# CNCL: Contents

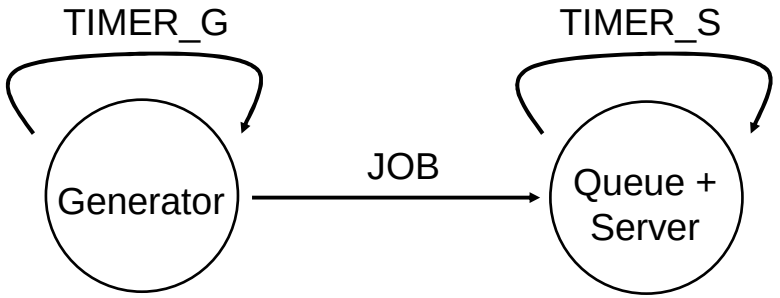
- CNCL – C++ library for supporting event driven simulations
- Learning CNCL by examples
  - Example 1: GI/GI/1 system, combined queue and server
  - Example 2: steady state simulation using independent runs
  - Example 3: GI/GI/1 system, separate queue and server
- CNCL project work instructions

# Extra material

- All code examples referred to in this lecture available from
  - http://www.netlab.hut.fi/opetus/s38148/
- Available files are: Makefile, mm1v1.c, mm1v2.c, mm1v3.c
- Usage
  - Copy files to a directory
  - Create a configuration file “use.cncl” with the following line:
    - Tcsh users:
      setenv LD_LIBRARY_PATH /usr/lib:/usr/local/lib”
    - Bash users:
      LD_LIBRARTY_PATH=$LD_LIBRARY_PATH: /usr/lib:/usr/local/lib
      export LD_LIBRARY_PATH
  - In the directory, set up your paths by writing “source use.cncl”
  - In the directory, create a “.depend”-file by writing “touch .depend”
  - Run “make”

# CNCL: modeling the GI/GI/1 system

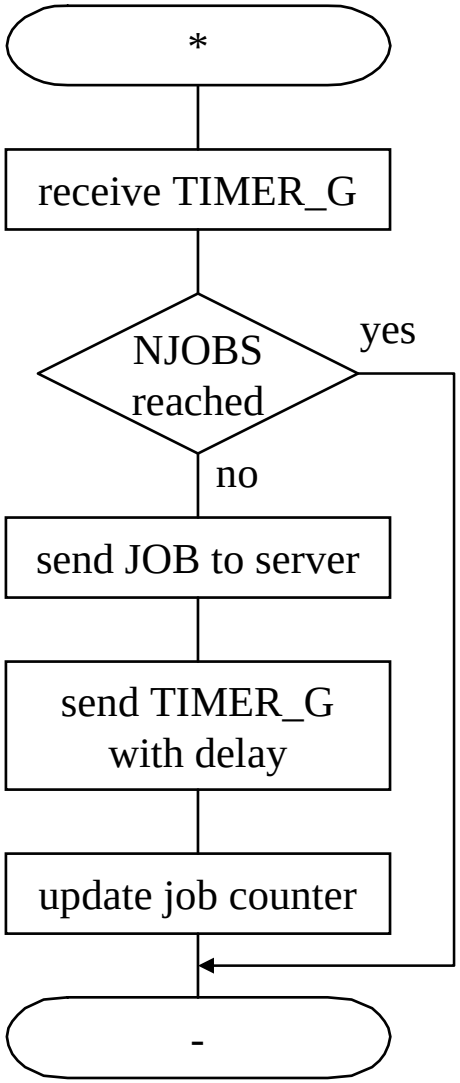
- Modeling packet arrivals easy
  - packet arrival times are independent of each other
  - generator only needs to send a packet every *interval* time units, where *interval* is a random variable with a given (general) distribution
- Modeling queue/server
  - in the simple GI/GI/1 system the queuing discipline is just FIFO, so server does not have any real functionality => queue and server can be in the same process
- Event handlers:
  - generator
  - queue + server
- Three event types
  - TIMER_G: a new job is generated
  - JOB: generator sends a job to server
  - TIMER_S: server is free to take a new job from queue



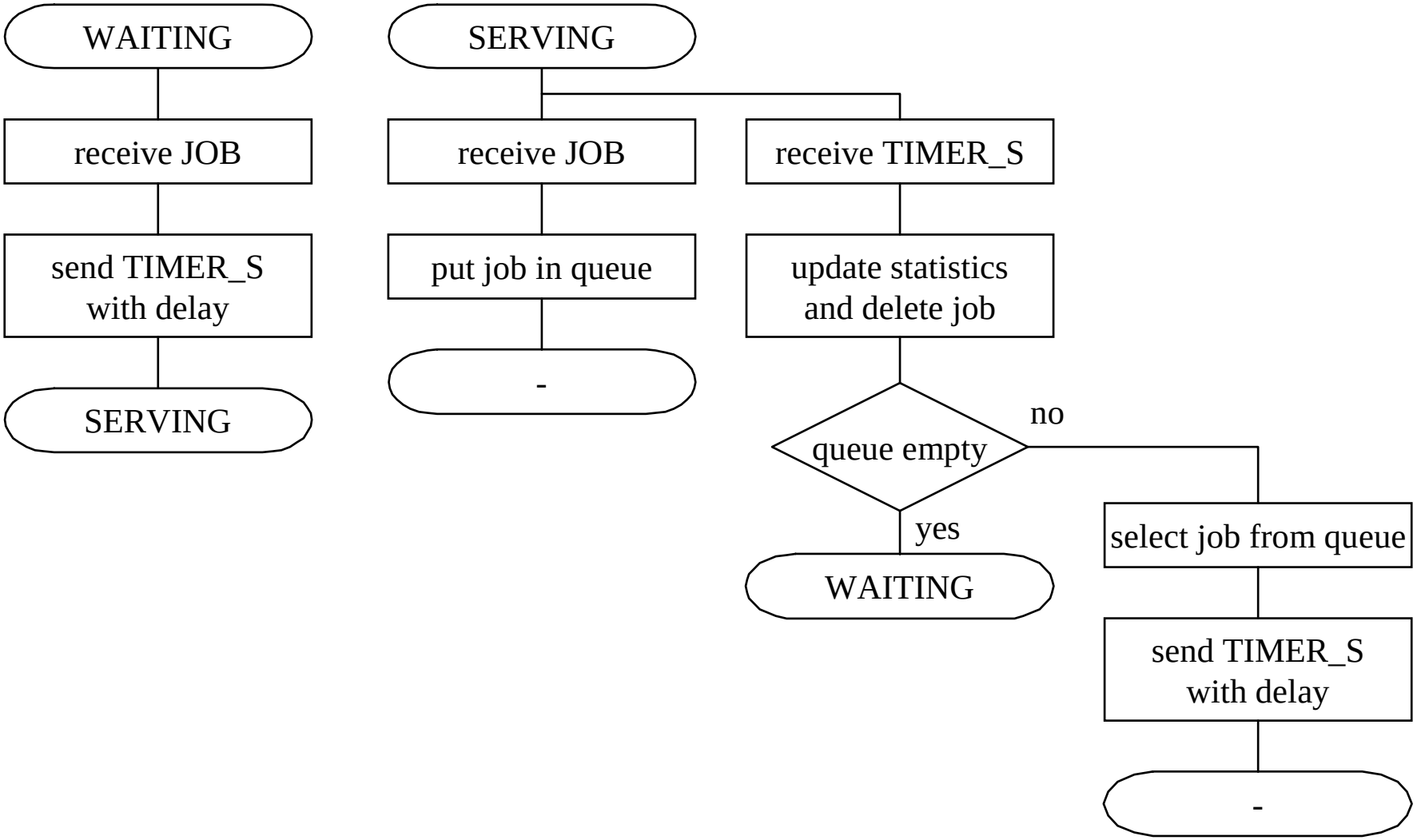
# Example 1

- Basic GI/GI/1 functionality
  - with Poisson arrivals and exponential service times
- Statistics collection
  - queuing delay
  - sojourn times
- Contains initial (and final) transient
  - simulates fixed nof packets starting from an empty system until last packet has been served
- Example code:
  - mm1v1.c

# Flow chart (A) / generator



## Flow chart (A) / server



# CNCL: Contents

- CNCL – C++ library for supporting event driven simulations
- Learning CNCL by examples
  - Example 1: GI/GI/1 system, combined queue and server
  - Example 2: steady state simulation using independent runs
  - Example 3: GI/GI/1 system, separate queue and server
- CNCL project work instructions

# Example 2

- Adding functionality to the basic GI/GI/1 example
- Aim
  - steady state simulation of mean sojourn times as a function of offered load
  - does not affect the overall model of the system (i.e., the process model)
- Statistics collection
  - initial (and final) transient removal
  - statistics output to a file (“out.dat”)
  - comparing simulations and analytical results in Matlab (M/M/1 mean delay = $1/(\mu - \lambda)$)
- Example code:
  - mm1v2.c

# Example 2

- Discussion:
  - how do you change the code (state machine) such that the queue size is finite (GI/GI/1/K-system)?
  - how is packet loss probability measured/estimated?

- Answers:
  - buffer size K packets
  - need to add a new state variable to class Server: N, number of packets in the system
  - upon arrival (event JOB) the state variable is checked (N = K), and if true packet is discarded
  - to estimate packet loss probability, only a count of lost packets is needed (and the number of arrived packets)
  - again, need to take care of starting measurements only after initial transient

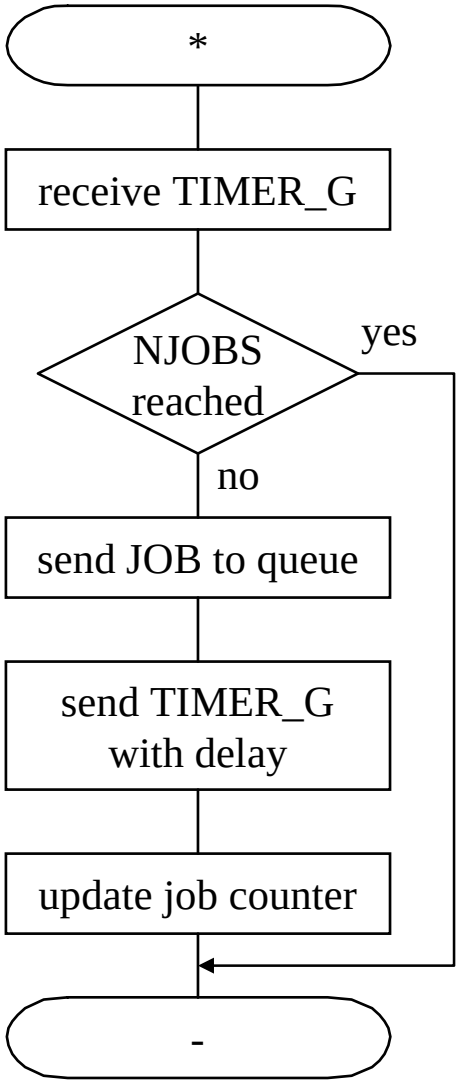
# CNCL: Contents

- CNCL – C++ library for supporting event driven simulations
- Learning CNCL by examples
  - Example 1: GI/GI/1 system, combined queue and server
  - Example 2: steady state simulation using independent runs
  - Example 3: GI/GI/1 system, separate queue and server
- CNCL project work instructions

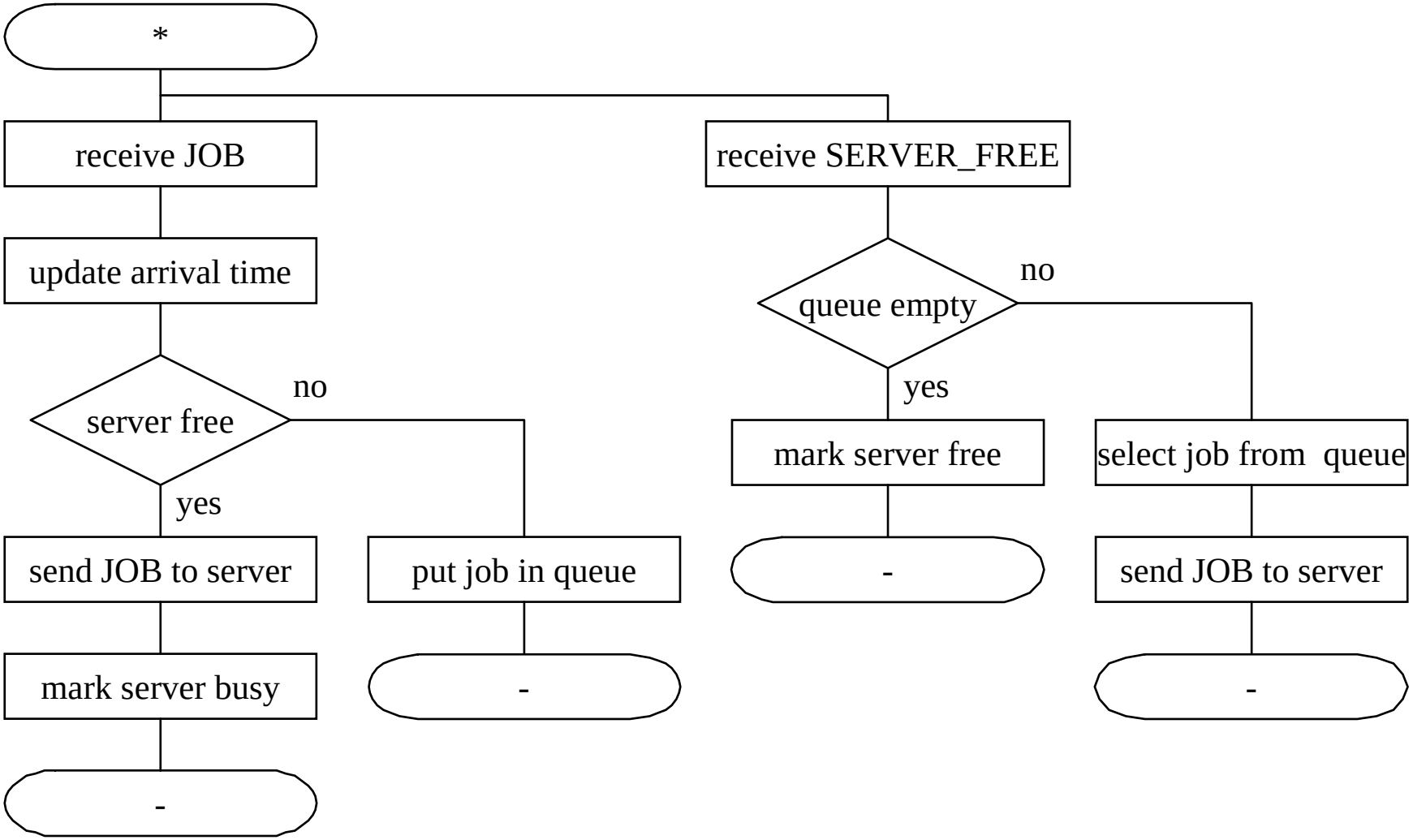
# Example 3

- Basic GI/GI/1 example
- Separating the queue and the server from each other
  - e.g., if we have a queuing system with multiple queues and a single server with an advanced scheduling algorithm
- Example code:
  - mm1v3.c

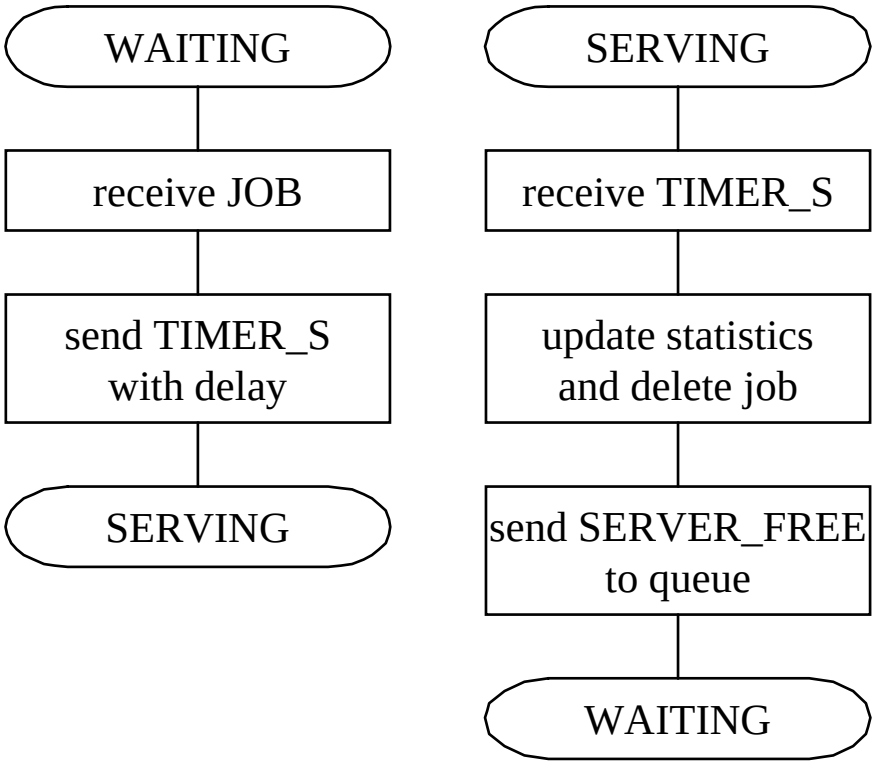
# Flow chart (B) / generator



# Flow chart (B) / queue



# Flow chart (B) / server



# Example 3

- Discussion:
  - how do you change the code (state machine) to simulate a 2 class queue with advanced scheduling?
- Answers:
  - 2 classes => 2 instances of Generators, 2 instances of queues
    - Stopping rule at generator does not anymore make sense
  - Scheduler implemented in server, so some changes need to be made to current design
  - Changes in Queue-class
    - Server needs to know if Queue1 or Queue2 is empty (methods must be added to Queue class)
    - Queue class can not have local info about Server status (why?)
    - Upon receipt of SERVER_FREE, queue does not anymore check if queue is empty or not (functionality moved to server)
  - Changes in Server-class
    - Must have a server_status()-method
    - Upon receipt of JOB-event, the delay of the event TIMER_S may depend on the class of the job
    - Upon receipt of TIMER_S event,
      - if class 1 queue is non-empty, sends SERVER_FREE to Queue1
      - if class 1 is empty and 2 non-empty, sends SERVER_FREE to Queue2
      - If both queue are empty, server state changes to WAITING

# Example 3 : DiffServ router

- DiffServ QoS architecture
  - Traffic grouped into N different classes
  - Objective: relative delay differentiation across different time scales, e.g., using WTP scheduler
  - Packets are classified on the edge
  - In the core network, packets are routed simply based on class
  - Each router implements a queuing block as below/output port