CptS 464/564 Project #3
Implementation of “logical clock” and making “consistent global cut”
Given: Friday, April 10, 2015
Due: 12:00pm, Thursday, May 7, 2015, via BlackBoard
Weight: 25% of Final Grade

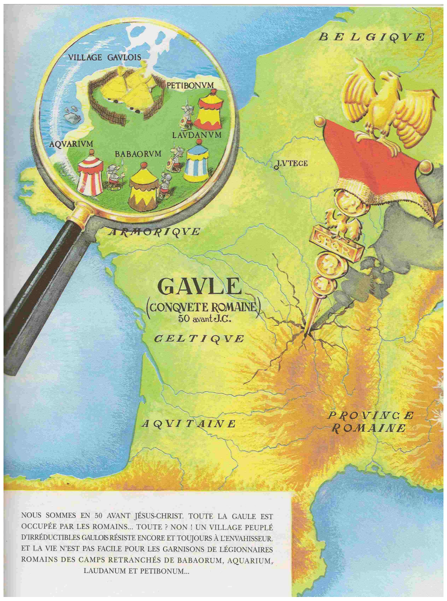
April 10, 2015

1 Overview

In this programming assignment you will implement **logical clock** in a distributed system and to make a **consistent global cut using the Chandy-Lamport snapshot algorithm (link to original paper)**. You will use DETERLab for this project, so you *must* read the “Student Introduction to DETERLab” – `https://education.deterlab.net/DETERintro/DETERintro.html` before starting the experiments. I hope you already have been enrolled to use DETERLab, if not, then get in touch with the TA ASAP.

2 Introduction

Caesar summoned three of his favorite generals Brutus, Pompus, and Operachorus to share his latest strategy to attack the Gaul village. “This time, we’ll attack them from four directions”, said Caesar. The generals agreed and all four armies began their march towards the village right away. Upon arriving near the village, the the generals realized that the village is surrounded by four hills. So, Caesar ordered each army to set camp on each of the hills and wait for further orders. Caesar is a highly methodical person and wants all generals to be in-sync. Roman generals are known for their excellent vector clock skills so Caesar plans to use it to synchronize his armies. As his first order, Caesar sends a message to all three generals to report him back with the count of their army units – *catapults, archers, cavalry, spearmen, and infantry.* He wants each army



NOUS SOMMES EN 50 AVANT JÉSUS-CHRIST. TOUTE LA GAULE EST OCCUPÉE PAR LES ROMAINS... TOUTE ? NON ! UN VILLAGE PEUPLÉ D'IRRÉDUCTIBLES GAULOIS RÉSISTE ENCORE ET TOUJOURS À L'ENVAHISSEUR. ET LA VIE N'EST PAS FACILE POUR LES GARNISONS DE LÉGIONNAIRES ROMAINS DES CAMPS RETRANCHÉS DE BABAORUM, AQUARIUM, LAUDANUM ET PETIBONUM...

to have an equal number of units. Those days, the only form of communication was to use a human messenger. Messengers are usually the fastest soldiers in the army but they are not reliable. Each general obeys Caesar's order and sends their respective unit counts in a single message. Thus assume trustworthy generals. Upon receiving the unit counts from each general, Caesar calculates how many units should be relocated to and from each army and instructs the respective generals. For example, one such relocation command could be "Brutus, send 30 infantry to Pompus". Each general carries out the order he receives by redeploying the units mentioned in their message.

2.1 Instructions

- Take continuous backup of your code, in DETERLab when you swap out then nothing is saved, so keep on putting everything in your `home` folder
- Implement vector clocks for a 4 node distributed system. While you are free to use a programming language of your choice, the programming platform must be Linux/Unix
- Use sockets as the communication paradigm and UDP as the transport protocol
- Use a random number generator [1-100] to generate the initial unit count for each army

- Implement the event sequence outlined in the *Introduction* section. You are more that welcome to extend events (and extend the story!!) beyond what is mentioned ☺
- Make a *consistent global cut* of the system using the Chandy-Lamport snapshot algorithm in the midst of troops redeployment (i.e., some troops will be still in-route to their destination)

3 General Instructions

The project is due at 12:00 PM (noon) on Thursday, May 7, 2015, to be submitted via BlackBoard. Please put all relevant doc/code files in a folder, and compress it before uploading to Blackboard. Please follow this naming convention of the compressed file:
`CS<464/564>Project3<your_username>.zip`
Apart from Blackboard, you will also need to make sure your experiment is ready to run when I swap-in after the due-date (in DETERLab) [read below]. Also, *do not* contact the DETERLab support if you have an issue, we might *deduct points* if you do it, as this is breaking DETERLab protocols. If you face an issue regarding DETERLab, contact the TA on his eecs email ASAP [sghosh1 at eecs dot wsu dot edu].
DETERLab is a *shared testbed*, which means *do not* leave your work during the last week the project is due, you must start early. Some general good practices/information in using DETER:

- First, read the student introduction to DETERLab: `https://education.deterlab.net/DETERintro/DETERintro.html`, you will need to understand some basic things, like learn how to save the work and retrieve it on the next swap in.
- Set the *idle swap* out period to 1 hour. You should swap out manually whenever you are done with a chunk of work, but this setting will cover you when you forget to do so.
- Contact the TA for any issue that you may be facing (use his eecs email), do not contact DETER support directly
- Some absolutely crucial things to re-reiterate – `https://education.deterlab.net/DETERintro/DETERintro.html#controlnet`
- You would first need an NS (Network Simulator) file to begin your experiment; you will find useful information in creating experiments here: `https://trac.deterlab.net/wiki/Tutorial/CreatingExperiments`
- Some Frequently Asked Questions on DETER: `https://trac.deterlab.net/wiki/FrequentlyAskedQuestions`

4 Deliverables

- A short report of ~5 pages (PDF format please) with the of the design and implementation of your system. Your report should include the followings:
  - Introduction
  - How to run your software (Remember! I give you the liberty to choose your pro- gramming language of choice. Don't expect my system to have all the software needed to run your program. So make sure to include such information)
  - The output of a sample run (including screenshots where applicable)
  - Proof showing that your cut is in fact consistent. This can be verified by showing the total number of units (in each army and in transit messages) is equal to that at the beginning of the run
  - Output of a sample run, with a clear/legible snapshot
  - Summary and Issues encountered, Challenges, et al (if applicable)
- You must ensure that your experiment is ready to compile and run (i.e., all necessary software is automatically installed) when I swap-in. Clearly mention where to find your code, the instructions I need to follow, and their respective order.
- Your NS file