Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Berlin, 07.08.18
Ort, Datum

Schacht
Unterschrift

Service Offloading Solution in Heterogeneous Networks

Linus Schacht
Matrikelnummer: 365615

DAI-Labor, Technische Universität Berlin
Faculty of Electrical Engineering and Computer Science
linus.schacht@campus.tu-berlin.de

Abstract. In this seminar paper, I evaluate different approaches for service offloading in a heterogeneous network. Therefore I present a taxonomy which should represent the crucial points for offloading in fog computing. With the help of the taxonomy, I take a closer look at the approaches and evaluate how sufficient they tackle the challenge of services offloading in a fog network.

1 Intro

In the future, the importance of the Internet of Things (IoT) will increase, and even today there is a large number of wirelessly connected devices [10]. Those personal an IoT devices generate a significant amount of data which need to be handled.
However, those devices have limited resources. As a result, they are not capable of computing and process all these data onboard and therefore will need to offload them. One Solution is to offload those tasks to the cloud, but the cloud lacks some desirable features for some IoT devices such as low latency, location awareness and widespread geographical distribution. Therefore a different technique is required to acquire those abilities, the approach of fog computing can provide these [10].
The concept of fog computing is to bring the computation or processing of data from resource-limited end devices to the edge of the network as instead to the cloud. Therefore the physical way is much shorter, and latency is decreasing. However, the concept of fog computing is not a replacement of the cloud it functions as an extra layer between cloud datacenter and IoT devices [11]. The concept of fog computing is that it uses traditional network components, cloudlets and more as computational resources to allocate computation processes at the edge of the network near to the end devices. Such components are called fog nodes, and for an efficient fog computing network it is essential to manage the different task like application management, data management or power management. In this seminar paper, we will focus on the topic of application management and more specifically on the offloading of service in a fog network [11].

1.1 Motivation

The main idea of the urban vehicular networks is to provide resources for future intelligent transport systems to ensure driving safety, traffic efficiency and exchanging of valuable information[4]. Those computational and network resources are not use if the cars are standing on a parking space. The result is unused resources, but if all those cars build a distributed network of fog nodes which can provide limited computational resources as well as transmitting messages. For example the resource intensive process of virtual reality, where low latency is critical, can be offloaded partly be offload to the fog to boost it performance and utilize all the resources which would be otherwise unused.

1.2 Problem Description

Fog computing seems to fill a gap which is actually needed, but in reality, fog computing is not widespread. To find the cause, it is essential to take a closer look at the challenges which fog computing provide for service offloading.

The first challenge is to determine when it is beneficial to offload a task to the fog or cloud. The end device has limited processing resources for the tasks on board, so it needs to define a threshold where it makes sense to offload some task for example when the latency is lower than the waiting time in the local queue for the process.
Secondly, where to offload a service in a fog? In contrast to the cloud, the fog is building out of heterogeneous devices which have different resource limits and are distributed located. Therefore the latency is different for each fog node. Somehow the system needs to determine the best fog node for a task in a reasonable time, or it exists a central control fog node who orchestra the process of offloading. Also, as well decide when it is more useful to offload the task to the cloud instead of offloading them to a fog node.
Thirdly to ensure up to date information. The position of end devices or IoT can be changing so the distance between device and fog node will be different. In that case, it is necessary that a system update constantly the latency between the end device and fog nodes to ensure the decision to offload is made with correct information.
Fourthly is the challenge of a heterogeneous network where the network consists for example routers, cloudlets or a vehicle as a fog node. Therefore the different fog nodes need to have a protocol or an API to communicate which each other.
The next challenge is to provide security and privacy. In the case of cloud data centres, they are owned by a trusted cloud service provider. In Fog computing the owner of a fog node can, for example, be an internet provider, cloud service provider which expanded their service to the edge of the network or even a private user who want to lower the cost for their local cloud. Consequently, a sufficient offloading strategy has to consider the trustworthiness of the fog nodes in the network.
The Last challenge is reliability in a distributed network like fog computing is it complicated to detect individual failures. As well as it is crucial to ensure Packet Reliability to ensure the delivery of every data packet, event reliability which ensures that at least one of the package which the sensor detected gets delivered and End-to-end packet reliability where every packet get retransmitted if its arrival did not get confirmed [5].

1.3 Taxonomy

From all those challenges mentioned above we derived the following taxonomy which is used to classify how sufficient each solution tackle the challenges of fog computing.

Service Delay

Fog computing is targeting delay-sensitive applications. They are several possibilities of delay in fog computing for example data aggregation is not done before processing, resource provisioning. To ensure low service delay a offloading algorithm has to consider all the possible delays like propagation, processing and transmission delay. Furthermore to find the right policy for offloading a task an not processing it locally is crucial for the Quality of Service. All in all to ensure a low latency should be the goal of all offloading solutions.

Integration

Fog computing is build from different devices with different architecture, APIs and resource limits at the edge of the network and builds a distributed heterogeneous network. Therefore the API of the fog nodes are different and it need a more general API which use the existing API's and protocols to make it possible that he nodes can communicate with each other.

Mobility

How react the system if a fog client or fog node is changing his location frequently? Does it update the connection point? Does it recognize new latency caused by a higher or lower distance to other fog nodes? Can a task easily migrate from one fog node to another?

Security and Privacy

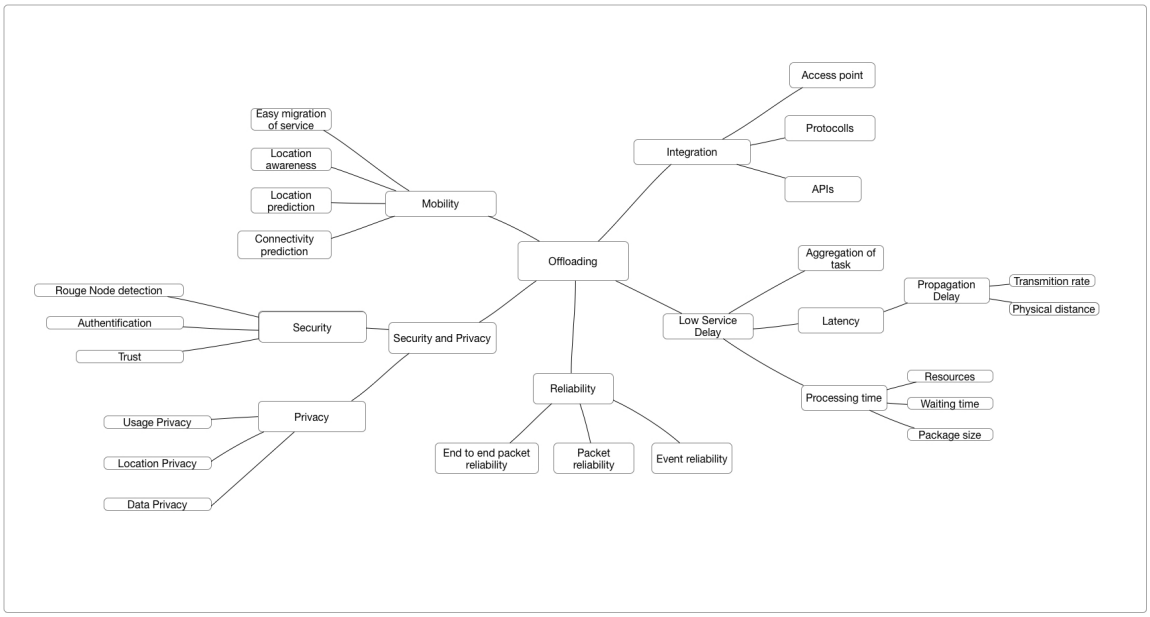

Fig. 1. Graphical presentation of the Taxonomy

One of the most important challenge for fog computing is Security and Privacy and the authentication on the various level of fog nodes is the main challenge to ensure that the every agent in this scenario is trustworthy. The privacy of the data from the user should be consider in offloading as well.

Reliability

Fog computing has a large amount of nodes, high distribution and a more complex interdependencies between the components with that it is difficult to provide a good Fault-Tolerance [3]. As well IoT devices have often higher QoS requirements cause unbounded failures will result in physical effects [3].

2 Approaches

2.1 On Reducing IoT Service Delay via Fog Offloading [11]

The framework of this approach consists of three layers. First the IoT Layer, where IoT and end devices are located, second the fog layer which has the fog nodes in it and the third is the cloud layer. The IoT and end devices can offload a task to the fog layer or the cloud layer. A fog node in the fog layer can decide if he offloads a task to another fog or the cloud. The cloud can process requests and send them back to the end devices or IoT.

Service Delay

The decision to offload a task is given by the probability of task to be offloaded. These probabilities for processing locally, sending the task to the fog layer and sending it to the cloud are given as Input to this model.
To ensure low service delay, the task gets divided into two types. A light type for light processing, for

example, a humidity sensor and a heavy type for heavy processing, for instance, some image recognition task. They have two different interaction modes the first one is with a central node which orchestra the communication and a decentralised node where the fog nodes use a protocol to share their state with their neighbouring nodes. In both modes they have for each node W the estimate waiting time which consists of the waiting time is the estimated processing time of the task in the queue plus the estimated processing delay of the requested service.
The estimate processing delay is composed of the measured processing delay for the heavy and light task. These delays include the propagation and transmission delay. It starts that the IoT device sends the task to the fog node with the lowest propagation delay. Then every fog nodes or the central node computed a reachability table with the round trip delay and W and marked the best neighbour. The estimated waiting time is given by the sum of the heavy and light processing task which are gamma distributed. With that, they compute the probability that the task get accepted.
The decision for offloading is based on the response time of the fog nodes, which is called service delay. Theta is the threshold where a fog node j accept a task if W is smaller than theta. In the case that he did not accept the task, he offloads the task to the next fog node. If the task got E (offload limit) times offloaded or all fog nodes got visited, the task will get offloaded to the cloud.

Mobility

As the location of the fog nodes can change they propose to frequently measure the round-trip delay and update the reachability table. The propagation delay from end device or IoT get not considered because it is not practical as the fog node would to know the delay of the neighbouring fog nodes to the IoT node.

2.2 Scalable Fog Computing with Service Offloading in Bus Networks [9]

The framework for this approach has roadside cloudlets and bus fog servers. The roadside cloudlet has three components: the first component is a dedicated local server which acts as cloud computing, second is the location-based service (LBS) which provide the real-time location of the buses and third the Access Point(AP) which is the gateway for mobile user and the bus fog nodes to communicate with the cloudlets.

Service Delay

The algorithm works as a circle mechanism. A task is composed of the numbers of CPU circles which are required to compute the task, the data size of the task and the tolerance time of the user which correlates with the user experience.
The Problem gets formulated as a linear optimisation problem. It minimize the cost for offloading from the cloudlet, maximize the utility of each bus fog server, ensure the time of computation is smaller than the predicted time in reach of the fog node and the tolerance time of the user, that the data size is smaller than the storage capacity of the fog node and ensure that each computational task is placed at only one fog node. The Total cost of the cloudlet is the price it has to pay the bus servers for offloading plus the transmission delays. The Utility function of the bus server consists of the willingness to receive a set off task, which increases with higher payments of the cloudlet, minus the computational cost for the tasks.
This NP-Hard problem gets solved with the genetic algorithm. Therefore we create solutions randomly when the solutions fit the criteria of the optimisation problem it is an effective solution. To each effective solution, the cloudlet adjusts the unit price to maximise the utility. Then they compute total cost and use that as the fitness of the solution. Select the solution with the highest fitness and create a second generation with crossover and evolutions that repeat X times which is defined before. With a higher x value, the algorithm will find a more optimal solution but will take more time to compute the task.

Mobility

The LBS provides the offloading with the real-time location of the bus. The information on the location,

the speed of the bus and the communication coverage of the AP get used to compute: first the time the bus is in the range of the AP and the total time to solve the task and transmitted it back to ensure that the bus does not leave the coverage range of the AP before the task is finished.

Security and Privacy

The cloudlets and bus fog nodes are public facilities therefore the users do not need to worry about privacy exposure.

Integration

The use of the Access Point (AP) which is used to communicate between the different components give that approach the possibility to extend the fog nodes to a more heterogeneous fog computing network by adding some protocols or API at the AP which let the new fog nodes communicate with the AP as it functions as a middle layer in the offloading.

2.3 Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing [7]

The framework of this approach has three layers as well. The bottom layer contains the IoT devices which send the data to the upper layers. The middle layer contains edge devices and edge data centres (EDC). The topmost layer contains the cloud server.

Service Delay

The End devices offload the task to the nearest EDC. For load balancing all the EDC, it uses three parameters: m the current load, n the maximal capacity and p the current workload computed by p = m/n. If one of the EDC's overloads the EDC will send a control packet with the load information to other EDCs. They first check their load information, and if p is smaller than 0.6, they try to get the available resources. Therefore it checks its resources if they are higher than the required amount it will send the response to the original EDC back if it has not enough resources it will stay silent. The original EDC gets several responses, and it chooses the response with the lowest p and sends the task to the corresponding EDC.

Security and Privacy

First, the precondition is that the cloud is deployed in a secure environment and does not need an authentification. The cloud assigns all EDCs an ID, a key and a shared key after the EDCs got initialised they start to authenticate the EDCs in their region with that they can avoid rogue EDCs later. They create authentication packets and encrypt it which their key and send it to all the other EDCs. The Other EDCs decrypt it with the shared key they got from the cloud and send as well a response encrypted with their key. In the end, all EDCs have authenticated each other. If they communicate with each other, they will encrypt every time with their key, and the other EDCs will decrypt it with the shared key.

2.4 Bandwidth Allocation and Computation Offloading for Service Specific IoT Edge Devices [1]

The IoT devices are located at tier four which are connected to the gateways at tier three. The gateway function as a middleware between tier four and tier two which consist cloudlets and fog nodes and tier two is connected to tier one where the cloud server is.

Service Delay

First, every IoT devices get a Time Sensitivity which represents the urgency for the response and a Data Loss Tolerance which represent the percentage of data of an IoT device which can be sacrificed. The weight w which influences how much bandwidth gets allocated to the IoT device gets computed with the Time Sensitivity and the Data Loss Tolerance. Then they minimise the Optimization $min \sum_{i=1}^{n} \omega_i \times (D_i - R_i)$ where D_i is the rate of Data generation of an IoT device and R_i the transmission rate. It gets optimised with the constraints

of the total bandwidth, specific bandwidth for each IoT device, Buffer Constraints, Time constraint and Energy constraint. The goal of that optimisation is that the IoT devices have a minimum amount of data in the buffer.

Reliability

With the variable Data Loss Tolerance can the system divide between task which is strongly immune against data loss like humidity sensor and task where a data loss can be fatal for example some health sensors. So IoT devices with a lower loss tolerance get more bandwidth allocated and therefore reduce the problem of data loss.

2.5 Computing at the Mobile Edge: Designing Elastic Android Applications for Computation Offloading [6]

Service Delay

For each task they use five different optimisation strategies i) speed up computation, ii) save energy, iii) save bandwidth, iv) low latency and v) use a balance mode. Solving the optimisation problem on a mobile device would require many computational resources. Therefore they use a basic rule set to reduce the numbers of times of full optimisation. If the runtime is less than 30 seconds, it will use the same offloading as the last successful offloading with the same optimisation target. If the runtime is less than five minutes, the offloading get decides with past metrics and a reasonably up to date connectivity prediction. The last rule is if the runtime is longer than 5 minutes it uses a new connectivity prediction and use a well-evaluated offloading strategy.

Mobility

The Context manager uses a generic forecasting process. The Context manager has the following Information: date, time, location, calendar events, sharings, LTE/WiFi/Bluetooth: on/off, bandwidth, ping, signal strength and some reasoned attributes like user at Home/work, travelling, moving and resting to make a connectivity prediction. This connectivity prediction gets then used for the offloading decision.

Security and Privacy

As the request get coordinated by the Coordinator component, the fog node which computes the task has no knowledge to which client the information belongs, this increase the privacy of the client.

Reliability

The Paper denotes that the Coordinator is for error handling responsible. How the error handling will be implemented was not part of the paper.

2.6 Multi-agent and Reinforcement Learning Based Code Offloading in Mobile Fog [2]

Service Delay

The precondition is that the application was executed the first time at the host machine to collect runtime data for the offloading decision. The offloader module handles the offloading, and the network manager gives the information of the availability of fog nodes. If the task is smaller than the resources at the host machine, the task gets processed locally. In case it can not it will compute the local processing time and compare that with the expected execution time of the fog nodes while it starts to process the block locally. The code compiler performs Bread First Search on a flowchart to find independent blocks to offload them in the fog for parallel execution. The System has different agents who perform a specific task, for example, offload a task from IoT device to the fog layer or migrate the task from one layer to another. The agent of the fog node checks their state of memory, CPU and bandwidth as well as the queuing status.
Agents build a group to compete with other agent groups. Therefore they compute their expected response time with the help of the M/M/1/K queuing model. The leader of the agent group evaluates these response times and creates a bid for the task. The best bid got the task and solved it if it the response time is lower than the actual bid it reduces the estimate for the next round if the response time is higher

it adjust it estimated response time.
Not functional groups of agents get dissolved after a task is finished and for the new task randomly put together to new groups. For that reason, the system gets better with each task it solved.

Integration

The use of an Access Point (AP) as a middle layer of communication makes adding of new fog nodes with new API easier as the change need just to be implemented at the AP.

Security and Privacy

The inter Fog communication and the local authentication of a mobile station is the evolved packet data gateway (ePDG) responsible. The 3GP AAA is responsible for the global authentication of a mobile station with the Extensible authentication protocol-authentication and key agreement (EAP-AKA) over Internet key exchange protocol version 2 (IKEv2).

2.7 QoS-aware Dynamic Fog Service Provisioning [12]

The primary goal of the dynamic fog computing proposed in the paper is to provide a better QoS for the client by defining a maximum delay threshold which should be fulfilled. Therefore the client makes an SLA with the Cloud Service Provider on the QoS.

Service Delay

In this approach, they use two different metrics for the optimisation. The first Metric is Min-Viol where the goal is to deploy services on fog node with a high traffic rate, release services on services with a low traffic rate and at the same time have low service violations. Therefore a traffic monitor observed the traffic rate of the for nodes and sorted them in that order. As long as the violations are in the bound of the SLA, it deploys services on the fog nodes. A response will violate the SLA if the service delay is higher as the agreed threshold.
The Service delay is composed of the average propagation delay, the average transmission delay and the waiting time. They state that average value is acceptable as fog nodes are placed near to IoT device, and the connectivity values of different IoT devices to one fog node do not differ much.
If the predicted service delay violates the SLA, the loop will be terminated. A second loop test if at fog nodes with low traffic rate service can be released without violating the SLA. In case it cannot the loop terminate as a node which a higher traffic rate will cause more SLA violations.
The second metric is Min-Cost. The goal of that metric is to minimise the cost. The fog nodes get sort in the same way as the Min-Viol metric. Then it is iterate through the list of fog nodes an checked if deploying a service minimise the cost. Cost can be saved if the cost of communication, the cost of storage and processing and the cost of possible penalties for SLA violations decreased.
The cost for communication of a fog node is formed from the communication cost per bandwidth, the traffic rate of the fog node, the average size of the request package and the time interval for solving the optimisation problem. The communication cost for the cloud calculates the same way just with the average size of the response packages get to add to the formel.
The Cost for processing for the cloud and a fog node are calculated with the processing cost per million instructions, required amount of processing of service a per request and the traffic rate.
The Cost for storage is formulated by the unit cost for storage and the required amount of memory for a service.
In the same way, as in the min-viol algorithm, it iterates through the list of fog nodes and starts with the fog nodes with the least amount of traffic. There it checks if realising the service would reduce the cost.

2.8 Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game [8]

In this approach, the goal is not an overall performance optimisation, but a QoE optimisation for each user as they are selfish. Therefore they use a game theoretical approach to determine the best offloading

strategy for IoT devices under competition.

Service Delay

First, the IoT device determined their local cost for computing that task. Therefore it uses the size of the task and processing density divided by the processing power of the IoT device and multiplies with constant weights for computation energy and computation time. In case they need to offload a task they compute the processing time for the task. A Fog node gives all task an equal share of resources. The cost to offload a task is composed out of the energy cost for that task on the fog node and the cost for the time. The cost for the time results out of the propagation delay, the round trip delay and the time for the computation. The QoE is defined by the difference between the offloading cost and local computation cost. The goal is to find the best response strategy for a service for the strategies of the other service to have a maximum QoE. This algorithm converges if the best response for a service does not change and a Nash equilibrium is reached.
As an alternative to the best response algorithm, they propose a better response algorithm. Therefore an e value gets defined. A player changes its strategy if the benefit of the strategy change is higher than e. The paper shows that such e-nash equilibrium is in a polynomial time complexity reachable.

3 Summary

On Reducing IoT Service Delay via Fog Offloading

With the reachable table where the latency to other fog nodes and their best neighbour as well as that the defined cutoff where it is better to offload a task to the cloud and not offload it to another fog not which prevent a to a long search for offloading if every fog node is full. Moreover, with the frequent updating of the reachable table, it ensures that the delay times are reasonable up to date for an adequate offloading decision. Problematic is the classification of two processing types which have average processing, transmission and propagation delay. The Extendibility is in the centralised mode better as there is just one node which needs to implement new communication protocols. If a different type of fog node gets added to the fog, but in the decentralised mode every fog node would need the new protocol but there is no central unit which could distribute this protocol, and no mechanism is designed for sharing communication protocols or APIs. The point of reliability and Security and Privacy is not tackled in that approach.

Scalable Fog Computing with Service Offloading in Bus Networks

The reducing of latency to the point of the tolerance of user is a very efficient way. As a result, tasks which need low latency will be in that way more prioritised than task where the tolerance for the latency is higher. As a result, the task which requires low latency has a higher availability of resources. As the LBS provides the cloudlet with the real-time location and their AP gives a connection range they can prevent for example that a bus leaves the connection radius before finishing the offloading. The AP gives an excellent opportunity to extend the fog network, but the claim that user does not need to worry for privacy exposure is not right as the approach does not describe how it will secure the data from malicious attacks.

Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing

The decision to offload it based on the workload of the fog nodes. That is not optimal if the goal is low service delay as the fog node with the lowest workload does not have necessarily the lowest service delay. Nevertheless, the load balancing approach has shown that it is capable of reducing the service delays in the tests conducted in the paper. The authentication of fog nodes with the help of a cloud server is a practical solution to prevent registration of a rogue fog node. Moreover, the use of encryption help for the man in the middle attacks.

Bandwidth Allocation and Computation Offloading for Service Specific IoT Edge Devices

The approach concentrated on the allocation on the bandwidth for the IoT Devices. However, did not consider the different resources and location of fog node for an optimal offloading decision. As a result, the offloading cannot be optimal. Nevertheless, the model can be helpful to integrate a different bandwidth allocation for offloading in a different model.

Computing at the Mobile Edge: Designing Elastic Android Applications for Computation Offloading

Solving of a linear optimisation problem can be a very resource intensive task. The reusing of solutions is a way of limiting, but it can lead to an overload at one fog nodes if a significant amount of 30 seconds tasks get added as the last optimal solution get used. The use of the Location awareness plus several other data to predict the connectivity in the future is an excellent way to ensure in the case that end devices move that the connectivity is considered.

Multi-agent and Reinforcement Learning Based Code Offloading in Mobile Fog

The exciting part of that approach is the separation of a task in independent blocks, which give the system more possibilities to optimise the service delay in the way that they parallelised the computation of the blocks. They as well consider that migration of task to different fog nodes can boost the overall system performance. With their security concept, they ensure that just authenticated fog node is part of the network.

QoS-aware Dynamic Fog Service Provisioning

That approach reflects very well that different services need different requirements. As the QoS is defined beforehand, the optimisation can be concentrate on the goal to fulfil the SLAs. It gives the system the possibility to give IoT devices a higher priority if it has a stricter SLA. As different IoT device need a different level of QoS, it is beneficial for the system if this is considered at the offloading.

Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game

The game theoretical approach has shown that an optimal solution for each IoT service which is selfish is not much less optimal than an optimal solution for the whole system. This approach delivers a solution for a situation where several services want to offload a task at the same time, but it did not consider that tasks can arrive at different times. Nevertheless, can this algorithm repeat for every new incoming task but therefore the migration of task to different fog nodes should be considered.

Approach	Metric	Result	Service Delay	Integration	Mobility	Security and Privacy	Reliability	Strength	Weakness
On Reducing IoT Service Delay via Fog Offloading	Queuing status	the proposed offloading policies reduce the service delay. Especially if the rate of heavy request is high	considered all the different delays in the offloading decision	medium	medium	not considered	not considered	Easy algorithmen for offloading in the fog layer	do not tackle the problem when a task get offloaded to the fog layer
Scalable Fog Computing with Service Offloading in Bus Networks	Delay and Processing cost	with more bus fog server the cost for the cloudlet fell and the percentage of task which did not satisfy the user experience fell as well	sufficient consideration of service delay	medium	high	low	not considered	considering of mobile fog nodes	is limited on the radius of the bus tracks
Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing	Load Balancing	shows that they proposed solution has a better response time than other load balancing solutions	does not optimize for low service delay	low	low	high	not consideered	Security	Goal is to balance the load and not the service delay
Bandwidth Allocation and Computation Offloading for Service Specific IoT Edge Devices	Bandwidth	shows that there solution BACOFF provide a smaller service delay and less lost packages than CORAL	does not optimize for an low service delay	low	low	not considered	low	task with higher priority get more bandwidth	does not consider the service delay
Designing Elastic Android Applications for Computation Offloading	Computation	Present a architecture but did not provide an evaluation.	does not compute best offloading decision for every task	medium	high	medium	low	considered connectivity and movements of users in the future	limited use
Multi-agent and Reinforcement Learning	Service Delay	shows that they have a smaller delay than ThinkAir	optimize for the lowest service delay	low	low	high	not considered	A self learning system	needs time to learn the best offloading solution
QoS-aware Dynamic Fog Service Provisioning	QoS	shows that the proposed solution minimize cost and reduce QoS violation	optimize to deliver negotiated service delay	low	low	not consodered	not considered	focus of different QoS needs from services	no consideration of Security and Privacy
A Computation Offloading Game	Service Delay	If user optimise their own service delay it is not a big difference to a optimized solution for the whole system		user tries to minimise his own service delay	low	not considered	not considered	Model of selfish user	does not consider task which arrive at diffrerent times

3.1 Prospect

The comparison of all of these approaches shows that the topic Security and Privacy and Reliability are often not tackled or insufficient addressed. For a fog computing system to be practical in the real world, these two points are crucial to the success. For example, Data Protection Laws need to be fulfilled, or systems cannot accept the loss of critical pieces of information. Several papers identify these crucial problems and propose an abstract solution but yet to fail to integrate them in an actual fog computing offloading approach. For example, for reliability, it gets proposed to use the reliability approaches of Wireless Sensor Networks (WSN), yet no approach consider that transmitting of a task or the answer can fail and the packet gets lost on the way or even that the sensor data are false. Therefore a more comprehensive solution for offloading needs first to fulfill the three points Service Delay, Integration, Location Awareness which state of the art approaches already do reasonably and on that basis add adequate security and privacy and reliability concepts in tradeoff with low latency because those concepts will create more significant overhead.

References

1. Al-Zihad, M., Akash, S.A., Adhikary, T., Razzaque, M.A.: Bandwidth allocation and computation offloading for service specific iot edge devices. In: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). pp. 516–519 (Dec 2017)
2. Alam, M.G.R., Tun, Y.K., Hong, C.S.: Multi-agent and reinforcement learning based code offloading in mobile fog. In: 2016 International Conference on Information Networking (ICOIN). pp. 285–290 (Jan 2016)
3. Bermbach, D., Pallas, F., Pérez, D.G., Plebani, P., Anderson, M., Kat, R., Tai, S.: A research perspective on fog computing. In: International Conference on Service-Oriented Computing. pp. 198–210. Springer (2017)
4. Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., Chen, S.: Vehicular fog computing: A viewpoint of vehicles as the infrastructures. IEEE Transactions on Vehicular Technology 65(6), 3860–3873 (June 2016)
5. Madsen, H., Burtschy, B., Albeanu, G., Popentiu-Vladicescu, F.: Reliability in the utility computing era: Towards reliable fog computing. In: 2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP). pp. 43–46 (July 2013)
6. Orsini, G., Bade, D., Lamersdorf, W.: Computing at the mobile edge: Designing elastic android applications for computation offloading. In: 2015 8th IFIP Wireless and Mobile Networking Conference (WMNC). pp. 112–119 (Oct 2015)
7. Puthal, D., Obaidat, M.S., Nanda, P., Prasad, M., Mohanty, S.P., Zomaya, A.Y.: Secure and sustainable load balancing of edge data centers in fog computing. IEEE Communications Magazine 56(5), 60–65 (May 2018)
8. Shah-Mansouri, H., Wong, V.W.S.: Hierarchical fog-cloud computing for iot systems: A computation offloading game. CoRR abs/1710.06089 (2017), http://arxiv.org/abs/1710.06089
9. Ye, D., Wu, M., Tang, S., Yu, R.: Scalable fog computing with service offloading in bus networks. In: 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). pp. 247–251 (June 2016)
10. Yi, S., Hao, Z., Qin, Z., Li, Q.: Fog computing: Platform and applications. In: 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). pp. 73–78 (Nov 2015)
11. Yousefpour, A., Ishigaki, G., Gour, R., Jue, J.P.: On reducing iot service delay via fog offloading. IEEE Internet of Things Journal 5(2), 998–1010 (April 2018)
12. Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H.C., Zhang, Q., Xie, W., Jue, J.P.: Qos-aware dynamic fog service provisioning. CoRR abs/1802.00800 (2018), http://arxiv.org/abs/1802.00800