Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 130 (2018) 90–97

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)

Dynamic bridge generation for IoT data exchange via the MQTT protocol

Alexandre Schmitt[a], Florent Carlier[a], Valérie Renault[a]

[a]*Le Mans Université - Laboratoire Centre de Recherche en Éducation de Nantes (CREN), Avenue O. Messiaen, 72085 Le Mans, France*

Abstract

Infrastructures create many problems in the security, economic or health sector. The high level of connectivity on Internet of Things (IoT) field is at the center of new networks and requires interoperable solutions that promote redundancy, and that fault tolerance. Many standard protocols currently allow extensive exchanges between different IoT groups. In this paper, we are interested in reconfigurable links management between sets of IoT linked by *Message Queue Telemetry Transport* (MQTT). This approach is based on the use of multi-agent systems in charge of arranging network topologies.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Internet of Things (IoT), Networks, Dynamic bridge, Multi-Agent Systems

1. Introduction

The Internet of Things (IoT) are ubiquitous in the industrial world and for individuals: an estimation from Intel evaluates the number of IoT to 200 billion by 2020[10]. Other studies provide lower figures, but they all agree that the current number around the world is already estimated at several billion. Many network-related solutions are emerging for IoT interconnection. Taking into account the possibility of network modifications (failure, IoT mobility and transparency connect/disconnect), this paper examines how to interconnect IoT groups. IoT have several technical constraints, such as energy management. As their longevity depends mainly on the low consumption resulting from their interactions with the rest of the world, a data exchange model adapted to network topologies is therefore essential[11, 16].

Among the different protocols available for network communication[2, 6, 17, 19], we analyse and propose a new interaction model between IoT groups for the *Message Queue Telemetry Transport* (MQTT) protocol. We will focus on server links and their dynamic interactions. Finally, this study will be complemented by an experimentation adapted to the field of multi-agent systems.

E-mail address: firstname.name@univ-lemans.fr

1877-0509
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2018.04.016

2. The MQTT potocol

To carry out machine-to-machine exchanges (Message oriented Middleware : MoM[15]), the *Request/Response* interaction protocol is often found in technical literature[8,14]. This model restricts interactions to direct communication between a client and its server. The IoT field requires exchanges of data between various clients without server processes. The *Publish/Subscribe* model allows IoT networks to carry out large and light exchanges without overloading them with queries. It integrates three levels of quality of service, and takes into account security solutions equivalent to *Request/Response* (TLS/SSL).

OASIS standardises mechanisms of the MQTT[3,5,9,12] protocol and will soon be released in its version 5. This standard defines, among other things, the *broker* as the central node of a star network. Each client must refer to his broker to exchange with the rest of the clients. This model is based on use of a broker with *topics*, organised in a tree structure like a folder hierarchy. A client can submit data on a topic (*publisher* mode), or listen to one or more topics simultaneously to instantly benefit from the data (*subscriber* mode). A customer can be both *publisher* and *subscriber*. This model allows synchronous as well as asynchronous exchanges.

As clients are likely to lose their connections with their brokers, they may use multiple brokers to publish or to subscribe to the same data, but at different locations in the network. This paper focuses on this idea by introducing the concept of broker topology. We are interested in the management and exchanges between brokers so that clients can behave in a similar way regardless of the topology of the network, and regardless of the broker that they are referring to.

Many broker[2] projects allow the generation of gateways between different server instances. Some of them propose the exchange of all or parts of their topics tree structure[1], so we call this feature a *bridge*. This solution makes it possible to create a "mirror" of data between several servers.

Having direct links between servers allows many ways to manage data, such as hierarchical organisation of clients according to their physical location, nature, or criticality[18,20]. They make it possible to set up redundancy or scatter data over different geographical sites. The MQTT protocol does not offer a solution to manage bridges between different heterogeneous brokers (Eclipse Paho / Mosquitto). Several problems arise: how to standardise exchanges between brokers? Do these exchanges have to respond to a specific quality of services? How can we initiate the creation of exchange links and make them evolve dynamically? It is on this last point that we make proposals illustrated by several experiments.

3. Dynamic bridge management by MQTT

The first step in these reflections is to study solutions to create bridges which can be dynamically arranged. We note that the various MQTT open source broker projects available mainly propose the use of a configuration file in which all the options necessary for the implementation of bridges are defined. This solution limits bridge management and makes it static: this configuration file is analysed at the launch of the broker and does not evolve anymore. Brokers can offer hot proofreading of this file. This solution makes it possible to upgrade broker's bridges by modifying his configuration file, but constrains the broker to many modifications.

Figure 1 shows the critical aspect that a static management of brokers' bridges can have. At time $t0$, five brokers are connected in star with the broker 5 as the main node. All the configurations were defined in broker 5's configuration file, and allow broker 5 to connect to brokers 1 to 4. A disconnection occurs on the network at $t1$, then broker 5 is offline and can no longer ensure links between brokers 1 to 4. So, brokers 1 to 4 do not have a solution to reorganise their bridges, and are thus unable to ensure links that allow them to exchange data again.

We propose a new approach that does not change the behaviour of the initial configuration analysis of a broker. Moreover, this proposal is intended to be as close as possible to the philosophy of the MQTT protocol. The standard succinctly defines the use of reserved topics for the needs of the broker called:$ topic. These topics are used to provide data about the server (number of active connections, network load, number of packets transmitted, etc.).

We see these topics as an opportunity to interact with the broker, beyond merely retrieving data. Thus, the server's configurations can be modified to create new scenarios of data exchange without having to restart the service. We propose here a new approach for the OASIS standard and the processing of topics reserved by the $ pattern.

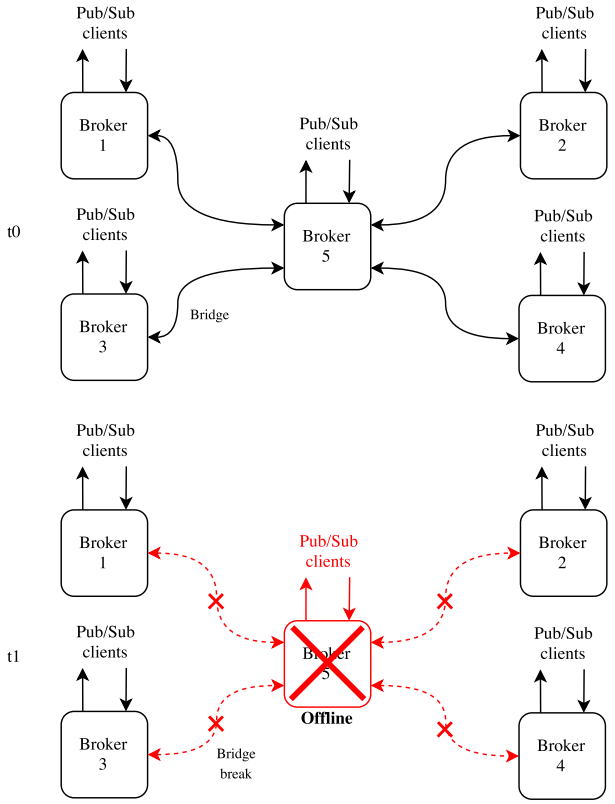

Fig. 1. Star topology of a network of MQTT brokers being disconnected

Thanks to this evolution of the standard, a software authority superior to the protocol can reorganise data mesh without requiring privileged access to the system. This authority can, in this case, be a stand-alone service, and have sole responsibility for the consistency of the brokers' network. In order to ensure this management, we introduce the concept of dynamic bridge with the aim of dynamically organising bridges without interruption of service.

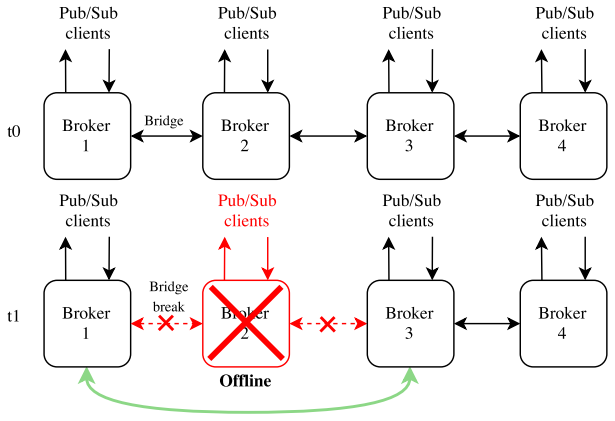

Fig. 2. Line topology of a network of brokers being disconnected with dynamic bridges

Figure 2 illustrates how a dynamic broker network evolves over time if one of the brokers disconnects from the network. At time $t0$ we have a network of brokers with a line topology. This topology represents a significant risk. In the case of a disconnection of one of the links in the line, the entire line after this link is disconnected. However,

our approach automatically resolves a network line-break. Each broker has a link to its nearest superior neighbour. At time $t1$, broker 2 is offline. Dynamic management makes it possible to generate a new link with the new neighbour close to broker 1. This link now connects 1 to 3. Reorganising takes place without restarting the service and without any intervention at the system level.

4. Our approach

The Mosquitto project[1] is a collaborative project (Open Source) between researchers and industrialists adapted to embedded systems for interactions and acts of communication within IoT networks, among other things. Our approach is based on dynamic management of possible bridges between different MQTT platforms/brokers.

Mosquitto uses the principle of topics reserved by the pattern $SYS. The broker lists more than 30 topics that periodically return its data, in addition to propose setting-ups of bridges between different brokers. They allow the exchange of data on specific topics or on its entire tree structure. This bridge management is static and can only be modified by a user with privileged access to the system. The broker's configuration is only taken into account when starting the server.

For bridge management, we added the interpretation of a new branch of bridge topics to the data model. Broker's status is supplemented by 2 topics:

- $SYS/broker/bridge/new, topic for create.
- $SYS/broker/bridge/del, topic for delete.

This solution allows to edit the broker's bridges. The security problem on these topics are managed by the global add-on mechanism proposed by the Mosquitto project. In our case, we restrict access to certain users by using ACL (Access Control List) files.

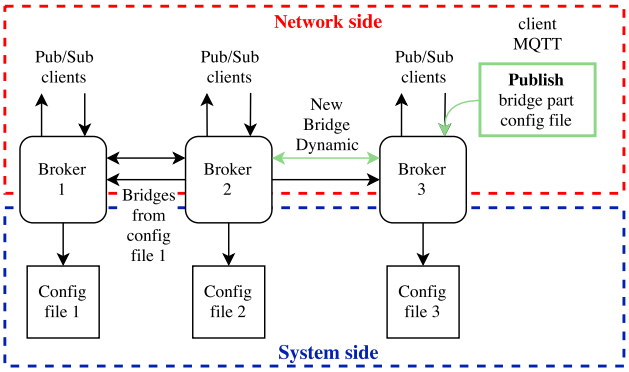

Fig. 3. Bridge creation procedure

Our approach is based on using the parameters available for setting up Mosquitto. We use the content needed for a configuration file. As illustrated in Figure 3, the procedure consists of publishing a text message on the topics mentioned above to interact with the broker. If a message does not meet the same characteristics as the content expected in a configuration file, the message is not interpreted. If the message is valid, then the information is processed. The broker calls the same mechanisms as when it was started to create a bridge. This solution requires no system access and can be managed by any MQTT client.

This solution allows us to dynamically create new bridges without changing the broker's system configuration. We have the possibility to dynamically create any topology of links between brokers. To complete what is proposed in the figure, we are working to match our brokers network with other types of topologies, such as the one in tree or redundant connection. It would be possible to mix some topologies for maximum connectivity without overloading the network.

5. Management by Multi Agent System

Our research team is involved in the development of Artificial Intelligence in the field of the Internet of Things[7]. Our aim is to connect software agents, adjusted to embedded systems, within a multi-agent platform. Thus, each connected object participates in a global intelligence for an autonomous and distributed working. Our model is based on the concept of IoT-a (Internet of Things - agent)[4]. We currently have a multi-agent platform (Triskell3S) distributed on several embedded boards embodying our connected object-agents. All of our agents and boards, inter-connected in a network and communicating via the MQTT protocol, constitute our multi-agent system. Each agent must be able to interact with the rest of the platform at different levels of interaction, intra or inter-board.

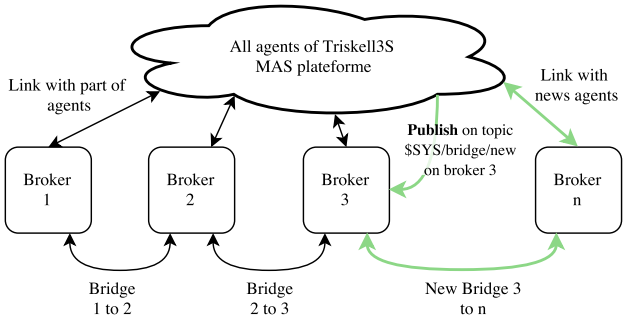

Fig. 4. Bridge management by Multi Agent System

Our approach to dynamic bridges gives us the opportunity to reorganise the topology of our system. Therefore, we offer a solution based on Multi-Agent Systems[13] for that bridge organisation, which increases the interoperability of our system and makes it more reliable when disconnecting network components. Consequently, agents have the possibility to publish on the appropriate topics the creation or deletion of a bridge to allow connecting or isolating an element of the platform.

Figure 4 illustrates our bridge management agents within the triskell3S platform.

6. Experimentation

6.1. Context

Our experimental context aims at connecting five MQTT brokers linked by dynamic bridges. An agent is in charge of detecting brokers' connectivity, and dynamically implementing bridges between all of them.

Figure 5 illustrates the bridges established by our agent during the five stages of our experimentation. In order to visualise the activity of each broker, one client per broker subscribes to a topic shared by all brokers, which is relayed by the bridges. Broker 1 is always connected and receives a publication from a publisher 1 client every 10ms on the shared topic. Brokers 3, 4, and 5 are randomly disconnected to simulate a loss of connectivity. The number of messages received on broker 2, which undergoes automatic re-logging of the bridges, is an image of the reactivity of our experimentation.

Our goal is to establish the highest level of connectivity for our brokers to ensure maximum transmission of messages to subscribing clients (*client_sub_*1 to *client_sub_*5 on figure 5). The constraint given in this experimentation is to perform the lowest amount of creation and deletion of bridges while ensuring at least one link to a main node. Our main heuristic is to link all of our brokers by a star topology where its node will be the connected broker with the highest unique number. This number is automatically assigned during its network discovery. Our constraint leads to what we call "ghost-links". Indeed, when a link is established from broker A to broker B, and broker B comes to disconnect, the link existing in broker A is put on hold until broker B reconnects. If no deletion of this link is made and broker B never reconnects, the link becomes "ghosted" because it is always stored in broker A's memory and periodically tested to evaluate the broker B's connection. If, however, broker B comes to reconnect a β time is noticed before the link is reactivated. This time is due to the internal analysis cycle of broker A checking the connection

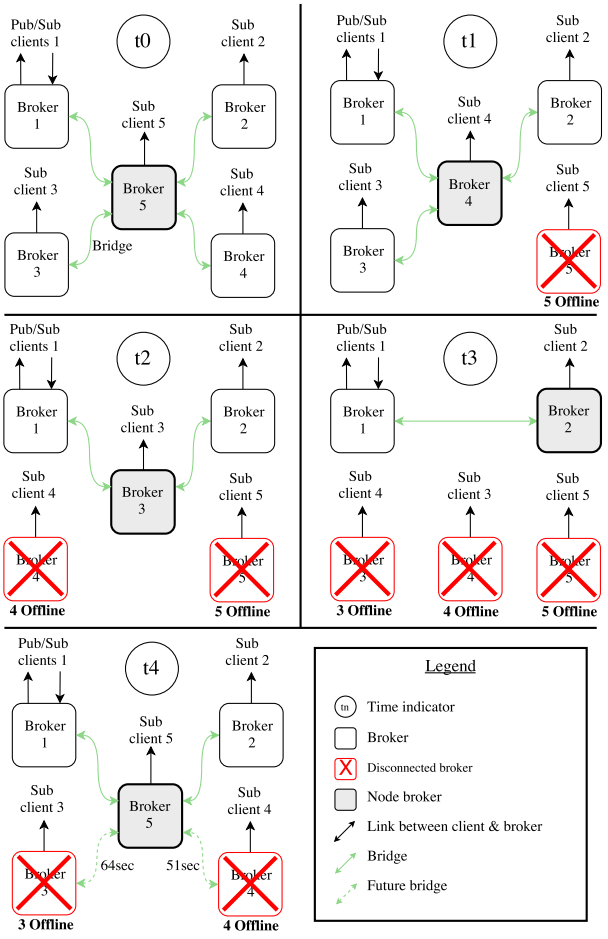

Fig. 5. Experimental scenario: evolution of the topology of brokers linked by dynamic bridges over time

status of all of its bridges. It is important to notice the especially remarkable α time at the $t3$ transition on the figure 6, but it is present at each tn state change. This time is due to our agent's periodic analysis of the connectivity of the different brokers. The γ time, visible on the figure 6, is the sum of the α processing delay of the agent and the β time of re-connection of a ghost-link.

$$\gamma = \alpha' + \beta \tag{1}$$

On stages $t0$, $t1$ and $t2$, successive disconnections of brokers 5, then 4, and then 3 generate a re-negotiation of the network of bridges by our agent. The process of creating and/or suppressing dynamic bridges causes an invisible λ time on the figure 6 will be neglected in our experimentation as it is very close to zero.

6.2. Results

Results in the figure 6 show the advantages of our dynamic bridge solution. The number of messages obtained on broker 1 is our reference in terms of received messages. It is the only broker to receive messages directly from publisher 1, no intermediary or any bridge will interfere with its message reception. It is possible to notice constant sections in the number of messages received for brokers 3 to 5. Those sections are the result of a failure to receive messages, which indicates that brokers 3 to 5 are disconnected. Then, it is the analysis of the number of messages received by broker 2 that allows us to deduce the results obtained by using our dynamic-bridge approach. Ghost-links are active on brokers 1 and 2 to brokers 3, 4, and 5 until broker 5 reconnects ($t4$ transition). From $t4$ the agent removes

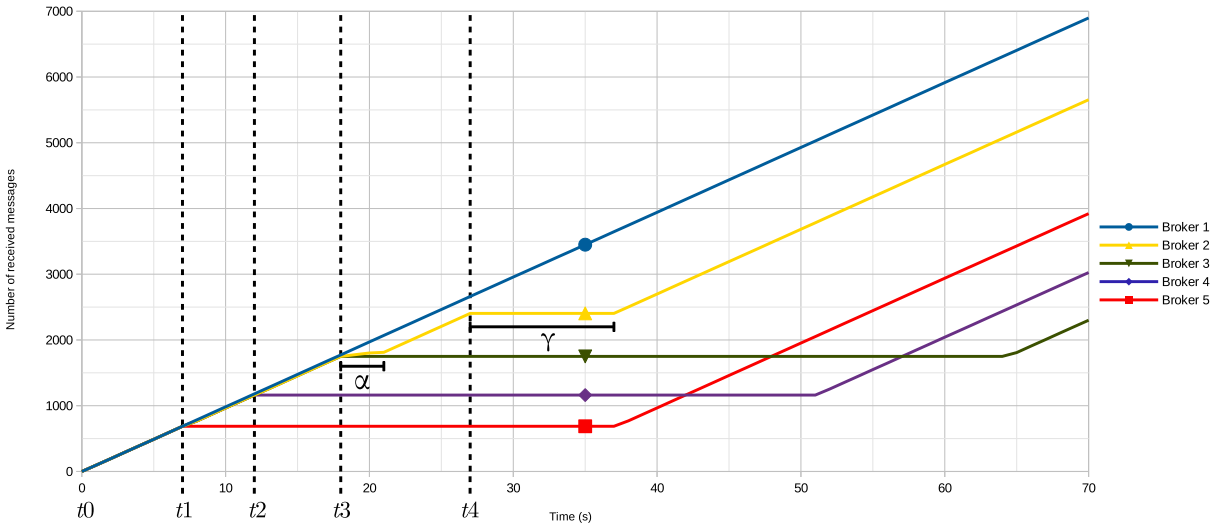

Fig. 6. Number of messages received per broker for experimentation of dynamic bridges organized by the agent

ghost-links from brokers 1 and 2 to brokers 3 and 4, leaving only the one to broker 5. We notice an important γ time before the brokers 1, 2, and 5 reconnect after transition $t4$. Since the re-connections of brokers 3 and 4 after $t4$ no longer implement ghost-links, their connections to the star only involves an α time not visible on the figure.

6.3. Discussion

After analysis of our experiment we can discuss the following 3 main results:

- Losses by static bridge: $L_{sb} \approx 40\%$
- Losses by dynamic bridge: $L_{db} \approx 20\%$
- Losses by pure dynamic bridges: $L_{pdb} \approx 2\%$

L_{sb} corresponds to the loss of the number of messages relayed by a static bridge without using an agent to manage dynamic gateways. L_{db} represents the loss of global messages from our experimentation on broker 2, which only undergoes the reorganisation of bridges. Finally, L_{pdb} is the loss of messages produced by using pure dynamic bridges before the $t4$ transition. A decrease of about 20% between L_{sb} and L_{db} shows the interest of our dynamic data exchange model for a similar broker structure. L_{pdb} does not take into account the beta time that normal behaviours that the ghost-links of the system can produce. These behaviours are related to our experimental constraint requiring the least creations/deletion of bridges. By freeing ourselves from this constraint, we then increase the very low (ms) λ negotiation time, while freeing ourselves from the long β time (s). We then get L_{pdb} about 2% of lost messages in the case of this experiment.

7. Conclusion

In this paper, our approach is able to address the problems of interconnection of embedded systems in networks. Therefore, we are able to dynamically model and create links between MQTT brokers based on multi-agent systems. Compared to the initial model, our results give an efficiency of 20% higher and confirm a real advance in dynamic interaction management. Nevertheless, some parameters (α and β), still need to be evaluated more precisely and adjusted to obtain results that fully satisfy the constraints of high connectivity. Finally, we are planning future trials of topologies in order to provide the best answer to the interactions required for our multi-agent platform.

Acknowledgements

Please note that works led in this article were funded as part of a CIFRE thesis in partnership with the Education research laboratory of Nantes (*CREN*) and the company SARP - SOA Le Mans, with the cooperation of Frédéric Le Bouguénec, Technical and Innovation Manager.

References

1. Roger A Light. Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13), may 2017.
2. Ala Al-Fuqaha, Khreishah Aledhari, Guizani Mohsen, Rayes Ammar, and Mohammadi Mehdi. Toward better horizontal integration among iot services. *IEEE Communications Magazine*, 53(9):72–79, September 2015.
3. Andrew Banks and Rahul Gupta. MQTT protocol specification. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf`. Accessed 19 December 2017.
4. Florent Carlier and Valerie Renault. Iot-a, embedded agents for smart internet of things: Application on a display wall. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA)*, pages 80–83, Omaha, Nebraska, USA, 10/2016 2016. IEEE Computer Society.
5. Whei-Jen Chen, Rahul Gupta, Valerie Lampkin, Dale M Robertson, and Nagesh Subrahmanyam. *Responsive Mobile User Experience Using MQTT and IBM MessageSight*. IBM Corp., 2014.
6. D. K. Choi, J. H. Jung, H. W. Kang, and S. J. Koh. Cluster-based coap for message queueing in intemet-of-things networks. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 584–588, Feb 2017.
7. Enrique de la Hoz, Jose Manuel Gimenez-Guzman, Ivan Marsa-Maestre, Luis Cruz-Piris, and David Orden. A distributed, multi-agent approach to reactive network resilience. *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1044–1053, May 8–12 2017.
8. R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Message syntax and routing. RFC 7230, RFC Editor, June 2014. `http://www.rfc-editor.org/rfc/rfc7230.txt`.
9. Hwang H.C., Park J.S., Lee B.R., and Shon J.G. *An Enhanced Reliable Message Transmission System Based on MQTT Protocol in IoT Environment*, volume 421, pages 982–987. Springer, Singapore, 2017.
10. Intel. A guide to the internet of things infographic. `https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html`, 2016.
11. Xu Junming. *Topological Structure and Analysis of Interconnection Networks*. Network Theory and Applications 7. Springer US, 1 edition, 2001.
12. Jorge E. Luzuriaga, Miguel Perez, Pablo Boronat, Juan Carlos Cano, Carlos Calafate, and Pietro Manzoni1. Improving mqtt data delivery in mobile scenarios: Results from a realistic testbed. *Mobile Information Systems*, 2016:11 pages, 2016.
13. R.-C. Mihailescu, P. Davidsson R. Spalazzese, and Clint Heyer. A role-based approach for orchestrating emergent configurations in the internet of things. *Internet of Agents Workshop 2017 (IoA@AAMAS 2017)*, 2017.
14. Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). RFC 7252, RFC Editor, June 2014. http://www.rfc-editor.org/rfc/rfc7252.txt.
15. Thirunavukkarasu Sivaharan, Gordon Blair, and Geoff Coulson. *GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing*, pages 732–749. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
16. T. Takabatake. Simulations of noc topologies for generalized hierarchical completely-connected networks. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–5, June 2011.
17. Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. Performance evaluation of MQTT and CoAP via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2014.
18. Shubha Rao V and M. Dakshayini. Priority based optimal resource reservation mechanism in constrained networks for iot applications. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1228–1233, March 2016.
19. Karagiannis Vasileios, Chatzimisios Periklis, Vázquez-Gallego Francisco, and Jesus Alonso-Zarate. A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing (TICC), 2015*, 1(1), January 2015.
20. P. Zhu, J. Han, Y. Guo, and F. Lombardi. Reliability and criticality analysis of communication networks by stochastic computation. *IEEE Network*, 30(6):70–76, November 2016.