Applied Computer Science, vol. 16, no. 4, pp. 5–20
doi:10.23743/acs-2020-25

Submitted: 2020-11-29
Revised: 2020-12-07
Accepted: 2020-12-17

software-defined networks, fog computing, smart sensors, IoT

Ihor PYSMENNYI [0000-0001-7648-2593]*,
Anatolii PETRENKO [0000-0001-6712-7792]*,
Roman KYSLYI [0000-0002-8290-9917]*

GRAPH-BASED FOG COMPUTING NETWORK MODEL

Abstract

IoT networks generate numerous amounts of data that is then transferred to the cloud for processing. Transferring data cleansing and parts of calculations towards these edge-level networks improves system's, latency, energy consumption, network bandwidth and computational resources utilization, fault tolerance and thus operational costs. On the other hand, these fog nodes are resource-constrained, have extremely distributed and heterogeneous nature, lack horizontal scalability, and, thus, the vanilla SOA approach is not applicable to them. Utilization of Software Defined Network (SDN) with task distribution capabilities advocated in this paper addresses these issues. Suggested framework may utilize various routing and data distribution algorithms allowing to build flexible system most relevant for particular use-case. Advocated architecture was evaluated in agent-based simulation environment and proved its' feasibility and performance gains compared to conventional event-stream approach

1. INTRODUCTION

Introduction of high-bandwidth mobile networks, portable sensors and actuators, energy-efficient microprocessors and communication protocols enabled wide adoption of Internet of Things (IoT) – an adaptive network connecting real-world things (including sensors which perceive the environment and actuators which may actively interact with it) between each other and the internet. Being lightweight

* National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Institute of Applied Systems Analysis, Department of System Design, 37, Peremohy ave., Kyiv, Ukraine, ihor.pismennyy@gmail.com, tolja.petrenko@gmail.com, kvrware@gmail.com

and portable the majority of these devices are resource-constrained in terms of computing and networking power by definition meaning the need for intermediate computing layer between them and the cloud (Rahmani, Liljeberg, Preden & Jantsch, 2018).

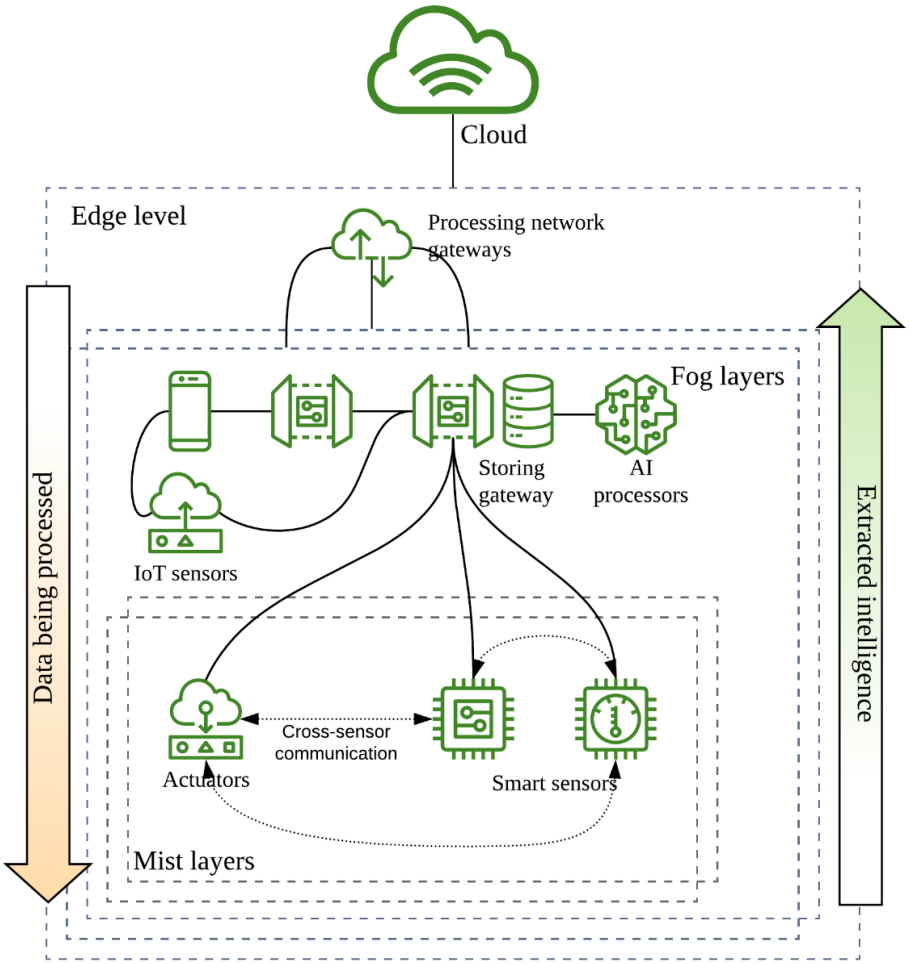

Fig. 1. Edge computing

As discussed in academia, this intermediate layer may utilize 3 complimentary approaches (Fig. 1):

1. Mist Computing. Computations are being performed on the extreme edge of the network – smart sensors and actuators themselves. Only pre-processed data is sent via network and IoT devices are not dependent on Internet connection (Yogi, Sekhar & Kumar, 2017).

2. Fog Computing. Layer close to the perception layer with computing, networking (aka gateway) and storage capabilities (Rahmani et al., 2018). Spans from the data creation point to its storage location allowing decentralized computing of gathered data. Any device with support of required network technologies, storage and networking capabilities can be utilized as fog a fog node (Joshi, n.d.). Can be considered as a superset among the mist layer.
3. Edge computing. (Satyanarayanan, 2017) defines any computing and network resource between data source and destination data centre (either cloud or local) as edge computing node. Further in this article edge-computing will be used as an umbrella term for all 3 levels.

Transferring part of data processing from cloud to edge level puts a lot of resource-related constraints such as computing power restrictions, absence of dynamic horizontal scalability and energy consumption limitations, but brings following benefits:

1. Location-awareness. Edge computing systems are aware of context in which computations are being performed
2. Latency reduction. Classical cloud-computing approach with aggregating data on *smart hub*, batch or stream sending it to cloud for processing and retrieving results back in synchronous or async manner introduces 2 pitfalls critical to real-time applications: network latency and possible network failure. Moving data processing to the edge can help to tackle these problems as discussed below and already found usage in various systems including
3. Security. Any data transmission is subject to man in the middle attacks and data protection requires energy-consuming encryption algorithms (Petrenko, Kyslyi & Pysmennyi, 2018a).
4. Eliminating bandwidth restrictions. Some data, especially media, require a high-bandwidth communication channel. By processing it on the edge level we eliminate the need for this expensive transfer. For example, smart doorbells which unlock the door using face identification tech may process video stream locally instead of sending it to cloud.
5. Energy consumption reduction. Data transfer is significantly more expensive in terms of energy consumption than basic processing. (Shi, Cao, Zhang, Li & Xu, 2016) hence offloading cleansing, aggregation and extraction operations to mist and fog layers may increase the time of autonomous work of IoT device as discussed in chapter 4.
6. Cost reduction. Edge computing helps utilize maximum of available resources resulting in improved cost-efficiency.

Modern edge computing use-cases and architectures are discussed in the next chapter with novel graph-based edge network architecture and its evaluation following.

2. LITERATURE REVIEW AND PROBLEM STATEMENT

Shifting distributed calculations paradigm from remote cloud to network edge is a complex task and involves solving novel architectural problems which can be grouped to resource constraints, communicational, privacy and security and fault tolerance domains. Further in this chapter state-of-the-art findings in the field are discussed.

(Ray, 2018) surveys and structures use-cases, technologies, and domains of modern IoT applications. Authors emphasize on technical challenges of designing service-oriented architecture of extremely heterogenous horizontally wide system, its' infrastructure and security requirements.

(Rahmani et al., 2018) provides a comprehensive high-level overview on most edge computing aspects with and various use-cases. Authors analyse architectural constraints, essential components, and management of fog-layer computational infrastructure. Edge computing has used in most parts of modern infrastructure from smart cities to medicine. In automotive cars already have sophisticated sets of various sensor and data processing systems allowing semi-autonomous driving (Hussain & Zeadally, 2019). Xiao & Zhu (2017) suggests using smart vehicles as moving fog nodes allowing on-demand computational resource distribution and expanding vehicle to vehicle (V2V) communications. These fog nodes may be connected to city infrastructure such as smart traffic lights allowing more efficient traffic distribution (Stojmenovic & Wen, 2014).

Chan, Estève, Escriba & Campo (2008) wraps a review of smart-home systems including permanent patient monitoring capabilities usually built on smart sensor networks. Gope & Hwang (2016) suggests Body Sensor Networks (BSN) architecture for distributed edge-level computations with regard to user's privacy. Data acquired from these wearable networks can be processed by deep convolutional neural networks on fog nodes for immediate anomaly detection (Petrenko, Kyslyi, & Pysmennyi, 2018b).

In use-cases mentioned above, critical security concerns are raised as health data is classified as sensitive (World Health Organization, 2010). In (Al Ameen, Liu, & Kwak, 2012) authors analyse privacy and security issues and requirements in regard to wireless sensor (WSN) and body area (WBAN) networks suggesting measures to cope with them. (Diogenes, 2017) suggests utilization of generic zoned approach with security boundaries for privacy preservation. Each of the *device zone, field gateway zone, cloud gateway zone, cloud services zone and remote users' zone* operates on constrained scope of user's data and has security requirements most suitable to given context, sensitivity of data being processed and persistence requirements. (Petrenko et al., 2018b) takes the idea further to cloud level allowing secure multi-party computations between akin organizations with help of hyperledger.

With big amount of open-source and proprietary communication solutions development of edge network on heterogeneous hardware is restrained by the absence of wide adopted standard open messaging protocol specification for exchanging and processing structured data. In (Kharchenko & Beznosyk, 2018) authors concentrate on building a unified data-flow format for various IoT devices suggesting JSON-based format for data and processing description. This approach enables proper distribution of computational resources and is essential for edge level information processing systems.

With absence of permanent remote monitoring capabilities concept of *self-awareness* become particularly important. According to (Rahmani et al., 2018) self-aware system has following capabilities:

1. Understands its current context and evaluates it to the desired state of the environment.
2. Understands its own state, monitors it to detect possible malfunctions and deviations.
3. Input data-aware – tracks its changes over history, performs semantic attribution and interpretation mapping properties to desirability scale.

(Lewis, Platzner, Rinner, Tørresen & Yao, 2016) introduces notion of *collective self-aware systems* where there is no central 'awareness' node emphasizing increased robustness and adaptability brought up by decentralized approach. Classification of awareness levels is also defined in book mentioned above. According to it, edge layer belongs to interaction-aware systems group as it acquires meaningful data from stimuli acting upon it and interactions with other systems, environment, and itself.

IEEE has recently adopted reference architecture for fog computing implementation for both hardware and software (IEEE Communications Society, 2018). Each fog node is operated by software backplane layer orchestrating internal (thing-to-fog) and external (fog-to-fog, fog-to-cloud) communications via service discovery, state management and pub-sub mechanisms, additionally enforcing authentication and system integrity.

Due to extremely distributed and heterogenous nature fog networks and absence of dynamic horizontal scalability vanilla service-oriented software architecture approach doesn't work, thus some ideas may be refurbished. (Oma, Nakamura, & Duolikun, 2019) advocates a fault-tolerant tree-based fog network model. Each node calculates input from data obtained from one or more child nodes with sensors being leaves. System has process-transfer capabilities for fault-recovery and tree balancing to support dynamic network topology. This approach disregards variety of computational power between different nodes and certain elements of the system may become overloaded and create bottlenecks due to the hierarchical topology. Another disadvantage is the need for each node to know its downstream network and ancestors for the recovery mechanism effectively meaning need to persist the whole network structure in each of its element.

In addition to security and storage overhead critical for lightweight edge computing this approach requires frequent propagation of changes to the network topology unacceptable for dynamic fog networks. In further research authors suggested dynamic network-based fog computing (DNFC) model with auction method used to determine set of source nodes by each target node. Authors suggest broadcasting request to compute data from source node to possible fog computing targets, if target has enough energy to process given block or its part, target sends back acknowledgement, performs calculation and sends result upstream the network. This approach has clear advantage in case if all nodes are directly connected but does not support heterogeneous multi-layered smart sensor networks and does not regard potential overflow of communicational capacity of fog nodes.

In this paper we focus on development and evaluation of novel Graph-Based Fog Computing Network Model (GBFCNM) aimed to cope with issues mentioned above.

3. THE AIM AND OBJECTIVES OF THE STUDY

The purpose of this research is to analyse edge computational systems' architecture and propose a lightweight and flexible approach for distributing data processing among fog nodes. Given approach should support following capabilities:

1. Fault-tolerance – fog nodes may accidentally become inactive due to various reasons, for instance, low power supply, loss of communication channel and environmental situation.
2. Malfunction detection – fog nodes should be able to detect faulty behaviour of other node and have a recovery strategy for such cases.
3. Energy-efficiency – smart sensors are frequently designed wearable form-factor and thus have limited power supply.

Anylogic simulation environment is used to evaluate suggested architecture.

4. SMART-SENSOR DATA STREAM PROCESSING ARCHITECTURE

Traditional approach for processing data stream, including IoT generated values streams, is lambda-architecture (Fig. 2), which unifies real-time and historical batch analyses within the single framework (Marz & Warren, 2015). Data streams are ingested to message queue (or other data source) and then:

1. Processed via speed processing layer. Results are provided in real-time in synchronous (via responses to published API calls) or asynchronous manner (via exposing dedicated read API).

2. Batch-processing layer: raw data is being stored in Data Lake and then processed and stored to some sort of data warehouse to be analysed by batch jobs on schedule or on-demand.

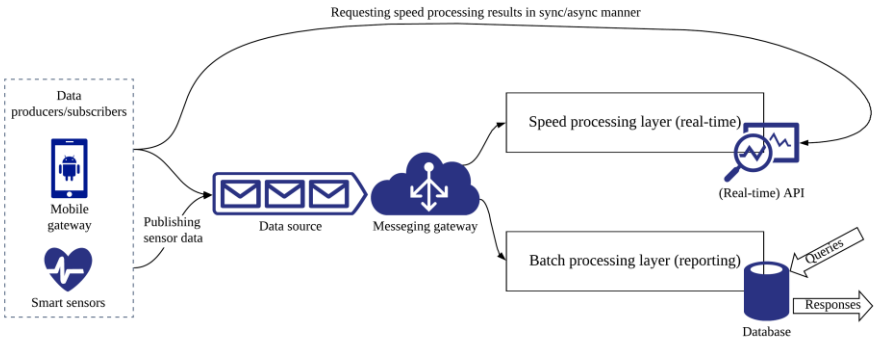

Fig. 2. Processing smart sensor data in conventional lambda architecture

Introducing edge computing brings following modifications to given data flow (Fig. 3):

1. Speed processing layer is moved from cloud to the network edge. Processing results may also be transferred to cloud for persistence. In addition to latency, load and security benefits this approach significantly reduces operational costs eliminating need for always-on cloud operation which allows using spot instances with dynamic pricing(“Spot Instances – Amazon Elastic Compute Cloud,” n.d.) and less strict fault-tolerance requirements which may result in abolishing infrastructure redundancy.
2. Data is transferred to cloud in pre-processed state eliminating the need for data lake and significantly reducing batch processing layer’s load by definition.

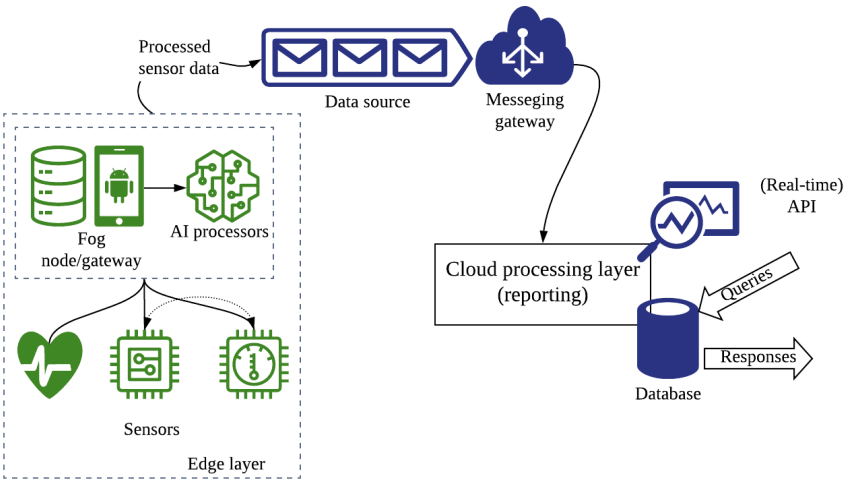

Fig. 3. Edge-optimized lambda architecture

Fog-layer is very heterogeneous as nodes may vary in terms of computational power and connectivity, thus need for dynamic and fault-tolerant computing model emerge. Proposed model is based on Software Defined Networking (SDN) principle: data flows within the network are performed and configured via the use of standardized application programming interfaces (API) (Kirkpatrick, 2013). By definition, GBFCNM is aimed to effectively process generated data. For the sake of simplicity let us assume processing time as the only effectiveness criteria and divide components of edge network to 3 groups based on their capabilities:

1. Smart sensor/actuator:
   - Emitting data and performing interactions with environment.
   - Internal (inside fog network), may be not directly connected to internet.
   - Low computational power as a trade-off for size and energy efficiency.
2. Fog node:
   - Internal traffic.
   - High computational power.
3. Gateway:
   - Accepts and redirects internal traffic.
   - Transmitting data to the cloud.
   - May also have high computational power.
4. Cloud – external component, destination of data.

Most edge-node devices utilize wireless radio network protocols including Bluetooth Low Energy, NFC, 6LoWPAN, Wi-Fi, ZigBee, and RFID, which have limited connection range. Therefore, each node N_x is assumed as connected to node N (belonging to its group G_N) if the distance between them is smaller than defined Δ:

$$N_x \in \{G_N\}: d(N, N_x) < \Delta \quad (1)$$

Proximity-based network partitioning is already adopted in various distributed event-based systems. Approaches with assigned brokers and dynamic plane are utilized (Castro-Jul et al., 2017). Fog network architecture suggested in this paper is classified as Distributed Control WSN as no central routing decision points are present and all nodes exchange information using dynamic data flow defined by this information itself and available processing capabilities (Mostafaei & Menth, 2018).

Each node advertises a set of available resource for each capability. As nodes are heterogeneous in terms of their capabilities it makes sense to have the possibility to transfer tasks between them (given assumption will be evaluated in following chapter), therefore fog node should also expose capabilities of its connections. As all network communications require time, computational resources, and energy, it is proposed to use penalty coefficient μ for each data transfer. In addition, this would limit number of task split (mapping) operations. So each fog node advertises following list of resources where N_i is identifier of resource owning node ($i = 0$ for current node, $i > 0$ for each connected node), Cap_j is capability category (e. g. "outbound traffic, kb/sec") and K_i is capacity of specified category available (e. g. 256 kb/sec):

$$\left[\{N_i, Cap_j, \alpha * K_i\}, \dots\right], \alpha = \begin{cases} 1, i = 0 \\ \mu, i \neq 0 \end{cases} \quad (2)$$

With this scenario it is easy to see that that after some time, when each service will send all advertisements (O(n^2) complexity where n is number of network nodes) with each request all current network topology will be sent, which is unfeasible in big multi-layered fog networks. To avoid this situation, it is proposed that each node advertises resources with maximum relative depth D. Further in this paper coefficient D=1 for the sake of simplicity (Fig. 4).

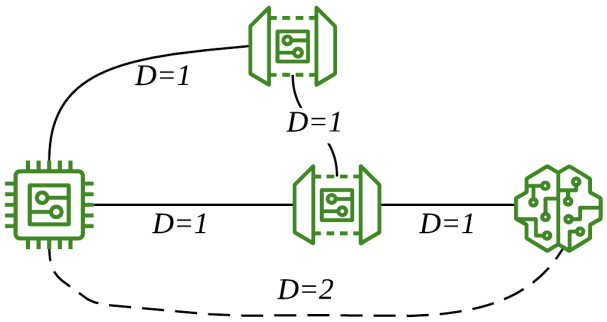

Fig. 4. Example fog network topology

Advertiser sends its' capabilities data at fixed schedule. Task owner node (source) receives set of capabilities from connected advertisers and selects target from recent targets (resources no older than given threshold τ) using multi-objective optimization methods (Fig. 6). Pareto efficiency with value weights calculated dynamically for each request based on task distribution (calculation or transfer) within it is used in this paper. If Pareto frontier consists of more than one element – random option is chosen. Timeout threshold minimizes connection attempts to no longer available nodes.

Suppose node N_0 requires computations in set of categories Cap (e. g. processing, cloud transfer) and has set of available resources from itself R_0. R_0^{Cap} is set of available resources in required categories. To process the required data N_0 received "offers" from nodes $N_i \dots N_j$ with corresponding available resources $R_i \dots R_j$ within τ time window. As N_0 is also a fog node it also advertised its capabilities, so $R_0 \in R_{i..j}$:

$$W(N) = \left(\sum_{Cap} \sum_{1..n} K\right) - \sum_{Cap} K0, \tag{3}$$

where N is number of connections (may be unique for each node).

Once data is processed it is being delivered to closest node with cloud-sending capabilities. Feasible path of distinct nodes towards the output node is called admissible path and in conventional SDNs is performed via SDN controller (Agarwal, Kodialam & Lakshman, 2013). As for networks are dynamic and decentralized, this node is selected via eager path searching algorithm (Sedgewick & Wayne, 2011). This approach requires trail of visited nodes' unique identifiers to be sent along with each request.

Due to the dynamic nature of edge-level networks their fragmentation is very possible, different network parts may become isolated and unable to transmit data further (Fig. 5).

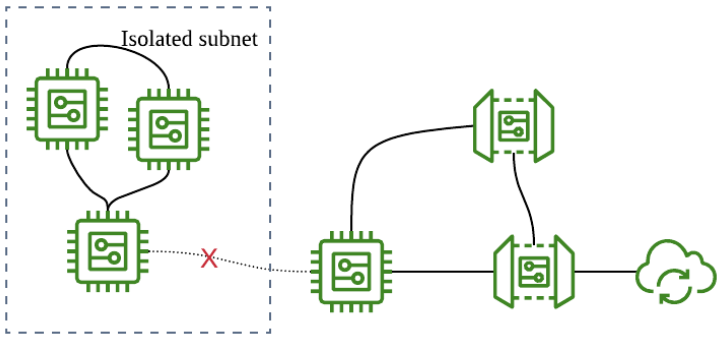

Fig. 5. Network fragmentation in fog computing

To overcome such situation asynchronous acknowledgement downstream is suggested (Fig. 6). Each node with storage capabilities should cache processed data stream. Node may erase cache only once it received acknowledgement that this data was processed upstream and after forwards it downstream. If this acknowledgement was not received within given timeout retry mechanism should be implemented.

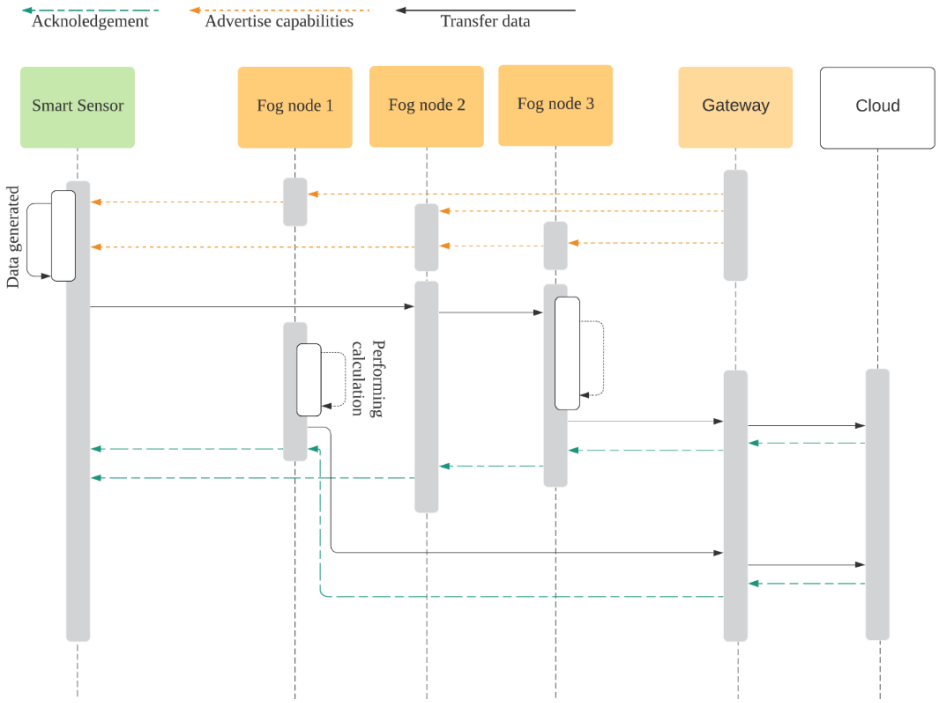

Fig. 6. GBFCNM happy flow

5. GBFCNM PERFORMANCE EVALUATION

IoT consist of the wide variety of interconnected devices with different power and capabilities. Testing and evaluating large-scale configurations of heterogeneous adaptive networks is almost unfeasible with conventional approaches, therefore utilization of multi-agent-based computing is suggested (Laghari & Niazi, 2016).

Each element of the system as well as the environment is modelled as an agent with dedicated behavioural and communicational strategy and capabilities creating system's digital representation in simulated environment (Klügl & Bazzan, 2012). Such approach allows us to evaluate behavioural strategies in various scenarios, identify their potential pitfalls, and determine optimal policies which will be implemented in end-product for different network configurations. There a lot of simulation toolkits available among which AnyLogic demonstrates significant usage growth dynamics and maintains vivid community (Dias, Vieira, Pereira & Oliveira, 2016). It allows combining agent based, discrete event and system dynamics simulations to single multi-method model. GBFCNM architecture was evaluated in Anylogic agent-based simulation environment (Fig. 7). Each actor (smart sensor, fog node, gateways and remote cloud) has dedicated set of state charts, behaviours and parameters which may dynamically change over time ("Multimethod Simulation Modeling for Business Applications – AnyLogic Simulation Software," n.d.). The goal of simulated system is to process and transfer data streams from smart sensors to cloud via fog and gateway nodes. Each node type has unique combination of capabilities as described above in system architecture chapter.

In (Table 1) conventional IoT setup (computations being performed in cloud only) with 4 smart sensors, 2 fog computing nodes acting as internal traffic routers and single gateway is evaluated against same setup with edge-computing capabilities. Each smart sensor emits N packets per second, which may be processed and forwarded by fog nodes reducing amount to be sent by F%. Processing rate of such node is K packets per second. Gateway is able to send to cloud L packets per second.

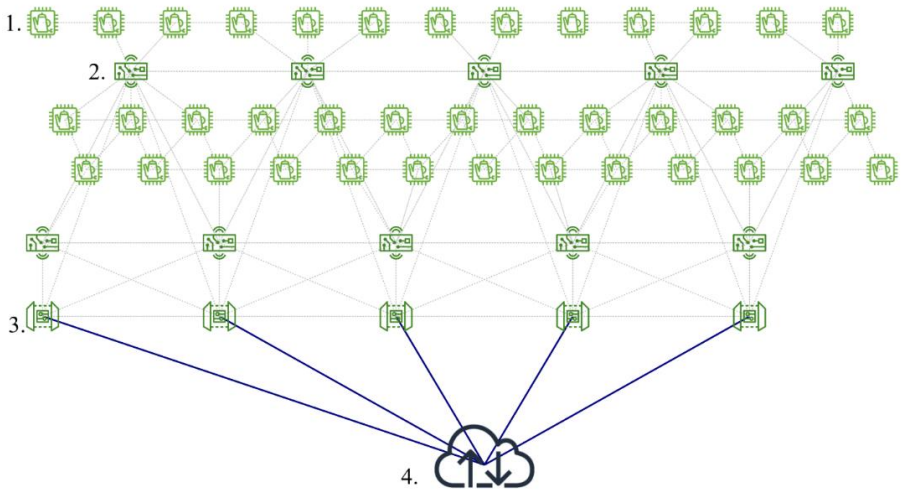

Fig. 7. Screenshot of simulated fog network consisting of smart sensors (1), fog nodes (2), gateways (3) and remote cloud (4)

6. DISCUSSION

The downside of given approach is the need to advertise capability information on connected nodes and append, which consumes network resources of the system. Possible solution is to add threshold after which resource is considered too weak and is no longer advertised. Developing strategies of route caching will allow decreasing configuration sharing frequency.

In addition, utilization of generic frontier search algorithm for routing may significantly increase message size in large distributed systems as path is sent together with message. Introduction of dedicated control nodes forming indirectly (via cross-sensor multi-hop communications) connected SDN controller may help solving this issue and is a subject for further research (Mostafaei & Menth, 2018).

Possible loss of data if some subset of nodes become unavailable due to dynamic nature of the fog network is tackled with its' caching on intermediate nodes. Fog network nodes have limited storage capabilities (for example, most STM32 boards come with 32 KB to 1MB flash memory), therefore utilization of data compression techniques is suggested. In (Pysmennyi, Kyslyi, & Petrenko, 2019) authors advocate using moving average on the oldest stored data windows, so most up-to-date information will still have highest possible resolution:

$$CMA_n = \frac{x_1+\cdots+x_n}{n};\ CMA_{n+1} = \frac{x_{n+1}+n*CMA_n}{n+1}. \quad (4)$$

Suggested approach also allows to tackle spikes of volume of generated data if it exceeds network bandwidth.

Another issue of suggested algorithm is gradient load of computational resources – the further they are from data emitting smart sensor the less they would be loaded due to data transfer penalties. With generic fog network where sensors are distributed evenly with fog nodes this will not pose a problem, but in case of network with homogenous resource clusters this approach requires adjustments which are subject for the further research.

7. CONCLUSIONS AND FUTURE WORK

As shown in this paper, SDN paradigm perfectly fits fog computing networking tasks. Utilization of edge nodes for computations clearly showed advantages in decreasing load of cloud system, improved resource utilization on the edge level and enabled fast feedback to actuator nodes. Power, computational and bandwidth constraints combined with narrow scope of capabilities (specialization of nodes) introduce the need for flexible framework for distributed processing and routing of data suggested in this paper.

As shown in simulation above, given approach results in significant reduction of bandwidth and computational load to cloud infrastructure and improves overall system efficiency.

Possible areas of future research include but are not limited to:

1. Adaptive selection of most suitable mesh networking algorithm. Given challenge implicitly requires enabling monitoring of low-powered distributed system.
2. Introduction of SDN control plane to the system for improvement of network efficiency.
3. Combining routing with processing task mapping for increasing load distribution efficiency.
4. Faulty and malicious nodes detection. Evaluation of using AI-empowered fog nodes for this purpose.

REFERENCES

Agarwal, S., Kodialam, M., & Lakshman, T. V. (2013). Traffic engineering in software defined networks. *2013 Proceedings IEEE INFOCOM*, 2211–2219. https://doi.org/10.1109/INFCOM.2013.6567024

Al Ameen, M., Liu, J., & Kwak, K. (2012). Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of Medical Systems*, *36*(1), 93–101. https://doi.org/10.1007/s10916-010-9449-4

Castro-Jul, F., Conan, D., Chabridon, S., Díaz Redondo, R. P., Fernández Vilas, A., & Taconet, C. (2017). Combining Fog Architectures and Distributed Event-Based Systems for Mobile Sensor Location Certification. *Lecture Notes in Computer Science*, *10586*, 27–33. https://doi.org/10.1007/978-3-319-67585-5_3

Chan, M., Estève, D., Escriba, C., & Campo, E. (2008). A review of smart homes-Present state and future challenges. *Computer Methods and Programs in Biomedicine*, *91*(1), 55–81. https://doi.org/10.1016/j.cmpb.2008.02.001

Dias, L. M. S., Vieira, A. A. C., Pereira, G. A. B., & Oliveira, J. A. (2016). Discrete simulation software ranking — A top list of the worldwide most popular and used tools. *2016 Winter Simulation Conference (WSC)*, 1060–1071. https://doi.org/10.1109/WSC.2016.7822165

Diogenes, Y. (2017). Internet Of Things Security Architecture. Retrieved December 31, 2018, from Microsoft website: https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture

Gope, P., & Hwang, T. (2016). BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network. *IEEE Sensors Journal*, *16*(5), 1368–1376. https://doi.org/10.1109/JSEN.2015.2502401

Hussain, R., & Zeadally, S. (2019). Autonomous Cars: Research Results, Issues, and Future Challenges. *IEEE Communications Surveys and Tutorials*, *21*(2), 1275–1313. https://doi.org/10.1109/COMST.2018.2869360

IEEE Communications Society. (2018). IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing. In *The Institute of Electrical and Electronics Engineers*. https://doi.org/10.1109/IEEESTD.2018.8423800

Joshi, N. (n.d.). Fog vs Edge vs Mist computing. Which one is the most suitable for your business? Retrieved June 21, 2020, from https://www.allerin.com/blog/fog-vs-edge-vs-mist-computing-which-one-is-the-most-suitable-for-your-business

Kharchenko, K., & Beznosyk, O. (2018). The input file format for IoT management systems based on a data flow virtual machine. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (139–142). IEEE. https://doi.org/10.1109/DESSERT.2018.8409115

Kirkpatrick, K. (2013). Software-defined networking. *Communications of the ACM*, *56*(9), 16–19. https://doi.org/10.1145/2500468.2500473

Klügl, F., & Bazzan, A. L. C. (2012). Agent-Based Modeling and Simulation. *AI Magazine*, *33*(3), 29. https://doi.org/10.1609/aimag.v33i3.2425

Laghari, S., & Niazi, M. A. (2016). Modeling the Internet of Things, Self-Organizing and Other Complex Adaptive Communication Networks: A Cognitive Agent-Based Computing Approach. *PLOS ONE*, *11*(1), e0146760. https://doi.org/10.1371/journal.pone.0146760

Lewis, P. R., Platzner, M., Rinner, B., Tørresen, J., & Yao, X. (2016). Self-aware Computing Systems. In P. R. Lewis, M. Platzner, B. Rinner, J. Tørresen, & X. Yao (Eds.), *Natural Computing Series*. https://doi.org/10.1007/978-3-319-39675-0

Marz, N., & Warren, J. (2015). *Big Data: Principles and best practices of scalable realtime data systems* (1st ed.). Manning Publication.

Mostafaei, H., & Menth, M. (2018). Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *119*(June), 42–56. https://doi.org/10.1016/j.jnca.2018.06.016

Multimethod Simulation Modeling for Business Applications – AnyLogic Simulation Software. (n.d.). Retrieved October 5, 2020, from https://www.anylogic.com/resources/white-papers/multimethod-simulation-modeling-for-business-applications/

Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018a). Designing security of personal data in distributed health care platform. *Technology Audit and …*, *2*(42). https://doi.org/10.15587/2312-8372.2018.141299

Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018b). Detection of human respiration patterns using deep convolution neural networks. *Eastern-European Journal of Enterprise Technologies*, *4*(9(94)), 6–13. https://doi.org/10.15587/1729-4061.2018.139997

Pysmennyi, I., Kyslyi, R., & Petrenko, A. (2019). Edge computing in multi-scope service-oriented mobile healthcare systems. *System Research and Information Technologies*, (1), 118–127. https://doi.org/10.20535/SRIT.2308-8893.2019.1.09

Rahmani, A. M., Liljeberg, P., Preden, J.-S., & Jantsch, A. (2018). *Fog Computing in the Internet of Things*. Springer. https://doi.org/10.1007/978-3-319-57639-8

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Oma, R., Nakamura, S., & Duolikun, D. (2019). A fault-tolerant tree-based fog computing model. *International Journal of Web and Grid Services*, *15*(3), 219. https://doi.org/10.1504/IJWGS.2019.10022420

Satyanarayanan, M. (2017). Edge Computing. *Computer*, *50*(10), 36–38. https://doi.org/10.1109/MC.2017.3641639

Sedgewick, R., & Wayne, K. (2011). Algorithms. In *Foreign Affairs* (4th ed.). Westford: Addison-Wesley.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

Spot Instances – Amazon Elastic Compute Cloud. (n.d.). Retrieved July 7, 2020, from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

Stojmenovic, I., & Wen, S. (2014). *The Fog Computing Paradigm: Scenarios and Security Issues*. *2*, 1–8. https://doi.org/10.15439/2014F503

World Health Organization. (2010). Telemedicine Opportunities and developments in Member States. In *World Health Organization* (Vol. 2).

Xiao, Y., & Zhu, Ch. (2017). Vehicular fog computing: Vision and challenges. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 6–9. https://doi.org/10.1109/PERCOMW.2017.7917508

Yogi, M. K., Sekhar, K. C., & Kumar, G. V. (2017). Mist Computing: Principles, Trends and Future Direction. *International Journal of Computer Science and Engineering*, *4*(7), 19–21. https://doi.org/10.14445/23488387/IJCSE-V4I7P104