Computing with Nearby Mobile Devices: a Work Sharing Algorithm for Mobile Edge-Clouds

In this paper author is describing concept to share high computation work between nearby mobile devices to reduce execution time and energy consumption. Now a days all mobiles are using mobile cloud to offload heavy computation task but it requires high connectivity to mobile cloud and to reduce this connectivity burden author is saying to utilize nearby mobile devices by applying work stealing algorithm and this algorithm consists of two parts called Delegator and Workers.

Delegator: This is a source mobile which finds nearby mobile devices and then offload task to nearby mobile devices. All tasks will be maintained in queue

Workers: Workers steals task from queue and then process and send result back to source mobile but while implementation we need to concentrate on worker node heterogeneous and mobility where heterogeneous refers to different working capacity of mobiles and mobility refers to movement of mobile as nearby mobiles will not be static and they keep on moving and in such situation we can used work stealing algorithm to offload task to nearby mobiles. Work stealing algorithm perform below steps to implement propose paper Honeybee architecture.

1. In work stealing algorithm source mobile first put all tasks inside queue and publish/send queue to nearby devices
2. Nearby devices obtained task from queue and then process and send result back to client and if nearby devices moves out of source range then it will not send heartbeat message.
3. If work stealing algorithm did not received any heartbeat message from nearby device then it will add same task to the end of the queue and wait to find new nearby devices
4. If new nearby devices found then work stealing algorithm share task with new mobiles and get the result back.
5. Algorithm repeats above 4 steps till all task executes successfully. High processing mobiles can complete task faster so energy and execution time will be reduced. High processing nodes obtained task from low processing nodes if they not produce response faster.

By sharing task between high processing mobiles execution time and energy consumption can be reduce. To implement this project author has designed 3 applications called Worker, mobile crowd (Honeybee) and android delegator to publish task via WIFY.

Here we cannot used android mobile devices as it require WIFY everywhere to delegate task, so we have designed python simulation based devices to delegate or publish task.

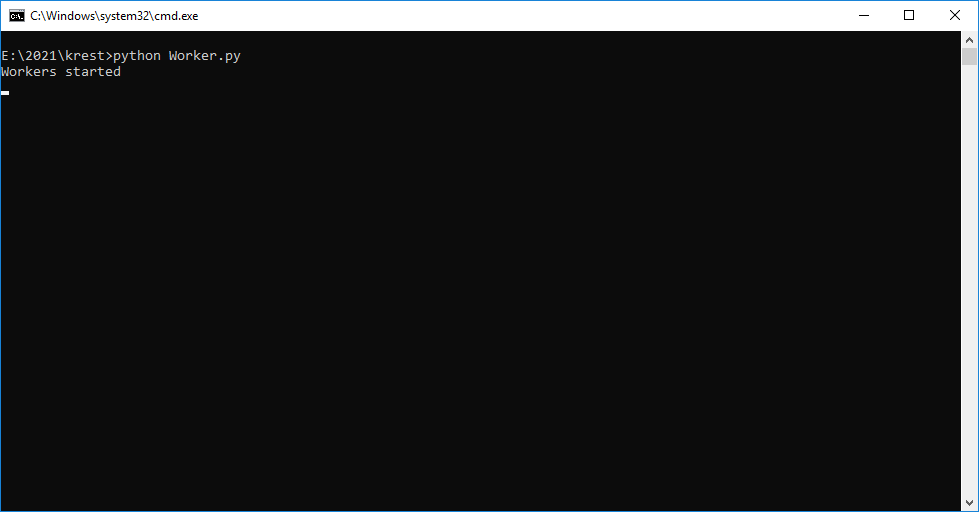
Mobile crowd (Honeybee) application designed to find nearby devices, schedule task in queue and work stealing algorithm

Worker application designed to execute task.

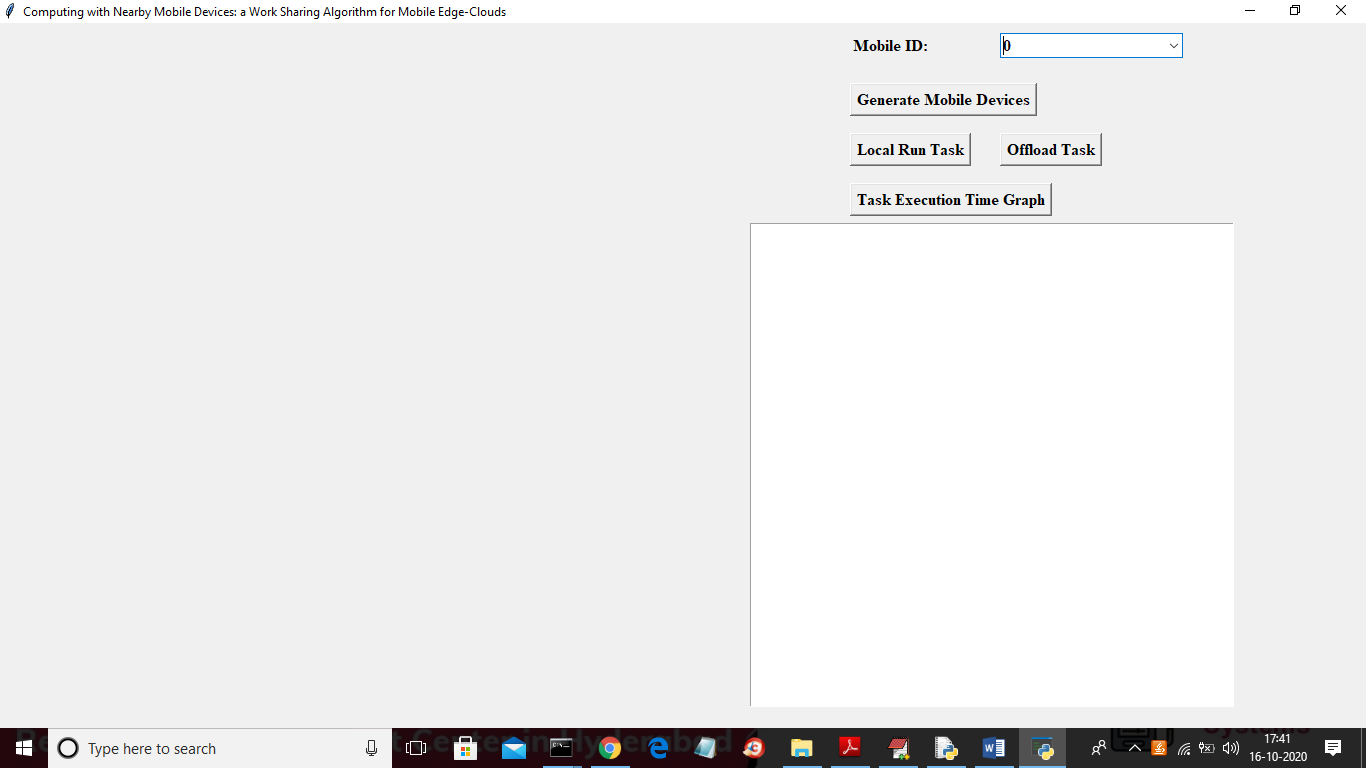
To implement above concept we are using face detection from images as the task for the workers. Source mobile will upload images and then send to worker to find faces from images and send result back to source mobile.

SCREENSHOTS

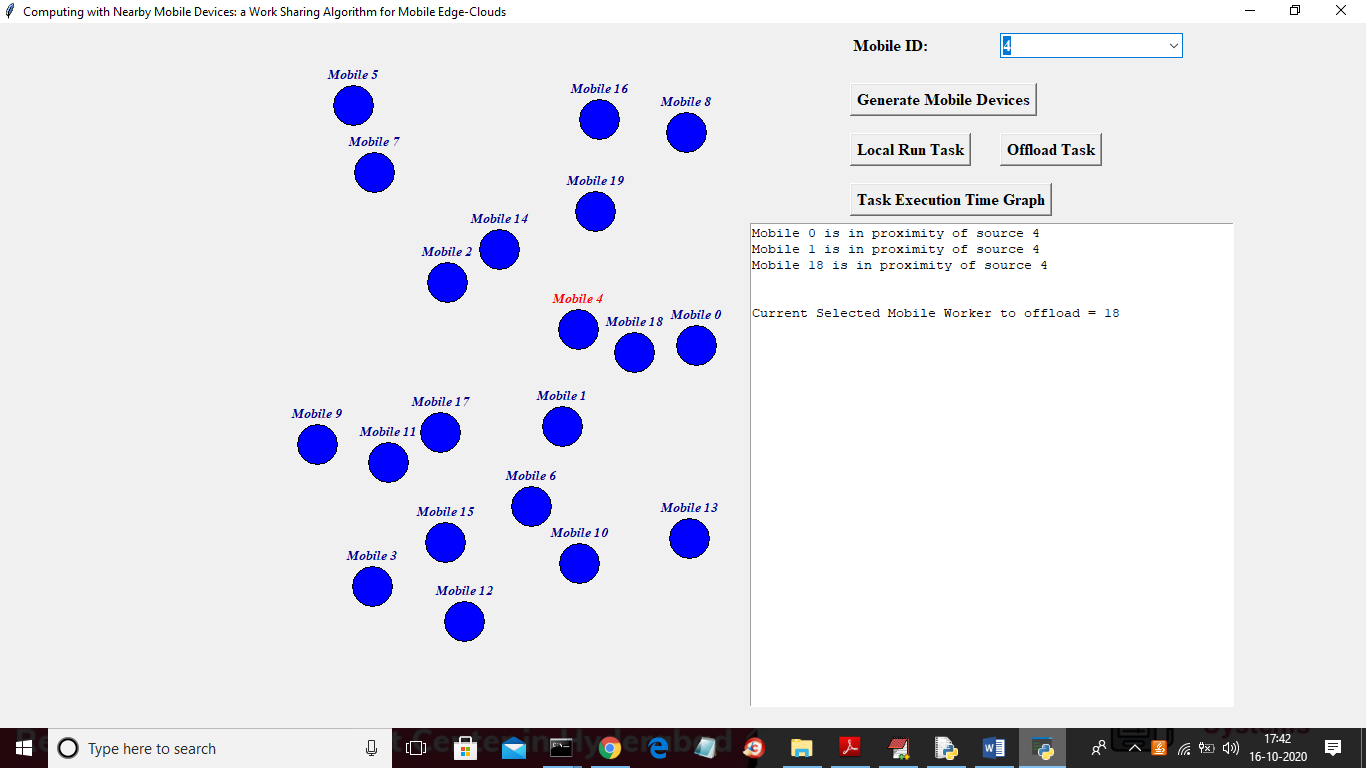
Double click on ‘run\_worker.bat’ file to start workers application and to get below screen



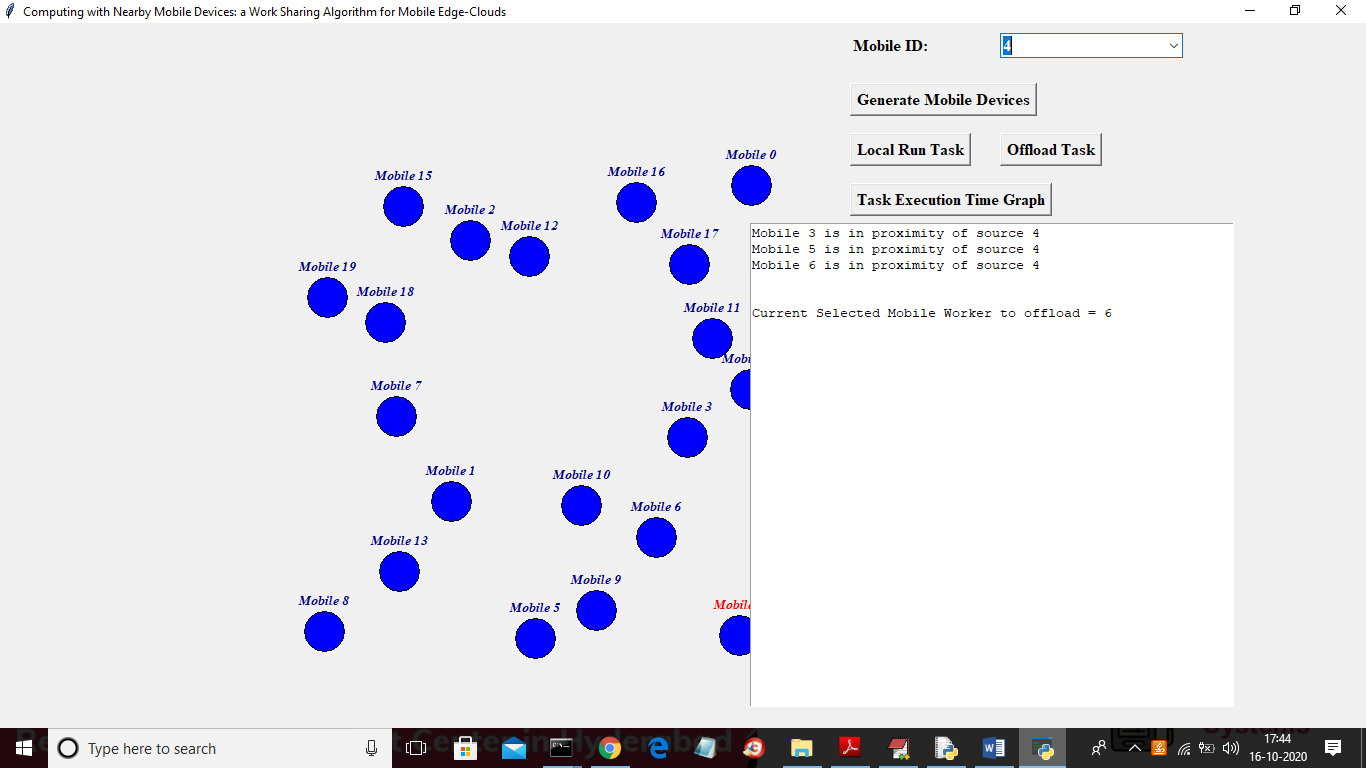
In above screen workers are started and now double click on ‘run.bat’ file to start mobile simulation and to get below screen



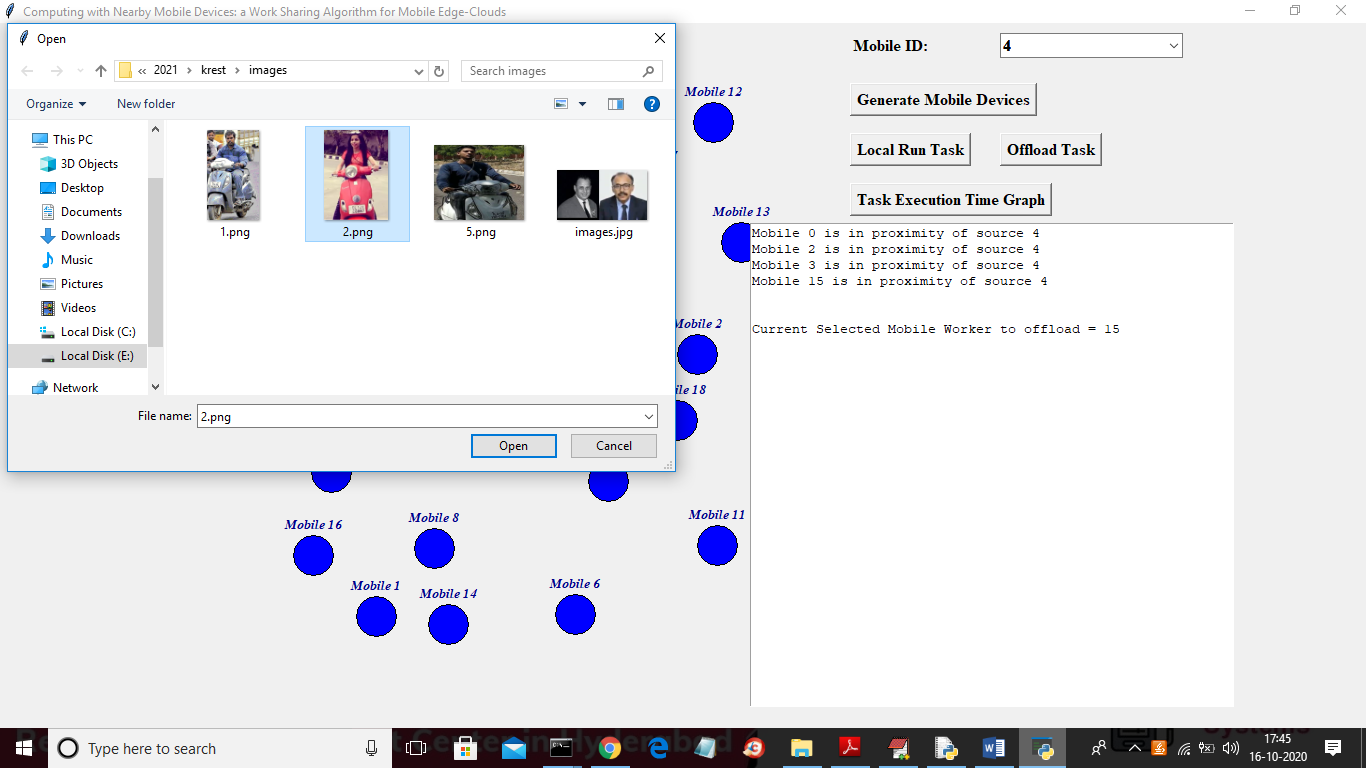
In above screen select any mobile id from drop down box and then click on ‘Generate Mobile Devices’ button to get below screen



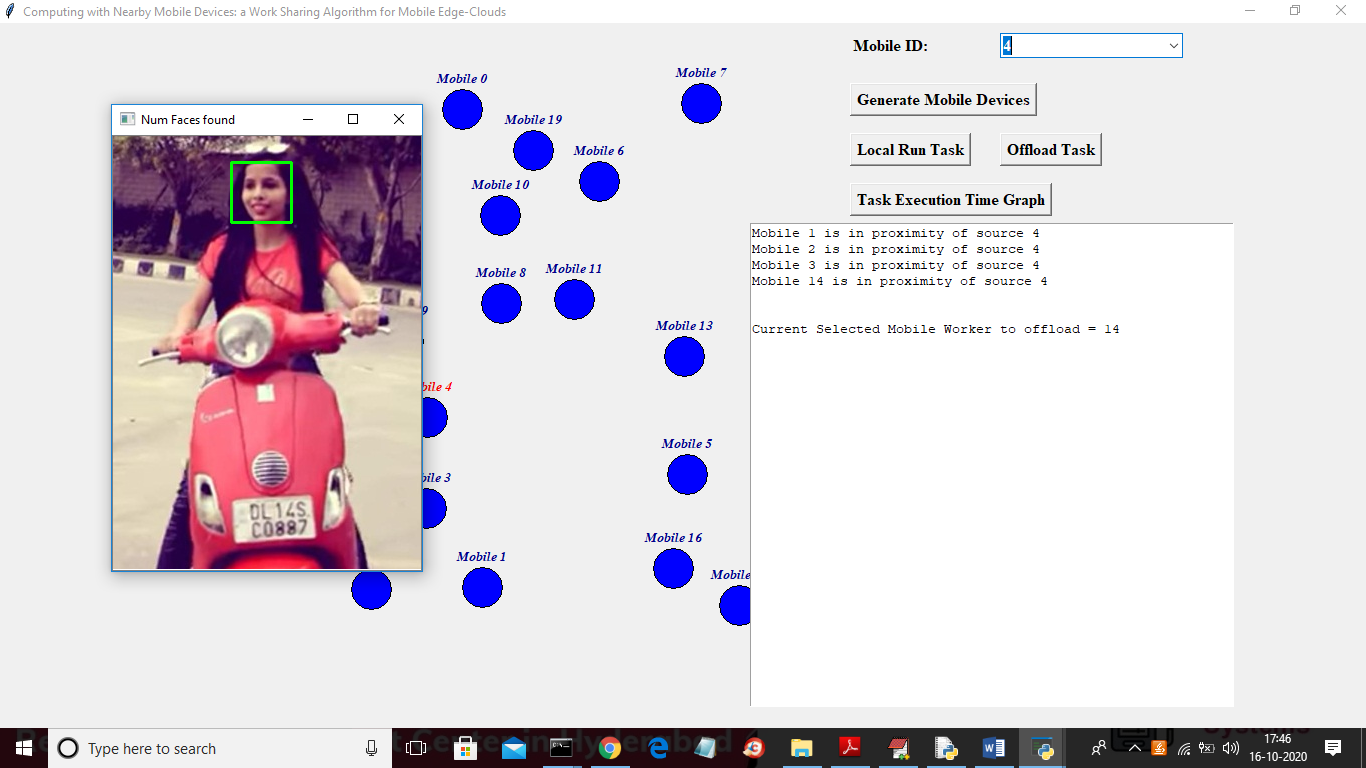
In above screen I selected 4 as the source mobile id from drop down box and in right side we can see all mobiles in the form of circles and source mobile is in red colour and in text area we can see all then nearby devices to source and the mobile keeps on moving and source nearby devices will also change.



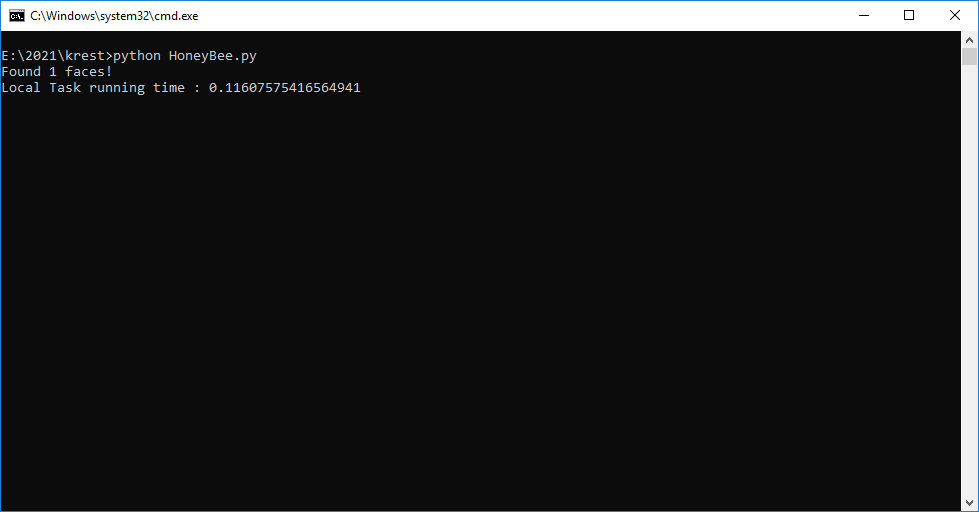
In above screen we can see now the nearby devices changed compare to last screen and now click on ‘Local Run Task’ button to upload image and then run task in local machine and capture execution time



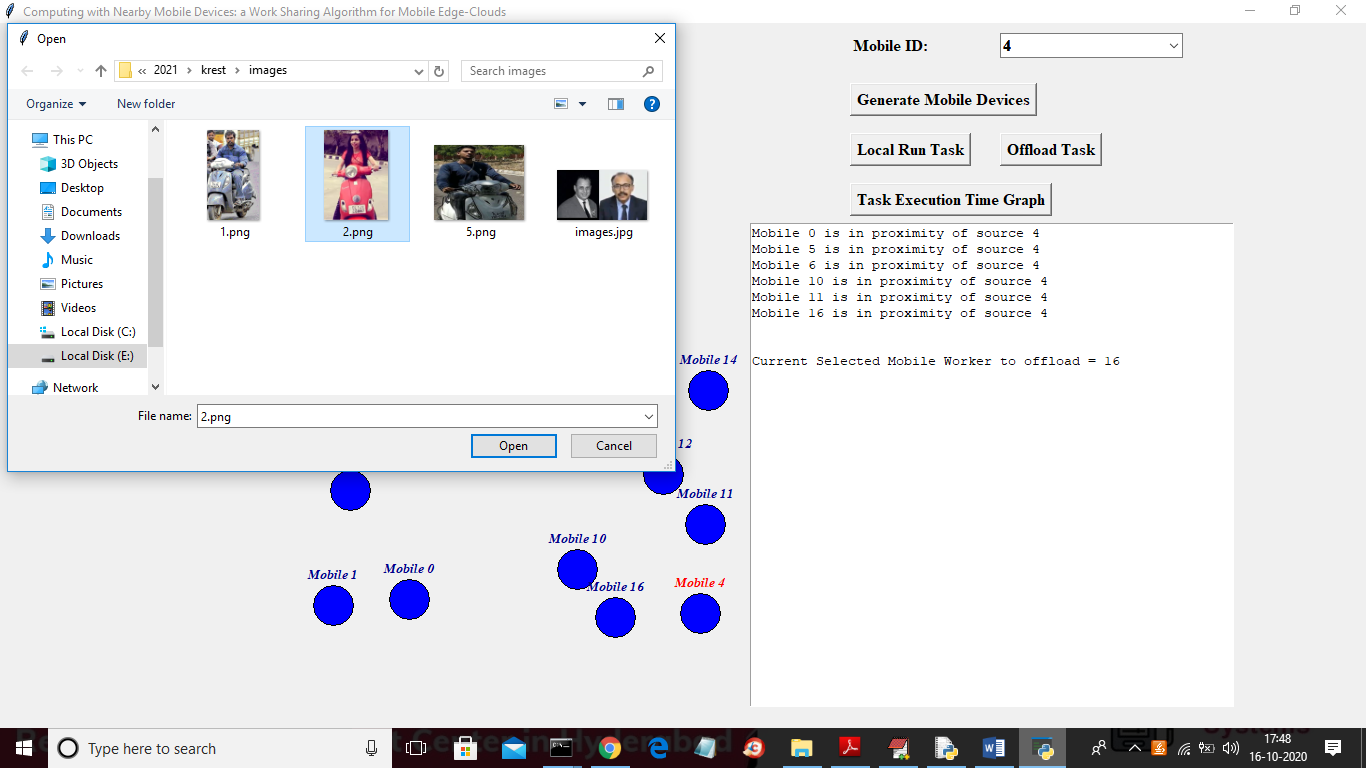
In above screen I am uploading 2.png image and then run that task in local machine without sending to workers and then capture execution time of face detection by clicking on ‘open’ button



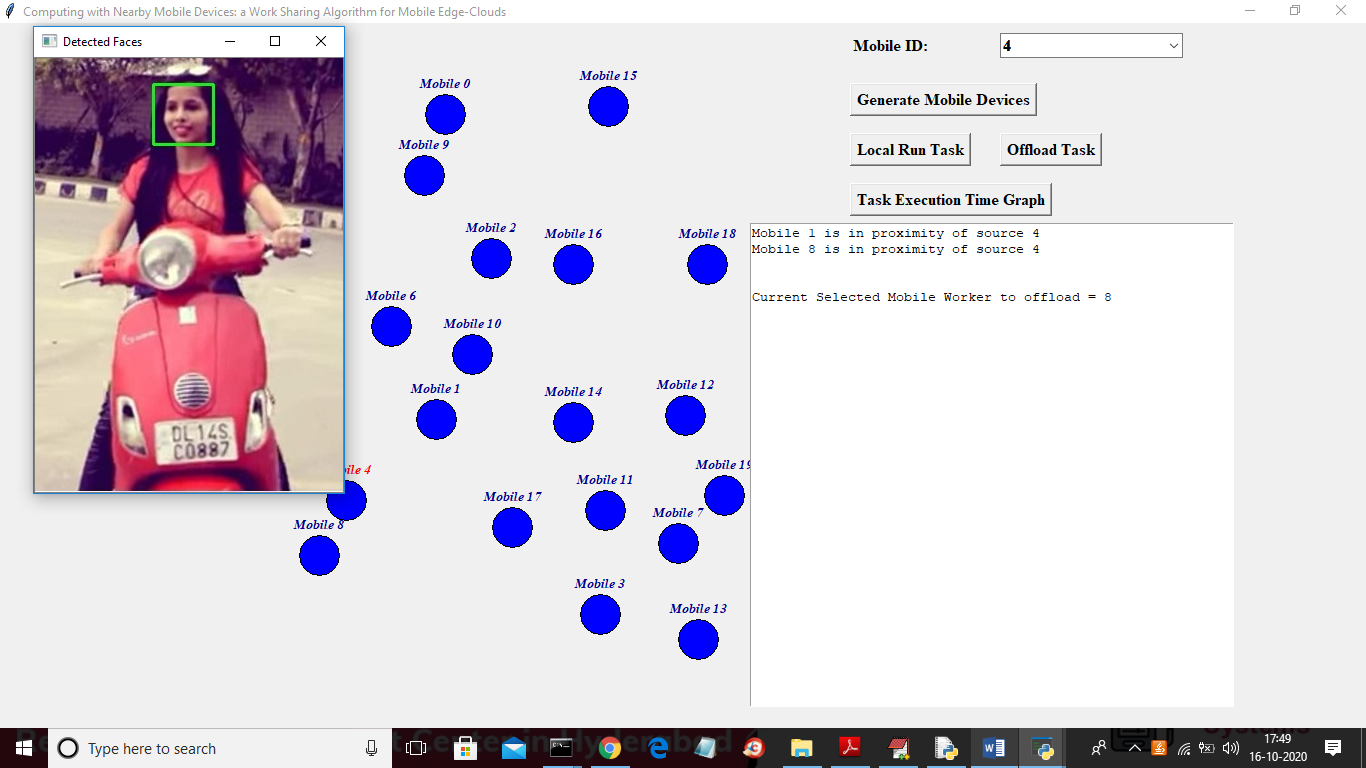
In above screen application has execute task of face detection and in below screen we can see execution time of local running



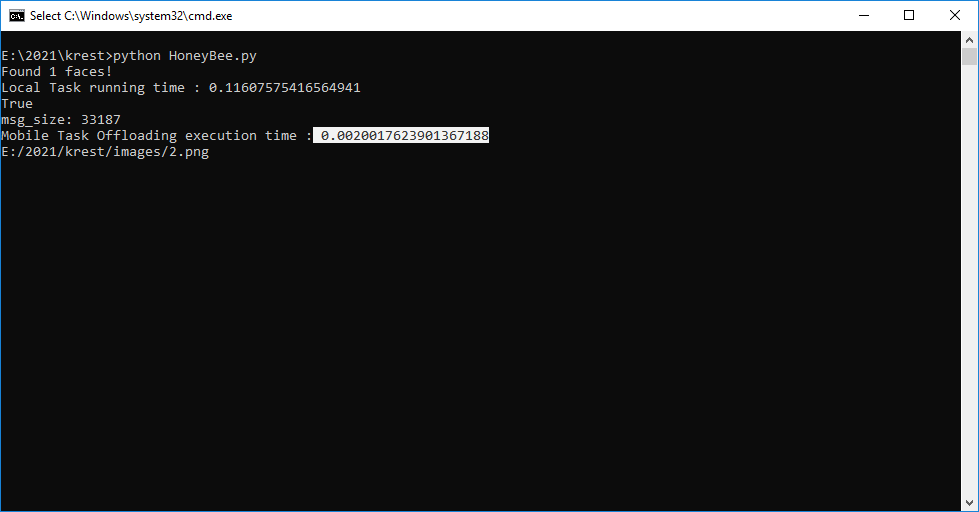
In above screen we can see local task running took 0.11 milli. Seconds to run task and now click on ‘Offload Task’ button to upload same image and then send that task to worker to execute with high speed workers and to capture execution time



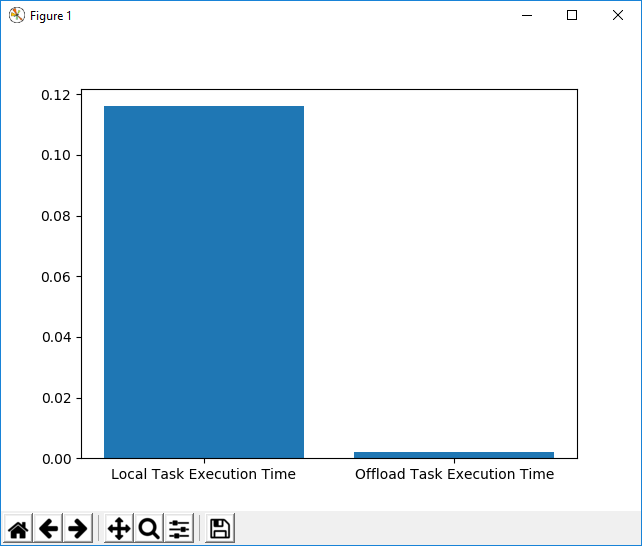
In above screen for offload also we have selected same image and now click on ‘Open’ button to offload task



In above screen by offloading task also we got face detection result and in below screen we can see task execution time for offload work stealing algorithm



In above screen mobile offloading task execution took 0.002 milli seconds time which is lower than local running time and now click on ‘Task Execution Time Graph’ button to get below graph



In above screen x-axis represents technique name and y-axis represents execution time and from above graph we can conclude that offload Task execution time is faster than local running task