Fault tolerance in IoT sensor networks, a distributed systems perspective.

Abstract

This poster discusses fault tolerance in distributed systems specifically within the internet of things. IoT systems require robust sensing and networking both can be crucial for the correct operation of the system. An experiment is conducted utilising the MQTT protocol, NTP for clock synchronisation and a UDP connection between nodes and a UPS to achieve a fault tolerant and fault aware sensor system. An analysis is conducted to evaluate the effectiveness of the system in being fault tolerant.

Introduction

The goal of my research is to improve fault tolerance in IoT distributed sensor networks. This poster considers various fault tolerant systems to evaluate the effectiveness of existing solutions and to propose a solution derived from relevant literature to perform an experiment to analyse the solution effectiveness. This posters' scope looks specifically at fault tollereance at the "thing" level with a focus on hardware to support reliable edge computing for the transmition of data to a remote server.

The rationale for this problem is that due to the heterogeneous nature of distributed systems there are many components and layers that can introduce a multitude of various faults. Due to the nature of a distributed system fault handling is necessary to prevent an entire system crash. This is often achieved using hardware and software redundancy, robust networking and the implimentation of an uninterruptable power supply and appropriate security.

Security is also considered due to the possibility of a hacker being able to introduce faults within the system through data interception and corruption through a man in the middle attack during data transmission.

Constraints on the experiment detailed in this paper include hardware limitations to 2 node MCU units with finite memory of 1044464 Bytes each for program, certificate and key storage including the networking environment in which the experiment is conducted.

Hypothesis

Introducing hardware redundancy increases the reliability and availability of the resource the redundancy is intended to protect and allows the application to continue to function during a fault instance. A UPS prevents power failure of nodes within the system for a specific ammount of time based on the device power draw.

Problem Statement

The problem this document will explore is Fault tolerance within a distributed sensor network. The application scenario I will explore is fire detection. This problem is significant as if a sensor node (IoT device) is unable to communicate with the server to publish its data this creates a critical error that makes the entire system crash and renders the application unusable. To evaluate the fault tolerance of the system throughput consistency is considered during a LAN failure. Measurable’s will be the timestamp accociated with each message and the payload data of the message.

Aim

The aim of this paper is to design a fault tolerant architecture for NodeMCU based sensing systems.

Objectives

- Derive a fault tollerant architecture with specific considerations within the application scenario to represent the use case for the system.
- Hardware redundancy, 2 IoT nodes each with a temp sensor, if any sensor goes offline the system can continue to operate.
- Local connection (UDP) between nodes in case of a network failure on a single node.
- UDP receiver and cloud publisher program (Master Node) & UDP sender program (Redundant node).
- Using Amazon cloud services for node temperature subscription.
- Identify possible failure instances.
- Analyse the consistency in message throughput during a local area network failure.
- Analyse the effect of power failure of nodes within the system.
- Analyse the effect of failed sensors on the system.

Background

IoT fault tolerance is imperative, users expect their smart applications to operate correctly and continuously (Terry D. 2018). Sensor networks are expected to remain functional however hardware faults and damage does occur. Within the application scenario of fire detection damage to hardware is highly likely. It is also important that the sensor system is readily available with no sensor fault to avoid an instance where a fire occurs yet it is not detected due to a fault within the system. By creating a linked topology of IoT nodes the failure of a sensor can be identified and the system can continue to operate. The network time protocol is used to sync the CPU clock of the master node to the network this is completed using the NTP pool project a virtual cluster of timeservers (T. Rytilahti et al. 2018) that sets the master node clock to UTC (coordinated universal time).

The message que telemetry transport protocol MQTT is a lightweight publish-subscribe network protocol developed by IBM (Johari, R et al. 2020). Quality of service is an important variable within the MQTT protocol, quality of service level 0 is the minimal QoS level. This service level guarantees a best-effort delivery. There is no guarantee of delivery. The recipient does not acknowledge receipt of the message and the message is not stored and re-transmitted by the sender. Quality of service level 1 refers to when the client receives a return packet from the broker to acknowledge receipt of the message and so yeilds more reliability as this method guarantees that the message will be transferred successfully to the broker this provides the same guarantee as the underlying TCP protocol.

Alqinsi, P et al (2018) detail an IoT Based UPS Monitoring System Using MQTT this system uses a digital ammeter and voltmeter to monitor AC input and UPS output. In terms of fault tolerance this system can be used to monitor power supply parameters to identify AC failure and calculate the amount of time the system can continue to operate based on the power draw from the UPS. The system will continue to function for only a finite amount of time before the UPS battery is drained.
Sadio, O et al. (2019) state that security within the MQTT protocol is left to the implementer, security often uses TLS at the transport layer. Arjadi, R et al (2018) conduct a received signal strength indicator comparison of ESP8266 based modules. Modules with an integrated dipole antenna yeilded the strongest RSSI. The use of better antennas would help to increace the fault tollerance of the system.

Experimental design

In this experiment 2 Iot edge devices, node micro controller units with the ESP8266 WiFi module equipped with a 2.4GHz transceiver with a meander PCB antenna. These modules are used to publish temperature data to amazon web services cloud. A MQTT protocol connection is established between the master node and AWS to publish data using quality of service level 0. UDP is implimented as a connection between nodes. Each node within the system uses a DHT11 sensor. TLS security is also implimented at the transport layer. Network layer security is not considered in this paper whilst TLS will be implimented to better represent the use case of this system.

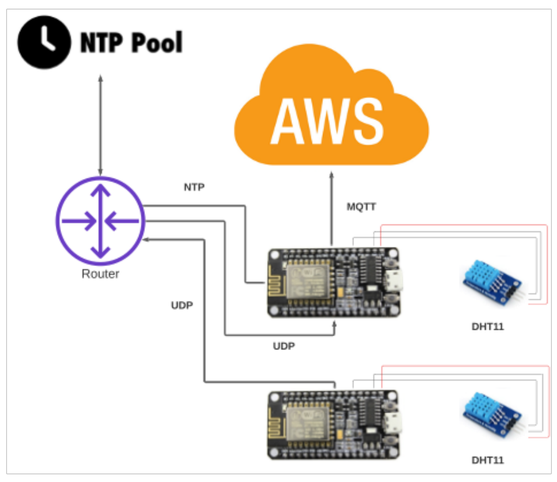


Figure 1: Illustration of experiment architecture.

- More nodes can be added if needed.
- Redundant sensors can be added for each individual node to switch between.
- Temperature data is published to AWS and a timestamp is assigned.
- NTP is used to sync time for the master node to the coordinated universal time of the internet to integrate with AWS.
- OutTopic for the publication of sensor data.

Independent variables

- Introduction of fault instances hardware failure, power failure and network failure of each node.

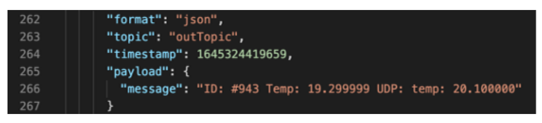
Controlled variables

- Protocols used, node hardware identical microcontroller unit architecture, identical sensors.

Dependent variables

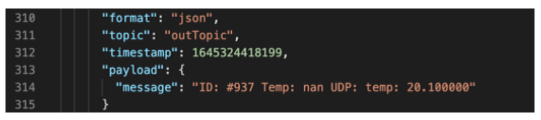
- Timestamp and message data.

Results



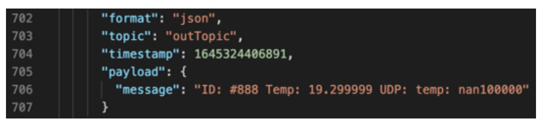
```
"format": "json",
"topic": "outTopic",
"timestamp": 1645324419659,
"payload": {
  "message": "ID: #943 Temp: 19.299999 UDP: temp: 20.100000"
}
```

Figure 2: JSON showing temperature on both nodes.



```
"format": "json",
"topic": "outTopic",
"timestamp": 1645324418199,
"payload": {
  "message": "ID: #937 Temp: nan UDP: temp: 20.100000"
}
```

Figure 3: JSON showing failure of sensor on master node.



```
"format": "json",
"topic": "outTopic",
"timestamp": 1645324406891,
"payload": {
  "message": "ID: #888 Temp: 19.299999 UDP: temp: nan100000"
}
```

Figure 4: JSON showing failure of sensor on UDP node.

- A timestamp is assigned by AWS for each message the timestamp is in milliseconds from the coordinated universal time of the internet.
- Messages continue to be received by AWS if a UDP node fails.
- Nan (Not a number) is published as a flag to show when a sensor has failed.
- Power outage on the second node results in no temperature being displayed for UDP yet the system is still able to function. This only occurs when the second node has not yet sent data to the MQTT node as this node retains the previous message.

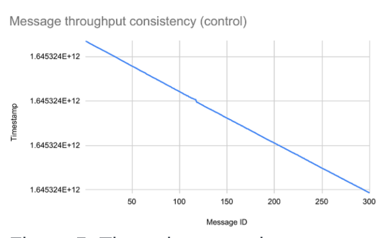


Figure 5: Throughput consistancy control

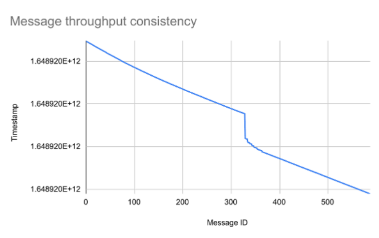


Figure 6: Message consistancy, network failure.

The message throughput results obtained from AWS are shown in Figures 5 & 6, figure 5 shows the control where the sensor network was placed next to the router. Figure 6 shows when the master node is moved beyond the WiFi signal threshold.

Discussion

- Figure 1 shows the architecture of the proposed sensing system, Figures 2,3 and 4 show the data received by AWS where the system continues to function when sensor hardware fails represented by nan (not a number).
- For IoT applications that require near real-time wireless connections, the traditional TCP protocol is not adept enough due to the size of its packet header therefore I have used UDP, in this case UDP has been successful in transmitting real-time data. There is less data integrity using UDP due to packet drops however real-time data is more important in this scenario and packet drops are not mission critical.
- Figure 6 shows the consistency of messages received during a network failure. After message 300 the graph shows the last message received by the broker since no timestamp is assigned to a new message due to the network failure. The connection is re-established and throughout is shown to be inconsistent until moved closer to the access point.
- The experiment outcome proves my hypothesis that introducing redundancy allows the application to continue to function during a hardware failure. Networking remains a key aspect in the implementation of a sensor network and must be reliable and robust to support the fault tolerant application.
- Power is another key considerable for fault tolerance, the power supply to each node must be maintained. This can be achieved using an uninterruptable power supply UPS. A UPS implements mains power supply coupled with a battery backup to continue operation if mains power is not supplied. A simple solution was implemented within the experiment where a battery that is simultaneously charged is used to supply power to the nodes.

Conclusion

The fault tolerant architecture presented works as intended and is successful in publishing both node temperature and not a number to identify failed sensors.

The resource redundancy was intented to protect was sensor data that is sent to AWS. The hypothesis was proven by the experiment as redundancy was sucsessful in protecting the resorce (tempreture sensor data) for transmition to the broker.

The master node retains the previous message received via UDP in the event of a redundant node failure this makes a fault with the sender node undiagnosable by the master node, programming the master node to set a flag when the UDP connection goes offline would solve this problem.

The solution proposed published nan where a sensor has failed, it would be more effective to publish to a separate failure topic acting as a flag to faults where a subscriber can display whether a failure has occurred. A separate subscriber can also be implemented to display sensor measurements on any device that supports MQTT.

Future work

- To further improve fault tolerance when the master node goes offline the MQTT connection could be re-established on another functioning node this would further protect the system from the single point of failure from the master node that would render the entire system inoperable.
- J.Grover et al. (2018) discuss using a mobile agent to transfer load if the server cannot handle throughput, a similar system could be implemented where if a node cannot handle throughput or goes offline it can balance using MQTT on redundant nodes, if the master node is disconnected from the network a reduntant node can be programed to establish a new MQTT connection.
- A redundant sensor for each node can be implimented to further increase the fault tollerance within each node.
- The implimentation of a digital voltmeter and ammeter within the experement would have improved the fault handling within the system.

References

Grover, J., & Garimella, R. M. (2018, October). Reliable and fault-tolerant IoT-edge architecture. In*2018 IEEE sensors*(pp. 1-4). IEEE.

Terry, D. (2016). Toward a new approach to IoT fault tolerance.*Computer*,*49*(8), 80-83.

Rytilahti, T., Tatang, D., Köpper, J., & Holz, T. (2018, April). Masters of time: An overview of the NTP ecosystem. In*2018 IEEE European Symposium on Security and Privacy (EuroS&P)*(pp. 122-136). IEEE.

Masirap, M., Amaran, M. H., Yussoff, Y. M., Ab Rahman, R., & Hashim, H. (2016, May). Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT). In*2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*(pp. 200-205). IEEE.

Kang, D. H., Park, M. S., Kim, H. S., Kim, D. Y., Kim, S. H., Son, H. J., & Lee, S. G. (2017, February). Room temperature control and fire alarm/suppression IoT service using MQTT on AWS. In*2017 International Conference on Platform Technology and Service (PlatCon)*(pp. 1-5). IEEE.

Johari, R., Bansal, S., & Gupta, K. (2020, September). Routing in IoT using MQTT Protocol. In*2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*(pp. 1-5). IEEE.

Alqinsi, P., Edward, I. J. M., Ismail, N., & Darmalaksana, W. (2018, July). IoT-Based UPS monitoring system using MQTT protocols. In*2018 4th International Conference on Wireless and Telematics (ICWT)*(pp. 1-5). IEEE.

Sadio, O., Ngom, I., & Lishou, C. (2019, October). Lightweight security scheme for mqtt/mqtt-sn protocol. In 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS) (pp. 119-123). IEEE.

Arjadi, R. H., Candra, H., Prananto, H. D., & Wijanarko, T. A. W. (2018, October). RSSI Comparison of ESP8266 Modules. In 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS) (pp. 150-153). IEEE.