Workgroup: ANIMA WG
Internet-Draft:
draft-bernardos-anima-fog-monitoring-06
Published: 24 May 2022
Intended Status: Experimental
Expires: 25 November 2022
Authors: CJ. Bernardos, Ed. A. Mourad P. Martinez-Julia
UC3M InterDigital NICT

Autonomic setup of fog monitoring agents

Abstract

The concept of fog computing has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices. The term fog is referred to any networked computational resource in the continuum between things and cloud. In fog computing, functions can be stiched together composing a service function chain. These functions might be hosted on resources that are inherently heterogeneous, volatile and mobile. This means that resources might appear and disappear, and the connectivity characteristics between these resources may also change dynamically. This calls for new orchestration solutions able to cope with dynamic changes to the resources in runtime or ahead of time (in anticipation through prediction) as opposed to today's solutions which are inherently reactive and static or semi-static.

A fog monitoring solution can be used to help predicting events so an action can be taken before an event actually takes place. This solution is composed of agents running on the fog nodes plus a controller hosted at another device (running in the infrastructure or in another fog node). Since fog environments are inherently volatile and extremely dynamic, it is convenient to enable the use of autonomic technologies to autonomously set-up the fog monitoring platform. This document aims at presenting this use case as well as specifying how to use GRASP as needed in this scenario.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The concept of fog computing has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices. The term fog is referred to any networked computational resource in the continuum between things and cloud. A fog node may therefore be an infrastructure network node such as an eNodeB or gNodeB, an edge server, a customer premises equipment (CPE), or even a user equipment (UE) terminal node such as a laptop, a smartphone, or a computing unit on-board a vehicle, robot or drone.

In fog computing, functions might be organized in service function chains (SFCs), hosted on resources that are inherently heterogeneous, volatile and mobile. This means that resources might appear and disappear, and the connectivity characteristics between these resources may also change dynamically. This calls for new orchestration solutions able to cope with dynamic changes to the resources in runtime or ahead of time (in anticipation through prediction) as opposed to today's solutions which are inherently reactive and static or semi-static.

1.1. Problem statement

Figure 1 shows an exemplary scenario of a (robot) network service. A robot device has its (navigation) control application running in the fog away from the robot, as a network service in the form of an SFC "F1-F2" (e.g., F1 might be in charge of identifying obstacles and F2 takes decisions on the robot navigation). Initially the function F1 is assumed to be hosted at a fog node A and F2 at fog node B. At a given point of time, fog node A becomes unavailable (e.g., due to low battery issues or the fog node A moving away from the coverage of the robot). There is therefore a need to predict the need of migrating/moving the function F1 to another node (e.g., fog node C in the figure), and this needs to be done prior to the fog/edge node becoming no longer capable/available. Such dynamic migration cannot be dealt with in today's orchestration solutions, which are rather reactive and static or semi-static (e.g., resources may fail, but this is an exceptional event, happening with low frequency, and only scaling actions are supported to react to SLA-related events).

```
           --------------
           |    ====    |
          ------+F1+----------
         / |  | ==== |  |     \
        /  |  +------+  |      \
        |  | fog node C |       \
        |  --------------        \
        |                         \
        |       --------------  ---\----------
        |       |    ====    |  |   \====    |
        | -----------+F1+------------+F2|    |
        |/      |  | ==== |  |  |  | ==== |  |
        o       |  +------+  |  |  +------+  |
        |       | fog node A |  | fog node B |
--------+-      --------------  --------------
|        |
--0----0--
```

Figure 1: Example scenario

Existing frameworks rely on monitoring platforms that react to resource failure events and ensure that negotiated SLAs are met. However these are not designed to predict events likely to happen in a volatile fog environment, such as resources moving away, resources becoming unavailable due to battery issues or just changes in availability of the resources because of variations of the use of the local resources on the nodes. Besides, it is not feasible in this kind of volatile and extremely mobile environment to perform a continuous monitoring and reporting of every possible variable or parameter from all the nodes hosting resources, as this would not scale and would consume many resources and generate extra overhead.

In volatile and mobile environments, prediction (make-before-break) is needed, as pure reaction (break-before-make) is not enough. This prediction is not generic, and depends on the nature of the network service/SFC: the functions of the SFC, the connectivity between them, the service-specific requirements, etc. Monitoring has to be setup differently on the nodes, depending on the specifics of the network service. Besides, in order to act proactively and predict what might need to be done, monitoring in such a volatile and mobile environments does not only involve the nodes currently hosting the resources running the network service/service function chain (i.e., hosting a function), but also other nodes which are potential candidates to join either in addition or in substitution to current nodes for running the network service in accordance with the orchestration decisions.

In the example of Figure 1, the fog node initially hosting function F1 (fog node A) might be running out of battery and this should be detected before the node A actually becomes unavailable, so the function F1 can be effectively migrated in a time to a different fog node C, capable of meeting the requirements of F1 (compute, networking, location, expected availability, etc.). In order to be able to predict the need for such a migration and have already identified a target fog node where to move the function, it is needed to have a monitoring solution in place that instructs each node involved in the service (A and B), and also neighboring node candidate (C) to host function (F1), to monitor and report on metrics that are relevant for the specific network service "F1-F2" that is currently running.

1.2. Fog monitoring framework

Fog environments differ from data-center ones on three key aspects: heterogeneity, volatility and mobility. The fog monitoring framework is used to predict events triggering and orchestration event (e.g., migrating a function to a different resource).

The monitoring framework we propose for fog environments is composed of 2 logical components:

* Fog agents running on each fog node. An agent is responsible for sending the value of a variable or parameter to a fog monitoring controller and to other fog agents. What variable or parameter will be monitored and what data will be sent (including frequency) is configured per agent considering the specifics of the network service or SFC. A fog agent might also take some autonomous actions (such as request migration of a function to a neighbor node) in certain situations where connectivity with the fog monitoring controller is temporarily unavailable.

* A fog monitoring controller (e.g., running at the edge or at a fog node). This node obtains input from the orchestration logic (MANO stack) and autonomously decides what variables or parameters will be monitored, where will the data be collected, and how it will be done, based on the requirements provided by the orchestration logic managing the network services instantiated in the fog. This configuration is specific to a network service, a function, or an SFC as whole.

  - It interacts with the orchestration logic to coordinate and trigger orchestration events, such as function migration, connectivity updates, etc. In some deployments, this entity might be co-located with the orchestration logic (e.g., the NFVO).

  - It interacts with the fog agents to instruct what variables and/or parameters need to be monitored. It also interacts to get the resulting monitoring data. This interaction is not limited to fog agents at nodes currently involved in a given network service or SFC, but also includes other nodes that are suitable for hosting a function that needs to be migrated. This allows to provide the orchestration logic with candidate nodes in a pro-active way.

  - It is capable of autonomously discover and set up fog agents.

1.3. Supporting simple and complex monitoring metrics

Fog monitoring nodes will be capable of providing raw monitoring data as well as processed data. The former are obtained directly from the measured variables or parameters. The latter are obtained by applying some processing function to several monitoring data items. The fog monitoring controllers will specify the function to be executed, which data will be collected and processed by the functions, and the additional parameters that will control the processing and will determine the particularities of the output of each function.

The complexity of the functions that can be executed is arbitrary. They can be either pre-instructed in the fog agents or dynamically instructed by the requester (the fog monitoring controller) by providing the sequence to execute the functions and their input parameters.

Complex monitoring metrics, the processed data, can also be used as part of the condition that determines the distributed and autonomic actions. Thus, the logic that defines those actions is simplified and the actuation components can be concentrated on their task without requiring extra effort to process the raw monitoring data.

Adding support for complex monitoring metrics enables the fog monitoring framework to avoid the transmission of unneeded data and thus optimize its overall operation. For example, if the controller is interested in the average of the CPU load of a fog agent for the last 5 minutes, it can just request it, providing the period to average as input parameter and specifying the source from which measuring the CPU load variable.

2. Terminology

The following terms are using in ths document:

fog: Fog goes to the Extreme Edge, that is the closest possible to the user including on the user device itself.

fog node: Any device that is capable of participating in the Fog. A Fog node might be volatile, mobile and constrained (in terms of computing resources). Fog nodes may be heterogeneous and may belong to different owners.

orchestrator: In this document we use orchestrator and NFVO terms interchangeably.

3. Autonomic setup of fog monitoring framework

Fog nodes autonomously start fog agents at the bootstrapping, then start looking for other agents and the fog monitoring controller. This autonomic setup can be performed using GRASP. The procedure is represented in Figure 2. The different steps are described next:

```
+--------+    +--------+    +--------+
|  fog   |    |  fog   |    |  fog   |
| node C |    | node A |    | node B |                       +------+
|        |    |        |    |        |                       | fog  |
| |    | |    | |    | |    | |    | |        +------+       | mon. |
| +----+ |    | +----+ |    | +----+ |        | NFVO |       | ctrl |
+--------+    +--------+    +--------+        +------+       +------+
                   |             |                |              |
            (fog nodes A & B bootstrap)           |              |
                   |             |                |              |
                   |             |   periodic mcast advertisement|
                   |             |               (ID, fog_scope) |
                   |             |  <----------------------------+
                   | Mcast discovery (fog_node_ID, scope)        |
                   +-------------------------------------------->|
                   +------------>|                |              |
                   |    Mcast discovery (fog_node_ID, scope)     |
                   |             +------------------------------>|
                   |<------------+                |              |
                   |             |                |              |
                   |       Unicast advertisement (ID, fog_scope) |
                   |             |<------------------------------+
                   |<--------------------------------------------+
                   |             |                |              |
                   |    Unicast registration (ID, fog_node_ID    |
                   |             |            fog_scope, capab.) |
                   |             +------------------------------>|
                   +-------------------------------------------->|
                   |             |                |              |
            (fog nodes A & B registered)          |              |
                   |             |                |              |
(fog node C bootstraps)          |                |              |
     |             |             |                |              |
     | Mcast discovery (fog_node_ID, scope)       |              |
     +---------------------------------------------------------->|
     +-------------------------->|                |              |
     +------------>|       Unicast advertisement (ID, fog_scope) |
     |<----------------------------------------------------------+
     |<--------------------------+                |              |
     |<------------+    Unicast registration (ID, fog_node_ID    |
     |             |             |            fog_scope, capab.) |
     +---------------------------------------------------------->|
(fog node C registered)          |                |              |
     |             |             |                |              |
```

Figure 2: Autonomic setup of fog agents

*The fog monitoring controller is regularly sending periodic multicast advertisement messages, which include its ID as well as the scope for the advertisement messages (i.e., the scope of where the messages have to be flooded).

M_DISCOVERY messages are used, with new objectives and objective options. GRASP specifies that "an objective option is used to identify objectives for the purposes of discovery, negotiation or synchronization". New objective options are defined for the purposes of discovering potential fog agents with certain characteristics. Non-limiting examples of these options are listed below (note that the names are just examples, and the ones used have to be registered by the IANA):

- FOGNODERADIO: used to specify a given type of radio technology, e.g.,: WiFi (version), D2D, LTE, 5G, Bluetooth (version), etc.

- FOGNODECONNECTIVITY: used to specify a given type of connectivity, e.g., layer-2, IPv4, IPv6.

- FOGNODEVIRTUALIZATION: used to specify a given type of virtualization supported by the node where the agent runs. Examples are: hypervisor (type), container, micro-kernel, bare-metal, etc.

- FOGNODEDOMAIN: used to specify the domain/owner of the node. This is useful to support operation of multiple domains/ operators simultaneously on the same fog network.

An example of discovery message using GRASP would be the following (in this example, the fog monitoring controller is identified by its IPv6 address: 2001:DB8:1111:2222:3333:4444:5555:6666):

```
[M_DISCOVERY, 13948745, h'20010db811112222333344445555 6666',
["FOGDOMAIN", F_SYNCH_bits, 2, "operator1"]]
```

GRASP is used to allow the fog agents and the controller discovery in an autonomic way. The extensions defined above, together with the use of properly scoped multicast addresses (as explained below), allow to precisely define which nodes participate in the monitoring and to gather their principal characteristics.

*When a fog node bootstraps, such as nodes A and B in the figure, they start sending multicast discovery messages within a given scope, that is, the intended area that composes the fog. The definition of the scope depends on the scenario, and examples of possible scopes are:

- All-resources of a given manufacturer.

- All-resources of a given type.

- All-resources of a given administrative domain.

- All-resources of a given user.

- All-resources within a topological network distance (e.g., number of hops).

- All-resources within a geographical location.

- Etc.

Combination of previous scopes are also possible.

The discovery messages are multicast within the scope, reaching all the nodes that compose the specified fog resources. This can be done for example using well defined IPv6 multicast addresses, specified for each of the different scopes. This signaling is based on GRASP. Different IPv6 multicast addresses need to be defined to reach each different scope, using scopes equal or larger than Admin-Local according to [RFC7346].

* In response to multicast fog discovery messages, the fog monitoring controller replies with unicast messages providing its information.

* Fog agents can then register with a controller. The registration message is unicast, and includes information on the capabilities of the fog node, such as:

  - Type of node.

  - Vendor.

  - Energy source: battery-powered or not.

  - Connectivity (number of network interfaces and information associated to them, such as radio technology type, layer-2 and layer-3 addresses, etc.).

  - Etc.

Note that registration to multiple fog monitoring controller instances could also be possible if a fog node wants to belong to several fog domains at the same time (but note that how the orchestration of the same resource is done by multiple orchestrators is not covered by this invention). The defined mechanisms support this via the use of fog IDs and FOGNODEDOMAIN options.

*A fog node C bootstraps after nodes A and B are already registered. The same discovery process is followed by fog node C, but in addition to the regular advertisement, registration procedures described before, existing neighboring fog agents (such as A and B in this example), might also respond to discovery messages sent by bootstrapping nodes to provide required information. This makes the procedure faster, more efficient and reliable. In addition to helping the fog monitoring controller in the fog agent discovery process, fog agents learn themselves about the existence and associated capabilities of other fog agents. This can be used to allow autonomous monitoring by the fog agents without the involvement of the central controller.

4. IANA Considerations

TBD.

5. Security Considerations

TBD.

6. Acknowledgments

The work in this draft will be further developed and explored under the framework of the H2020 5G-DIVE project (Grant 859881).

7. Informative References

[RFC7346] Droms, R., "IPv6 Multicast Address Scopes", RFC 7346, DOI 10.17487/RFC7346, August 2014, <https://www.rfc-editor.org/info/rfc7346>.

Authors' Addresses

Carlos J. Bernardos (editor)
Universidad Carlos III de Madrid
Av. Universidad, 30
28911 Leganes, Madrid
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Alain Mourad
InterDigital Europe

Email: Alain.Mourad@InterDigital.com
URI: http://www.InterDigital.com/

Pedro Martinez-Julia
NICT
4-2-1, Nukui-Kitamachi, Koganei, Tokyo
184-8795
Japan

Phone: +81 42 327 7293
Email: pedro@nict.go.jp