Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Reducing IoT Bandwidth Requirements by Fog-based Distributed Hash Tables

Ahmad Rabay'a, Pascal Kudla, Lukas Kubitza, Kalman György Graffi, Michael Schöttner

2020

Preprint:

This is an accepted article published in 4th Conference on Cloud and Internet of Things (CIoT) . The final authenticated version is available online at: https://doi.org/10.1109/CIoT50422.2020.9244294 Copyright 2020 IEEE

Reducing IoT Bandwidth Requirements by Fog-Based Distributed Hash Tables

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

Pascal Kudla
Institute of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
pascal.kudla@hhu.de

Lukas Kubitza
Institute of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
lukas.kubitza@hhu.de

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

Michael Schöttner
Institute of Computer Science
Heinrich Heine University Düsseldorf
Düsseldorf, Germany
michael.schoettner@uni-duesseldorf.de

***Abstract*—Internet-of-Things (IoT) devices often rely on cloud services for computations and data management. As the number of IoT devices is growing fast, the Internet connection to the cloud data-center can become a bottleneck regarding bandwidth and latency. Fog computing has been introduced by Cisco to address these challenges. The proposed peer-to-peer (p2p) Fog model allows fog-nodes to cooperate and reduce the bandwidth requirement to the cloud and the system as a whole. In this publication we evaluate Pastry and CAN for implementing a DHT on fog nodes compared to implementing Chord on fog nodes. The simulation shows that Pastry offers better results in terms of bandwidth consumption compared to non-cooperative Fog computing, Chord Fog, and CAN Fog models. The overall reduction in bandwidth reflects as well to reduction in costs.**

***Index Terms*—Cloud computing, Fog computing, Internet-of-Things, Peer-to-peer, Bandwidth, Simulation**

I. Introduction

In this age of IoT everything is more and more connected. As reported by Cisco, there will be 5.3 billion total Internet users (66 percent of global population) by 2023. While in 2017 a single person had 2,4 devices and a monthly traffic of 16,2 GB, in 2023 the devices count will increase by 50% but the monthly traffic will more than triple to 49,8 GB. Out of the 29.3 billion devices connected to the Internet by 2023, there will be 14.7 billion IoT devices. Moreover, the annual global IP traffic will reach 4.8 Zettabytes per year by 2022, and the monthly IP traffic will reach 50 GB per capita by 2022, up from 16 GB per capita in 2017 [1] [2]. Because IoT devices lack computation power and storage, many IoT computations are outsourced to the cloud. Around 90% of the internet is depending on services offered by cloud directly or indirectly [4]. Therefore, cloud computing is becoming the overarching Internet approach and IoT devices become the major outlets. This means that a key task for the next networks generation is the successful integration of cloud computing and IoT devices. The cloud centralization combined with rising number of IoT devices induces many challenges, such as mentioned in [5]:

- Network Bandwidth: The growing number of connected IoT devices is creating data at an exponential rate. With the long-thin connection between cloud and mobile users, the network bandwidth becomes a bottleneck, as predicted in [6].
- Latency Requirement: many IoT applications require latencies below a few tens of milliseconds. Mainstream cloud services cannot guarantee such latencies [6].
- Real-time response: many IoT devices rely on real-time response which is too latency-sensitive to be on cloud [7]

To overcome those challenges the Fog Computing paradigm was proposed which is: A cloud-like layer set up just above the edge of the internet at the proximity of the edge devices. This Fog layer has the capability to handle part of the edge traffic, reducing the bandwidth load of the cloud [6]. Fog Computing is already a successful approach in reducing internet traffic to the cloud while also providing reduced response time and latency. Nevertheless a single Fog node is limited in computational resources thus still highly dependent on the cloud. When a Fog node cannot satisfy a client's request, the cloud has to be contacted and the internet backbone has to be used again thus cloud bandwidth is burdened.

Because sole Fog computing has limited computational resources to relieve load from the backbone, in [8] a new model called peer-to-peer fog (p2pfog) has been proposed. This model enhances the three-layered client-fog-cloud by introducing Chord peer-to-peer mechanism into the Fog layer to become client-p2pfog-cloud, as illustrated in Fig. 5. In this model, in case the Fog node does not have sufficient computing and storage capacity, it can, using Chord overlay, delegate some of the computation tasks to the peered Fog nodes in order to fulfill the IoT devices requests rather than contacting the cloud. It was noticed that this model using Chord overlay achieved improved results in terms of reducing the bandwidth consumption for both the Cloud node and the system as a

whole [8].

In this paper we implement in simulation and evaluate a realistic fog computing workload on three p2p overlays, namely: Chord, Pastry and content addressable network (CAN). We show that each overlay react and produce different results. This research focuses on an IoT data access setup as a use case to measure and evaluate the bandwidth consumption of the different scenarios.

The contribution of the paper is to corroborate that further research into peer-to-peer fog computing is worth the effort and large-scale networks should adopt p2p fog computing architectures. The proposed p2p fog model is motivated by IoT service requirements, user demands, and technology progress.

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. After that the concept of peer-to-peer is introduced along with the three overlays examined in this research namely: Chord, Pastry and CAN, followed by a background about the simulator used through the evaluation process.

A. Cloud Computing

Cloud computing is a "model for enabling convenient, on-demand access to a virtualized shared pool of computing resources that can be rapidly provisioned and released with minimal management effort" [9]. The 2010 Davos World Economic Forum highlighted the advantages of cloud technologies over 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 [10] [11].

B. Fog Computing

The term Fog Computing was first introduced by Cisco in 2012 [4], which is a light-weight cloud-like facility at the proximity of mobile users. Fog computing serves mobile users with a direct short-fat connection as compared to the long-thin mobile cloud connection [5]. Being deployed at the proximity of end 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 [6]:

- Low latency: Fog nodes store and process data by edge devices which reduces data movement across the Internet and provides speed high-quality localized services.
- Save bandwidth: As many computational tasks are done locally by the fog nodes, only small portion of the data is transmitted to the cloud.
- Support for mobility: With the help of routing, fog nodes can directly interact with mobile IoT devices and users.

The fog node is a generic virtualized equipment with on-board storage, computing and, communication capability, that can deliver more intelligent services to users [5]. The architecture of Fog computing extends the three-layer structure by introducing a fog layer between end devices and the cloud [6], as illustrated in Fig. 2.

C. Peer to Peer (P2P)

Peer-to-peer (P2P) is a distributed network architecture where all participating devices are independent regarding processing power, storage capacity and bandwidth. The devices have the capability to access each node in the network and are capable of sharing their resources. So in respect to the classic server-client architecture the devices in a P2P network can represent both roles, client and server [12]. Further overlay networks create a virtual topology on top of the physical topology [13]. While Distributed Hash Table (DHT) is a distributed data structure that provides a bijective key to resource-value mapping function to accomplish easier and more determined routing [14]. In the scope of this work peer-to peer (P2P) is necessary for routing the communication between the fog nodes, which allows sharing computing resources through direct exchange between systems. Three of the most popular overlays are investigated in this paper: Chord, Pastry, and CAN. These overlays are classified as decentralized structured overlays. In a decentralized P2P system every peer has equal rights and responsibilities, while structured P2P systems use a mapping between data and peers [15].

In Chord every node has a unique overlay identifier, these identifiers are arranged in a ring structure, where every node has a successor and predecessor [16]. Moreover, each Chord node knows some short-cuts to nodes further away, are called fingers, and every lookup is sent to the closest finger. Each data is assigned to a key, and the key/data item pair is stored at the node whose identifier succeeds the data object's identifier. 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 [17] [18].

In Pastry each node has a unique numeric identifier ($nodeId$) and holds routing information to nodes with $nodeIds$ numerically closest to its own as well as to nodes in exponentially growing distances. When presented with a message and a numeric key, a Pastry node efficiently routes the message to the node with a $nodeId$, which is numerically closest to the key among all currently live Pastry nodes [19].

CAN utilizes a d-dimensional Cartesian coordinate space, a d-torus, so each participating node owns a zone of that space. A hash function is used to map any (key, value) pair onto a specific coordinate on the d-torus. A node which owns a zone in the coordinate space only knows its direct neighbors. Routing utilizes the coordinate that is calculated from the key,

so a lookup is sent to the neighbor that is nearest to the corresponding coordinates to forward the request [20].

D. Basics of PeerfactSim.KOM

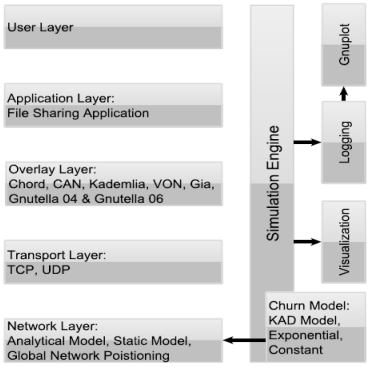


Fig. 1: Layered Architecture of PeerfactSim.KOM [17]

In this research PeerfactSim.KOM simulator is used to evaluate the computing scenarios. PeerfactSim.KOM is an event-based simulator which gives us the capabilities to do large scale, multi-parameter comparative evaluations focusing on multi-layered P2P systems, as shown in Fig. 1 [18]. Each layer provides well-defined interfaces to express its functionalities and services [17]. The overlay layer implements the overlays, and 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, and loss between nodes [18].

III. Use case and simulator setup

Fog computing is used for various computational tasks. However, our goal is to implement in simulation and evaluate a realistic fog computing workload. We need a scenario that causes traffic between connected devices. Even though choosing a resource intensive application might serve our goal, such a complex applications might add undesired complexity to our attempt. That is why a basic IoT data access scenario, where users pull relevant content from the network, such as sensor information, is considered as a use case to measure and evaluate the bandwidth consumption of the different scenarios.

A. Use Case

An IoT data access scenario is used for the evaluation, where devices retrieve relevant data from the network or update information to the network through push/pull based functionality. It allows to both store content at a specific location (push) or to contact a specific node that provides a desired content (pull). The scenarios we are examining consist of an IoT setting where a number of edge devices send pull requests for certain data source repeatedly. Starting by a client sending an information pull request to a server, then, in response, the server pushes the information to the client. Cloud node receives pull requests, and responds with the requested information. Fog nodes behave like storage units that pre-cache the information, deliver information to clients, and request information from the cloud. The investigated IoT data access scenario provides different pull and push operations, where the push operation pushes random resources from a given set. The pull operation looks up a random information piece from a data-set, but only takes information that were pushed before [17].

B. Setup and configurations of PeerfactSim.KOM

PeerfactSim.KOM gives us several configuration options, Table 1 shows the parameters used for configuring the different scenarios in PeerfactSim.KOM. The considered configuration parameters are: 100 nodes, 100 minutes, 100 data elements with 10, 2 and 1 second lookup distribution period, the rest of the parameters did not change. These values have been chosen to illustrate our comparison as this setting supports the clear identification of the differences of the chosen overlays.

The 100 nodes are divided into 1 cloud node, 10% fog nodes and around 90% client nodes. The fog nodes implement a caching mechanism that store up to 90% of the data-set. When the cache is full, a random cached information item is replaced by the requested data piece. This way we ensure that most but not all of the pull requests are handled by the fog nodes directly over a long runtime.

In the Data-set every data element has the same probability to be stored and looked up. The network layer is simple network layer as it addresses high memory efficiency and reduces complexity [17]. To avoid any random factor, no churn is applied, which means nodes are always online [18] [21].

For simulating the fact that fog nodes are at the proximity of end users, the fog and client nodes are placed in Germany, and the cloud is placed in America. The simulation time is set at 100 min which allows the scenario to simulate enough load to obtain reasonable results [21]. The investigated overlays are customized to fit our scenarios.

In general, a simulation begins where all nodes join the overlay. After that, the cloud node prepares the data items to be pushed, then client nodes randomly start to pull for random information in distributed intervals [21]. The lookup distribution reflects how often a client sends a new information pull request. We start with 10 seconds, then gradually reduce it to 2 seconds, and 1 second in further simulations to get varied amount of traffic. With a decreased lookup distribution value we have more pull requests from the clients.

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

IV. Scenarios and Evaluation

In this section we discuss the architectures of the four different scenarios. After that we evaluate and compare the scenarios separately and against each other with the help of the data provided by PeerfactSim.KOM. There are many output metrics provided by PeerfactSim.KOM, however, due to the

TABLE I: PeerfactSim.KOM configuration parameters

Parameter	Value
Number of nodes	100
Simulation time	100 min
Network layer	Simple(static delay, no packet loss)
Churn model	none (see text)
Overlays	Centralized Storage, Chord, Pastry, and CAN
Application	IoT data access setup
Data-set	100 data elements
Data element size	10 kilo bytes
Lookup distribution	10s / 2s / 1s Interval

limited space of the publication, the metrics that are directly related to our research goal are chosen:

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

The diagrams in this section show the simulation results for 100 nodes (divided as 89 clients, 10 fog nodes, and 1 cloud node), and under 2 lookup distribution values; 2 and 1 seconds.

A. Simple Fog Computing

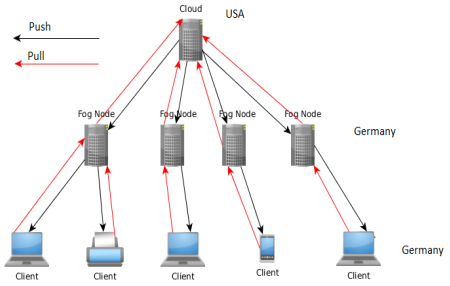


Fig. 2: Architecture of fog computing scenario

The simple fog scenario is a three-layer fog computing architecture, as illustrated in Fig. 2. The fog layer responds to clients requests and sends those requests to the cloud if needed. The fog nodes act as pre-cache units that are empty at the beginning. Whenever a fog node receives an information request from a client, first it checks whether it contains that information. If yes, it sends the information item to the client. Otherwise, the fog node starts a pull request to the cloud. The cloud then sends the requested information to the fog node. When the fog node receives the information piece, it caches the information and sends it to the client. The bandwidth consumption of the fog computing scenario is illustrated in Fig. 3 for the whole system and Fig. 4 for the peers.

Fig. 3 shows that through the simulation time as more nodes are joining and starting new requests for information, the bandwidth consumption is raising linearly. However, the raise is more sharp when the lookup distribution is 1 sec compared to 2 sec, which illustrate the impact of the lookup distribution on the bandwidth consumption. Moreover, all clients' pull requests go to the fog nodes, which in return either respond with the data item, if it has it, or send a pull requesting the

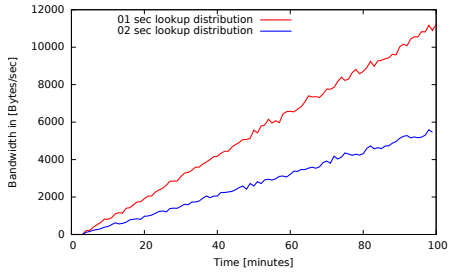


Fig. 3: Simple Fog Scenario - Net Bandwidth

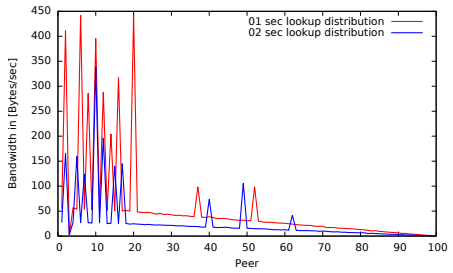


Fig. 4: Simple Fog Scenario - NetPeers Bandwidth

information from the cloud, then send the information back to the client. As the fog nodes are the first point of contact, they are handling most of the traffic as illustrated in Fig. 4. From the figure we see that the fog nodes have the highest bandwidth consumption values compared to the other nodes including the cloud. Moreover, after last fog node (the 11th node) the bandwidth consumption starts to drop gradually, with some exceptions. Furthermore, Fig. 4 shows that changing the lookup distribution from 2 to 1 second has a huge impact on the cloud node bandwidth consumption, which increases from around 160 bytes/second for 2 sec lookup distribution to more than 400 bytes/second for 1 sec lookup distribution.

B. Chord Fog Computing

The following three p2p fog scenarios extend the Fog computing scenario by routing the fog nodes among each other with a P2P overlay mechanism (Chord, Pastry, or CAN) as illustrated in Fig. 5. It works the same way as the simple fog scenario; client nodes still request information from the closest fog node to its range. Also, fog nodes remain pre-cache units that directly respond to the clients if they have the requested piece of information. But as the fog nodes are now connected together through a P2P overlay, when the fog node does not have the requested data item, instead of contacting the cloud immediately, it first starts a pull request for the information from the peered fog nodes.

However, instead of contacting all the peered fog nodes, the system tries to search for the data in a static amount of time. Every P2P lookup has a maximum time to search for the data item, if one of the routed fog nodes contains the data, it replies with the data piece to the requesting fog node. Again the data item 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 information, either because the P2P lookup time is over, or because the data item

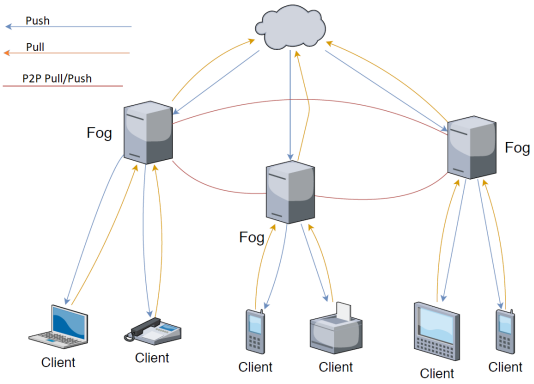


Fig. 5: Architecture of p2p fog scenario

TABLE II: Layer settings of the Chord fog scenario

Layer	Client	Fog	Cloud
Application	IoT data access	IoT data access	IoT data access
Service	FogNodeDiscovery	CloudNodeDiscovery	-
Overlay	Centralized Storage	Chord	Centralized Storage
Translayer	TCP	TCP	TCP
Netlayer	Simple	Simple	Simple

has never been requested before, so it is not cached in any fog node. In this case the requesting fog node starts a new pull request. This time it directly requests the information from the cloud. The cloud then responds with the data items to the requesting fog node, which caches the information, before sending it to the client.

In Table II we see the settings of the Chord fog model based on the PeerfactSim.KOM layered architecture. The Application, Translayer, and Netlayer remain the same. For the Service layer, FogNodeDiscovery service is used in the Client nodes, which contains the contacts information of all fog nodes, while CloudNodeDiscovery service is used in the Fog nodes, and it contains the contact information of the cloud node. Furthermore, the overlays used are; Centralized Storage overlay which is used in both the Client nodes and the Cloud node, while Chord overlay is used in the Fog nodes. In the Centralized Storage overlay the clients contact the next higher hierarchy level, whereas the cloud acts as a server. The net bandwidth consumption of the Chord fog scenario is illustrated in Fig. 6 for the whole system and Fig. 7 for the peers.

Fig. 6 shows that the bandwidth consumption starts from 0 then raises sharp, before drops a little and raises again very sharply to reach a peak. Then it starts to drop until it reaches a stable point where it increases rather steadily through the simulation time. The sharp raise at the beginning is caused because the fog nodes' caches are empty initially, so the clients' requests go through the contacted fog node, to a number of peered fog nodes, until the lookup timeout is reached. Then the cloud is contacted to retrieve the information item and sends it to the requested fog node, which caches the information before sending it to the client. All this creates a huge traffic at the beginning of the simulation.

However, as the simulation goes, more data items are cached at the fog nodes. As Fig. 6 shows, between 15 - 20 minutes,

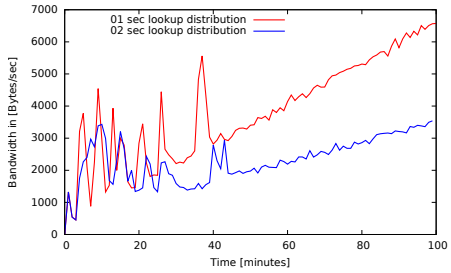


Fig. 6: Chord Fog Scenario - Net Bandwidth

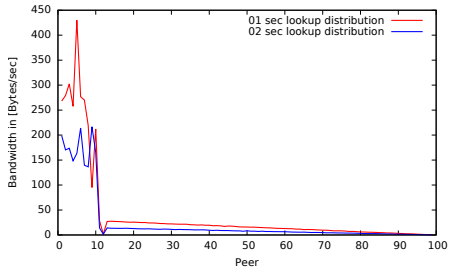


Fig. 7: Chord Fog Scenario - NetPeers Bandwidth

there is a drop in the bandwidth consumption as the fog nodes are able to fulfill most of the clients' requests, either individually or through peered fog nodes, without contacting the cloud. This is illustrated as well in Fig. 7 as the bandwidth consumption drops sharply after the 11th node, which is the last fog node.

The figures 6 and 7 illustrate the impact of lookup distribution on the bandwidth consumption. Fig. 6 shows that with 1 sec lookup distribution the bandwidth consumption almost doubled compared to when the lookup distribution is 2 sec. Furthermore, Fig. 7 shows that changing the lookup distribution value from 2 sec to 1 sec has a huge impact on the fog nodes bandwidth compared to the impact it has on the cloud. For the fog nodes, the bandwidth consumption value peak at slightly over 200 bytes/second for the 2 sec lookup distribution to more than 400 bytes/second for 1 sec lookup distribution. However, for the cloud bandwidth consumption, it increases slightly from 200 bytes/second for the 2 sec lookup distribution to 270 bytes/second for the 1 sec lookup distribution. When comparing the increase in the bandwidth consumption between the fog nodes and the cloud, under the 2 lookup distributions, we notice that using the Chord fog model reduces the bandwidth consumption of the cloud in a significant manner, because that the peered nodes in the fog layer are now able to fulfill most of the clients requests.

C. Pastry Fog Computing

The settings of the Pastry fog model are similar to the Chord fog model in Table II except for the Overlay layer. The overlays used in the Pastry fog model are; the fogPastry overlay, which is derived from the Centralized Storage overlay, is used in both the Client nodes and the Cloud node, and the Pastry overlay is used in the Fog nodes. The architecture of the Pastry fog model is the same as the Chord fog model, except that the fog nodes are connected to a Pastry overlay

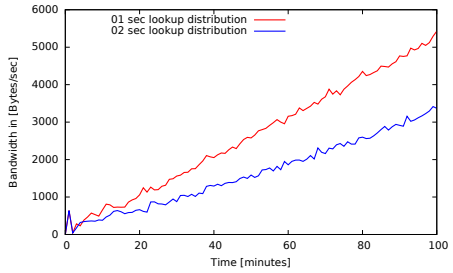


Fig. 8: Pastry Fog Scenario - Net Bandwidth

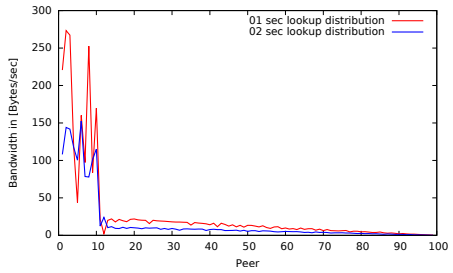


Fig. 9: Pastry Fog Scenario - NetPeers Bandwidth

instead of a Chord overlay. The net bandwidth consumption of the Pastry fog computing scenario is illustrated in Fig. 8 for the whole system and Fig. 9 for the peers.

Fig. 8 shows the bandwidth consumption starting from 0 and raising little sharp, then drops before raising rather steadily through the simulation time until the end of the simulation. The reason for the little raise at the beginning and the general behavior is similar to the Chord fog scenario that the fog nodes' caches are empty at the beginning, and being filled gradually overtime. However, due to the nature of the Pastry overlay as it has a different routing algorithm that handles the communication between the fog nodes in a different way, different diagrams with different values are produced compared to the Chord fog scenario. More details about the comparison between the scenarios are in section 5. Furthermore, if we look as well at Fig. 9 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 consumption. Therefore, the bandwidth consumption drops sharply after the 11th node, which is the last fog node.

On the other hand, the impact of lookup distribution on the bandwidth consumption is illustrated in Fig. 8. With 1 sec lookup distribution the bandwidth consumption increases in a sharp raising manner compared to when the lookup distribution is 2 sec. Furthermore, Fig. 9 shows that changing the lookup distribution value from 2 sec to 1 sec has an impact on both the cloud and fog nodes bandwidth consumption. As we can see, the cloud node peaks around 150 bytes/second for the 2 sec lookup distribution, to more than 250 bytes/second for 1 sec lookup distribution. However, unlike the Chord fog scenario, in the Pastry fog the cloud node handles more bandwidth consumption than the fog nodes for the 1 sec lookup distribution. Despite that, still the increase in the cloud node bandwidth consumption in the Pastry fog scenario is not significant as illustrated in Fig. 9.

D. CAN Fog Computing

The settings of the CAN fog model are similar to the Chord fog model in Table II except that fog nodes at the CAN overlay communicate via UDP on Translayer, while Cloud and client nodes communicate via TCP like the other models. Moreover, in CAN fog model the Centralized Storage overlay is used in both the Client nodes and the Cloud node, while the Fog nodes use CAN overlay. The abstract concept of message traffic is identical to Chord and Pastry fog scenarios. The net bandwidth consumption of the CAN fog scenario is illustrated in Fig. 10 for the whole system, and Fig. 11 for the peers.

Fig. 10 shows the bandwidth consumption 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. The reason for the sharp raise at the beginning and the overall behavior is similar to the Chord fog scenarios. Moreover, after some time, between 15 - 20 minutes, according to Fig. 10, there is a drop in the bandwidth consumption as the fog nodes are able to fulfill the clients requests, either individually or through peered nodes, without contacting the cloud. The behavior of the CAN fog is more similar to the Chord fog in many ways rather than Pastry fog. On the other hand, the impact of lookup distribution on the bandwidth consumption is illustrated in Fig. 10, as with 1 sec lookup distribution the bandwidth consumption increases in a raising manner compared to the 2 sec lookup distribution.

Furthermore, as illustrated in Fig. 11, we notice that the cloud and fog nodes are handling most of the traffic and the bandwidth consumption drops sharply after the 11th node, which is the last fog node. Moreover, the figure shows that changing the lookup distribution value from 2 sec to 1 sec has an impact on the fog nodes bandwidth consumption, jumping from around 370 bytes/second for the 2 sec lookup distribution to more than 450 bytes/second for 1 sec lookup distribution. There is a similar increase in the bandwidth consumption value of the cloud node, which increases from 220 bytes/second for the 2 sec lookup distribution to 350 bytes/second for the 1 sec lookup distribution.

V. Comparison and Discussion

In this section we compare the different scenarios in term of bandwidth consumption and shed the light on each of the p2p fog models, compare them to each other and to the simple fog computing scenario.

For the overall bandwidth consumption, the data in Figures 3, 6, 8 and 10 are summarized into Table III which shows the overall bandwidth consumption of the four scenarios under 10, 2 and 1 sec lookup distributions values. Looking at Table III we notice that for a 10 sec lookup distribution value the Simple fog scenario performs better than the three other p2p fog scenarios, and the Pastry fog scenario is the second best. However, when the lookup distribution value is reduced to 2 sec and 1 sec, we notice an increasing gap in the bandwidth consumption values between the Simple fog scenario and the p2p fog scenarios. The bandwidth consumption value for the

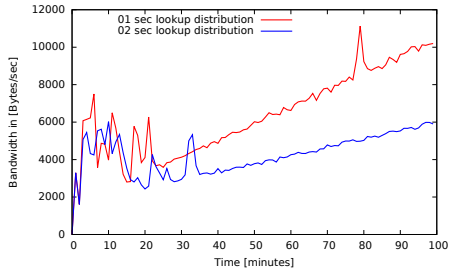


Fig. 10: CAN Fog Scenario - Net Bandwidth

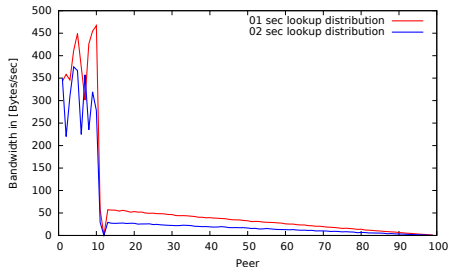


Fig. 11: CAN Fog Scenario - NetPeers Bandwidth

Simple fog scenario increase 5 and 10 times with 2 sec and 1 sec lookup distribution values respectively, comparing to the bandwidth consumption values when the lookup distribution value is 10 sec. For the three p2p fog scenarios; the CAN fog produces the highest values in terms of bandwidth consumption which are close to the values of the Simple fog scenario, while the Pastry fog scenario gives the best results followed by the Chord fog scenario as the second best.

On the other hand, for the peers bandwidth consumption, the data in Figures 4, 7, 9 and 11 are summarized into Table IV which shows the Cloud node bandwidth consumption for the four scenarios under 10, 2 and 1 sec lookup distributions values. We notice that for the 10 sec lookup distribution the Simple fog scenario gives the best results in terms of bandwidth consumption for the cloud node with just 46 byte/sec. However when reducing the lookup distribution to 2 sec, the cloud node bandwidth consumption jumps to 160 byte/sec and a maximum of 410 byte/sec when the lookup distribution is reduced to 1 sec. On the other hand, looking at the Pastry fog scenario, it proves to be the best scenario in terms of Cloud node bandwidth consumption when reducing the lookup distribution to minimum of 2 sec and 1 sec. When

TABLE III: Overall bandwidth consumption - bytes/sec

Lookup Distribution	Simple Fog	Chord Fog	Pastry Fog	CAN Fog
10 second	1157	2008	1532	2510
2 second	5456	3520	3367	5901
1 second	11258	6580	5428	10195

TABLE IV: Cloud node bandwidth consumption - bytes/sec

Lookup Distribution	Simple Fog	Chord Fog	Pastry Fog	CAN Fog
10 second	46	140	100	176
2 second	160	200	143	220
1 second	410	300	273	358

we reduce the lookup distribution to 2 sec and 1 sec, the cloud node bandwidth consumption increases slightly from 100 byte/sec for 10 sec lookup distribution to 143 byte/sec and 273 byte/sec for 2 sec and 1 sec lookup distribution respectively. This increase is almost insignificant compared to the huge increase in the Simple fog scenario under the same lookup distribution values.

For the 2 other scenarios, the overall behavior of the Chord fog scenario is similar to the Pastry fog scenario. At 10 sec lookup distribution the bandwidth consumption of the cloud node in the Chord fog scenario shows a higher value than the Simple fog and Pastry fog scenarios. But when the lookup distribution is reduced, the bandwidth consumption of the cloud node shows better results compared to the simple fog scenario, however, still not better than the Pastry fog scenario. On the other hand, when looking at the CAN fog scenario, we can see a similar pattern to the other p2p fog scenarios (Pastry fog and Chord fog), but with a higher bandwidth consumption values for the cloud node compared to the two other p2p fog scenario for all the different lookup distribution values.

In this research, different patterns have been noticed regarding how the four scenarios react against decreasing the lookup distribution value. While the Simple fog scenario is affected massively by reducing the lookup distribution value from 10 sec to 2 sec and 1 sec, the p2p fog scenarios (Pastry fog, Chord fog and CAN fog) are affected rather slightly and in a less margin than the Simple fog scenario. However, each of the p2p fog scenarios produces different values and are varied in how they respond to reducing the lookup distribution values. Moreover, the reason that for 10 sec lookup distribution value the Simple fog scenario produces better bandwidth consumption values compared to the p2p fog scenarios, that in the p2p fog scenarios the fog nodes cache is empty at the beginning, so the clients requests travel through multiple nodes before it eventually retrieves the information from the cloud. And for a scenario with not much pull requests from the clients, very little requests will go to the fog nodes, therefore the fog nodes cache is filled rather slowly, this causes extra bandwidth consumption compared to retrieving the information from the cloud immediately as in the Simple fog scenario. However, for 2 sec and 1 sec the p2p fog scenarios show a very significant behavior by producing better bandwidth consumption compared to the Simple fog scenario, because the fog nodes caches are filled faster, and more and more requests are fulfilled at the fog later without the need to contact the cloud. This means that the proposed model works better in resource intensive scenarios such as IoT systems.

Through Tables III and IV we see, that applying p2p fog models shift the traffic load from the cloud to the fog nodes, especially under frequent queries from the clients. During times of very sparse queries, the p2p fog model performs still well but comes at a higher overall maintenance load at the fog nodes. This means that the simple fog scenario is suitable for the typical systems that rarely contact the cloud, such as hourly sensor updates. However, the proposed p2p fog scenarios show an improved saving in terms of reducing bandwidth

consumption for the systems that require an intensive resources and communications with the cloud such as smart homes and self driving cars. That is mainly due to the nature of the p2p overlay network of communication and cooperation between the fog nodes to fulfill the client requests without contacting the cloud. Moreover, the Pastry overlay produces better results compared to Chord and CAN because it has a different routing algorithm that handles the communication between the peered fog nodes in a different way. This routing algorithm determines the best routes to store in the routing table, which minimize the travel time of the message between the peered fog nodes by quickly locate the responsible node. In addition to reducing the overall cost of routing by avoiding the need to flood packets.

From all the above, we notice the impact of the p2p fog models, especially the Pastry fog model, by reducing impact on the bandwidth consumption of the cloud node and the system as a whole in a huge manner, compared to the Simple fog model. Furthermore, it is worth to mention that, even though CAN fog produces the highest bandwidth consumption values among the p2p fog models, in this research the most simple CAN case has been examined, which is a CAN architecture using only two dimensions for DHT and routing. CAN gives the option to increase the dimensions to achieve a faster routing by increasing the resources needed to calculate the DHT values. Having that in mind, CAN might still worth more research, at least when p2p fog computing refines and more practical use cases appear.

Because of the limited space, only the bandwidth consumption metrics has been evaluated in this publication. Other important metrics such as latency and response time are planned to be evaluated and discussed to have a bigger and more detailed overview about the impact of the proposed p2p fog models. This means that even though Pastry proves to be the best overlay to reduce the bandwidth consumption of an IoT scenarios that might be in an expense of other costs.

VI. Conclusion

Fog computing has been introduced as an intermediate layer between the cloud and IoT devices to address the bandwidth limitation of Internet connections. Applying peer-to-peer (p2p) concepts in the fog is a promising approach to aggregate the resources of fog nodes. We propose a cooperating p2p fog model by deploying a distributed hash table on fog nodes in order to cache request results from the cloud data center. Overall this allows reducing bandwidth requirements and costs at the cloud operator.

In the past we have studied Chord as a basic DHT in the fog layer. In this paper we extend our research by evaluating Pastry and CAN as alternative and compare all the three approaches. The concepts are implemented and evaluated using the PeerfactSim KOM simulator. The results show clear benefits over a basic fog layer with non-cooperating nodes. The simulation results further show that Pasty performs best regarding bandwidth reduction. The simulations have been run for 100 minutes and we expect even better results for real-world setups where applications run for months or years (of course depending on the access patterns and fog node resources). In future work we plan to focus on request latency which is important for time-critical applications, in addition to implementing a cache replacement strategies in the fog nodes.

References

[1] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, vol. 1, 2018.

[2] G. M. D. T. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[3] Cisco, "Cisco annual internet report (2018–2023) white paper," *White Paper*, vol. 1, 2020.

[4] 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.

[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] 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.

[7] 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.

[8] E. Schleicher, K. Graffi, and A. Rabay'a, "Fog computing with p2p: Enhancing fog computing bandwidth for iot scenarios," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 82–89.

[9] 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.

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

[11] 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.

[12] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," 09 2001, pp. 101 – 102.

[13] D. DD and D. O'Mahony, "Overlay networks: A scalable alternative for p2p," *Internet Computing, IEEE*, vol. 7, pp. 79 – 82, 08 2003.

[14] S. Sarmady, "A survey on peer-to-peer and dht," *arXiv preprint arXiv:1006.4708*, 2010.

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

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 149–160. [Online]. Available: http://doi.acm.org/10.1145/383059.383071

[17] K. Graffi, "Peerfactsim. kom: A p2p system simulator—experiences and lessons learned," in *2011 IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2011, pp. 154–155.

[18] 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.

[19] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2001, pp. 329–350.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383072

[21] M. Feldotto and K. Graffi, "Comparative evaluation of peer-to-peer systems using peerfactsim. kom," in *2013 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2013, pp. 99–106.