Distributed fault management for IoT Networks  
draft-hongcs-t2trg-dfm-00

Abstract

Recent advances in Internet of Things (IoT) have increased the use of sensing technologies for IoT applications. However, monitoring sensor nodes is still a challenging issue in distributed remote environments, especially wireless environments. Different from conventional centralized mechanism, Fog Computing becomes an essential role in a scalable IoT system. Fog Node can control and monitor its subdomain’s devices and perform aggregation tasks to support the central server at the cloud. Since node故障 detection can strongly affect the performance and accuracy in most IoT analysis applications, fault detection mechanism should be integrated into IoT Networks. Accordingly, these fault nodes could be detected and replaced by others available nodes in the same domain for the analysis by a distributed fault detection and node replacement mechanism based on their sensory values in a considered domain.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
1. Introduction

IoT Networks are composed of massive, small and low-cost sensor nodes scattered deployed. Using IoT nodes, the sensory data can be collected for IoT applications through fog nodes. Accordingly, the central server and fog node need a fault detection mechanism to monitor sensor nodes in their domain. Failed nodes may affect the quality of service (QoS) from IoT analysis applications. It is an important feature in IoT management systems since faults in IoT Networks occur often due to the following reasons:

- Failure in sensor nodes can occur due to massive low-cost sensor nodes are often deployed in low-cost IoT platform.
. The critical applications are very sensitive to the quality of sensory values.

. Faults can be occurred due to battery depletion in battery-powered nodes.

. The wireless link can be disconnected and the sensory values cannot be updated at the central server.

Faults in IoT domain can be classified into two types as in [a]:

. ‘Hard fault’ is when a sensor node cannot communicate with the monitoring server (e.g., communication failure due to the failure of the communication module, energy depletion of a node, being out of the communication range of entire mobile network because of the nodes moving and so on).

. ‘Soft fault’ means the failed nodes can communicate with the monitoring server but the data sensed or transmitted is not correct.

1.1. Terminology and Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Communication Process

The machine-to-machine (M2M) interaction model, known Constrained Application Protocol (CoAP) similar to client/server model in HTTP is adopted. This method can provide a flexible interaction environment to handle message exchanges between client and server nodes. Unlike HTTP, the message interchanges asynchronously over UDP.

One complete message exchange for the application is handled in three stages. In stage-1, the sensor node (client) instantiates the device registration process by forwarding the device information in JSON format (see Figure 1). The server stores the device information in TDB, and trigger application-driven control message (Start/Stop observation) in Stage-2. In Stage-3, if the sensor node is not sending observation data, it publishes the sensory data (e.g., temperature, humidity) to the server with Start control message. It may also require to stop sending observation data. It will be trigger by the application running on the server.
2.1. Constrained Application Protocol (COAP) message exchange

COAP messages are exchanged asynchronously between COAP endpoints [b]. In M2M interaction with COAP implementation, nodes act as both server and client roles. Using a Method Codes, a client sends COAP request on a resource (identified by a URI) on a server. Correspondingly, the server implements Response Code to send a response, which may include a resource implementation.

The COAP message exchange of JSON payload is illustrated in Figure 2. A Fog Node acts as an agent to facilitate the distributed scenario. The interaction between the Sensor Node and Server is managed by the Fog Node, or can proceed as in Figure 1. The Sensor Node sends a registration message to register itself. It waits for the control message to start sending the observation data.
2.2. Network Setup

A tree topology is used as shown in Figure 3.

(Sensor Node a)----+
   \                
(Sensor Node b)-------+(Fog Node1) (Fog Node2)
       /           /                /
   (Sensor Node c)------+
         \    \    /                /
            \  /    /                /
              \ /    /                /
                (Sensor Node d)-------+(Server)

Figure 3: A tree topology
All the nodes MUST be a COAP endpoint for message exchange. A COAP endpoint is capable of both client and server roles. A sensor node can directly interact with the server or via a Fog Node.

2.3 Message description in Fiesta-IOT
The message format complies with Fiesta-IOT ontology [c]. It maintains three level of service description as shown in Figure 4. The observation data is accommodated in the sensor level description.

```
+------------------------------------------------------+
    | <platform>                                           |
    |     <location>                                       |
    |        <lat></lat>                                   |
    |        <long></long>                                 |
    |   </location>                                        |
    +------------------------------------------------+  |
    |  +------------------------------------------------+  |
    |  | <system>                                       |  |
    |  |     <coverage></coverage>                      |  |
    |  |     <service></service>                        |  |
    |  |  +-------------------------------------------+ |  |
    |  |  |  <sensor>                                 | |  |
    |  |  |       <quantityKind></quantityKind>       | |  |
    |  |  |       <unit></unit>                       | |  |
    |  |  |       <observation>                       | |  |
    |  |  |        <measurementType></measurementType>| |  |
    |  |  |          <time></time>                    | |  |
    |  |  |          <sensorOutput>                   | |  |
    |  |  |          <value></value>                  | |  |
    |  |  |          <unit></unit>                    | |  |
    |  |  |        </sensorOutput>                    | |  |
    |  |  |       </observation>                      | |  |
    |  |  |  </sensor>                                | |  |
    |  |  +-------------------------------------------+ |  |
    |  |  </system>                                     |  |
    +------------------------------------------------+  |
    | </platform>                                           |
+------------------------------------------------------+
```

Figure 4: Message description
3. Distributed Fault Management

After collecting sensory data from sensors through Fog Node and Central Server, the distributed fault management MUST run the abnormal value fault detection algorithm to detect the fault sensors for a particular location. Then, the detected fault sensor nodes SHOULD be replaced with the supplementary sensor nodes that are currently off. These algorithms MAY be implemented in a centralized server (domain) and fog node at a particular location such as room, building.

3.1 Abnormal Value Fault Detection

Abnormal value fault detection is an important fault detection in IoT domain. In order to know whether values observed from IoT sensors are fault or not, the observation values of particular sensors SHOULD be compared with the values from the neighboring sensor nodes. A flow chart for abnormal value fault detection at a particular location is shown below. The abnormal value fault detection MAY be performed based on two parameters, the current distance of observation values, the distance between the previous and current distance of observation values. If the observation value of a particular sensor is similar to the majority of sensors observation values which means the distances are within the predefined threshold, that sensor MAY be detected as normal. Otherwise, it is a fault sensor.
Figure 5: Flow chart of abnormal fault detection
3.2. Sensor Replacement for Fault Sensors

The detected fault sensors SHOULD be replaced with the supplementary sensor nodes that are currently off for further analysis or monitoring. A flow chart for sensor replacement is shown below. For each fault sensor, an off sensor SHOULD be replaced by turning on it. If the sensor can be turned on, it is replaced with the fault sensor. Otherwise, the off sensor SHOULD be updated as malfunction sensor. Finally, the fault sensor MAY be turned off.
4. IANA Considerations

There are no IANA considerations related to this document.

5. Security Considerations

This note touches communication security as in M2M communications and COAP protocol.

6. References

6.1. Normative References


6.2. Informative References
Authors’ Addresses

Choong Seon Hong
Computer Science and Engineering Department, Kyung Hee University
Yongin, South Korea
Phone: +82 (0)31 201 2532
Email: cshong@khu.ac.kr

Minh N H Nguyen
Computer Science and Engineering Department, Kyung Hee University
Yongin, South Korea
Phone: +82 (0)31 201 2987
Email: minhnnn@khu.ac.kr

Shashi Raj Pandey
Computer Science and Engineering Department, Kyung Hee University
Yongin, South Korea
Phone: +82 (0)31 201 2987
Email: shashiraj@khu.ac.kr

Chit Wutyee Zaw
Computer Science and Engineering Department, Kyung Hee University
Yongin, South Korea
Phone: +82 (0)31 201 2987
Email: cwyzaw@khu.ac.kr

Seung Il Moon
Computer Science and Engineering Department, Kyung Hee University
Yongin, South Korea
Phone: +82 (0)31 201 2987
Email: moons85@khu.ac.kr