A queueing network with SMPL

Chang-Gun Lee (cglee@snu.ac.kr)

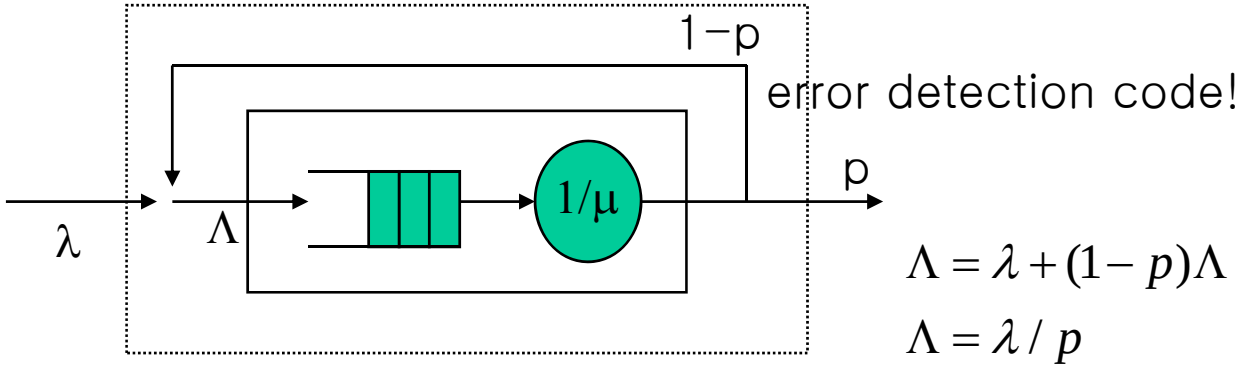
Assistant Professor

The School of Computer Science and Engineering

Seoul National University

Example

- Error recovery transmission system



$\lambda = 4$ messages per second

$1/\mu = 0.22$ sec

$p = 0.99, 1 - p = 0.01$

$\Lambda = 4.0404$ messages per sec

$\rho = 0.8889$

$L = 8$ messages

$W_{small} = 1.98$ sec

$W_{big} = 2.0$ sec

$$\Lambda = \lambda + (1-p)\Lambda$$

$$\Lambda = \lambda / p$$

server utilization : $\rho = \dfrac{\lambda / p}{\mu} = \dfrac{\lambda}{p\mu}$

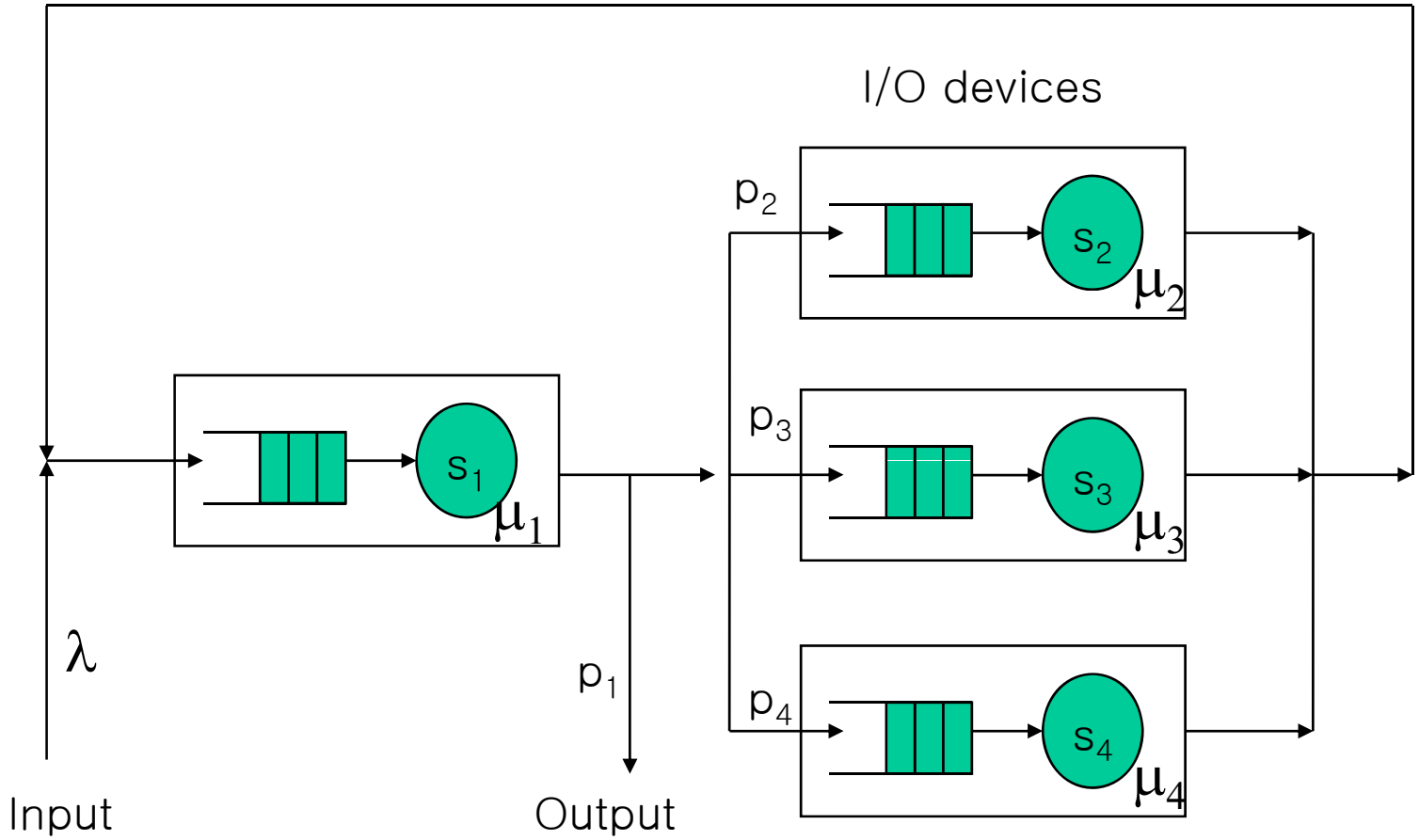
M/M/1 sytem

$$L = E(N) = \frac{\rho}{1-\rho} = \frac{\lambda}{p\mu - \lambda}$$

$$W_{small} = \frac{L}{\Lambda} = \frac{\dfrac{\lambda}{p\mu - \lambda}}{\dfrac{\lambda}{p}} = \frac{p}{p\mu - \lambda}$$

$$W_{big} = \frac{L}{\lambda}$$

System to simulate



- What to measure?
  - L1, L2, L3, L4, W1, W2, W3, W4, L, W
  - Utilization of each server

Facilities

- Four facilities
  - f1=facility (“server1”,1);
  - f2=facility (“server2”,1);
  - f3=facility (“server3”,1);
  - f4=facility (“server4”,1);

Events

- Event 1: Job arrival at Server 1
  - from outside
- Event 2: Job arrival at Server 1 internally
- Event 3: Request Server 1
- Event 4: Job completion at Server 1

- Event 5: Job arrival at Server 2
- Event 6: Request Server 2
- Event 7: Job completion at Server 2

- Event 8: Job arrival at Server 3
- Event 9: Request Server 3
- Event 10: Job completion at Server 3

- Event 11: Job arrival at Server 4
- Event 12: Request Server 4
- Event 13: Job completion at Server 4

Event Handling

- Event 1: Job arrival at Server 1 from outside
  - schedule a new arrival (when to occur?)
  - schedule Request Server 1 (when to occur?)
- Event 2: Job arrival at Server 1 internally
  - schedule Request Server 1
- Event 3: Request Server 1
  - if (request(server1) == 0), schedule Job completion at Server 1
  - else, do nothing (queueing will be done by the “request” function!)
- Event 4: Job completion at Server 1
  - release(server 1)
  - schedule Job arrival at server 2, 3, 4 with prob 2, 3, 4, respectively
- All other events, the same.

Any optimization?

- Do we need to separate the following two events?
  - Event 5: Job arrival at Server 2
  - Event 6: Request Server 2
- What about ?
  - Event 8: Job arrival at Server 3
  - Event 9: Request Server 3
- What about ?
  - Event 11: Job arrival at Server 3
  - Event 12: Request Server 3

Any further idea?

- Why do we need the similar codes for so many events?
- Let’s make token carry the visiting server info
- Then, we don’t have to separate events for different servers
  - Job arrival event
    - check which server?
    - request that server

How to make a token carry info?

- A Token is a simple integer.
- How to make a token carry info?
- Define an array of token pool where each entry is a structure
- A token (integer) is an index of the array
- Now, at each event that happens with a token, we can use the token's information
- But, we need to build functions for managing the token pool
  - initTokenPool
  - getToken
  - freeToken

Homework 3

- Let
  - Service rates: μ_1=1.333, μ_2=1/1, μ_3 = 1/2, μ_4 =1/4
  - Job transferring probs: p_1=p_2=p_3=p_4=0.25
- As increasing λ (x-axis) from 1/20 to 1/1, draw the followings by Simulation
  - L1, L2, L3, L4, W1, W2, W3, W4, L, W
  - Utilization of each server
- Draw the same curves by analysis
- Answer the followings
  - Which server is the bottleneck that first makes the system unstable
  - How to reassign server rates so that λ can be maximized while keeping the system stable