CS776/876 – Architectural Support for Cloud Computing

Professor Stephan Olariu
Department of Computer Science
Old Dominion University

– Programming Project –

Note: The project is due the week of December 5, 2022

Objective

The goal of your programming project is to simulate a Vehicular Cloud built on top of vehicles on a highway. The challenge facing the implementation of the Vehicular Cloud is to minimize job completion time in the face of the dynamically changing resources. Tradeoffs will be identified and analyzed and several possible solutions will be contrasted. The project is deliberately open ended, allowing each student to make their own assumptions and to add performance-enhancing "bells and whistles".

Before you begin

Please understand that "borrowing" code, either wholesale or merely selectively, from fellow students, or indeed from any other sources including the Internet, constitutes a violation of the Honor Code of Old Dominion University and will be dealt with accordingly.

Baseline project

In this project you are to simulate a *Vehicular Cloud* (VC, for short] that is harvesting the compute power of vehicles moving on a highway. In order to implement this idea, the VC controller is connected by optical fiber to pre-installed *access points* (*AP*s, for short) deployed along the highway. Referring to Figure 1, the access points are placed d meters apart along the highway and are numbered consecutively as $AP_0, AP_1, \cdots, AP_n, \cdots$. As illustrated in Figure 1, each AP has a radio coverage area of c meters.

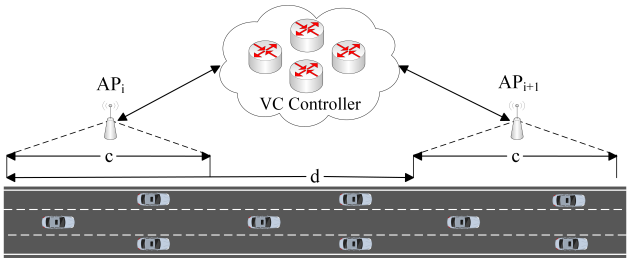

Figure 1: *Illustrating two consecutive APs and their coverage areas.*

A vehicle can communicate with an AP only when it is in its coverage area. To simplify matters, you may assume that the APs are placed at the entrance and exit ramps along the highway. This assumption implies that a vehicle entering the highway is under the coverage area of an AP. Similarly, a vehicle exits the highway under the coverage area of an AP.

Upon entering the highway, each vehicle is assigned a job for processing. Consider a vehicle that just entered the highway at AP_i. The vehicle informs AP_i of the access point AP_j at which it will exit. Given this

information, and the average speed on the highway between AP_i and AP_j, the VC controller can estimate the amount of time the vehicle will spend on the highway. This helps determine the workload that can be allocated to the vehicle for processing.

Jobs to be executed by vehicles are encapsulated as Virtual Machines (VMs, for short). The vehicle will begin by downloading the corresponding VM, will execute the job and, upon termination, will upload the results to the first available AP. In case the vehicle leaves the highway *before* completing job execution, the corresponding VM will have to be migrated to another vehicle. The process of migrating a VM involves uploading the VM before the vehicle departs and then reassigning the VM to a fresh vehicle entering the highway. Beware: VM migration takes time and several migration strategies should be explored as part of the project. When a vehicle departs without having migrated its guest VM, the corresponding job will have to be restarted from scratch.

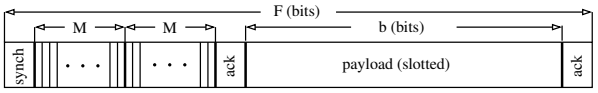

Figure 2: *General structure of a communication frame.*

Communication Model

The communication between APs and passing vehicles is supported by a variant of DSRC that we discuss below.[1] Since each coverage area may contain several vehicles, the communication between the APs and the vehicles in their coverage area is, necessarily, contention-based. Specifically, the APs continuously send out frames of size F bits containing a fixed length payload of b bits as illustrated in Figure 2. In each frame, the vehicles that are under the coverage area and wish to communicate with the AP compete with each other to secure a communication slot in that frame. We assume that each frame has two adjacent contention periods of M minislots each. In each contention slot a vehicle that wishes to communicate with the AP picks a random number between 1 and M and transmits its identity in the corresponding minislot. This is repeated in the second contention period. If two vehicles transmit in the same minislot a collision occurs and the AP receives a garbled message that cannot be disambiguated. Vehicles whose message gets to the AP ungarbled in at least one of the contention periods are considered successful and are allocated *one* data slot in the frame.

From the standpoint of a given vehicle, a frame is *successful* if the vehicle has secured a communication slot in that frame. In each frame, the payload of b bits is partitioned evenly among the successful vehicles in that frame. We assume a communication bandwidth of $B = 27$Mbps which is the maximum data transmission rate in DSRC.

Statistical data to be collected

You are to collect statistical data as described next.

- Define *job completion time* as the total amount of time it takes a job to complete. Observe that job completion time is the sum of job duration and the various overheads incurred, such as downloading the VM (and data) and migrating the corresponding VM. Note that, as mentioned before, in case a job is lost due to the inability to migrate the VM in a timely manner, the job will have to be restarted from scratch, and the job completion time will have to reflect the time wasted in the failed attempt at executing the job;

[1]The Dedicated Short Range Communications (DSRC) is the wireless communication standard for vehicular communications designed to facilitate vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications. DSRC provides high data transfer rates (i.e. 27 Mbps) with minimized latency, which is convenient for the highly mobile nature of vehicles and transportation systems.

- The goal is to minimize job completion time;
- As part of the project, you are to
  - design and evaluate various VM migration strategies;
  - collect simulation statistics showing the performance of your various strategies.

Additional specifications

1. For the purpose of this project you are expected to work individually;
2. Feel free to use the programming language of your choice;
3. Your code must be fully documented in such a way as to make it easy to read and comprehend;
4. Please understand that hard-to-read or undocumented code will loose up to 10% of the marks!
5. Submit the code to olariu@cs.odu.edu following the directions posted.