UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

# Discrete Event Simulation

Renato Lo Cigno

http://disi.unitn.it/locigno/index.php/teaching-duties/spe

# DES: definitions

"Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system."

*Robert E. Shannon 1975*

# DES: definitions

- A simulation is a dynamic model that represent (all) the essential characteristics of a real system
- "Essentiality" of the characteristics is what differentiate a good from a bad simulator
  - Not always more details mean a better simulator
- Simulations may be deterministic or stochastic, static or dynamic, continuous or discrete
- DES is (pseudo)stochastic, dynamic, and discrete
- DES is essentially a computer program

# DES: definitions

- DES has been around since 40 years at least
  - Partially art
  - Partially theory
  - Partially programming skills
- Repeatability
- Ability to use multiple parameter sets
- Usually involves posing one or more PE questions
  - Measures of the simulation gives (partial) answers
  - Answers must be properly interpreted (like in physical measures)

# DES: definitions

- A Discrete Event Simulation Model is:
  - *Stochastic*: some state variables are (pseudo)**random**
  - *Dynamic*: the system **evolves in time**
  - *Discrete Event*: significant changes occur at **discrete time instances**
    - The mapping onto a DTMC is natural!

- A continuous time system sampled at constant times is a sub-case of DES

# DES Modelling Process

- Determine the goals and objectives
- Build a conceptual model
- Convert into a specification model
- Convert into a computational model
- Verify
- Validate

Iterate to improve if necessary

# Modeling Layers

1. **Conceptual**

- Highly abstract level
- How comprehensive should the model be?
- What are the *state variables?*
- Which are dynamic?
- Which are the most important?
- Which are random?
- What are the external events that influence the system?
- What are the actions/reactions of the system to the inputs?

# Modeling Layers

1. **Specification**

- On paper, with schemes, block diagrams
- May involve equations, pseudocode, etc.
- How will the model receive input?
- What will be its outputs?
- What are the measure units of my variables?
- What is the “lifetime” of the simulation?

# Modeling Layers

1. **Computational**

- A computer program
- General-purpose PL or simulation language?
- What simulator or what language?
- Is computational performance and issue
- Do my simulator need a lot of memory (e.g., multiple “objects” to instantiate)
- Data structures
- Time representation (more later)
- Event list (more later)

# Verification

- The computational model must reflect the specification model
- Does our program implement the correct model?
- Is our program reasonably bug-free?

1. Control memory leakage
2. Control limit case for which you know the answer
   1. E.g., load a finite M/G/1 queue with $\lambda = 2\mu$ and verity that 50% of the customers are lost
3. "Look" at the evolution of your outputs to spot strange behaviors (interactive graphics may help)

# Validation

- Once we are reasonably sure that the implementation is correct ...
- Check that the specification model is consistent with the system ore ... did we build the **right model**?
- Can an expert distinguish simulation output from system output?
  - Easily ... wrong model!
  - With a very expert eye ... good match!
  - Never ... you are probably cheating!

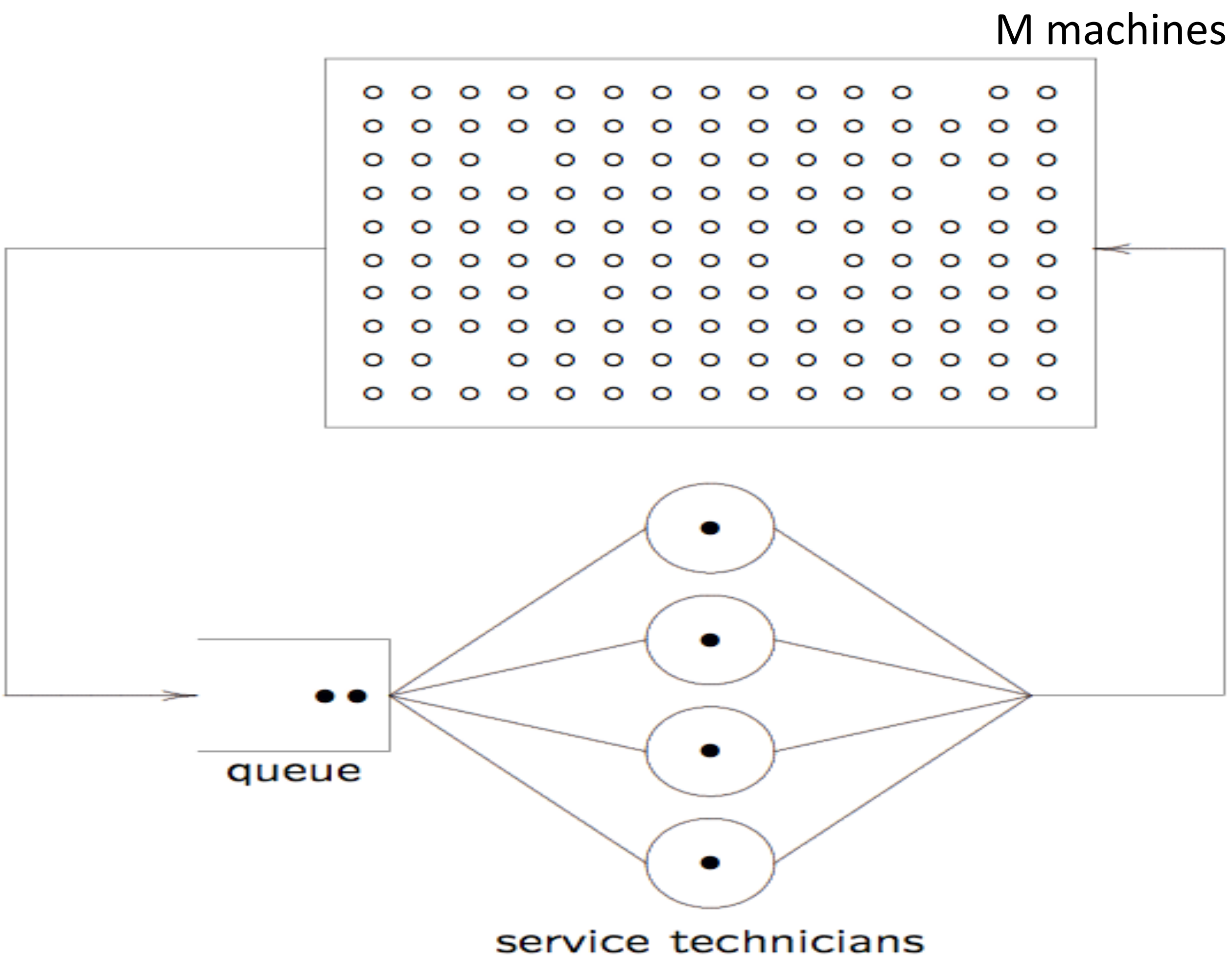
# Programming Languages

- General-purpose programming languages
  - Flexible and familiar (?!?)
  - Well suited for learning DES principles and techniques
  - C, C++, Java, pyton, ...
- Special-purpose simulation languages
  - Good for building models quickly
  - Provide built-in features (e.g., queue structures, mobility models, ... )
  - Normally “focused on a specific domain (networking, communication links, systems biology, hardware design, ...)
  - Graphics and animation provided (usually pretty bad)
  - Simula, Simulink, Omnet++, OPNET, SUMO, ...

# An example

- M identical machines (e.g., public laundry shops)
  - Operate at 60% of their time for 12 hr/day, 300 days/yr
  - Operation is independent, failure is independent
  - Repaired in the order of failure
  - Income: €10/hr of operation
- Service technicians:
  - Each works 1600 hr/yr, shifts of 8 hours for 200 days
  - Labor cost 40000 €10/year
- How many service technicians should be hired to maximize revenue?

# System Diagram



# Example: Steps to build a DES

1) Goals and Objectives
   - Find number of technicians and work organization for max profit
   - Extremes: one technician only, one technician per machine
2) Conceptual Model
   - State of each machine (failed, operational)
   - State of each techie (busy, idle, not on shift)
   - Provides a high-level description of the system at any time
3) Specification Model
   - What is known about time between failures?
   - What is the distribution of the repair times?
   - How will time evolution be simulated?

# Example: Steps to build a DES

4) Computational Model
   - Simulation clock data structure
   - Queue of failed machines
   - Organization of servicing technicians
5) Verification
   - Software engineering activity
   - Extensive testing
6) Validation
   - Is the computational model a good approximation of the actual laundry shops?
   - If operational, compare against the real thing
   - Otherwise, use *consistency checks*

# Hints and Observations

- Make each model as simple as possible
  - Never too simple ...
  - Check relevant characteristics
  - Do not add irrelevant details
- Model development is not sequential
  - Steps are often iterated
  - In a team setting, some steps will be in parallel
  - Do not mess up verification and validation
- Develop models with a clean separation of desing processes
  - Do not jump immediately to computational level
  - Think a little, program a lot (and poorly)
  - Think a lot, program a little (and well)

# DES Termination

- When should a simulation run terminate?
- Transients & Steady State
- How to measure the end of the transient
- Simulations may be too short
  - Our laundry machines never break ... so the best solution is not to hire any technician
- Or too long
  - We go beyond the lifetime of laundry machines ... or maybe of technicians life!!

# Good Simulation Studies

Once the simulator is ready, verified and validated ...

1. Design simulation experiments
   - What parameters should be varied?
   - Perhaps many combinatorial possibilities
2. Make production runs
   - Record initial conditions, input parameters, generator seed
   - Record statistical output
3. Analyze the output
   - Use common statistical analysis tools
4. Document the results
5. Make decisions (if it is your duty)

# Good Simulation Studies

1. Design Experiments
   - Vary the number of technicians
   - What are the initial conditions?
   - How many simulation replica do you need to have reliable results?
2. Make Production Runs
   - Manage output wisely
   - Must be able to reproduce results *exactly (memorize the seed)*
3. Analyze Output
   - Observations are often correlated (not independent)
   - Take care not to derive erroneous conclusions

4. Document Results
   - System diagram
   - Assumptions about failure and repair rates
   - Description of specification model
   - Software used (description, code repository)
   - Tables or figures of output (remember captions)
   - Output analysis and comments
5. Make Decisions
   - The output gives the optimal number of technicians and its sensitivity
   - Take additional constraints into account (e.g., illness of technicians)

# Time modelling and management

- DES allows time compression-expansion
- In general fixed sampling is less efficient
- With fixed sampling a good idea can be to use an integer for time increments
- Beware of possible time warping
- Events are ordered and executed in time
- Time coincidence is a problem
  - Provide for time collision management
- Generate events “as they happen” and not a-priori
- Good/bad examples based on M/M/1

# List of events

- Events drive the simulation
- When generated they must be ordered → Events List
- A long list is highly inefficient, if possible keep the list short
  - Some (rare) cases are inherently ordered (single queues)
- If the list is unavoidably long may be worth using a heap or tree, but balancing is costly
- A vector of lists is a dirty-but-efficient solution

# Actions vs Events

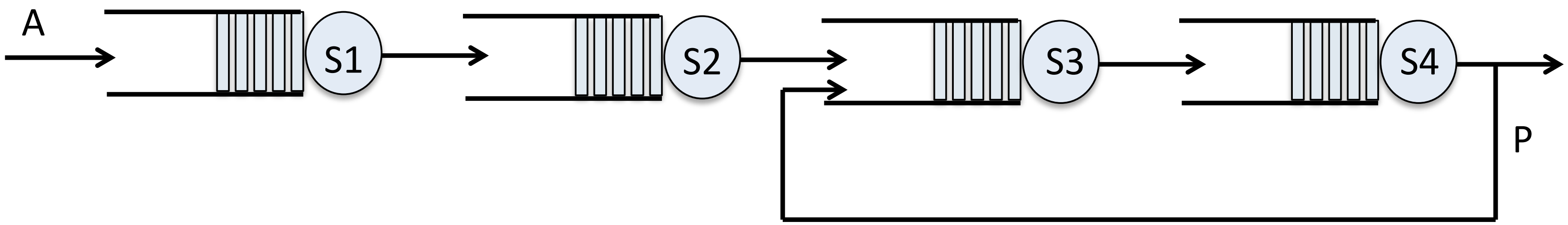
- Events imply Actions (by the system)
- Actions can be complex & computationally heavy
  - E.g., compute the next state in a meteorological simulation
- ... or very simple
  - E.g., queuing de-queuing packets
- This define two types of simulations
  - The first dominated by actions computation
  - The second dominated by events management (see list of events)

# Simulating with objects

- Whether using an OO language or not, often  a system is described by a collection of objects
  - Stations in a networks
  - Peers in a distributed system
  - Virtual Machines in a data-center
  - ...
- Sometimes Objects are volatile
- Be careful in allocation – de-allocation
- Often useful to use a pool of objects and not allocating – de-allocating
  - E.g., Packets in the Internet

# DES: an Example

- Consider the following queuing network

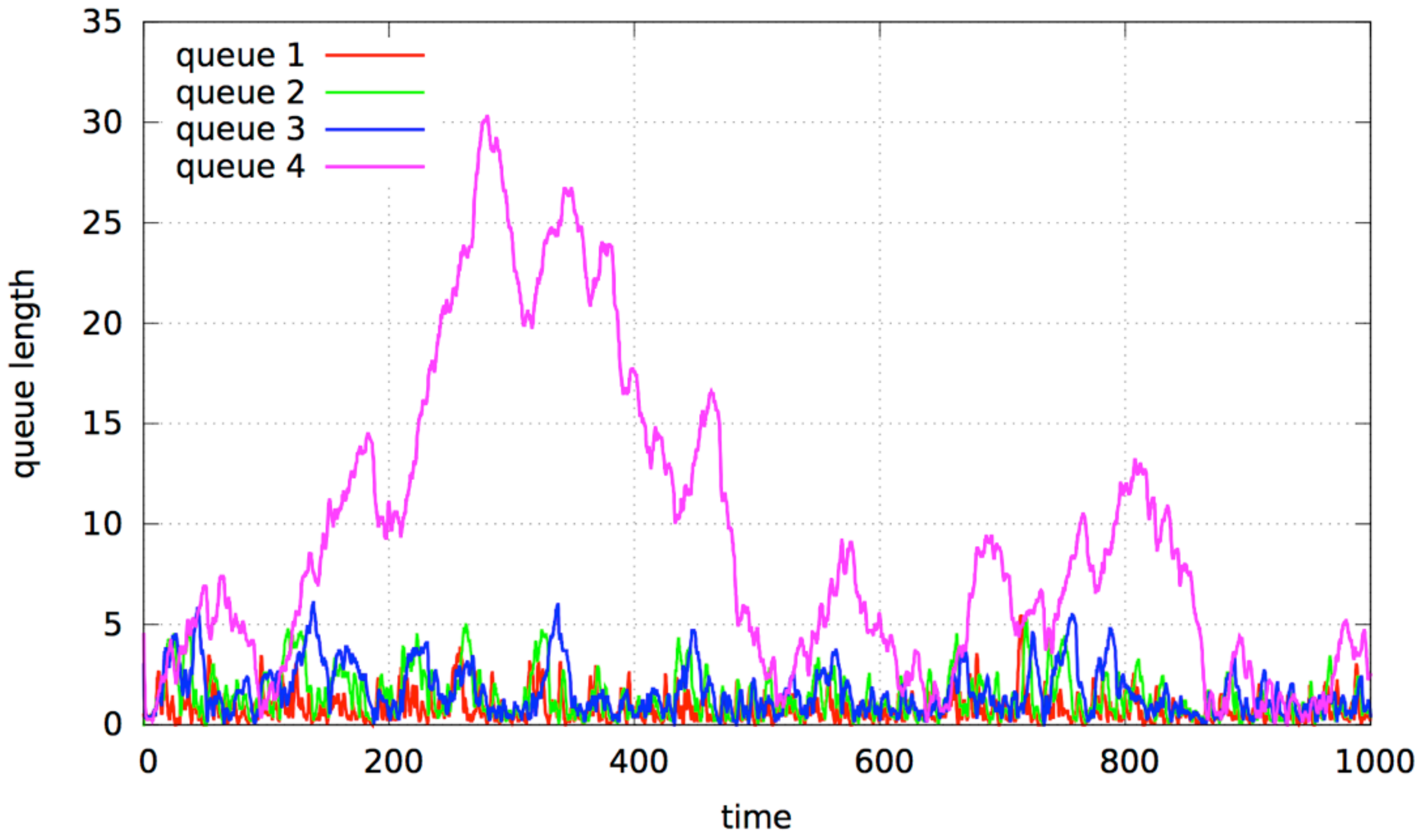



- A is Poisson with l = 1 (normalization ... who cares)
- Si are U(0.5Ti,1.5Ti)
- T1=0.5, T2=0.7, T3=0.8, T4=0.95
- P = 0.0, 0.04, 0.06
- Simulate the queue network & evaluate the transient period

# DES: an Example

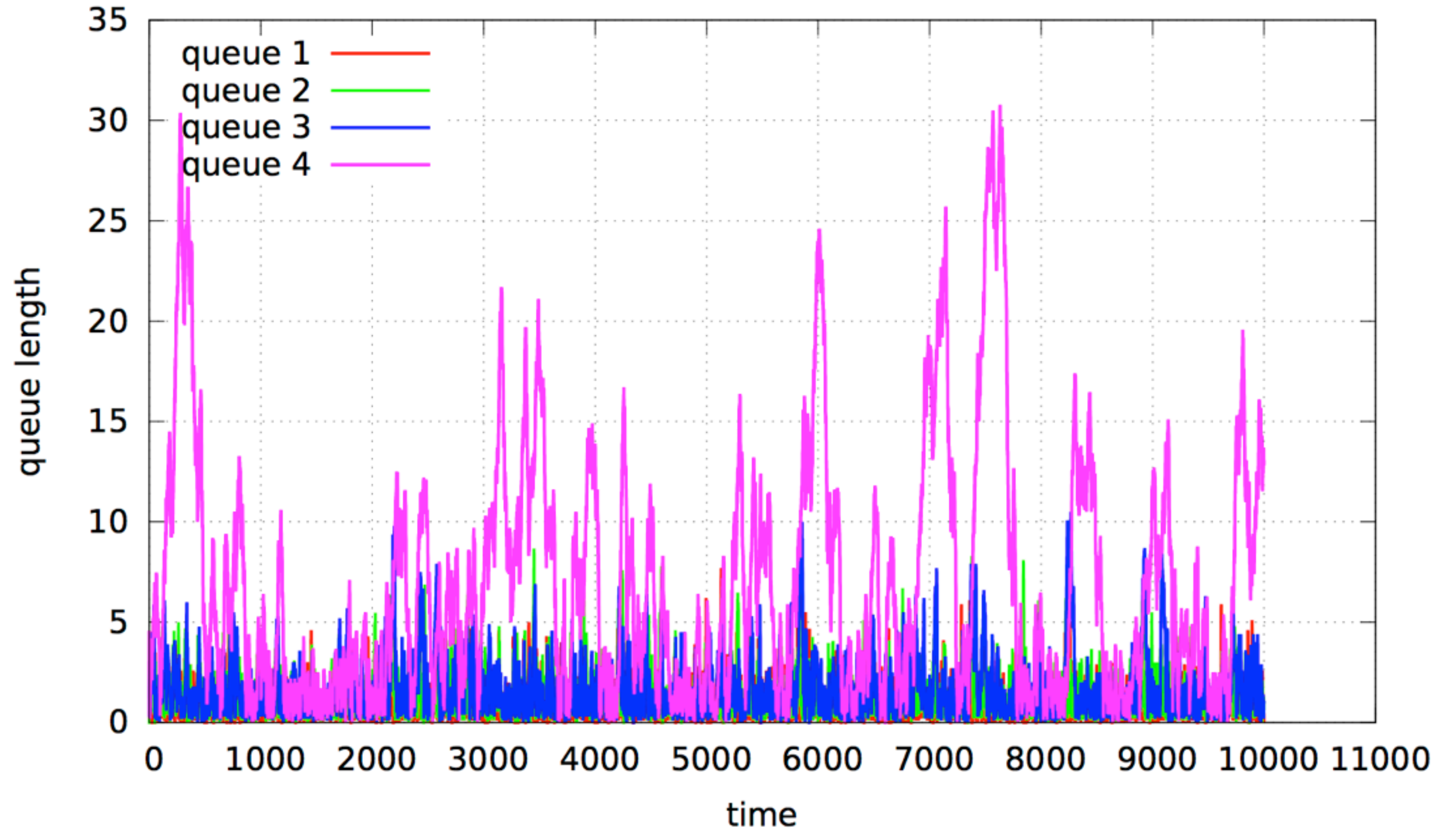
- Simulate the queue network & evaluate the transient period
- A quick&dirty pyton simulator is available on the course web site
- Graphics are obtained with gnuplot (script also available)
- We simply "observe" the output of a realization (number of customers in each queue)
- Results are smoothed with a sliding window of size 10

- Is Q4 at steady state?

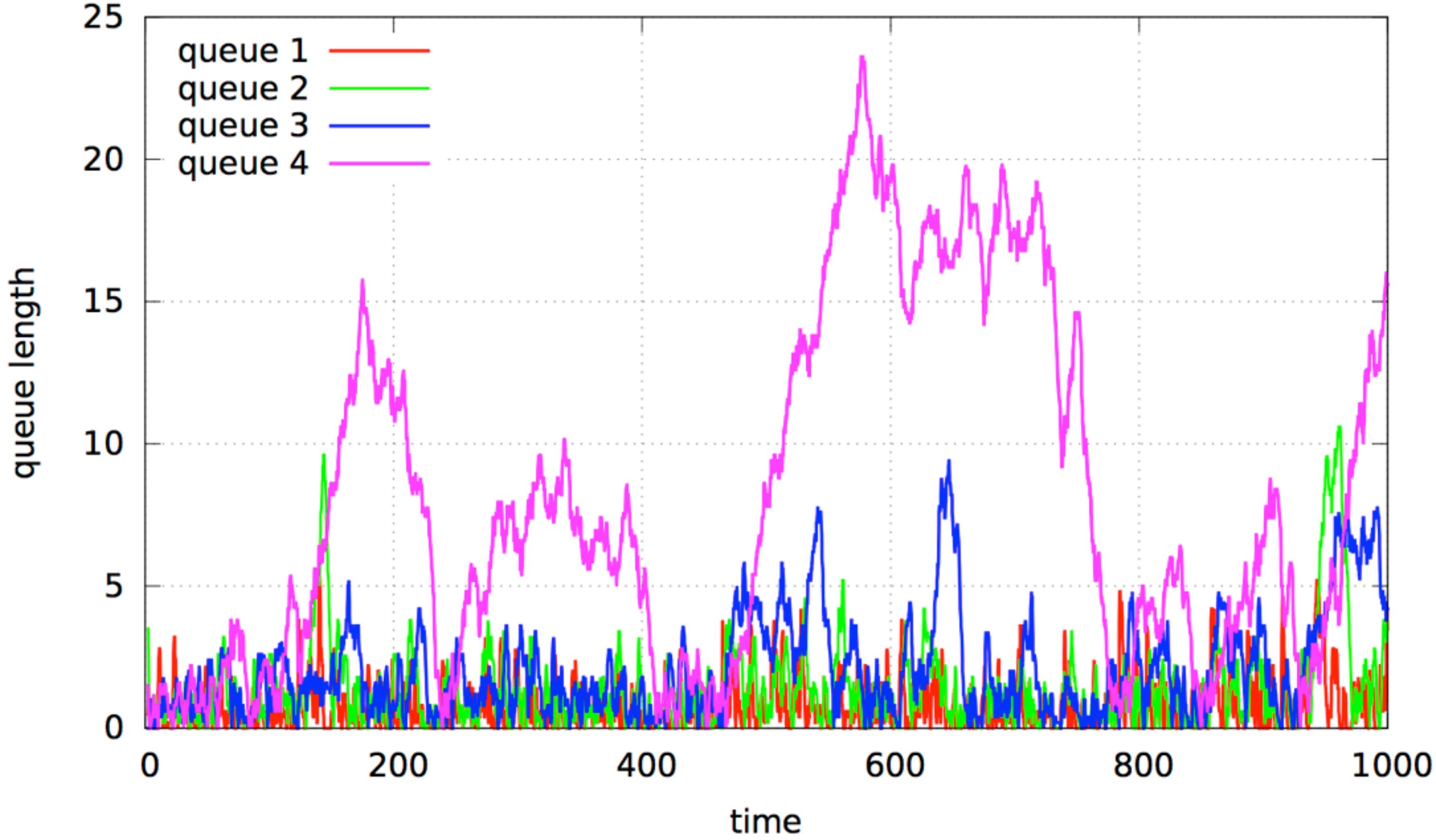


# P=0.0 ; T = 10000

- ... probably yes after 1-2000, but not really a quantitative answer

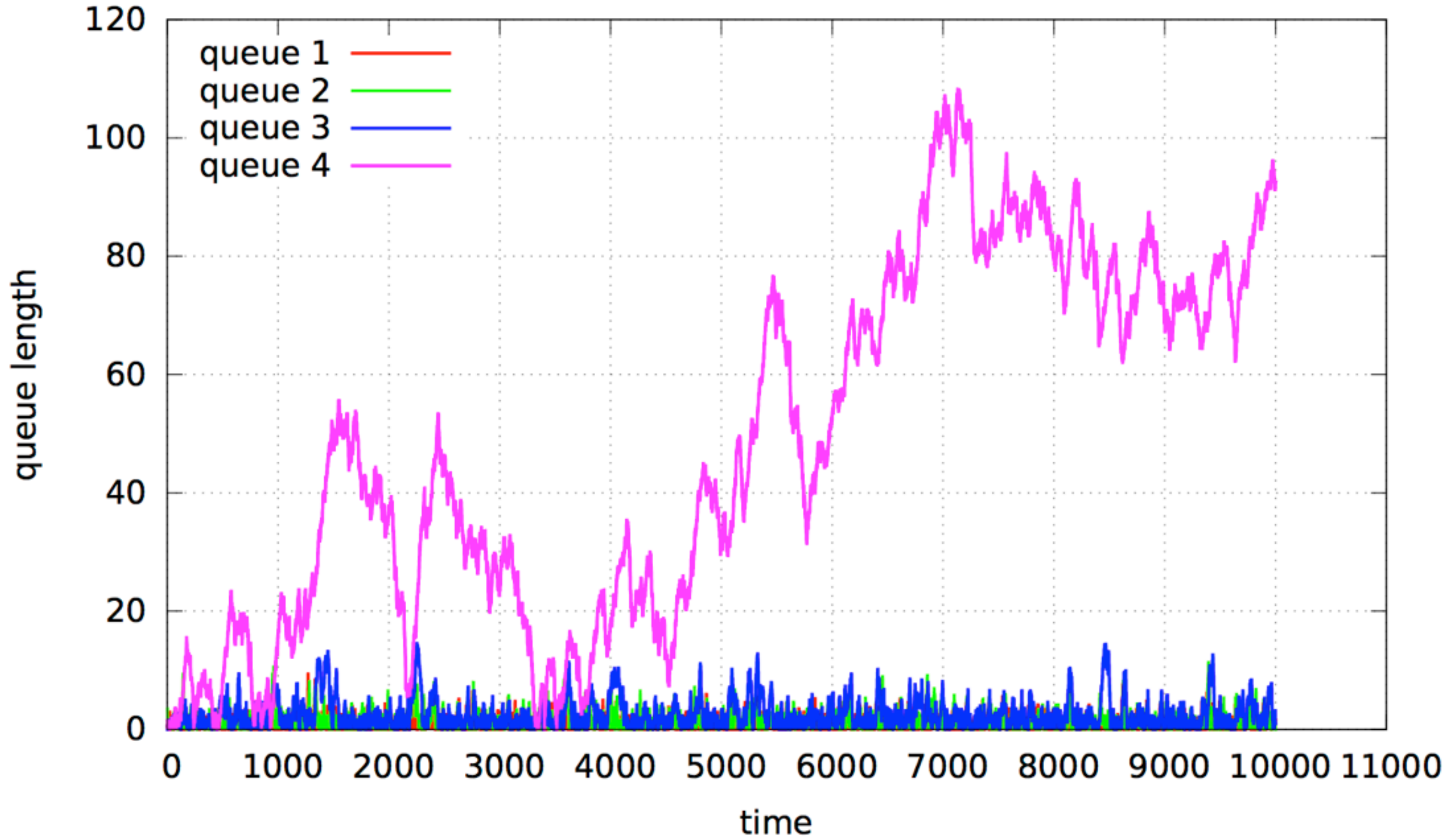


- Is Q4 at steady state?



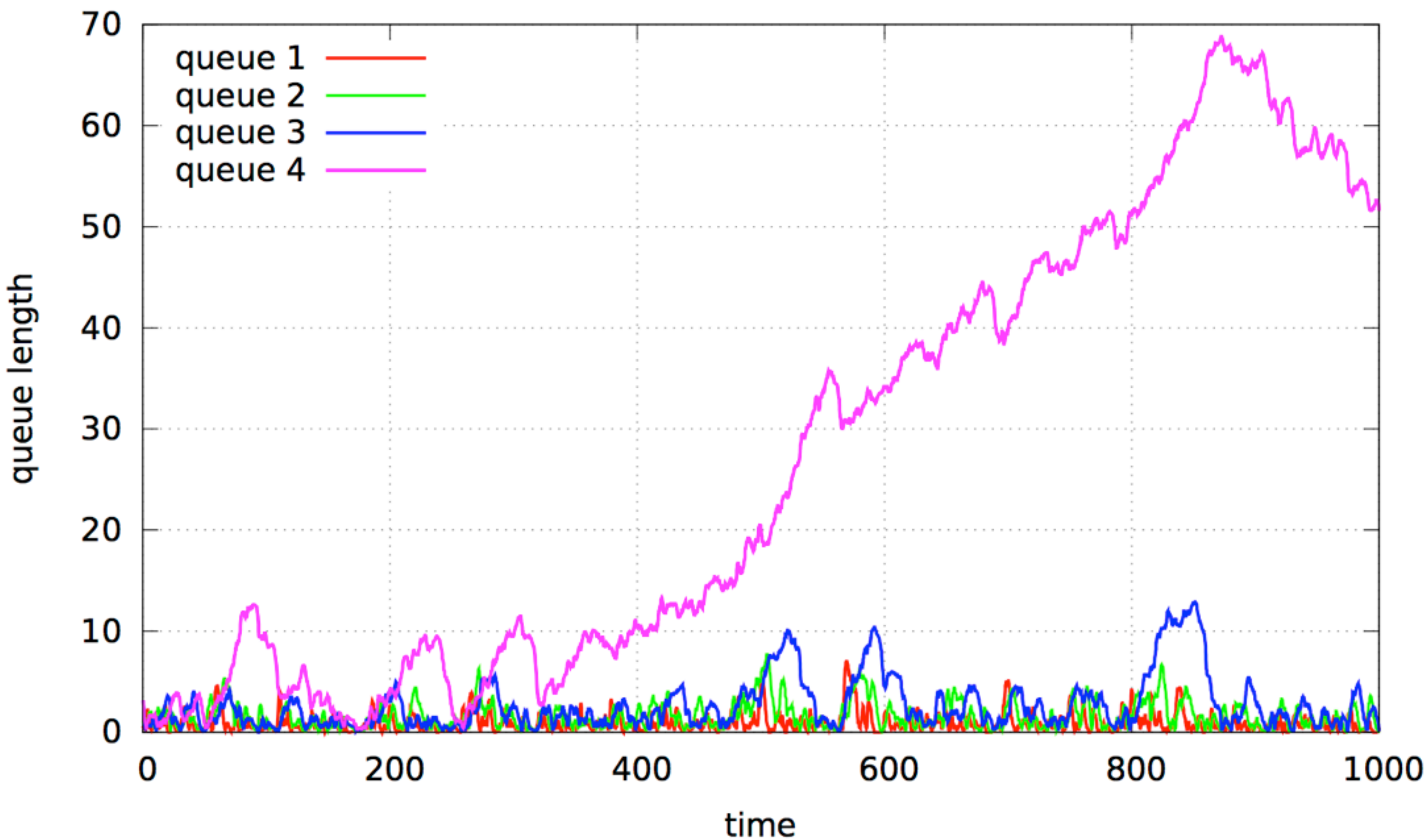
# P=0.04 ; T = 10000

- ... more difficult to say, but definitely not before 8000

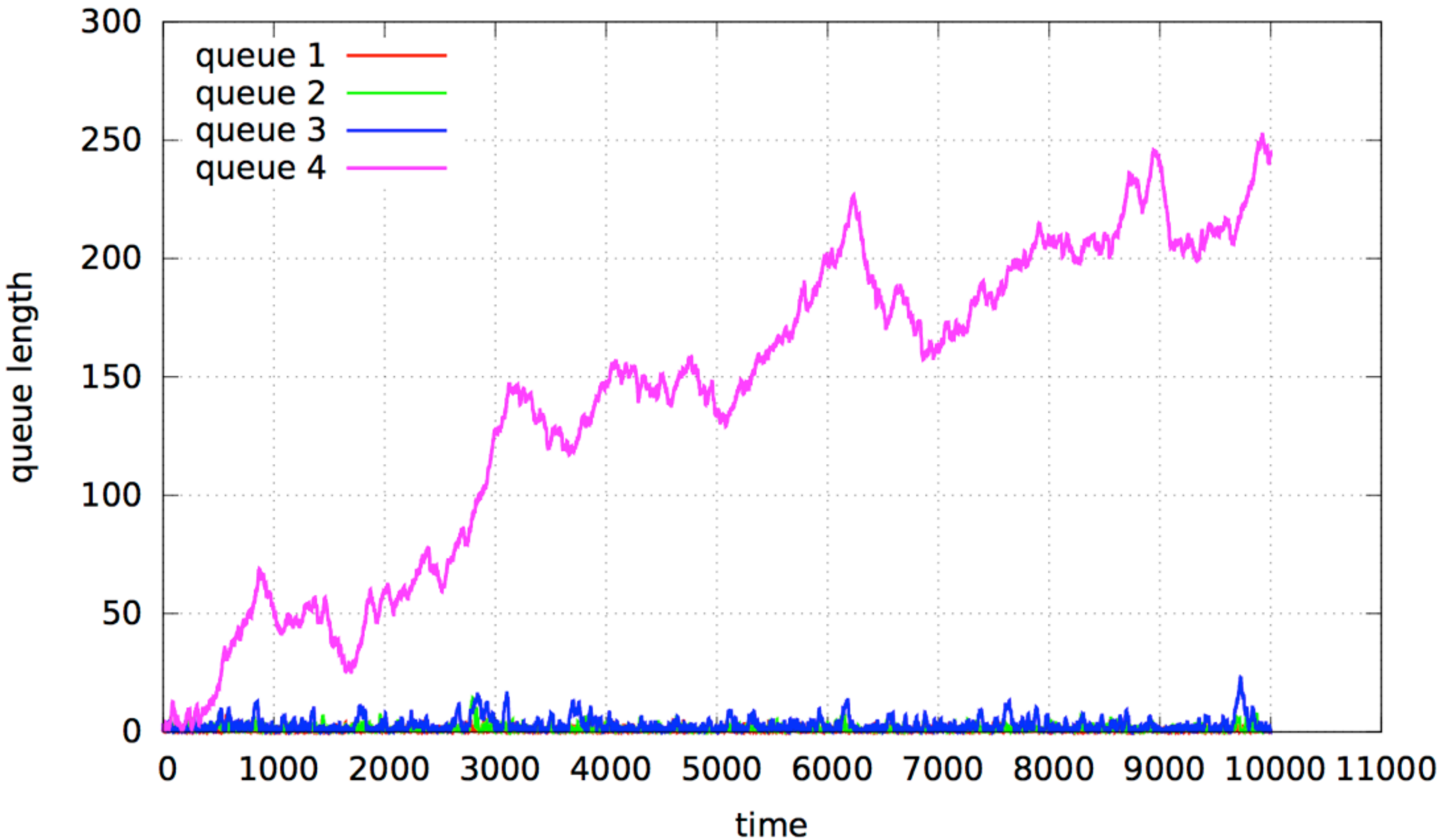


# P=0.06 ; T = 1000

- Is Q4 at steady state?



- ... probably not, when will it reach stability?



# Takeaway

- If you have conceptual tools to evaluate the feasibility of a simulation beforehand, use them
- Transient estimation may be very difficult observing outputs
- Unlimited parameters, measures, variables may be dangerous, avoid them if possible
- Outliers (in code, not seen here) must NEVER be ignored, if something not foreseen happen, record it and possibly stop the simulation (error log)

# Homework

- Modify the simulation program (available on the web site) to introduce finite queues and measure loss rates on the different queues
- Concentrate on the size of queues 3 and 4
  - 1 and 2 a lightly loaded, their size is irrelevant unless you set it very small ... which can be interesting in any case

# Assignment 2

- Simulation of a queuing network
- Common generic structure customized for each student
- The text is already available, customization will follow in the next few days together with moodle setup
- Deadline for “early” delivery: 30 June, or 1 week before taking the oral exam