribution as in figure 7(b), the bandwidth throughput increased in a raising manner compared to figure 7(a) when the lookup distribution is 2 second. Furthermore, figure 8 shows that changing the lookup distribution from 2 to 1 second has a huge impact on the cloud bandwidth throughput, which increased from around 150 bytes/second for the 2 sec lookup distribution as in figure 8(a) to more than 400 bytes/second for 1 sec lookup distribution as in figure 8(b).

C. P2P Fog Computing

Our proposed P2P fog scenario can be seen in figure 9. We extend the fog computing scenario by routing fog nodes among each other with peer-to-peer Chord mechanism. Client nodes still request files from the closest fog node to its range. Also, fog nodes remain pre-cache units, that directly respond to the client nodes if they have the requested file. But as fog nodes are now connected together through peer-to-peer and use Chord lookup operations to search for files, when the fog node does not have the requested file, it starts a lookup request for the file from other peered fog nodes.

However, instead of contacting all the fog nodes, the system tries to search for the file in a static amount of time. Every P2P lookup has a maximum time for searching the files, if one of the routed fog nodes contains the file, it replies with the file to the requesting fog node. Again the file is cached in the fog node before being sent to the client. But, there is still the opportunity that none of the routed fog nodes respond with the file, either because the P2P lookup time is over, or because the file has never been requested before, so it is not cached in any fog node. In this case the requesting fog node starts a new lookup request. This time it directly requests the file from the cloud. The cloud responds with the file, and the fog node caches the file, before sending it to the client.

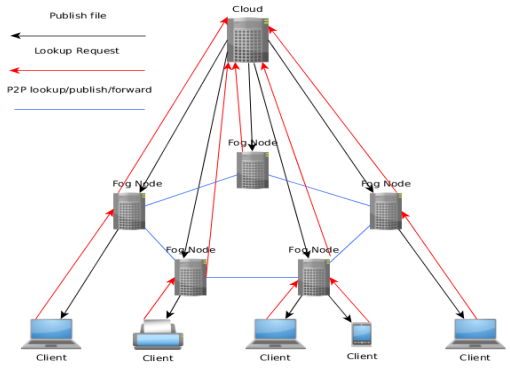
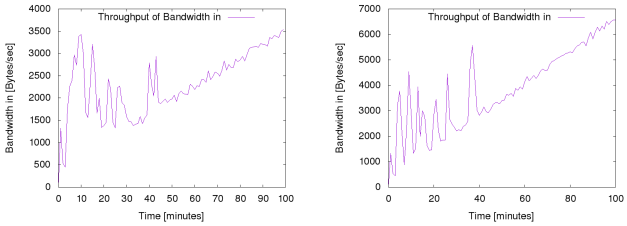


Fig. 9: Architecture of p2p fog scenario

The following diagrams show the simulation results for 100 nodes. The 100 nodes in the p2p fog scenario are divided as 89 clients, 10 fog nodes, and 1 cloud node. We have evaluated the results under the 2 different lookup distribution values; 2 and 1 seconds. The net bandwidth throughput of the p2p fog computing scenario is illustrated in figure 10 for the whole system and figure 11 for the peers.

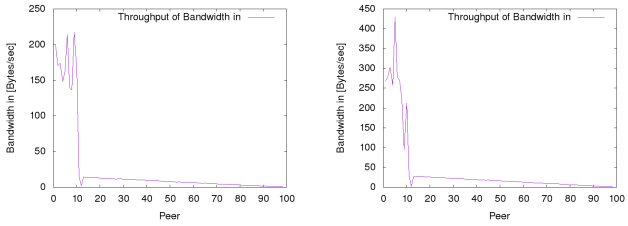
Figure 10 with both parts 10(a) and 10(b) shows the bandwidth throughput starting from 0 and raising sharp, then drops a little before raising again very sharply to reach a peak. After that it starts to drop until it reaches a more stable point where it increases rather steadily through the simulation time, with some exceptions. The reason for the sharp raise at the beginning, is that the fog nodes caches are empty, so all the clients requests go through the contacted fog node, to a number of peered fog nodes, until lookup timeout is reached. After that the requesting fog node contacts the cloud to retrieve the file, then sends it to the client. All this creates a huge traffic at the beginning of the simulation.

However, when time passes, more requests have been sent through the fog nodes, which means more files have been



(a) 2s distribution (b) 1s distribution

Fig. 10: P2P Fog Computing Scenario - Net Bandwidth



(a) 2s distribution (b) 1s distribution

Fig. 11: P2P Fog Computing Scenario - Net Peers bandwidth

cached. So after some time, between 15 - 20 minutes, according to figures 10(a) and 10(b), there is a drop in the bandwidth throughput as the fog nodes are able to fulfill the clients requests, either individually or through peered fog nodes, without contacting the cloud. This is clearly illustrated as well in figures 11(a) and 11(b), as the cloud and fog nodes are the first point of contact for the clients, they are handling most of the traffic, and have most of the bandwidth throughput. So, the bandwidth throughput drops sharply after the 10th node, which is the last fog node.

On the other hand, the impact of lookup distribution on the bandwidth throughput is illustrated in figures 10(a) and 10(b). With 1 sec lookup distribution as in figure 10(b), the bandwidth throughput increased in a raising manner compared to figure 10(a) when the lookup distribution is 2 sec.

Furthermore, figures 11(a) and 11(b) show that changing the lookup distribution value from 2 sec to 1 sec has a huge impact on the fog nodes bandwidth throughput, which peaked around 200 bytes/second for the 2 sec lookup distribution as in figure 11(a) to more than 400 bytes/second for 1 sec lookup distribution as in figure 11(b). However, the cloud node bandwidth throughput increased slightly compared to the sharp increase in the bandwidth throughput for the fog nodes. The cloud node bandwidth throughput increased from 200 bytes/second for the 2 sec lookup distribution to 270 bytes/second for the 1 sec lookup distribution as illustrated in figures 11(a) and 11(b) respectively.

When comparing the increase in the bandwidth throughput between fog nodes and the cloud node, under the 2 lookup distribution values, illustrated in figures 11(a) and 11(b), we could notice the impact our proposed scenario have by reducing impact on the bandwidth throughput of the cloud node in a huge manner, compared to the other scenarios.

V. Comparison and Discussion

As we have evaluated each scenario and provided an overview of the simulation results. In this section we will compare the different scenarios in term of bandwidth throughput and shed the light on the impact of our proposed p2pfog model compared to the cloud computing and fog computing.

For the overall bandwidth throughput, we have summarized the data in figures 4, 7 and 10 into Table IV which shows the overall bandwidth throughput of the three scenarios under 2 sce, 1 sec, in addition to 5 sce lookup distributions. And in order to give a better view of the comparison, Table III shows the number of finished operations for the three scenarios under the same lookup distribution values.

TABLE III: Number of Finished Operations

Lookup Distribution	Cloud	Fog	P2P Fog
5 second	1180	1050	1160
2 second	2900	2550	2650
1 second	5900	5200	5300

Based on the data from Table III and IV we notice that the cloud computing scenario has the highest bandwidth throughput value compared to the other two scenarios for the all simulations. But on the same time, the cloud scenario managed to finish more operations than the other two in the same period. On the other hand, remarkably, the proposed p2pfog scenario managed to finish more operations than the fog scenario in the same time period, and still have much less overall bandwidth throughput. As well, we notice the impact of the lookup distribution value, as with the cloud and fog scenarios the overall bandwidth throughput value has multiplied around 5 times. But for the proposed p2pfog scenario the impact was less and showed a better bandwidth throughput value which is multiply around 4 times.

On the other hand, for the peers bandwidth throughput, we have summarized the data in figures 5, 8 and 11, into Table V which shows the peers bandwidth throughput for the three scenarios under 2 sce, 1 sec, in addition to 5 sce lookup distribution values. We notice the huge difference in the bandwidth throughput for the cloud node between the three scenarios, in addition to the impact of the lookup distribution value. For the cloud scenario, the cloud node bandwidth throughput is 670 bytes/sec, which is very high compared to the other scenarios. As in the cloud computing scenario, the cloud node is responsible for all the requests that come form the clients. The cloud node bandwidth throughput has changed hugely when we reduced the lookup distribution from 5 sec

TABLE IV: Overall bandwidth throughput

Lookup Distribution	Cloud	Fog	P2P Fog
5 second	2400 bytes/sec	2100 bytes/sec	1500 bytes/sec
2 second	6100 bytes/sec	5500 bytes/sec	3500 bytes/sec
1 second	12200 bytes/sec	10500 bytes/sec	6600 bytes/sec

TABLE V: Cloud node bandwidth throughput

Lookup Distribution	Cloud	Fog	P2P Fog
5 second	670 bytes/sec	77 bytes/sec	170 bytes/sec
2 second	1700 bytes/sec	160 bytes/sec	200 bytes/sec
1 second	3400 bytes/sec	410 bytes/sec	240 bytes/sec

to 2 and 1 sec, to reach 1700 and 3400 bytes/sec respectively.

For the fog scenario, the cloud node bandwidth throughput under 5 sec lookup distribution is 77 byte/sec, which is the lowest compared to the two other scenarios, because the load is now shared between the cloud node and the fog nodes. It was increased to 160 byte/sec when we reduced the lookup distribution to 2 sec, but it is still lower than the other scenarios. However, it increased sharply to 410 bytes/sec for 1 sec lookup distribution, due to the huge amount of requests coming from the clients.

Regarding our proposed p2p fog scenario, the cloud node bandwidth throughput showed a very significant behavior when we reduced the lookup distribution value from 5 sec to 2 and 1 sec. At the time the other two scenarios showed huge increase in bandwidth throughput when we reduced the lookup distribution from 5 sec to 2 and 1 sec. The cloud node bandwidth throughput in the p2p fog scenario increased slightly from 170 bytes/sec for the 5 sec lookup distribution to only 200 and 240 bytes/sec for the 2 sec and 1 sec lookup distribution value respectively, which is less than the other two scenarios with a huge margin.

This enhancement caused due to the fact that after some time, and with more requests, more files will be cached at the fog nodes, and at the long run, it will be very rare that the fog node will contact the cloud asking for files.

VI. Conclusion

P2P Fog computing is a proposed model to enhance the bandwidth throughput of fog computing, in order to meet the emerging needs of IoT devices. Our p2p fog model extends the fog computing architectures by introducing p2p Chord mechanism into the fog layer. To provide evidence to our claim we used PeerfactSim.KOM to simulate and compare cloud computing, fog computing and p2p fog computing scenarios. We have chosen metrics that highlight our research goal, to evaluate and compare the bandwidth throughput of the three scenarios.

In our simulations, we have decided for a 100 minutes simulation time, however in reality such systems will run for weeks or even months, meaning that the difference margin will be much bigger. However, even for this small simulation time, compared to realistic IoT scenarios, we could see that our proposed p2p fog model provides better bandwidth throughput compared to the cloud computing and fog computing. We could notice the impact our proposed scenario will have in the IoT systems by reducing the bandwidth throughput of the cloud node in a huge manner.

The results of this research provide a strong basis for future work toward optimizing fog computing using p2p mechanism. One area of future work is examining other p2p overlays to see whether they could have better outcomes in terms or bandwidth throughout as well as latency and response time. Also there could be researches about optimize the caching system, so fog nodes could try to predict the user demands and pre-cache the files that are often needed. In addition to try using a different file-set where files could have priorities, so when the cache is full, the file with the least priority might be discarded rather than discarding a file randomly.

VII. Acknowledgement

We would like to thank Dr. Andreas Disterhöft from Heinrich Heine University Düsseldorf for his suggestions and contributions which provided insight and expertise that greatly assisted the research. In addition to the "Centre for Information and Media Technology" (ZIM) at the Heinrich Heine University Düsseldorf (Germany) which provided the computational support and infrastructure.

References

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," CISCO white paper, vol. 1, pp. 1–11, 2011.

[2] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). IEEE, 2015, pp. 73–78.

[3] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, 2016.

[4] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," Journal of Network and Computer Applications, vol. 98, pp. 27–42, 2017.

[5] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," arXiv preprint arXiv:1502.01815, 2015.

[6] B. Ohlman, A. Eriksson, and R. Rembarz, "What networking of information can do for cloud computing," in 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises. IEEE, 2009, pp. 78–83.

[7] C. Shapiro, S. Carl, H. R. Varian et al., Information rules: a strategic guide to the network economy. Harvard Business Press, 1998.

[8] J. Gordon, C. Hayashi, D. Elron, L. Huang, and R. Neill, "Exploring the future of cloud computing: riding the next wave of technology-driven transformation," in World Economic Forum, 2010.

[9] 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. ACM, 2012, pp. 13–16.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[11] A. Davis, J. Parikh, and W. E. Weihl, "Edgecomputing: extending enterprise applications to the edge of the internet," in Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. ACM, 2004, pp. 180–187.

[12] S. U. Khan, "The curious case of distributed systems and continuous computing," IT Professional, no. 2, pp. 4–7, 2016.

[13] Q. H. Vu, M. Lupu, and B. C. Ooi, Peer-to-peer computing: Principles and applications. Springer Science & Business Media, 2009.

[14] M. Feldotto and K. Graffi, "Systematic evaluation of peer-to-peer systems using peerfactsim. kom," Concurrency and Computation: Practice and Experience, vol. 28, no. 5, pp. 1655–1677, 2016.

[15] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking (TON), vol. 11, no. 1, pp. 17–32, 2003.

[16] D. Stingl, C. Gross, J. Rückert, L. Nobach, S. Kaune, and K. Pussep, "Peerfactsim. kom: A large-scale simulation framework for peer-to-peer systems," HPCS 2011, July, 2011.