Reliable and Fault-Tolerant IoT-Edge Architecture

[1]Jitender Grover and [2]Rama Murthy Garimella
[1,2] Signal Processing and Communication Research Center,
[1,2] International Institute of Information Technology, Hyderabad
[1]jitender.grover@research.iiit.ac.in, [2]rammurthy@iiit.ac.in

***Abstract*— Edge computing has emerged as an effective solution for delay sensitive IoT applications. In the edge-cloud hierarchy, reliability and fault tolerance are important issues. This paper proposes a novel agent-based reliable and fault-tolerant hierarchical IoT-cloud architecture which can survive the failures of the edge servers. In the proposed architecture, the cloud is distributed over four levels (*cloud-fog-mist-dew*) based on the processing power and distance from the end IoT device. It makes the whole system reliable by replicating the data on the edge of the network. The proposed system is fault tolerant as it redirects the application on an alternate server. In case of a server's failure, redirection is done at the best possible level in the hierarchy, based on delay-tolerance of the IoT application. The application keeps on working even if the system fails on any level, on cloud, fog or mist. The solution is proposed using mobile agents (MAs) on servers to share systems' state and other important information with other agents in the hierarchy, to be used for application redirection in case of server's failure. The proposed system is simulated using Matlab and results prove its efficiency.**

***Index Terms*— Edge Computing, Real-time IoT, Reliability, Fault Tolerance, Fault Localization, Cloud, Mobile Agent.**

I. Introduction

Many of the future IoT applications seem to be delay sensitive and/or bandwidth hungry. A few examples are patient monitoring in ICU, automated cars, smart cities, AR/VR, VANET, real-time crowd surveillance etc. To manage IoT data and applications remotely, cloud computing has been used [1]. IoT devices sense any physical phenomenon like distance, sound, audio, video etc. They have the capability of connecting to the internet using wired or wireless connections, directly or through some gateway. Sensed data is processed on cloud and feedback is sent back to actuators. What if due to the network or cloud fault, IoT device is unable to connect to the cloud server? Edge computing is emerging as a solution to such scenarios. The application can survive using a local edge server even if there is any fault in core internet connectivity.

A. Edge Computing: Fog - Mist - Dew

The amount of delay between IoT devices and cloud, the load on cloud and communication network during sensed data transmission gave birth to the idea of computation on the edge of the network [2][3]. Figure 1 shows levels of edge networks (fog, mist and dew) and how they are connected to the cloud via the core internet.

Extreme Edge is called *dew*. These are actually the end IoT devices. On dew, the delay is almost negligible as there is no communication required with any other node to take any decision. Dew computing is used when the application requires real-time decision making without any delay. It works well if the requirement of computation power and previously sensed data is very low. In such cases, dew computing is very effective and saves energy, bandwidth etc. For example, an automatic honk system on mountains for the drivers on U-turns.

Mist (Roof) computing also exists at the extreme edge network [4]. In Mist computing, a server is established very near to the IoT device, usually in the same building/institute and connected via a wired/wireless network. It is appropriate when the processing power and the previously sensed data requirement is more than the capability of dew level. The delay is also very low in mist. The communication distance varies from a few meters to a few hundred meters.

Middle edge or *fog* connects extreme edge to the cloud via core network (WAN) [5]. Fog can be as distant as a few kilometres from the IoT devices [6]. The Fog server can exist with the BS of an ISP or separately in the network. The computation power and storage requirement is higher in fog server compared to mist. There can be many mist connected to one fog. The distance of fog from the IoT devices is more than dew/mist but it is still preferred due to its higher processing power and least delay as compared to mist and dew [7].

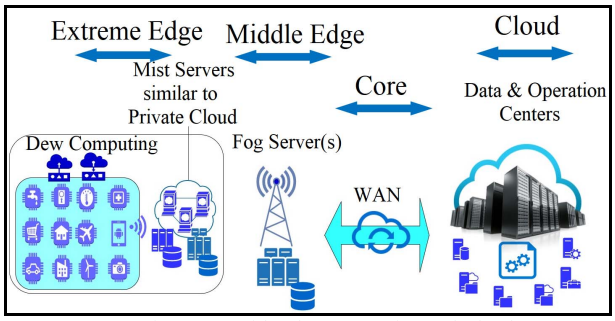

Fig. 1. IoT with Edge Computing and Cloud

The core is the WAN, comprised of all the traditional network elements like routers, bridges etc. If Cloud is considered at the top, then it creates a hierarchy of networks with fog, mist and dew on the lower levels. Edge computing [8] seems to solve the issue of delay, bandwidth and fault on cloud level, but what if any fault occurs on edge levels, either on middle edge or on extreme edge? In this paper, a reliable and fault-tolerant architecture for the IoT-Cloud hierarchy is proposed.

II. Related Work

Many researchers have proposed solutions for fault tolerance in IoT-Cloud infrastructure. Some of the proposed solutions are given in this section. A framework for providing a hybrid fault tolerance in cloud computing [9][10][11] addressed the issues of fault tolerance that can cause failure of the cloud.

978-1-5386-4707-3/18/$31.00 ©2018 IEEE

In [12], the authors proposed a distributed cloud storage architecture in which services or data is provided to the user in a manner that they can access it locally without having an internet connection. The proposed architectures do not provide any solution to handle replicated data stored on users' machines. In [13], an algorithm is proposed to increase the reliability of the system by using virtual machines. Virtual machines adapt changes of the system in every cycle and generate correct results. The drawback of the proposed algorithm is that there is no schedule management system or a decision-making system.

In [14], the authors proposed a fault tolerance session layer to overcome the drawback of event-driven communication. With the help of adding new layers and checkpoints, it maintains fault tolerance. This mechanism uses a large number of threads that may reduce the performance of the server. A fault tolerance framework based on the artificial neural network to detect faults and maintains a gap between cloud servers and users is proposed in [15]. In [16][17], the authors presented an evaluation study on different existing solutions for managing efficient cloud storage and utilization of resources with different quality assurance parameters. They proposed a scheduling mechanism in which user requests and clouds services are managed or scheduled efficiently but those algorithms work only with the centralized environment but fails in a distributed environment. Also, the proposed framework works based on static information of resources and users requests.

Leveraging fog computing for scalable IoT datacenter, a mechanism to overcome latency and bandwidth issues is proposed in [18]. In this mechanism, the concept of spin leaf helps to manage big data on Cloud by offload and batch processing, but it does not support multinode architecture. Also, it is not so efficient for scheduling related problems.

III. Reliability, Fault Tolerance & Fault Localization

With the increase of IoT devices in the system, maximization of the computation on edge network is required to minimize the communication cost. Self-managing and self-configuring solutions are also required on edge network. The IoT application must be able to recover from any issues that arise during its lifetime. Mist and fog computing should offer these features. So in some sense, it has been believed that mist and fog will address many of the challenges that are being faced in dealing with large-scale IoT systems.

Now, if this hierarchical architecture is adopted, some of the biggest concerns are reliability and fault tolerance. There are several concerns that need to be taken care of.

- Reliability and fault tolerance means the whole system architecture should be able to provide services even if any node (server) on any level fails.
- The sensed data should be replicated and available on other parallel nodes to take over the control in case of a node failure.
- There should be application interface available on the newly assigned alternate server. It should be managed virtually without any inconvenience and knowledge to end user.
- The switching from one server to another should be automatic, and it could be based on priority. For example, if one fog fails then its IoT applications should be shifted to another fog, mist or to cloud (in the worst case), automatically. The decision depends upon the delay constraints of the applications.
- The new server's reconfiguration and path redirection time should be minimal.

IV. Proposed Architecture

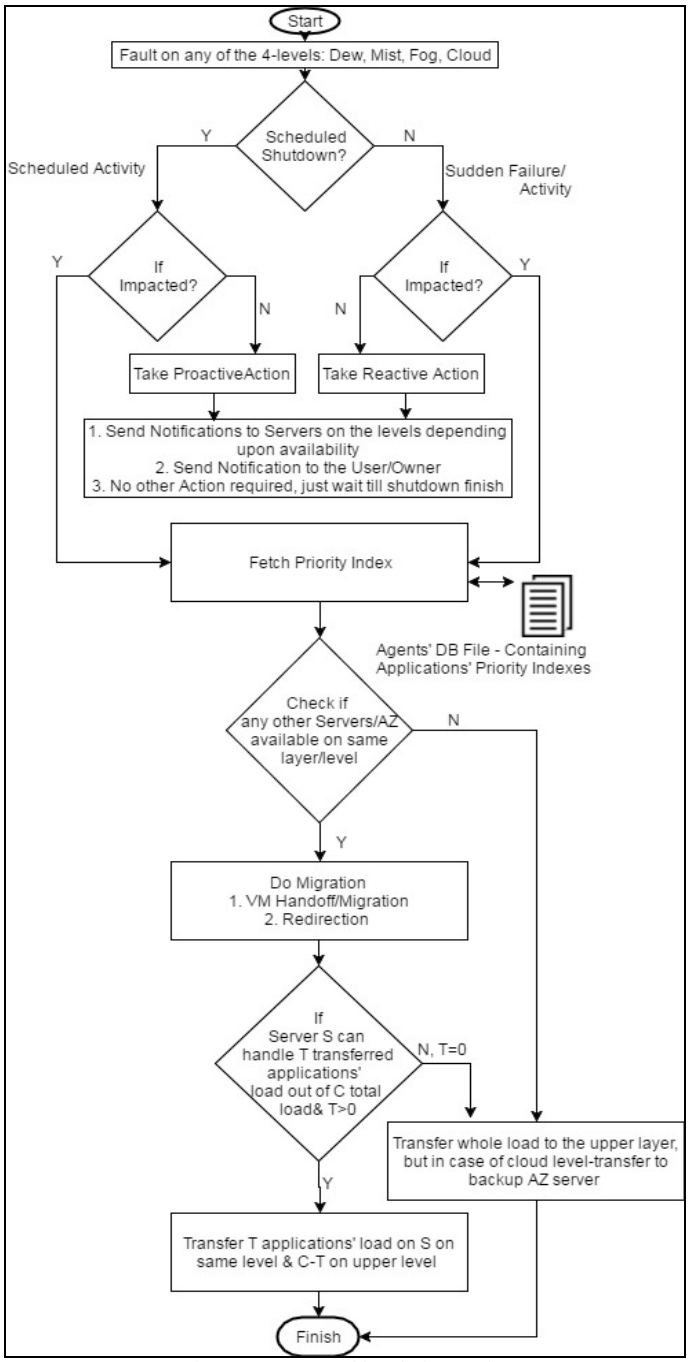

Fig. 2. Agents' Working during Faults

A. Reliability & Backup Policy

To construct a reliable IoT-Cloud infrastructure, the replication of sensed data is very important. The sensed data's redundancy makes sure the feedback for the actuators in case of any server-side failure. [19][20] The backup system can take over the control in case of any failure. For clouds, the mechanism of AZ (Availability Zone) for the backup is used in case of any disaster or failure in the cloud server.

Now, while using the concept of edge computing, the focus is on running IoT applications from the edge of the network rather than from the far cloud. So fog and mist level servers should also have backup sibling servers to keep the applications running in case of any edge server's failure. The proposed multi-level architecture is reliable in the sense that even if there are no alternative servers on the same level with sensed data replication, there is a guaranty of replication on the higher level. The volume of data for the same application may vary on mist, fog and cloud levels. But data would exist on all the levels to take over the application's control in the case of a server's failure.

B. Fault Tolerance and Agent

By definition, mobile agent (MA) is a special purpose software code that can transfer itself from one machine to another. Practically, MA is the same code running in all the machines and transmits data from one machine to another. MAs are very helpful in managing distributed systems. In the proposed work, the faults in the entire hierarchy will be managed by MAs.

The MA works as a resource and network monitoring agent. It shares the application and link state information with other agents on alternative servers in the hierarchy. Except for monitoring, they are also responsible for assigning the priority to the IoT applications depending upon their delay-tolerance. In the case of impacted failure or impacted scheduled shutdown, this priority information is used at the time of load distribution [21] of a particular server. It also helps in new path discovery [22] after the load distribution. It also keeps a check on periodic monitoring and backup of data. Figure 2 shows the recovery process in case of faults and scheduled shutdowns, reactively and proactively respectively.

Figure 2 depicts the complete fault tolerance cycle of the hierarchy between dew, mist, fog and cloud. Here the fault tolerance has been achieved by exploiting the capabilities and benefits of an agent, which is basically a software program running on each server.

At the time of any fault, the whole responsibility is assigned to the same level agent via a higher level agent in the hierarchy. *Agents' working* is given below:

1(a). As per assumption, the reason behind any fault/shutdown can either be any scheduled activity or any sudden activity.

1(b). Any sudden or scheduled activity comes with two possibilities that either it is going to affect respective server or no effect at all on it.

1(c). If the ongoing activity has no effect then for a scheduled event, the agent will move with proactive action which is basically to send notifications to all respective agents, whereas for sudden activity agent will look for reactive action.

2. For any sudden fault, the agent will fetch priority index for all applications running on the affected server and it will immediately check if any other server is available on the same level via a higher-level node in the hierarchy. After that, it will do application migration and connection redirection and then will do the load transfer activity as per the sequence is given below:

Total load=C ; Server S can handle (load) =T
if T>0, then
Transfer T (load) --> S
Transfer (C-T) load --> upper level
else if T=0
Transfer whole load --> upper level

3. But, if no other same level server is available then it will point to the upper level to transfer the whole load.

V. Simulation Results

Both agent-based and without-agent solutions are simulated using Matlab. Figure 3 shows the application assignment efficiency of the system during faults with MAs in comparison to random assignment. With the help of MAs, job distribution for the idle server is high in numbers and null for under maintenance servers. While job distribution for busy and backup servers performed efficiently. If the system proceeds without an agent then it distributed jobs randomly without acknowledging the servers' state. Figure 4 shows the CPU time unit consumption with the increase of server counts. When the system proceeds with MAs then performance is very high while the centralized system just fell down in performance.

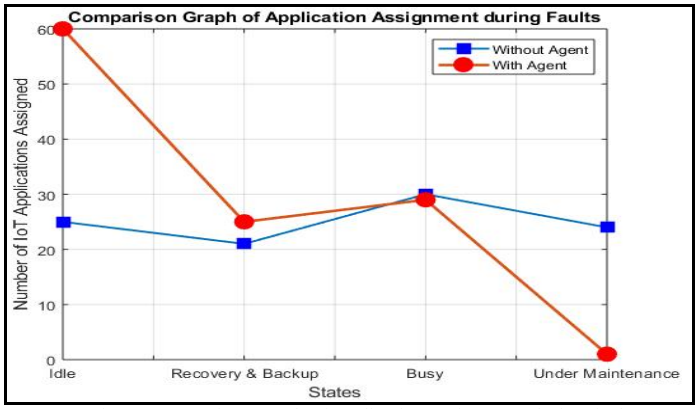

Fig. 3. Comparison graph of application assignment during faults

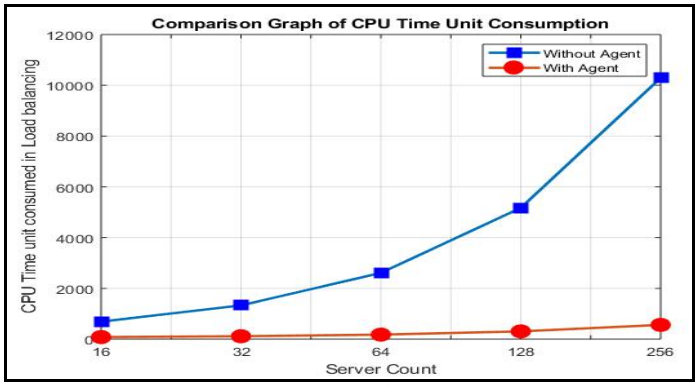

Fig. 4. CPU time unit consumption

VI. Conclusion

The proposed work puts efforts for improving fault-tolerance and reliability of edge computing with the help of an intelligent agent. The given architecture provides solutions for the possible faults at any level of the cloud-hierarchy. To deal with any fault or issue, the proposed concept works with both types of solutions that are proactive and reactive. Results also show the efficiency of the proposed architecture. This is a practical solution and in future, it can be tested by implementation on real test-beds.

REFERENCES

[1] Wan, Jiafu, and Min Xia. "Cloud-Assisted Cyber-Physical Systems for the Implementation of Industry 4.0." Mobile Networks and Applications (2017): 1-2.

[2] Mahmud, Redowan, and Rajkumar Buyya. "Fog Computing: A Taxonomy, Survey and Future Directions." arXiv preprint arXiv:1611.05539 (2016).

[3] Mach, Pavel, and Zdenek Becvar. "Mobile edge computing: A survey on architecture and computation offloading." IEEE Communications Surveys & Tutorials (2017).

[4] Satyanarayanan, Mahadev, et al. "Edge analytics in the internet of things." IEEE Pervasive Computing 14.2 (2015): 24-31.

[5] Satyanarayanan, Mahadev. "The Emergence of Edge Computing." Computer 50.1 (2017): 30-39.

[6] Songqing Chen, Tao Zhang, Weisong Shi, "Fog Computing", IEEE Internet Computing, 2017, pp. 4-6.

[7] Satyanarayanan, Mahadev. "The emergence of edge computing." Computer 50.1 (2017): 30-39.

[8] Datta, Soumya Kanti, Christian Bonnet, and Jerome Haerri. "Fog Computing architecture to enable consumer centric Internet of Things services." Consumer Electronics (ISCE), 2015 IEEE International Symposium on. IEEE, 2015.

[9] Amoon, Mohammed. "A framework for providing a hybrid fault tolerance in cloud computing." Science and Information Conference (SAI), 2015. IEEE, 2015.

[10] Charity, Talwana Jonathan, and Gu Chun Hua. "Resource reliability using fault tolerance in cloud computing." Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on. IEEE, 2016.

[11] Mohamed, Nader, and Jameela Al-Jaroodi. "A collaborative fault-tolerant transfer protocol for replicated data in the cloud." Collaboration Technologies and Systems (CTS), 2012 International Conference on. IEEE, 2012.

[12] Wang, Yingwei, and Yi Pan. "Cloud-dew architecture: realizing the potential of distributed database systems in unreliable networks." Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, pp. 85-89.

[13] Sheheryar Malik and Fabrice Huet, "Adaptive Fault Tolerance in Real Time Cloud Computing." 2011 IEEE World Congress on Service.

[14] Ivaki, Naghmeh, Serhiy Boychenko, and Filipe Araujo. "A fault-tolerant session layer with reliable one-way messaging and server migration facility." Network Cloud Computing and Applications (NCCA), 2014 IEEE 3rd Symposium on. IEEE, 2014.

[15] Amin, Zeeshan, Harshpreet Singh, and Nisha Sethi. "Review on fault tolerance techniques in cloud computing." International Journal of Computer Applications 116.18 (2015).

[16] Madsen, Henrik, G. Albeanu, Bernard Burtschy, Fl. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable fog computing.", Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on. IEEE, 2013.

[17] Cardellini, Valeria, et al. "On QoS-aware scheduling of data stream applications over fog computing infrastructures." Computers and Communication (ISCC), 2015 IEEE Symposium on. IEEE, 2015.

[18] K. C. Okafor, Ifeyinwa E. Achumba, Gloria A. Chukwudebe, and Gordon C. Ononiwu, "Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology." Journal of Electrical and Computer Engineering 2017 (2017).

[19] Martin, Andre, Christof Fetzer, and Andrey Brito. "Active replication at (almost) no cost." Reliable Distributed Systems (SRDS)", 2011 30th IEEE Symposium on. IEEE, 2011.

[20] Shen, Min, Ajay D. Kshemkalyani, and Ta-yuan Hsu. "Causal consistency for geo-replicated cloud storage under partial replication." Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International. IEEE, 2015.

[21] Skobelev, Petr, and Damien Trentesaux. "Disruptions Are the Norm: Cyber-Physical Multi-agent Systems for Autonomous Real-Time Resource Management." Service Orientation in Holonic and Multi-Agent Manufacturing. Springer, Cham, 2017. 287-294.

[22] Bittencourt, Luiz F., et al. "Mobility-aware application scheduling in fog computing." IEEE Cloud Computing 4.2 (2017): 26-35.