# CS528

# Edge Computing/IoT /FoG

A Sahu

Dept of CSE, IIT Guwahati

# Outline

- Edge Computing
  – Latency Sensitive, Edge Servers
- IoT
  – Data Collection
  – Scheduling of Resources

# Edge Computing and IoT

# Edge Computing

- Edge computing
  - Brings computation and data storage closer to the sources of data
  - This is expected to improve response times and save bandwidth
- Edge computing is a topology- and location-sensitive form of distributed computing
- IoT uses the Edge Computing

# Edge Computing

- The origins lie
  - in **content distributed networks (CDN)**
  - that were created in the late 1990s
  - to serve web and video content from edge servers that were deployed close to user
- Example: Netflix, Utube
- CDN focus on Data not Processing
- Edge Computing focus on both Processing and Data

# IoT

- Describes physical objects (or groups objects)
  - with sensors, processing ability, software, and other technologies
  - that connect and exchange data with other devices and systems over the Internet or other communications networks
  - Do a whole one collaborative work
- Crowd souring Google map traffic, Building/Campus for energy efficiency

# IoT

- IoT can be used for Smart home, Smart Speaker, Elder Care
- Smart health care, Manufacturing, Transportation
- Agriculture, Environmental Monitoring
- Internet of Battle field of things, Ocean of Things

# IoT : Characteristics

- Source of Data : Sensors, Camera, Microphone, Monitors
- Sources may be mobile, ubiquitous, battery powered, all may not be directly connected to cloud
- Collection of Data and Storing of data
  - Network, Delay
  - How to send data to server for storing efficiently
- Processing of Data
  - Resource management
- Action based on Decision on Data Processing
- Many IoTs action : timely manner

# User App Data Distribution in Edge Computing

**Cost-Effective App Data Distribution in Edge Computing, IEEE Trans Parallel Dist. System, Jan 21**

# Edge Data Distribution: Intro

- Cloud computing is the practice of
  - using a network of remote servers hosted on the internet
  - to store, manage, and process data,
  - rather than a local server or a personal computer.
- Latency influenced by
  - the number of router hops, packet delays
  - introduced by virtualization in the network,
  - or the server placement within a data center,
  - has always been a key issue for cloud migration.
- Other conventional network paradigms
  - cannot handle the huge increase in the network latency and congestion
  - caused by the resources at the edge of the cloud.

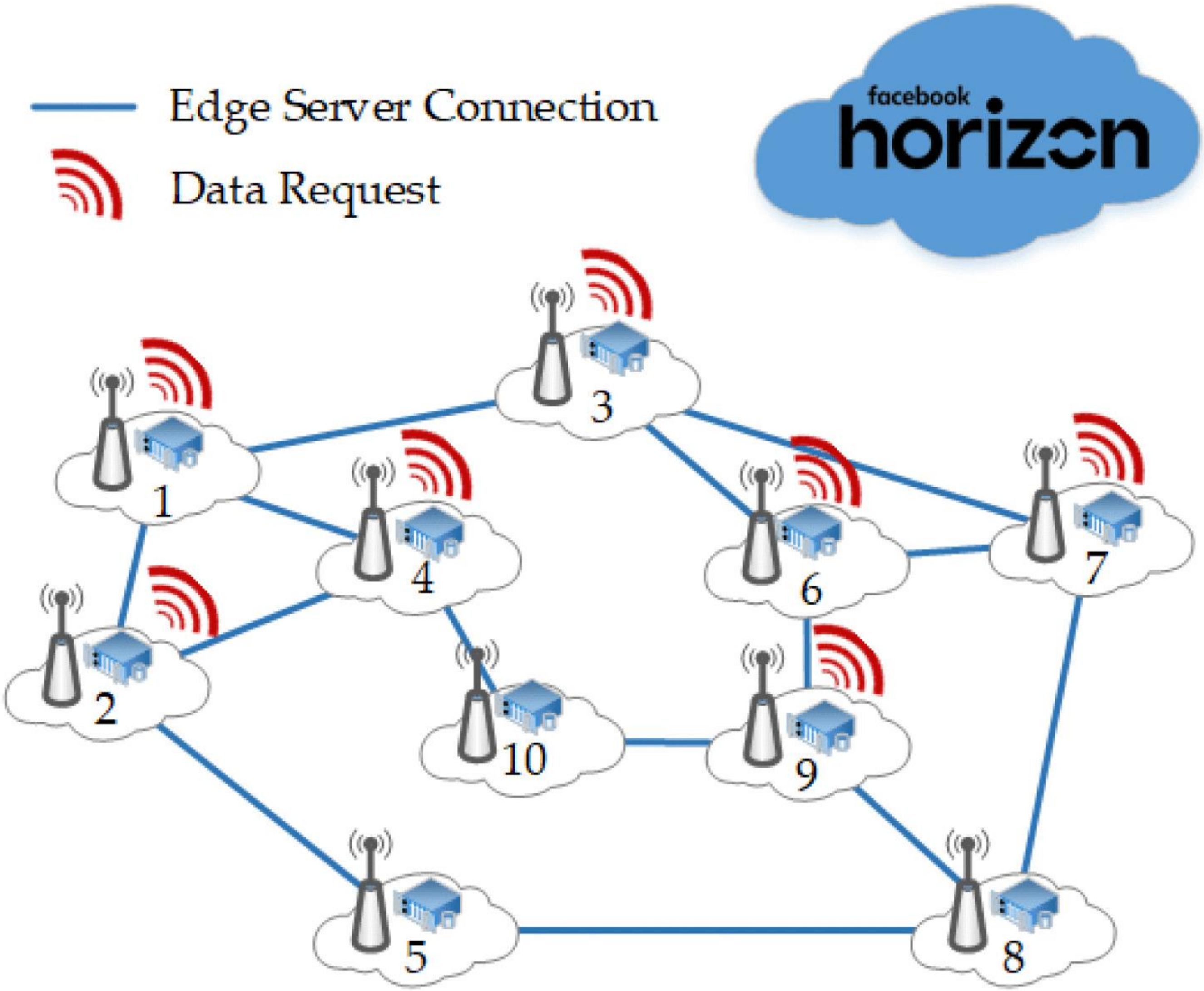
# Edge Data Distribution: Intro

- edge computing
  - which is essentially the process of decentralizing the computer services
  - with the help of edge servers deployed at the base stations.
- From an app vendor's perspective,
  - caching data on edge servers
  - considerably reduce both the latency for their users to fetch data and the volume of their app data transmitted between the cloud and its users.
  - Thus, reducing the transmission costs.

# An Industry Example: Facebook Horizon

- Facebook Horizon: VR application
- Facebook Horizon can benefit greatly
  - from distributing most popular VR videos and VR games onto the edge servers.
- VR applications are very latency sensitive
  - thus caching these data onto edge servers
  - will increase VR performance, experience and sensitivity.
- Cost-Ineffective app data distribution
  - Can thereby cost Facebook Horizon significantly more.

# An Industry Example: Facebook Horizon



# Problem Definition: EDD

- $N$ edge servers in a particular area and model as graph $G$.
  - For each edge server $v$, graph $G$ has a node $v$.
  - For each of the linked edge servers $(u,v)$, graph $G$ has a corresponding edge $e_{(u, v)}$.
- $G(V,E,W)$ to represent the graph
  - where $V$ is the set of nodes or edge servers in the graph,
  - $E$ are the set of edges in the graph, and
  - $W$ is the set of weights corresponding to the edges.
- Let $R$ denote the set of destination edge servers in graph $G$.

# Problem Definition: EDD

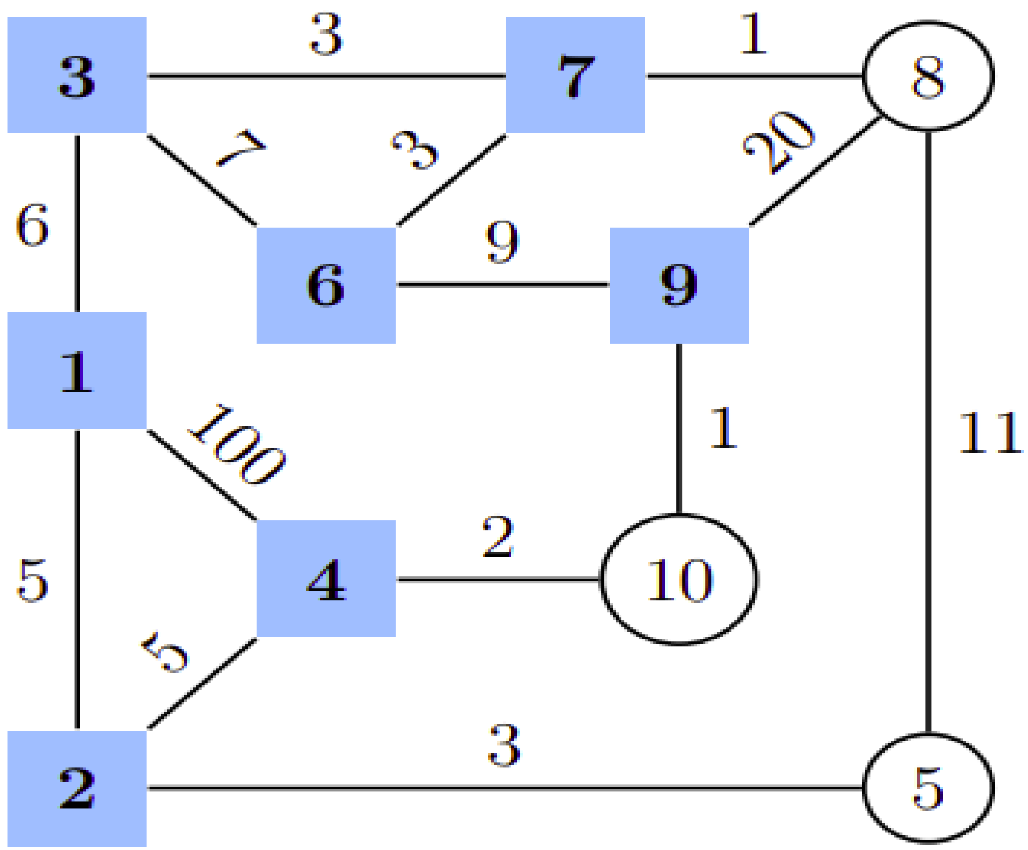
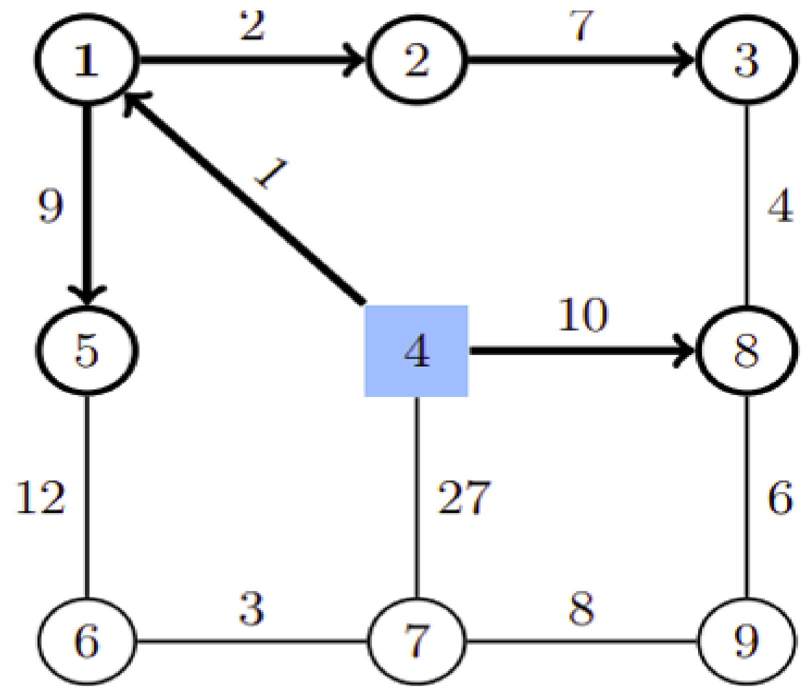



**Fig. 4.1:** EDD scenario with 10 edge servers

# Problem Definition : EDD

- Let $L_{limit}$ be the vendors' EDD length constraint.
- Two Possible Scenario – C2E and E2E.
- We define a ratio λ to specify the constant weight for the cloud to edge server edges.




EDD example to demonstrate $L_{limit}$

# Problem Definition: EDD

$$s_{\mathrm{v}} = \begin{cases} 1, & \text{if } v \text{ is an initial transit edge server} \\ 0, & \text{if } v \text{ is not an initial transit edge server} \end{cases} \tag{1}$$

$$\tau_{(u,\ v)} = \begin{cases} 1, & \text{if data is transmitted through } \mathrm{e}_{(\mathrm{u},\ \mathrm{v})} \\ 0, & \text{if data is not transmitted through } \mathrm{e}_{(\mathrm{u},\ \mathrm{v})} \end{cases} \tag{2}$$

$$Connected(S,\ T,\ u,\ v) = true,\ \forall v \in R,\ \exists\, u, s_{\mathrm{u}} = 1 \tag{3}$$

$$P_{\mathrm{delay}} = \frac{L_{\mathrm{link}}}{s_{\mathrm{medium}}} \tag{4}$$

$$W_{(\mathrm{c},\ \mathrm{v})} = \lambda, \forall\, v \in V \setminus \{c\} \tag{5}$$

$$0 \leq L_v \leq L_{limit}, L_v \in Z^+, \forall v \in R \tag{6}$$

$$minimize\left(Cost_{\mathrm{C2E}}(S) + Cost_{\mathrm{E2E}}(T)\right) \tag{7}$$

# Edge Computing: Compute Sharing

**Cost-Effective App User Allocation in an Edge Computing Environment, IEEE Transaction Cloud Computing, 2022 (Early Access)**

# Edge Computing: Compute Sharing

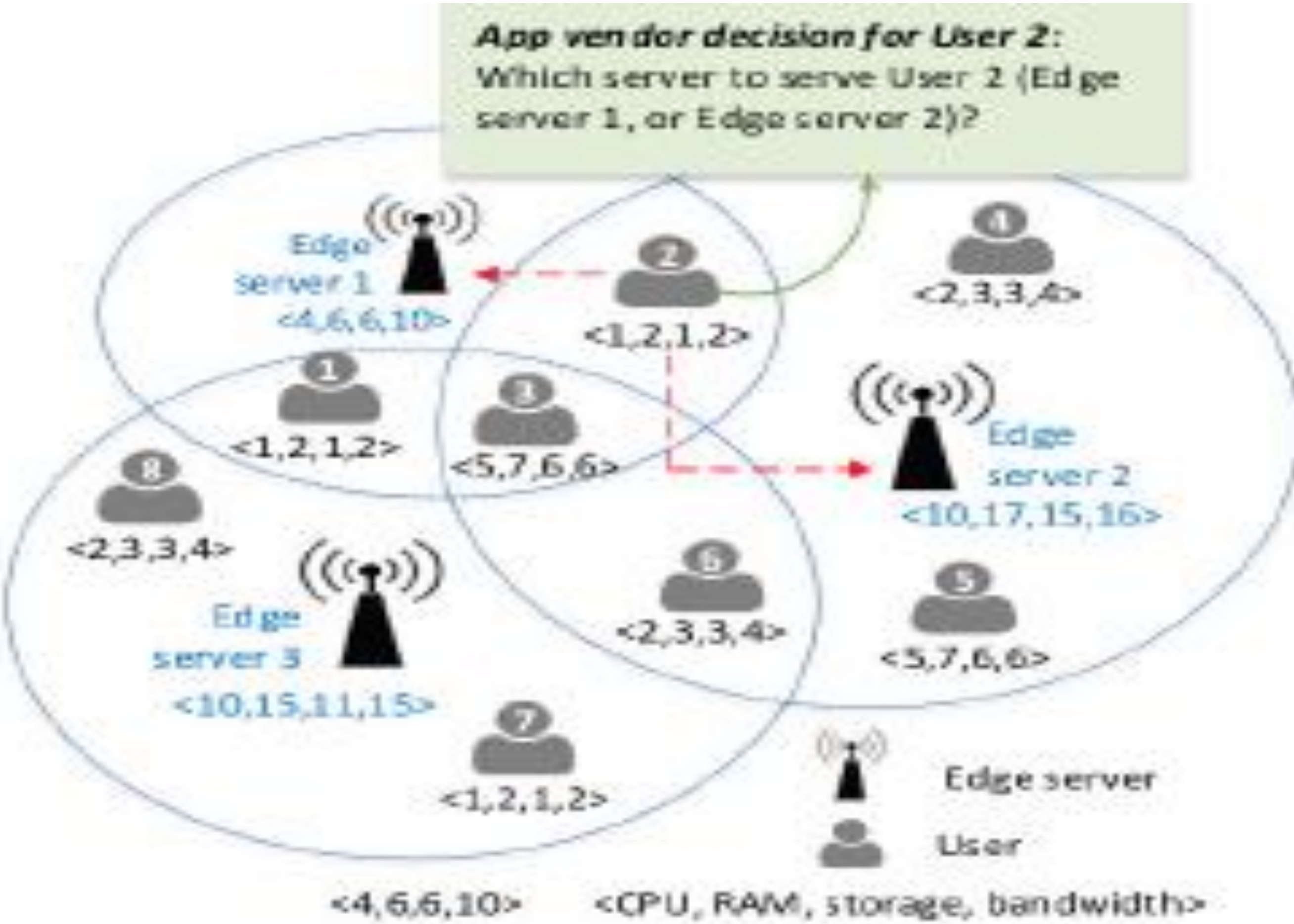



Fig. 1. The number of users allocated vs the number of servers

# Problem Statement

- $S=\{s_1,s_2,.., s_m\}$: set of servers
- D={cpu, ram,disk, bw}: set of resource type
- $C_i=\{C_i^1,C_i^2,..C_i^d\}$: Capacity vector of server Ci
- $U=\{u_1,u_2,...u_n\}$:Finite set of users, $u_i$ is ith user
- Set of user allocated to $s_i$ : $U_{si}$ is subset of U
- Set of user covered by $s_i$: $Cov(s_i)$ is subset of U
- $w_j=\{w_j^1,w_j^2,...w_j^d\}$: resource use vector of user j

# Problem Statement: Edge User Allocation

- For each server total capacity of the server must be greater than the resource requirement of the user allocated on the server

$$\sum_{u_j \in U_{s_i}} w_j \leqslant C_i, \forall s_i \in S$$

- Primary Objective: Maximize numbers of user allocation to the edge servers

$$maximize \sum_{s_i \in S} |U_{s_i}|$$

- Secondary Objective: Minimize the number of edge servers

$$minimize \Big|\{s_i \in S \,|\, \sum_{u_j \in U_{s_i}} w_j > 0\}\Big|$$

# FoG Computing

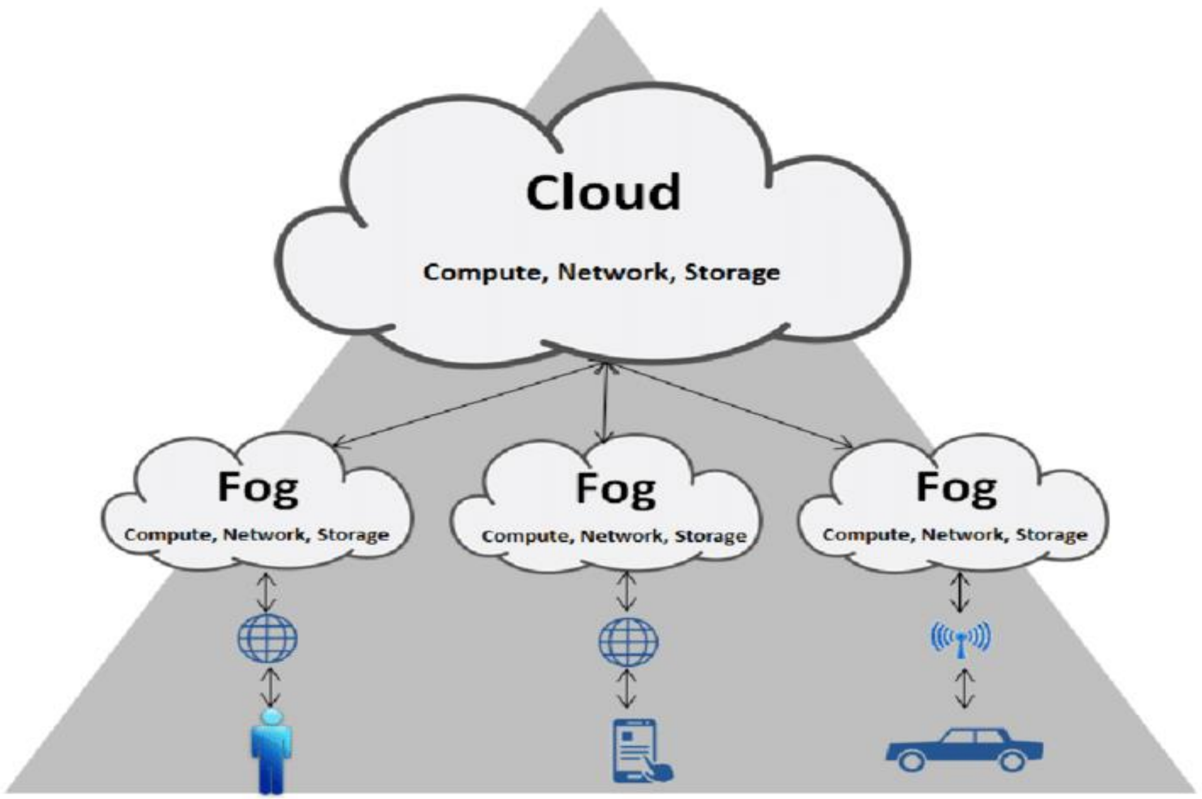
Efficient Welfare Maximization in Fog-Edge Computing Environment, HPCC 2021

# FoG Computing

- *Fog computing* is an extension of cloud computing. It is a layer (multiple layers) in between the edge *and* the cloud.
- Sharing maybe greatly defined at FoG level
  - Economy and Energy point of view

# FoG Computing: FoG Network

- A good task offloading framework should have the capability to achieve optimal energy conservation when it comes to the execution of tasks.
- Meanwhile, an incentive mechanism has to be ensured in collaboration withnodes.



# FoG Network Sharing

- FOG network helps to accomplish tasks on nearby nodes
  - if some node  doesn’t have enough computing power to process the task.
- If a task is being executed by a helper node then
  - the helper node must get higher share of task revenue and
  - the task node which has offloaded the task to the helper/nearby node, should get very less share of task revenue

# System Model

- Homogeneous FOG network environment primarily consisting of M nodes
  - V={ $V_1$, $V_2$, …$V_M$} where all M nodes share computing resources with each other in a collaborative way.
- Let T = { $t_1$, $t_2$, $t_3$, …, $t_N$ } be set of N tasks where each task $t_i$ has attributes associated with it such as
  - task originating node($O_i$),
  - deadline ($d_i$), task compute size ($l^c_i$ ) in MI,
  - task data i/o size ($l^s_i$ ) in KB
  - and revenue associated with that task ($r_i$).
- So the task : in tuple form $t_i(l^c_i , l^s_i\ d_i, O_i, r_i)$
  - The value is *Oi* is from 1 to M, this specify the where the task is originated or created.

# System Model

- Bandwidth between the FoG node $V_i$ and $V_j$ is proportional to $1/D_{ij}$ , where $D_{ij}$ is Euclidean distance $V_i$ and $V_j$
- Every FOG node $V_j$ is characterized
  - number of cores $c_j$ (consisting of same computing capacity) ,
  - processing energy consumption rate per core $ecr^{proc}{}_j$ ,
  - communication energy consumption rate $ecr^{comm}{}_j$ ,
  - computing capacity per core in MIPS $cp_j$,
  - energy cost per unit energy consumption $\rho_j$.

# System Model

- Effective time to the execute task $t_i$ at originated node $V_j$

$$ET_{i,j} = l^c{}_i / cp_j$$

- Effective time to the execute task $t_i$ on helper node $V_j$

$$ET_{i,j} = l^c{}_i / cp_j + l^s{}_i / bw_{i,j}$$

- Cost of execution of task $t_i$, originated at FoG node $V_j$

$$Cost_{i,j} = (l^c{}_i / cp_j) * ecr^{proc}{}_j * \rho_j$$

# System Model

- Cost of execution of task $t_i$, at helper node $V_j$

$$Cost_{i,j} = [(I^c_i/cp_j)*ecr^{proc}_j+(I^s_i/bw_{O(i),j})*ecr^{comm}_j]*\rho_j$$

- If a task *ti* is executed at the originated node $V_j$ then Revenue $R_{i,j}$ will be $R_{i,j} = r_i$.
- If a task $t_i$ originated at $V_k$ is executed at the helper node *Vj* then
  – Revenue at helper node *Vj* can be given as $R_{i,j} = r_i * \lambda$ .
  – Revenue at Originated node $V_k$

  $R_{i,k} = r_i\ (1- \lambda)$ where value of is 0.9.

# System Model

- Executing the task $t_i$ originated at node $V_k$ on helper node $V_j$.
  - $U^y{}_{ij}$ and $U^y{}_{ik}$ is profit at node k and node j respectively.

    $U^y{}_{ik} = R_{i,k}$

    $U^y{}_{ij} = R_{i,j}\text{-}Cost_{i,j}$
- Profit of node $V_j$=$O_i$ of executing the task $t_i$ on node $V_j$ in case of task is executed in originated node

  $\boldsymbol{U^x{}_{ij} = r_i - Cost_{i,j}}$

# Maximization: Profit and Welfare

- Maximize

$$TP = \sum_{j=1}^{m} profit_j$$

where j represents a node.

- Minimize

$$DP = \arg\max\{profit_j\} - \arg\min\{profit_j\}$$

where disparity DP in the profit can be calculated as difference between the  maximum profit among all the nodes and the minimum profit among all thenodes

# Heuristics Solutions

- Global pool-based : The global scheduler uses
  - information like distance matrix  between nodes, data size of tasks, compute size and revenue of tasks
  - to schedule tasks to approximate nodes to optimize both *TP* maximization and minimization of *DP*.
- Mixed Pool:
  - FCFS-based : The task selection criteria for the local queue from the locally originated task is based on FCFS basis.
  - Revenue-based : Here the top  percentage of the highest revenue-generating tasks get reserved for the local execution and the rest of the tasks are added to the global pool.