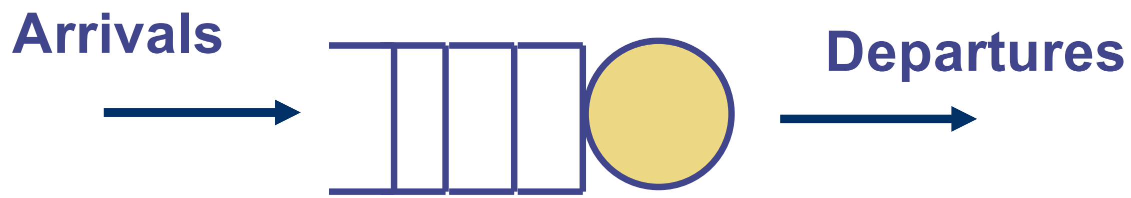
COMP9334
Capacity Planning for Computer Systems and Networks

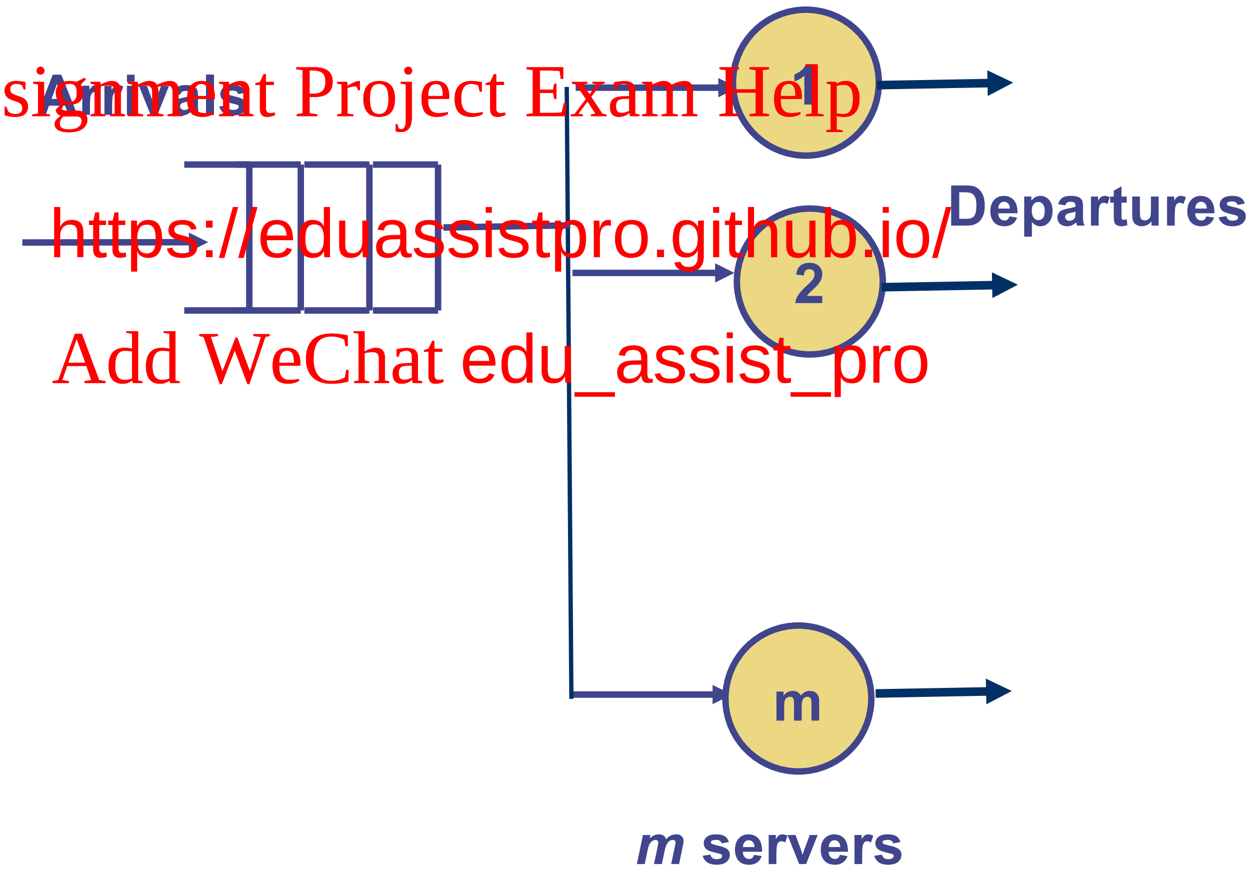
Week 3B

Last lecture: Queues with Poisson arrivals

- Single-server

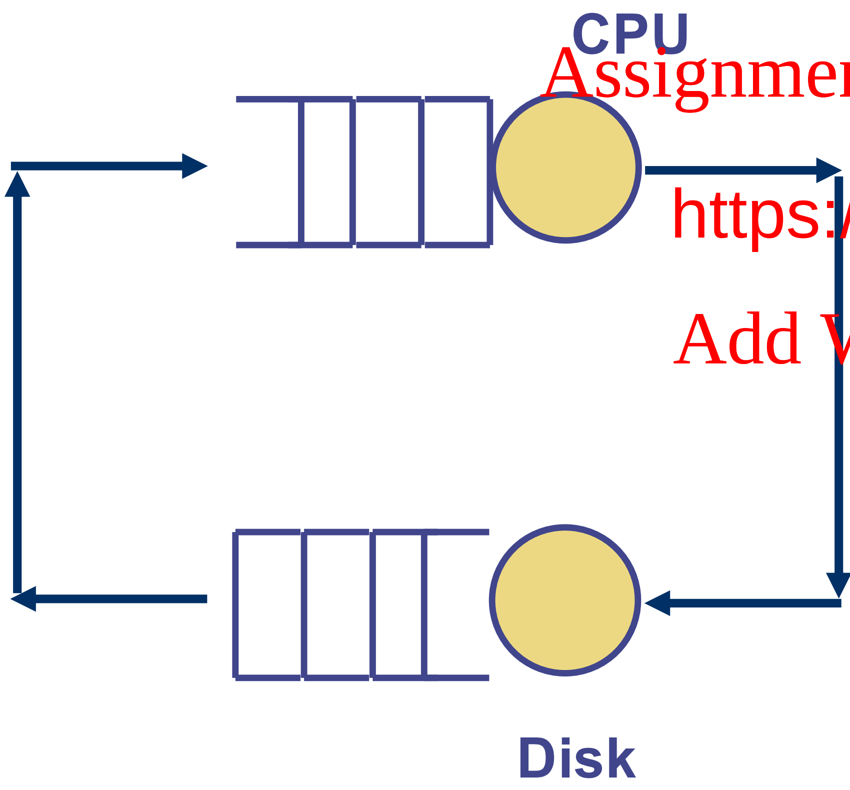


- Multi-server



This week: Markov Chain

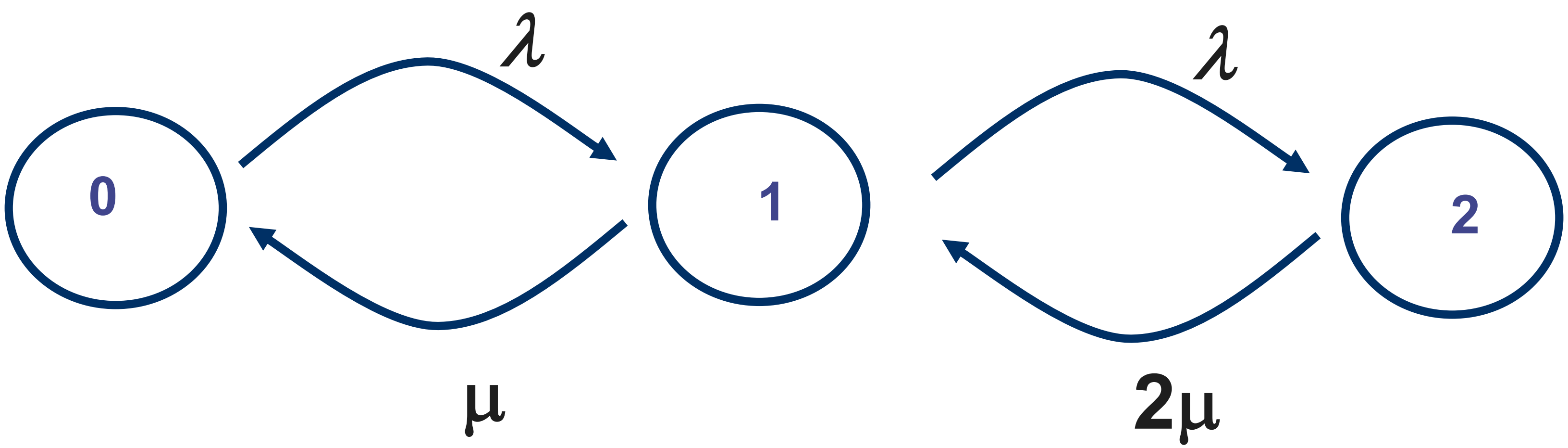
- You can use Markov Chain to analyse
  - Closed queueing network (see example below)
  - Reliability problem



- There are *n* jobs in the closed system
- e response time of
- response time if we replace the CPU with one that is twice as fast?

Markov chain

- The state-transition model that we have used is called a continuous-time Markov chain
  - There is also discrete-time Markov chain
- The transition from a state of the Markov chain to another state is characterised by an exponential distribution
  - E.g. The transit is exponential with rate r_{pq}, then consid
  - Prob [Transition from State p to me δ | State p] = $r_{pq}\ \delta$



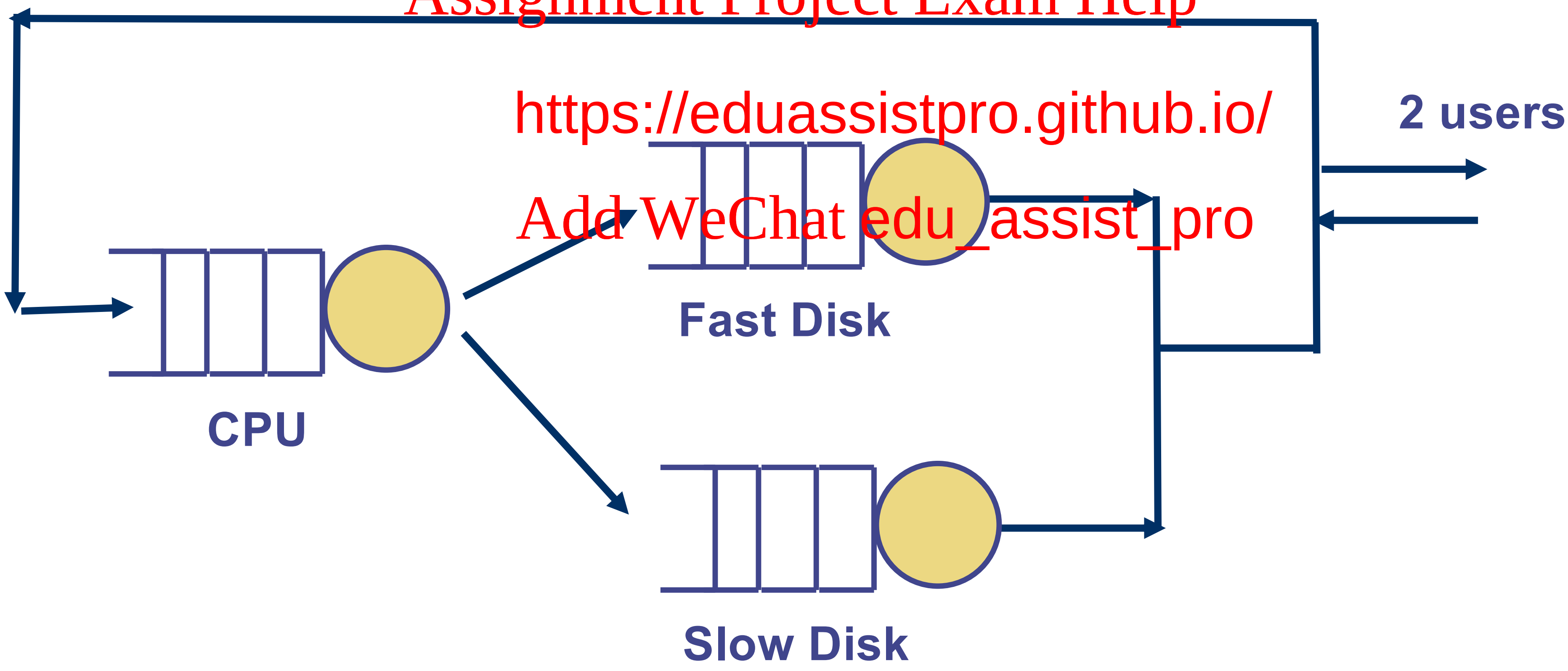
Method for solving Markov chain

- A Markov chain can be solved by
  - Identifying the states
  - Find the transition rate between the states
  - Solve the steady state probabilities
- You can then use the steady state probabilities as a stepping stone t terest (e.g. response time etc.)

- We will study two Markov chai in this lecture:
  - Problem 1: A Database server
  - Problem 2: Data centre reliability problem

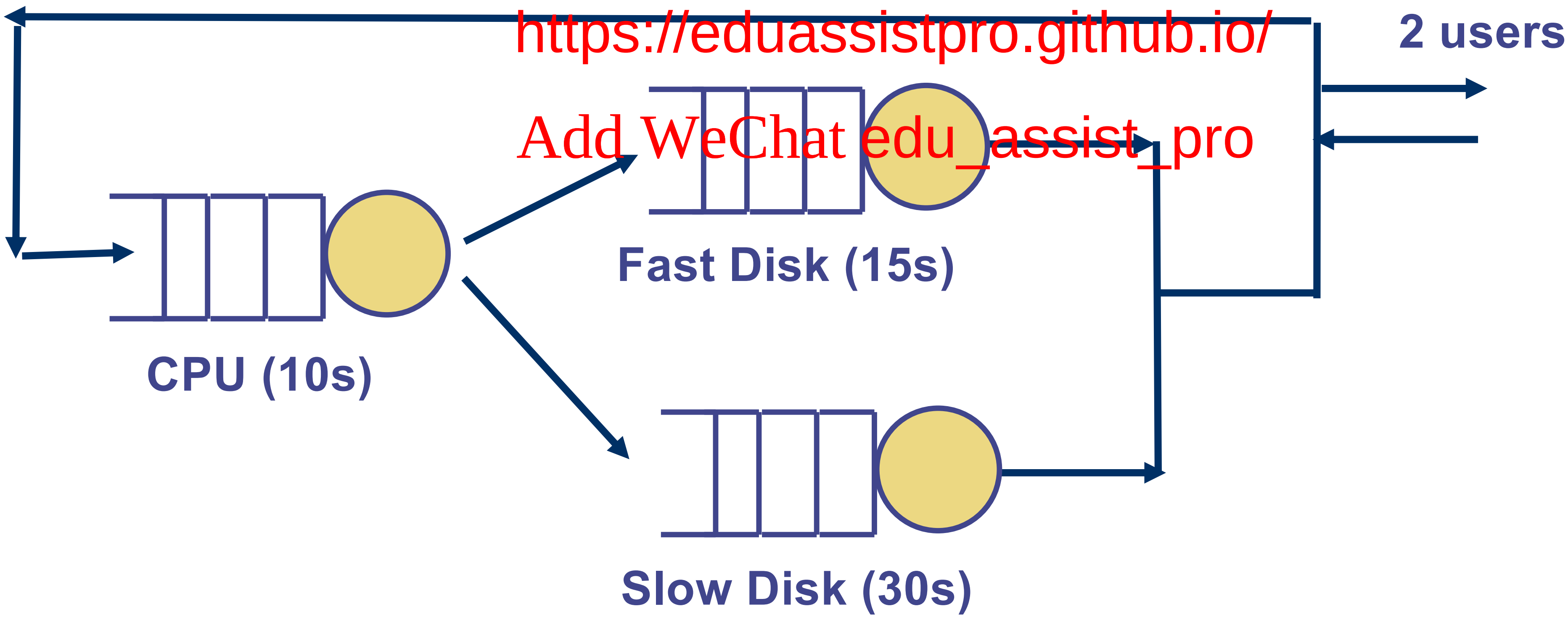
Problem 1: A DB server

- A database server with a CPU, a fast disk and a slow disk
- At peak demand, there are always two users in the system
- Users alternate between the CPU and the disks
- The users will equally likely find the file on either disk



Problem 1: A DB server (cont'd)

- Fast disk is twice as fast as the slow disk
- Typical transactions take on average 10s CPU time
- Fast disk takes on average 15s to serve all files for a user
- Slow disk takes on average 30s to serve all files for a user
- The time that each transaction requires from the CPU and the disks is exponentially distributed



Typical capacity planning questions

- What response time can a typical user expect?
- What is the utilisation of each of the system resources?
- How will performance parameters change if number of users are doubled?
- If fast disk fails and all files are moved to slow disk, what will be the new

Choice of states #1

- Use a 2-tuple (A,B) where
  - A is the location of the first user
  - B is the location of the second user
  - A, B are drawn from {CPU,FD,SD}
    - FD = fast disk, SD = slow disk
  - Example states
    - (CPU,CPU):
    - (CPU, FD): 1st user at CPU, 2n disk
  - Total 9 states
- Question: If there are *n* users,
  - What are the states?
  - How many states are there?

Choice of states #2

- We use a 3-tuple (X,Y,Z)
  - X is # users at CPU
  - Y is # users at fast disk
  - Z is # users at slow disk
- Examples

  - (2,0,0): both users at CPU
  - (1,0,1): one use low disk
- There are six po list them?

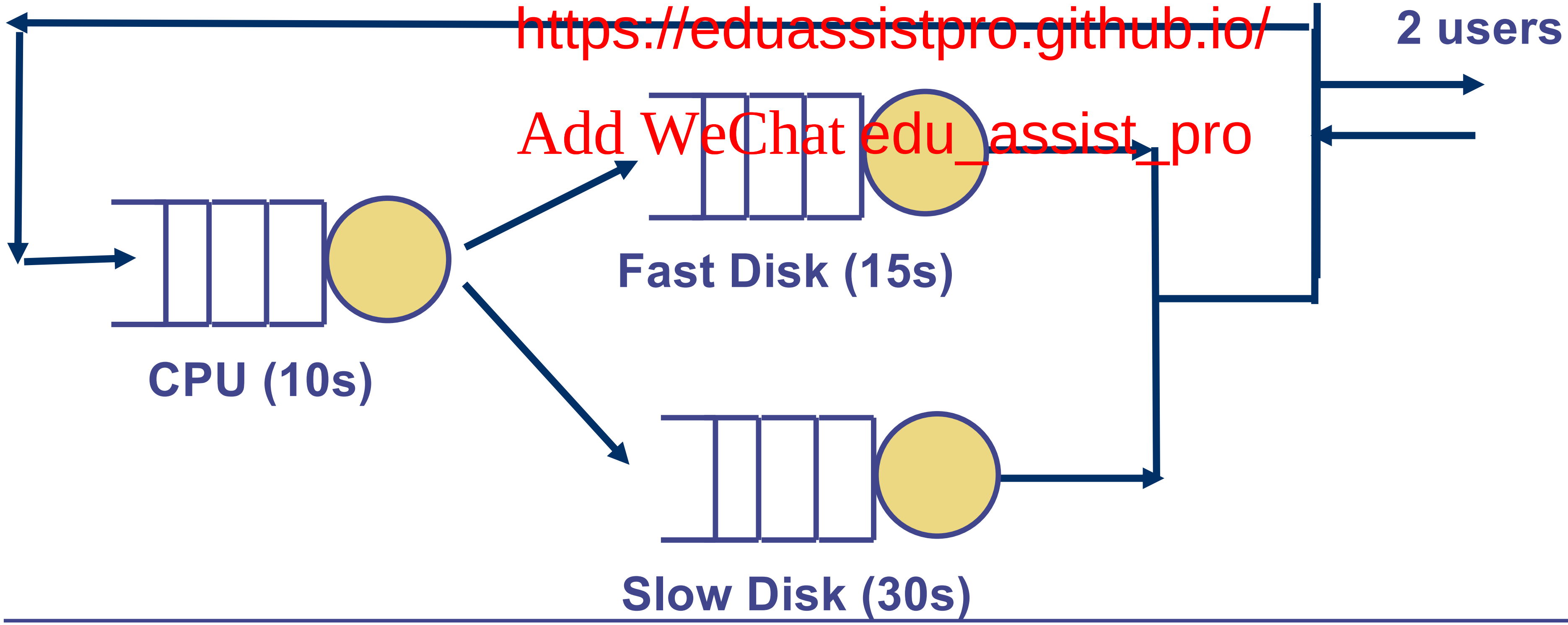
  -
- If there are n users, how many states are there?

$$\frac{(n+1)(n+2)}{2}$$

Choice #2 requires less #states but loses certain information.

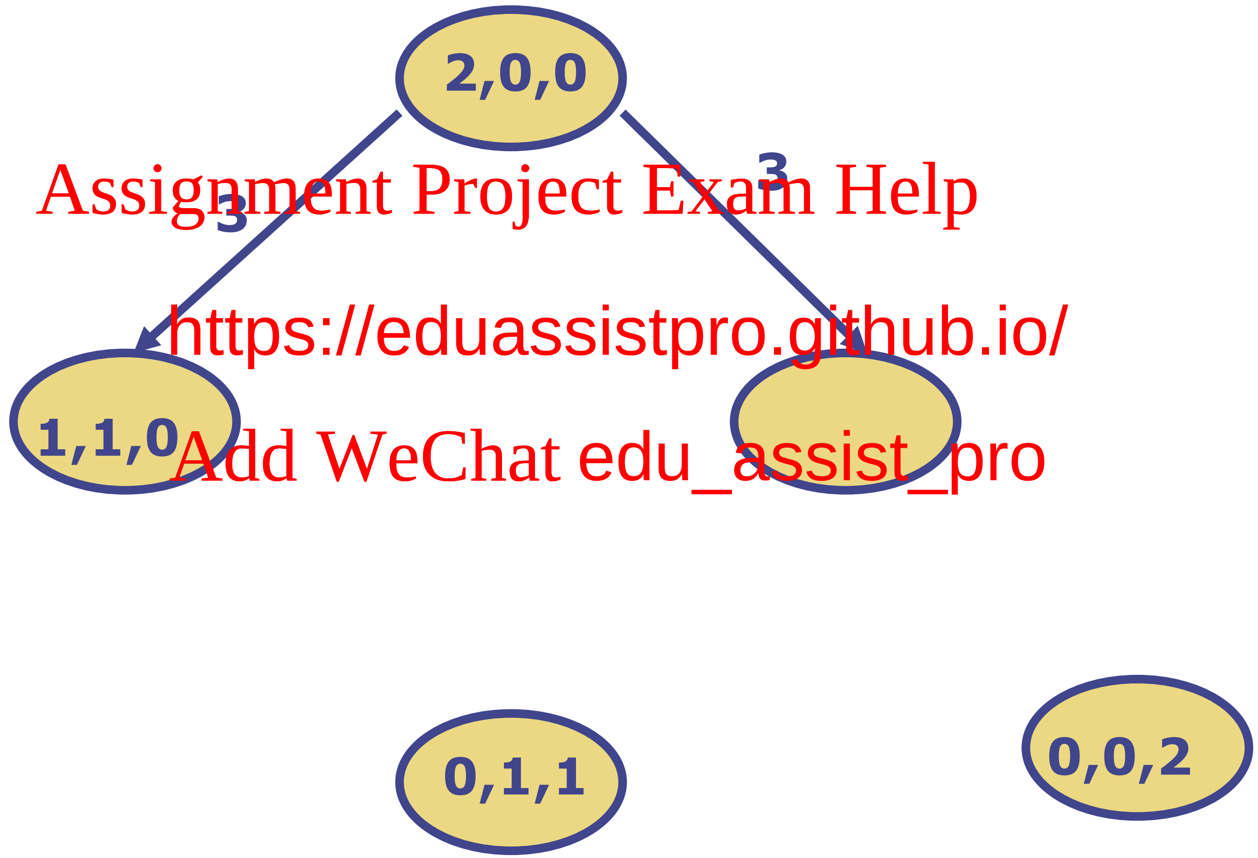
Identifying state transitions (1)

- A state is: (#users at CPU, #users at fast disk, #users at slow disk)
- What is the rate of moving from State (2,0,0) to State (1,1,0)?
  - This is caused by a job finishing at the CPU and move to fast disk
  - Jobs complete at CPU at a rate of 6 transactions/minute
  - Half of the jobs go to the fast disk
- Transition rate from (2,0,0) ➔ (1,1,0) = 3 transactions/minute
- Similarly, transition ra 3 transactions/minute



State transition diagram (2)

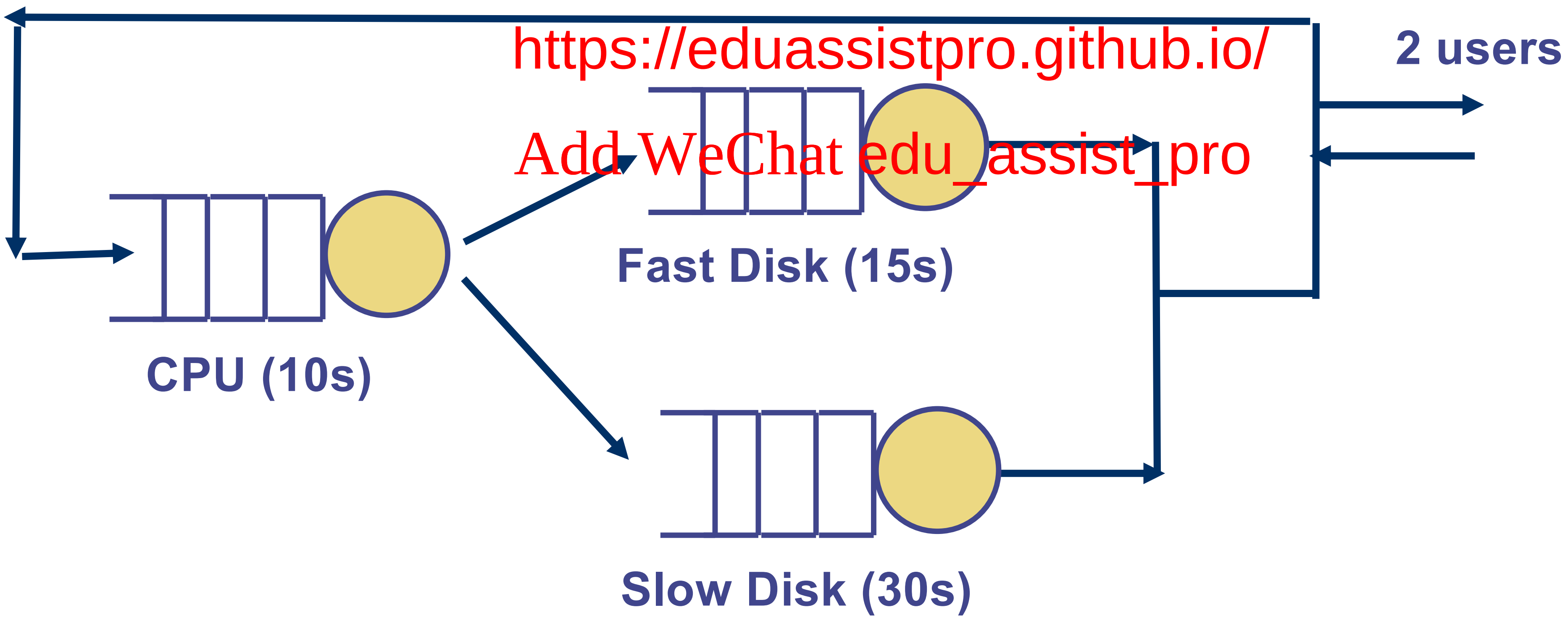
- Transition rate from (2,0,0) ➔ (1,1,0) = 3 transactions/minute
- Transition rate from (2,0,0) ➔ (1,0,1) = 3 transactions/minute



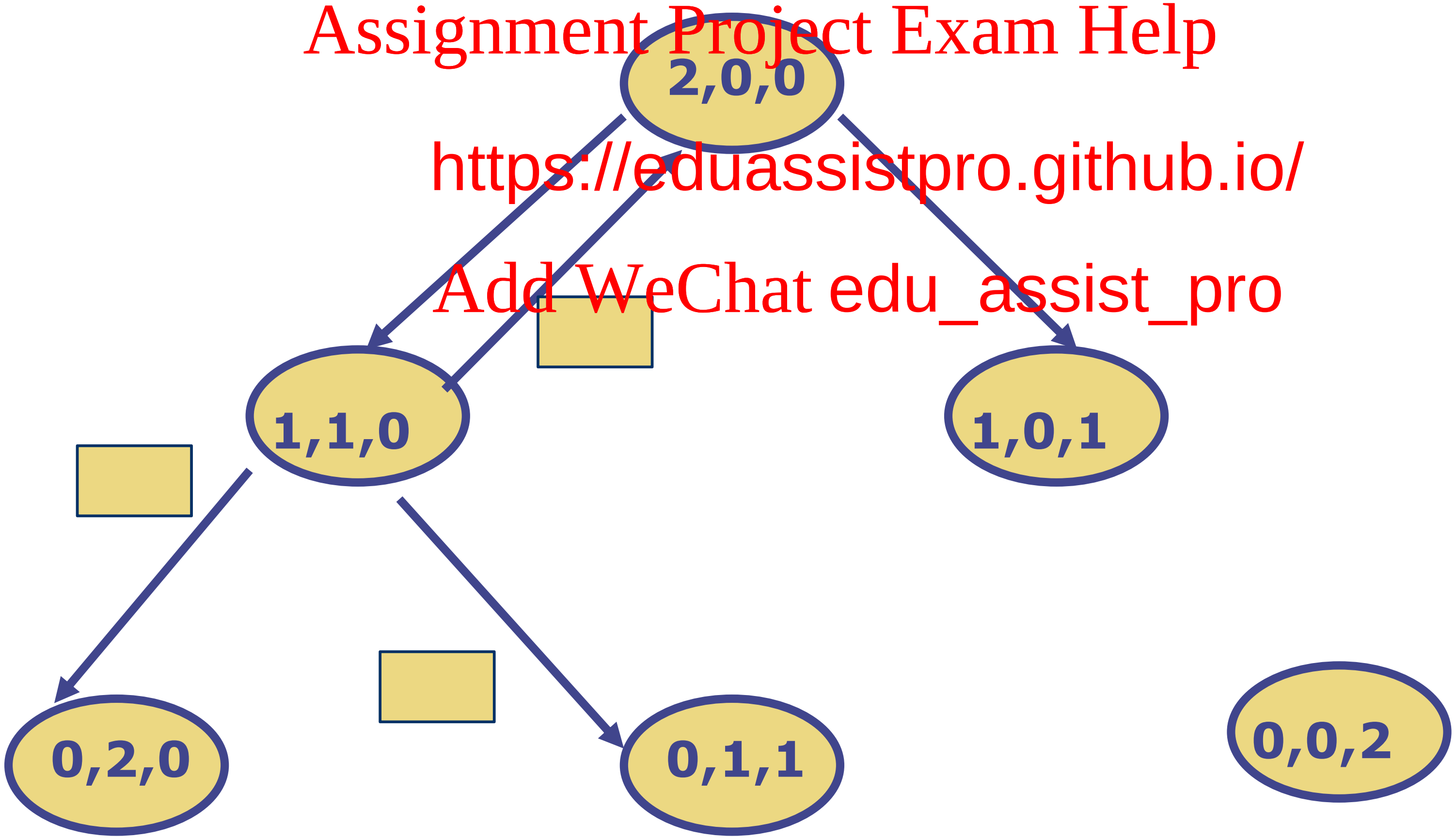
- Question: What is the transition rate from (2,0,0) ➔ (0,1,1)?

Identifying state transitions (2)

- From (1,1,0) there are 3 possible transitions
  - Fast disk user goes back to CPU (2,0,0)
  - CPU user goes to the fast disk (0,2,0), or
  - CPU user goes to the slow disk (0,1,1)
- Question: What are the transition rates in number of transactions per minute?



Completing the state transition diagram



Exercise

- The state transition diagram is still not complete. Choose any two state transitions and determine their rates.

Complete state transition diagram

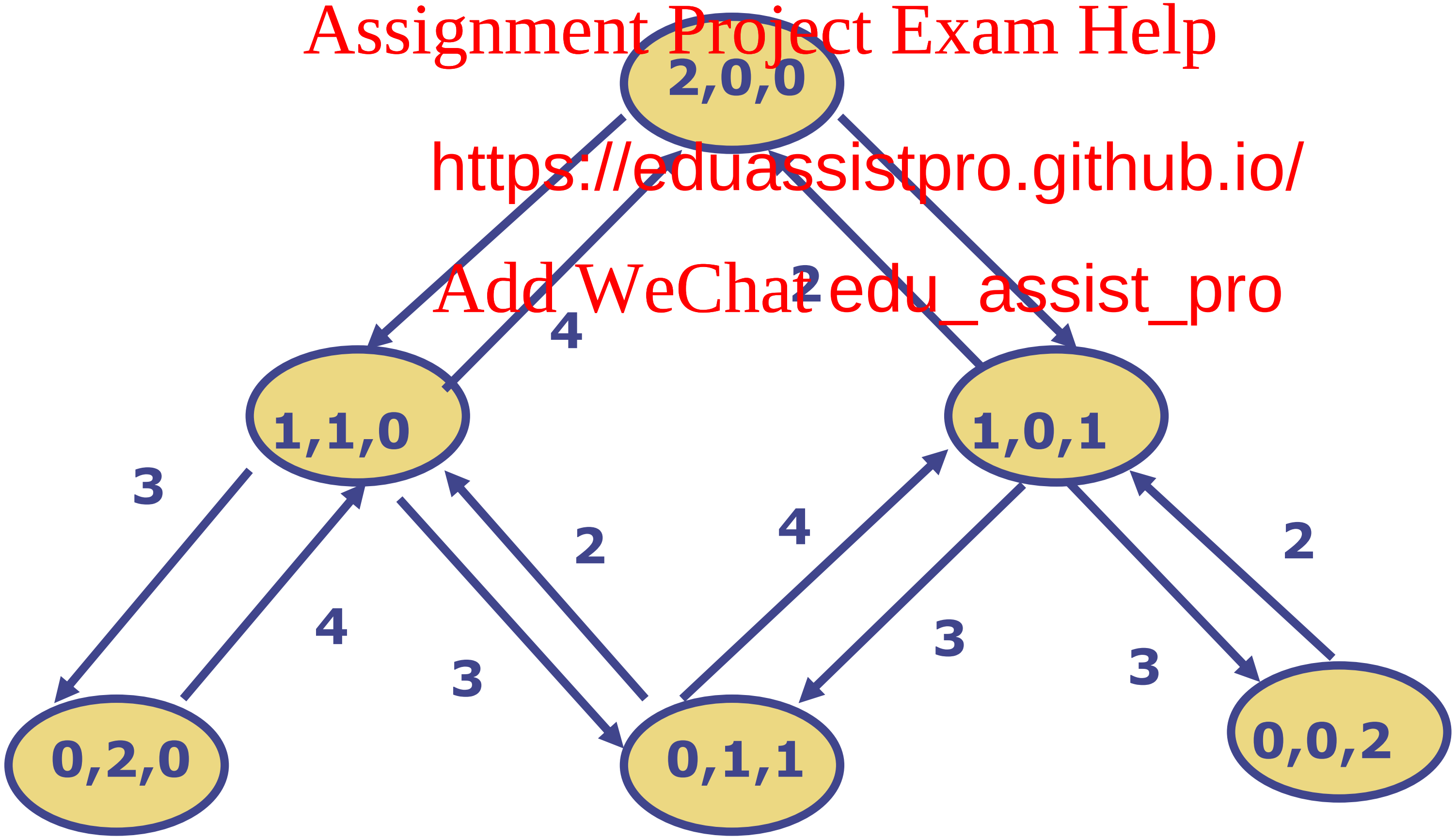
Balance Equations

Define

$P_{(2,0,0)} =$ Probability in state (2,0,0)

$P_{(1,1,0)} =$ Probability in state (1,1,0) etc.

Exercise: Write down the balance equation for state (2,0,0)



Flow balance equations

- You can write one flow balance equation for each state:

$$6 P_{(2,0,0)} - 4 P_{(1,1,0)} - 2 P_{(1,0,1)} + 0 P_{(0,2,0)} + 0 P_{(0,1,1)} + 0 P_{(0,0,2)} = 0$$

$$-3 P_{(2,0,0)} + 10 P_{(1,1,0)} + 0 P_{(1,0,1)} - 4 P_{(0,2,0)} - 2 P_{(0,1,1)} + 0 P_{(0,0,2)} = 0$$

$$-3 P_{(2,0,0)} + 0 P_{(1,1,0)} + 8 P_{(1,0,1)} + 0 P_{(0,2,0)} - 4 P_{(0,1,1)} - 2 P_{(0,0,2)} = 0$$

$$0 P_{(2,0,0)} - 3 P_{(1,1,0)} + \quad P_{(0,1,1)} + 0 P_{(0,0,2)} = 0$$

$$0 P_{(2,0,0)} - 3 P_{(1,1,0)} - 3 P_{(1,0,1)} + 0 P_{(0,2,0)} \quad {}_{(0,1,1)} + 0 P_{(0,0,2)} = 0$$

$$0 P_{(2,0,0)} + 0 P_{(1,1,0)} - 3 P_{(1,0,1)} + 0 P_{(0,2,0)} + 0 P_{(0,1,1)} + 2 P_{(0,0,2)} = 0$$

- However, there are only 5 linearly independent equations.
- Need one more equation:

Steady State Probability

- You can find the steady state probabilities from 6 equations
  - It's easier to solve the equations by a software packages, e.g
    - Python, Matlab, Octave, etc.
- The solutions are:
  - $P_{(2,0,0)} = 0.1391$
  - $P_{(1,1,0)} = 0.1043$
  - $P_{(1,0,1)} = 0.2087$
  - $P_{(0,2,0)} = 0.0783$
  - $P_{(0,1,1)} = 0.1565$
  - $P_{(0,0,2)} = 0.3131$

- I used Python (the numpy library) to solve these equations
  - The file is "data_server.py" (can be downloaded from the course web site)
- How can we use these results for capacity planning?

Model interpretation

- Response time of each transaction
  - Use Little's Law R = N/X with N = 2
    - For this system:
      - System throughput = CPU Throughput

    - Throughput = Utilisation x Service rate
      - Recall Utilie time (From Lecture 1B)

    - CPU utilisation (using states is a job at CPU): $P_{(2,0,0)} + P_{(1,1,0)} + P_{(1,0,1)} = 0.452$
    - Throughput = 0.452 x 6 = 2.7130 transactions / minute
    - Response time (with 2 users) = 2 /2.7126 = 0.7372 minutes per transaction

Sample capacity planning problem

- What is the response time if the system has up to 4 users instead of 2 users only?
  - You can't use the previous Markov chain
  - You need to develop a new Markov chain
    - The states are again (#users at CPU, #users at fast disk, #users at slow disk)
    - States are (
    - There are 1
    - Determine the transition rates
    - Write down the balance equa e them.
    - Use the steady state probabilities and Little's Law to determine the new response time
    - You can do this as an exercise
    - Throughput = 3.4768 (up 28%), response time = 60.03 seconds (up 56%)

Computation aspect of Markov chain

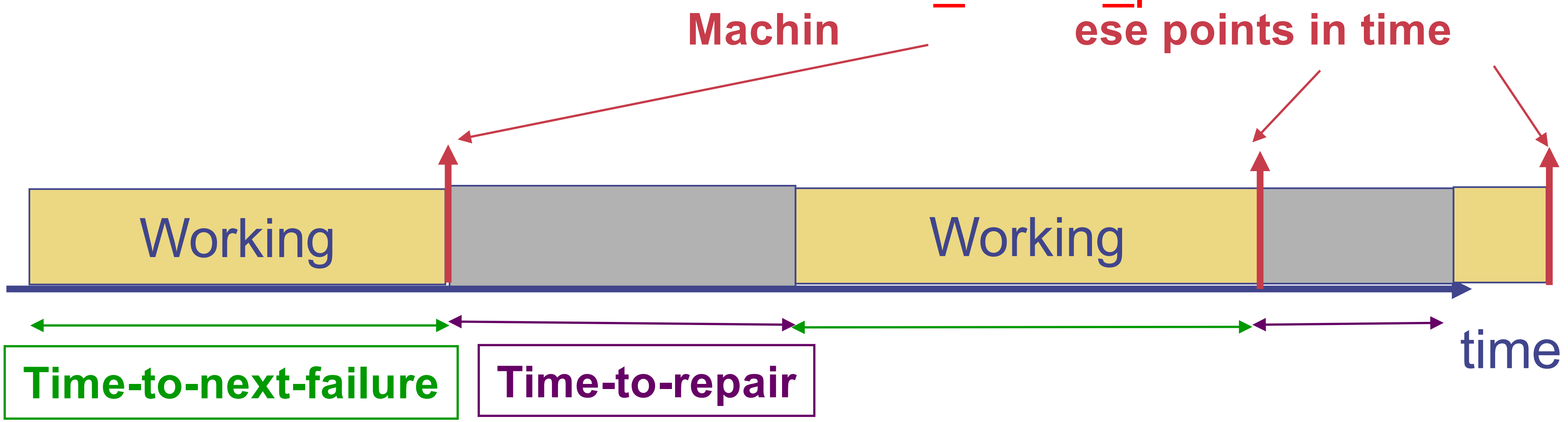
- This example shows that when there are a large number of users, the burden to build a Markov chain model is large
  - 15 states
  - Many transitions
  - Need to solve 15 equations in 15 unknowns

- Is there a faster

  - Yes, we will lo in a few weeks and it can obtain the response time much y

Machine working-repair cycle

- A data centre consists of a number of machines
- Machines can fail and have to be repaired
- Terminology:
  - Time-to-next failure: From the time a machine has been fixed to the time it next fails
  - Time-to-repair = ti ue + service time to repair the machine
    - Time-to-repair is a response time



Data centre reliability problem

- Example: A data centre has 10 machines
  - Each machine may go down
    - Time-to-next-failure is exponentially distributed with mean 90 days
  - Service time to repair is exponentially distributed with mean 6 hours

- Capacity planni
    - Can I make ines are available 99.9999% of the time?
    - What is the probability that chines are available?
    - How many repair staff are required to guarantee that at least *k* machines are available with a given probability?
    - What is the mean-time-to-repair (MTTR) a machine?
      - Note: Mean-time-to-repair includes waiting time at the repair queue.

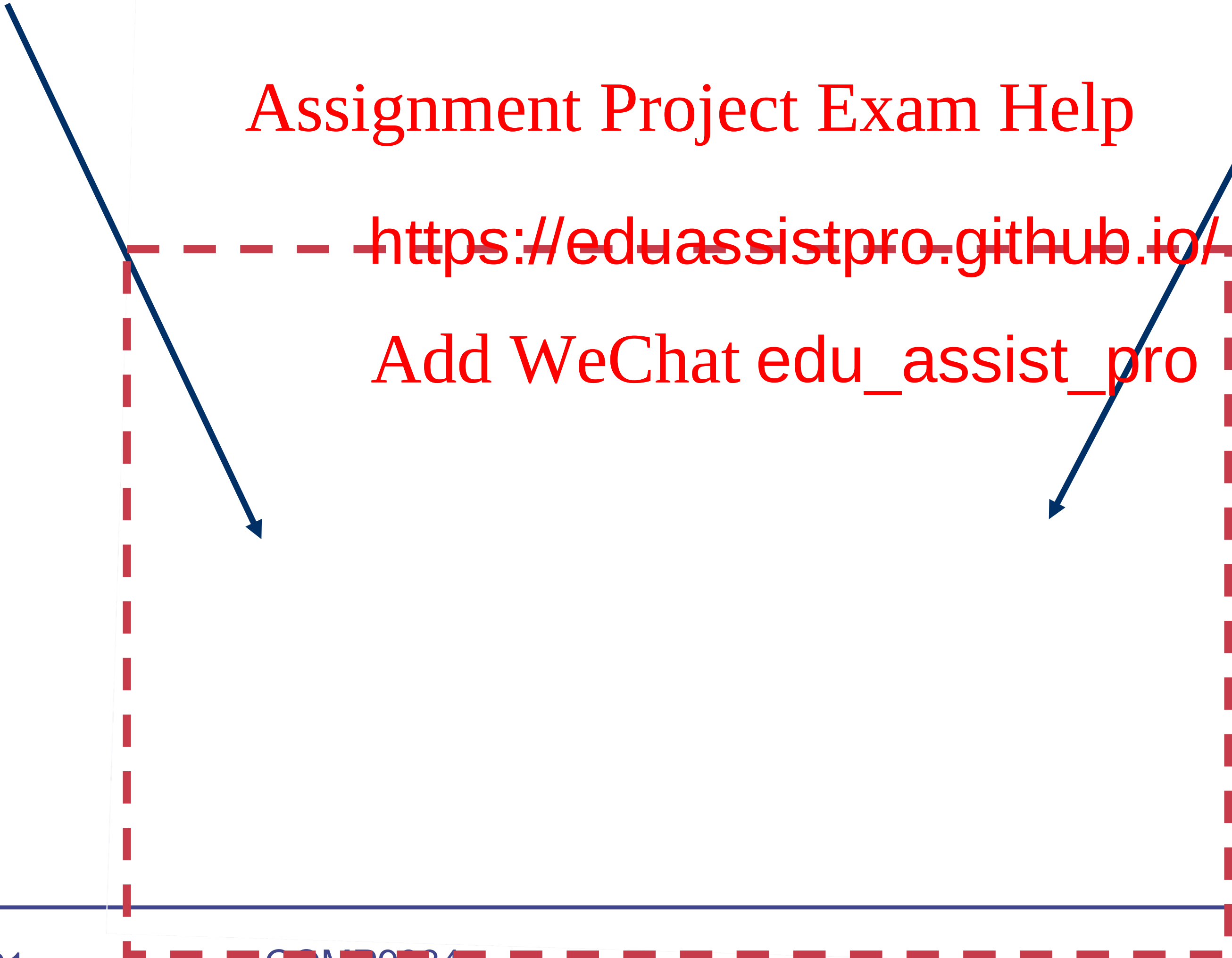
Data centre reliability - general problem

- Data centre has
  - M machines
  - N staff maintain and repair machine
  - Assumption: $M > N$
- Automatic diagnostic system
  - Check “heartbeat” by “ping” (Failure detection)
  - Staff are informed i
- Repair work
  - If a machine fails, any one of the idle there is one) will attend to it.
  - If all repair staff are busy, a failed machine will need to wait until a repair staff has finished its work
- This is a queueing problem solvable by Markov chain!!!
- Let us denote
  - ***λ = 1 / Mean-time-to-failure***
  - **μ = 1/ Mean service time to repair a machine**

Queueing model for data centre example

An arrival is due to a machine failure.

A departure occurs when a machine has been repaired.



We build a Markov chain for this box.

Markov model for the repair queue

- State *k* represents *k* machines have failed
- Part of the state transition diagram is showed below

The rate of failure for one machine is λ. In State 0, there are M working machine, the failure rate is $M\lambda$.

The same argument holds for other state transition probability.

Markov Model for the repair queue

Note: There are (M+1) states.

Why is it $N\mu$?

Why not $(N+1)\mu$?

Solving the model

- We can solve for P(0), P(1), …, P(M)

Where

$$P(0) = \left[\sum_{k=0}^{N} \left(\frac{\lambda}{\mu}\right)^k C_k^m + \sum_{k=N+1}^{M} \left(\frac{\lambda}{\mu}\right)^k C_k^m \frac{N^{N-k} k!}{N!} \right]^{-1}$$

Using the model

- Probability that exactly *k* machines are available =
- Probability that at least *k* machines are available
  =
- But expression for P(*k*)'s are complicated, need numerical software

- Example:
  - M = 120
  - Mean-time-to-failure = 500 minutes
  - Mean service time to repair = 20 minutes
  - N = 2, 5 or 10
  - The results are showed in the graphs in the next 2 pages
    - I used the file "data_centre.py" to do the computation, the file is available on the course web site.

Probability that exactly k machines operate

Probability that at least k machines operate

Think time ~ Mean-time-to-failure (MTTF) = 1 / λ

Throughput
~ Mean machine failure
rate
(see next page)

Mean time to repair (MTTR)
= Queueing time for repair + actual repair time

Can compute MTTR using Little's Law.

Mean machine failure rate

State	Probability	Failure rate
0	P(0)	Mλ
1	P(1)	(M-1)λ
2	P(2)	(M-2)λ
…		
k		-k)λ
…		
M	P(M)	0

$$\bar{X}_f = \sum_{k=0}^{M-1} (M-k)\lambda P(k)$$

Continuous-time Markov chain

- Useful for analysing queues when the inter-arrival or service time distribution is exponential
- The procedure is fairly standard for obtaining the steady state probability distribution
  - Identify the state
  - Find the state transition rates
  - Set up the bala
  - Solve the stea
- We can use the steady state p obtain other performance metrics: through se time etc.
  - May need Little's Law etc.
- Continuous-time Markov chain is only applicable when the underlying probability distribution is exponential but the operations laws (e.g. Little's Law) are applicable no matter what the underlying probability distributions are.

Markov chain

- Markov chain is big field in itself. We have touched on only continuous-time Markov chain
  - There are also discrete time Markov chains
  - Markov chain has discrete state, a related concept is Markov process whose states are continuous

- Markov chain / p pplications
  - Page rank algo explained in terms of discrete-time Markov chain

  - Graphical Models (from machin
  - Transport engineering
  - Mathematical finance
- Personally, I use Markov chains to understand how living cells process information

References

- Recommended reading
  - The database server example is taken from Menasce et al., “Performance by design”, Chapter 10
  - The data centre example is taken from Mensace et al, “Performance by design”, Chapter 7, Sections 1-4

- For a more in-depth, and mathematical discussion of continuous-time M

  - Alberto Leon-Gracia, “Probabilitie m processes for Electrical Engineering”, Chapter 8

  - Leonard Kleinrock, “Queueing Systems”, Volume 1