HW1 – Vector Timestamps

Excepted Effort: 12 hours

Objectives

- Become familiar with vector timestamps and how the behavior in a distributed system with variety of communication patterns
- Review inter-process communication concepts and implementation techniques, including: using UDP and TCP; protocols; message numbers; message serialization/deserialization; concurrent threads; and critical sections for shared data among threads, etc.

Overview

In this assignment, you will implement a set of processes for the purpose of experimenting with inter-process communication patterns and vector timestamps. Specifically, you with implement three different, but similar programs and run one or more instances of each one at the same time. All of the processes running at one time will work together as a distributed application.

Application Description

This distributed application is a model of an infectious disease surveillance system, consisting of multiple instances of programs: an electronic medical record (EMR) simulator (EMR), a health-district system (HDS), and a disease outbreak analyzer (DOA). See Figure 1.

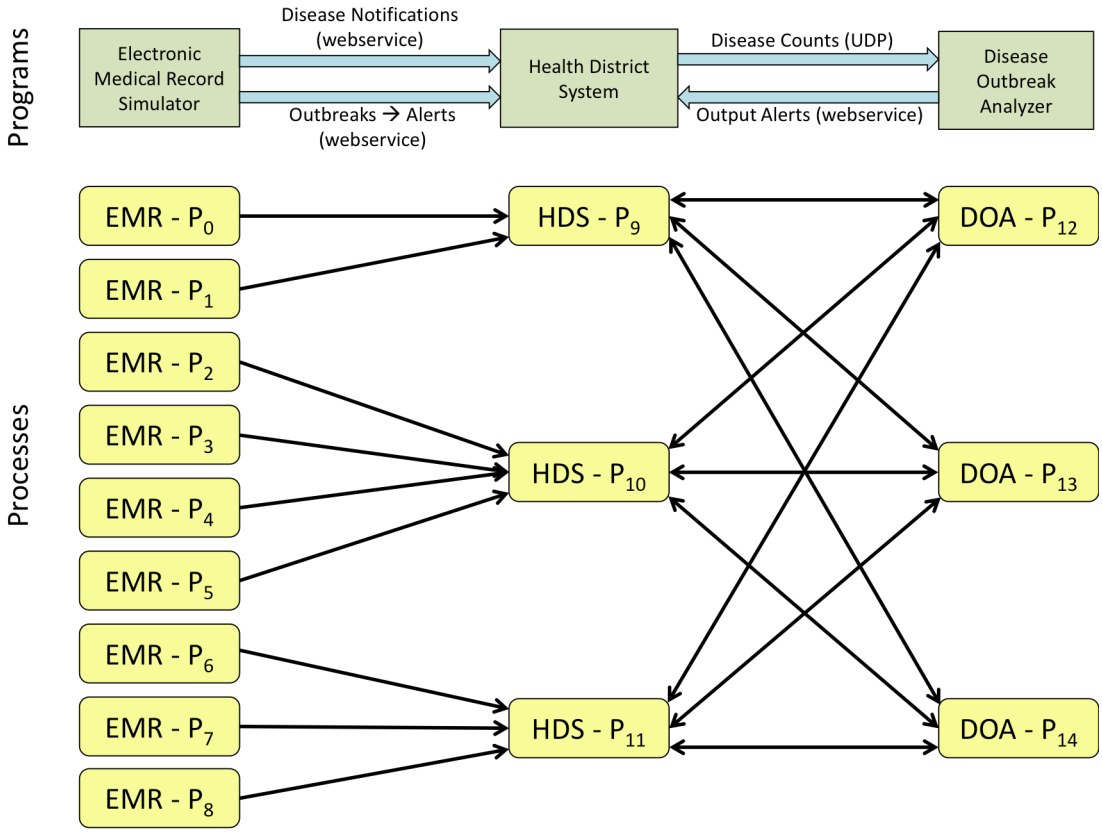
Each EMR process represents the data system operated by a medical facility, like a hospital and clinic. However, it will simulate just one specific aspect of a real EMR, namely it will periodically send information about infectious diseases that have been diagnosis at the medical facility to the HDS.

A disease notification must include the following:

EMR, an integer 0 - 8
Disease, an integer 1-3, where 1=Influenza, 2=Chicken Pox, 3=Measles)
Date and time of the diagnosis, a physical clock's date and time
Vector timestamp, an integer array of size 15.

The EMR will send a disease notification to its assigned HDS via a RESTful webservice. See Figure 1.

Figure 1: Programs & Instances (Processes)



Since this a simulation, the rate at which infectious diseases are reported to a health district can be much faster that in real life. For example, the EMR could send 2 – 200 notifications per second.

Each HDS will collect infectious disease information from one or more EMRS and keep track of a running count of each type of disease. It will periodically (like every second) send also delta count of each disease to the corresponding Disease Analyze (DOA) via a UDP datagram. Influenza counts go to DOA1; Chicken pox counts go to DOA2; and Measles counts to DOA3.

A disease count datagram (message) must include:

DHS, an integer 9 - 11
Disease, an integer 1-3, where 1=Influenza, 2=Chicken Pox, 3=Measles
Delta Count
Vector timestamp, an integer array of size 15.

Each DOA will keep track of a disease rate (notifications per minute) for one type of disease. If the disease rate surpasses a threshold, like 2000 notifications/minute, it

will send an outbreak notification to each HDS, with at least the following information:

> Disease, an integer 1-3, where 1=Influenza, 2=Chicken Pox, 3=Measles
> Vector timestamp, an integer array of size 15.

Eventually, but not part of HW1, an EMR or Webrowser should be able to query the HDS for a list of current alerts and the HDS's current timestamp.

Instructions

Some starting code will be provided to you, so you can focus most of your time on the vector timestamps. However, by reusing the starting code, you make it your own. You become response for know what it does and make sure it works.

You job is to implement all three programs, configure a start-up script that configures and launch all the necessary instances of each. It will also run through a variety of scenarios, logging the vector times for each instance. Finally, analyze vector timestamps for the scenarios and report on your observations.

We will use **node.js** as our implement programming language, with **WebStorm** as the EDI and debugger.

The simulator code in the EMR really simple, but make the initial rates at which it generates notifications for each type of disease parameters. The simulator code should gradually increase each rates to a maximum, which should also be a parameter, hold it that level until the program stops.

On way of implementing the outbreak detection logic in the DOA, is to track delta disease counts in a circular buffer, where each slot in the buffer represents counts for 1 second intervals. Then compute the rate over a fixed window of some size, like 5. Doing that will help smooth out bumps and dips caused by network traffic congestion.

All processes need to keep track of a local vector timestamp and all message must contain the sending process's local vector timestamp. All process should log their current vector timestamps anytime they change, so you can analyze them afterwards.

Finally, run your system with various rates for notification initial rate, max rate, and acceleration (to be provided later). Examine your logs. Then, makes some observations and document them in a short report.

Zip up root directory, with the report in it, and submit to Canvas.

Grading Criteria

	Points
	=====
Well-design, well-implement, and working EMR	20
Well-design, well-implement, and working DHS	20
DOA	20
Report	15