# Chapter 6
# Fog Computing


## 6.1 Defining Fog Computing

In order to define Fog computing, a recap of the concept of Cloud computing is in order. Cloud computing refers to a model that provides users with on-demand access of a shared pool of computing resources over a network. These resources can be quickly provisioned and released through a self-service model. One of the key characteristics of the Cloud computing model is the notion of resource pooling, where workloads associated with multiple users (or tenants) are typically colocated on the same set of physical resources. This guarantees the economy of scale of the Cloud computing model. Hence, essential to Cloud computing is the use of network and compute virtualization technologies. Cloud computing provides elastic scalability characteristics, where the amount of resources can be grown or diminished based on user demand.

Fog computing, or in short Fog, refers to a platform for integrated compute, storage and network services that are highly distributed and virtualized. This platform can extend in locality from IoT end devices and gateways all the way to Cloud data centers but is typically located at the network edge. Fog augments Cloud computing and brings its functions closer to where data is produced (e.g., sensors) or needs to be consumed (e.g., actuators). Fog is not an alternative to Cloud computing; rather the two synergistically interplay in order to enable new types and classes of IoT applications that otherwise would not have been possible when relying on Cloud computing stand-alone (Fig. 6.1).

## 6.2 Drivers for Fog

There are several IoT requirements that act as the drivers for the Fog architecture. These will be discussed next.

© Springer Nature Switzerland AG 2019

A. Rayes, S. Salam, *Internet of Things From Hype to Reality*,
https://doi.org/10.1007/978-3-319-99516-8_6

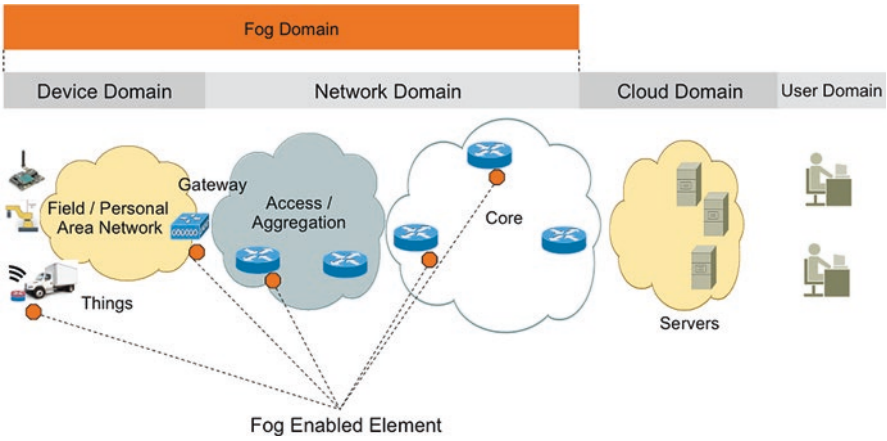



**Fig. 6.1** Fog and Cloud

### *6.2.1 Data Deluge*

It has been claimed that five exabytes of data have been generated from the dawn of humanity to 2003.[1] Now this much data is generated every 2 days[1], and the rate is only increasing. The billions of devices that are projected to be connected to the Internet will only exacerbate the data deluge problem. At heart of the issue is the question of whether the state of the art will evolve fast enough to handle the imminent explosion of data? There are two technology evolution curves at play here: one represents the evolution of compute and storage technologies, which is governed by Moore's Law, and the second represents the growth of bandwidth at the network edge, which is covered by Nielsen's Law. Moore's Law stipulates that compute and storage technologies will double in capability/capacity every 18 months. Nielsen's Law, on the other hand, projects that the bandwidth at the network edge doubles every 24 months. Acknowledging that there is a positive correlation between the growth of compute and storage technologies and the growth in data volume, it is conceivable to foresee an IoT future where data will be produced at rates that far outpace the network's ability to backhaul the information, from the network edge where it is produced by the billions of Things, to the Cloud where it will ultimately need to be processed and potentially stored. This disparity between the data volume and the available bandwidth is best exemplified with the analogy of attempting to push a golf ball through a straw. Luckily, Moore's Law is not only a culprit by contributing, in part, to the problem but is also a key enabler to the solution: it can be leveraged to augment the functions of the network itself with compute and storage capabilities at the edge. This allows the network to perform processing, analysis, and storage of data in lieu of blindly pushing all data

[1] As quoted by Eric Schmidt, Executive Chairman of Google

up to the Cloud. With that, Cloud computing is brought closer to the data sources, the Things, which gives rise to the notion of Fog computing. *Cloud* becomes *Fog* when it is closer to *Things*, pun intended.

### 6.2.2 Rapid Mobility

Certain IoT use cases require support for rapid mobility of Things, for example, sensors on a speeding vehicle communicating with roadside infrastructure or a passenger commuting on a train. Due to rapid mobility, network conditions may vary frequently, due to signal fading, interference, or other conditions. This may even lead to severe service degradation or intermittent loss of connectivity to the Cloud. Another consideration is the characteristics of the communication path to the Cloud: bandwidth and/or latency limitations may have adverse side effects on the operation of the IoT application. Multiple variables will typically be at play to contribute to these characteristics, including radio coverage, interference, and the amount of resources shared with other mobile nodes.

To guarantee the quality of service and reliability required by the application, especially when dealing with mobility over extended geographic distances, the Cloud infrastructure needs to be augmented with compute and storage functions that move with the mobile Things. The mobility of these functions may be either physical or virtual. In the former case, the compute and storage are physically situated with the moving Thing, whereas in the latter, these functions maintain close proximity by shadowing and following the Thing albeit in the network edge. In this capacity, Fog augments the Cloud to achieve the required pervasiveness and reliability required by rapid mobility in IoT.

### 6.2.3 Reliable Control

IoT applications that focus on closed-loop control and actuation often share the following characteristics: the data input space and the processing logic required to produce the control decision have intensive computational and considerable storage demands. The sensing and actuating devices are typically constrained devices and therefore need to offload the storage and compute functions to external systems or infrastructure. In many cases, these control applications require very low latency for correct operation. In a subset of the scenarios, connectivity to the Cloud may be either too expensive (e.g., satellite links connecting sensors deployed in oilfields) or unreliable due to rapid mobility patterns.

The combination of the above characteristics makes it unpalatable to rely on Cloud computing to support reliable real-time control with fixed latency. This is where Fog computing can complement the Cloud to address that IoT application niche.

### 6.2.4 Data Management and Analytics

A class of IoT applications characterized with the confluence of very large scale, in terms of the number of devices generating data, widespread geographic footprint where these devices are deployed, vast amounts of data that need to be collected, aggregated, processed, and exposed to consuming entities, as well as real-time analytics or closed-loop control. For such class of applications, a data management and analytics platform that can handle the scale and performance requirements is needed. Experience with large-scale information and communication systems has proven that distributed systems built on hierarchical division of functions provide the elasticity required while maintaining key performance metrics. Such systems typically exploit locality of data for their most basic functions. In other words, they tend to minimize the amount of data required from remote sources for critical functions. Interactions between widespread entities are typically confined to system wide functions. For data management and analytics, this operating paradigm is even more relevant because the IoT data often needs to be operated on within a context, which is well known at the edge of the network, close to the data sources, and is often lost or is irrelevant as the data travels deeper in the network and into the Cloud. Take as an example an ambient noise sensor in a smart city application, which is constantly measuring noise levels and streaming the recorded data. Backhauling all the data to the Cloud is both unnecessary and inefficient, especially when compared with an alternate design where a local analytics function situated close to the sensor filters readings below a specified threshold (depending on the context associated with where the sensor is deployed) and only propagates to the Cloud interesting readings above that threshold, e.g. to alert city personnel.

Fog computing, in concert with Cloud computing, provides the necessary compute and storage infrastructure required to support such distributed and hierarchical data management and analytics.

## 6.3 Characteristics of Fog

The Fog and the Cloud both comprise of the same three building blocks: compute, storage, and networking. However, there are multiple characteristics that uniquely shape the Fog and distinguish it from the Cloud:

First are the network edge location, location awareness, and low latency. Fog locates the services close to the data sources and consumers where it is possible to enrich the data with location context and operate on it with minimal latency.

Second is geographical and architectural distribution. This is in stark contrast to the Cloud model where are all services are centralized in the data center.

Third is the extremely large number of nodes. While the Cloud drives demand for massively scalable data centers (MSDC), the Fog pushes the envelope further on scalability.

Fourth is mobility of nodes and endpoints. The data sources, consumers, compute, or storage resources can all be mobile.

Fifth is real-time interaction. In the Fog, the focus is on real-time analysis of streaming data as opposed to batch processing. Fog requires analysis of Data in Motion as opposed to Data at Rest.

Sixth is predominance of wireless access. In the Cloud, connectivity relies on wireline technologies, predominantly Gigabit Ethernet (10Gbps, 40Gbps, and soon 100Gbps). The Fog will be mostly connected over wireless links, both because of the impracticality of running wires everywhere and to support the mobility requirements.

Seventh is the heterogeneity of resources. In the Cloud, a given data center is managed by a single business entity, which goes about deploying homogeneous resources in order to minimize complexity and operational costs. With the Fog, the architecture is federated over resources managed by different business entities. Hence, these resources will vary widely in capabilities, form factors and operating environment.

The table below summarizes the main facets of difference between Cloud and Fog computing (Table 6.1).

## 6.4 Enabling Technologies and Prerequisites

The realization of the vision of Fog computing relies on a number of technologies that provide enabling building blocks and are key prerequisites for the architecture. These include lightweight compute virtualization, network mobility, orchestration, and application enablement technologies. In what follows, we will discuss each of those technologies in more detail.

**Table 6.1** Summary comparison of Cloud and Fog computing

| Requirement | Cloud computing | Fog computing |
|---|---|---|
| Latency and jitter | High/medium | Low |
| Location of service | Within Internet | Network edge |
| Distance between data sources/ consumers | Multiple hops | Single hop |
| Location awareness | No | Yes |
| Geo-distribution | Centralized (data center) | Distributed |
| Number of nodes | Large | Larger |
| Support for mobility | No | Yes |
| Data analytics | Data at Rest | Data in Motion |
| Connectivity | Wire line | Wireless |

### 6.4.1 Virtualization Technologies

Inherent to Fog computing is the ability to locate compute functions close to data producers and/or consumers. This assumes the availability of lightweight compute virtualization technologies that allow workloads to be instantiated, as needed, on Fog nodes. The latter act as shared compute resources among potentially a multitude of IoT applications.

Virtualization technologies combine or partition computing resources to present one or more operating environments using techniques such as hardware and software partitioning or aggregation, hardware emulation, resource sharing or time multiplexing, etc. Virtualization provides a number of advantages: It enables consolidation of both hardware and applications, thereby eliminating the expense associated with procuring and managing underutilized infrastructure. It also enables sandboxing, i.e., providing application with secure isolated execution environments. Virtualization also provides the flexibility of supporting multiple simultaneous operating systems over the same hardware infrastructure. It eases the migration of software stacks and allows the packaging of applications as stand-alone appliances. Furthermore, virtualization enables the portability and mobility of applications from one hardware or physical location to another with ease.

Virtualization technologies generally differ in the abstraction level at which they operate: CPU instruction set level, hardware abstraction layer (HAL) level, and operating system level.

Virtualization at the CPU instruction set level allows an "emulator" to provide to an application the illusion of running on one processor architecture, whereas the real hardware actually belongs to a different architecture. It is the job of the emulator to translate the guest instruction set (offered to the application) to the host instruction set (used by the actual hardware).

Virtualization at the hardware abstraction layer level involves a virtual machine manager, or hypervisor, which is a software layer that sits above the physical hardware (sometimes referred to as "bare metal") and provides a virtualized view of all its services. The hypervisor can create multiple virtual machines (VMs) on top of the bare metal. The VMs can be running different operating systems. Applications can run within their respective operating systems and are completely oblivious to the underlying virtualization.

Virtualization at the operating system level relies on virtualization software that runs on top of or as a module within the operating system. It provides an abstraction of the kernel-space system calls to user-space applications, in addition to security and sandboxing capabilities to prevent one application from causing collateral damage to another.

Other higher levels of virtualization are possible, such as library and application level virtualization, but these are not relevant for the purpose of this discussion.

#### 6.4.1.1 Containers and Virtual Machines

Both containers and virtual machines are popular virtualization constructs employed in Cloud computing today. Each of the two technologies has its own set of advantages and trade-offs. Virtual machines (VMs) are a virtualization technology at the hardware abstraction layer level. VMs provide an abstraction of a compute platform's hardware and software resources, complete with all the drivers, full operating system and needed libraries. Containers, on the other hand, are a virtualization technology at the operating system level. They include portions of the operating system and select libraries: the minimal pieces that are absolutely required to run the application. Containers share the same operating system and, where applicable, common libraries. Due to this, containers are lighter-weight when compared to VMs, both in terms of their memory as well as processing requirements. As a result, given a specific hardware (e.g., a server) with a fixed resource profile, it is possible to support more containers than VMs running concurrently. This gives containers a clear scalability advantage over VMs, not only for Cloud computing but also for the Fog. In fact, the compact memory footprint for containers gives them another advantage in the Fog context: they are faster to migrate from one hosting node to another, a matter which characterizes them with the nimbleness required to support rapid mobility (Fig. 6.2).

However, the lightweight nature of containers comes with a set of trade-offs: since containers share the same underlying operating system, it is not possible to

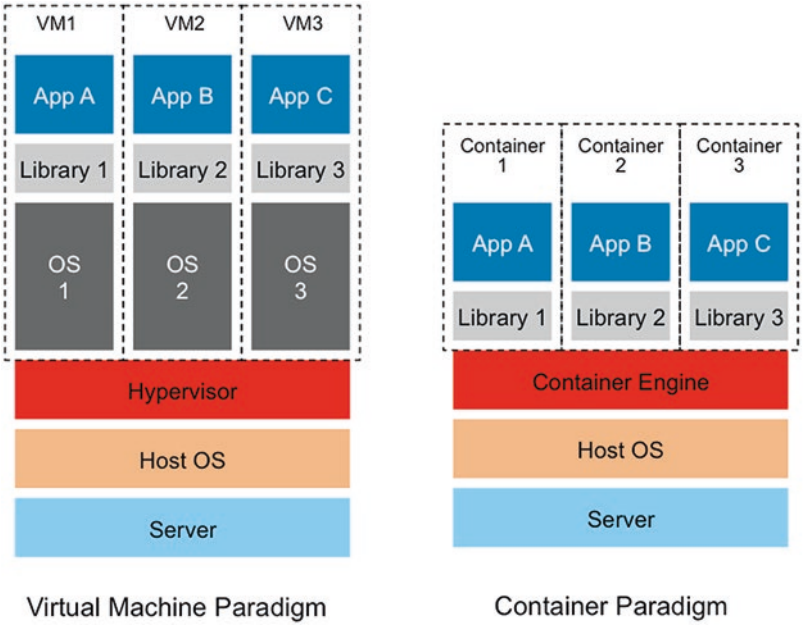



**Fig. 6.2** VMs and containers

use them to deploy applications that require disparate operating system environments, or different OS versions, on the same physical hardware. Such restriction does not apply to virtual machines, since they include their own copy of the operating system. Another trade-off associated with the shared operating system in containers is the security implications: there is potential for an application in a container to be subjected to security threats due to malicious or misbehaving code running in another container on the same operating system. With virtual machines, the security threat is smaller in comparison, because the attack surface is minimized due to the fact that each VM has an independent operating system instance. Therefore, an application in one VM is better sandboxed and isolated from applications or code running in another VM.

Linux, the leading open operating system platform, supports both virtual machines and containers. Both kernel-based virtual machines (KVM) and Linux containers (LXC) are available in the standard distribution.

Containers and VMs both provide the capability to sandbox Fog applications from one another and to control their resource usage. In addition to these relatively low-level functions, Fog requires a framework for the packaging, portability, sharing, and deployment of applications. One such framework that has been gaining popularity in the industry is Docker, which will be discussed next.

#### 6.4.1.2 Docker

Docker is an open source project that provides a packaging framework to simplify the portability and automate the deployment of applications in containers. Docker introduces scripts composed of a series of instructions that automate the deployment process from start to finish. These scripts are referred to as "Dockerfiles". Docker defines a format for packaging an application and all its dependencies into a single portable object. The portability is guaranteed by providing the application a runtime environment that behaves exactly the same on all Docker-enabled machines. Docker also provides tooling for container version tracking and management. In addition, it provides a community for sharing useful source code among developers.

#### 6.4.1.3 Application Mobility

Virtualization technologies decouple the application software from the underlying compute, storage, and networking resources. As such, it enables unrestricted workload placement and mobility across geographically dispersed physical resources. For instance, multiple hypervisors support different flavors of virtual machine migration, including "cold" migration and "live" migration. In the former case, a VM that is either powered down or suspended is moved from one host to another. In the latter, a VM that is powered on and operational is moved across hosts, without any interruption to its operation. The VM mobility solution takes care of moving the

VM's memory footprint and if applicable, any virtual disk/storage from the old to the new hardware. In order to ensure seamless mobility in the case of "live" migration, the VM retains its original Internet Protocol (IP) and Medium Access Control (MAC) addresses. This ensures that any clients or services that are in communication with the migrating VM can continue to reach it using the same communication addresses. The successful orchestration of such seamless live migration requires the underlying network infrastructure to support mobility. This will be the topic of the next section.

### *6.4.2 Network Support for Mobility*

As previously discussed, rapid mobility is one of the drivers for Fog computing. To ensure uninterrupted operation of the IoT application, the network infrastructure that is providing the underlying communication fabric for the Fog deployment must support seamless mobility of the communicating endpoints.

Networking systems rely on the address of the endpoints in order to deliver messages to their intended recipients. Depending on the technology at hand, the address either connotes the identity or the location of the endpoint. For example, Media Access Control (MAC) addresses are identity addresses, because they are burnt into the machine and uniquely identify it on a network. Internet Protocol (IP) addresses, on the other hand, are typically used as location addresses because they indicate the geographic locality of the endpoint. In some contexts, IP addresses are used as identity addresses as well, for example, in wireless mobile IP applications.

Applications that are deployed in a virtualization construct, such as a virtual machine, can perform seamless mobility. With seamless mobility, the application's MAC and IP addresses remain unchanged as the associated VM moves from one physical server node to another. The network infrastructure needs to handle the application mobility event and update the forwarding information on the routers and/or switches to deliver the messages correctly to the right physical server that is now hosting the VM. In order to do this, the network infrastructure needs to treat the VM's IP and MAC addresses as identity addresses, and correlate them with dynamic location addresses that get updated automatically as the VM moves from one locality to another. In order to properly scale the solution, the knowledge of identity addresses should be confined to the edge of the network, whereas the core of the network performs forwarding solely based on the location addresses. This is achieved by relying on tunnels established between the edge nodes of the network to forward the end-host traffic over the core. The tunnel encapsulation uses location addresses and hides identity addresses from the core network nodes. The correlation between identity addresses and location addresses is established through a mapping service provided by the network infrastructure. In a way, this is similar to how the post office mail forwarding service works: If a person moves her home then she informs the post office in order to update the association of her name (identity address) from an old home address (old location address) to a new home address

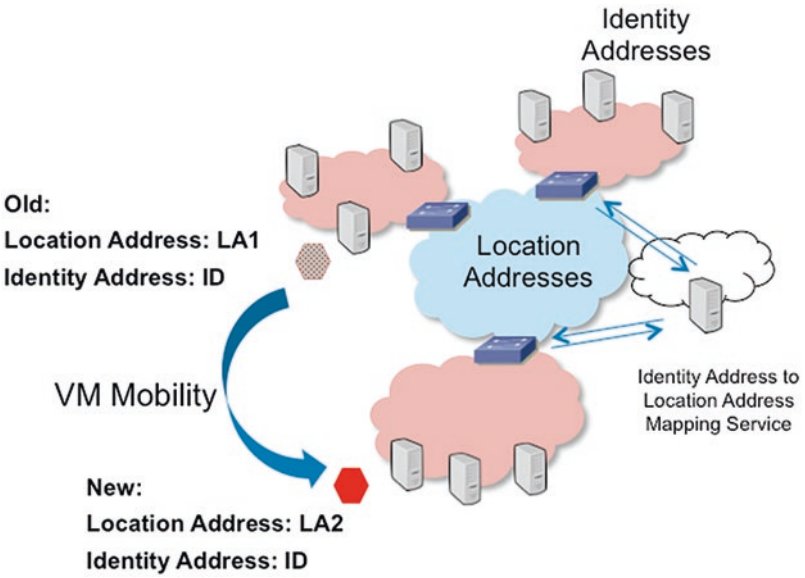



**Fig. 6.3** Identity vs. location addresses with application mobility

(new location address), in order to guarantee uninterrupted delivery of mail (packets) (Fig. 6.3).

The industry has been working on defining networking solutions to support seamless VM mobility, primarily driven by enterprise mobility, data center, and Cloud use cases. The solutions generally differ in how the mapping service (for identity to location address) is implemented: some proposals use a centralized server for the mapping service, whereas others rely on a distributed control protocol. These solutions can be leveraged by Fog computing. We will discuss two of the most prominent solutions: Ethernet Virtual Private Network (EVPN) and Locator/Identifier Separation Protocol (LISP).

#### 6.4.2.1 EVPN

Ethernet Virtual Private Network (EVPN) is an overlay technology that allows Layer 2, and even Layer 3, virtual private networks to be created over a shared Internet Protocol (IP) or Multiprotocol Label Switching (MPLS) transport network. EVPN was standardized by the IETF in RFC 7432. EVPN uses the Border Gateway Protocol (BGP) in order to build the forwarding tables on the participating network elements. Given that EVPN is an overlay technology, only network elements that are at the edge of the network need to support it, and core network elements are oblivious to the fact that EVPN is running in the network. The edge nodes, which run EVPN, are known as EVPN Provider Edge (PE) nodes. PE nodes learn the MAC and IP addresses of connected hosts, from the access side, either by snooping

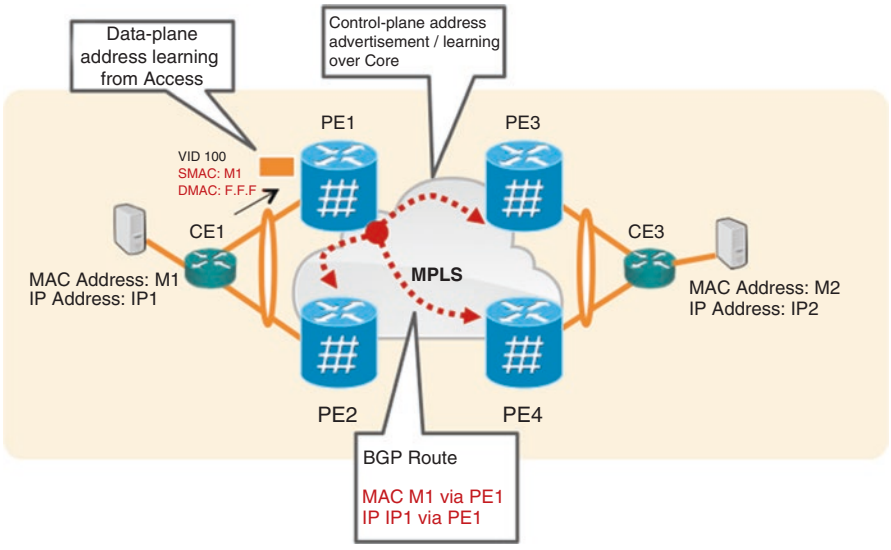



**Fig. 6.4** Ethernet Virtual Private Network (EVPN) architecture

on the host traffic in the data-plane (similar to how Ethernet bridges learn addresses) or by running some control protocol (e.g., the Address Resolution Protocol—ARP). The PE nodes then build a database of the local addresses and advertise these addresses to remote PEs using BGP route messages. Remote PEs, which receive the BGP route messages, build their own forwarding databases where they associate the MAC and IP addresses (identity addresses) of the hosts with the next hop address (location address) of the PE that advertised the route. Host traffic packets received by ingress PE nodes are tunneled (using IP or MPLS encapsulation) over the core network to egress PE nodes, where the tunnel encapsulation is removed, and the original host packets are forwarded to their intended destination(s) (Fig. 6.4).

To handle application mobility, EVPN introduces new BGP messages and dedicated protocol machinery. These mechanisms provide a solution for two issues: first, updating the network infrastructure with the new identity address to location address mappings, and second, guaranteeing optimal forwarding to the default IP gateway after mobility. These two issues and how they are addressed with EVPN will be discussed next.

#### 6.4.2.1.1 Updating the Identity to Location Address Mappings

When an application running in a VM starts sending traffic, the EVPN PE that is servicing the physical server on which the VM is hosted will receive this traffic and learn the application/VM IP and MAC addresses. This PE, call it $PE_{origin}$, will then advertise the VMs addresses in BGP to all the remote PEs in the virtual private network instance. The remote PEs will then update their forwarding tables to indicate

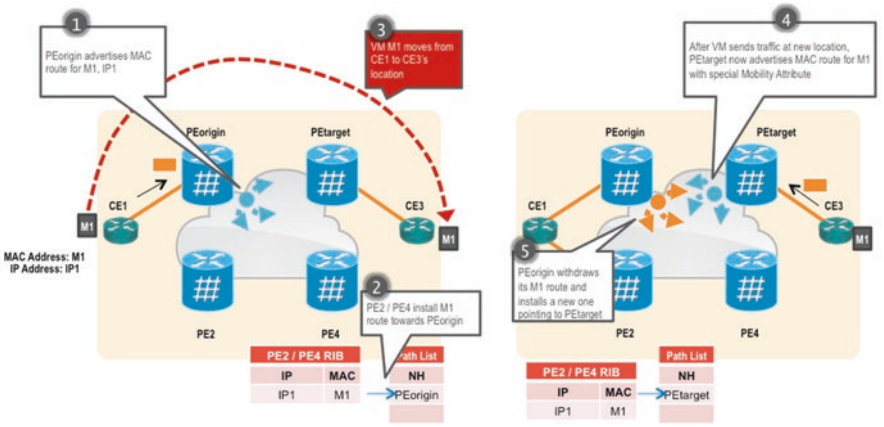



**Fig. 6.5** Mobility in EVPN

that the VM IP and MAC addresses are reachable via $PE_{origin}$. Now, assume that the VM moves to a new physical server, which is serviced by a different PE, call it $PE_{target}$. If the PE nodes continue to send traffic for the VM to $PE_{origin}$, then this traffic will not be delivered to the VM because the latter is no longer on the old server. EVPN solves this issue as follows: when the VM starts sending traffic from its new location, $PE_{target}$ will receive the packets over its access interfaces and will deduce that the VM is locally connected. $PE_{target}$ would also recognize that the VM's IP and MAC addresses were previously learnt from a remote PE, $PE_{origin}$, via a previous BGP route advertisement. Hence, $PE_{target}$ deduces that the VM must have moved, and so it needs to update the rest of the network with the new location of the VM. $PE_{target}$ would then advertise BGP routes for the VM's IP and MAC addresses with a special attribute to indicate the mobility event. This route is sent to all remote PEs, including $PE_{origin}$. When $PE_{origin}$ processes the BGP route message, the special attribute indicates to it that the VM has moved, so $PE_{origin}$ withdraws its previously advertised BGP route for that VM's addresses. This handshake mechanism results in all the PEs converging on using $PE_{target}$ as the new next hop (location address) for the VM traffic (Fig. 6.5).

#### 6.4.2.1.2 Default IP Gateway Problem

As a VM moves from one physical server to another, both its memory (RAM) and disk image are maintained unchanged. This means that the VM's configuration remains unmodified. The configuration includes, among other things, the address of the Default IP Gateway that the VM should use in order to forward network traffic to remote nodes. Typically, the Default IP Gateway should be in close topological proximity to the server that is hosting the VM, in order to guarantee optimal forwarding of network traffic originating from the VM. However, with VM mobility,

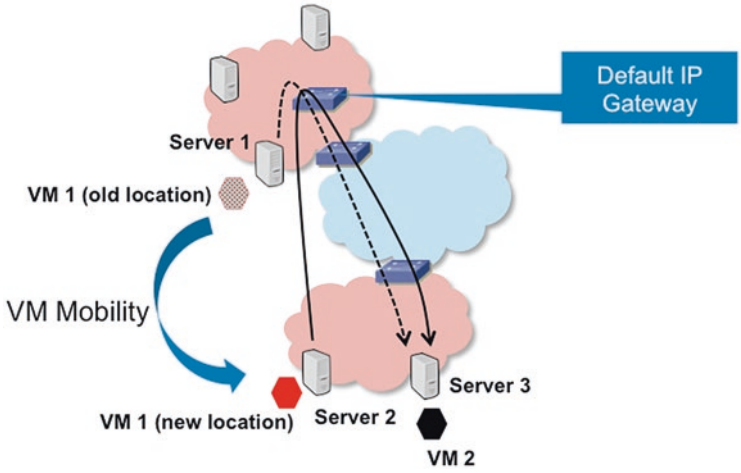



**Fig. 6.6** Default IP Gateway Problem with VM mobility

the VM may land on a new host server that is topological distant from the original Default IP Gateway. In such a case, network traffic sourced by the VM will most likely follow a sub-optimal forwarding path to its destination.

For example, consider the network of Fig. 6.6 above, where VM1 is in communication with VM2 (hosted on Server 3). VM1 is originally hosted on Server 1, and its network traffic that is destined to VM2 initially follows an optimal forwarding path through the Default IP Gateway (the dotted black line). When this VM moves from its initial location to a new location on Server 2, the network traffic will start following a sub-optimal path from Server 2, via the same default gateway, to Server 3 (the solid black line).

To address this problem, EVPN delegates the Default IP Gateway function to the edge of the network (the PE nodes) and enables all the PEs to act as a distributed logical default gateway for hosts that are attached over the PE access interfaces. When a host sends an ARP request for the Default IP Gateway IP address, the EVPN PE intercepts the ARP message and responds to it with its own MAC address. The default gateway IP address is the same across all the participating EVPN PEs. This is specifically to cater for the fact that the VM retains its configured default gateway address after a mobility event (Fig. 6.7).

This approach solves the problem by ensuring that the default gateway is always in topological proximity to the VM after it moves from one physical host to another.

#### 6.4.2.2 LISP

Locator/Identifier Separation Protocol (LISP) is an overlay networking solution that allows complete decoupling of the addressing structure of end-hosts from that of the network infrastructure. LISP formally defines two namespaces for IP addresses:

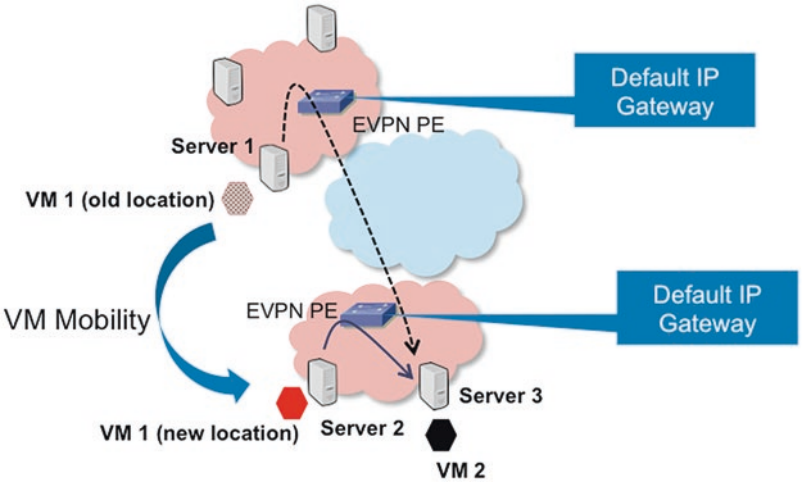



**Fig. 6.7** EVPN Default Gateway solution

Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). EIDs are identity addresses associated with end-hosts, whereas RLOCs are location addresses primarily assigned to routers. LISP dedicates an entire system for the directory service that performs the mapping between EIDs and RLOCs and provides two approaches by which that system can be implemented: a distributed approach that relies on BGP over an Alternative Logical Topology (ALT) and a centralized approach that uses a dedicated database for the mapping known as Dedicated Database Tree (DDT). LISP is standardized in IETF RFC 6830.

Network elements that sit at the edge of a LISP network are known as Ingress Tunnel Routers (ITRs) and Egress Tunnel Routers (ETRs). The ITR receives traffic from end-hosts and is responsible for encapsulating the traffic within a tunnel to be transported over the LISP network. The ETR decapsulates the tunneled traffic and forwards the original end-host packets to their destinations. ITRs and ETRs are identified based on their RLOCs. In order to determine which ETR to forward the traffic to, the ITR consults with a Map Resolver to resolve the RLOC of the ETR associated with the destination EID of the traffic. The Map Resolver is responsible for identifying which Map Server to direct the query to in order to determine the RLOC associated with a given EID. The Map Server is a database that holds all EID/ETR associations. It may be deployed on a pair of devices or a full-blown hierarchy of devices for large-scale implementation (LISP-DDT). Each ETR registers with the Map Server the EID address space that it is authoritative for. When triggered in the data-plane by a packet destined to a remote EID, the ITR issues a "Map Request" toward the Map Resolver. The latter forwards it to the right Map Server, which in turn forwards the request to the authoritative ETR. This ETR replies to the requesting ITR with a "Map Reply" message that contains the list of the RLOCs

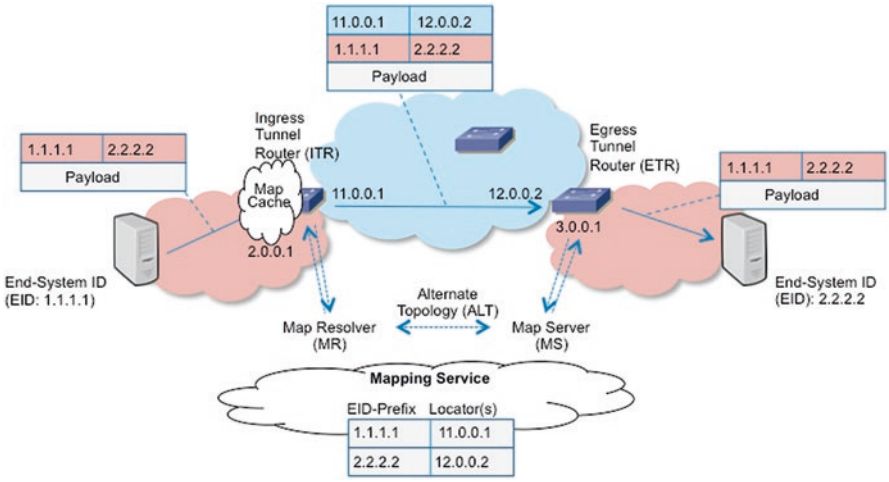



**Fig. 6.8** LISP architecture

having the capability to reach the requested EID, with their characteristics in terms of priority of usage and weighted load partitioning (Fig. 6.8).

To handle application mobility, LISP introduces specific protocol mechanisms. These mechanisms provide a solution for the two issues discussed in the previous section: first, updating the network infrastructure with the new identity address to location address mappings, and second, guaranteeing optimal forwarding to the default IP gateway after mobility.

#### 6.4.2.2.1 Updating the Identity to Location Address Mappings

Mobility is enabled on an ETR by configuring the node with the list of the mobile IP subnets (EIDs) that the ETR is to support. This ETR then becomes the local Default IP Gateway for these mobile EIDs. When an application, with its unique EID, moves into the LISP site, the first packet that it will send to its local Default IP Gateway will trigger the mobility detection on the ETR. The ETR then registers this specific EID with the Map Server. The latter, in turn, deregisters the EID from the previous authoritative ETR. What remains is to update the map caches of all the ITRs that have communicated with the application prior to its move, as those ITRs will have stale entries to the RLOC of the old authoritative ETR. This function is performed by the old authoritative ETR itself, which upon receiving any data traffic for the EID that has moved sends back a "Solicit-Map-Request" message to the originating ITR. This message instructs the ITR to refresh its cache (Fig. 6.9).

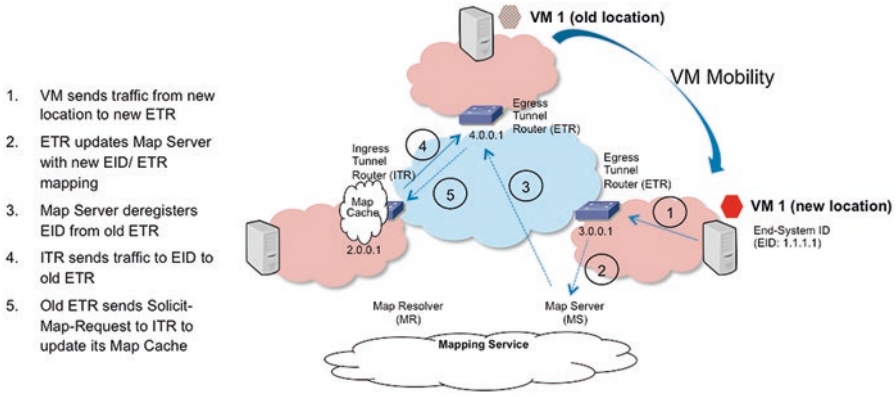



**Fig. 6.9** LISP mobility

#### 6.4.2.2.2 Default IP Gateway Problem

LISP solves the Default IP Gateway Problem by ensuring that every site has a default gateway configured for the same prefix. This gateway must use the same (virtual) IP and MAC Addresses in order to guarantee that the traffic originating from the moved VM follows an optimal path out of the local LISP Tunneling Router rather than being forwarded to another site. First Hop Redundancy Protocols (e.g., VRRP) must be configured with identical gateway and MAC addresses in all sites, and their packets must not be allowed to leak beyond a given site. This way, when a VM moves it will always find the same default gateway regardless of its location.

### *6.4.3 Fog Orchestration*

Orchestration, in the context of Fog computing, refers to the process of automating the various workflows that perform the full lifecycle management of the Fog infrastructure. This includes the provisioning and management of its three components (compute, network, storage) and associated resources. For illustration, tasks such as deploying, debugging, patching, and updating applications or operating systems, setting up network connectivity between application entities and reserving bandwidth, and allocating and expanding disk space are all examples of workflows that fall under orchestration.

Orchestration is a complex task in Fog environment as it involves components spread across heterogeneous systems and distributed across multiple locations. Due to the Fog's multitiered hierarchical organization, it requires a hierarchically organized orchestration plane that supports dynamic policies and interplay with Cloud orchestration (Fig. 6.10).

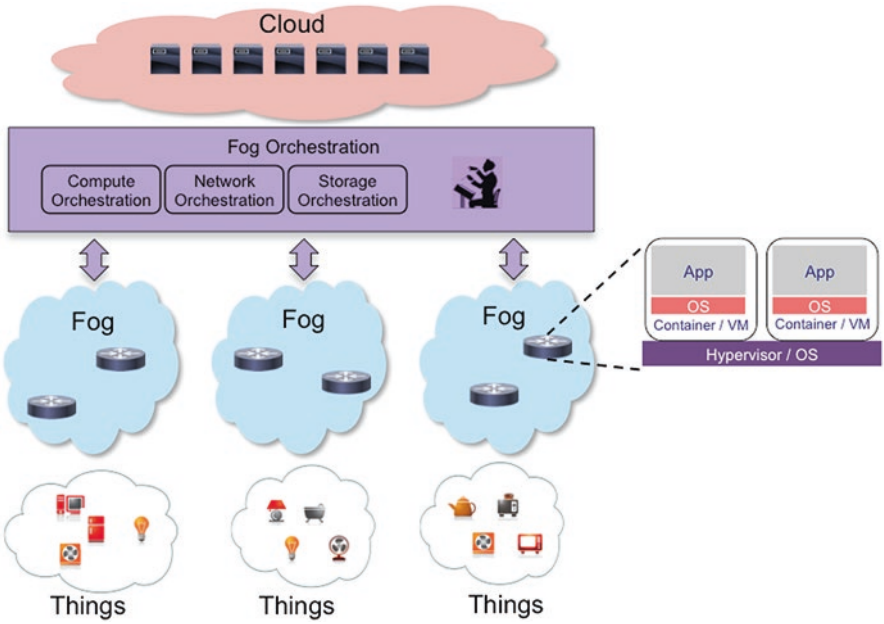



**Fig. 6.10** Fog orchestration

Fog orchestration differs from Cloud orchestration in three different facets: topology, Things connectivity, and network performance guarantees.

#### 6.4.3.1 Topology

Cloud orchestration systems that are available today make assumptions about the network: the physical layout of the topology (three-tiered, four-tiered, fat tree, etc.), the abundance of available bandwidth and the fact that the network elements are capable devices and therefore have no restrictions on the size of the routing tables. While these assumptions are valid in the Cloud, they do not hold true in the Fog. Fog topologies are ad hoc best-fit affairs. They have heterogeneous interconnects as well as dynamically varying bandwidth, latency, and reliability characteristics. Fog orchestration software has to deal with an isomorphic topologies that are directly connected to Things.

#### 6.4.3.2 Things Connectivity

With Fog, the orchestration software needs to be able to deploy applications, which need direct access to Things (e.g., legacy applications), on Fog nodes that are physically connected to these specific Things. To enable the communication between the

applications and their Things, specialized device drivers need to be initialized on the Fog nodes by the orchestration system. Furthermore, applications may require data from remote Things, in which case the orchestration software needs to dynamically establish network overlays to facilitate network communication between the applications and those remote Things.

#### 6.4.3.3 Network Performance Guarantees

Orchestration systems for the Cloud are capable of deploying applications on nodes that can offer the right performance guarantees in terms of processing power, memory, and disk space. For Fog, these performance guarantees alone are not enough. Another dimension of complexity arises due to control applications that require network performance guarantees, in terms of upper bounds on latency and jitter, in their communication with Things. In order to support these control applications in the Fog, the orchestration system needs to be able to incorporate network latency and jitter into the application placement and scheduling algorithms. Mobility complicates this further, as the placement decisions need to be recalculated with changing conditions.

### *6.4.4 Data Management*

#### 6.4.4.1 Data in Motion

There are vast amounts of data crossing the network every day. However, those bits and bytes provide a wealth of information about actions, time, location, and devices. By gathering and combining pieces of information together, it is possible to start seeing patterns and gain greater insights. In other words, it is possible to gain knowledge. And it is through knowledge that we, as humans, can learn and apply wisdom, leading to better outcomes.

New data sources are being created and added to the network every day. From a video camera in a transit bus, a tire pressure sensor in a truck, a jet engine, to a smart meter attached to a house. These devices are creating a constant stream of data. Very soon, the data generated by the IoT will make up the majority of all information available on the Internet and will change the face of big data. It will not be possible to store all this data and analyze it later. The real-time nature of these new sources of data requires that their output be evaluated in motion and in meaningful way. The value of data is often dictated by time—being at its highest value when it is first created. Actionable insights can be extracted and acted upon, as data is generated, to create advantage here and now or even predict the future. Mastery of data—moving from data to wisdom—has the potential to improve various aspects of our personal and business life. Organizations can make better decisions, provide enhanced experiences, and achieve competitive advantage (Fig. 6.11).

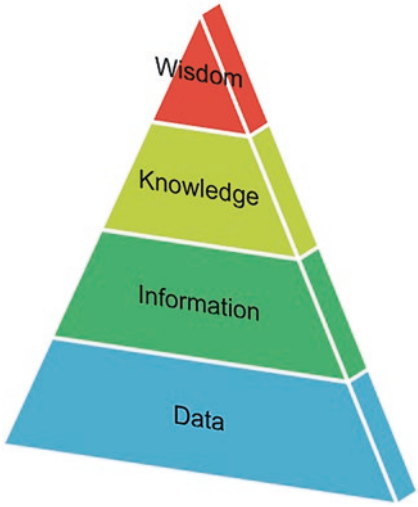



**Fig. 6.11** DIKW pyramid

Most of the new data that will be generated in the IoT is real-time data that fits into a broad category called Data in Motion. This refers to the constant stream of sensor-generated data that defies traditional processes for capture, storage, and analysis.

Historically, in order to find actionable insights, enterprises have focused their analytics or business intelligence applications on data captured and stored using traditional relational data warehouses or "enterprise historian" technologies.

However, the limits of this approach have been tested by the increase in volume of this so-called Data at Rest. The challenges inherent in collecting, searching, sharing, analyzing, and visualizing insights from these ever-expanding data sets have led to the development of massively parallel computing software running on tens, hundreds, or even thousands of servers. As innovative and adaptive as these big data technologies are, they still rely on historical data to find the proverbial needle in the haystack.

As the IoT gathers momentum, the vast number of connections will trigger a flood of data, at an even more accelerated pace. While this new Data in Motion has huge potential, it also has a very limited shelf life. As such, its primary value lies in it being analyzed soon after it is created—in many cases, immediately after it is created. Hence, the traditional data management paradigm where raw data is stored first and analyzed later does not fit the temporal nature of IoT data. A new paradigm for handling Data in Motion is required, where data is analyzed as soon as it is generated and then optionally stored if required. The analysis can involve one or more of the following: aggregation, reduction/filtering, categorization/classification, contextualization, dimensioning, compression, pattern matching, normalization, and anonymization. All of these functions can be applied in micro-services that are hosted in the Fog (Fig. 6.12).

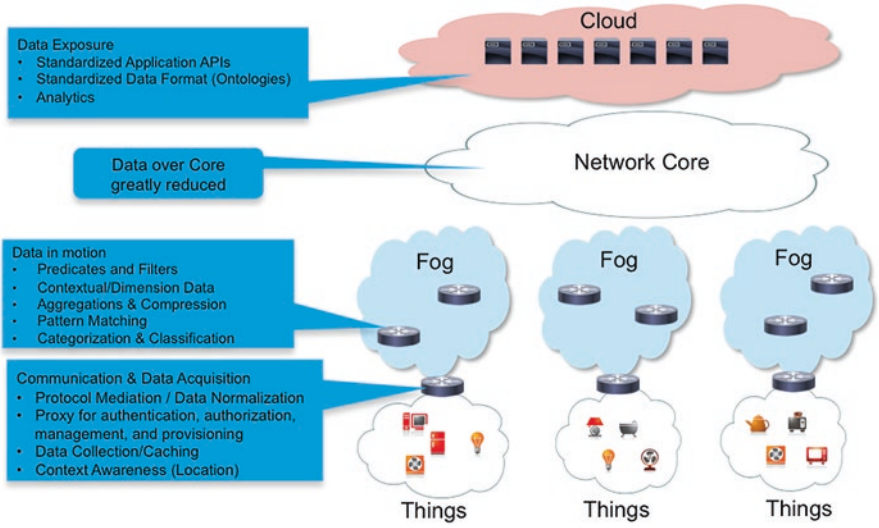



**Fig. 6.12** Data management in the Fog

### 6.4.4.2 Search Technologies and Engines

With the availability of massive amounts of data, the need arises for reliable and effective mechanisms of searching for information that is useful and relevant. Search technologies have made great strides since the inception of the World Wide Web. However, these technologies, and the engines that utilize them, target static or slowly changing web data and are generally lacking when dealing with the constantly streaming data in IoT.

IoT requires a solution for distributed data search, where queries can be propagated throughout the Fog domains. The solution can be logically organized into two planes: Things Plane and Search Plane. The Things Plane encompasses the physical Things, Network, and Compute nodes in the Fog. The Search Plane is a logical view of the various Fog nodes that support the distributed search functionality together with the network overlay that enables communication between them. Such overlay could be implemented, for instance, using a Federation Message Bus. Search queries are injected into the Search Plane at some Fog node and propagate throughout the Search Plane. Special considerations are required to ensure that such propagation does not lead to traffic storms that overwhelm the network or the Fog nodes. Furthermore, mechanisms are required to limit the search scope, or radius, order to guarantee scalability and relevance of returned results. One approach would be to rely on Wave algorithms, such as the Echo algorithm, for query distribution and perform tree-based aggregation of partial results. These algorithms typically result in very low latency, have a low overhead, and generally scale to hundreds of thousands of nodes (Fig. 6.13).

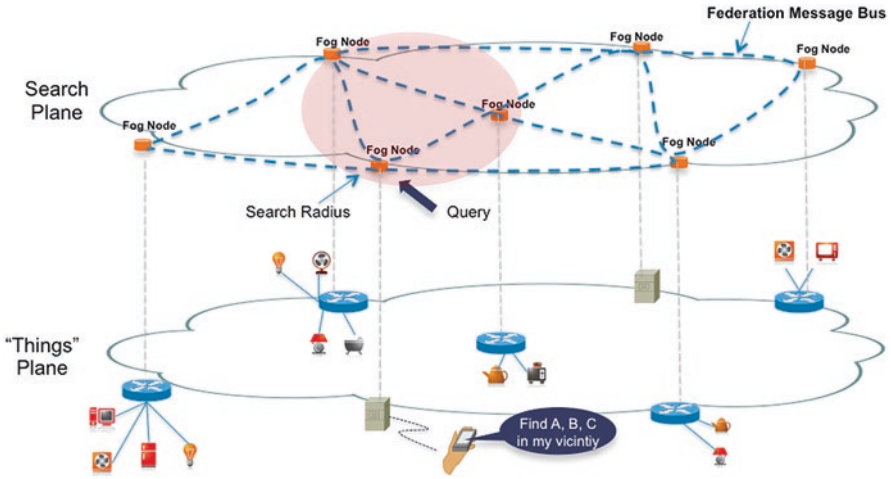



**Fig. 6.13** Data search in Fog

As discussed in Chap. 5, both the ETSI and oneM2M standards define basic mechanisms for data search based on metadata. However, these mechanisms only allow elementary search procedures based on string matching between requests and the resource metadata. This provides a syntactic search capability with binary (yes/no) outcomes based on exact matches. Exact matches are highly unlikely in real-world IoT deployments with heterogeneous devices and Things from different vendors and providers. As such, effective search mechanisms should allow for "fuzzy matches," with partial correspondence between the request and the available data. Such mechanisms, ideally, would provide a measure of the semantic similarity between the original request and the retrieved results. To achieve this, Semantic Web technologies could be applied to the IoT: the IoT data can be enriched with semantic-based annotations that reference shared domain conceptualizations, and the search mechanisms can utilize semantic matching techniques to perform the ranking of potential results. Ruta et al. propose such a framework that utilizes and enhanced version of CoAP as the underlying protocol for the Federation Message Bus.

### *6.4.5 More Gaps Ahead*

Clouds are deployed in data centers, where network topologies are well defined and the infrastructure is physically secured with solid walls and cages. Network input and output between applications deployed in the data center and the outside world (e.g., Internet) are mediated through security appliances, such as firewalls, which provide applications with a well-incubated environment under which they can

operate. Furthermore, network bandwidth is abundant, and it is relatively easy to change the network physical topology. With Fog, applications may be logically grouped together but not necessarily part of the same physical set up. The first gap to address is providing an orchestration system that enables the connection of applications deployed on Fog nodes to other applications, which are part of the same group, but are on desperate Fog nodes, as well as to applications that are in the Cloud. These connections could be over bandwidth-constrained links that cannot be changed due to the physical realities of the deployment. In light of this, open questions remain as to whether the Fog nodes need to replicate the entire functionality of the data center, including server, switch, and gateway functions (data-center-in-a-box) or whether these functions should be distributed across multiple nodes and assembled together logically through the notion of "service chains." Another open gap is security: Fog nodes may be mounted in the field or on top of a light pole, so anyone could potentially gain physical access to them, attach wires, and compromise the security of the application or the network connectivity. New mechanisms of anomaly and tampering detection are needed. Yet another gap is in how would Fog nodes talk to Things: should that be through direct electric connectivity (e.g., PCI bus) or via the networking stack. Furthermore, in order for applications to leverage Fog, a high-level programming model is required which simplifies the development of large-scale distributed software. Such model provides simplified programming abstractions and supports dynamic application scaling at runtime.

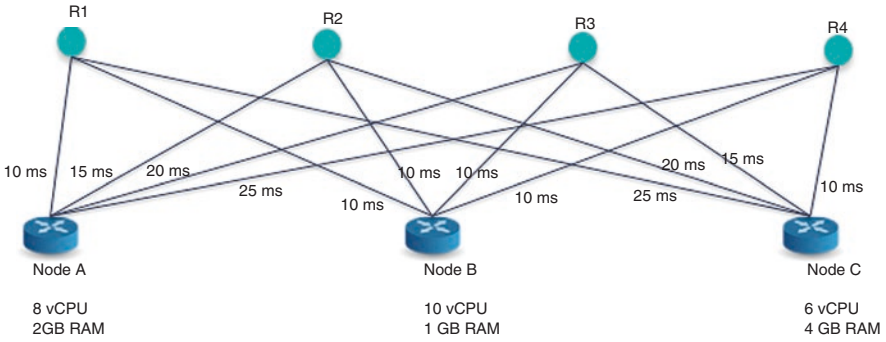
## 6.5 Summary

In this chapter we introduced the concept of Fog computing and discussed its relationship to Cloud computing. The various IoT requirements driving the need for Fog were covered. We also discussed the prerequisites and enabling technologies for Fog, in terms of virtualization technologies, network mobility technologies, orchestration, and data management technologies.

## Problems and Exercises

1. Will Fog computing replace Cloud computing? Why or why not?
2. What is the definition of Fog computing?
3. What are the characteristics that uniquely distinguish Fog from Cloud computing?
4. What makes containers lighter-weight virtualization constructs compared to virtual machines? Why is this attribute of containers important for Fog?
5. What are the two problems that all network mobility solutions aim to address?
6. Why can't traditional data management and analytics techniques be applied to IoT?

7. What three functions should a Fog orchestration solution address and solve?
8. What is "Data in Motion"?
9. Why are semantic search mechanisms important for IoT?
10. Consider the following Fog domain shown in the figure below. For each Fog node, the diagram shows the number of virtual CPUs (vCPU) and RAM available. Also, the communication latency from each node to a remote sensor (labeled $R_1$ through $R_4$) is captured.




There are five applications that need to be placed on the Fog nodes, and each application has specific demands for CPU, RAM, and communication as depicted in the table below:

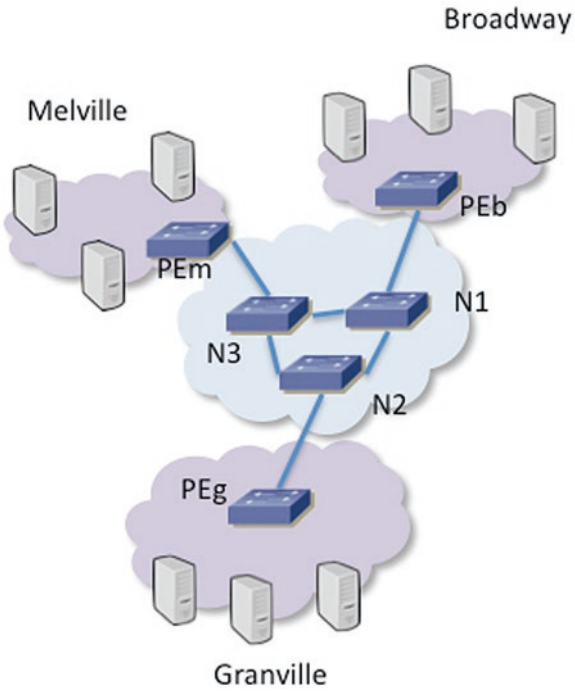
| Application | CPU demand (vCPU) | RAM demand (GB) | Communications demand |
|---|---|---|---|
| 1 | 5 | 0.5 | R1 (< 12 ms) |
| 2 | 2 | 1 | R2 |
| 3 | 1 | 0.25 | R3 |
| 4 | 1 | 1 | R4 |

Find the optimal placement of the five applications on the three Fog nodes such as to minimize the communication latency between each application and the sensor that it needs to connect to.

11. A Fog domain is using EVPN to support workload mobility. The topology of the domain is as shown in the figure below. Every BGP speaker requires approximately 10 milliseconds to process a BGP message, including any transmission/reception delay. A VM moves from the Melville server farm to the Granville server farm.

    (a) If N1, N2, and N3 form a BGP route-reflector (RR) cluster (i.e., fully meshed BGP sessions) and each of PEb, PEg, and PEm have a BGP session with their directly attached RR, how long would it be before all other applications are capable of communicating with the VM in its new location assuming it takes 20 millisecond for GARP messages to be received and processed by the PE connected to the new server?

(b) If N1, N2, and N3 are MPLS core routers, rather than route-reflectors, how does the above convergence time change?




12. An IT administrator is trying to decide on whether to use Linux container or virtual machine for an interactive location-based interactive marketing application. Each instance of the application requires 200 MB of RAM to run, including all dependencies/libraries. The Linux distribution she is considering has a runtime memory footprint of 800 MB. A given application instance needs to move frequently in the Fog domain, to maintain close proximity to a target customer and deliver an immersive HD video/audio experience. Assume that the wireless links interconnecting the Fog nodes operate at 100 Mbps.

    (a) In the best-case scenario, how long would it take for the memory image of the application to move from one Fog node to another in the case where the application runs in a virtual machine?
    (b) Repeat (a) for the case where the application runs in a Linux container.
    (c) Which virtualization construct should the IT administrator pick for her application and why?

13. A smart parking application is implemented in the future city of Metrotown using Fog computing. Fred is looking for parking in Metrotown's downtown

shopping district. His car is capable of communicating automatically with the city infrastructure to locate available parking. The Fog domain in Metrotown is such that Fog nodes are placed roughly 50 meters apart, on street lighting poles. The car's embedded application is searching for parking availability within a 1 km radius from the current vehicle's location. Assume that the Fog domain is using the Echo algorithm to search for data. If node processing latency and link propagation latency are 2 milliseconds and 1 millisecond, respectively, how long would it be before the search request has reached all nodes in the Fog domain?

14. A Fog orchestration system is responsible for the mobility of workloads among three Fog nodes dispersed in three locations: Coal Harbor, Yaletown, and West End. The choice of a server for a given workload is a function of the CPU load of that server and the network communication latency from the server to the client. The orchestrator assigns a score between 0 and 1 to each server based on its CPU load, with a score of 1 for servers having less than 25% utilization, a score of 0.5 for servers with utilization between 25% and 75%, and a score of 0.25 for utilization above 75%. The orchestrator ranks the servers based on network latency and assigns them a score between 0 and 1 linearly depending on their rank in the ordered list, with a score of 0 assigned to the server with the highest latency and a score of 1 assigned to the server with the least latency. Assume that a user on her smartphone is roaming between the three locations. The network latency from her phone to the Coal Harbor Fog node is 200 microseconds, to the West End Fog node is 300 microseconds, and to the Yaletown Fog node is 250 microseconds. The average CPU utilization for the servers is 80% for Coal Harbor, 13% for Yaletown, and 50% for West End Fog nodes.
    (a) If the Fog orchestrator is configured to give equal weight to communication latency as server CPU load, which server would the orchestrator select?
    (b) If the communication latency carries twice the weight of the server CPU load, what would be the server that the orchestrator selects?
15. Explain the difference between the three different levels of virtualization: CPU instruction set level, hardware abstraction layer (HAL) level, operating system level.
16. What distinguishes LISP from other networking solutions that support mobility?
17. Describe Nielsen's Law. How does it relate to Moore's Law? What are the implications for IoT?
18. How is network connectivity different in the Fog from the Cloud?
19. How does rapid mobility impact communicating IoT applications?
20. When you conduct a search on your favorite web search engine, is the search conducted over the Internet in real time? Will this model work for IoT?

## References


1. Nielsen's Law of Internet Bandwidth, J. Nielsen, April 5, 1998, Online: http://www.nngroup.com/articles/law-of-bandwidth/
2. M. Yannuzzi, et al., Key ingredients in an IoT recipe: Fog computing, cloud computing and more fog computing, IEEE 19th international workshop on computer aided modeling and design of communication links and networks (CAMAD), December 2014
3. F. Bonomi, et al., Fog computing and its role in the Internet Of Things, SIGCOMM 2012
4. P. Mell, T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology Special Publication 800–145, (2011)
5. S. Nanda, et al., A Survey on Vritualization Technologies. http://www.ecsl.cs.sunysb.edu/tr/TR179.pdf
6. Docker, www.docker.com
7. T. Kondo, et al., A mobility management system for the global live migration of virtual machine across multiple sites, Computer software and applications conference workshops (COMPSACW), 2014 IEEE 38th international, July 2014
8. A. Sajassi, et al., BGP MPLS-Based Ethernet VPN, IETF RFC 7432, February 2015
9. Y. Rekhter, et al., Network-related VM Mobility Issues, draft-ietf-nvo3-vm-mobility-issues, work in progress, June 2014
10. D. Farinacci, et al., The Locator/ID Separation Protocol (LISP), IETF RFC 6830, January 2013
11. Y. Hertoghs, M. Binderberg, End Host Mobility Use Cases for LISP, draft-hertoghs-lisp-mobility-use-cases, work in progress, February 2014
12. C. Morales, A Vision for Fog Software and Application Architecture, Fog Computing Expo, November 2014
13. M. Uddin, et al. Graph search for cloud network management. Network operations and management symposium (NOMS), 2014 IEEE, May 2014
14. E.J.H. Chang, Echo Algorithms: Depth Parallel Operations on General Graphs. IEEE Trans. Softw. Eng **SE-8**(4) (1982)
15. M. Ruta, et al. Resource annotation, dissemination and discovery in the semantic web of things: a CoAP based framework. IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013