Providing Different Levels of QoS

UMOBILE PROJECT REVIEW

BRUSSELS, 20 OCTOBER 2016

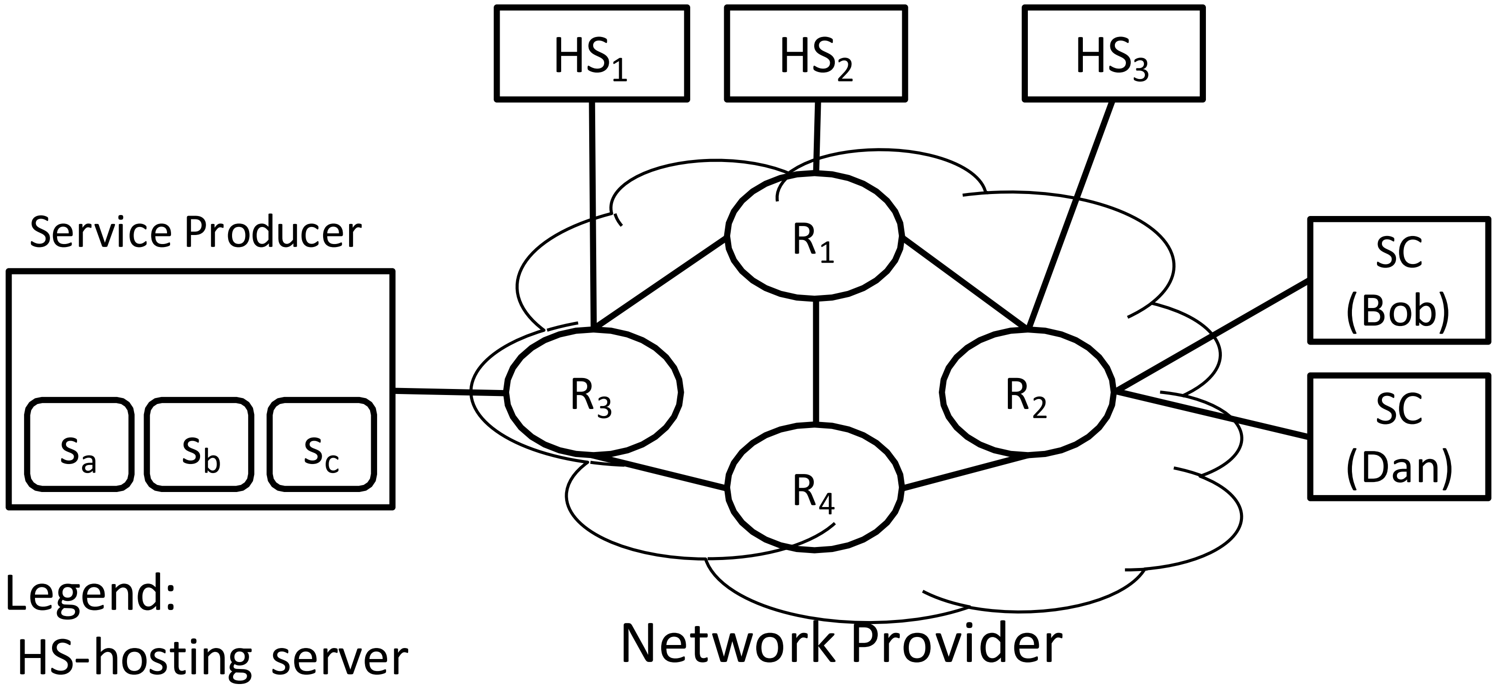
UNIVERSITY OF CAMBRIDGE

UMOBILE

Objective of Task 4.1

- The aim of Task 4.1 of WP4 is to abstract away network impairments from the user's view (as much as possible).
- To develop the mechanisms providing different levels of QoS: less-than-best effort, best effort and guaranteed.
- In our view and to simplify the problem, the issue can be tackled at different levels of the software stack

UMOBILE QoS Model



Legend:
HS-hosting server
s_a, s_b, s_c- services (applications) with different QoS requirements.
R_i- routers SC- service consumer

- There are different business models for service provisioning.
- We assume a business model with four stakeholders:
  - Service Producer (SP)
  - Service Distributor (SD)
  - Internet Service Provider (ISP)
  - Service Consumer (SC)

Why we choose this model?

- This is one of the simplest but realistic models.
- The network provider is responsible for and in full control of the QoS delivered by the services.
- He/she can deploy QoS mechanisms as needed.
- Commercial network providers also apply this model for VoD service

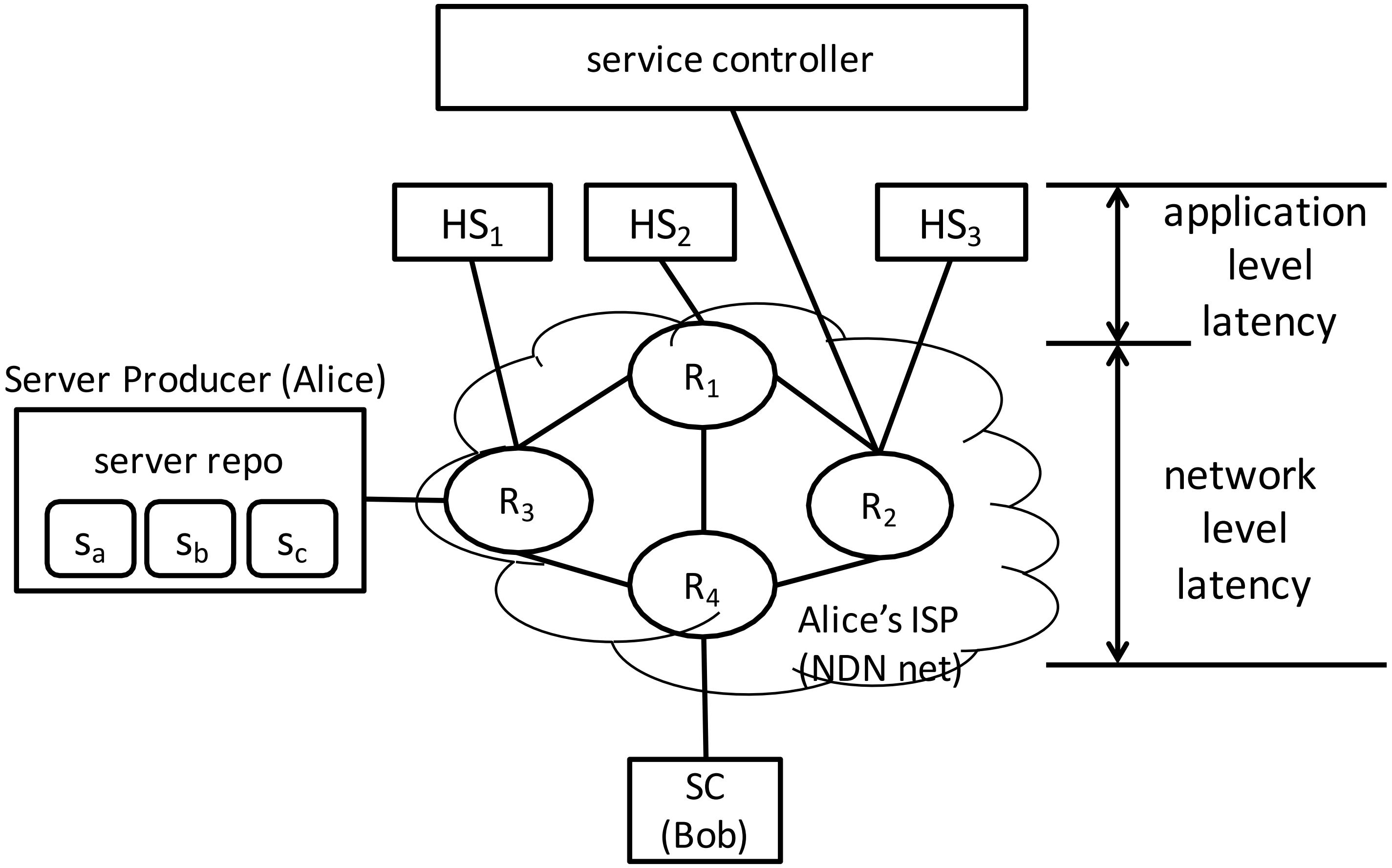
Relevant QoS Parameters in UMOBILE

- Potential QoS parameters are:
  - Latency
  - Availability
  - Throughput
  - Time to repair (service recovery), etc.
- We are focused (for the time being) on latency and availability, It seems to be universally relevant.

Classes of Services and QoS Mechanisms

Classes of Services

- High Priority
  - **Latency:** Low, **Availability:** Guarantee
- Best effort
  - **Latency:** Conventional, **Availability:** No Guarantee
- Less than best effort (Explicit Delay Tolerance)
  - **Latency:** No Guarantee , **Availability:** Guarantee



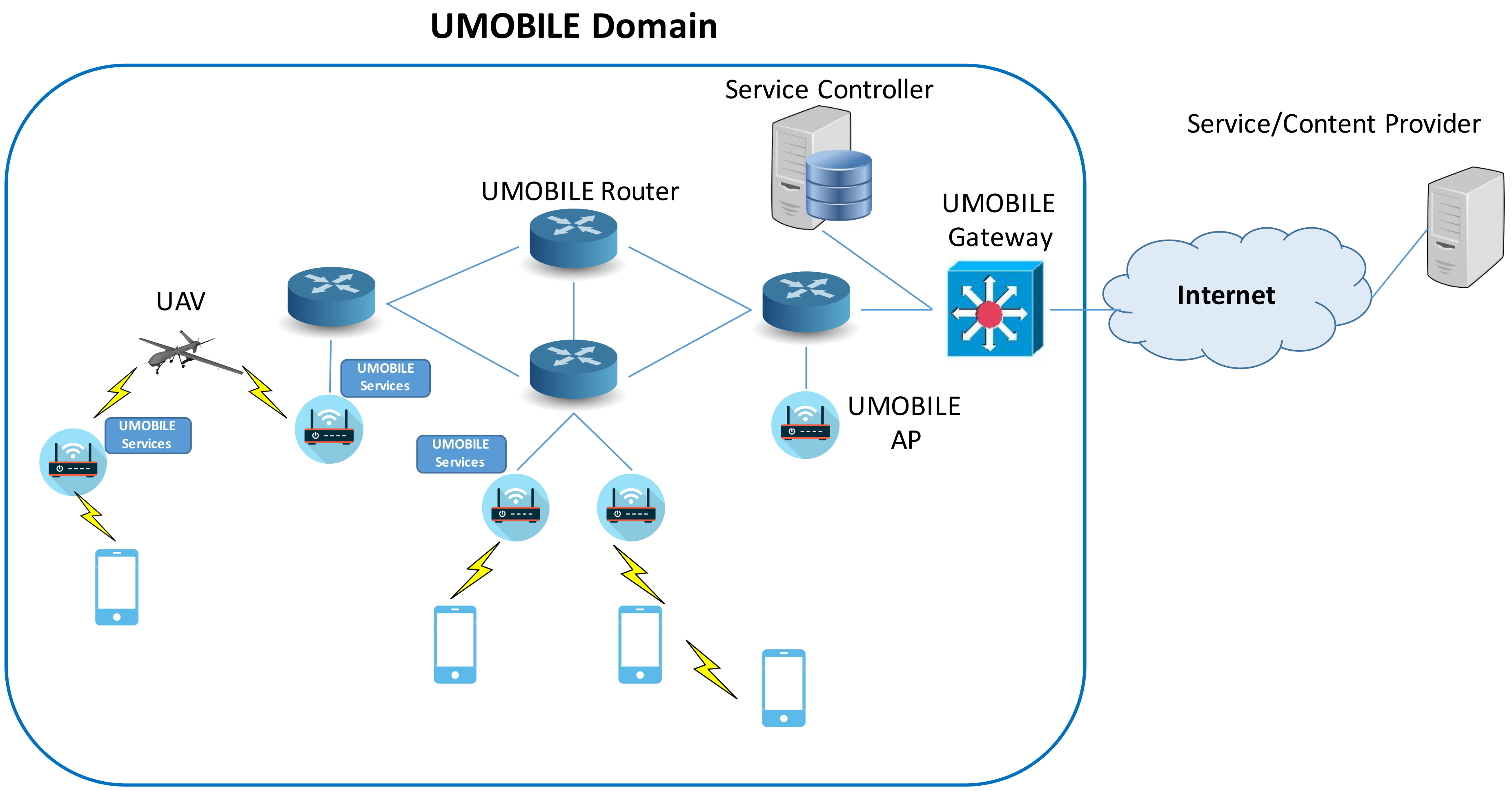
Multi-layers QoS mechanisms

- Development of application level QoS mechanisms
  - Manipulation of the application and its deployment: mainly **service migration/replication.**
  - Take advantage of ICN abstractions (for ex. in-network caching, data replication and multicast).
- Development of network level QoS mechanisms
  - Manipulation of network packets: mainly **congestion control**
  - Provide opportunistic communications through **DTN tunnelling**

UNIVERSITY OF CAMBRIDGE

UMOBILE

Application level QoS: Service Migration Issues



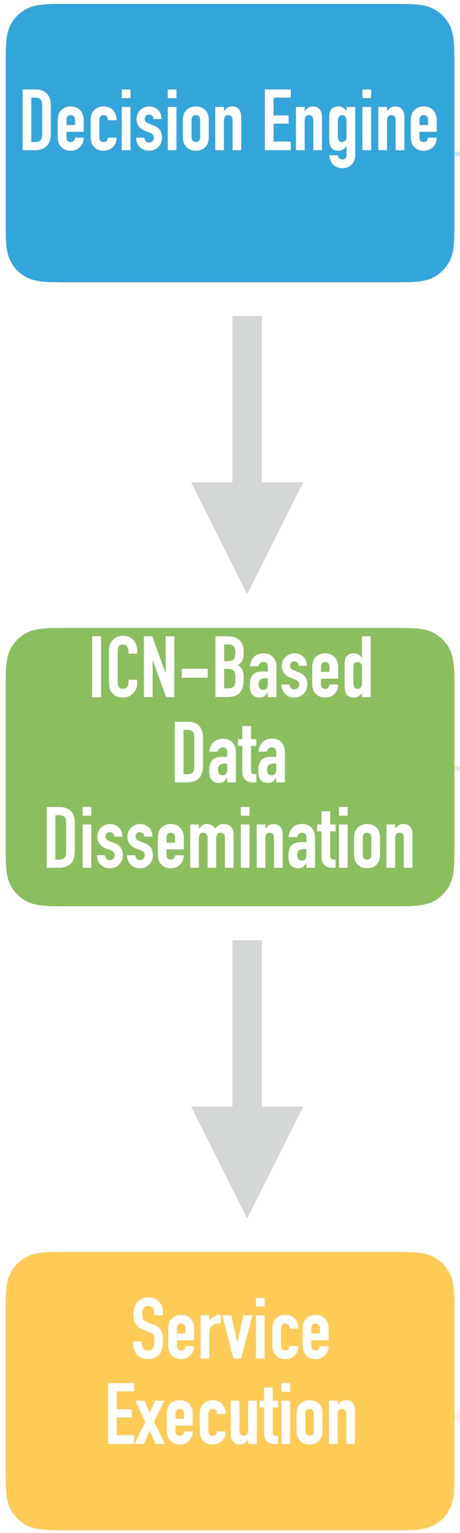
Common sense suggests that to reduce latency the service should be deployed close to the end user (edge computing).

"Close" can be interpreted as physical (geographical distance) and logical (link bandwidth) proximity.

What are the research questions?

- Determine a good place to migrate/replicate the service so that latency is reduced.
- Redirect users' requests to the right replica so that latency is reduced.
- Optimise the cost of service migration: Storage, Migration traffic (migrating the service across the network –> can cause congestion)
- We have started with replica placement.

High level design of service migration



Decision Engine

- Decide when and where to migrate/replicate the services
  - Improve QoS (e.g., access latency, availability)
  - Minimise the cost of migration/replication (e.g., storage, migration traffic)
  - Provide different classes of QoS (D4.4)

ICN-Based Data Dissemination

- Name based routing
  - Decouple the location of producer and consumer
  - Multicast by name
- Service/Content Distribution (Migrate service to the edge)
  - Benefit from in-network caching of NDN
  - Push communication model
- Service/Network monitoring
  - Pull communication model

Service Execution

- Operating the lightweight services with service virtualisation
  - Understanding the scalability issues and performance
  - Identifying the critical constrains of the system for deploying services

UNIVERSITY OF CAMBRIDGE

UMOBILE

Service Migration: Service Execution

UMOBILE Access Point (SEG)

UMOBILE Innovation

- Service is executable (edge computing)
- Service/Content is cacheable (edge caching)
- Supporting service migration
- Supporting ICN-DTN

- Benchmarking **scalability**[2]
  - How **many containers** can be supported by a specific raspberry Pi ?
  - How many **user requests** can be supported by a single container?

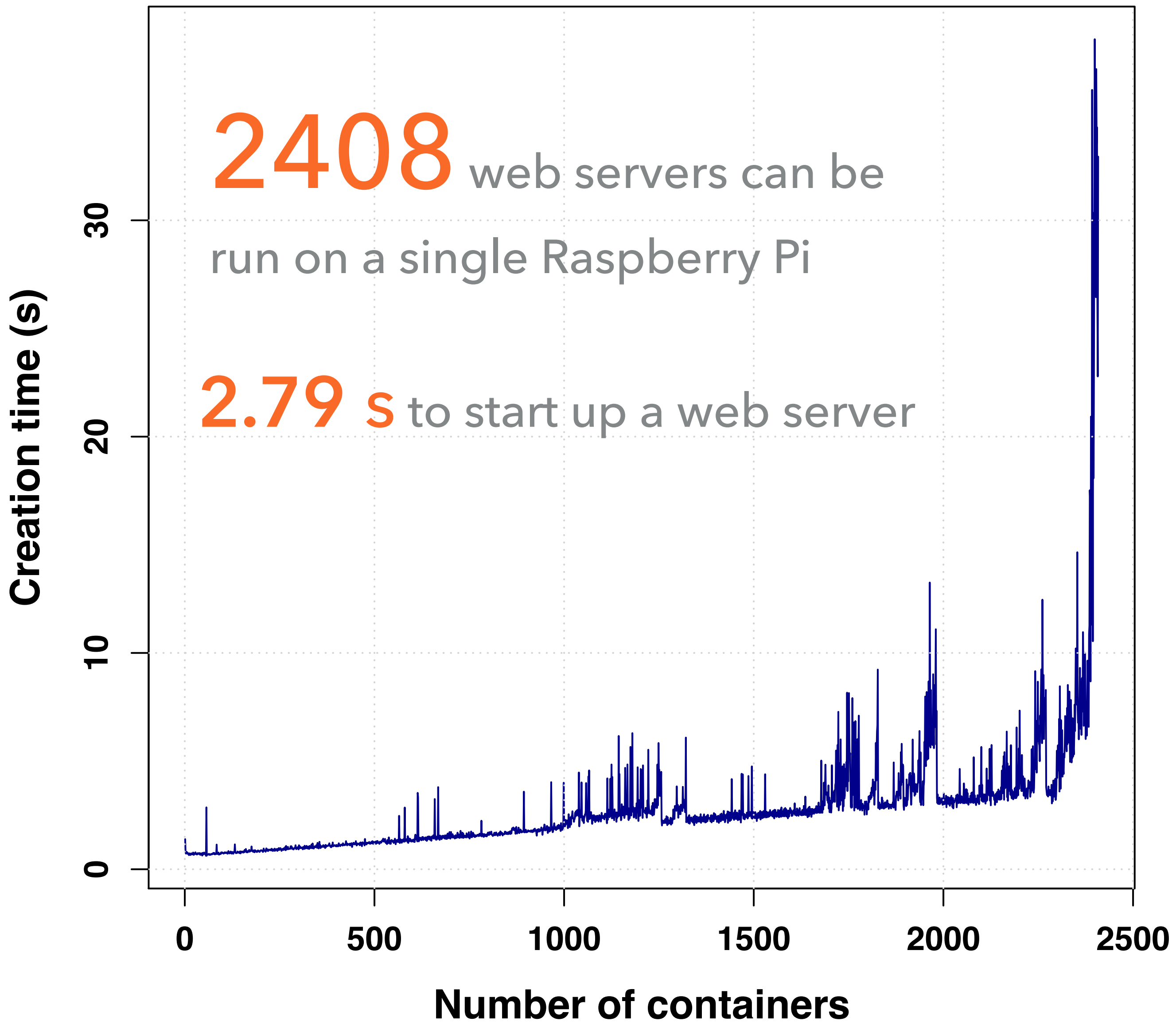
UNIVERSITY OF CAMBRIDGE

[2] A. Sathiaseelan, A .Lertsinsrubtavee, A. Jagan, P. Baskaran, J. Crowcroft, "Cloudrone: Micro Clouds in the Sky", ACM Mobisys Dronet, 2016.

UMOBILE

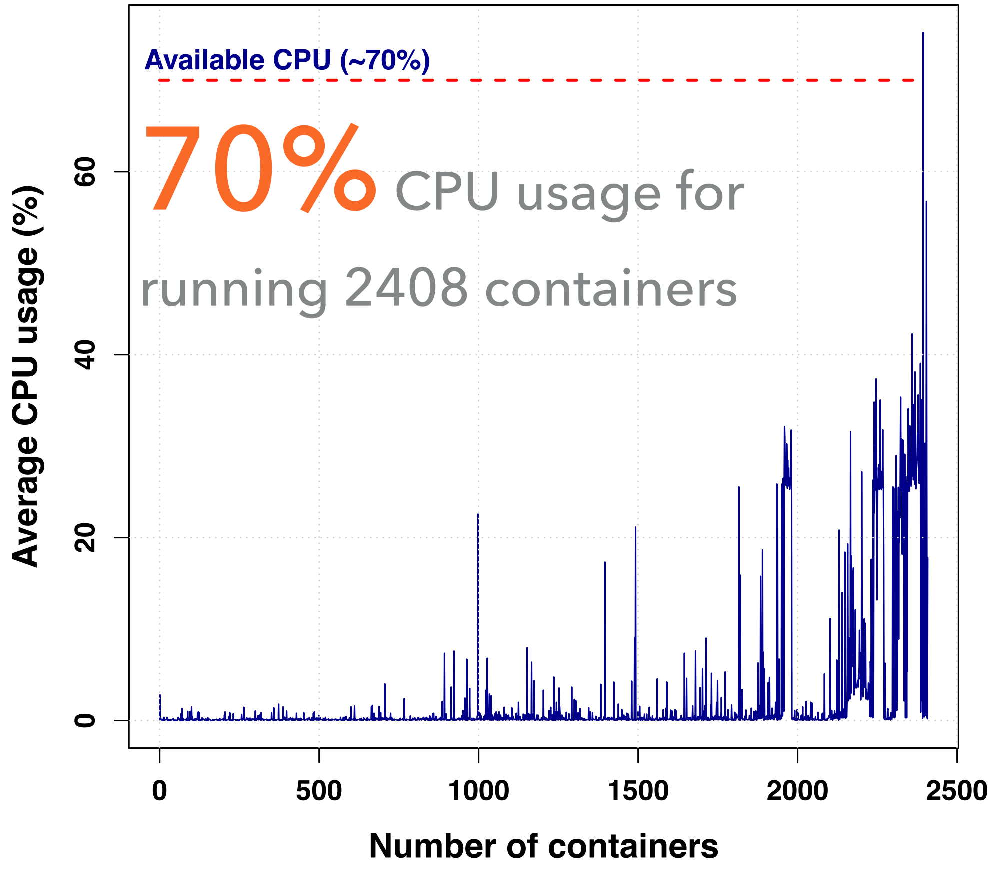
Scaling up the number of deployed containers within a PI

- Testing with simple web server image (html + a small jpg)
- Container size is about 90 KB

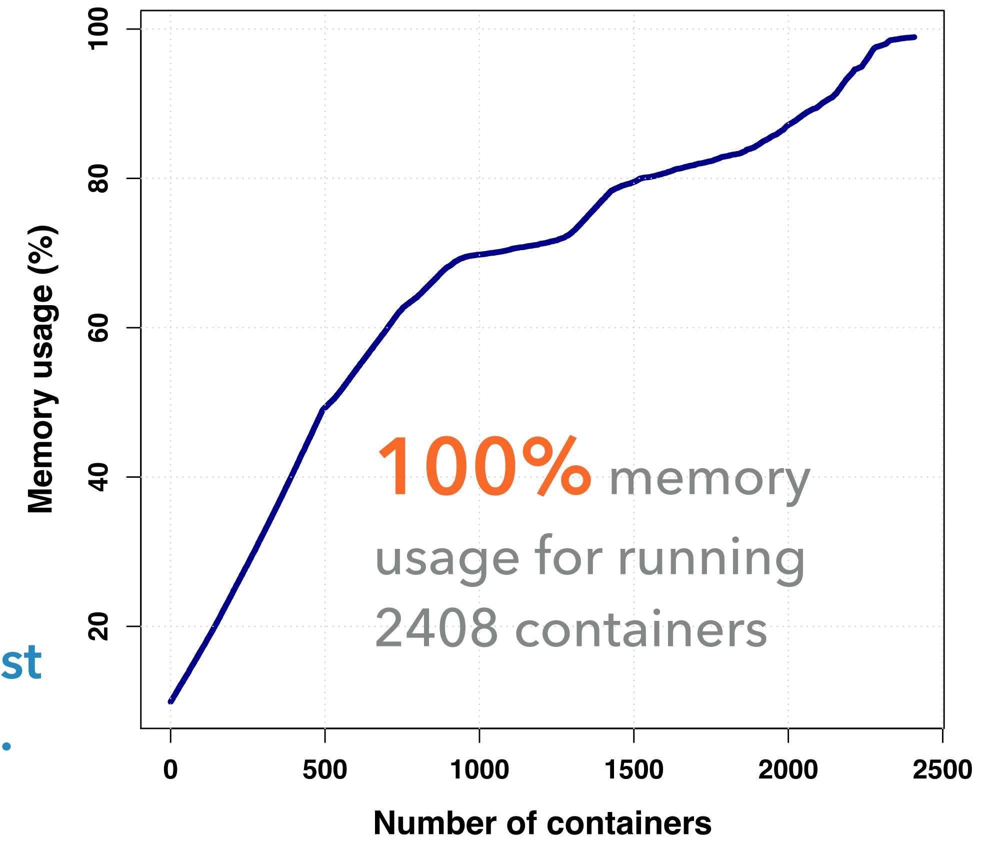


2408 web servers can be run on a single Raspberry Pi

2.79 s to start up a web server



70% CPU usage for running 2408 containers

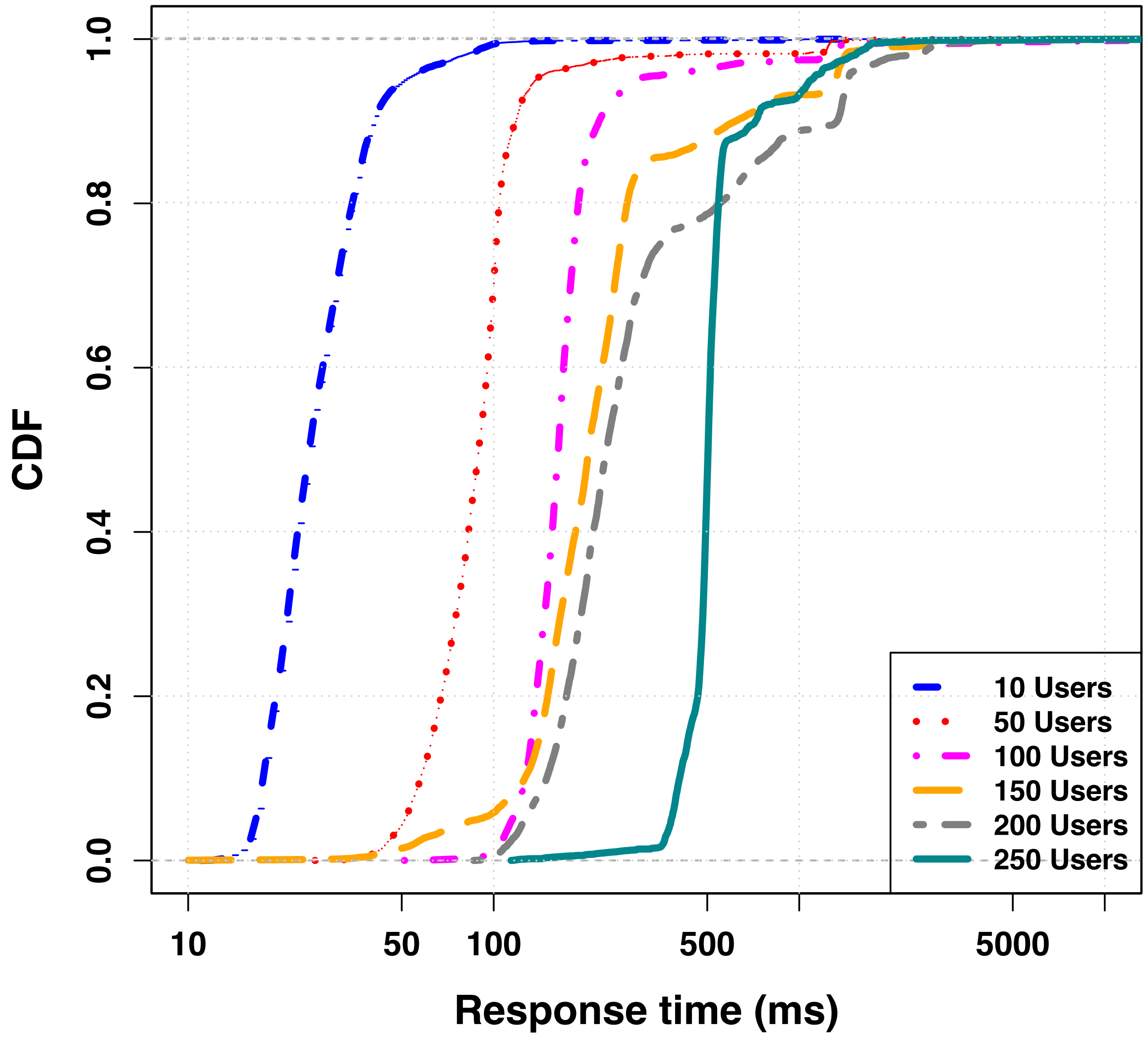


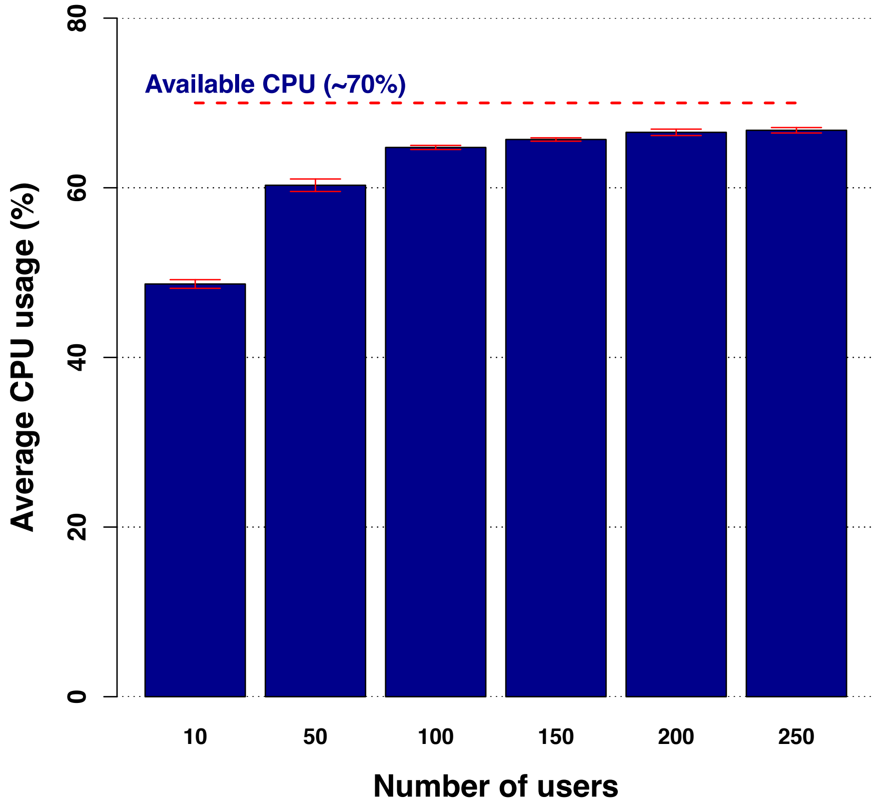
100% memory usage for running 2408 containers

The initial memory usage before creating the first container was about 98 KB (a PI has 1GB RAM).

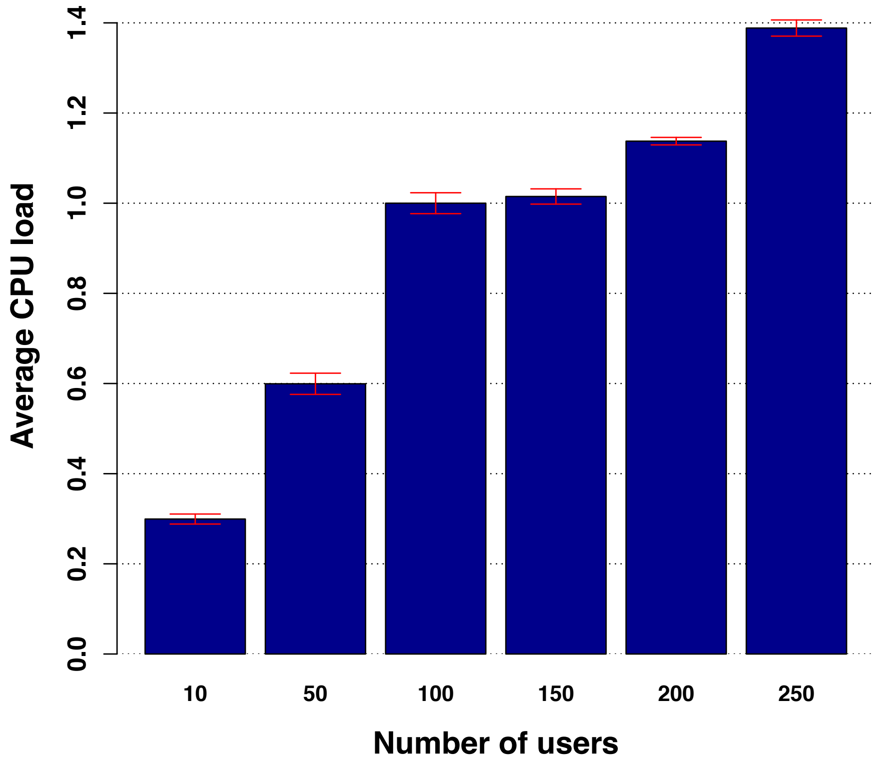
Scaling up the number of users accessing a single service

- Scaling the number of concurrent users from 10 to 250
- 10,000 transactions were set per experiment





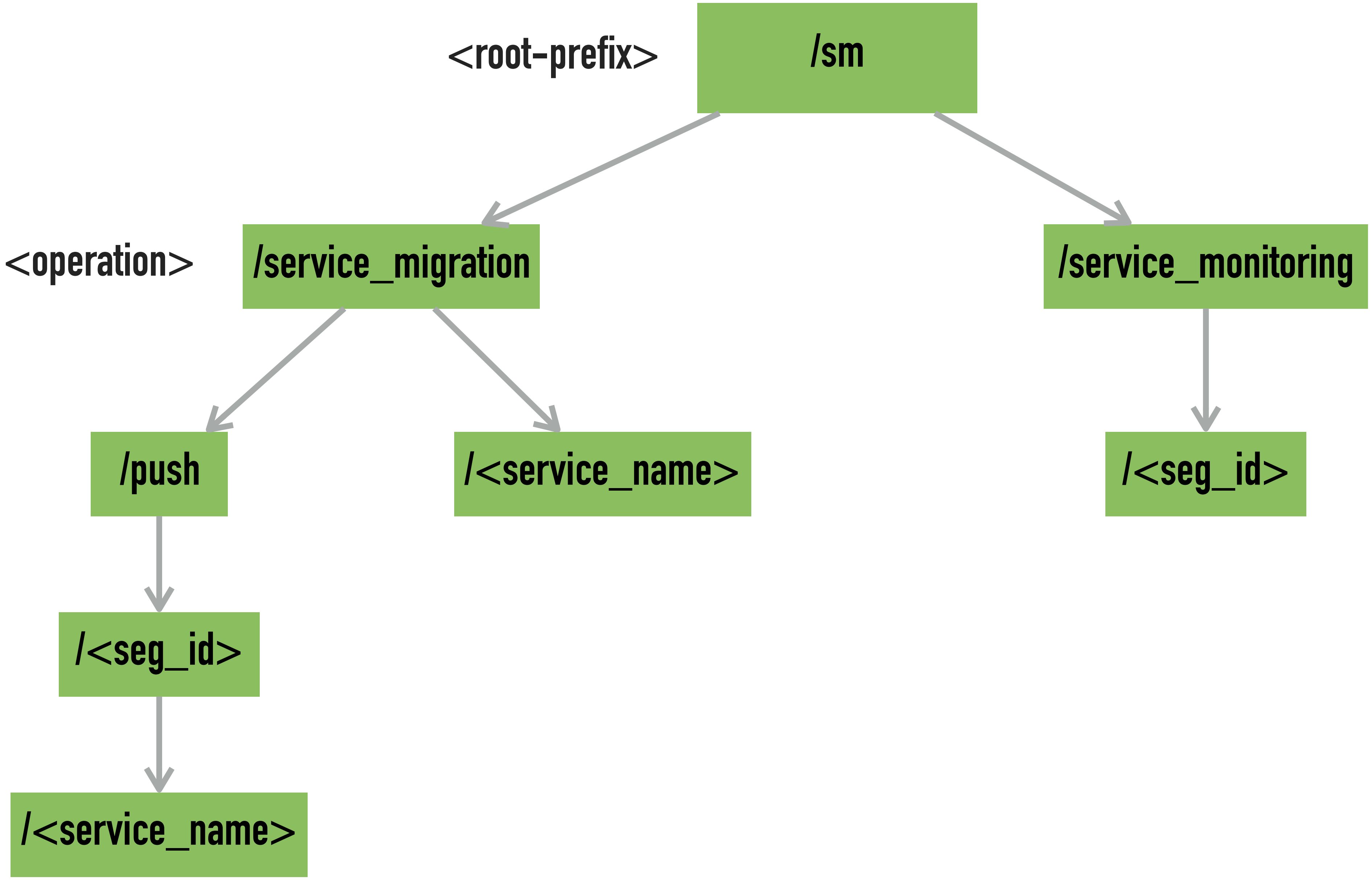
CPU load is increased up to 90%



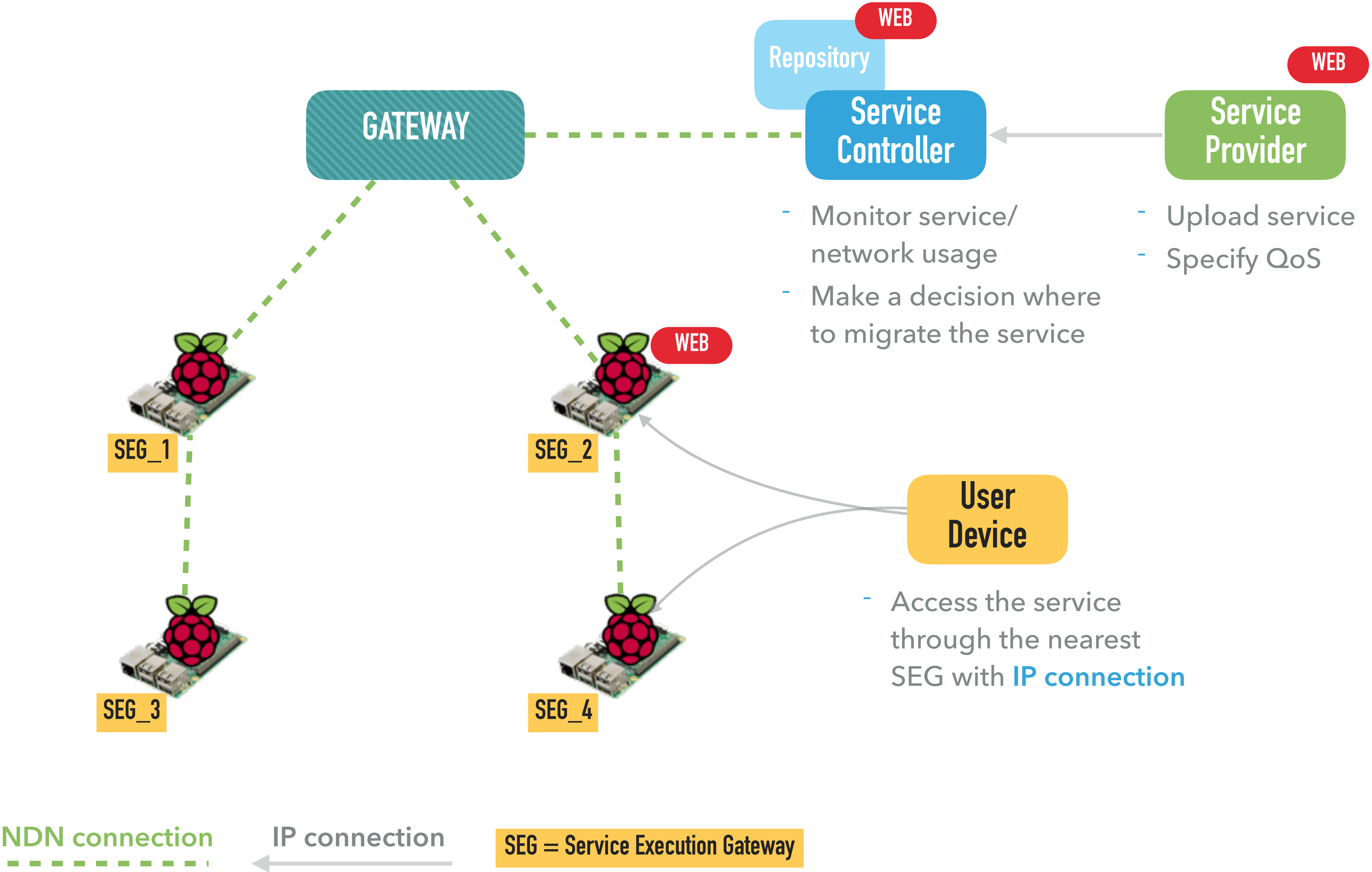
- High response time when number of users is large
- The amount of computational work that CPU needs to process (CPU load) is increased.

Communication Model and Naming Scheme

Operation	Model	Nature	Producer	Consumer
Monitoring	Pull based	Many to One	All SEGs	Service Controller
Migrating Services	Push and Pull based	One to Many	Service Controller	SEGs



An example of Service Migration scenario



Service Controller:

- Monitor service/ network usage
- Make a decision where to migrate the service

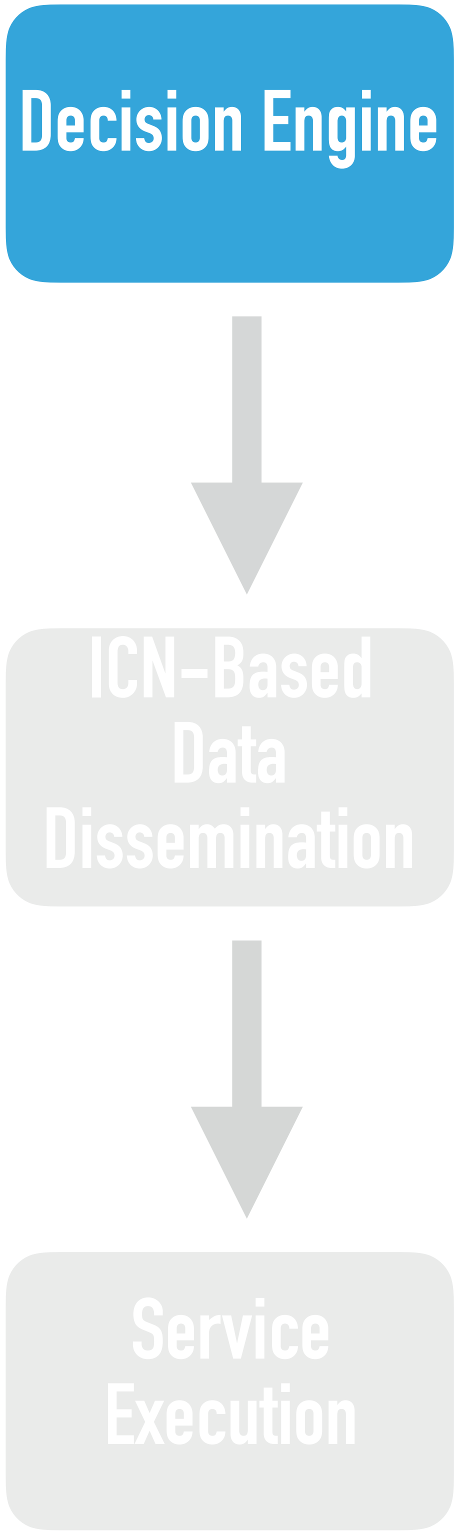
Service Provider:

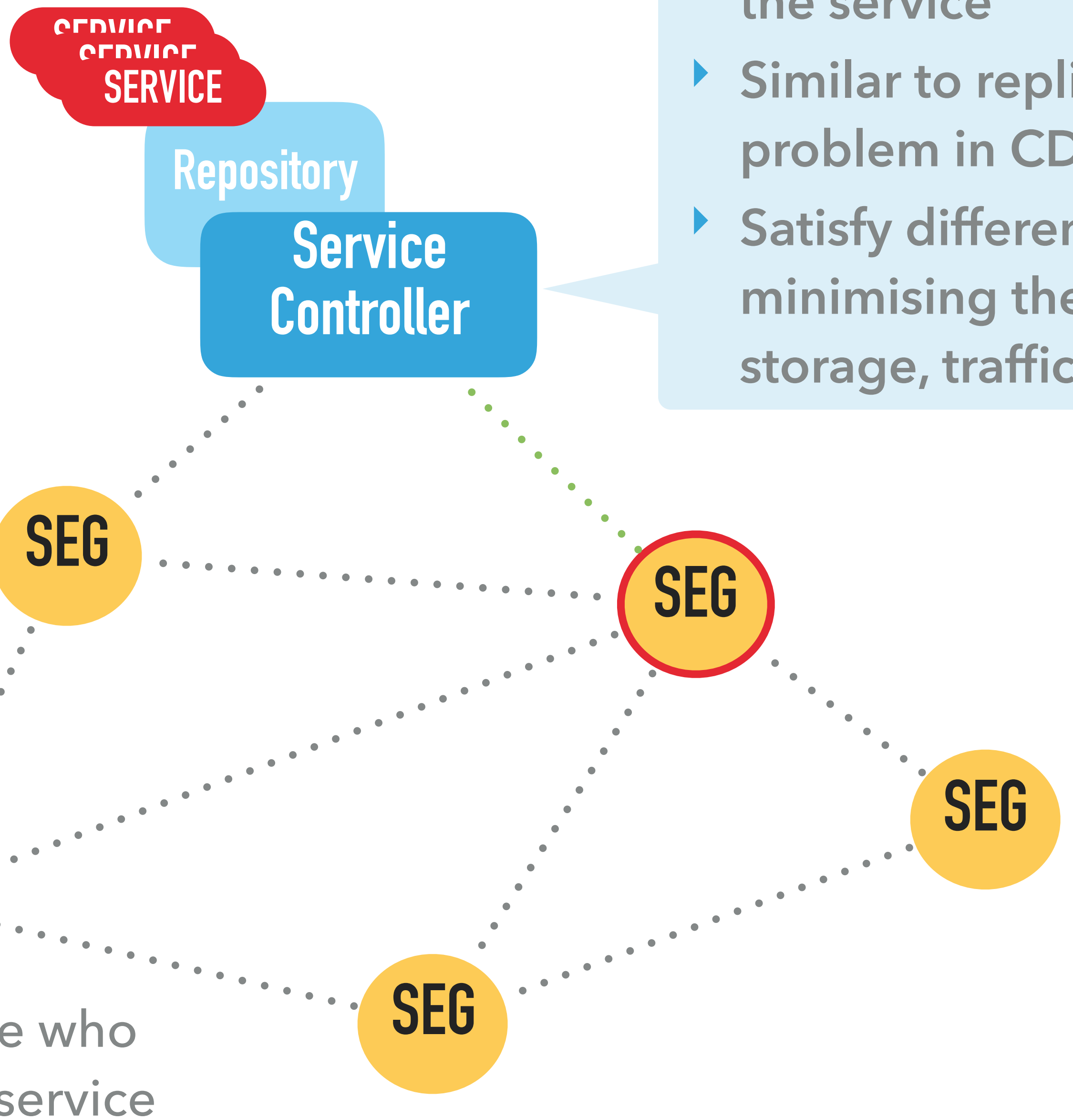
- Upload service
- Specify QoS

User Device:

- Access the service through the nearest SEG with **IP connection**

Service Migration: Decision Engine



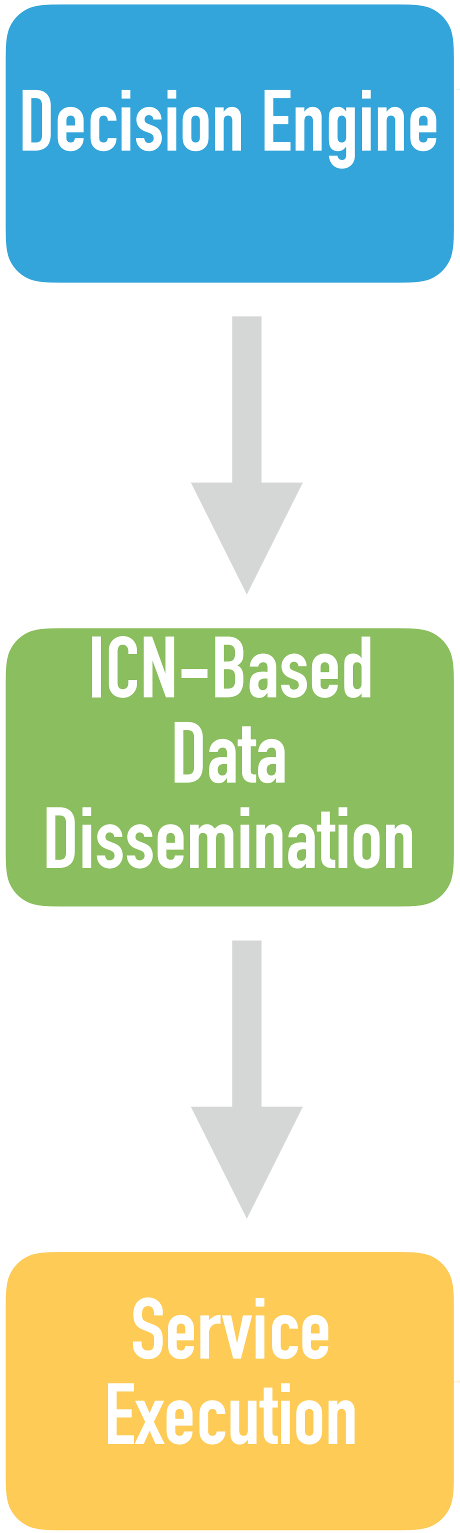


- Decide where/when to migrate the service
- Similar to replica placement problem in CDN[3]
- Satisfy different QoS levels while minimising the cost (e.g., storage, traffic)

Selected node who operates the service

[3] Xueyan Tang and Jianliang Xu, "QoS-aware replica placement for content distribution," in IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 10, pp. 921-932, Oct. 2005.

Status of Service Migration as of Month 18



Decision Engine

- Identify and measure critical constraints of the system [Done]
  - These parameters include CPU load, Memory, number of users, storage (from service execution benchmarking)
- Identify the QoS requirements [On going]
- Develop heuristic algorithms for decision engine [On going]

ICN-Based Data Dissemination

- Implemented the service migration frame work [Done]
- Network/Service monitoring using pull based communication [Done]
- Service Virtualisation over NDN (Docker and NDN integration) [Done]
- Multicast communication through named based routing [Done]
- Optimising the traffic through in-network caching [On Going]
- Redirect users' requests to the closet replica over NDN [On Going]

Service Execution

- Operating the lightweight services with service virtualisation [Done]
- Understanding the scalability issues and performance [Done]

DEMO

Service Migration with Push services