ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Network Protocol Design and Evaluation

Exercise 7

Stefan Rührup

University of Freiburg
Computer Networks and Telematics

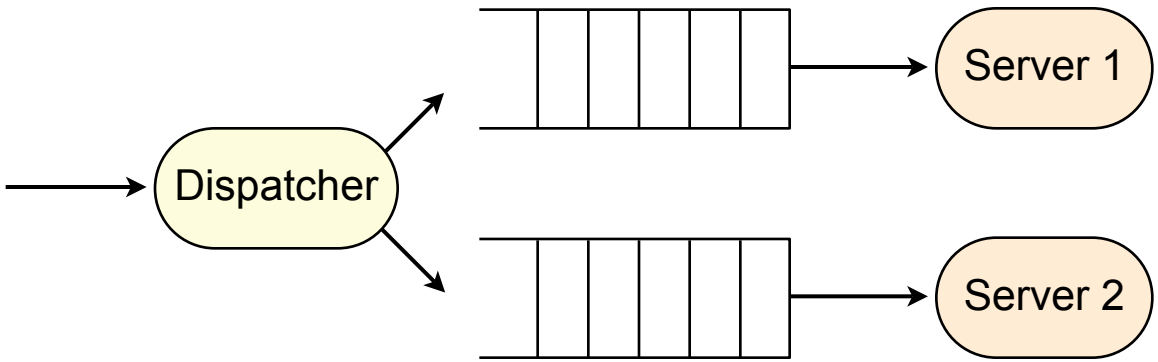
Summer 2009

Task 1

Task 1 *Simulating a Queue*

1. Write a simulator for a single queuing system (M/M/1) using an object oriented design as described in the lecture. Uses classes for simulator, server and an event dispatcher.
2. Extend your program so that tasks are handled by two servers with a queue for each server. The dispatcher should assign the task to the server with fewer tasks in the queue.
3. Run simulations for the following parameters: Mean inter-arrival time = 1; mean service time = 0.3, 0.5, 0.7, 0.9. Execute 10 simulation runs for each parameter set with 1000 tasks each.
4. Record statistics for average queue length, average waiting time and server utilization and plot them using error bars (showing average and standard error) or box-and-whiskers.

The Queueing System



The Implementation (1)

- Simulator contains the main event processing loop
- Events are stored in a priority queue
- Event dispatcher class contains event handler for arrival and departure
- Servers are separate classes
  - here, each server has its own queue
  - statistics are generated in each server

- **Example source code available on the website**

The Implementation (2)

- Main loop in the Simulator class

```
// Generate first event:
eventSet.add( new ArrivalEvent( getNextArrival() ) );
taskCount++;

// Main loop:
while (!eventSet.isEmpty() && taskCount < maxNumberOfTasks)
{
    Event e = eventSet.poll();
    dispatcher.handleEvent(e);
}
```

The Implementation (3)

- Event handler in the Dispatcher class

```
class Dispatcher {

    PriorityQueue<Event> eventSet;

    public void handleEvent(Event e) {
            Simulation.setSimTime(e.getScheduledTime());

        if (e instanceof ArrivalEvent)
                handleArrival( (ArrivalEvent)e );
            else
                handleDeparture( (DepartureEvent)e );
    }

...
```

Simulation results

- **Data recorded for Server 2**

  mean service time = 0.5

Run	Utilization	Waiting time	Queue length
1	0.21703612847854223	0.04883240387208119	0.029898866407067848
2	0.237063270258235	0.04803044608724357	0.026543717634252968
3	0.23855649855366426	0.03529710852369007	0.022069544314794604
4	0.24457648784737668	0.04224992019873808	0.026441243833361355
5	0.2453421166831905	0.05282802540522634	0.031103260712944752
6	0.2461801656983352	0.042392712283781796	0.02533025992393397
7	0.2472367286672406	0.04052182964103343	0.023303719691358738
8	0.2494853501387802	0.035308921300164296	0.018346280678450656
9	0.255897933029883	0.04940020932138992	0.029408961590068415
10	0.2646494566989262	0.0555753173876974	0.0329643488213804

→ sort and extract min/max, quartiles

Processing data

- **Data aggregation:** Average and deviation
- Deviation:
  - Sample standard deviation

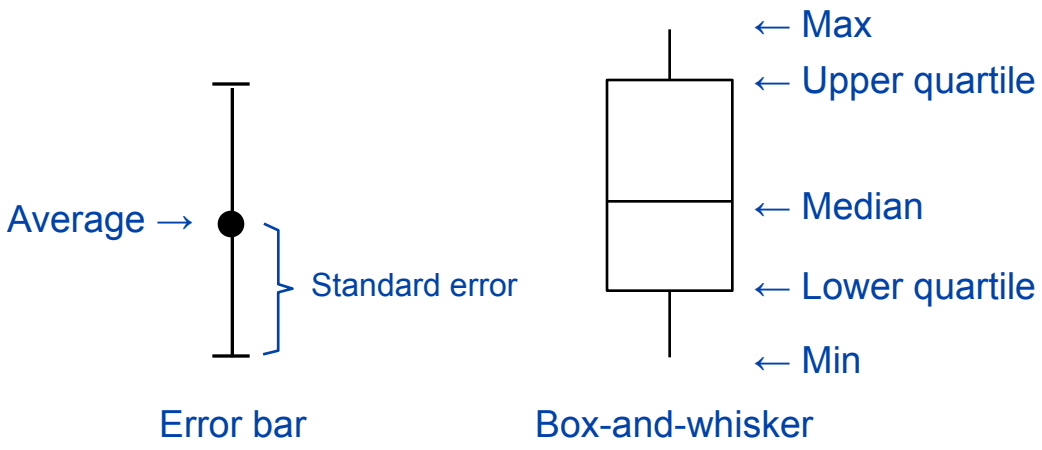
$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2},$$

  - Standard error:

$$SE_{\bar{x}} = \frac{s}{\sqrt{n}}$$

Presenting Results

- Aggregated data (e.g. average) gives a concise description, but deviations are not visible
- Ways to include deviation
  - Error bars (show standard error)
  - Box-and-whisker plot (show quartiles)



Utilization diagram