Pipes
Discrete Event Simulator Implementation in Java

Razvan Surdulescu

CS380n, Spring 2003

Part 2

Low-Level Design

Results

Challenges

Future Work

Low-Level Design

- Design areas
  - Simulation
    - How do you simulate transactions going through a system?
  - Statistics
    - How do you simulate different statistical distributions?
    - How do you gather and compute performance measures?

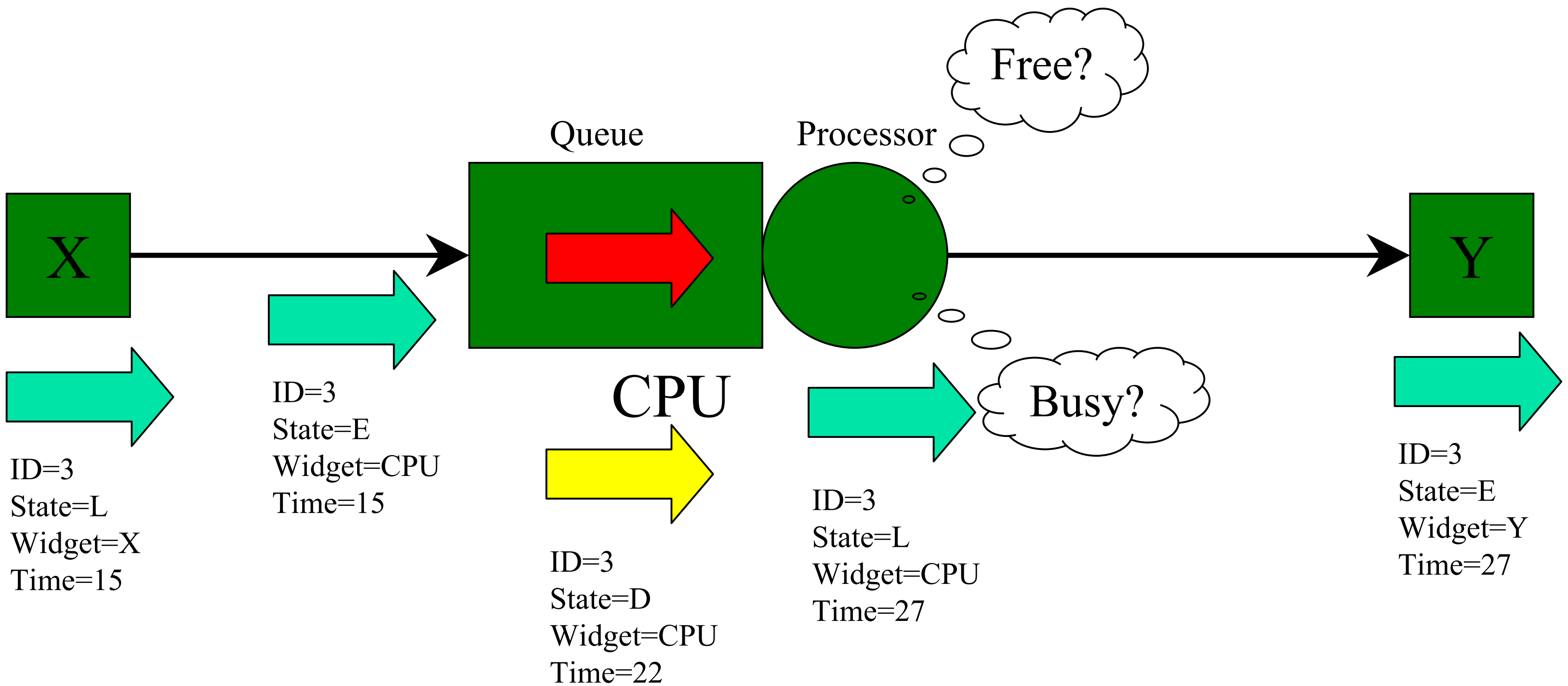
Low-Level Design

- Design areas
  - Architecture
    - How do you build a DES in an extensible manner?

Low-Level Design cont'd

- Simulation design
  - Event based simulation
    - Events are characterized by:
      - Transaction ID (unique), transaction category, transaction priority
      - Time
      - State (E(ntering), D(equeued), L(eaving))
      - Widget

Low-Level Design cont'd



Low-Level Design cont'd

- Simulation design cont'd
  - Events queue
    - Events are ordered by "Time"
  - Simulator controls
    - Start(): starts the simulation
    - Step(): performs one step in the simulation
    - End(): ends the simulation

Low-Level Design cont'd

- Simulation design cont'd
  - Start()
    - Reset all the statistics measures on the model
    - Creates the transactions for all Sink nodes
    - Each transaction is represented by an event in (state=E, time=0)
    - All events are added to the queue

Low-Level Design cont'd

- Simulation design cont'd
  - Step()
    - Remove the event at the top of the global queue
    - Advance it via the owner node: each node advances events differently
    - A node could temporarily remove an event from the global queue and store it in the node queue (for example, if the node is busy processing another event)

Low-Level Design cont'd

- Simulation design cont'd
  - Step() example: Service node
    - If current event state = E or state = D
      - If the node is busy processing another event, remove current event from the global queue and store it in the node queue
      - Otherwise mark node as busy, create a new event with (time = node processing time, state = L)
    - If current event state = L
      - Mark node as free, move events from the node queue into the global queue
      - Pick the appropriate outgoing arc and create a new event with (state = E, widget = neighbor)

Low-Level Design cont'd

- Simulation design cont'd
  - End()
    - Compute all the statistics measures on the model

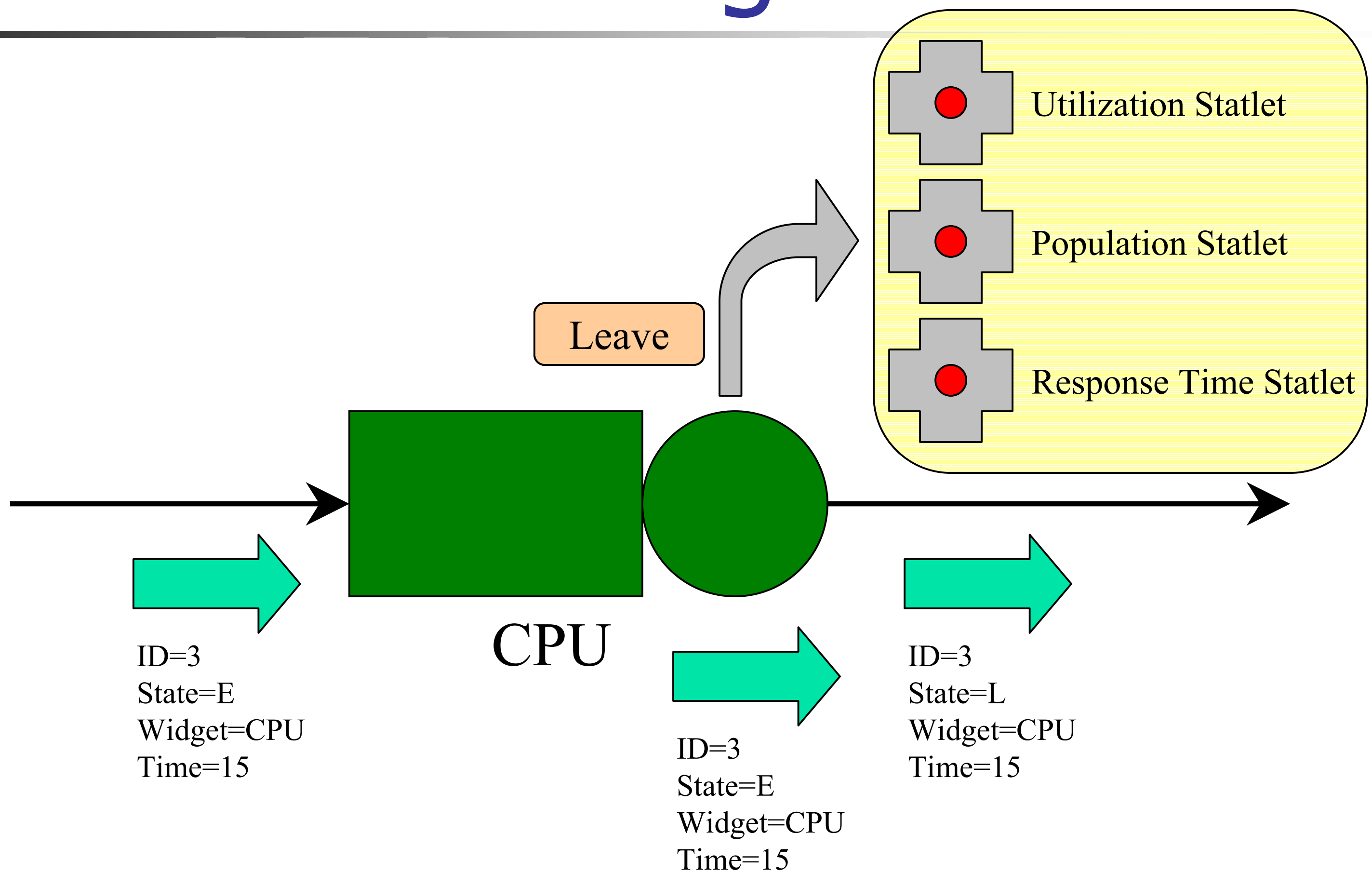
Low-Level Design cont'd

- Statistics design
  - Supported distributions
    - Fixed(a) = a
    - Uniform(a,b) = uniform distribution in the interval [a, b)
    - Normal(a,b) = normal distribution with mean a and stdev b
    - Negative Exponential(a): negative exponential distribution with mean 1/a
    - Poisson(a): Poisson distribution with parameter a
    - Erlang(a,b): Erlang distribution with mean 1/a and stdev 1/(a* sqrt(b))

Low-Level Design cont'd

- Performance gathering
  - Each node advances events
    - In the process of advancing an event (e.g. from state=E to state=L), it reports performance events
  - Each node has 0 or more *statlets* (a performance event listener)
    - There is one type of statlet for each supported performance measure (arrival rate, population, queue population, utilization, response rate, response time)

Low-Level Design cont'd



Low-Level Design cont'd

- Performance computation
  - Population queue statlet example
    - Listen to "enqueue", "dequeue", "end"
    - Keep a population count
    - On "enqueue", increase population count, store pair (event time, current population)
    - On "dequeue" decrease population count, store pair (event time, current population)
    - All pairs are ordered by event time

Low-Level Design cont'd

- Performance computation cont'd
  - Population queue statlet example cont'd
    - To compute the average queue size, let `sum = 0`, iterate over all stored pairs
    - `sum = sum + (prevPair.population * (currPair.time - prevPair.time))`
    - Return (`sum / totalTime`)

Low-Level Design cont'd

- Extensible design
  - MVC pattern used to separate model from UI
    - Model can be automated without having a UI
  - Widget template
    - Base Widget class defines common node features
    - Derived Widget classes define
      - Default name
      - Number and position of Connectors
      - How to advance an event through this widget

Low-Level Design cont'd

- Extensible design cont'd
  - Widget template cont'd
    - Derived Widget classes specialize
      - The customizable properties of the widget
        - The property list is displayed in a form for the user to modify
      - The Visitor entry point
        - Widgets are saved and loaded via a Visitor
      - For Widgets with a queue, specify how to select an event from the widget queue

Low-Level Design cont’d

- Extensible design cont’d
  - Statlets
    - Covered in previous slides
  - Automation
    - Models can be created and executed programmatically

Demo

- Automation: the car wash problem source

```
Model model = new Model();

WidgetSource cars = new WidgetSource(model);
cars.setName("Cars");
cars.setDistributionName(Distributions.NEGATIVEEXPONENTIAL);
cars.setDistributionValueA(11);
cars.setNumberOfTransactions(2000);
model.addWidget(cars);
...
Simulator simulator = new Simulator(model);

simulator.start();
while (!simulator.isFinished()) {
    simulator.step();
}
simulator.end();

System.out.println(allocate.getStatistics());
```

Demo

- Automation: the car wash problem output

```
C:\>java -classpath Pipes.jar;... edu.utexas.cs.surdules.pipes.demo.CarWashProblem
Statistics for widget 'Allocate':
QueuePopulationStatlet:
    sum=104596.49964194975,end=21779.64605225117,average=4.802488497334351
ResponseTimeStatlet: count=2000,think=104596.49964194951,average=52.29824982097476
```

Results

- Simulation accuracy
  - Pipes produces accurate results
    - Comparable to Workbench for similar models
- Software quality
  - Pipes is extensible, versatile and small
  - Pipes has a modern UI and an interoperable file-format
- Didactic effectiveness
  - I got a pretty solid grasp of DES concepts

Challenges

- DES approach
  - Event based vs. time based vs. mix?
- Widget and Statlet set
  - What is a minimal, useful set of Widgets and Statlets for modeling interesting problems?
- Software design
  - Extensible and easy to understand

Future Work

- Add Widget types
  - Transaction fork, Transaction join, Loop
- Add Widget features
  - Polling queuing priority
  - Round robin time rule
- Add statlets
  - Queue response, Quantity
  - Compute standard deviation, variance

Questions, Comments?