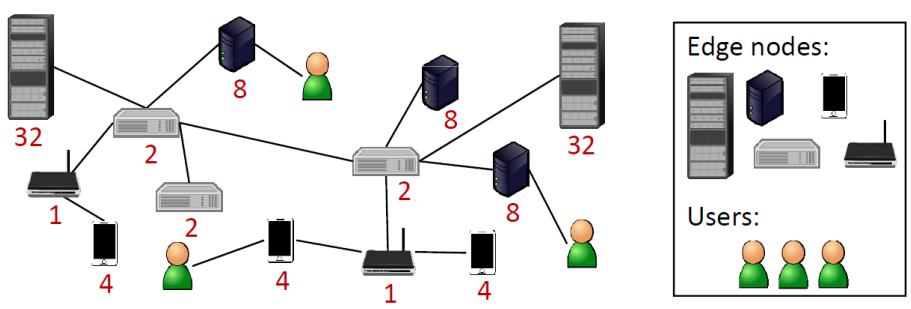
Service Placement with Provable Guarantees in Heterogeneous Edge Computing Systems

Stephen Pasteris (UCL), **Shiqiang Wang** (IBM US), **Mark Herbster** (UCL), **Ting He** (PSU)

Motivation

How to efficiently place services in a heterogeneous edge computing system for distributed analytics?



- Heterogeneous service and edge node sizes
- Heterogeneous rewards of serving users
  - Rewards can be related to coalition constraints and transmission latency

General Service Placement (GSP)

- We have a set *S*, of services: Each service *i* has **size** s_i
- We have a set *V*, of nodes: Each node *j* has **capacity** c_j
- We have a set *U* of users:
  - Each user *k* has a service ξ_k that it **requires**
  - Each user *k* has a map p_k from nodes into reals, $p_k(j)$ is the **reward** obtained from user *k* if the "closest" node (to *k*) that contains ξ_k is *j*
- Objective: place services on nodes so that:
  - Placement is feasible: i.e. the sum of the sizes of the services placed on a node is no more than the capacity of that node
  - The total reward from all users, subject to the above feasibility, is maximized

Service Placement with Set Constraints (SPSC)

- SPSC is a special case of GSP when, for all users *k* there exists a set of nodes W_k and a value r_k such that for all nodes *j*:
- If $j \in W_k$ then $p_k(j) = r_k$
- If $j \notin W_k$ then $p_k(j) = 0$
- We can convert any GSP with *n* users and to an SPSC with $n|V|$ users

➢ Both GSP and SPSC are NP-hard (proof is via reduction from set cover)

➢ We propose a **constant-factor approximation algorithm** by *leveraging the method of conditional expectations*

Linear Programming

To solve SPSC, we first solve the linear program:

$$\begin{aligned}
\max_{\{\omega_{i,j}\},\{\alpha_k\}} \quad & \hat{R} := \sum_{k\in U} \alpha_k r_k \\
\text{s.t.} \quad & \alpha_k \le \sum_{j\in W_k} \omega_{\xi_k,j}, && \forall k \in U \\
& \alpha_k \le 1, && \forall k \in U \\
& \sum_{i\in S} \omega_{i',j} s_{i'} \le c_j, && \forall j \in V \\
& \omega_{i,j} = 0, && \forall i \in S, j \in V : s_i > c_j \\
& 0 \le \omega_{i,j} \le 1, && \forall i \in S, j \in V
\end{aligned}$$

Slots

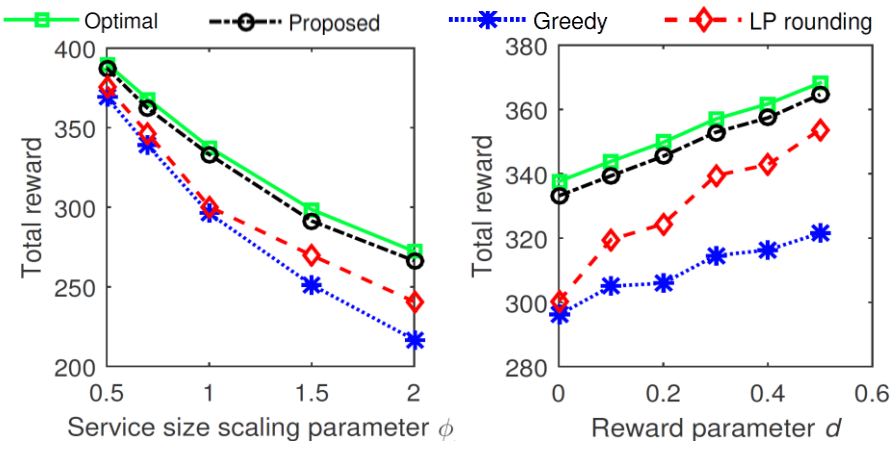
- Our first algorithm considers the case that the maximum size of a service is no more than β times the minimum capacity of a node (for some β<1); Define $\gamma = 1 - \sqrt{\beta}$ and $\delta = \gamma^2$
- For each node *j* and natural number *q* we define: $\Omega_{jq} := \{i \in S : \gamma^q c_j \beta < s_i \le \gamma^{q+1} c_j \beta\}$
- For every node *j* the above sets partition *S*
- A slot σ is a space on a node $\mu(\sigma)$ on which we can place a service in $\Omega_{\mu(\sigma)\lambda(\sigma)}$
- For every node *j* and natural number *q* we create $\lceil \delta \sum_{i\in\Omega_{jq}} \omega_{ij} \rceil$ slots σ with $\mu(\sigma) = j$ and $\lambda(\sigma) = q$

Allocating Services to Slots

- We define a probability distribution on allocations of services to slots as follows: for each slot σ independently, pick a service *i* from $\Omega_{\mu(\sigma)\lambda(\sigma)}$ with probability $\omega_{i,\mu(\sigma)}/(\sum_{l\in\Omega_{jq}} \omega_{lj})$ and assign service *i* to σ.

- We use the above probability distribution to guide our allocation of services to slots: for each slot σ in turn select the service from $\Omega_{\mu(\sigma)\lambda(\sigma)}$ that maximises (if assigned to σ) the expected total reward given the services assigned to slots so far.

- We prove an approximation ratio of $1 - e^{-(1-\sqrt{\beta})^2}$.
- Combining with our second algorithm (see paper for details), we have an algorithm that has an approximation ratio of $\max\left\{1 - e^{-(1-\sqrt{\beta})^2}; \frac{1-e^{-1}}{4}\right\}$.



Note: ϕ is proportional to the average size of each service.
Larger *d* makes the reward more heterogeneous.

Distributed Analytics and Information Science International Technology Alliance
Annual Fall Meeting, September 2017