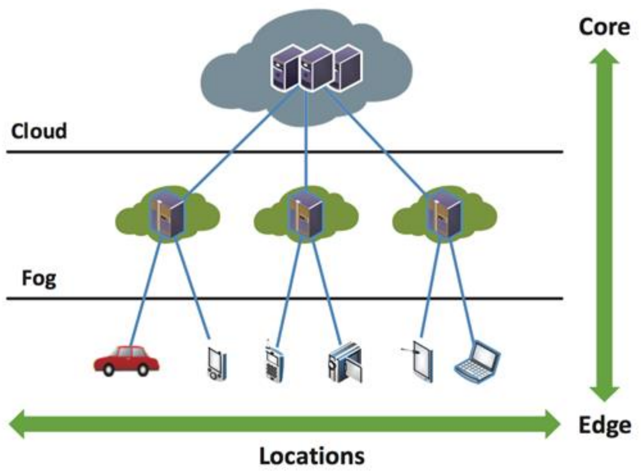
Introduction

In this project you will implement functionality of a node in a distributed system in the context of Fog Computing. Fog computing is a concept that aims to put the Cloud closer to the user for better quality of service (QoS). Fog computing is studied mostly in the context of Internet of Things (IoT) as it provides low latency, location awareness, support for wide-spread geographical distribution, and mobility support for IoT applications. Fog is not a substitute for but a powerful complement to the cloud.

A very simplistic model of a fog is shown in the figure below. In this architecture, the cloud is in the core, IoT nodes are at the edge of the Internet, and the fog sits between the cloud and the IoT nodes closer to the IoT nodes (normally within 1 or 2 hop distance). Fog nodes collaborate among each other and provide relevant cloud services such as computing, networking and storage to IoT devices. In this project, the main use of fog computing is to reduce the response time delay for requests coming from the IoT nodes.



When a fog node receives a request, depending on its current work load, it will either handle the request by itself or will forward it on. A request can be forwarded among the fog nodes at most *Forward_Limit* times. If a request is forwarded *Forward_Limit* times and it cannot be served by the last fog node (due to its current work load), the node will forward the request to the cloud for processing. The node processing the request (either a fog node or the cloud) will directly send its response back to the IoT node.

Protocol

According to the figure above, the system includes three types of nodes as (1) IoT nodes, (2) fog nodes, and (3) cloud node. In this project, your task will be to implement the functionality of IoT nodes, fog nodes, and the cloud node.

The operation of an IoT node is very simple; it has to generate requests every *interval* seconds (*interval* is an argument given when running the IoT code), choose a random neighboring fog node and send the request to it (the neighboring fog nodes will be given as arguments when running the IoT code). Also, an IoT node has to receive the response for its request and shows an appropriate message. Each request will carry its *Forwarding_Limit* and *Processing_Time*. The *Forwarding_Limit* is a randomly chosen number in the range of [2, 5] and the *Processing_Time* is a randomly chosen number in the range of [3, 7] in seconds.

The operation of a fog node includes two tasks: (1) receiving requests and, depending on its response time for this request, processing or forwarding the request and (2) periodically exchanging information with its 1-hop neighbors about their current queueing time so that they know who to forward a request when needed.

When a fog node decides to process a request, it will place it to the end of its processing queue that it keeps pending requests in. The decision on processing or forwarding a request will be based on the expected response time for the request. The response time includes queuing delay and processing delay for the request. Each request will include a parameter that will define the *Processing_Time* of the request. When a new request R_{new} arrives at a fog node A, using the *Processing_Time* info in R_{new}, the fog node A can calculate the queuing time for R_{new}. The queuing time will be given by the summation of the processing times of all the currently queued requests at A. As a result, the expected response time of R_{new} can be calculated by adding up the queuing delay of the fog node A and the *Processing_Time* of R_{new}.

When a request is up for processing at a fog node, the processing operation should be simulated by having a thread sleep or wait for processing time of the request. Once this time elapses, a response should be created and sent back to the request originator IoT node.

If a fog node decides that it is too busy to accept this new request (which will be defined by an input parameter of *Max_Response_Time* at each fog node), it will check if it can forward the request to a neighbor or not. This depends on the *Forward_Limit* of the request and how many times this request has been forwarded so far. If the request has not reached the *Forward_Limit*, the fog node will identify the best neighbor among its 1-hop neighbors and forward the request to that neighbor. The best neighbor is the one that reported the smallest response time in the most recent round with an exception that the best neighbor cannot be the neighbor who has forwarded this request to the current fog node.

If a fog node cannot accept a new request and the *Forward_Limit* is reached for the request, the fog node will need to forward this request to the cloud.

Selection of the best neighbor depends on the queueing time of the 1-hop neighbors. This information needs to be periodically exchanged among 1-hop neighbors so that they can use this up to date information to decide on their best neighbors to forward a request when needed. When the time comes to send a periodic queueing time information to 1-hop neighbors (every *t=3* seconds), a fog node will announce the queueing time that will be incurred for an incoming request at this very moment and will send it to its neighbors. Once each neighbor does this, everyone will be able to use this information to figure out their best neighbors.

The operation of the cloud node is very similar to the operation of a fog node; it has to put the forwarded requests in its queue and processes the requests from its queue one by one in a loop. Its processing operation should be simulated by having a thread sleep or wait for processing time of the request. Once this time elapses, a response should be created and sent back to the request originator IoT node. Note that the queue of the cloud node is assumed to be large enough so it can process every forwarded requests.

Given that a request may visit multiple nodes in the system, for debugging and testing purposes, you are required to come up with a scheme to carry necessary information on the message body. When the corresponding response arrives at the IoT node, the IoT node will print it out. The information should clearly indicate which fog nodes are visited and what each fog node did on the request. This information can be stored as a text string in the message body while it propagates in the system. Each fog node can append its own message to the end of the string. The node sending a response for this request, can copy over the text string into the response message and append its own information to it before sending it back to the requesting IoT node.

To avoid the loop scenario where a fog node sends back a request that it has received from its neighbors to the same neighbor (loop of two fog nodes), you will assume that the fog node offloads the request to its best neighbor, not considering to the neighbor it has received the request from. In this case, if the best neighbor is the neighbor that the request is just received from and an offload is needed, the fog node sends the request to the second best neighbor.

Implementation Details

For implementation, you are allowed to use C, C++, Java, or python. TCP or UDP sockets must be used for communication between nodes.

We believe that the project involves implementing three protocol operations between the nodes as below:

1) Periodic response time update messages among the fog nodes
2) IoT node to fog/cloud node request/response communication
3) Fog to fog/cloud request offloading

The operations (1) and (3) above should be implemented over TCP sockets and the operation (2) should be implemented using UDP sockets.

The IoT node will send periodically packets to randomly chosen fog nodes it has received as the input. The IoT node will be run in the terminal with the following command:

```
IoTnode interval MY_UDP IP1 UDP1 IP2 UDP2 …
```

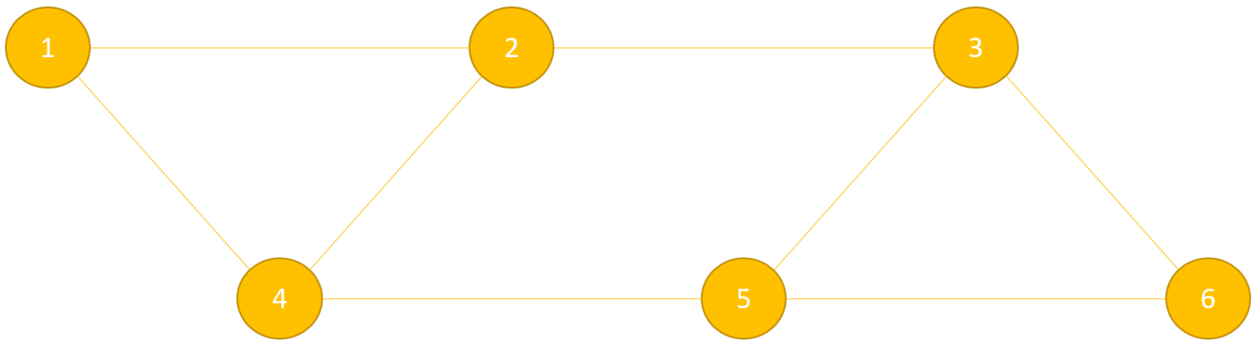
`interval` is the time between sending packets (in ms), `MY_UDP` is the UDP port number the IoT node is listening for incoming UDP messages, `IP1` is the IP address (or hostname) of the first fog node, `UDP1` is the UDP port of the first fog node on which the fog node listens for incoming data from IoT layer. `IP2` is the IP address (or hostname) of the second fog node, `UDP2` is the UDP port of the second fog node on which fog node listens for incoming data from IoT layer. There could be more than 2 fog nodes.

The information that IoT node puts in the packet are:

- Packet sequence number
- Request *Forwarding_Limit* and *Processing_Time*
- IP address of the IoT node
- UDP port number of IoT node

Note that the last two pieces of information are necessary for sending the reply back to the IoT node from fog/cloud nodes; fog/cloud nodes should know where they must send the reply to. This information will be recorded in these two pieces of information in the packet.

The following graph describes the topology of the fog nodes in the fog layer (different topology may be used during project demo).



A fog node will be run in the terminal with the following command:

```
FogNode Max_Response_Time t MY_TCP MY_UDP C TCP0 N1 TCP1 N2 TCP2 …
```

The first three arguments are the parameters of the protocol discussed in this document before. We set *t=3* in this project. *MY_TCP* is the TCP port number on which the fog node receives connection requests from other fog/cloud nodes, *MY_UDP* is the UDP port number on which the fog node receives requests from IoT node layer. *C* it the IP address (or hostname) of the cloud node and *TCP0* is the TCP port number that the cloud node is listening on. *N1* is the IP address (or hostname) of the fog's first neighbor and *TCP1* is the TCP port number that the first neighbor is listening on for accomplishing operation (1) and (3) discussed before. *N2* is the IP address (or hostname) of the fog's second neighbor and *TCP2* is the TCP port number that the second neighbor is listening on for accomplishing operation (1) and (3) discussed before. One fog node could have more than 2 neighbors. Note that the TCP port numbers are the listening port numbers in case a neighbor wants to establish the TCP connection to this fog node.

The cloud node will be run in the terminal with the following command:

```
CloudNode MY_TCP
```

MY_TCP is the TCP port number on which the cloud node receives requests from fog node layer.

Submission

This is a group project and groups of up to two (2) people are allowed in one team.

In the first step, you have to come up with a design and submit a design documents. The TA will go over the design documents and if necessary, set meetings with the group members to make sure that everyone is doing the right thing.

For submitting the final project, make sure you submit all the source codes to e-learning. (One project submission per group). Be sure to include sufficient comments. Include a README file specifying instructions to run/compile the project, platform, compiler, group information, etc.

Along with the code you are required to submit a short report. The report must in include in high level how you implemented the main functionalities, and any interesting challenge you faced and would like to discuss how you solved it. The report also should include a section showing how each person contributed to the project. The report shall be at most 3 pages.

You will be asked during the project demo session to download your program from e-Learning and run it on some distributed machines (*dcXY.utdallas.edu* machines, where *XY* can be numbers between *01* through *45*) for testing. We will run the topology for a certain configuration and go through the node's outputs to see your protocol works properly.