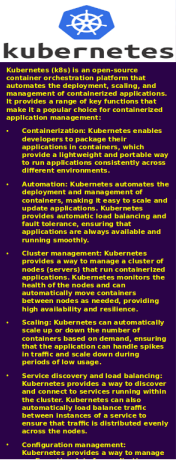
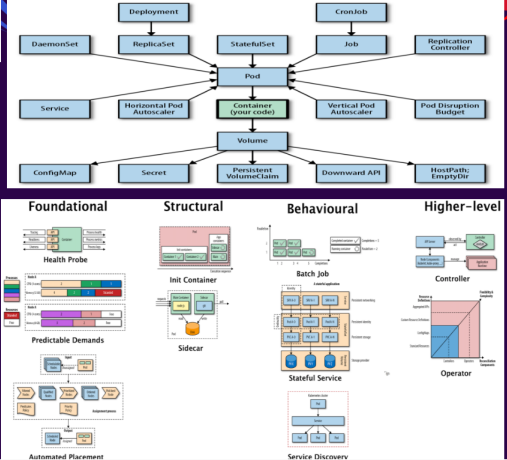
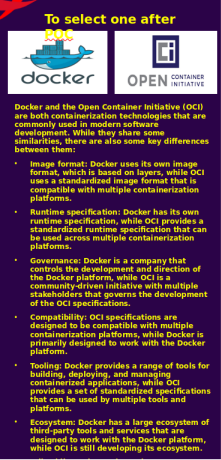
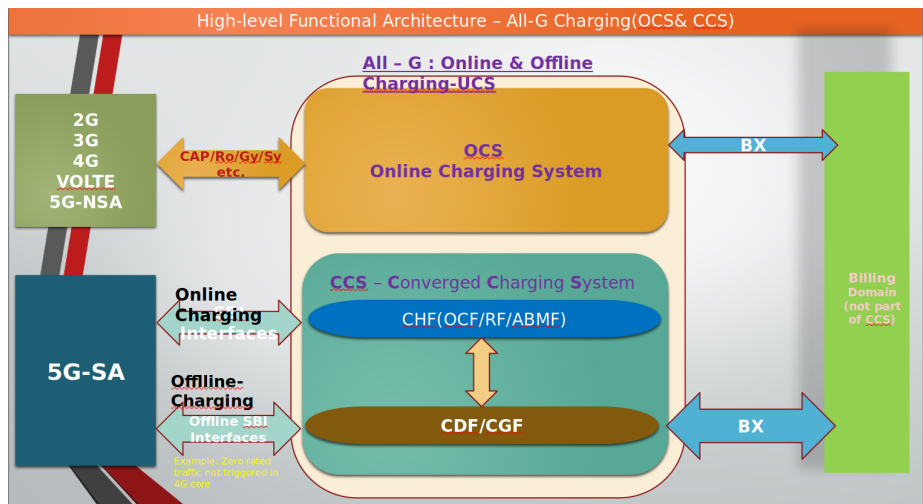


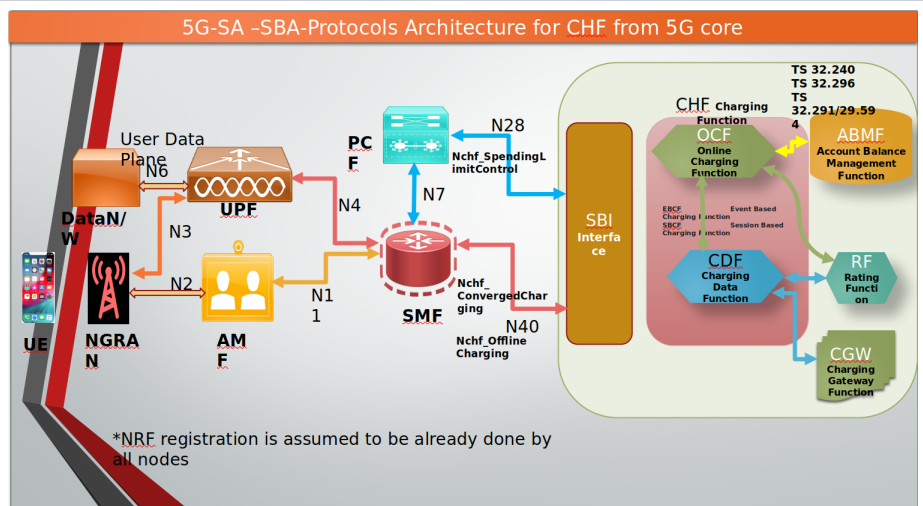
5G SBP : A charging system for 5G SA use cases using Open Source Products for SBP

	A charging system for 5G SA use cases using Open Source Products
Use Case Description: (Mandatory)	<p>For developing an open-source Universal Charging System that encompasses 5G and future generations of mobile networks which can be integrated to 5G Super Blueprint Architecture. This initiative has the potential to revolutionize the way countries manage their 5G networks, leading to substantial savings in both capital expenditure (CAPEX) and operational expenditure (OPEX).</p> <p>As you may be aware, the advent of 5G technology has ushered in a new era of connectivity and has immense potential to transform various sectors such as healthcare, transportation, education, and more. However, the deployment and management of 5G networks come with significant financial challenges, particularly in terms of charging systems and monetizing for network services.</p>
-Epic -Problem Statement (Mandatory)	<p>There is no reference Architecture in 3GPP for Charging system similar to ONAP . This is an initiative to build that and integrate with 5G SBP Architecture</p>
Blueprint Owner (Mandatory)	<i>Prajith Paran</i>
Users Stories (at least one (1) User Story is Mandatory)	<ul style="list-style-type: none"> a. b. c.
Interaction with other open source projects and components (Mandatory)	<div> <div>  <p>kubernetes</p> <p>Kubernetes (K8s) is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It provides a range of key functions that make it a popular choice for containerized application management.</p> <ul style="list-style-type: none"> Containerization: Kubernetes enables developers to package their applications in containers, which provide a lightweight and portable way to run applications consistently across different environments. Automation: Kubernetes automates the deployment and management of containers, making it easy to scale and update applications. Kubernetes provides automatic load balancing and fault tolerance, ensuring that applications are always available and running smoothly. Cluster management: Kubernetes provides a way to manage a cluster of nodes (servers) that run containerized applications. Kubernetes monitors the health of the nodes and can automatically restart containers on failed nodes or scale up/down the number of containers based on demand, ensuring that the applications can handle spikes in traffic and scale down during periods of low usage. Service discovery and load balancing: Kubernetes provides a way to discover and connect to services running within the cluster. Kubernetes can also automatically load balance traffic between instances of a service to ensure that traffic is distributed evenly across the nodes. Configuration management: Kubernetes provides a way to manage </div> <div>  <p>UCS(CCS/OCS/CHF) : Container Orchestration and related patterns & products</p> <p>The diagram illustrates the architecture of container orchestration and related patterns and products. It shows a central 'Pod' component connected to various other components. The components are organized into four categories: Foundational, Structural, Behavioural, and Higher-level.</p> <ul style="list-style-type: none"> Foundational: Includes Health Probe, Predictable Demands, and Automated Placement. Structural: Includes Init Container, Sidecar, and Stateful Service. Behavioural: Includes Batch Job, Stateful Service, and Service Discovery. Higher-level: Includes Controller and Operator. <p>Other components shown include Deployment, CronJob, DaemonSet, Replicaset, StatefulSet, Job, Replication Controller, Service, Horizontal Pod Autoscaler, Vertical Pod Autoscaler, Pod Disruption Budget, ConfigMap, Secret, Persistent VolumeClaim, Downward API, and HostPath/EmptyDir.</p> </div> <div>  <p>To select one after</p> <p>docker OPEN CONTAINER INITIATIVE</p> <p>Docker and the Open Container Initiative (OCI) are both containerization technologies that are commonly used in modern software development. While they share some similarities, there are also some key differences between them.</p> <ul style="list-style-type: none"> Image format: Docker uses its own image format, which is based on layers, while OCI uses a standardized image format that is compatible with multiple containerization platforms. Runtime specification: Docker has its own runtime specification, while OCI provides a standardized runtime specification that can be used across multiple containerization platforms. Governance: Docker is a company that controls the development and direction of the Docker platform, while OCI is a community-driven initiative with multiple stakeholders that governs the development of the OCI specifications. Compatibility: OCI specifications are designed to be compatible with multiple containerization platforms, while Docker is primarily designed to work with the Docker platform. Tooling: Docker provides a range of tools for building, deploying, and managing containerized applications, while OCI provides a set of standardized specifications that can be used by multiple tools and platforms. Ecosystem: Docker has a large ecosystem of third-party tools and services that are designed to work with the Docker platform, while OCI is still developing its ecosystem. </div> </div>
Resources -people (Mandatory)	<p>Resources (people) to execute on the blueprint:</p> <ul style="list-style-type: none"> <i>Prajith Paran (initial draft)</i> ...
Steps to Realization	<ul style="list-style-type: none"> TBD

(Mandatory)



(Mandatory)



(Mandatory)

☒ Yes

- Resources and Lab setup

or

☐ No

(Mandatory)

- 1 to 2 years depending on contributions

(Mandatory)

- *Once implemented in one of the Telcos (MVNOs initially) , this can be considered as TELCO grade*

<p>Blueprint Outputs</p> <p>(Mandatory)</p>	<p>check all that apply:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Code repository <input type="checkbox"/> Configuration files (e.g. Helm charts, etc.) <input type="checkbox"/> Upstreaming to relevant projects <input type="checkbox"/> Continuous Integration <input type="checkbox"/> Test requirements and test results (if applicable) <input checked="" type="checkbox"/> Documentation: <ul style="list-style-type: none"> <input type="checkbox"/> Overview and Theory of Operation (i.e., what does it do?) <input type="checkbox"/> Deployment and setup <input type="checkbox"/> Videos <ul style="list-style-type: none"> <input type="checkbox"/> demo <input type="checkbox"/> lab setup/behind the scenes <input type="checkbox"/> other
<p>High-level timeline</p> <p>(Mandatory)</p>	<ul style="list-style-type: none"> • Month that build can begin: <i>enter month/year</i> • Approximate duration of build: <i>enter number of weeks or months</i> • Approximate completion of all outputs: <i>enter month/year</i>
<p>Links to existing documentation (Build Guide, Slideware, etc), if available (optional).</p>	<p>https://www.linkedin.com/feed/update/urn:li:activity:7065016516987035648/</p>
<p>Links to existing demo /video, if available (optional).</p>	<p>https://www.linkedin.com/feed/update/urn:li:activity:7065016516987035648/</p>
<p>Links to existing code /repos, if available (optional).</p>	<p>docker pull prajithparan/universalchargingsystem-ucs.v.08 (Documents and initial code generated for N40/N28 Interfaces.)</p> <p>docker pull prajithparan/open5gcore_v5</p>