

COMP9334

# Capacity Planning for Computer Systems and Networks

---

Week 9: Web services and fork-join queues

# What you have studied so far ...

---

- Performance analysis
  - Aim: Find mean response time, throughput etc.
- Techniques
  - Markov Chain
  - Single- or multi-server queues
    - Open model - M/M/1, M/G/1, G/G/1
    - Open model - M/M/m
  - A network of queues
    - Mean value analysis
  - Discrete event simulation

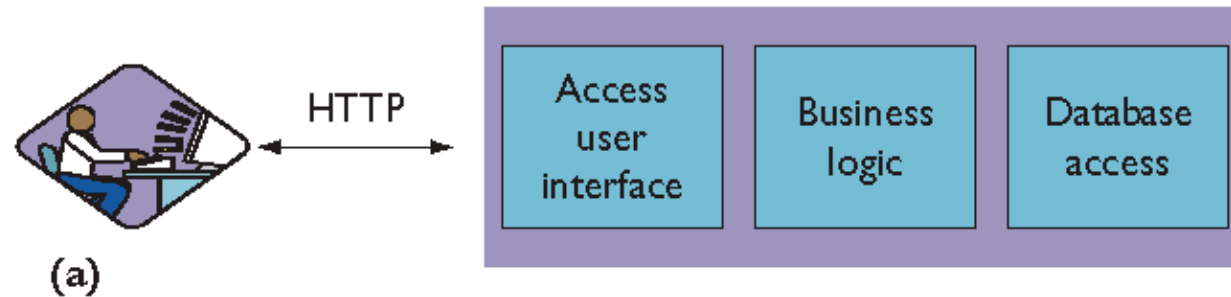
# This lecture

---

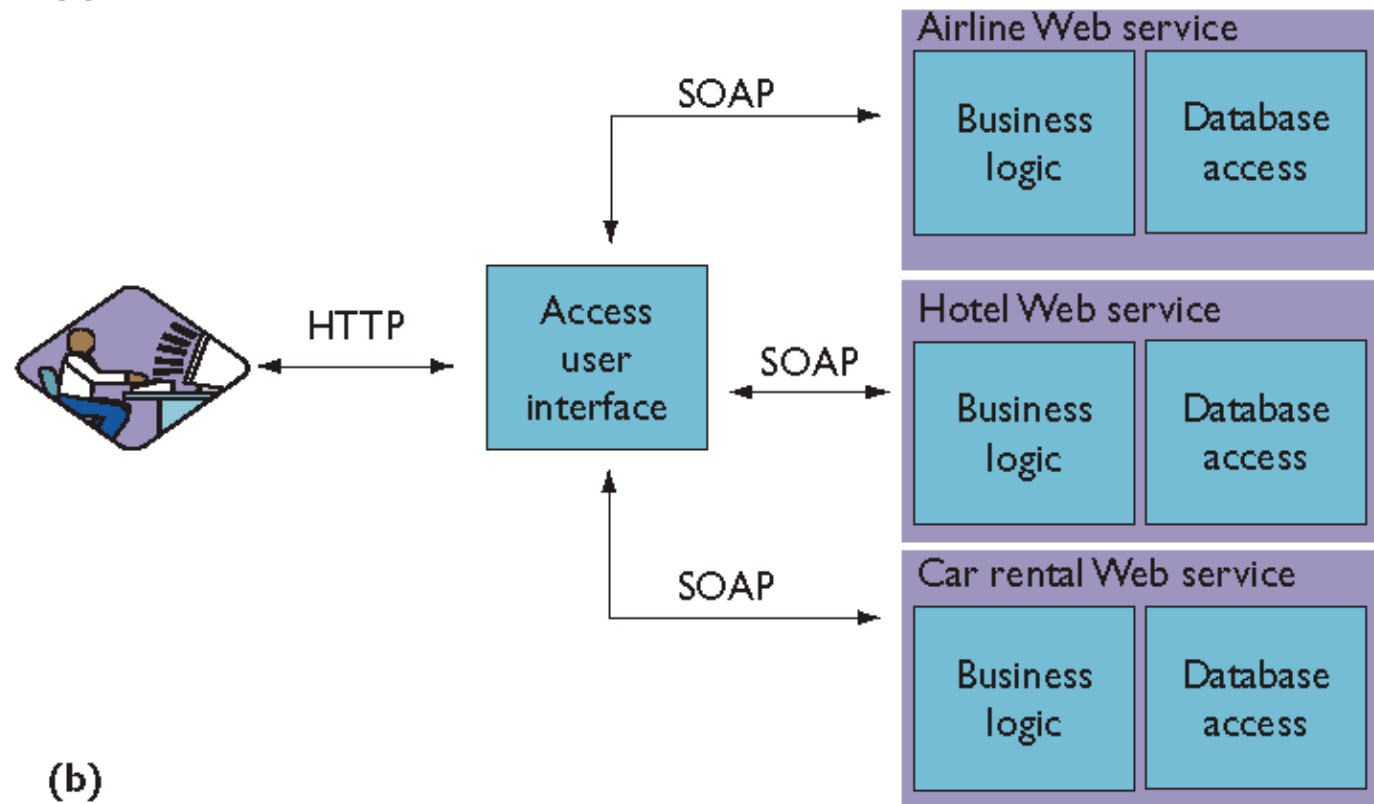
- Web services
  - What is it?
  - Performance analysis
- Fork-join queue
  - Markov chain
  - MVA

# Web access versus Web services

## (a) Web access



## (b) Composite web service for travel



# Web service performance issues

---

- Metrics
  - Response time
  - Throughput
  - Availability
- Performance analysis method
  - Operational analysis
  - Markov chain

## Web service flow graph

$V_a, V_h, V_c$ :  
Relative  
Visit  
Ratio

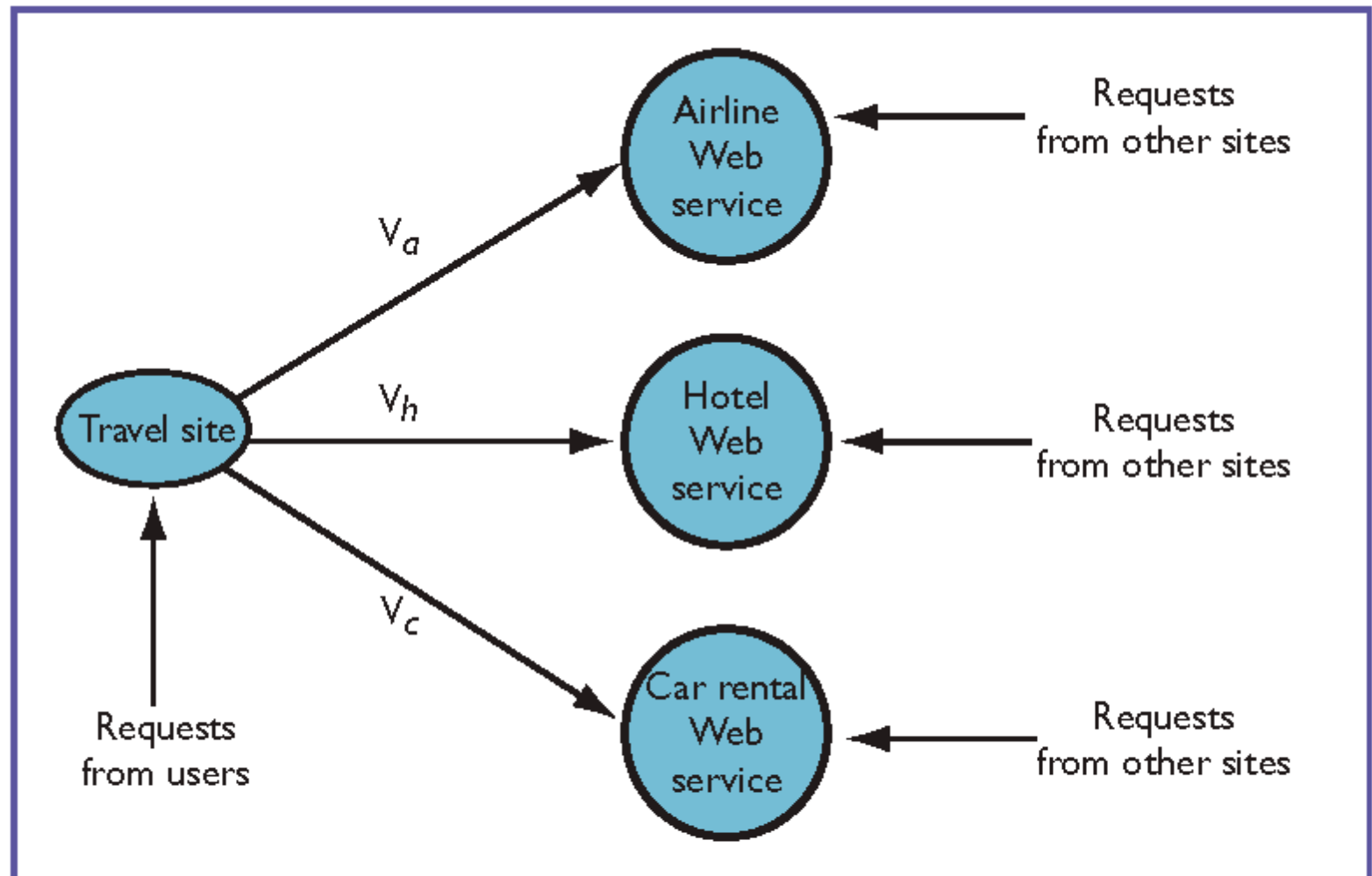
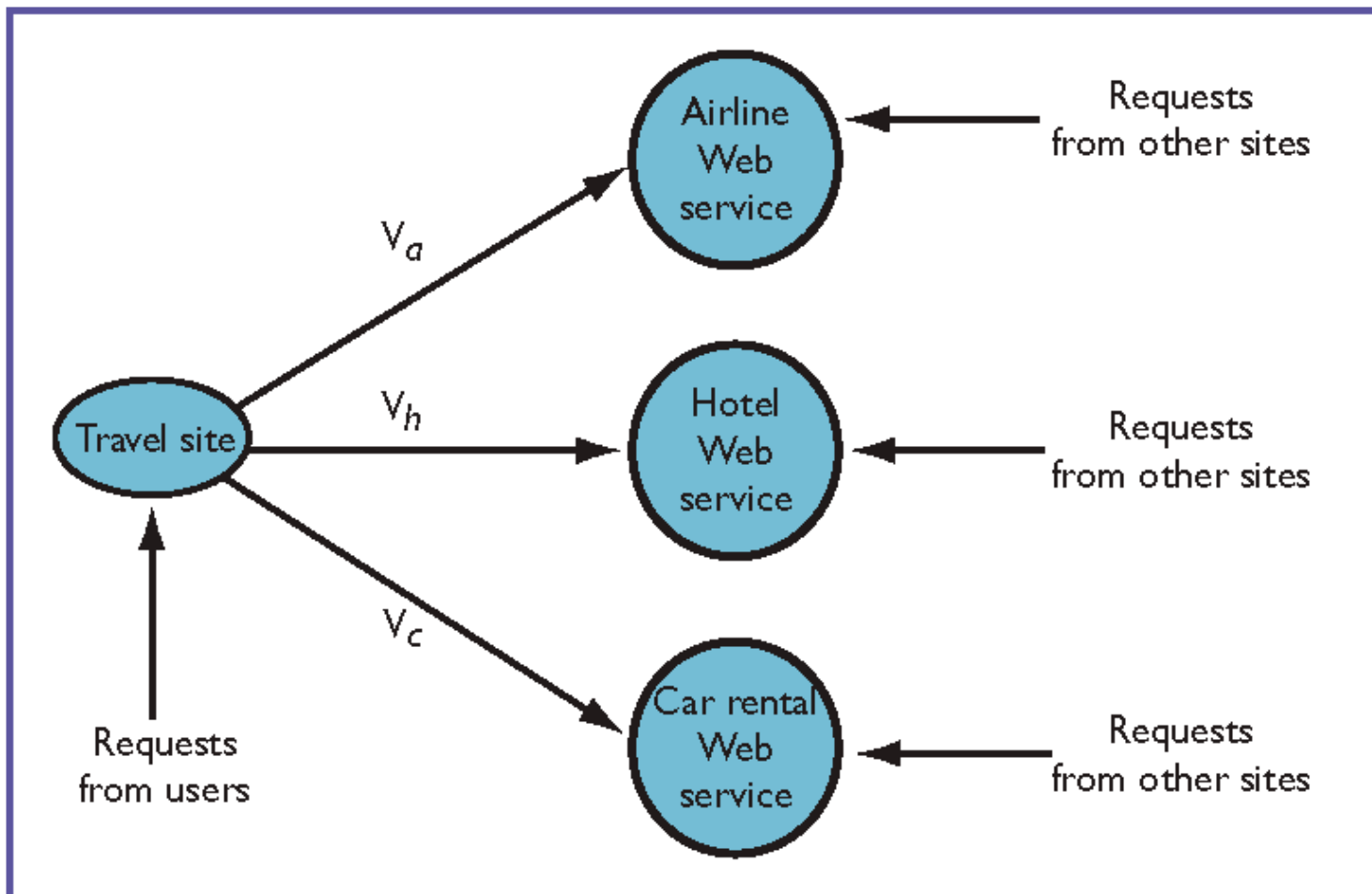


Figure 2. Web service flow graph. Arrows link the travel site to other Web services. The labels on the links indicate the average number of times a Web service is invoked per request to the travel site.

Every request to the travel site generates on average  $V_a$  requests to the Airline web service etc.



$X_{TA}$  = Throughput of travel site

$X_a$  = Throughput of airline Web service

$$X_a \geq V_a \times X_{TA}$$

Similarly,

$$X_a \geq V_a \times X_{TA}$$

$$X_h \geq V_h \times X_{TA}$$

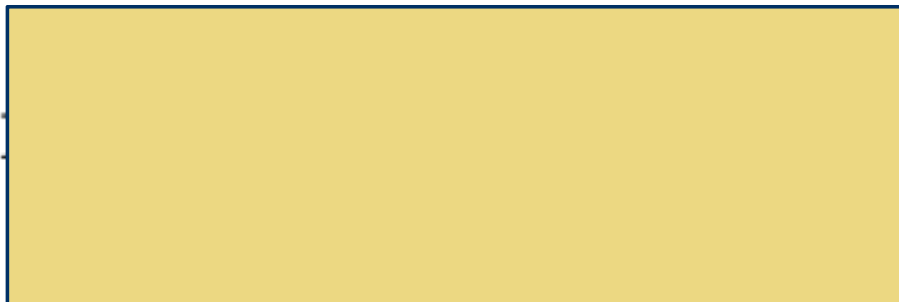
$$X_c \geq V_c \times X_{TA}$$

$X_h$  = Throughput of hotel web service

$X_c$  = Throughput of car rental web service

- Can you find an upper bound on the throughput of the travel site

$$X_{TA} \leq$$





## Example:

---

$X_a = 20$  requests/s

$X_h = 15$  requests/s

$X_c = 10$  requests/s

$V_a = 4, V_h = 2, V_c = 1$

$$X_{TA} \leq \min\left\{\frac{20}{4}, \frac{15}{2}, \frac{10}{1}\right\} = 5 \text{ requests/s}$$

The airline web service is the bottleneck of the travel web site.

## More complex web service graphs

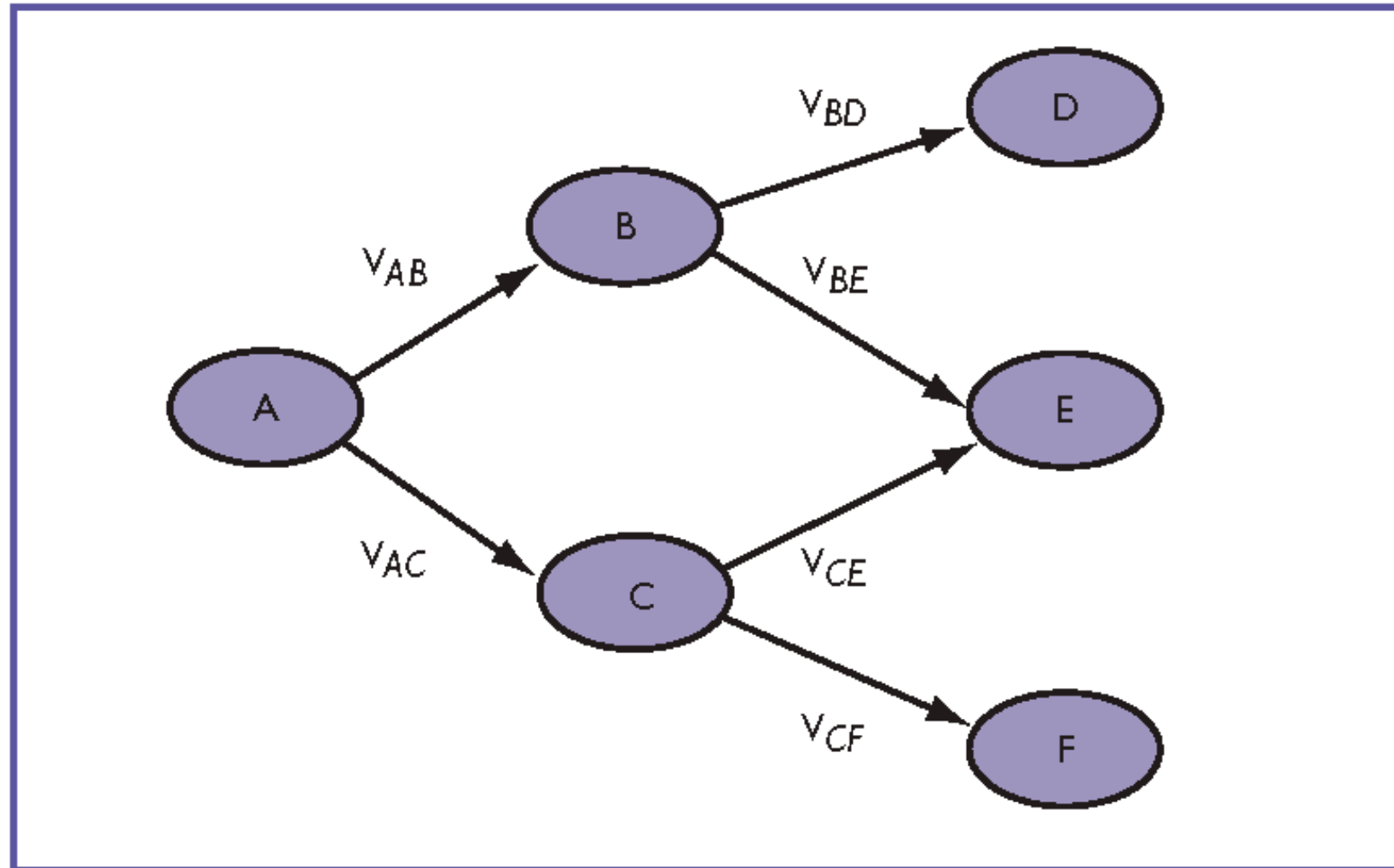



Figure 3. A more complex Web services flow graph. Web service A uses Web services B and C; B uses D and E; and C uses E and F.

**What is the bound on throughput of web service A?**

---

**Bound on the throughput of web service A is:**

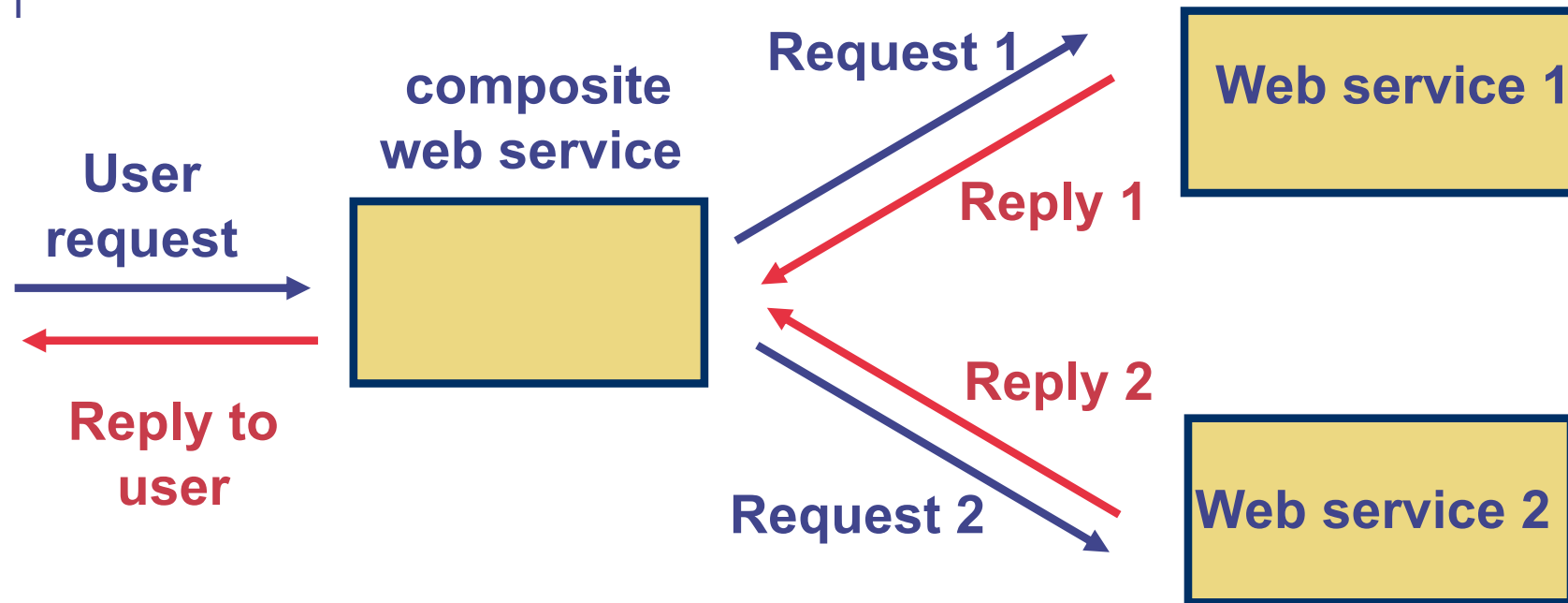
$$X_A \leq$$


# Response time analysis

---

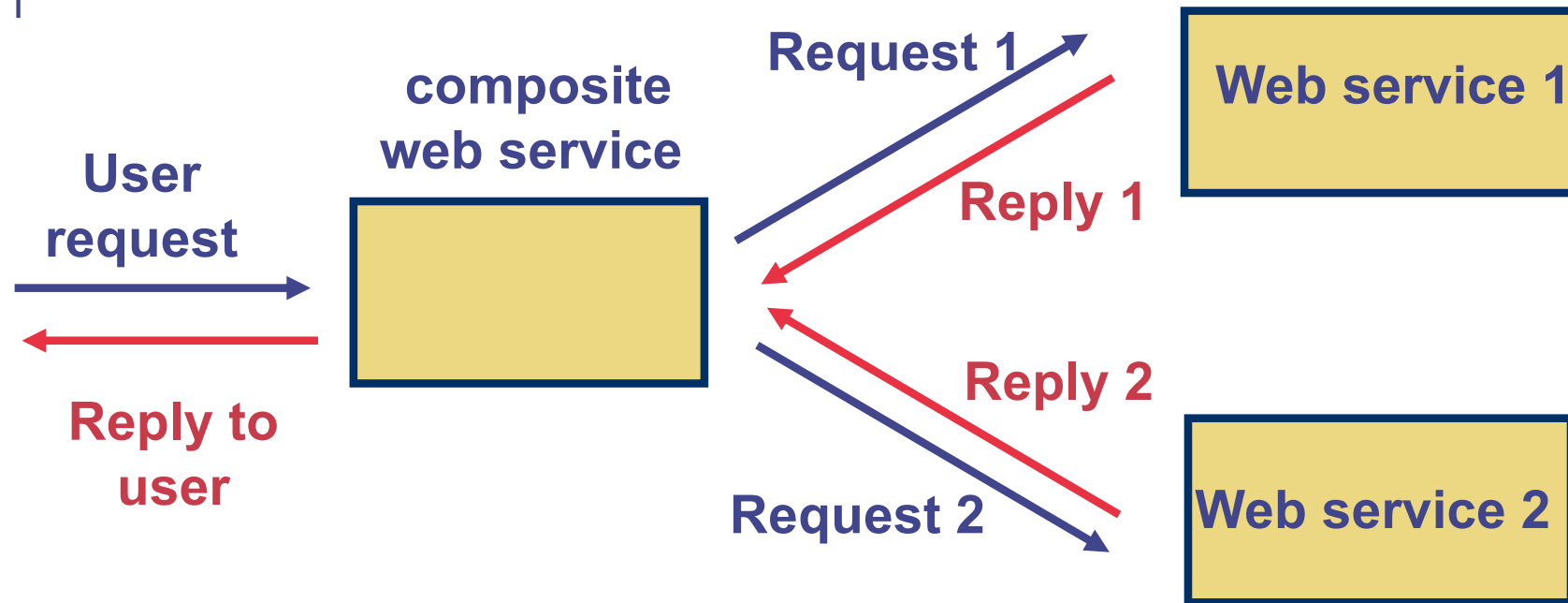
- The bottleneck analysis only gives an upper bound on the throughput
- Can we find the response time?
  - Markov chain
  - Approximate MVA
- We begin with a motivating example

# A simple web service scenario (1)



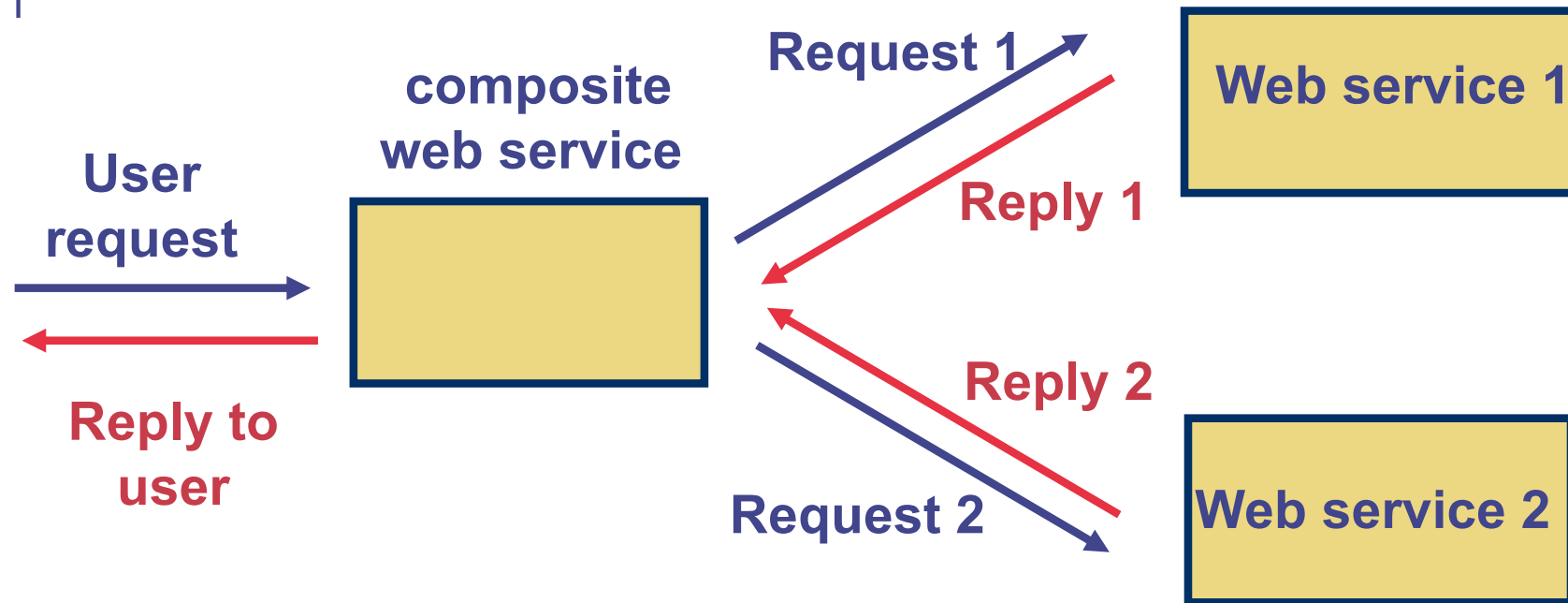
- A composite web service uses two web services
- Sequence of events
  1. Composite web service receives a user request
  2. Composite web service sends Request 1 and Request 2
  3. The web services reply *independently*
    - *That is, Reply 1 and Reply 2 may arrive at different times*
  4. After the composite web service receives *both* replies, it responds to the user

## A simple web service scenario (2)



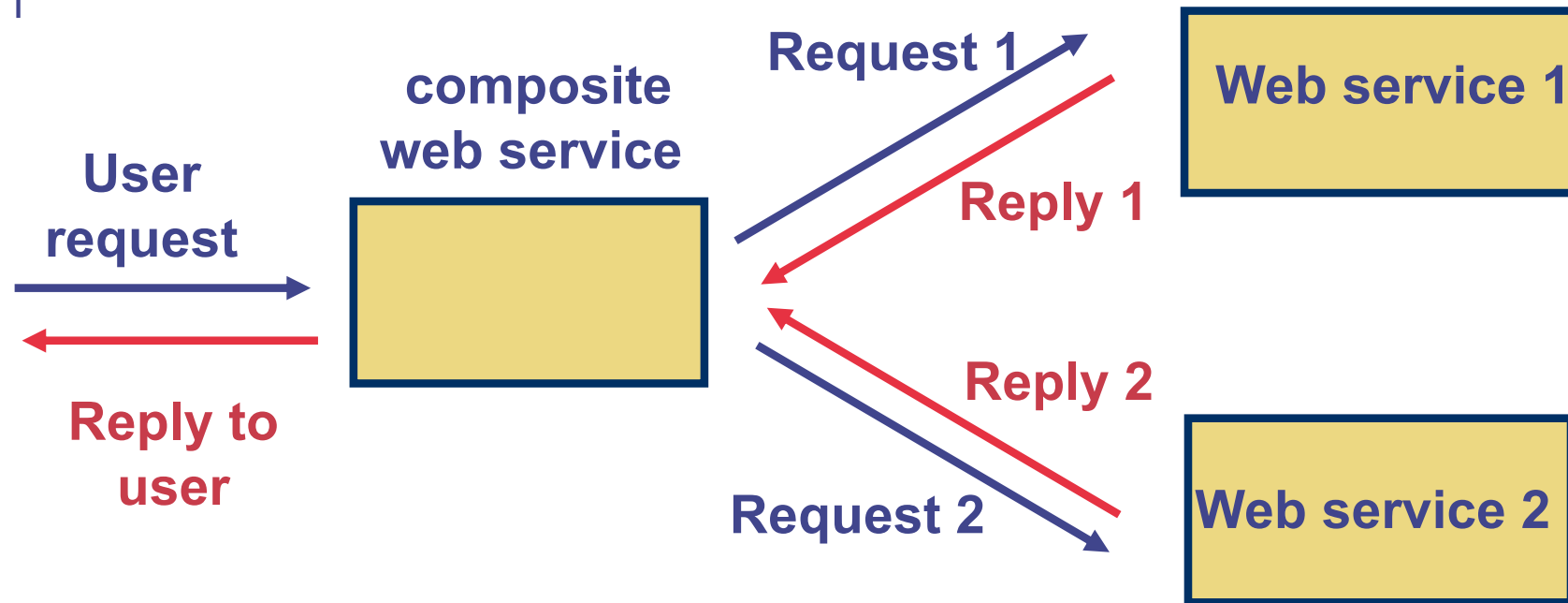
- Recall the definition of response time
- Response time of Web Service 1  
= Time at which composite web service receives Reply 1 *minus*  
Time at which composite web service sends Request 1
- Similarly, definition for Web Service 2

## A simple web service scenario (3)



- Assuming that:
  - Web service 1 has a response time distribution of
    - 0.2s with probability 0.5
    - 0.3s with probability 0.5
  - Web service 2 has a response time distribution of
    - 0.2s with probability 0.5
    - 0.3s with probability 0.5
- What is the average time that the composite web service has to wait until both replies are return?

## A simple web service scenario (4)



- What if the service time distribution is:
  - Web service 1 has a response time distribution of
    - 0.2s with probability 0.5
    - 0.3s with probability 0.5
  - Web service 2 has a response time distribution of
    - 0.2s with probability 0.5
    - 0.5s with probability 0.5
- What is the average time that the composite web service has to wait until both replies are return?



# Analysis scenario

---

- Lesson learnt: Slow web services can become the bottleneck for composite web service
- We consider Composite Web Services (illustration next slide)
  - With parallel invocation of  $N$  services
  - Web services 1 through  $N-1$  have a mean service time of  $S$  (exponentially distributed)
  - Web service  $N$  has a mean service time of  $g \times S$  (exponentially distributed)
  - The next service step can only be completed after all these  $N$  steps have been completed.

**Servers 1 to N-1 : mean response time =  $S$**

**Server N: mean response time =  $g \times S$**

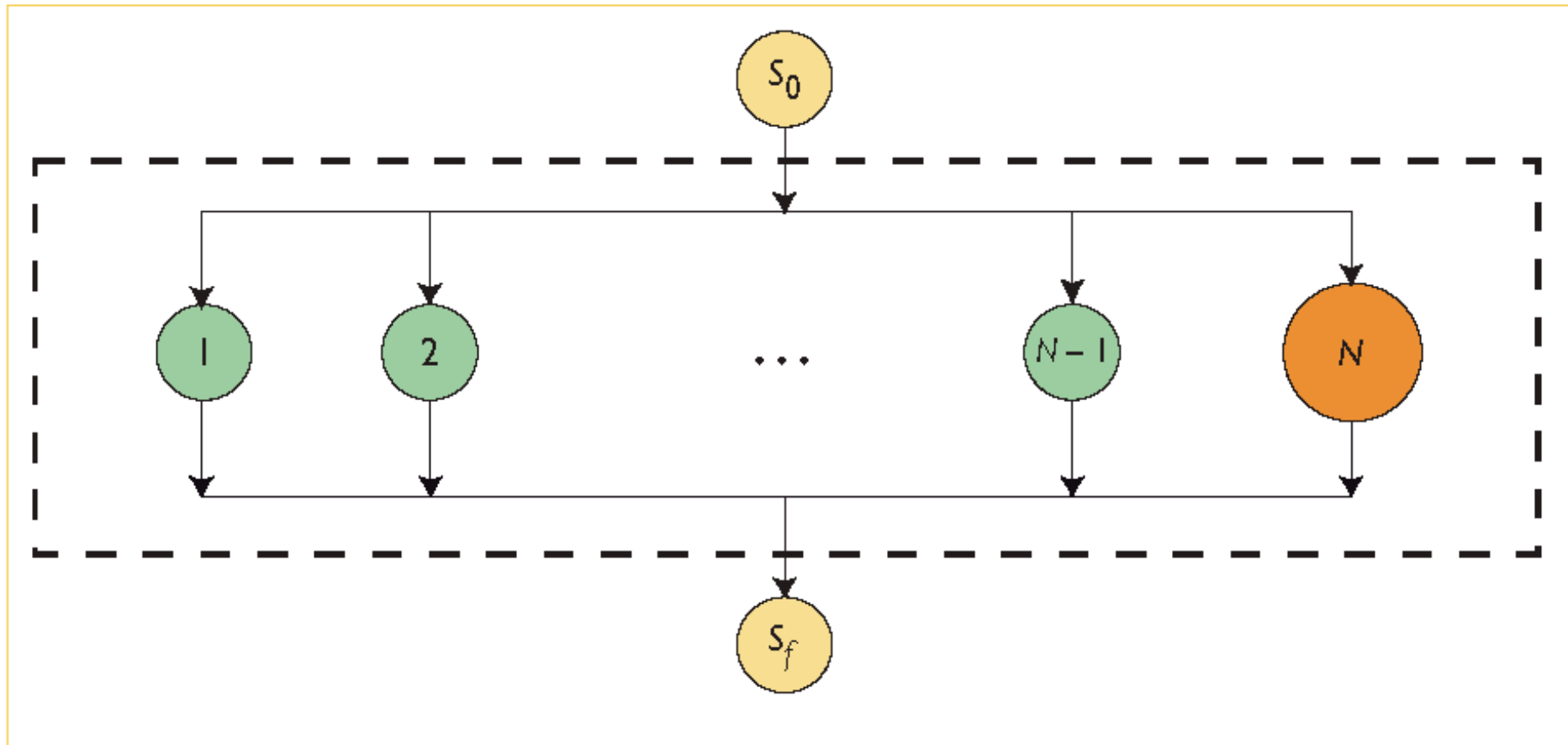


Figure 1. A composite Web service. After an initialization step  $S_0$ ,  $N$  Web services are invoked in parallel. Service  $N$  takes longer than the others, and the final step  $S_f$  can only be carried out after all  $N$  services have completed.

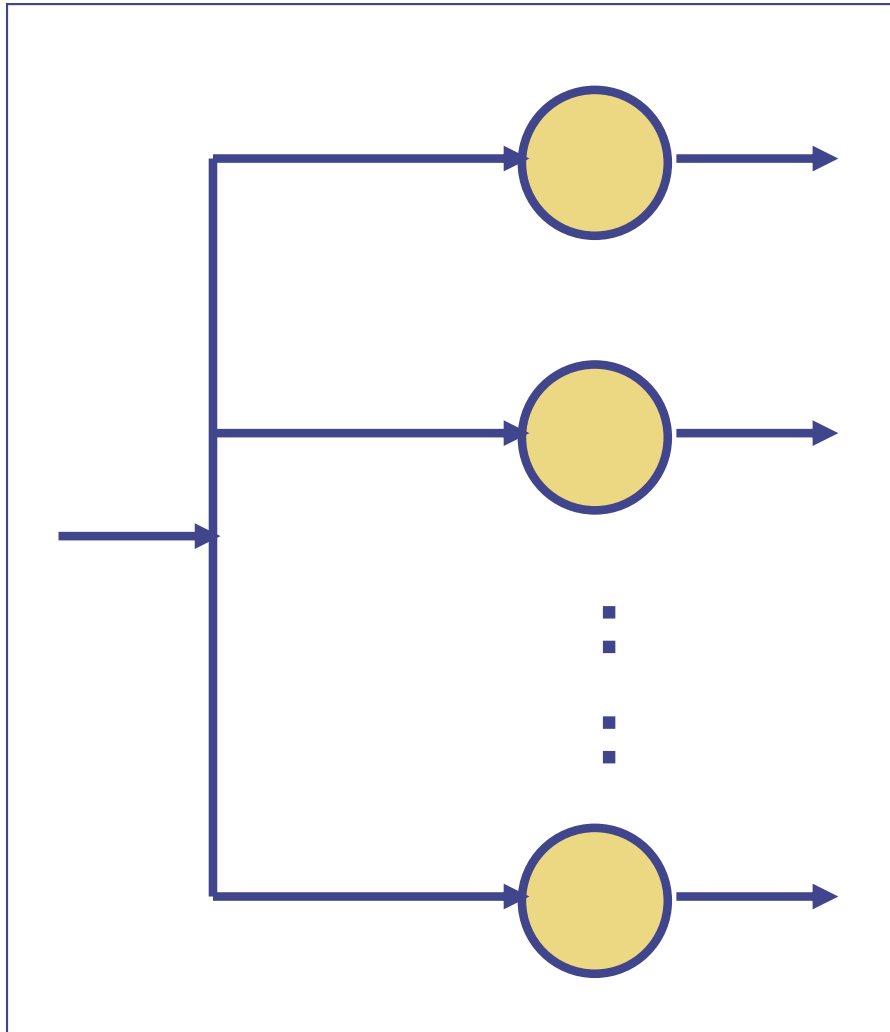
# Fork-join system

---

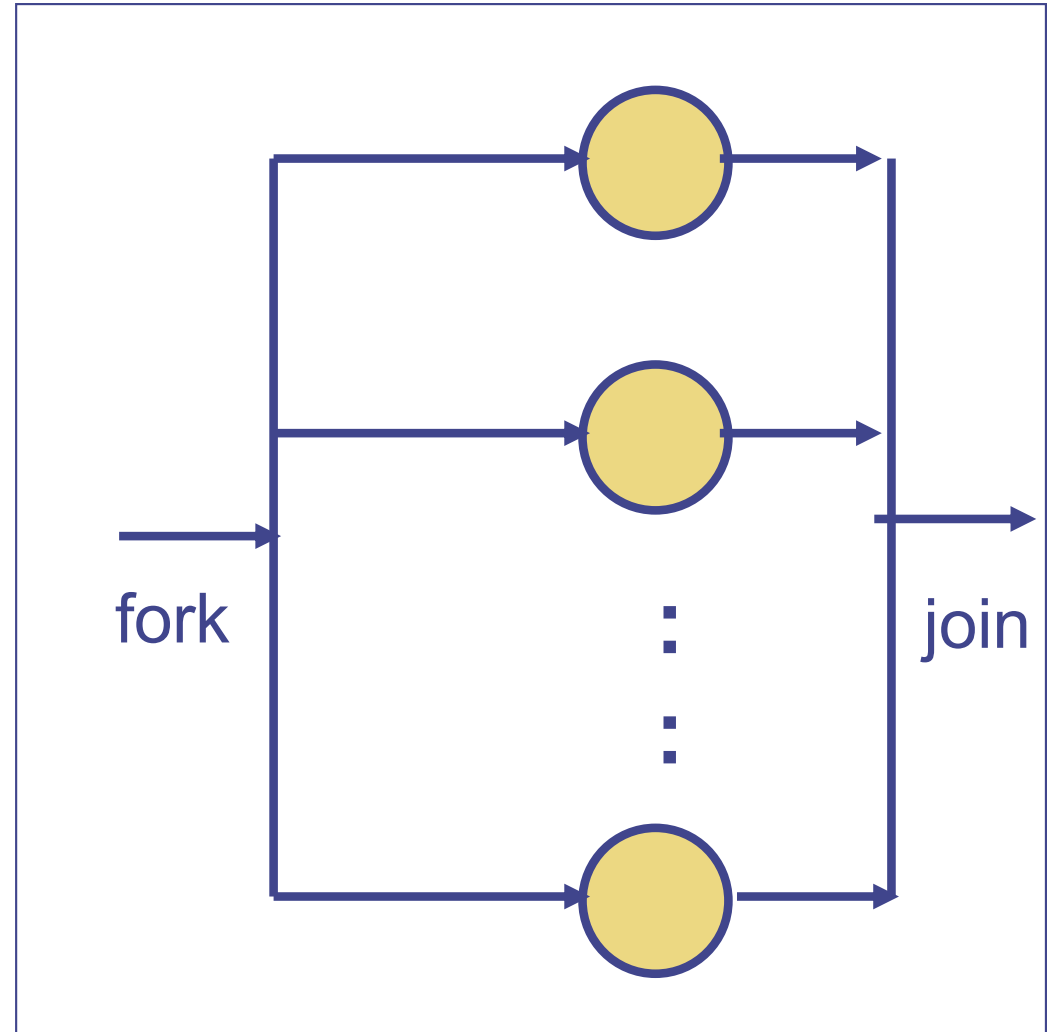
- The type of system described earlier is known as fork-join system
  - Fork is referring to the parallel invocation
  - All services must complete at the joining point before the next service can start

You've seen parallel processing before:

M/M/c queue



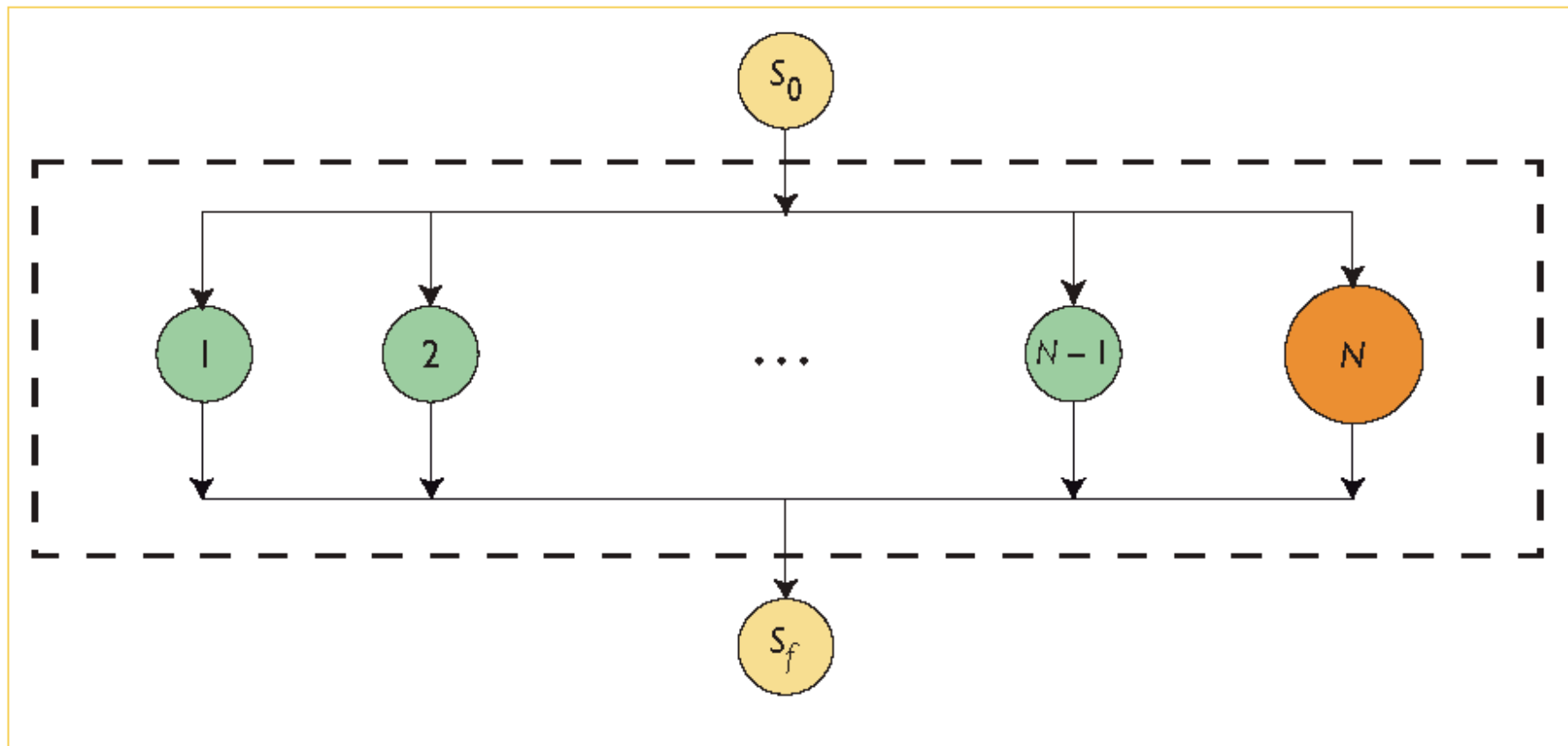
Fork-join queue



What is the difference between these two queueing networks?

**Servers 1 to N-1 : mean response time =  $S$**

**Server N: mean response time =  $g \times S$**



- We want to understand how  $g$  affects the response time of the composite web services

$T(g)$  = Response time of this system

## What is $T(1)$ ?

- In this case, all constituent web services have the same response time distribution
- If all mean response times are exponentially distributed with mean  $S$

$$T(1) = \underbrace{\left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}\right)}_{=H_N} S$$

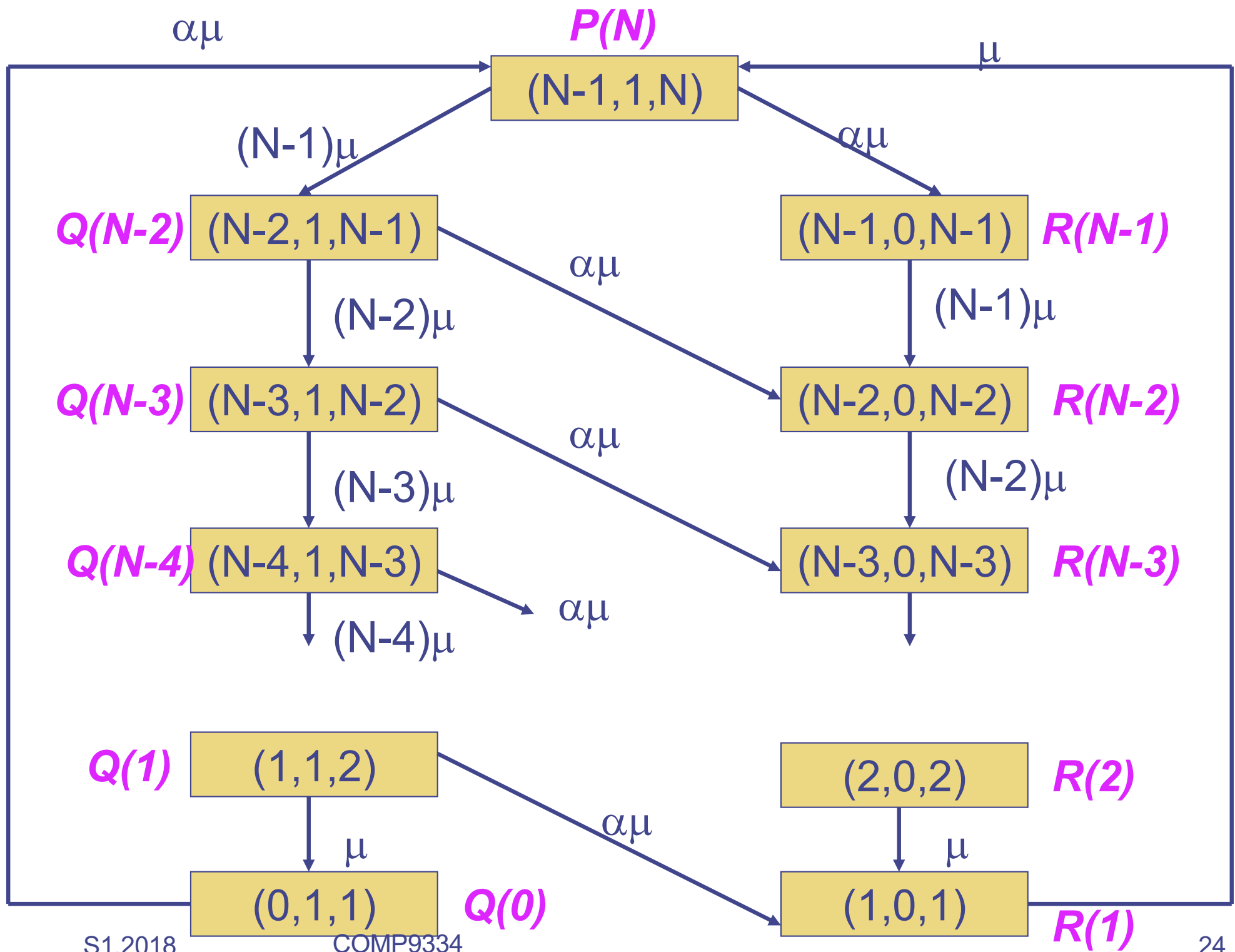
$H_N = N$ -th harmonic number

(We will prove this later.)

## How about $T(g)$ for $g > 1$ ?

---

- We use Markov chain.
- States  $(i,j,k)$ 
  - $i$  ( $i = 0, \dots, N-1$ ) is the number of web services still running in fast Web services
  - $j$  ( $j = 0, 1$ ) is the number of web services running on the slow Web service
  - $k$  ( $k = 1, 2, \dots, N$ ) is the number of web services yet to complete





$$T(g) = \frac{S}{\left(N - 1 + \frac{1}{g}\right)P(N)}$$


---

where

$$P(N) = \left[ 1 + \sum_{i=1}^{N-2} F(i) + gF(1) + V \right]^{-1},$$

$$V = \frac{1}{g} \sum_{j=1}^{N-1} \frac{1}{j} \sum_{i=j}^{N-1} F(i),$$

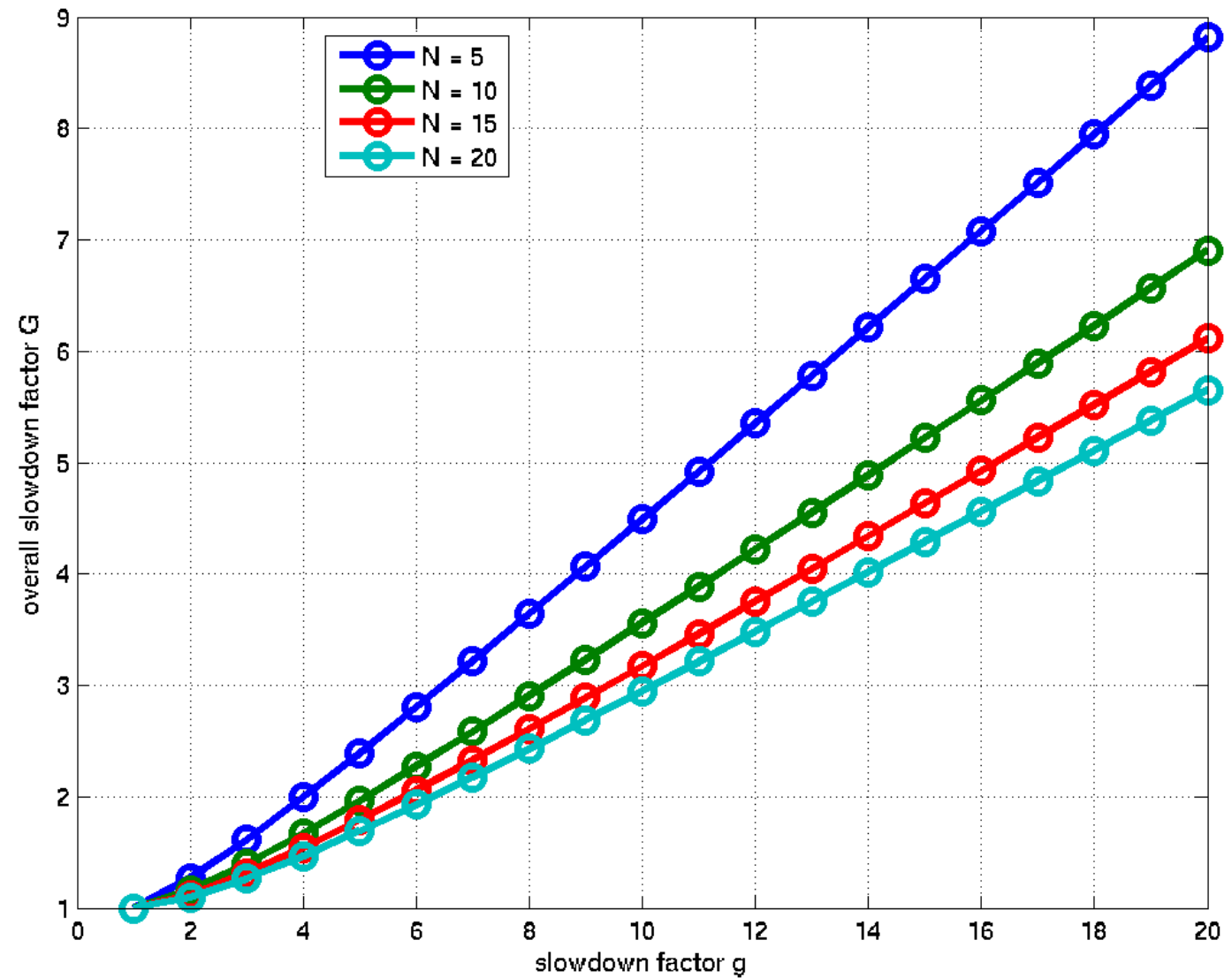
$$F(i) = \prod_{j=1}^{N-i-1} \frac{N-j}{N-j-1 + \frac{1}{g}}.$$

Note: When  $g = 1$ ,  
 $T(g) = H_N S$

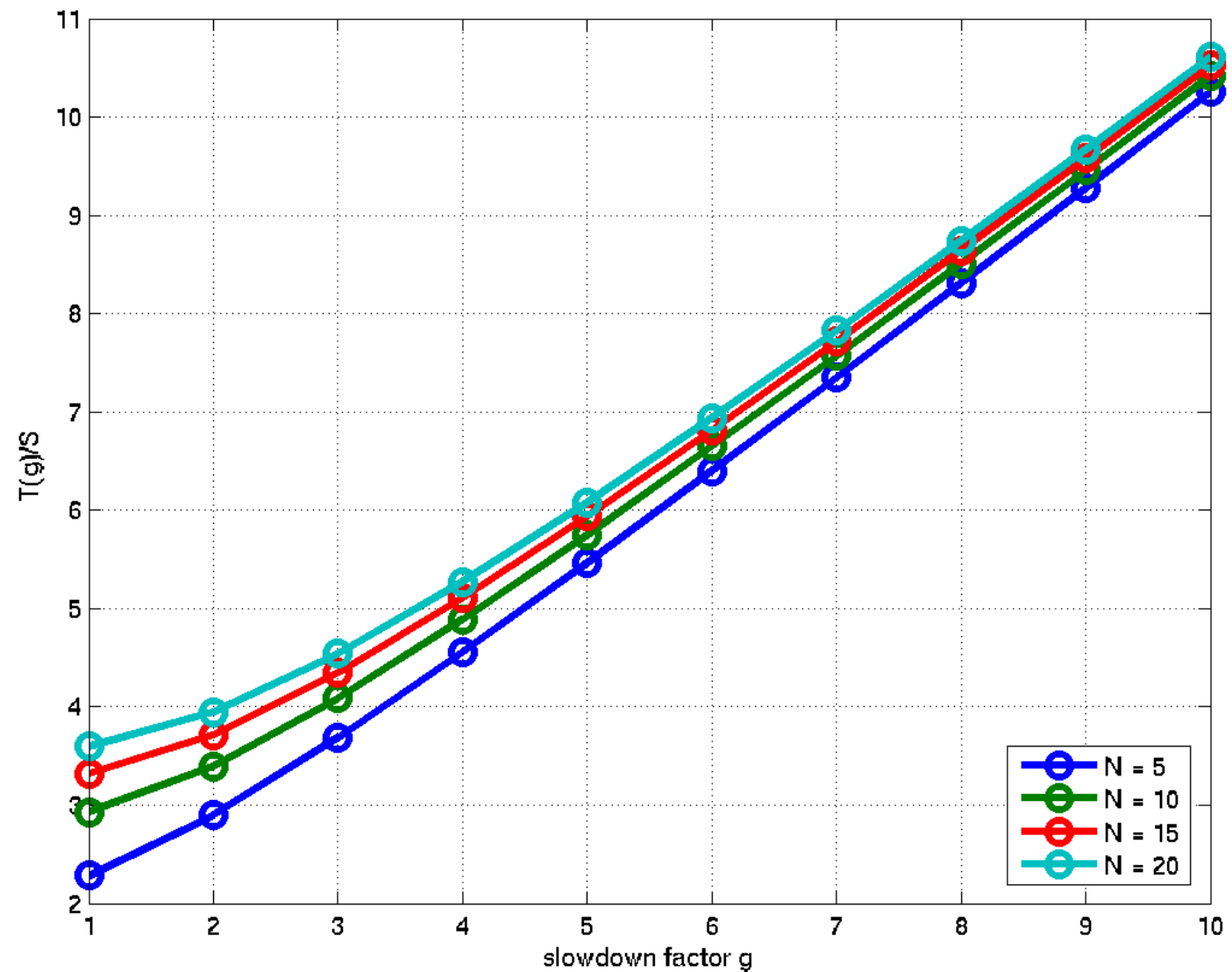
---

$$G = \frac{T(g)}{T(1)} = \text{Overall slowdown factor}$$

---

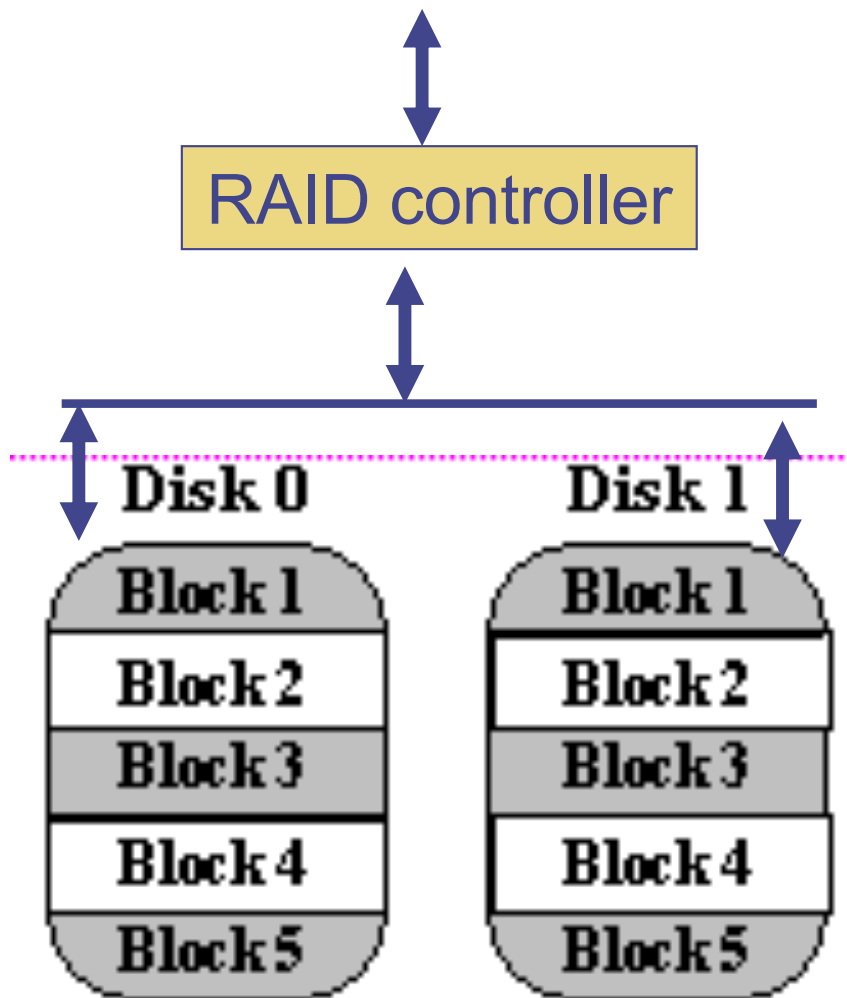


## How $T(g)/S$ varies with $g$ ?



## Other examples of fork-join QNs

- Disk array, e.g. RAID (Redundant Array of Independent Disks)



Example of RAID1  
Mirrored disks

# Fork-join in disk array

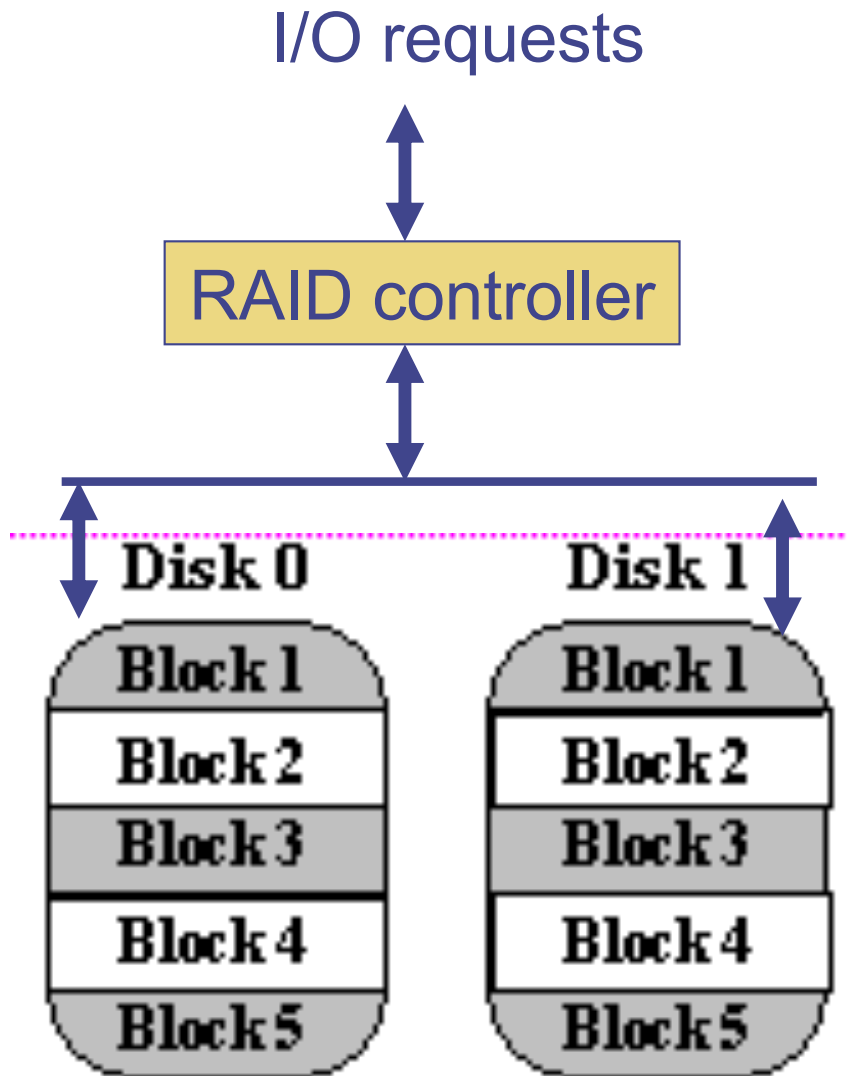
## Example 1

Read a file in parallel

1st half of the file from Disk 0

2nd half of the file from Disk 1

Need to wait for both halves of the file before the next operation



## Example 2

Write to disk.

Need to write to both disks (for consistency)

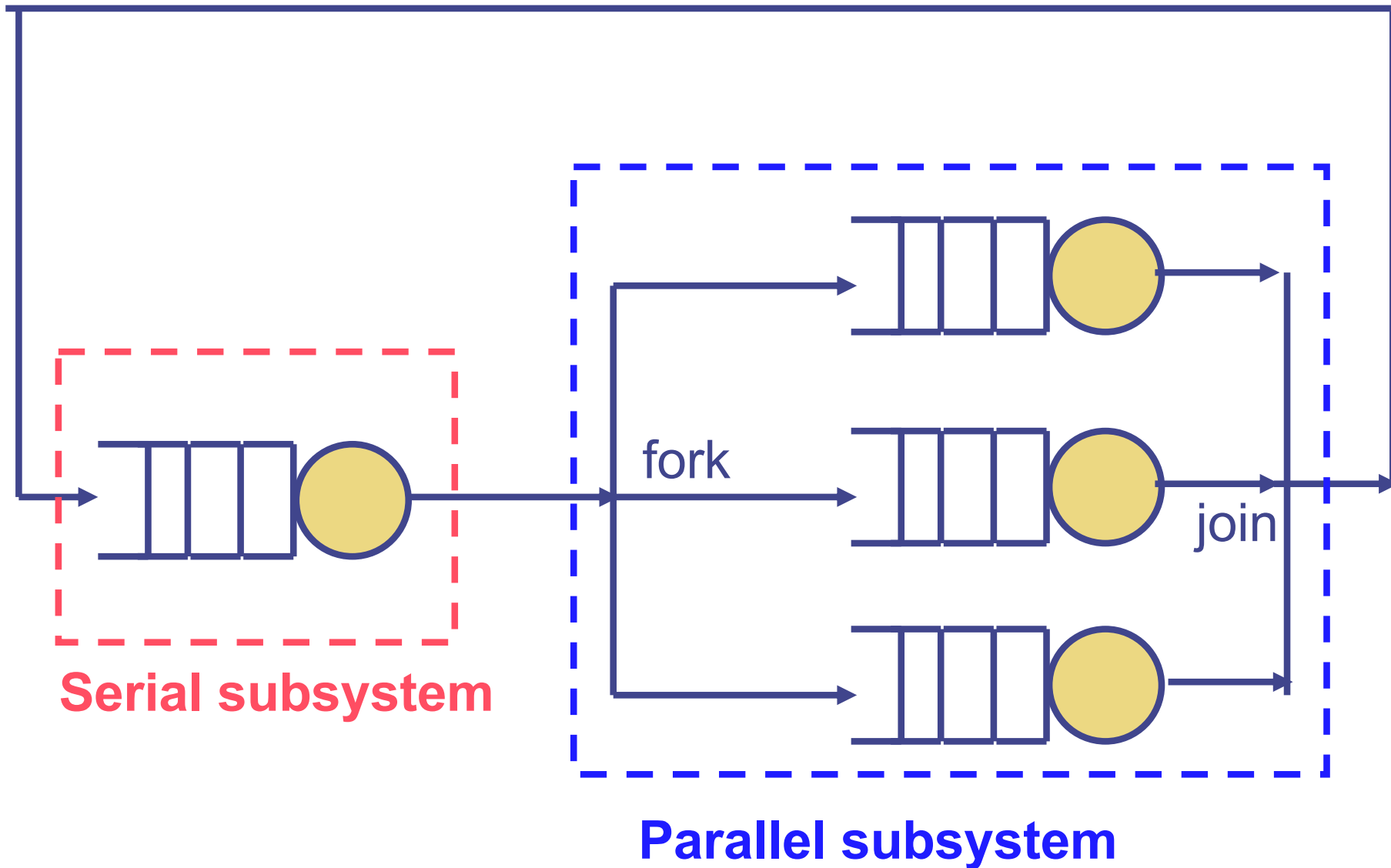
Need to wait for both disks to complete

# Fork-join queueing networks

---

- Exact results are hard to come by
- Approximate solution methods are used

# A Queueing network with a fork-join subsystem



# Approximate MVA for fork-join queueing networks

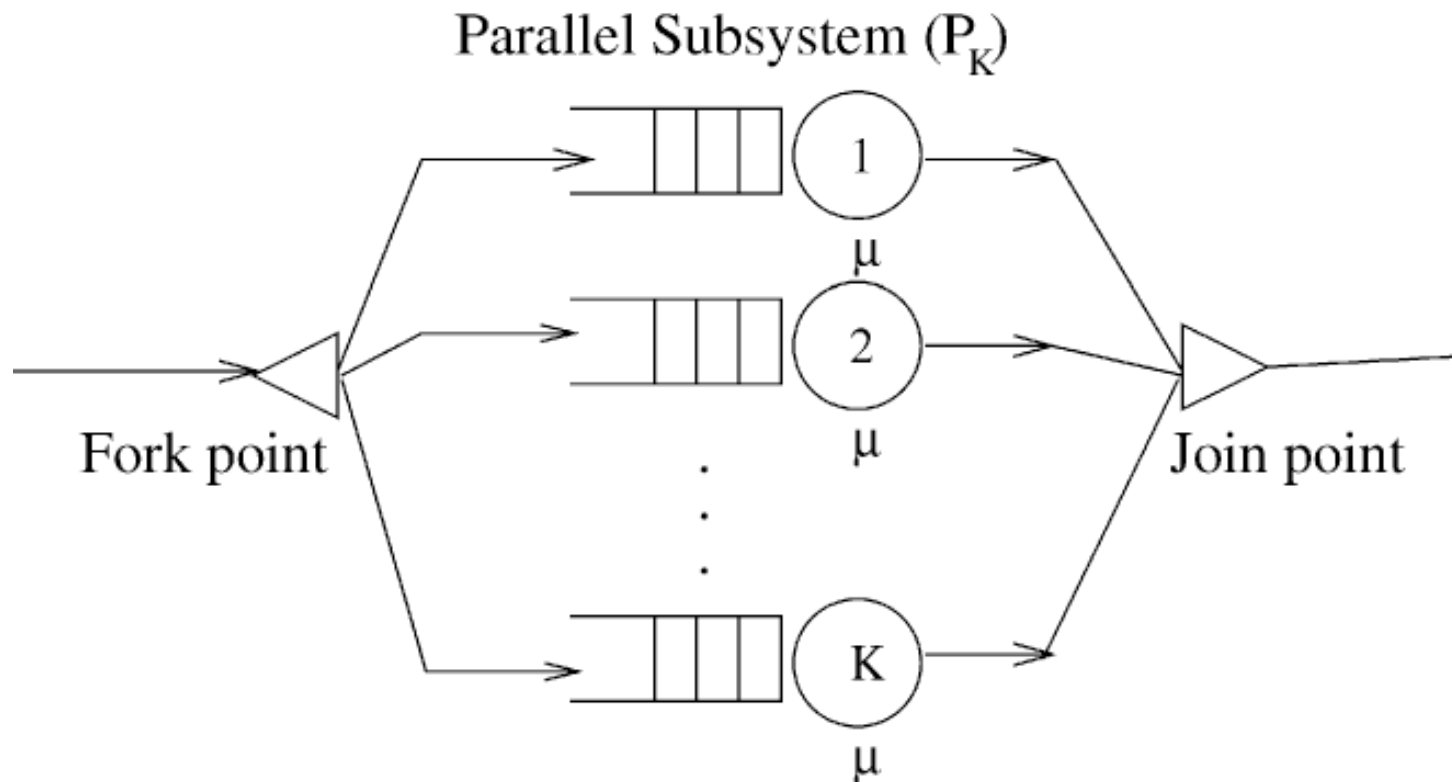
---

- For MVA with fork-join, the basic unit is a subsystem
  - A subsystem can be either a serial subsystem (= a device) or parallel one
    - A serial subsystem is a special case of parallel subsystem
  - C.f. The basic unit for MVA before is a device



## Arrival Theorem for Parallel Subsystems (1)

- Consider a parallel subsystem with  $k$  parallel service centres
- The average time each job requires at each service centre is  $S$  (exponentially distributed)



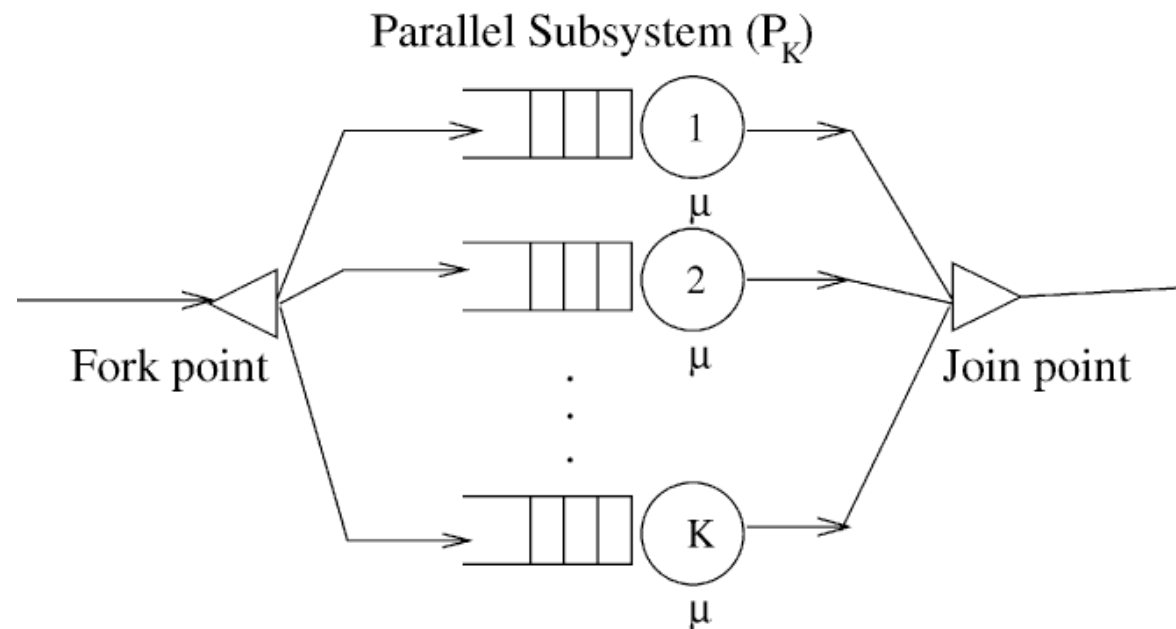
$$\mu = 1/S$$

## Arrival Theorem for Parallel Subsystems (2)

When there are  $n - 1$  jobs in the whole QN, the average number of jobs in the subsystem is  $z$ . When there're  $n$  jobs in the system



One of the  
 $n$  jobs (customers)



Waiting time =  $S \times z$ ; Service time =  $S \times H_k$

$\Rightarrow$  Response time =  $S \times (H_k + z)$

Note that if  $k = 1$ , the subsystem is serial and is identical to a device in MVA analysis that we have seen before.

$$\begin{aligned}\text{Response time} &= S \times (H_1 + z) \\ &= S \times (1 + z) \\ &\quad (\text{Since } H_1 = 1)\end{aligned}$$

This is the same form of arrival theorem that we've seen before.

## Notation:

---

$I$  = Number of subsystems in the QN

$S_i$  = Avg. service time of a station in subsystem  $i$

$k_i$  = # parallel stations in subsystem  $i$

$R_i(n)$  = Response time at subsystem  $i$   
when there're  $n$  jobs in the QN

$\bar{n}_i(n)$  = Avg. # of jobs at subsystem  $i$   
when there're  $n$  jobs in the QN

$V_i$  = Visit ratio of subsystem  $i$

---

## MVA for fork-join systems:

Mean # jobs in each subsystem

$$\bar{n}_i(n-1)$$

(n-1) jobs in system

n jobs in system

$$R_i(n) = S_i \times (H_{k_i} + \bar{n}_i(n-1))$$

Mean response time of each subsystem

$$R_i(n)$$

$$X_0(n) = \frac{n}{R_0(n)} = \frac{n}{\sum_{i=1}^I V_i R_i(n)}$$

Throughput of the system

$$X_0(n)$$

$$\bar{n}_i(n) = V_i \times X_0(n) \times R_i(n)$$

Mean # jobs in each subsystem

$$\bar{n}_i(n)$$

# Example

- A system consists of a processor and 2 disk arrays
- Disk arrays operate under synchronous workload
  - Transactions are blocked until I/O are completed

	Service demand	# parallel systems
Processor	0.01	1
Disk array 1	0.02	2
Disk array 2	0.03	3

**What is the system response time when there are 50 transactions? How many transactions can the system have if the system response time should not exceed 1s?**

## Exercise

---

- The MVA algorithm on p.37 assumes that you have both visit ratios  $V_i$  and mean service time  $S_i$  available
- You may recall that service demand  $D_i = V_i * S_i$
- Now, let us assume that you are only given the service demands  $D_i$ . That is, you know only  $D_i$  but you do not know  $V_i$  and  $S_i$ . How can you modify the MVA algorithm on p.37 so that it can work with knowing service demands only?

# References (1)

---

- Web services
  - D. Mensace et al. Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures," *Journal of Parallel and Distributed Computing*, Vol. 28 (1), July 1995, pp. 1-18.
  - D. Mensace, "QoS Issues in Web Services," *IEEE Internet Computing*, November/December 2002, Vol. 6, No. 6.
  - D. Mensace, "Response Time Analysis of Composite Web Services," *IEEE Internet Computing*, January/February 2004, Vol. 8, No. 1
  - D. Mensace, "Composing Web Services: A QoS View," D. Menasce, *IEEE Internet Computing*, Vol. 8., No. 6, Nov/Dec 2004.
  - These papers can be downloaded from the course website (use your CSE password)
    - We didn't cover the last paper but it's well worth a read.
- Derivation of Markov chain on pp. 23-25 is further explained in the file *forkjoin\_mc.pdf*



## References (2)

---

- Fork-join MVA
  - Menasce et al., "Performance by desing". Section 15.6.
- Addition references outside the scope of this course
  - Tutorial on RAID <http://www.slcentral.com/articles/01/1/raid/>