COMPUTER SCIENCE LARGE PRACTICAL

INTRODUCTION

Paul Patras

HOUSEKEEPING

- One lecture per week
  - When: Fridays, 12:10–13:00.
- Coursework accounts for 100% of your mark.
- Please ask questions at any time.
- Ask offline questions on Piazza first – self-enrol **here**.
- **Office hours:** Thursdays @ 10:00 in IF-2.03 (if you cannot make it and want to meet, email **ppatras@inf.ed.ac.uk**).

RESTRICTIONS

- CSLP only available to third-year undergraduate students.
- Not available to visiting UG, UG4, and MSc students.
- UG3 students should choose **at most one** large practical, as allowed by their degree regulations.
  - 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.

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.
- Requirements are more realistic than most coursework,
- But still a little contrived in order to allow for grading.

HOW MUCH TIME SHOULD I SPEND?

- CSLP is now a 20 credit course.
- 200 hours, all in Semester 1, of which
- 8 hours lectures;
- 4 hours programme level learning and teaching (office hours, PT meetings, training, etc.);
- **188 hours individual practical work.**

HOW MUCH TIME IS THAT REALLY?

- 12.5 weeks remaining in semester 1 (Weeks 2 to 14).
- 15 * 12.5 = 187.5 hours.
- You can think of it as 15 hours/week in the first semester.
- This could be just over 2 hours/day including weekends.
- You could work 7.5 hours in a single day, twice a week
  - for example work two days between 9:00-17:30, with an hour for lunch.

MANAGING YOUR TIME

You may not want to arrange your work on the LP as two days where you do nothing else, but 2 days/week all semester is the **amount** of work you should do for CSLP.

You should not have other deadlines overlap Weeks 11-13 as your are expected to concentrate on large practicals then.

Plan wisely to avoid unpleasant surprises!

DEADLINES

- **Part 1**
  - Deadline: Friday 7th October, 2016 at 16:00.
  - Part 1 is **zero-weighted**: it is just for feedback.
- **Part 2**
  - Deadline: Friday 11th November, 2016 at 16:00.
  - Part 2 is **worth 50%** of the marks.
- **Part 3**
  - Deadline: Wednesday 21st December, 2016 at 16:00.
  - Part 3 is **worth 50%** 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 finish the project early.
- In this case ensure that you **start** the project early.
- The coursework submission is electronic (commit through Bitbucket) so it is possible to submit remotely,
  - But you must **make sure that your code works as expected on DiCE.**

EXTENSIONS

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

THE COMPUTER SCIENCE LARGE PRACTICAL

THE CSLP REQUIREMENT

- Create a command-line application in a programming language of your choice (as long as it compiles and runs on DiCE).
- The purpose of the application is to implement a *stochastic, discrete-event, discrete time* simulator
  (I will come back to these terms).
- This will simulate the bin collection process in a "smart" city, with bin locations, capacities, etc. given as input.

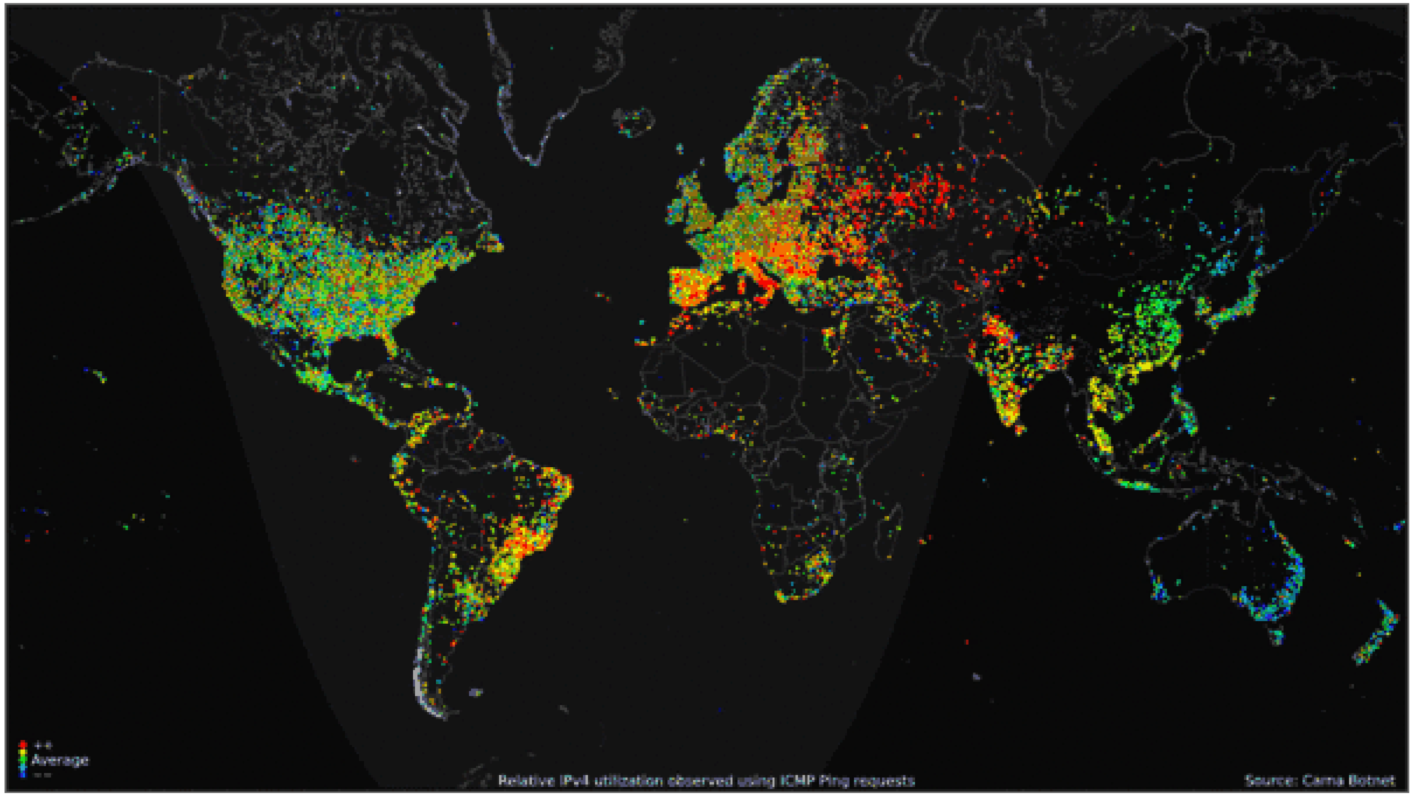
THE CSLP REQUIREMENT (C'TND)

- 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, etc.
- 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.

Two years ago **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.

WHY SIMULATE BIN COLLECTION?

- Waste management is a major operation in many cities.
- Part of ongoing smart cities initiatives, bins are being equipped with occupancy sensors to improve scheduling and route planning for lorries.
- Limitations of current periodic collection strategies:
  - Lorries make unnecessary frequent trips and sometimes take lengthy routes → increased operation cost and pollution.
  - User daily demand varies and could cause overflows before scheduled collection → increased health hazards and cleaning costs.

WHY SIMULATE BIN COLLECTION?

- With simulation we can investigate the impact of different service intervals and bin occupancy thresholds used to trigger scheduling.
- In this practical we will evaluate waste collection efficiency in terms of volume collected per service, percentage of overflows, etc.
  - small thresholds → longer trips, but cleaner streets;
  - large thresholds → cost efficient, but risk of overflows.

YOUR SIMULATOR

- Your simulator will be a command-line application.
- It will accept an input text file with the description of the serviced areas and a set of global parameters.
- 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.

BASH SCRIPTS

To allow for **automated testing** and give you the opportunity to choose the programming language you are most familiar with, you will be given two skeleton Bash scripts, which you need to modify.

```
$ ./compile.sh
```

will be used to compile your code. The choice of compilation method is yours (as long as it works on DiCE).

```
$ ./simulate.sh input_file_name [OPTIONS]
```

will be used to run your simulator with one (or more) command line argument(s).

SIMULATION ALGORITHM

The underlying simulation algorithm is fairly simple:

```
WHILE {time ≤ max time}
   determine the events that will 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 delays that follow an *Erlang-k* distribution.
- I'll explain this in more details, but for now this is effectively the distribution of the sum of *k* independent exponential variables with mean $\mu = 1/\lambda$, where λ is the rate and given.

COMPONENTS OF THE SIMULATION

INPUT - GLOBAL PARAMETERS

1. lorry volume,
2. maximum lorry load,
3. bin service duration,
4. bin volume,
5. rate and shape of the distribution of bag disposal events,
6. rubbish bag volume,
7. minimum and maximum bag weight limits,
8. number of areas.

COMPONENTS OF THE SIMULATION

INPUT

- Area description and dynamic parameters:
  1. Collection frequency;
  2. Bin occupancy threshold;
  3. Number of bins;
  4. Matrix representation of roads layout.

COMPONENTS OF THE SIMULATION

USERS

- At any bin, users dispose of waste bags at time intervals that follow an Erlang-k distribution (rate & shape given).
- Bags of fixed volume (in cubic meters), given as input.
- Bag weight (in kg) is a random value, uniformly distributed between a lower and an upper bound.

COMPONENTS OF THE SIMULATION

BINS & LORRIES

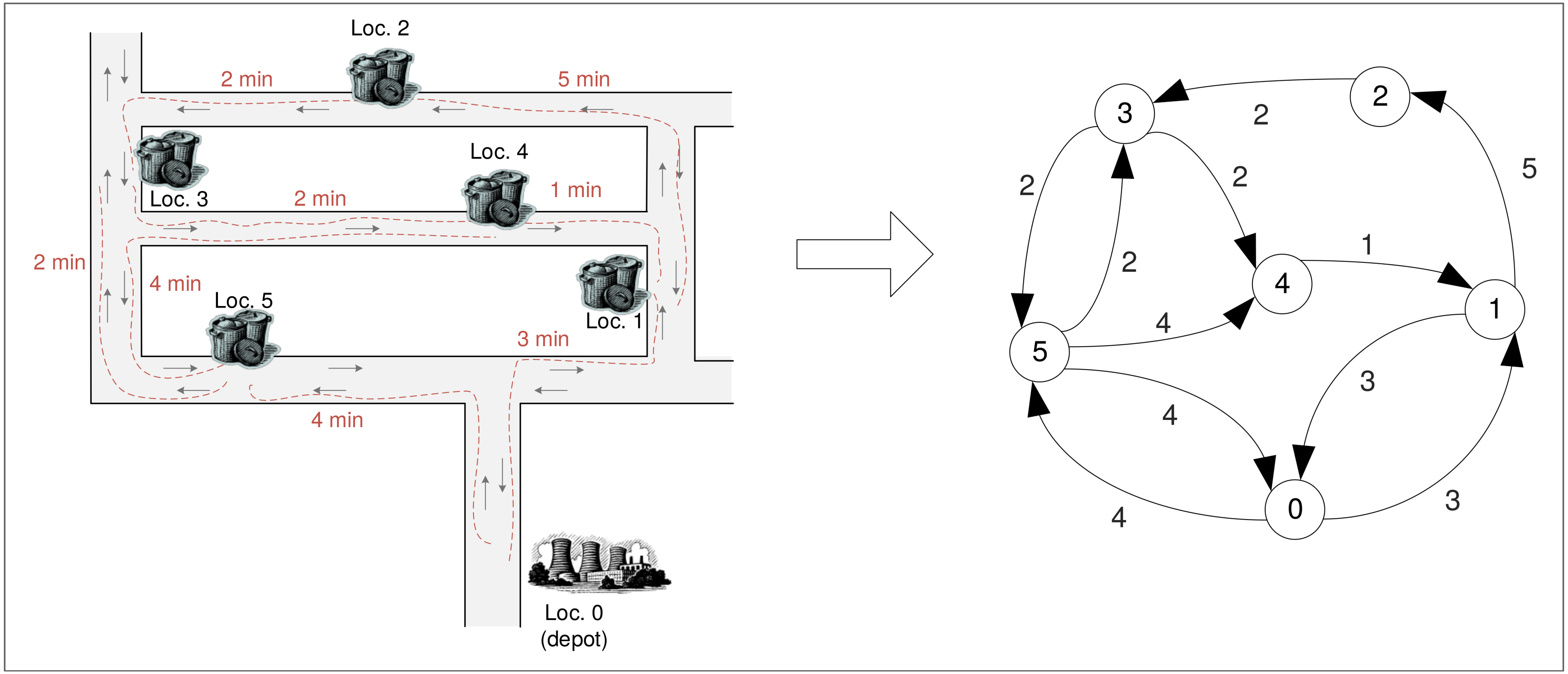
- Community bins have a fixed capacity expressed in m^3; no weight constraints.
- Bins have occupancy sensors and consider *occupancy thresholds* used to trigger collection.
- *Fixed service time* (in minutes) required to empty a bin, irrespective of its occupancy; service at depot 5x longer.
- Lorries have fixed capacity in terms of volume and weight.
- Lorries are scheduled periodically at fixed intervals.

COMPONENTS OF THE SIMULATION

SERVICE AREAS

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

EXAMPLE


Loc. 2
2 min
5 min
Loc. 4
Loc. 3
2 min
1 min
2 min
4 min
Loc. 5
Loc. 1
3 min
4 min
Loc. 0
(depot)

EXAMPLE

- Matrix representation

```
# Road layout and distances between bin locations (in minutes)
roadsLayout
 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

- At every schedule, occupancy thresholds used to decide which bins need to be serviced.
- All routes are circular, i.e. they must start and end at the origin (the depot).
- Goal: compute optimal routes that visit all bins requiring service and minimum costs.
- How you achieve this task is your design choice.
- Exploring different heuristics is appropriate.

COMPONENTS OF THE SIMULATION

EVENTS

- Your simulator will produce a sequence of events
    - bag disposed of;
    - bin load/contents volume changed;
    - bin occupancy threshold exceeded;
    - bin overflowed;
    - lorry arrived at/left location;
    - lorry load/contents volume changed.

COMPONENTS OF THE SIMULATION

EVENTS

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

```
‹time› -> ‹event› ‹details›
```

- Read the handout for full specification of output format.
- Remember your code will be automatically tested, thus you need to **strictly** follow the output specification.
- Some output could be valid in the sense that it is formatted correctly, but may be invalid for semantic reasons.

GETTING STARTED

- For this project you will use the git source control system to manage your code.
- Create a Bitbucket account with your university email address and fork the CSLP-16-17 repository.
- Simple instructions on how to do this in the handout.
- If you haven't done this already, **do this today!**
  It only takes a few minutes.
- Keep your repository private, but do give the marker and me permissions to access that.

CODE SHARING

- 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.
- We will perform similarity checks and report any possible cases of academic misconduct. Don't risk it!
- Start early, ask questions.

IMPORTANT!

- 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.
- Your code will be automatically tested every week and you will be able to track your progress through an online scoreboard (details over the next days).

ASSESSMENT

- Part 1, is just for feedback. You only need to submit a proposal document.
- For part 2, you must have a program that compiles and runs without errors, and:
  - Performs parsing and validation of input;
  - Generates and schedules disposal events;
  - Produces correctly formatted output;
  - Is accompanied by tests scripts;
  - Is appropriately structured and commented;

ASSESSMENT

- For part 3, must have a fully functional simulator that:
  - Performs route planning;
  - Produces correct summary statistics;
  - Supports experimentation;
  - Is tested and optimised;
  - Has evidence of appropriate source control usage;
- Also produce a written report describing architecture, design choices, testing efforts, experiments performed, results obtained, insights gained.
- This lecture is a summary and **by no means a substitute for reading the coursework handout**.