1 System architecture

1.1 Overview of basic system components

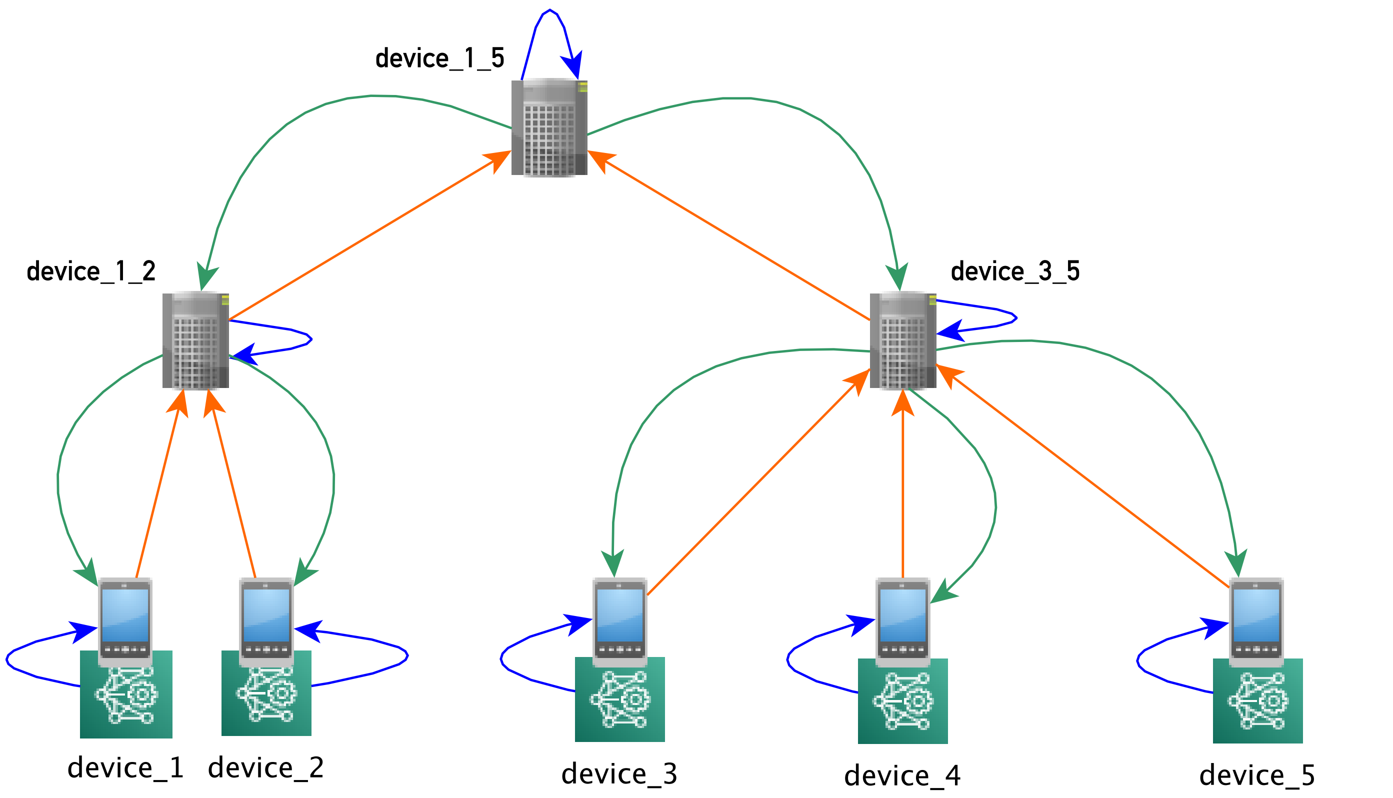

Figure 1.1: Overview of system architecture

The above diagram describes a typical system consists of 2 basic components:

- Client - end devices: this component plays an important role in the proposed system. It is responsible for sensor data collection, pre-processing (data filtering and normalization) and the local model training / retraining. The IoT nodes in this layer are constituted by heterogeneous devices, from mobile devices to many kind of sensors such as streetlamp, camera sensor, etc.
- Edge servers: each group of (near by) end devices are grouped and connected to a particular edge server. This grouping action can be done based on the context information of the end devices such as their location, their common characteristics, etc. The edge server mainly acts as a central point for sharing the model between client nodes within its group. It collects updated weights from the client nodes, performs weight aggregation using weighted averaging then sends back to clients.

The class distribution for this basic system configuration:

- $device_1$: class $[0, 1]$

- $device_2$: class $[2,3]$
- $device_3$: class $[4,5]$
- $device_4$: class $[6,7]$
- $device_5$: class $[8,9]$

1.2 Multi-level architecture

Starting from the base system introduced in the first part, I put it into a more flexible architecture. The basic overview of this multi-level system is demonstrated in Figure 1.2. Concretely, the edge servers can be grouped together in a similar way as client grouping in order to form a structure of multi-level hierarchical tree. The class distribution for this basic system configuration:

- $device_1$: class [0:3,1:3]
- $device_2$: class [1:3,0:3]
- $device_3$: class [2:3,3:3]
- $device_4$: class [3:3,2:3]
- $device_5$: class [4:3,5:3]
- $device_6$: class [5:3,4:3]
- $device_7$: class [6:3,7:3]
- $device_8$: class [7:3,6:3]
- $device_9$: class [8:3,9:3]
- $device_1$0: class [9:3,8:3]

Note: the pattern of class definition for each node follows this patter: *class id : number of instances per class*

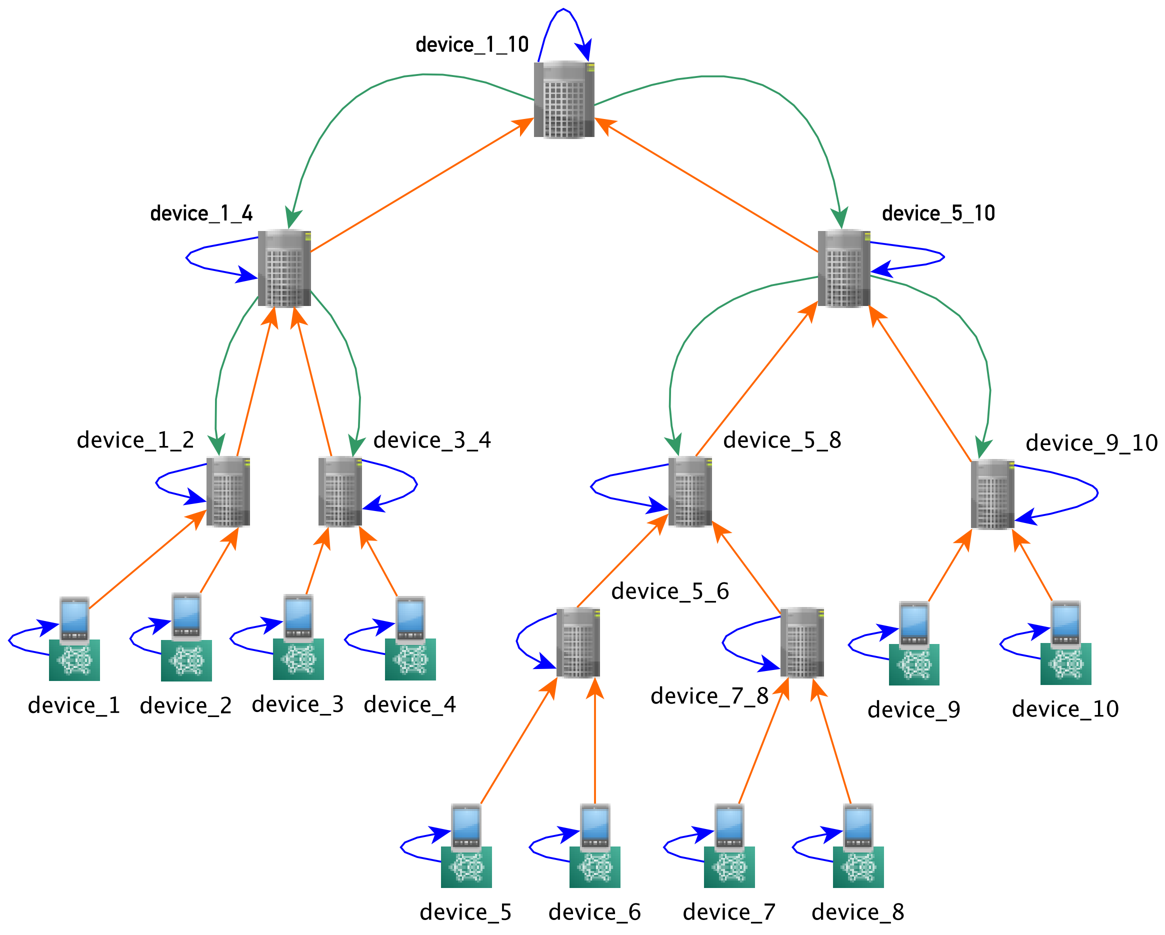

Figure 1.2: Multi-level architecture

1.2.1 Model update flow

As shown in Figure 1.3, the model weights sharing flow will be processed as follow:

- At the local model training / retraining process, the gradients and the weights of the model are recomputed at the end device. Model parameters will then be sent to the associate edge device.
- Each edge device is responsible for collecting updated parameters (weights) from local client devices within its region, averages them in order to update the weight of the "local partial model". Now we have 2 cases:
  - if this edge device is at the middle level of the hierarchy, the averaged weights are then forwarded to connected edge node at the upper layer to be aggregated once again.
  - if the current edge device is at the top layer of the hierarchy, it performs the aggregation and sent back the averaged weights to the lower level.
- The final averaged weights are sent back to the edge servers and clients following the path it has been forwarded before.

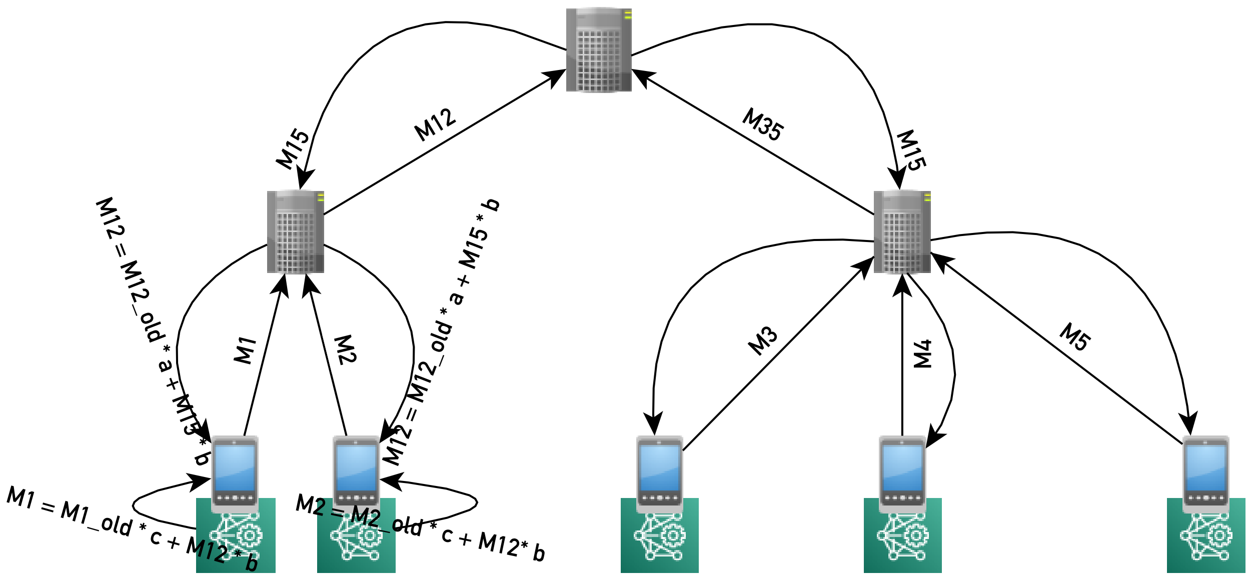
M15
M12
M35
M15
M12 = M12_old * a + M15 * b
M1
M2
M12 = M12_old * a + M15 * b
M3
M4
M5
M1 = M1_old * c + M12 * b
M2 = M2_old * c + M12* b

Figure 1.3: Model update process

- The parameters a, b and c are currently defined as follow: $\begin{aligned} a &= [1 \div 2] \times [1 \div n] \\ b &= 1 \div 2 \\ c &= 1 \div 2 \times 0.1 \end{aligned}$
- Note: n is the number of client devices connected to the edge node