

A blue parallelogram and a light green parallelogram are positioned on the left side of the slide, overlapping each other and the dark background. The blue shape is on the left, and the green shape is to its right, partially overlapping it.

# Exploring the Edge

February 19, 2020



# Recap of Jim's Talk

- Looked at IOT enablers
  - Economics, System Software capabilities, development platforms, consortia
- Looked at business models and IOT system models
- Defined course architectural models
- Deep dived into sensor and actuator function
- Looked at ARM based architectures
- Broke up into teams to address sensor and system design considerations for 4 problems

Today we're going to talk more about the infrastructure to connect devices and create the systems required to bring value to the IOT space.



# Agenda

- Cloud Computing reminder
- XaaS Explanation
- What is Edge computing
- Take a look at the use cases
- Discuss vRAN (Radio access network) use case
  - Dive into the Radio Network topology and issues
  - Discuss the problems required to solve
  - Show why this is a System SW edge problem
- Break out and assign 4 other use cases to teams
- Read back from teams
- Edge Computing Architectures/Differences
  - Openstack
  - Kubernetes
  - AWS Greengrass
- Summary



# Cloud Computing

It's a model for enabling convenient, on demand network access to a shared pool of configurable computing resources.

- On demand: resources are dynamically created.
- Multi-tenant: resources are shared between users.
- Elasticity: infrastructure is flexible (grow/reduce)
- Measured service: Users pay what they use.

### My Software Package



### IaaS Infrastructure as a Service



### PaaS Platform as a Service



### SaaS Software as a Service



#### Legends :

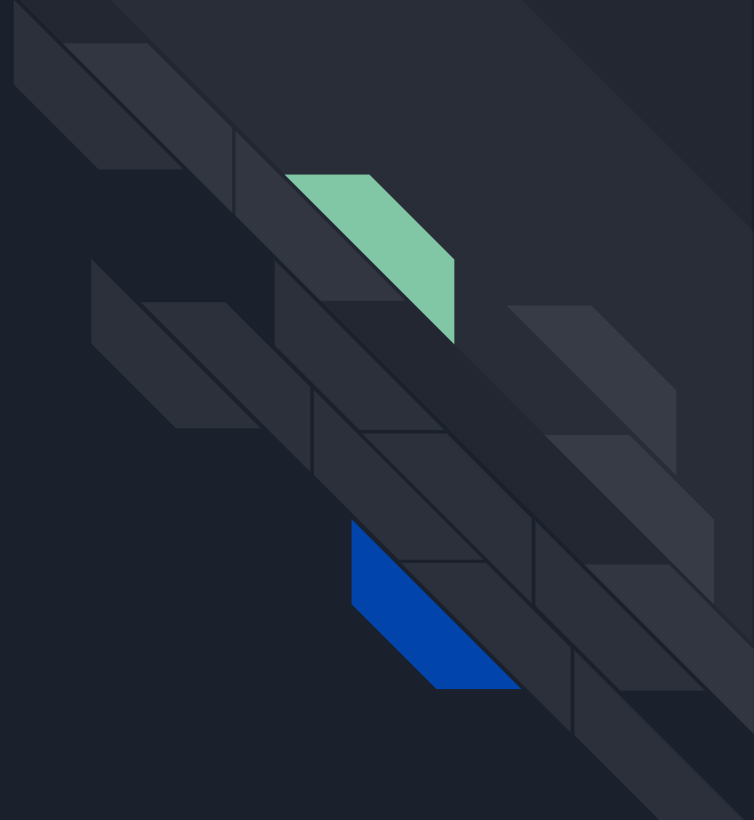
Managed by Me

Managed by the Vendor

## What is Edge Computing?

Edge computing is computing that takes place at or near the physical location of either the user or the source of the data. By placing computing services closer to these locations, users benefit from faster, more reliable services while companies benefit from the flexibility of hybrid cloud computing. Edge computing is one way that a company can use and distribute a common pool of resources across a large number of locations.

Fog Computing is analogous to Edge computing
























# Benefits of Edge Computing

Traditionally, cloud computing has focused on centralizing services into a handful of large data centers. Centralizing allowed resources to be highly scalable and shared more efficiently while maintaining control and enterprise security.

Edge computing addresses those use cases that can not be adequately addressed by the centralization approach of cloud computing, often because of networking requirements or other constraints. It focuses on several small computing sites that reduce network cost, avoid bandwidth constraints, reduce transmission delays, limit service failures, and better control the movement of sensitive data. Load times are cut by hundreds of milliseconds and online services deployed closer to users enable both dynamic and static caching capabilities.

For end-users this means a faster, more consistent user experience. For enterprises and service providers this means low-latency, highly available apps with real-time monitoring.

# Some Use Case Categories

<b>Remote Office Branch Office</b> Retail, Enterprise, ... Walmart  	<b>Radio Access Network (vRAN)</b> Telco verizon  T-Mobile	<b>Mobile Edge Computing (MEC)</b> Telco + ISP + App Disney+  NETFLIX	<b>(Universal) Customer Premise Equipment</b> Service Provider swisscom  AT&T	<b>Connected Sensors and Controllers (IoT)</b> Industrial, Energy  BOSCH SAMSUNG 
<b>AI/ML Surveillance and Monitoring</b> Smart City / IoT  ADT 	<b>Connected Vehicles (Car, Train, Plane)</b> Transportation  TOYOTA 	<b>FinTech (Fraud + Trading)</b> Financial Morgan Stanley  	<b>Healthcare Monitoring and Data Processing</b> Healthcare   	<b>In-field Operations</b> Public Sector   



# Edge Computing

## ADVANTAGES

## CHALLENGES

LATENCY  
BANDWIDTH  
RESILIENCE  
SECURITY  
SOVEREIGNTY  
CONNECTIVITY  
COST

TECHNOLOGY

SCALE

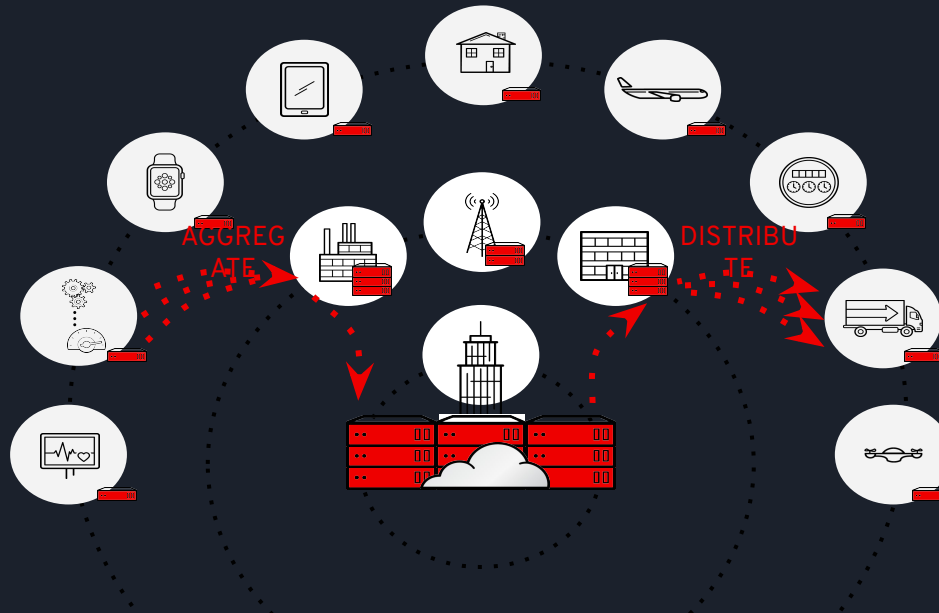
PEOPLE &  
EXPERTISE

PROCESSES & DATA

ENVIRONMENTAL

SECURITY

COST



# Edge Tiers

Centralize where you can, distribute where you must.

Provider/Enterprise Core



Core  
Data Center



Regional  
Data Center



Provider  
Aggregation  
Edge



Provider  
Access  
Edge



Provider  
Device  
Edge



Infrastructure  
Edge



Device  
Edge



Device or  
Sensor

SCALE



FOOTPRINT



"last  
mile"

\* Edge computing == Fog computing  
(there is no real difference other than marketing)

# vRAN Exploration

(Virtual Radio Access Network Exploration)



# Wireless Tech Alphabet Soup

ACRONYM	MEANING
<b>1xEV-DO</b>	Evolution-Data Optimized, a 3G CDMA technology for boosting data speeds and network capacity
<b>2G, 3G, 4G</b>	The second-generation, third-generation and fourth-generation of digital wireless technologies
<b>3GPP</b>	3rd Generation Partnership Project, a GSM wireless technology standards consortium
<b>CDMA</b>	Code Division Multiple Access, a foundational digital wireless technology
<b>cdmaOne</b>	The 2G version of CDMA, also called IS-95
<b>CDMA2000</b>	The 3G version of CDMA, which includes 1xEV-DO
<b>EDGE</b>	Enhanced Data rates for GSM Evolution, a 3G GSM technology
<b>GSM</b>	Global System for Mobile Communications, the world's most widely deployed 2G wireless technology
<b>LTE</b>	Long Term Evolution, a 4G technology platform
<b>UMTS</b>	Universal Mobile Telecommunications System, a 3G GSM technology
<b>WiMAX</b>	Worldwide Interoperability for Microwave Access; the Mobile version is a 4G technology and is also called 802.16e

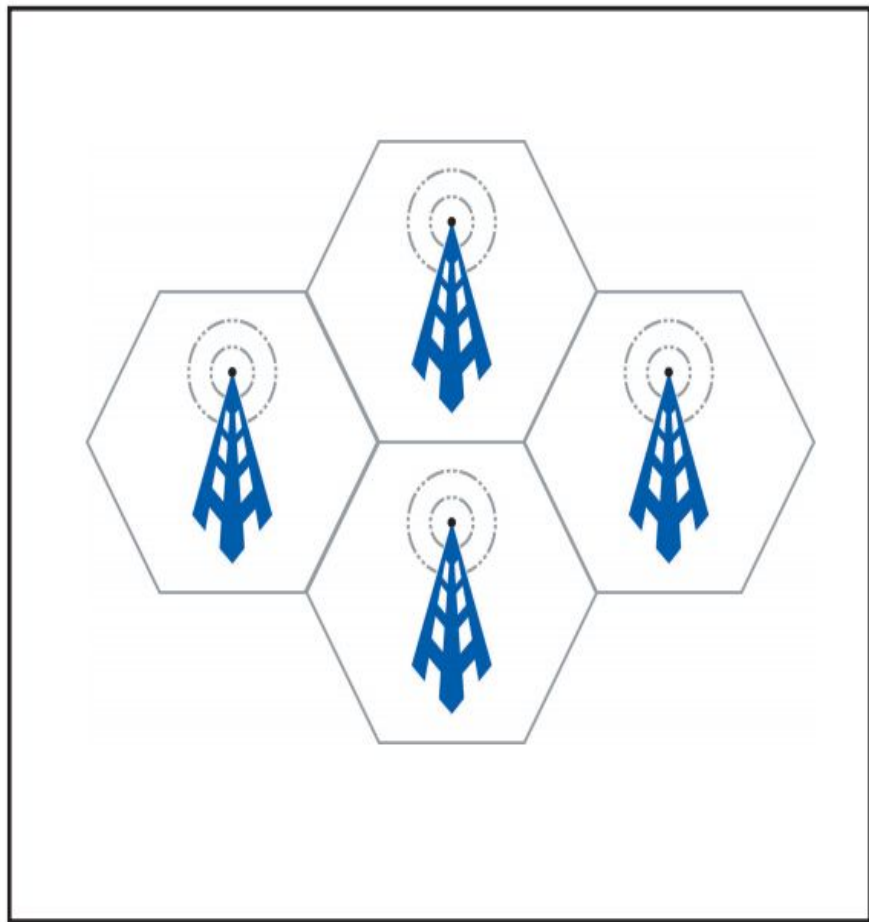


FIGURE 1: Towers and Cells in a Hex Pattern

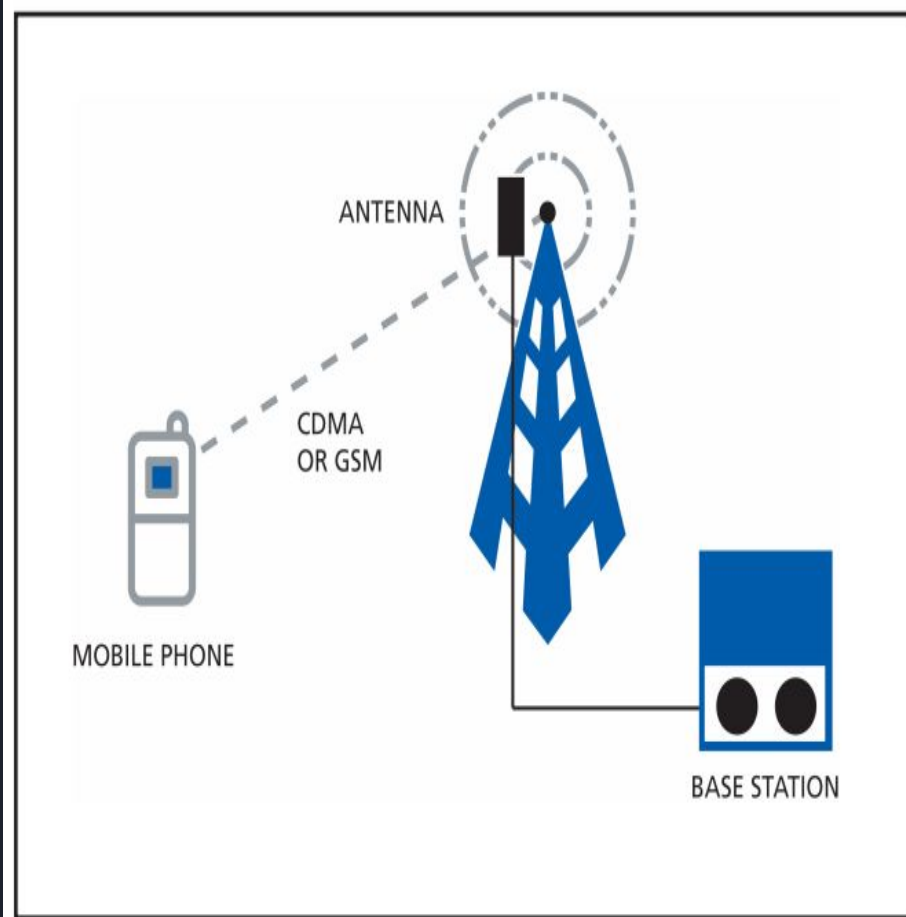


FIGURE 2: Wireless Radio Access Network

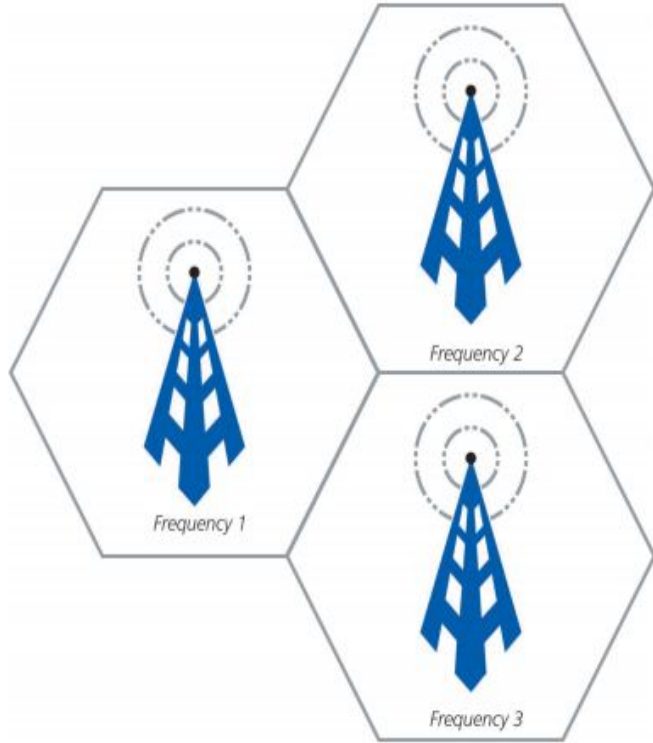


FIGURE 3: Large Cells with Omnidirectional Antennas

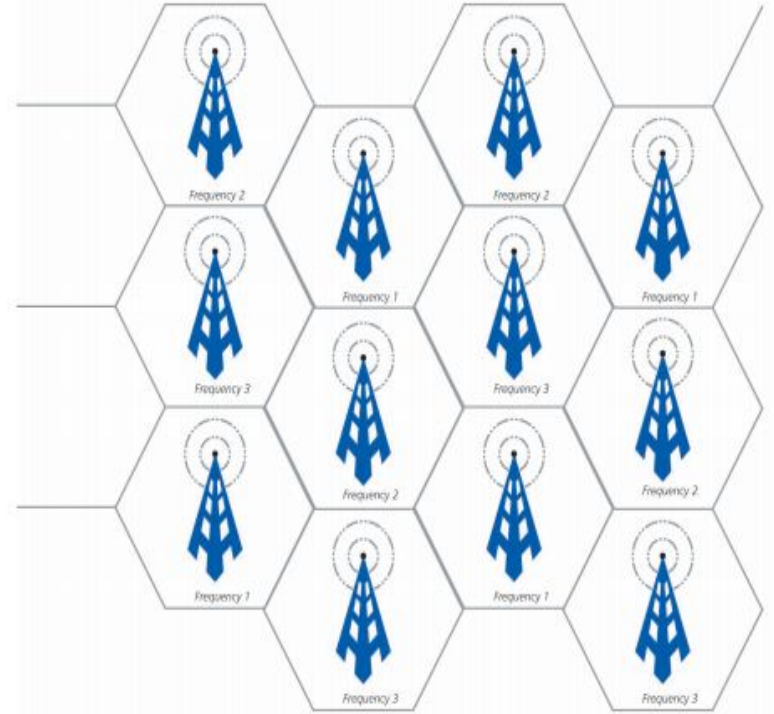


FIGURE 4: Reduced Cell Size with Frequency Reuse



TABLE 1: Cell Tower Types

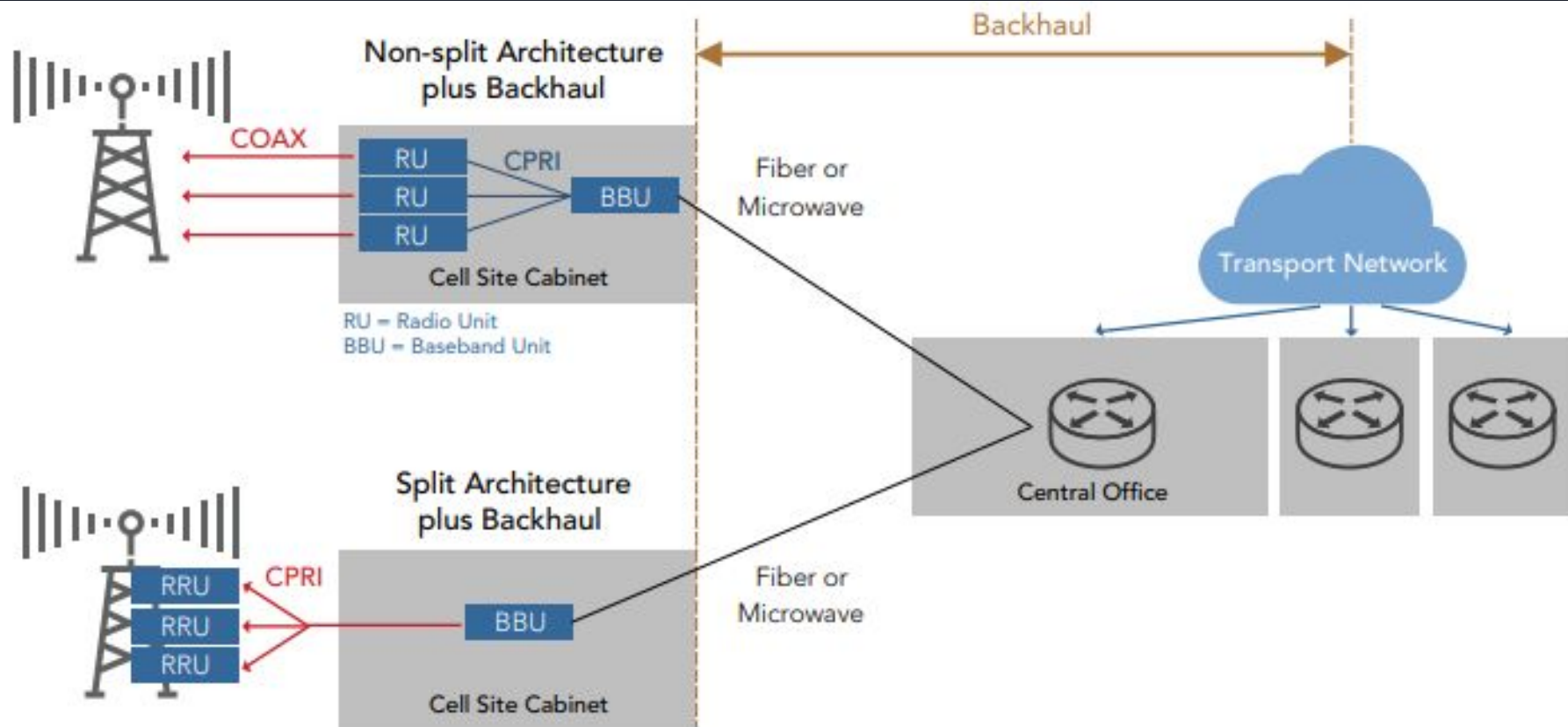
TOWER TYPE	MEANING	DESCRIPTION
<b>Macrocells</b>	10 miles	Standalone or structure attached
<b>Microcells</b>	1 mile in diameter	Urban and suburban
<b>Picocells</b>	250 yards	Office buildings, airports, campuses
<b>Femtocells</b>	Limited in building	Personal devices for home/office



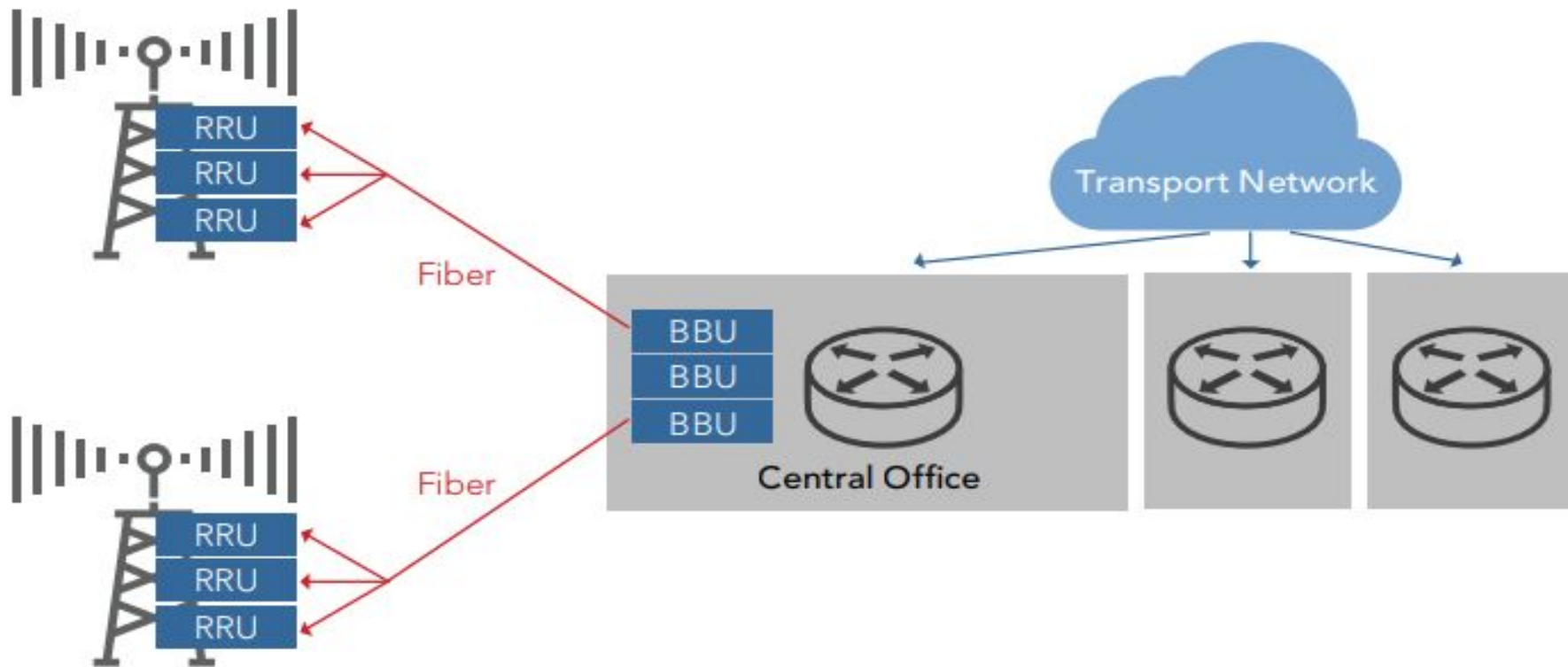
# Radio Topology Considerations

- Cell represents the geographical area of coverage
- 4G has an effective range of up to approximately 20 miles
- Towers have to be placed for desired capacity and coverage
- Radio is running in 800mhz frequency range (New frequencies being sold by FCC)
- US has 100's of thousands of cell towers
- 5G will have lesser range than 4G due to higher frequency
- 5G will at least double the cell tower needs
- 5G doesn't require the same constraints as 4G towers
- 5G radios are smaller format and can be attached on buildings and infrastructure





*Figure 1. Traditional, distributed RAN*



*Figure 2. Centralized RAN*

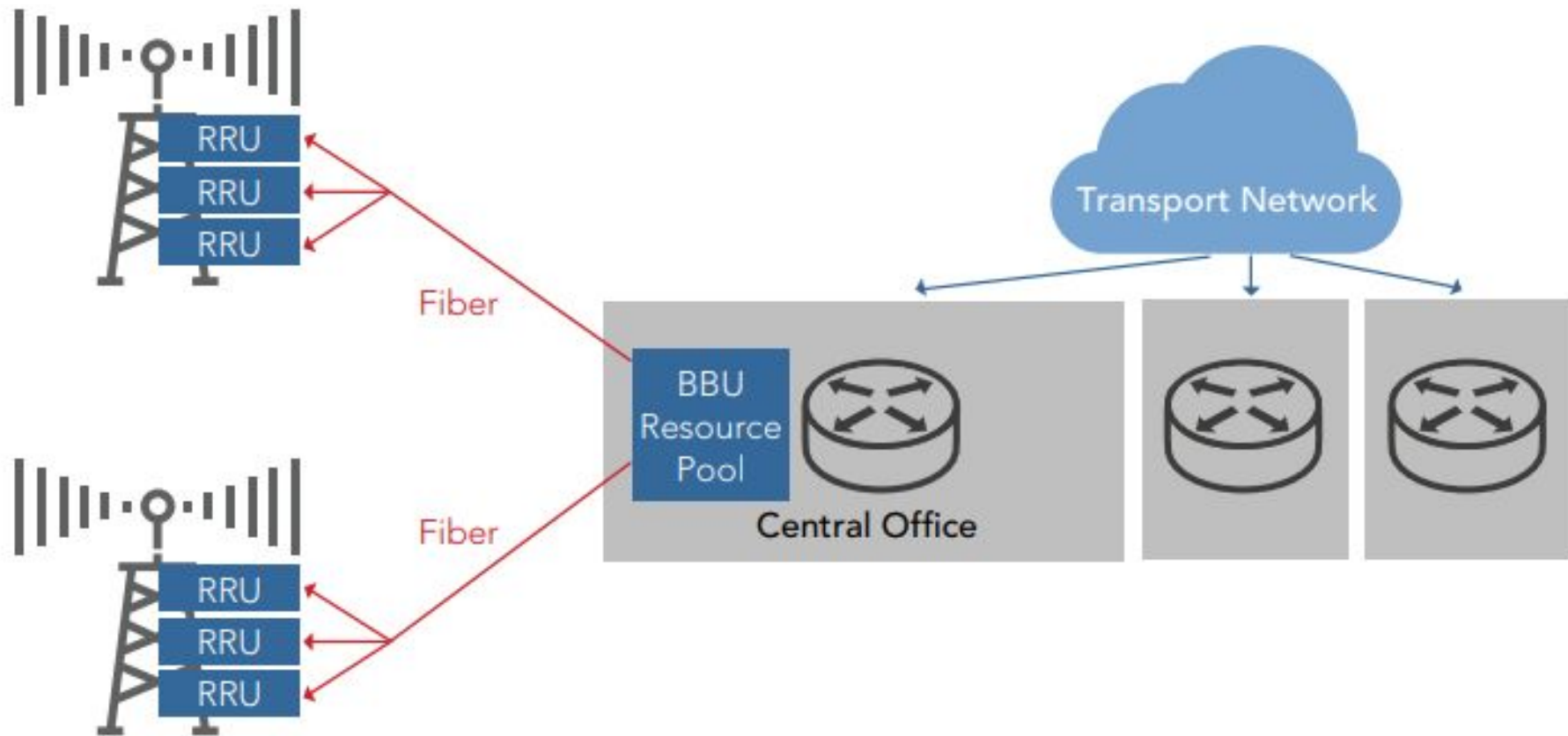


Figure 3. C-RAN baseband unit pooling

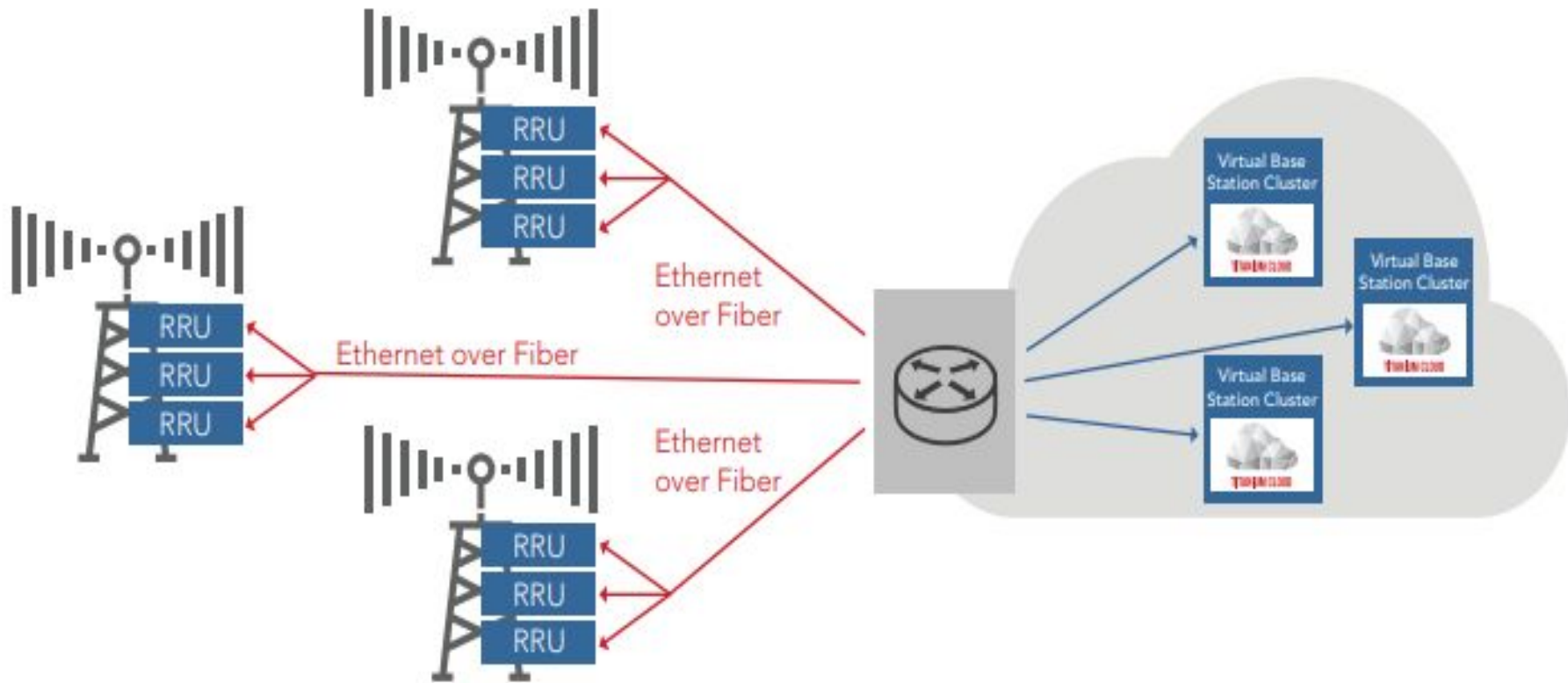


Figure 4. Virtual RAN

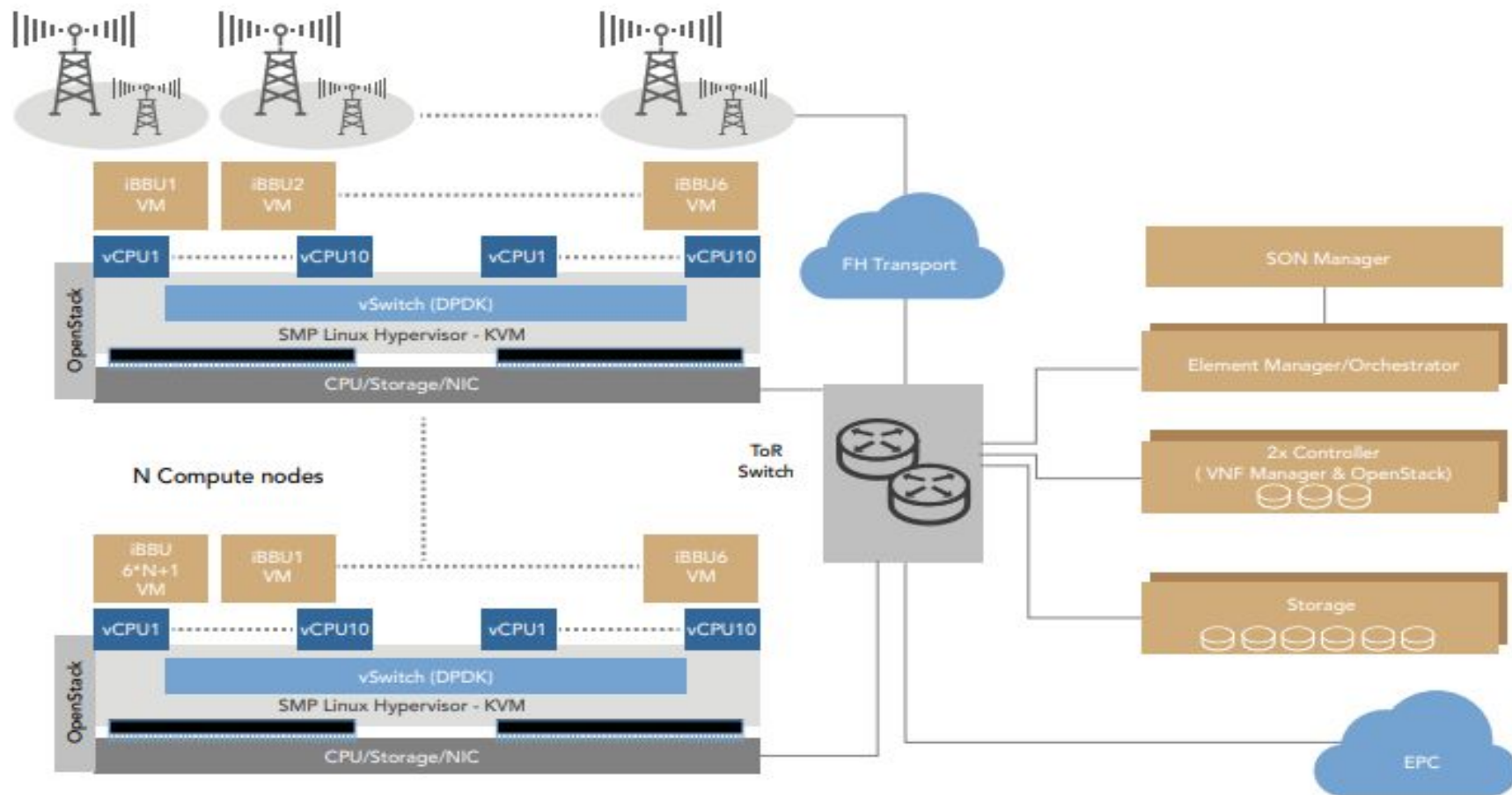
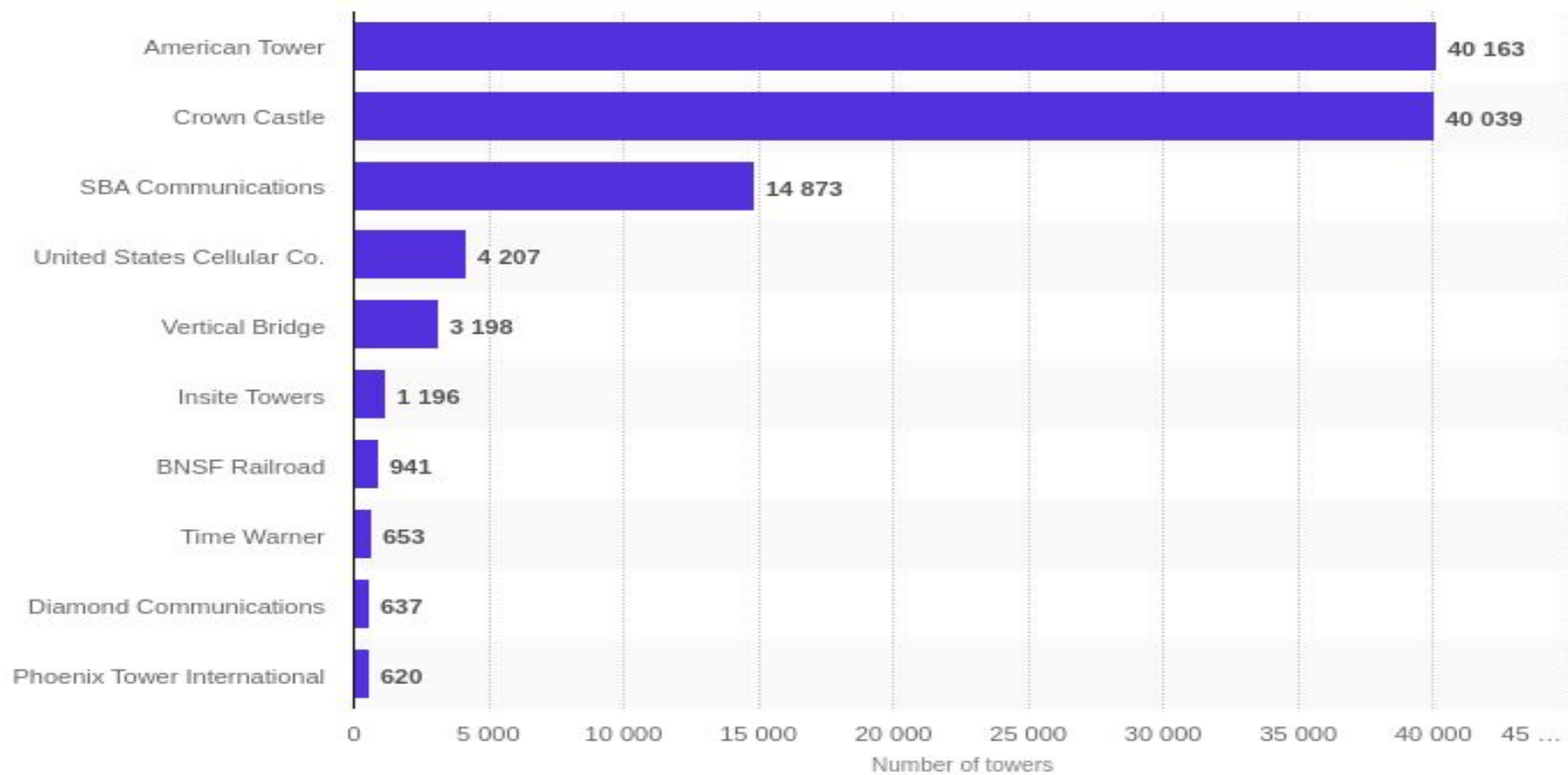


Figure 6. NFV architecture on commercial off-the-shelf hardware



© Statista 2020

[Show source](#)

[Additional Information](#)

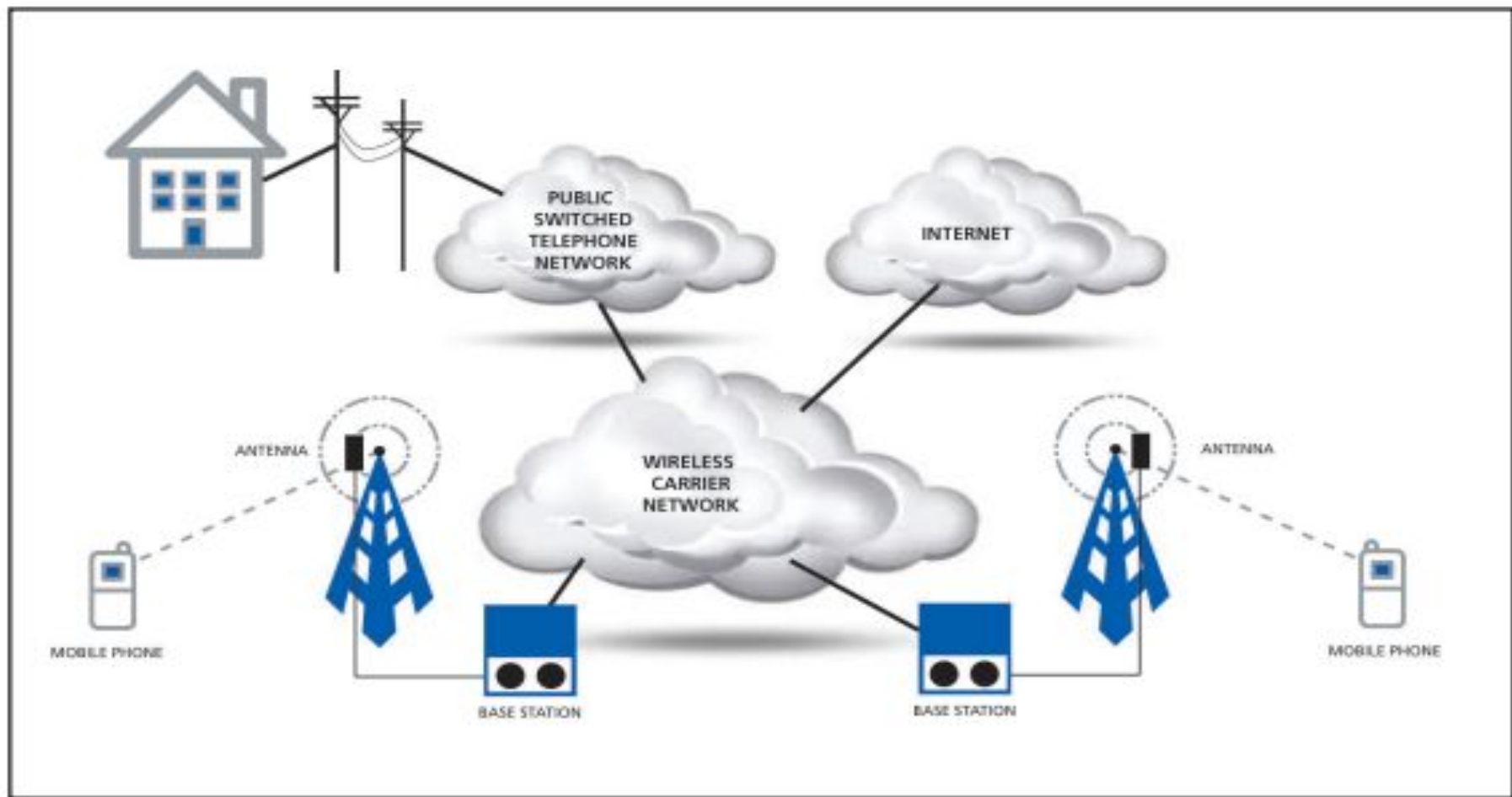


FIGURE 6: Wireless Communications Network



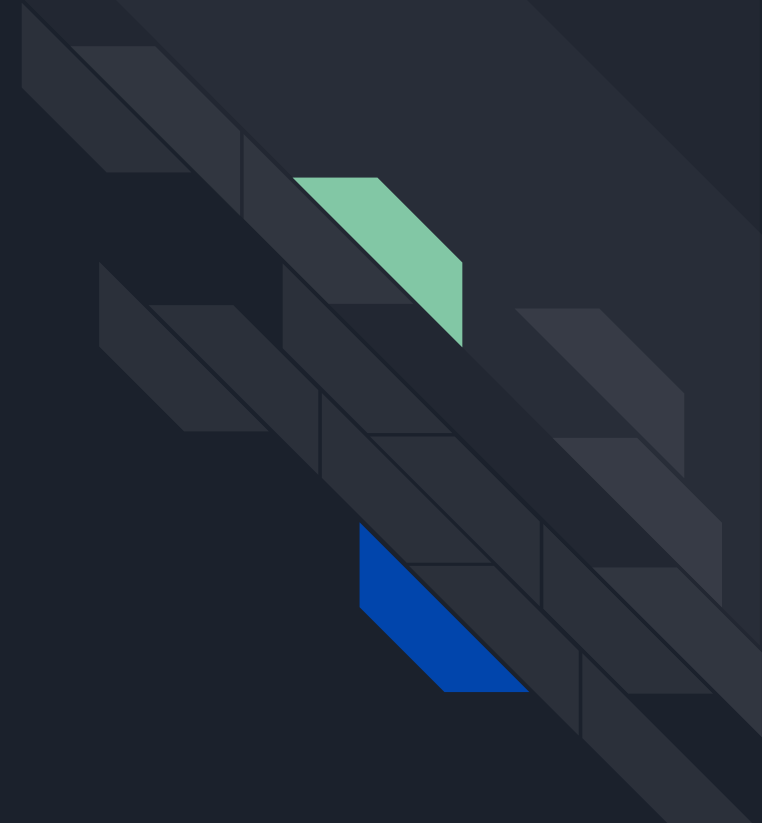
# vRAN Edge Computing Needs

- Handles base cell network service implementations
  - Voice, data, interactions with Telco's and internet
  - Account usage management and billing
  - Different carriers and Cell Network provider mapping
- Scale with devices and services required
  - Hot spot activity (i.e. Superbowl, significant news events,...)
  - Every increasing BW consumption
  - Proliferating number of devices
- Provide extreme reliability
  - Enterprise class reliability (6 - 9's of availability)
- Dynamic enough to change with load changes
  - Networked device is constantly changing
  - Multiple Carriers sharing infrastructure
- Simple enough for non-experts to support the 10s of thousand locations



Your taking a 500 mile road trip with your smartphone. You're receiving calls, sending SMS messages, and watching Youtube. (You're not driving)

What problems do you think the vRan SW needs to solve?





# Class Input - to be updated after 5 minutes

- Edge needs
  - ?
- Core needs
  - ?



# Edge Computing Breakout Discussion

- We'll break up into the same 4 teams as last week
- You look at the use case that was assigned to your group
  - City traffic management
  - Building environmental operations
  - Manufacturing floor operations
  - Remote weather station
- You create a list of all the potential benefits and challenges faced with edge computing
  - 15 minutes to complete
  - 10 minute break
  - A team member will read out results

# Micro-services and Kubernetes for Edge Computing





# Microservices Motivation

## Legacy App Issues

- Monolithic apps are hard to understand, maintain, and reuse
- Require complete OS environment
- Monolithic apps are more difficult to scale horizontally
- Lack of horizontal scaling in design harder to leverage cloud scaling
- ?

## Microservices Advantages

- Smaller usable services easier to develop, test, and support
- Can be deployed lightweight in a container vs. packaged requiring OS
- Easier scale with horizontal scaling capabilities of cloud
- Reuse and future leverage is easier
- Fits nicely into current cloud deployments
- ?

# Microservice Calling Methods

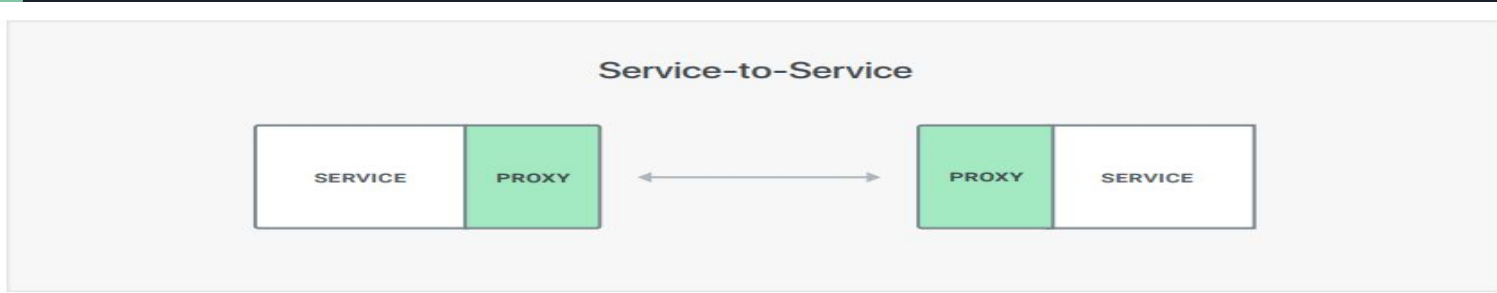


Figure 1: Service to service communication example. Notice the “sidecar” characteristic of this option



Figure 2: Asynchronous communication from service to service through a proxy

# Differences Between calling Methods

## Service-to-service

Microservices and clients **directly** consume and invoke other microservices.

Ideal for clients that require an **immediate** response, or need to aggregate multiple services together.

Done via HTTP, TCP/UDP, gRPC, etc.

Example: Making a request to retrieve an immediate response of some sort (ie, retrieve list of users).

## Asynchronous

Microservices and clients push events into an event collector that's being consumed by other Microservices.

Ideal for microservice-to-microservice communication for changing state **without** requiring an immediate response.

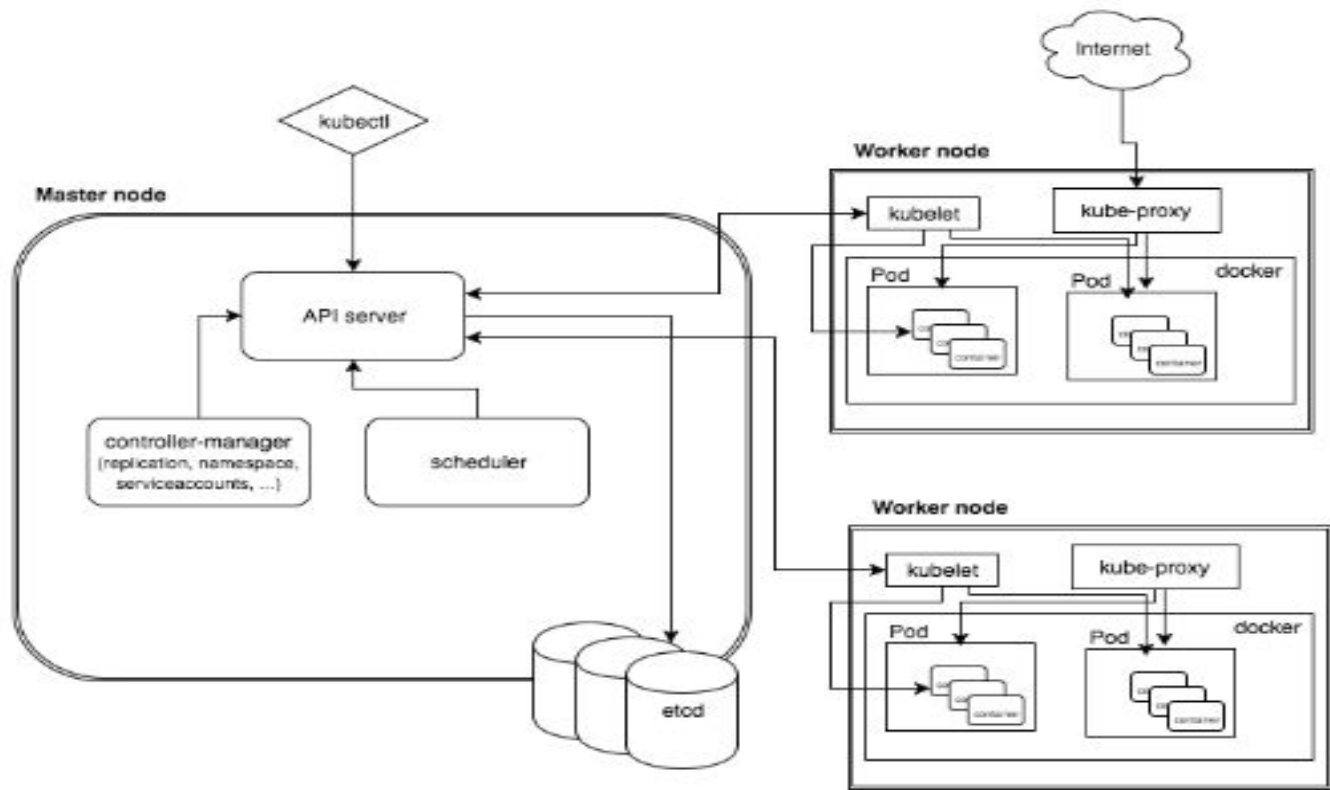
Done via Kafka, RabbitMQ, AWS SQS, etc.

Example: Making a request that doesn't require an immediate response (ie "orderCreated" event that triggers an invoice creation by other microservice).

# Kubernetes Concepts

- **Pod** - generally refers to one or more containers that should be controlled as a single application. A pod encapsulates application containers, storage resources, a unique network ID and other configuration on how to run the containers.
- **Service** - represents a logical set of pods and acts as a gateway, allowing (client) pods to send requests to the service without needing to keep track of which physical pods actually make up the service.
- **Volume** - applies to a whole pod and is mounted on all containers in the pod. Kubernetes guarantees data is preserved across container restarts, but ephemeral.
- **Namespace** (Virtual Cluster) Resources inside a namespace must be unique and cannot access resources in a different namespace. Also, a namespace can be allocated a resource quota to avoid consuming more than its share of the physical cluster's overall resources.
- **Deployment** - describes the desired state of a pod or a replica set, in a yaml file. The deployment controller then gradually updates the environment (for example, creating or deleting replicas) until the current state matches the desired state specified in the deployment file. For example, if the yaml file defines 2 replicas for a pod but only one is currently running, an extra one will get created.





High level Kubernetes architecture diagram showing a cluster with a master and two worker nodes

Source: <https://x-team.com/blog/introduction-kubernetes-architecture>

## Kubernetes Design Principles

Kubernetes was designed to support the features required by [highly available](#) distributed systems, such as (auto-)scaling, high availability, security and portability.

- **Scalability** - Kubernetes provides horizontal scaling of pods on the basis of CPU utilization. The threshold for CPU usage is configurable and Kubernetes will automatically start new [pods](#) if the threshold is reached. For example, if the threshold is 70% for CPU but the application is actually growing up to 220%, then eventually 3 more pods will be deployed so that the average CPU utilization is back under 70%. When there are multiple pods for a particular application, Kubernetes provides the [load balancing](#) capacity across them. Kubernetes also supports horizontal scaling of stateful pods, including NoSQL and RDBMS databases through Stateful sets. A Stateful set is a similar concept to a Deployment, but ensures storage is persistent and stable, even when a pod is removed.
- **High Availability** - Kubernetes addresses highly availability both at application and infrastructure level. Replica sets ensure that the desired (minimum) number of replicas of a stateless pod for a given application are running. Stateful sets perform the same role for stateful pods. At the infrastructure level, Kubernetes supports various distributed storage backends like AWS EBS, Azure Disk, Google Persistent Disk, NFS, and more. Adding a reliable, available storage layer to Kubernetes ensures high availability of stateful workloads. Also, each of the master components can be configured for multi-node replication (multi-master) to ensure higher availability.
- **Security** - [Kubernetes](#) addresses security at multiple levels: cluster, application and network. The API endpoints are secured through transport layer security (TLS). Only authenticated users (either service accounts or regular users) can execute operations on the cluster (via API requests). At the application level, Kubernetes secrets can store sensitive information (such as passwords or tokens) per cluster (a virtual cluster if using namespaces, physical otherwise). Note that secrets are accessible from any pod in the same cluster. Network policies for access to pods can be defined in a deployment. A network policy specifies how pods are allowed to communicate with each other and with other network endpoints.
- **Portability** - Kubernetes [portability](#) manifests in terms of operating system choices (a cluster can run on any mainstream Linux distribution), processor architectures (either virtual machines or bare metal), cloud providers (AWS, Azure or Google Cloud Platform), and new container runtimes, besides Docker, can also be added. Through the concept of [federation](#), it can also support workloads across hybrid (private and public cloud) or multi-cloud environments. This also supports availability zone fault tolerance within a single cloud provider.



# Kubernetes Advantages

- Supports bare metal, virtual, and all cloud deployments
- Linux processes and containers are only need an OS per deployed node
  - Whether node is physical or virtual
- Can run on any Linux distribution (of course we recommend RHEL, Industry leader)
  - Of course with dependent packages that are required
- No public public cloud vendor lock-in
  - Unless you start utilizing proprietary services
- Can migrate workloads between clouds
- Migrate workloads between clouds for cost and compliance issues

# AWS Greengrass





### AWS IoT Greengrass Core

Enables the local execution of AWS Lambda, messaging, device shadows, and security. AWS IoT Greengrass Core interacts directly with the cloud and works locally, even with intermittent connectivity



Device Type 01



Device Type 02



Device Type 03

Any device using Amazon FreeRTOS or AWS IoT Device SDK can be configured to interact with AWS IoT Greengrass Core via the local network



# AWS Greengrass System Features

- AWS IoT Greengrass Core SW and Device client SW runs on devices.
- Locally deployed Lambda functions and connectors are triggered by local events, messages from the cloud, or other sources
- AWS message system supported locally and maintains security model (thru disconnects)
- AWS IoT Greengrass protects user data:
  - Through the secure authentication and authorization of devices.
  - Through secure connectivity in the local network.
  - Between local devices and the cloud.
- Device security credentials function in a group until they are revoked, even if connectivity to the cloud is disrupted, so that the devices can continue to securely communicate locally.
- AWS IoT Greengrass provides secure, over-the-air updates of Lambda functions.



# AWS Greengrass SW Makeup

- Software distributions

- AWS IoT Greengrass Core software
- AWS IoT Greengrass core SDK

- Cloud service

- AWS IoT Greengrass API

- Features

- Lambda runtime
- Shadows implementation
- integration with services, protocols, and software

- Message manager

- Group management

- Discovery service

- Over-the-air update agent

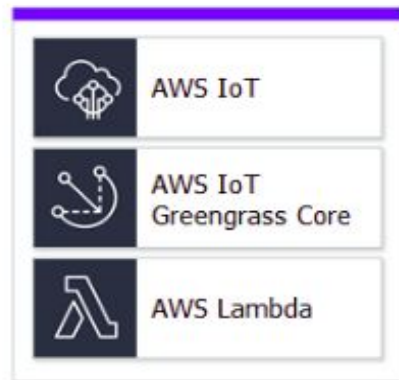
- Stream manager

- Local resource access

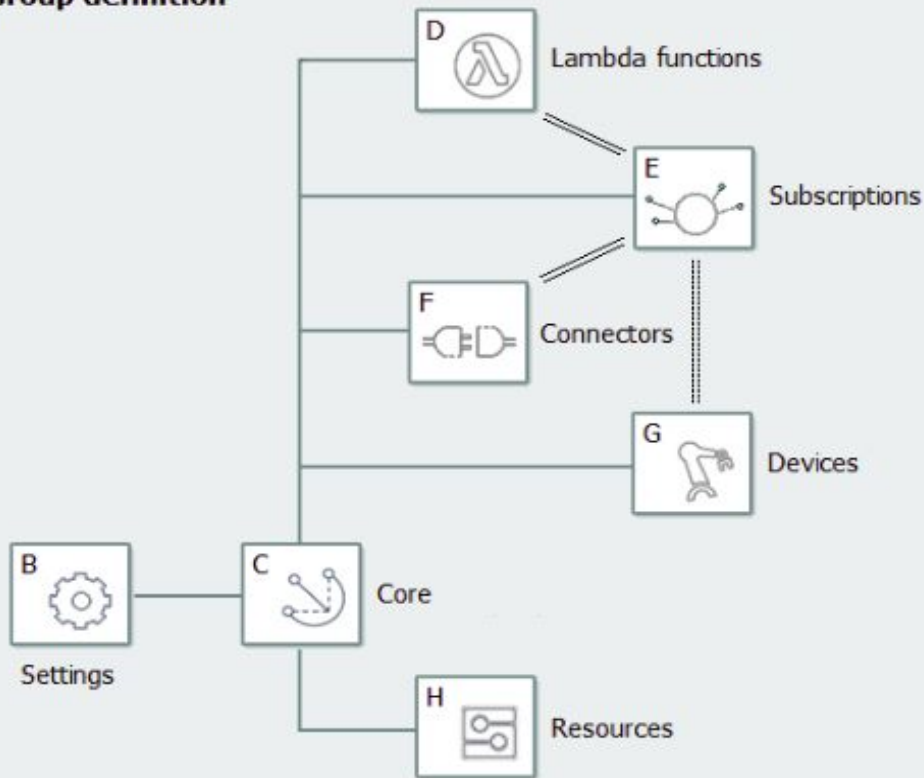
- Local machine learning inference

- Local secrets manager

- Connectors with built-in



### Group definition







# Greengrass SDK Availability

## AWS SDK

Use the AWS SDK to build applications that interact with any AWS service or AWS SDK in deployed Lambda functions

## AWS IoT Device SDK

Connect devices to AWS IoT or AWS IoT Greengrass services. Devices must know which AWS IoT Greengrass group they belong to and the IP address of the AWS IoT Greengrass core that they should connect to.

## AWS IoT Greengrass Core SDK

Enables Lambda functions to interact with the Greengrass core, publish messages to AWS IoT, interact with the local shadow service, invoke other deployed Lambda functions, and access secret resources.

## AWS IoT Greengrass Machine Learning SDK

Enables Lambda functions to consume machine learning models that are deployed to the Greengrass core as machine learning resources.



# AWS Greengrass Summary

## Pros

- Feature rich and ever expanding
- Security built in end to end
- You have access to all the AWS services and features

## Cons

- IOT device AWS lockin as devices must use AWS SDK
- Total rewrite to enable different IOT clouds



# Future Classes

- Leading Citrus and Cranberry IOT Supplier to discuss their IOT solution
- Presentation of your IOT device proposals
- Network Security discussion surrounding IOT
- Open Source SW an enabler for the IOT market
- Container discussions (Linux Container Support, Detailed Kubernetes)