VM Placement and Workload Assignment for Mobile Edge Computing

Wei Wang

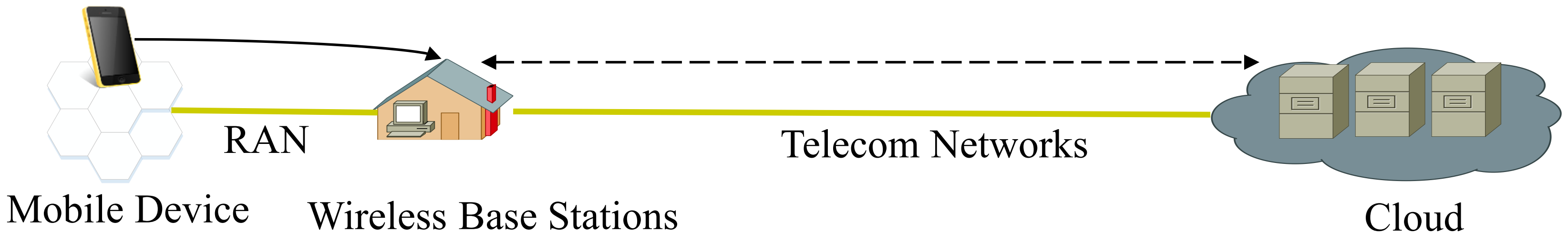
BUPT Ph.d candidate & UC Davis visiting student
Email: weiw@bupt.edu.cn, waywang@ucdavis.edu

Group Meeting, Apr. 7, 2017

Contents

Introduction

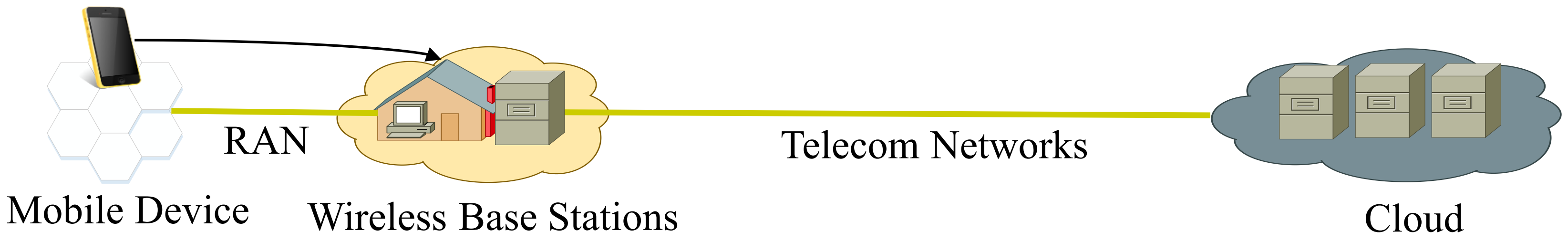
- Cloud Computing
  - Cloud computing is a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand.
  - Computer processing resources and data are usually deployed in centralized datacenters, which is far away from end users.
  - Drawbacks, long-distance network connection between user and cloud result in long service latency.



UCDAVIS UNIVERSITY OF CALIFORNIA

Introduction

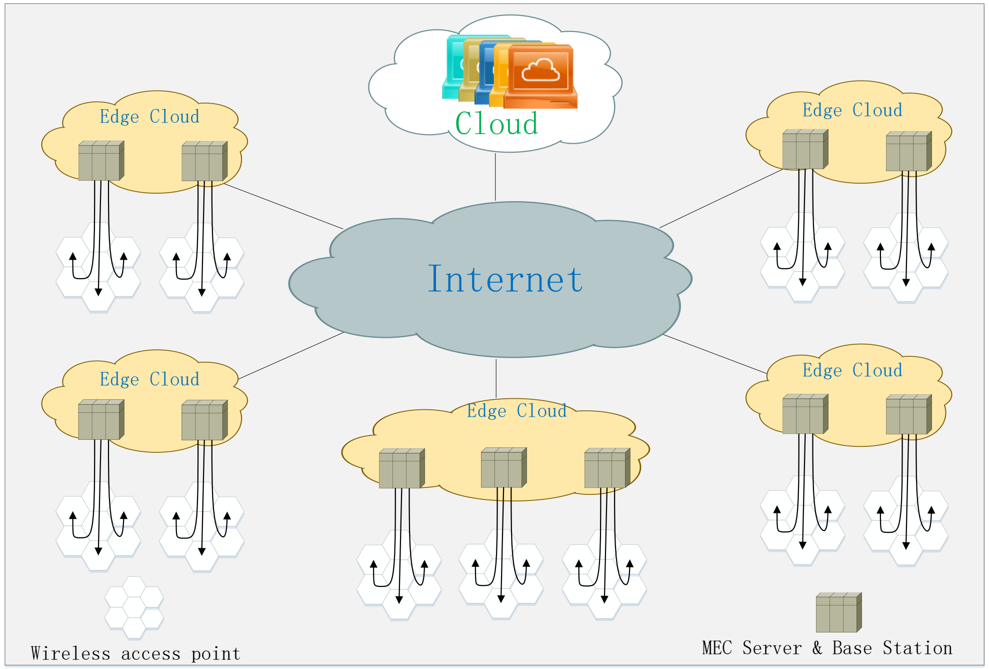
- Mobile Edge Computing(MEC)
  - Mobile Edge Computing provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers. The aim is to reduce latency, ensure highly efficient network operation and service delivery, and offer an improved user experience.[1]



[1] Hu, Yun Chao, et al. "Mobile edge computing—A key technology towards 5G." *ETSI White Paper* 11 (2015).

Introduction

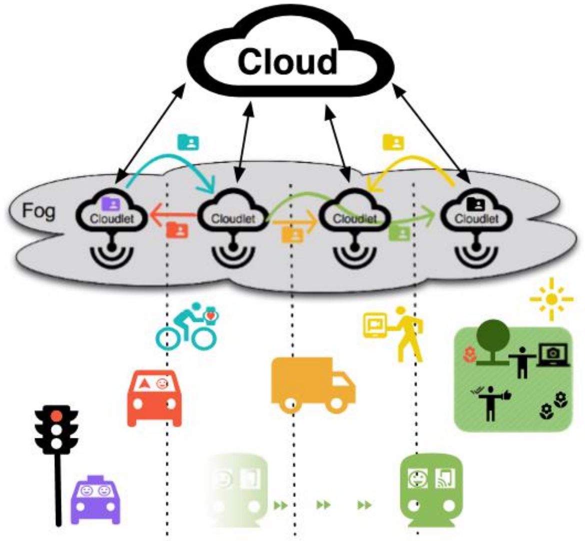
- MEC cloud and overall MEC system
  - Mobile operators are working on Mobile Edge Computing (MEC) in which the computing, storage and networking resources are integrated with the base stations.[2]



[2] Lav Gupta and Raj Jain, Mobile Edge Computing – An Important Ingredient of 5G Networks, IEEE Software Defined Networks

Existing work

- Mobility-driven service migration
  - Problem: mobility of user may increase the distance between user and its VM, and thus increase latency.
  - Solution: migrate user's VM across MEC clouds dynamically when user moves.



Wang, Shiqiang, et al. "Dynamic service migration in mobile edge-clouds." *IFIP Networking Conference (IFIP Networking), 2015*. IEEE, 2015.
Bittencourt, Luiz Fernando, et al. "Towards virtual machine migration in fog computing." *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on*. IEEE, 2015.

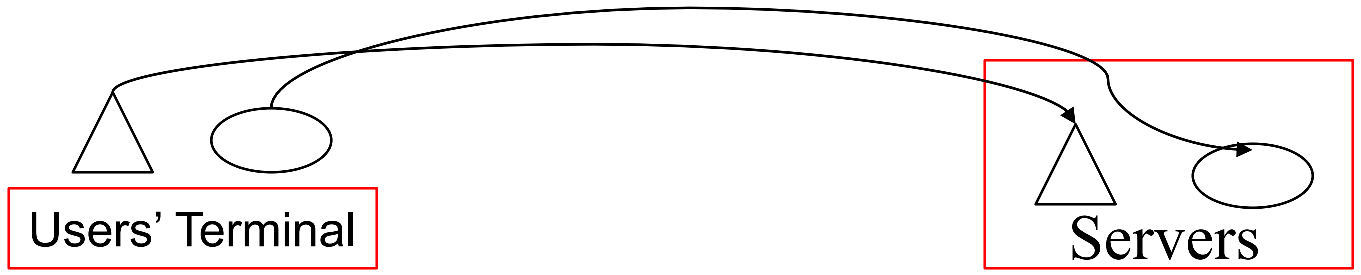
Existing work (cont.)

- SLA-driven VM Scheduling in Mobile Edge Computing
  - Problem: Service providers' cost of renting VMs at Edge clouds is calculated as $/time unit. Each type of VM has its maximum capacity to handle request. If the number of requests exceeds its capacity, some requests will go to remote clouds, and thus cause penalty for violating SLA. How to reduce cost while minimizing service penalty?
  - Approach: LYAPUNOV OPTIMIZATION-BASED scheduling algorithm for deploying and releasing VMs dynamically.

Katsalis, Kostas, et al. SLA-driven VM Scheduling in Mobile Edge Computing. 9th International Conference on Cloud Computing, IEEE, 2016

VM based service for MEC

- VMs' role in MEC



  - Virtual Machines (VM), which is composed of HW capacity with application specific software and data, play as servers.

- VM's influence on service Latency
  - Propagation Latency: VM's distance (L) from service source.
  - Processing + Queueing Latency: $T = \frac{1}{\mu - \lambda}$, in M/M/1 system, where μ is service rate, and λ is arrival rate. VM's # is the key factor of service rate.
  - E2E Latency = Propagation + Processing + Queueing Latency

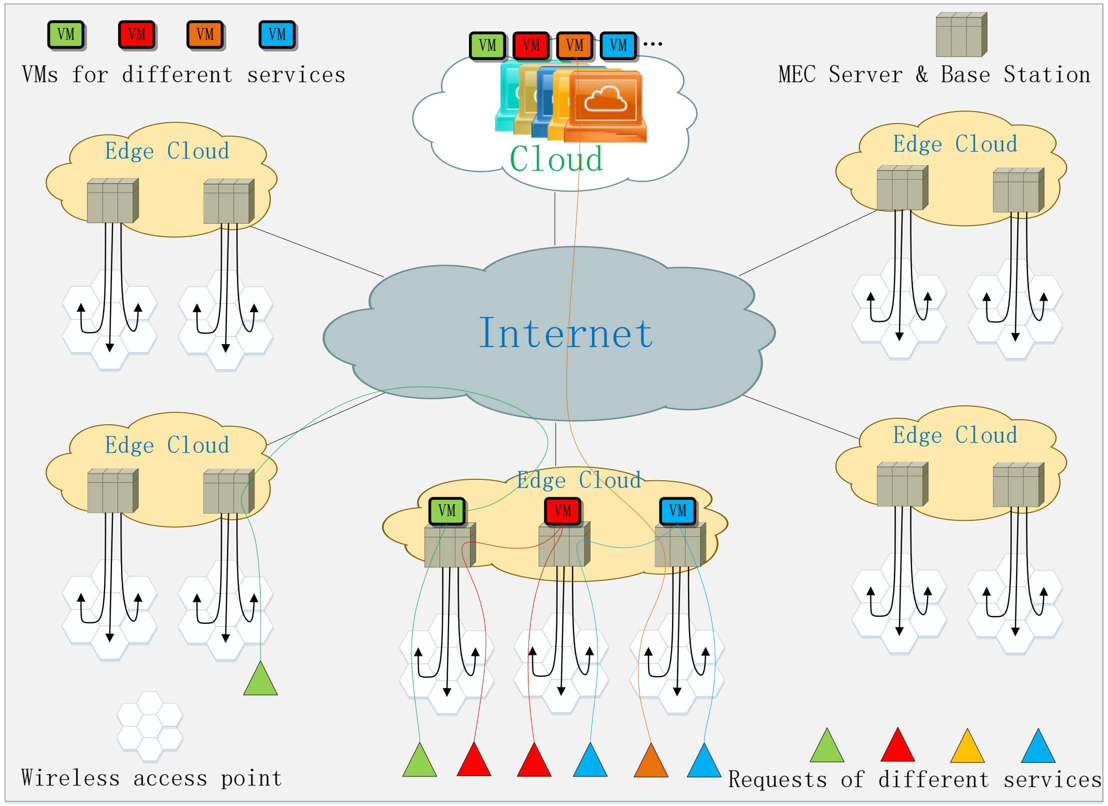
Traffic Patterns of MEC service

- Options for MEC service's dst:

(1)Local Edge DC

(2)Remote Edge DC

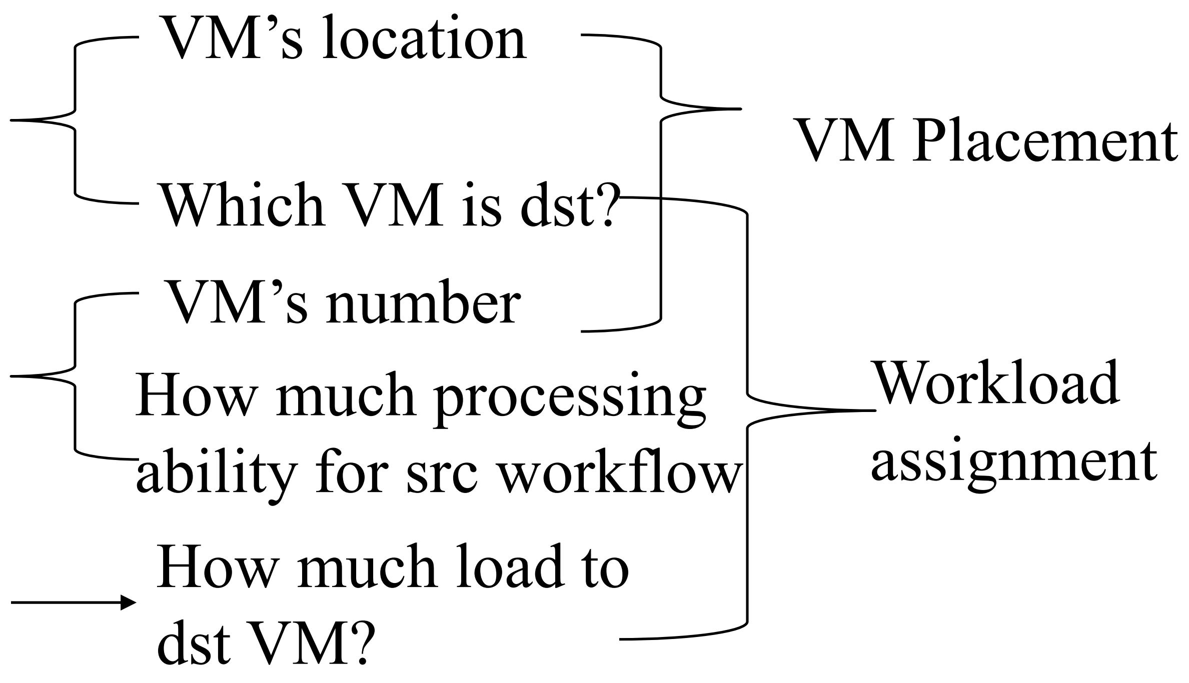
(3)Centralized Cloud


VM
VMs for different services
Cloud
MEC Server & Base Station
Edge Cloud
Internet
Wireless access point
Requests of different services

To meet latency requirement…

To meet latency of service flows from MEC nodes

- Distance to VM
- μ:service rate
- λ: arrival rate



Problem Statement

- Given:
  - E: Set of edge clouds
  - $L_{e1,e2}$: Propagation latency from $e1 \in E$ to $e2 \in E$
  - C_e: Hardware capacity of edge cloud $e \in E$
  - S: Set of services
  - R_s: Computing capacity for deploying a VM for service $s \in S$
  - T_s: Expected latency requirement of service $s \in S$
  - u_s: Service rate of one VM for corresponding service $s \in S$
  - λ_e^s: Request load of service $s \in S$ that originates from cloud $e \in E$
- Decide:
  - n_e^s: Integer, # of VMs for each service s at each mini DC e
  - $u_{src,dst}^s$, float, handling ability at dst for service s from src
  - $\lambda_{src,dst}^s$, float, request load for service s from src to dst
  - $x_{src,dst}^s$, binary, whether offload service s from src to dst

UCDAVIS
UNIVERSITY OF CALIFORNIA

VMs in one MEC DC

- (1) Required hardware resource for all VMs at each MEC $\mathrm{e} \in \mathrm{E}$ should not exceed DC's hardware capacity.

$$\sum_{\mathrm{e} \in \mathrm{S}} \mathrm{n}_e^s * \mathrm{R}_s < \mathrm{C}_e, \forall \mathrm{e} \in \mathrm{E}$$

Influence on latency:

When congestion occurs at one edge, some VMs will be placed at remote edges to handle requests from local edge, and thus introduce propagation latency.

Workflows to one DC

- All allocated handling ability for workflows to dst are the capacity of vms for service $s \in S$ at dst.

$$\sum_{\mathrm{src} \in \mathrm{E}} \mathrm{u}_{src,dst}^{s} = \mathrm{u}_s * \mathrm{n}_{dst}^{s}, \forall \mathrm{s} \in \mathrm{S}, \forall \mathrm{dst} \in \mathrm{E}$$

Influence on latency:

When allocate parts of handling ability from existing VM(s) for a new coming workflow, the processing + queueing latency of existing flows will increase.

$$T \uparrow = \frac{1}{\mu \downarrow - \lambda}$$

Workflows from one DC

- All workload needs to be assigned in terms of sub workflows.

$$\sum_{\mathrm{dst} \in \mathrm{E}} l_{src,dst}^{s} = \lambda_{src}^{s}, \forall src \in \mathrm{E}, \mathrm{s} \in \mathrm{S}$$

Influence on cost:

Parts of load can go to existing VMs, which has spare processing ability, and thus reduce the # of VMs need to be placed.

No violation of all service latency

- The estimated latency of each service s∈S should be lower than expected latency requirement

$$\frac{x^{s}_{src,dst}}{u^{s}_{src,dst} - \lambda^{s}_{src,dst}} \leq T_s - L_{src,dst}, \forall s \in S, \forall src \in E, \forall dst \in E$$

Influence on latency and cost:

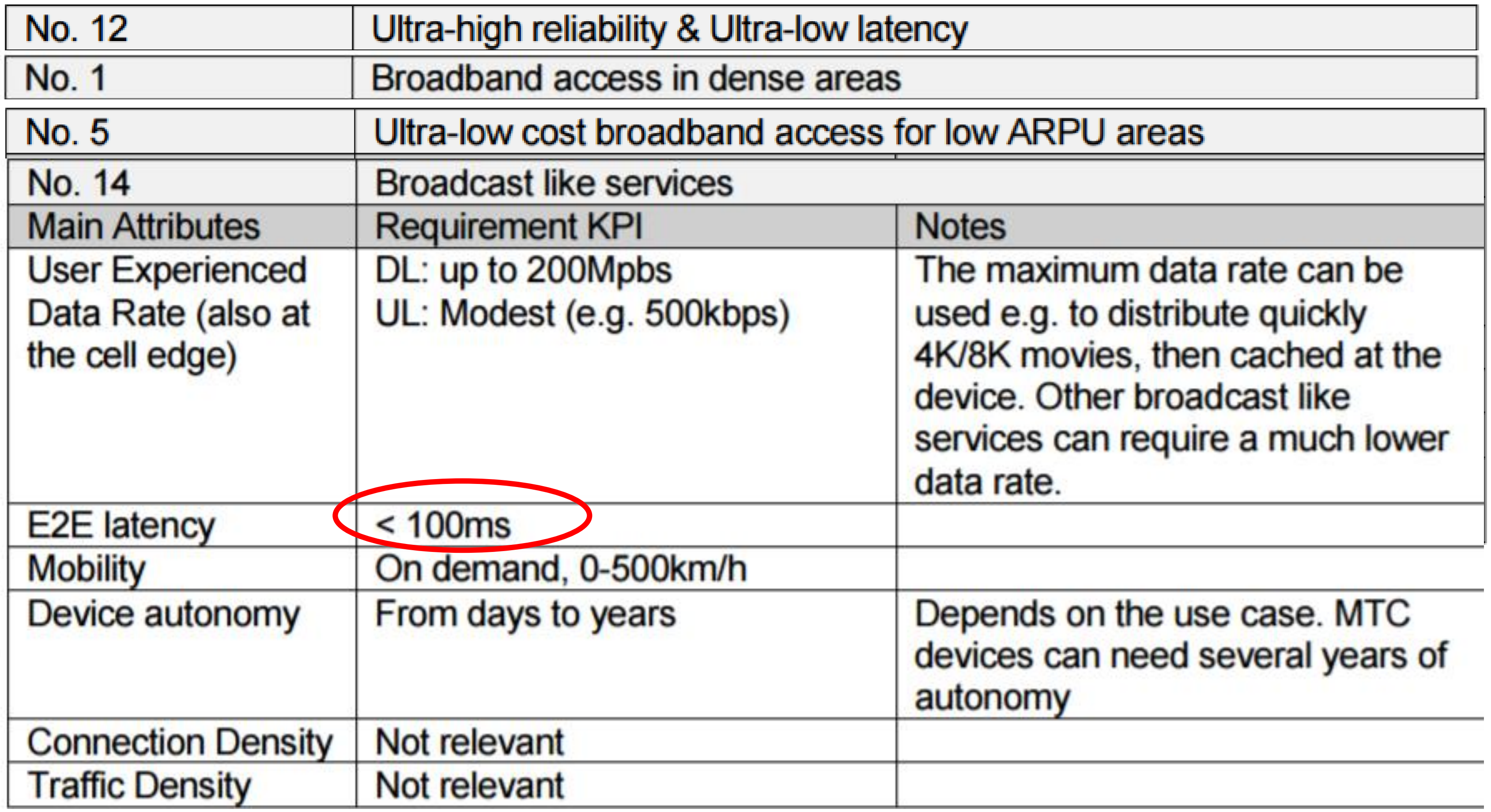
Remote offloading will introduce propagation latency.

Remote offloading need more handling ability to achieve same E2E latency, as compared with local processing.

Benefits and drawbacks of remote offloading

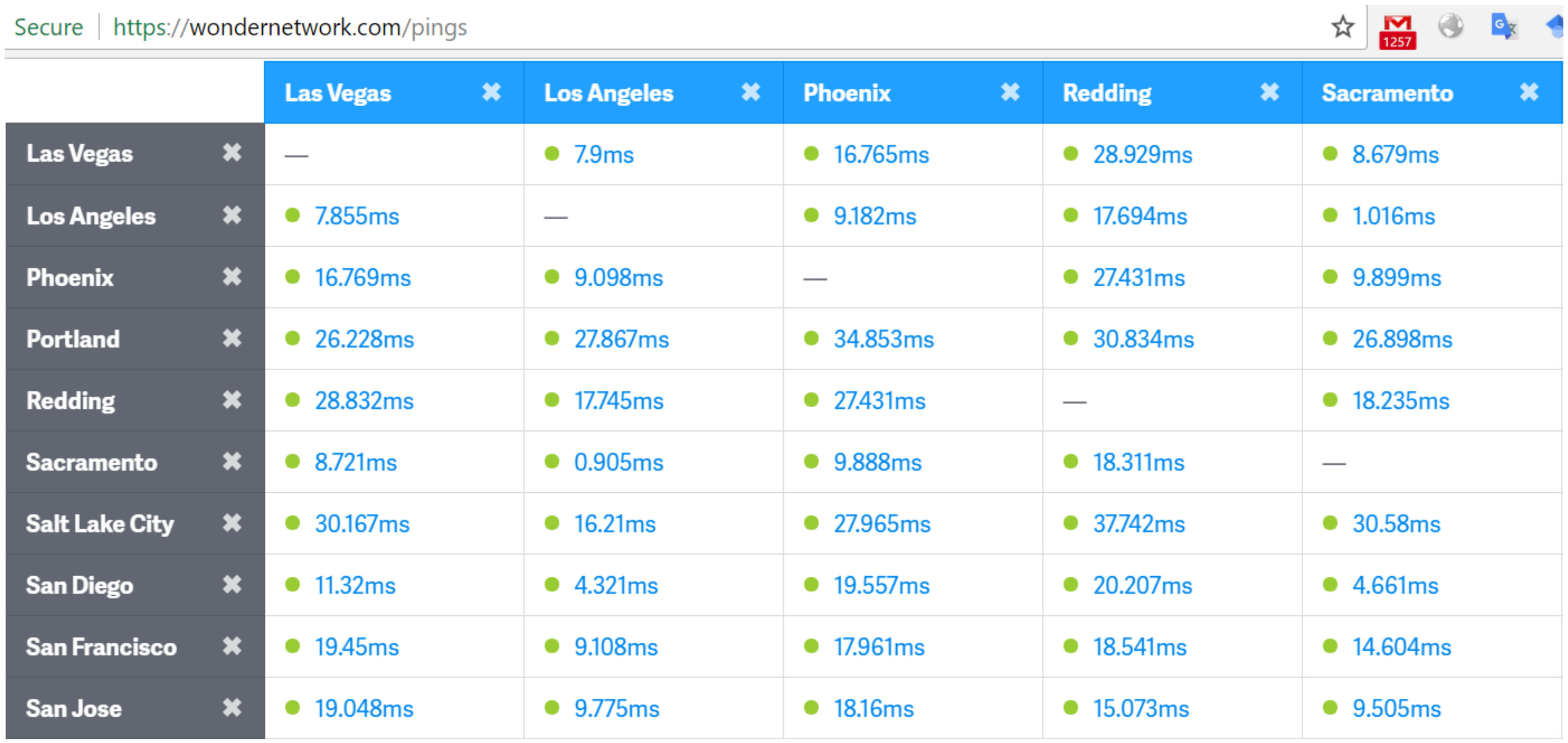
- Benefits:
  - (1) As a wrapper of software instance, VM's hardware requirement and handling capacity is not grid-less, and thus may introduce fragments when one VM is not fully used, remote offloading can utilize such fragments to save cost.
  - (2) MEC DC's capacity is limited, and may has computing resource congestion in certain cases. Remote offloading can use neighbor MEC DCs to guarantee SLA.
- Drawbacks:
  - (1) Remote offloading introduce propagation latency. To grantee E2E latency, and more processing ability will be used to reduce processing+queueing latency.
  - (2) Extra network traffic.

Latency Parameters



No. 12	Ultra-high reliability & Ultra-low latency	
No. 1	Broadband access in dense areas	
No. 5	Ultra-low cost broadband access for low ARPU areas	
No. 14	Broadcast like services	
Main Attributes	Requirement KPI	Notes
User Experienced Data Rate (also at the cell edge)	DL: up to 200Mpbs UL: Modest (e.g. 500kbps)	The maximum data rate can be used e.g. to distribute quickly 4K/8K movies, then cached at the device. Other broadcast like services can require a much lower data rate.
E2E latency	< 100ms	
Mobility	On demand, 0-500km/h	
Device autonomy	From days to years	Depends on the use case. MTC devices can need several years of autonomy
Connection Density	Not relevant	
Traffic Density	Not relevant	

Propagation Latency



Secure | https://wondernetwork.com/pings

	Las Vegas ✖	Los Angeles ✖	Phoenix ✖	Redding ✖	Sacramento ✖
Las Vegas ✖	—	● 7.9ms	● 16.765ms	● 28.929ms	● 8.679ms
Los Angeles ✖	● 7.855ms	—	● 9.182ms	● 17.694ms	● 1.016ms
Phoenix ✖	● 16.769ms	● 9.098ms	—	● 27.431ms	● 9.899ms
Portland ✖	● 26.228ms	● 27.867ms	● 34.853ms	● 30.834ms	● 26.898ms
Redding ✖	● 28.832ms	● 17.745ms	● 27.431ms	—	● 18.235ms
Sacramento ✖	● 8.721ms	● 0.905ms	● 9.888ms	● 18.311ms	—
Salt Lake City ✖	● 30.167ms	● 16.21ms	● 27.965ms	● 37.742ms	● 30.58ms
San Diego ✖	● 11.32ms	● 4.321ms	● 19.557ms	● 20.207ms	● 4.661ms
San Francisco ✖	● 19.45ms	● 9.108ms	● 17.961ms	● 18.541ms	● 14.604ms
San Jose ✖	● 19.048ms	● 9.775ms	● 18.16ms	● 15.073ms	● 9.505ms

UC DAVIS
UNIVERSITY OF CALIFORNIA

Future work

- Case A: Initial placement
  - Place VMs for multiple services on ALL empty MEC clouds.
- Case B: Incremental placement
  - Place new VMs for one or multiple services on MEC clouds, which already have other VMs.
- Case A and B can be covered by above formulations
- Case C: Dynamic VM management.
  - Heuristics algorithms for service load change.
  - Options: 1)VM clone & migration, 2)VM exchange, 3)Service map optimization.

Heuristic

- Sort services in increasing order of latency requirement
- For each kind of service, sort MEC nodes in increasing order of the service load originates from them
- For each node with each service, divide them into two parts, the first part of which is SuperBase, and the second part is SmallBase.

- Try to place VMs locally for each node in SuperBase, as follows.

$$T = \frac{1}{\mu * m - \lambda}$$

The number N of VMs would be the upper bound of float m

If N exceed HW capacity, the number of VMs would be the maximum # the DC can host, and some load(big orphan flow) cannot be served locally.

Heuristic

- Try to place VMs locally for each node in SmallBase, as follows.

$$T = \frac{1}{\mu * n - \lambda}$$

The number N of VMs would be the bottom bound of float m, and some load(small orphan flow) cannot be served locally.

- Map orphan load to nearest reachable SuperBase if possible.
- Assign big orphan flow to nearby MEC nodes. And generate small orphan flow.

Thank you!

Wei Wang

UCDAVIS
UNIVERSITY OF CALIFORNIA