Evolution of Mist Computing from Fog and Cloud Computing

The document summarizes the relation of *Mist Computing* to existing *Fog* and *Cloud Computing* architectures and the benefits of the individual computing architectures.

Fog and Mist computing

While *Fog computing* is the answer to many challenges in the IoT domain, such as high bandwidth requirements and manageability of applications, this concept can be further extended by pushing the computation to the end devices. In *Fog computing* the gateway bears the responsibility for IoT application execution, regardless whether the application is just simple data collection or building automation with many actuation tasks. Assembling the application logic to a gateway has many advantages, such as simplicity of coordination (the centralized control paradigm used with *Fog computing* is very similar to conventional programming paradigms), simple management of application logic (the application logic is all concentrated to a single device) and having access to macro-level information in the application as data from all nodes is collected to the gateway.

However, this approach can also have some drawbacks, such as increased delays in applications involving control, unnecessary high bandwidth requirements as all data must move through the gateway. The gateway is a single point of failure for applications that must be executed on the network and the operation of the entire network is dependent on the gateway.

Mist computing

Mist computing takes *Fog computing* concepts even further by pushing appropriate computation to the very edge of the network, to the sensor and actuator devices that make up the network. With *Mist computing* the computation is performed at the edge of the network in the microcontrollers in the embedded nodes. The *Mist computing* paradigm decreases latency and further increases the autonomy of a solution.

By applying the principles of service-based architecture among the end devices the application can be described as a combination of services, which are dependent on each other. Any device that has access to the network can subscribe to a service that is offered by any of the devices on the network. Hence in a temperature control application a heating unit can subscribe to temperature information from all the temperature sensors that are located in the room, which the heating unit is heating. The sensors can start providing their temperature data directly to the heating unit using the interval specified by the heating unit (the interval being dependent on the properties of the room and the power of the heating unit, and the relation of the parameters can be determined automatically at run time by the heater). No human involvement is therefore needed for setting up the application, simplifying network configuration.

Each device in the network must be aware of its location as most applications tend to be location dependent. The necessary ‘location awareness’ can be created at installation time (by “telling” the device, what its location), or the devices can determine their location autonomously by determining their location relative to some existing beacons, with known locations (for example a light fixture in a room may be aware of the room where it is located and all other devices in the room can determine their location based on proximity to the light fixture). Thinnect will be able to support this relative location functionality and incorporate this into *Mist*.

www.thinnect.com
440 No Wolfe Rd
Sunnyvale, 94085

The services provided by end devices may also be requested by mobile devices or servers, in which case the service request reaching a specific network is routed to the device, which is able to provide the specific service. This means that in one network end devices and a gateway may be both providing services to the same server. As an example in a building automation scenario we may be interested in room occupancy information for every single room, so all the occupancy sensors in the individual rooms must report information directly to the server, while the operation times of standalone AC units may be aggregated (in the gateway) to estimate the total power usage of AC units in the building.

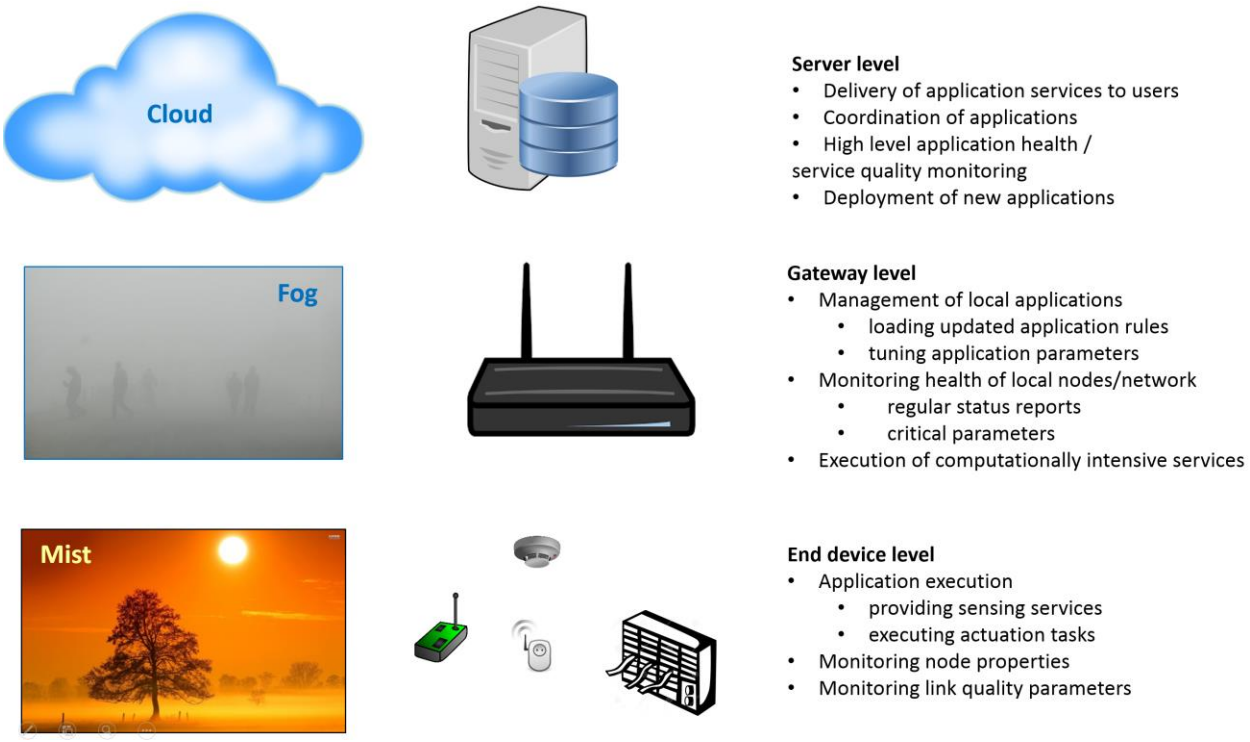

Figure 1 Distribution of tasks between *Cloud, Fog* and *Mist Computing*

Feasibility of Fog and Mist (and Cloud) co-existence

Fog computing was introduced because it is clear that the billions of IoT devices being deployed over time cannot operate by simply having connectivity to servers, instead the computation is pushed closer to the edge of the network – to the gateways. However, there is an opportunity to further distribute the computation tasks among the network nodes, based on their capabilities and roles any given application.

Significant computational power is available at the edge of the network – the microcontrollers at the edge in the majority of today's nodes have significant computational power, in excess of that needed to implement their basic IoT functions. It would be healthier for the entire ecosystem if all the ecosystem players were able to provide more functionality and more capable microcontrollers for the network nodes.

Clearly Fog and Mist (and Cloud) computing are complementary to each other – the application tasks, which are more computationally intensive can be executed in the gateway while the less computationally intensive tasks can be executed in the end devices. As an application in Mist computing is a collection of hierarchical services the services can be distributed among the computing nodes liberally.

Distribution of tasks among devices

The tasks and services must be distributed among the network devices based on their capabilities and the application logic. Clearly localized tasks such as providing sensor data for actuation or local data aggregation can be performed among a small subset of nodes at the edge, while macro-level tasks involving many nodes across the network should be executed at centralized devices, such as gateways.

Network/application management tasks (e.g., monitoring the health of the nodes, the health of the network) are also exposed as services, so the gateway can subscribe to such services from the nodes to monitor the health of the nodes and the network. The management of the end devices is not dependent on specific gateways, instead, gateways can collaboratively share the end device management information, regardless of which gateway a device reports to, this gateway can take over the management of the device.

Application optimization services should also be executed by centralized devices, such as gateways or servers. Such services require subscription to sensor data services for collecting data for the optimization task. Once enough data has been collected changes for application operation rules can be computed and loaded to the actuators, changing the behavior of the application.

Routing in the Mist

As in *Mist computing* the nodes need to talk directly to each other, the mainstream routing protocols that have been used in wireless mesh networks are not well applicable as these protocols are targeted for scenarios where the bulk of the communication occurs between the gateway and the end devices. Even in the case of the RPL protocol with P2P extensions, the direct routing between devices is sub-optimal as it relies on the gateway oriented graph structure.

In *Mist computing* the routing protocol supports direct device-to-device connectivity as sub-optimal routing paths will increase the total bandwidth requirements of the network. Also the network should support several gateways and any node could use any gateway for internet connectivity, eliminating dependence on a specific gateway.

In a large network consisting of hundreds of nodes new gateways may be added at any time. If connectivity to a known gateway is not available (because of connectivity loss of intermediate nodes or unavailability of gateway) but a node must deliver data to a server, a route must be established to a new gateway.

www.thinnect.com
440 No Wolfe Rd
Sunnyvale, 94085

Thinnect offering

Thinnect has implemented a Mist computing stack from PHY to the application level. The stack consists of a clustered, Layer 2 protocol which minimizes collisions, DYMO routing protocol, a service based application layer and rule based application logic framework. The entire stack takes about 100 kBytes (this includes the TinyOS operating system on top of which our stack currently runs).

The solution is based on more than a decade of academic research at the Research Laboratory for Proactive Technologies at Tallinn University of Technology and seven years of engineering in Defendec (where the Thinnect Mist networking technology has been created). Mist has been deployed and field tested by Defendec in a security application (for 6 years) and more recently by Cityntel in a Smart City application in multiple countries.

This document summarizes the main aspects of Fog and Mist computing in a high level fashion. If you would like to receive further information, please contact

Jurgo Preden,
CEO
jurgo@thinnect.com