COMPUTER SCIENCE LARGE PRACTICAL

INTRODUCTION

Paul Patras

HOUSEKEEPING

- Web: http://www.inf.ed.ac.uk/teaching/courses/cslp/
- One lecture per week
  - When: Fridays, 12:10–13:00
  - Where: Old Infirmary (Geography) building - Map Room 2.19 (Old Library) – this may change!
- Please ask questions at any time
- Coursework accounts for 100% of your mark
- Office hours: flexible, but email me first (paul.patras@ed.ac.uk)

RESTRICTIONS (I)

- CSLP is a third-year undergraduate course only available to third-year undergraduate students.
- CSLP is not available to visiting undergraduate students, or to fourth-year undergraduate students and MSc students, who have their own individual projects.

RESTRICTIONS (II)

- Third-year undergraduate students should choose **at most one** large practical, as allowed by their degree regulations.
  - Computer Science, ~~Software Engineering~~ and Artificial Intelligence large practicals.
  - On most degrees a large practical is compulsory.
  - On some degrees (typically combined Hons) you can do the System Design Project instead/additionally.
- See Degree Programme Tables (DPT) in the Degree Regulations and Programmes of Study (DRPS) for clarifications.

CLARIFICATIONS

- SELP is not offered this year, due to a number of unforeseeable circumstances.
- The School of Informatics apologises for this, and has put an important concession in place (I will come back to this shortly).
- Still, you will make extensive use of skills in the engineering of software through the CSLP.

ABOUT THIS COURSE

- So far most of your practicals have been small exercises
- This practical is larger and less rigidly defined than previous course works.
- The CSLP tries to prepare you for
  - The System Design Project (in the second semester);
  - The Individual Project (in fourth year).

REQUIREMENTS

- There is:
  - a set of requirements (rather than a specification);
  - a design element to the course; and
  - more scope for creativity.
- The requirements are more realistic than most coursework,
- But still a little contrived in order to allow for grading.

HOW MUCH TIME SHOULD I SPEND?

- 100 hours, all in Semester 1, of which
- 8 hours lecture/demonstrating,
- 92 hours practical work.

HOW MUCH TIME IS THAT REALLY?

- 12 weeks remaining in semester 1 (Weeks 2 to 13).
- 7.5 * 12 = 90 hours.
- You can think of it as 7.5 hours/week in the first semester.
- This could be one hour a day including weekends.
- You could work 7.5 hours in a single day,
  - for example work 9:00-17:30 with an hour for lunch.

MANAGING YOUR TIME

It is unlikely that you will want to arrange your work on your large practical as one day where you do nothing else, but one day per week all semester is the **amount** of work that you should do for the course.

Course lecturers have been asked not to let deadlines overlap Weeks 11-13 because students are expected to be concentrating on their large practical in that time.

DEADLINES

The Computer Science Large Practical has two parts:

- **Part 1**
  - Deadline: Thursday 22nd October, 2015 at 16:00
  - Part 1 is **zero-weighted**: it is just for feedback.
- **Part 2**
  - Deadline: ~~Thursday 17th December~~
    Monday 21st December, 2015 at 16:00
  - Part 2 is **worth 100%** of the marks.

SCHEDULING WORK

- It is **not** necessary to keep working on the project right up to the deadline.
- For example, if you are travelling home for Christmas you might wish to submit the project early.
- In this case ensure that you **start** the project early.
- The coursework submission is electronic so it is possible to submit remotely,
  - But you must make sure that your submission works as expected on DiCE.
  - This *might* be easier to do locally, but see working remotely and remote graphical login.

~~EARLY SUBMISSION~~
CONCESSION

- As compared to last year, work submitted less than a week before the deadline will **not** be capped at 90%.
- This year you will be benefiting from 1 week+ to work full time on the course after lectures have finished.
- This is incredibly rare in the current university calendar and especially valuable for projects!

EXTENSIONS

- Do not ask me for an extension as I cannot grant them.
- The correct place is the ITO who will pass this on to the year organiser (Vijay Nagrajan).
- See the policy on late coursework submission first.

THE COMPUTER SCIENCE LARGE PRACTICAL

THE CSLP REQUIREMENT

- Create a command-line application in C.
- The purpose of the application is to implement a *stochastic, discrete-event, discrete time* simulator
  (I'll come back to these terms).
- This will simulate an on-demand public transport system for future cities, with stop locations, minibus capacities, user behaviour, etc. specified by input.

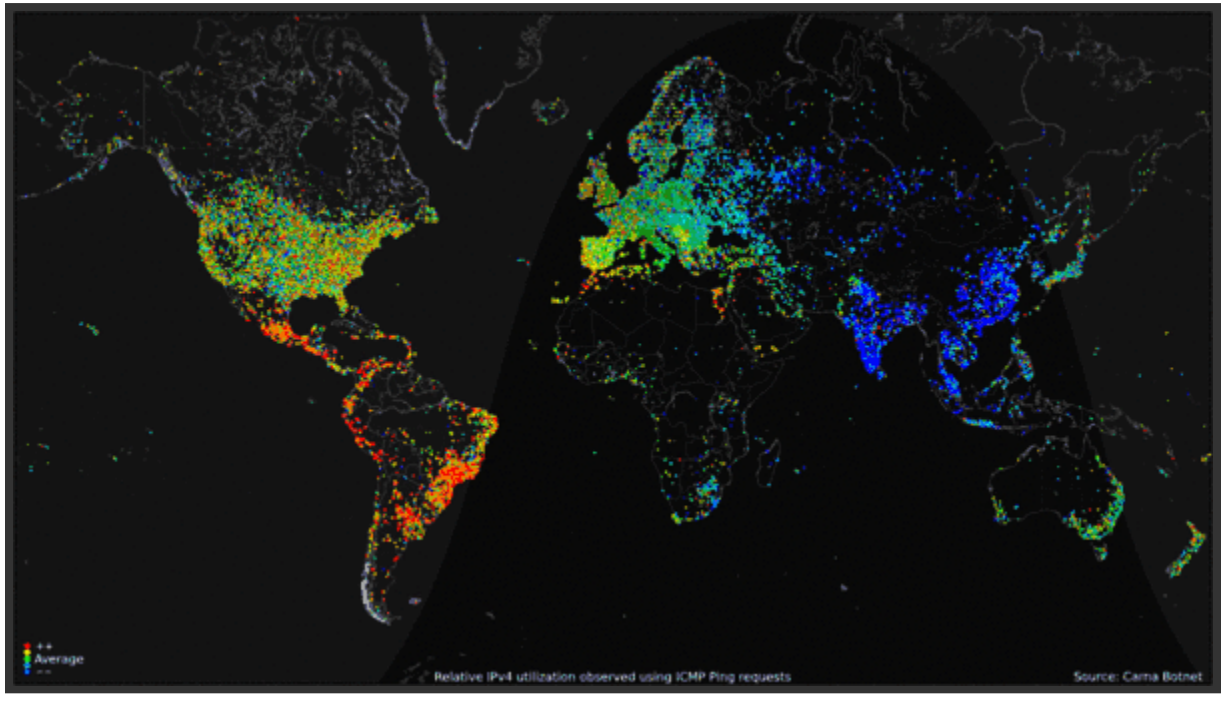
THE CSLP REQUIREMENT (C'TND)

- The output will be the sequence of events that have been simulated, as well as some summary statistics.
- Input and output formats, and several other requirements are specified in the coursework handout.
- It is your responsibility to **read the requirements carefully**.

WHY SIMULATORS?

- Stochastic simulation is an important tool in physics, medicine, computer networking, logistics, and many other fields.
- Particularly useful to understand complicated processes.
- Can save time, money, effort and even lives.
- Allow running inexpensive experiments of exceptional circumstances that might otherwise be infeasible.
- However, the simulator must have an appropriate model for the real system under investigation, to produce meaningful results.

EXAMPLE: PREVENTING INTERNET OUTAGES



Source: Internet Census –World map of 24 hour relative average utilization of IPv4 addresses.

Last year CBC news reported that in the U.S. Verizon dumped 15,000 Internet destinations for ~10 minutes.

PREVENTING INTERNET OUTAGES

- Global Internet routing table has passed 512K routes.
- Older routers have limited size routing tables; when these fill up, new routes are discarded.
- Large portions of the Internet become unreachable, thus online businesses are loosing money.
- Upgrading equipment is expensive and takes time; workarounds are being proposed.
- Ensuring the proposed solutions will work is not trivial.

PREVENTING INTERNET OUTAGES

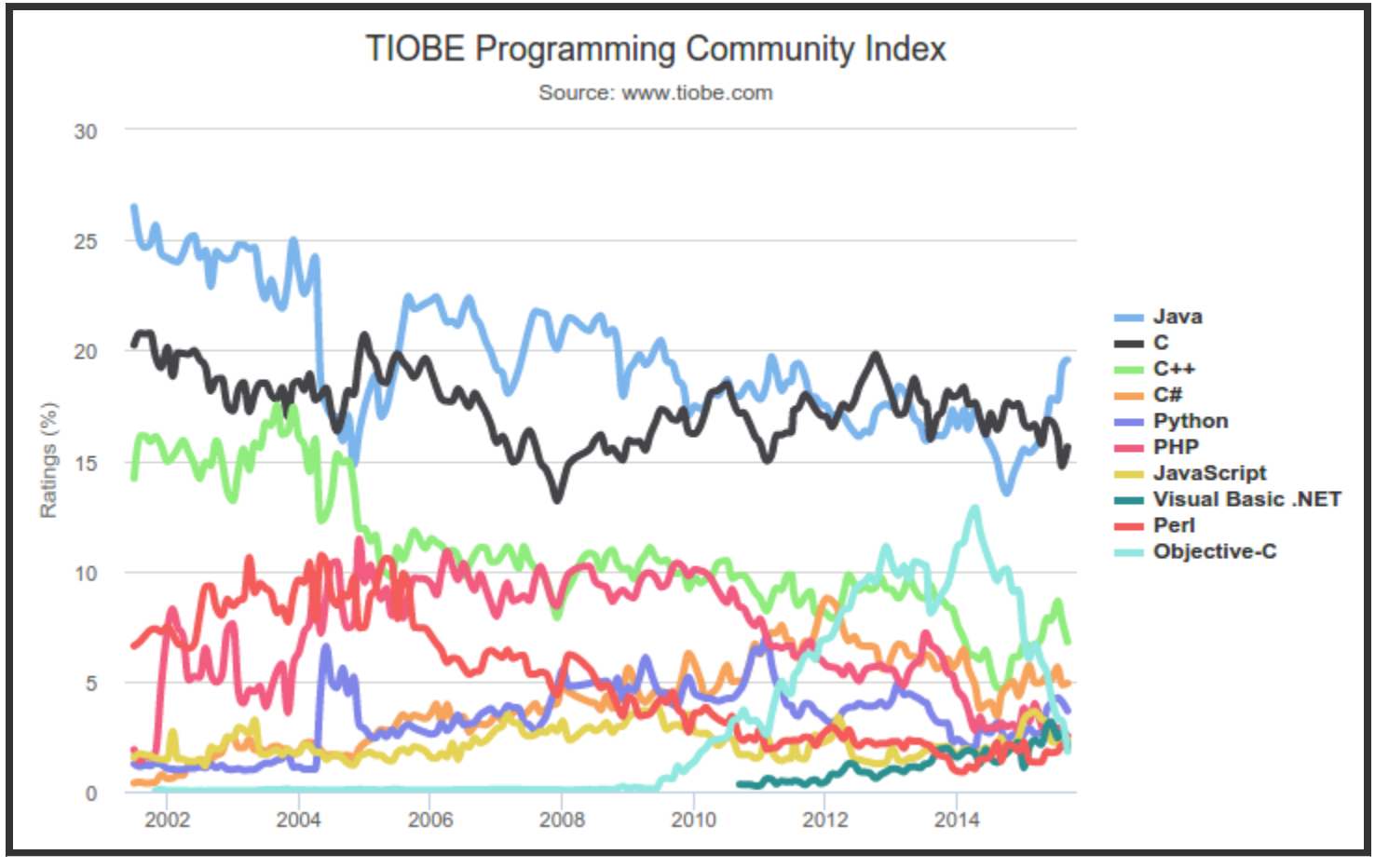
- Testing patches in live networks poses the risk of further disruption.
- Waiting for the next surge is not acceptable.
- Forwarding all traffic for new routes through a default interface has serious implications on routing costs.
- With simulation it is possible to generate synthetic traffic and test patches without disrupting the network.
- It is also possible to evaluate different metrics, e.g. round-trip delays, throughput, propagation latency of routing changes.

WHY C?

- Part of the challenge of this practical is to enhance your skills with a procedural language widely used for system programming.
- This is something you should expect when taking a job as a software developer in a company that has clear incentives to use a particular language.
- C is efficient (low execution time), portable, excellent for working directly with the hardware, and also usable for web programming.

WHY C?

- Currently ranked among the most popular programming languages --TIOBE Index, Sept. 2015.



CODE SHARING

- Code sharing sites are a great resource but please refrain from using them for this practical.
- This is an individual practical so code sharing is not allowed. Even if you are not the one benefiting.
- It is somewhat likely that in the future you will be unable to publicly share the code you produce for your employer.

WHY SIMULATE AN ON-DEMAND PUBLIC TRANSPORT SYSTEM?

- This approach is currently considered by European city councils as a potential major step towards eliminating the incentives for personal car ownership.
- Reduce carbon emissions in densely populated areas.
- Cut down commute times & improve user satisfaction.
- Limitations of current periodic scheduling practices:
  - Buses make unnecessary frequent trips; sometimes take lengthy routes → increased operation costs.
  - User demand varies geographically and in time → buses sometimes overcrowded.

WHY SIMULATE AN ON-DEMAND PUBLIC TRANSPORT SYSTEM?

- With simulation we can investigate the impact of different service demand rate and acceptable pick-up delays to trigger scheduling.
- In this practical we will evaluate the efficiency of the transportation process in terms of bus occupancy per unit of travel time, average waiting times, etc.
  - Short tolerable waiting times → journeys with fewer passengers; lesser efficiency.
  - Longer waiting intervals → more cost efficient, but impacting user satisfaction.

YOUR SIMULATOR

- Your simulator will be a command-line application.
- It will accept an input text file with the description of the serviced network and
- a set of global parameters: minibus capacity, boarding time, request arrival rate, departure delay, maximum waiting time.
- It should output information about occurring events.
- The **strict** formats for both input and output are described in the coursework handout.
- You will also need to produce summary statistics that you will later analyse.

SIMULATION ALGORITHM

The underlying simulation algorithm is fairly simple:

```
WHILE {time ≤ max time}
   determine the set of events that may occur after the current state
   delay ← choose a delay based on the nearest event
   time ← time + delay
   modify the state of the system based on the current event
ENDWHILE
```

SIMULATION ALGORITHM

```
WHILE {time ≤ max time}
    ...
    delay ← choose a delay based on the nearest event
    ...
ENDWHILE
```

- Some events are deterministic, some occur with exponentially distributed delays.
- I'll explain this in more details, but for now drawing from an exponential distribution can be done by:

```
  -(mean) * log(random(0.0, 1.0))
```

- Where *mean* is the average delay, which is the reciprocal of the rate.

COMPONENTS OF THE SIMULATION

INPUT - GLOBAL PARAMETERS

1. Mini bus capacity
2. Boarding/disembarking time
3. Request rate
4. Pick up interval
5. Maximum admissible delay
6. Number of buses
7. Number of stops
8. Matrix representation of the service network

COMPONENTS OF THE SIMULATION

STOPS

- Passengers only board or disembark at designated minibus stops.
- Consider users book journeys through e.g. a smartphone application and do not necessarily need to be present at a bus stop.
- New requests are 'queued' at bus stops.
- We are only interested in the back-end (thus will not develop a user app).

COMPONENTS OF THE SIMULATION

USERS

- We consider users place requests at exponentially distributed time intervals (with a given mean).
- Randomly choose departure and arrival stops.
- Chosen a desired boarding time that is an exponentially distributed delay after the request time (mean also given as an input parameter).
- Can tolerate a maximum delay after the desired departure time (also given as input).

COMPONENTS OF THE SIMULATION

MINIBUSES

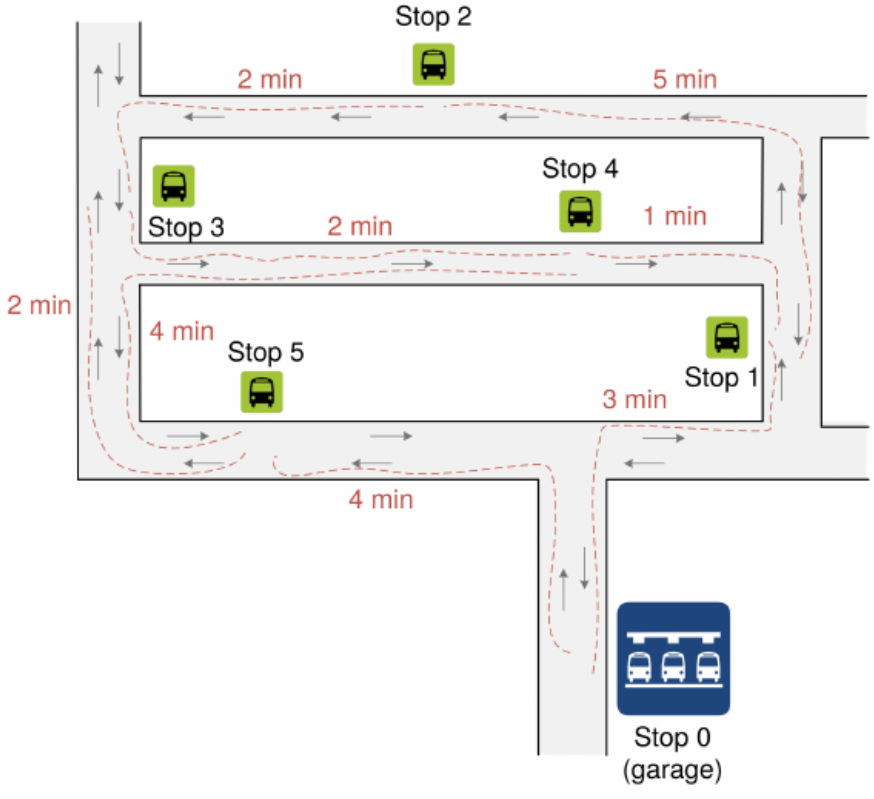
- Have fixed passenger capacity (given as input).
- Time to board/disembark a passenger is constant (and given as input).
- Scheduling and route calculation are things you have to decided on.

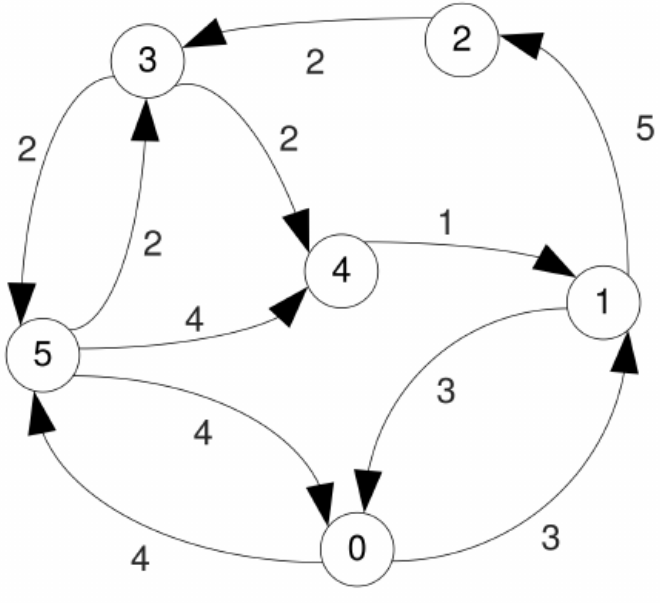
COMPONENTS OF THE SIMULATION

SERVICE NETWORK

- We consider a directed graph representation of the bus stop locations and the distances between them.
- The graph is given as an input in matrix form.
- The distances between any two locations are expressed in minutes.

EXAMPLE


Stop 2
2 min
5 min
Stop 4
Stop 3
2 min
1 min
2 min
4 min
Stop 5
Stop 1
3 min
4 min
Stop 0
(garage)


3
2
2
2
5
2
4
1
1
5
4
3
4
0
4
3

EXAMPLE

- Matrix representation

```
map
 0  3 -1 -1 -1  4
 3  0  5 -1 -1 -1
-1 -1  0  2 -1 -1
-1 -1 -1  0  2  2
-1  1 -1 -1  0 -1
 4 -1 -1  2  4  0
```

COMPONENTS OF THE SIMULATION

ROUTE PLANNING

- The start of a minibus journey can be triggered with any delay.
- Routes of minibuses already in services can be modified as long as this will not alter previously agreed departures.
- Ensure capacity is not exceeded.

COMPONENTS OF THE SIMULATION

ROUTE PLANNING

- Goal: Find the shortest routes that pick up and drop off the largest possible number of passengers that intend to take similar journeys.
- How you achieve this task is your design choice.
- This is a non-trivial problem, so exploring different heuristics is appropriate.

COMPONENTS OF THE SIMULATION EVENTS

- Your simulator will produce a sequence of events
  - New user places a request;
  - Request is scheduled for departure at a time instant;
  - Request cannot be accommodated;
  - Minibus leaves/arrives at a location;
  - A passenger boards / disembarks;
  - Minibus occupancy changes.

COMPONENTS OF THE SIMULATION EVENTS

- Your simulator will output a sequence of events in the following format:

```
‹time› -> new request placed at stop ‹unsigned int› for departure
        at ‹time› scheduled for ‹time›
‹time› -> new request placed at stop ‹unsigned int› for departure
        at ‹time› cannot be accommodated
‹time› -> minibus ‹unsigned int› arrived at stop ‹unsigned int›
‹time› -> minibus ‹unsigned int› left stop ‹unsigned int›
‹time› -> minibus ‹unsigned int› boarded passenger at
        stop ‹unsigned int›
‹time› -> minibus ‹unsigned int› disembarked passenger at
        stop ‹unsigned int›
‹time› -> minibus ‹unsigned int› occupancy became ‹unsigned int›
```

COMPONENTS OF THE SIMULATION EVENTS

- Depending on the actual event in your simulation, you will replace the variables with real values, e.g.:

```
00:01:20:00 -> new request placed at stop 2 for departure
        at 00:01:32:00 scheduled for 00:01:33:00
00:01:25:10 -> new request placed at stop 3 for departure
        at 00:01:34:00 scheduled for 00:01:36:00
00:01:28:00 -> minibus 1 left stop 1
00:01:33:00 -> minibus 1 arrived at stop 2
00:01:33:10 -> minibus 1 boarded passenger at stop 2
00:01:33:10 -> minibus 1 occupancy became 4
...
```

- This is valid output in the sense that it is formatted correctly, but may be invalid for semantic reasons.

IMPORTANT!

- Part of your code will be subject to automated testing.
- Strictly abiding to the input/output specification and command line formatting is mandatory.
- Your code may be functional, but you will lose points if it fails on automated tests.
- This is something you should expect with the evaluation of commercial products as well.

PART ONE & PART TWO ASSESSMENTS

- Part one, is just for feedback. You only need to have a working simulator
- For part two, there are additional requirements:
  - *Full functionality* should be implemented.
  - *Summary statistics*, such as trip efficiency, should be produced.
  - *Experimentation support*, e.g. varying the fleet size to see how this impacts different metrics.
  - *Validation*, checking that the input is valid.

PART ONE & PART TWO ASSESSMENTS

- These are all specified in the **coursework handout**, available at: http://www.inf.ed.ac.uk/teaching/courses/cslp/handout/cs 2015-16.pdf
- This was a brief summary of the major components of the simulation.
- It is **no** substitute for reading the coursework handout.
- Try to submit an alpha version of your simulator for part 1 carries zero weight, but will help you for part 2!