



LUND
UNIVERSITY

Users and back-ends

MARIA KIHLM



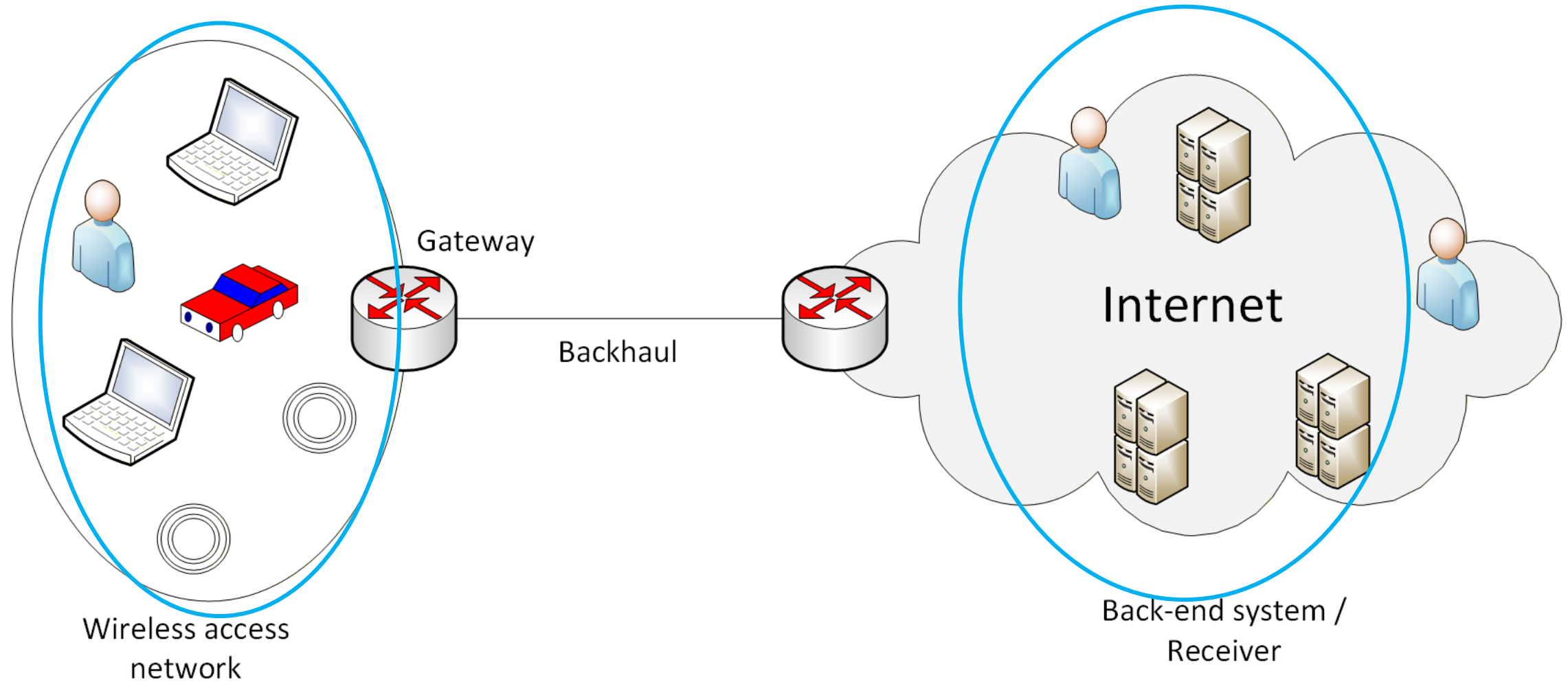
To do!

- Project
 - Project plan and client organisation scenario, deadline on Friday
- Tutorial
 - Review of probability theory and statistics, tomorrow

Literature

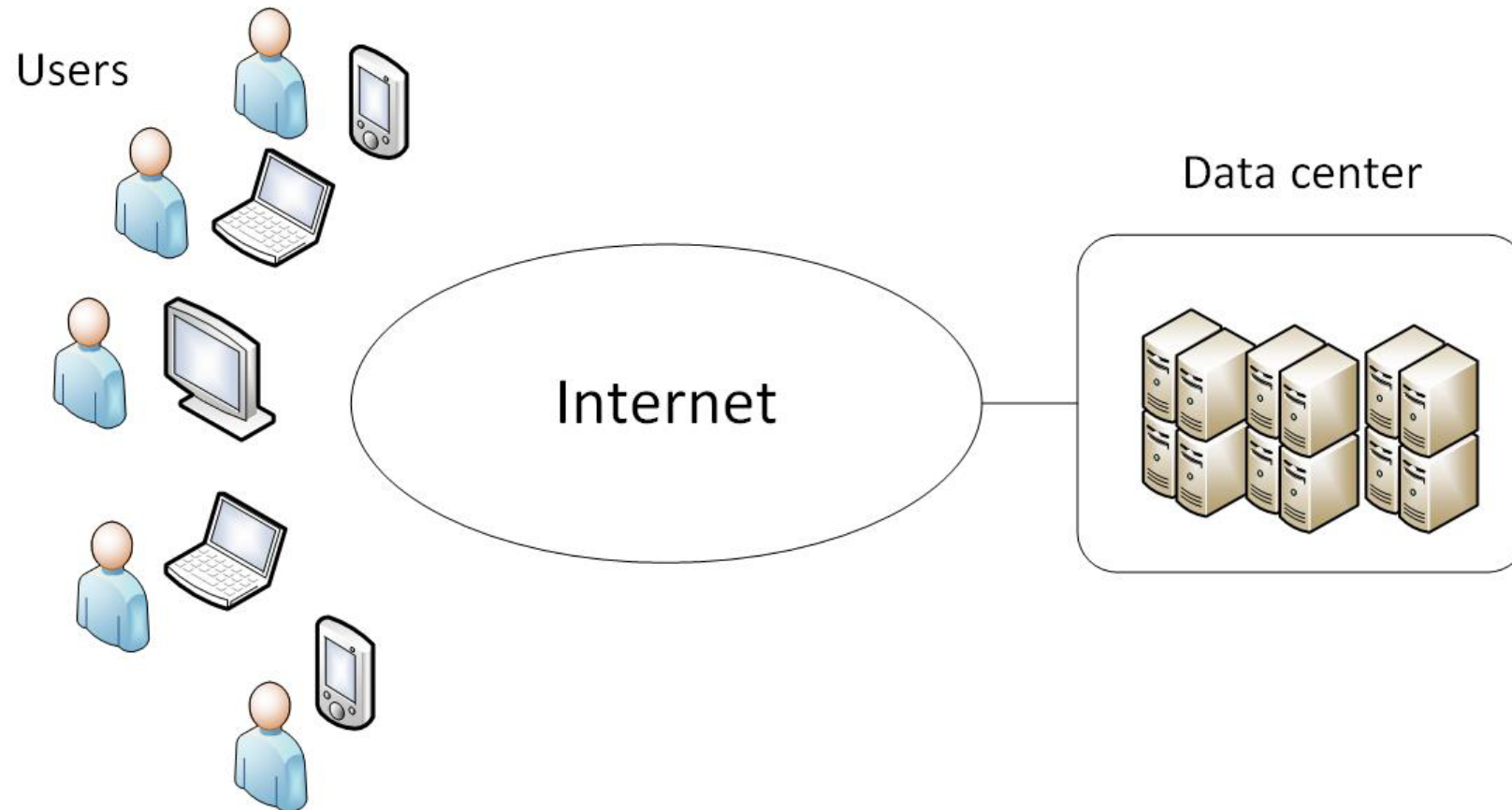
- **M. Kihl, "Queuing systems"**, Lecture notes, 2004.
 - Chapter 1-3, 4.1, 5.1
- **W. Tärneberg, "An introduction to Edge and Fog computing"**, In The confluence of Cloud computing, 5G, and IoT in the Fog, PhD Thesis, Lund University, 2019.
 - Chapter 1 and 4 (Chapter 2 and 3 are optional)
- **H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo and M. Dianati, "Popularity-Based Video Caching Techniques for Cache-Enabled Networks: A Survey₁"** in IEEE Access, vol. 7, pp. 27699-27719, 2019.
 - Section I (the rest is optional)

Lecture today: Users and back-ends



Today's use case: Client-server applications

For example: Users request web pages from a web site placed in a data center.



Web server systems: Multi-tier architectures

Today's web server systems are designed using **multi-tier architectures**, where each tier has specific tasks.

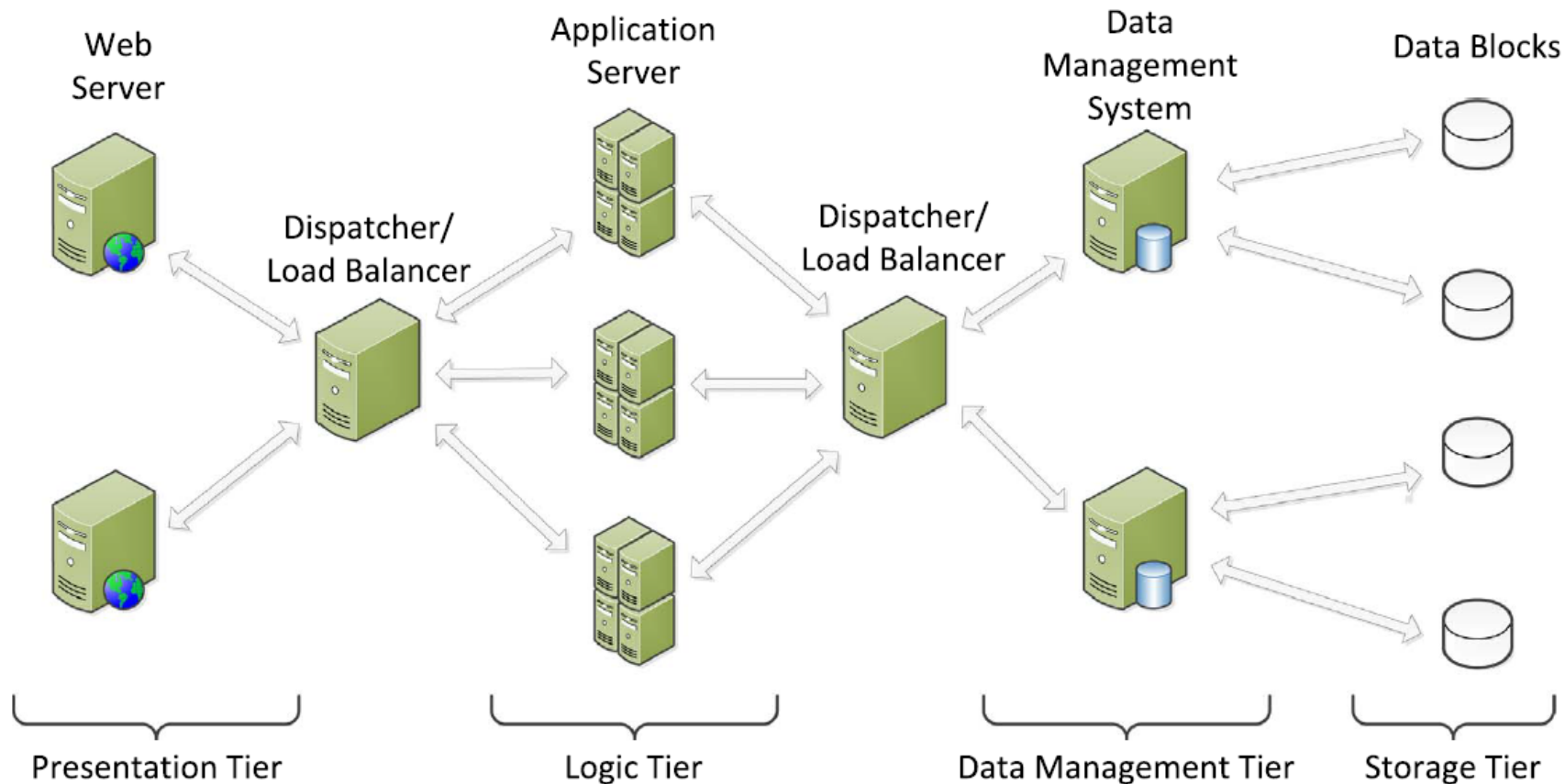


Fig. 2. Multi-tier application architecture.

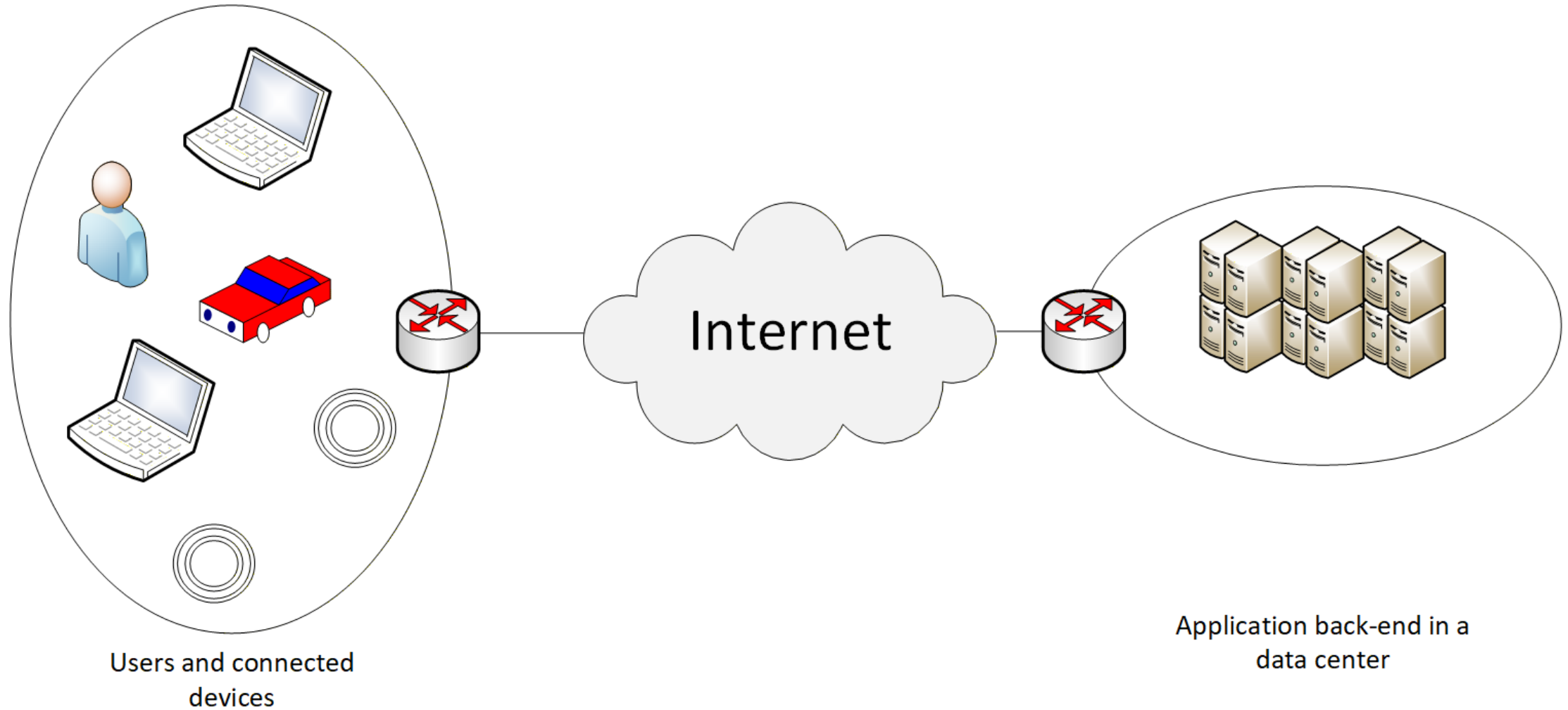
Today's scenario: WebFilmz

Webfilmz is a video streaming company. Users pay a monthly subscription fee to gain access to the company's catalogue of TV shows and movies, which they can then stream via the internet. The catalogue currently has about 5000 videos in total. WebFilmz is a relatively new company still trying to build market share, so it is very important that new users are provided with a consistently high user perceived quality so that they will continue to subscribe. WebFilmz currently has 15 million subscribers, located in all EU countries (it is not available outside the EU yet), and their subscriber distribution approximately follows the general population distribution.

WebFilmz has so far been serving all their content from a central data center located in Copenhagen. However, they now wish to investigate a more sophisticated solution. The solution should deliver video streaming content to WebFilmz subscribers with high user perceived quality.

Data centers and cloud services

Data centers



Data centers

- A data center is a physical installation that organizations use for computing and data storage.
- A data center is based on a **network with computing servers and storage resources**.
- The majority of data centers are used for **cloud computing and storage services**.



Cloud computing and storage services

- A cloud system provides on-demand availability of computer system resources, especially data storage (cloud storage) and computing power using a virtualization technology.

Two basic types of cloud services:

- **Cloud computing**
 - Virtualized computing capacity
 - **Examples of providers:** Azure, AWS, Google etc.
- **Cloud Storage**
 - Virtualized storage capacity
 - **Examples of providers:** Dropbox, Google Drive, OneDrive etc

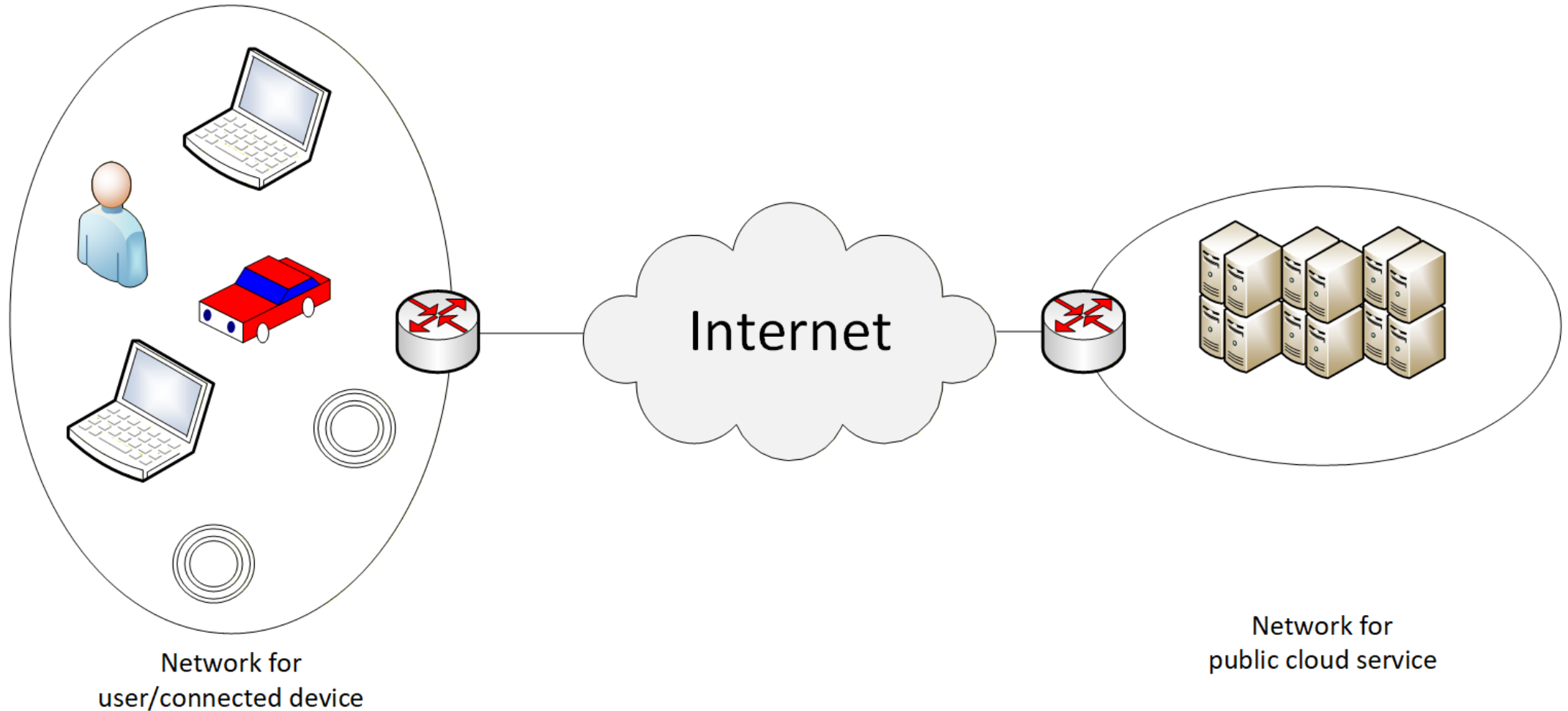
However, the concepts are not uniquely defined. Cloud storage can be said to be part of Cloud computing.

Cloud deployment models

There are three main models for how to deploy a cloud system:

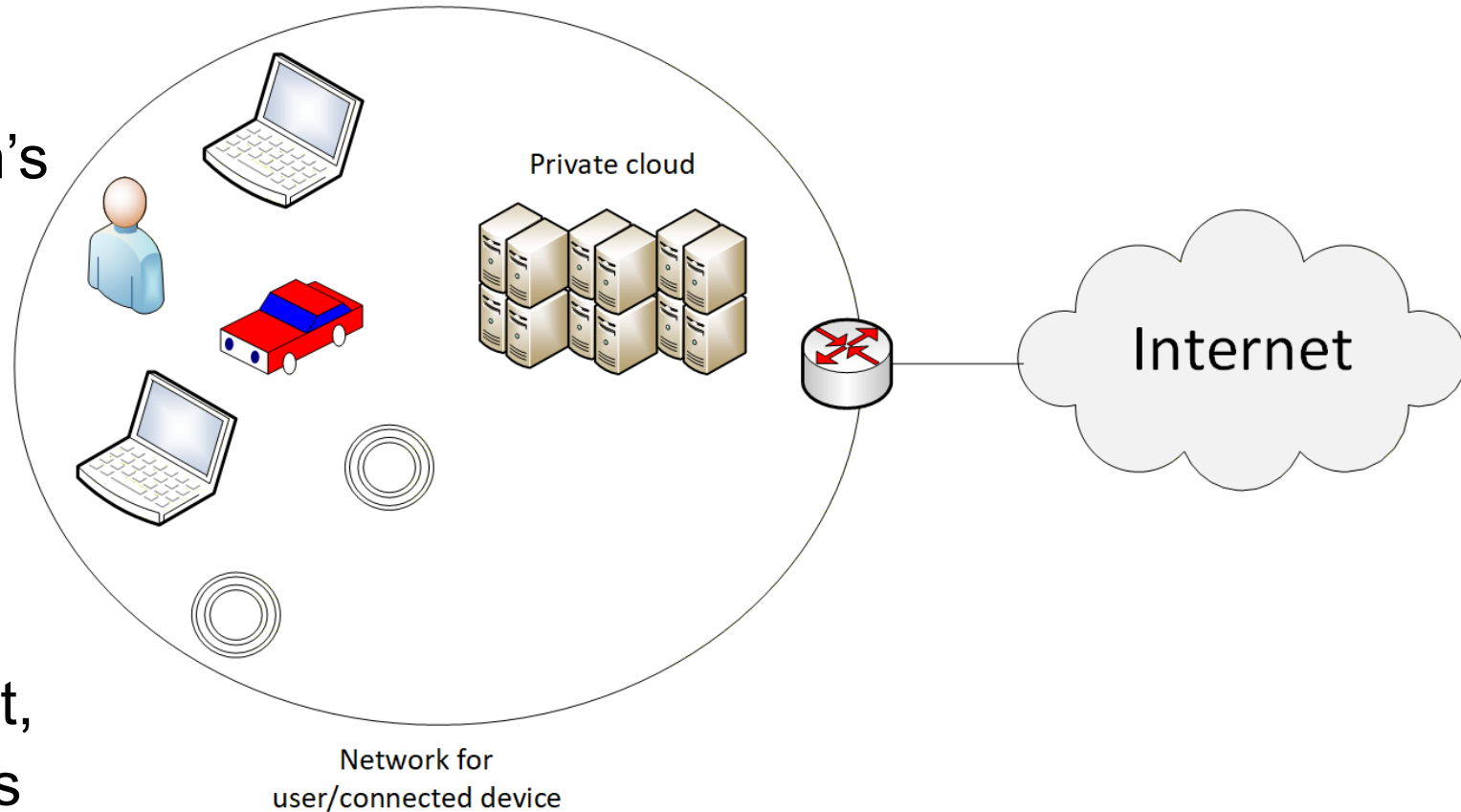
- **Public cloud:** Public clouds deliver resources, such as compute, storage, network, develop-and-deploy environments, and applications over the Internet. They are owned and run by third-party cloud service providers like Google Cloud, AWS etc.
- **Private cloud:** Private clouds are built, run, and used by a single organization, typically located on-premises.
- **Hybrid cloud:** Environments that mix a private computing environment with one or more public clouds are called hybrid clouds.

Basic network architecture for applications using Public cloud services



Basic network architecture for applications using a Private cloud

- Using a Private cloud, both the users and the cloud infrastructure can in theory stay within the organization's own network.
- However, in practice this depends on how the deployment is done.
- The private cloud may be accessible over the Internet, but within the organization's domain.



Distributed computing architectures

By distributing the computing capacity to more places, the capacity will come closer to the users and bottlenecks can be avoided.

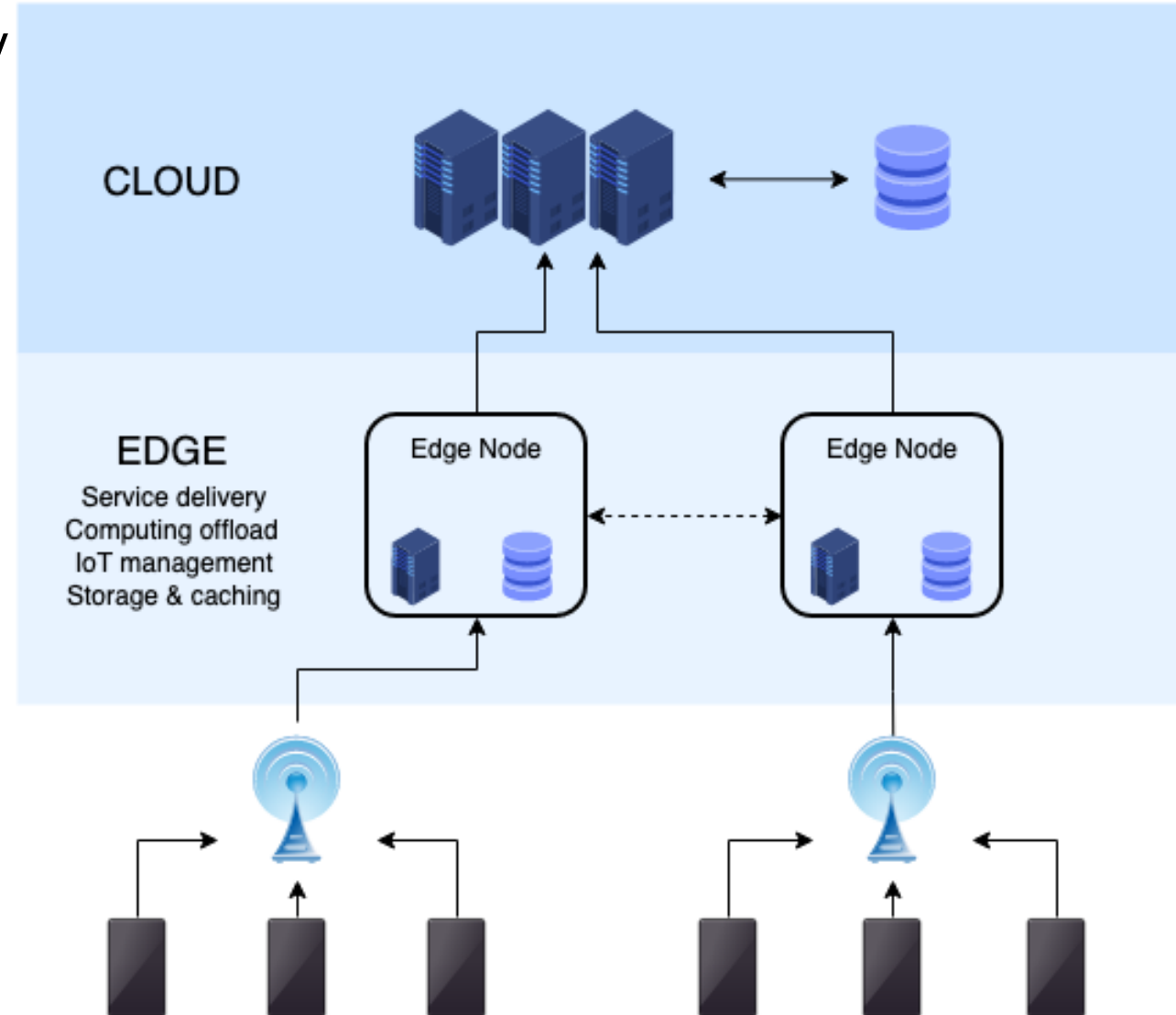
Two basic architecture solutions:

- **Edge computing architectures** for cloud computing.
- **Content delivery networks (CDNs)** for cloud storage aimed at streaming applications.

Edge computing = Move the cloud closer

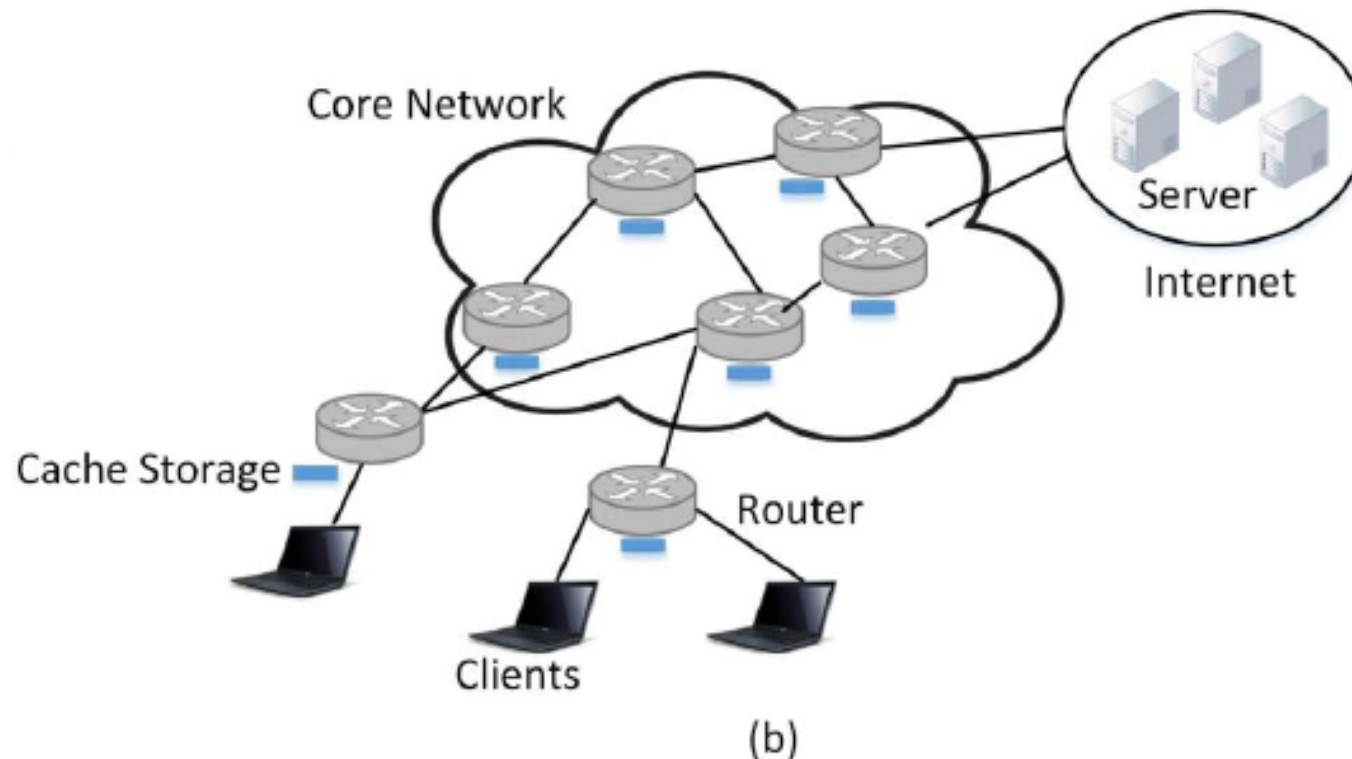
Edge computing can minimize the latency from when data is produced to when a response is required.

- Near-real-time action with the equipment.
- Data stays near its source, reducing security risks.
- Edge computing solutions are usually based on **Private or Hybrid cloud infrastructures**.



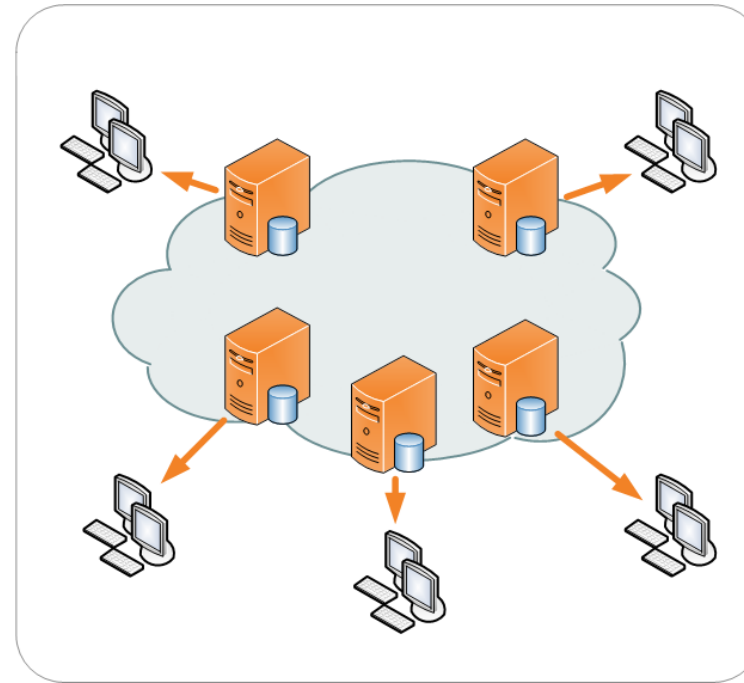
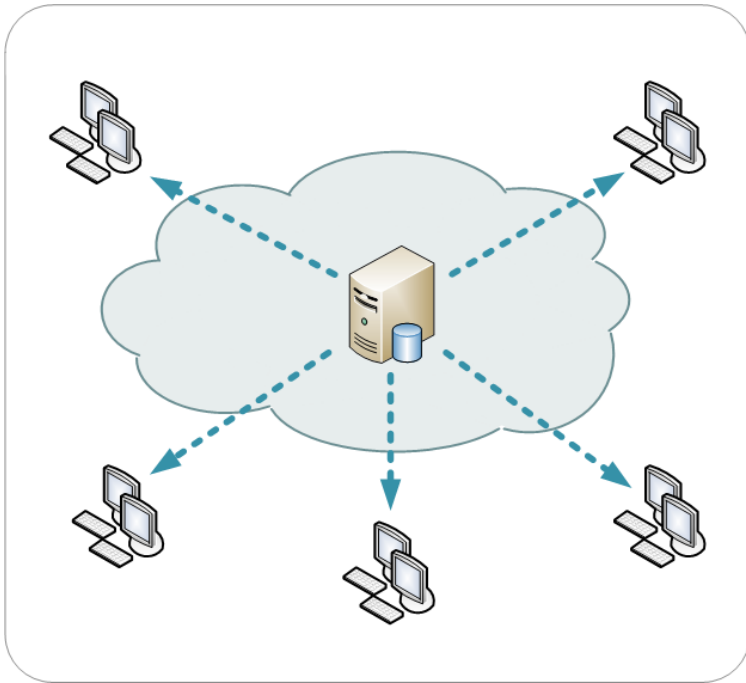
Content Delivery Networks (CDNs)

With a Content Delivery Network (CDN), content can be distributed to cache servers (can be cloud data centres) placed in the Internet. There are large CDN operators (for example Akamai, Limelight) that host the cache servers.



Reasons for a CDN architectures

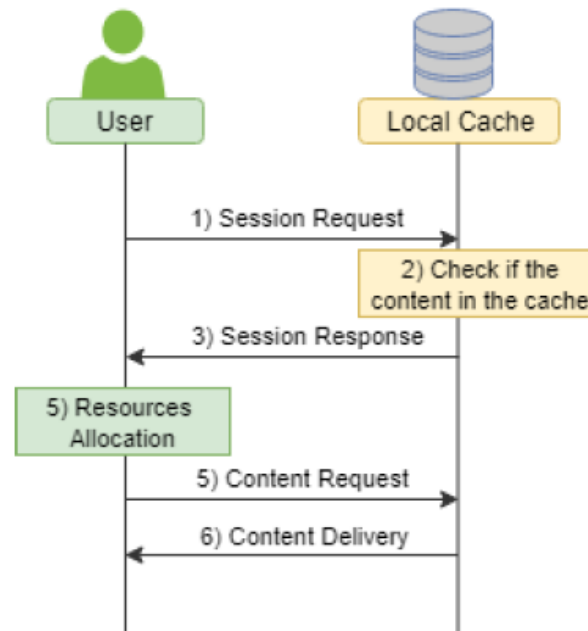
- CDNs decrease the network and server load by distributing the content to more servers.
- CDNs decreases the download latencies since content is closer to the users.



Content delivery from cache

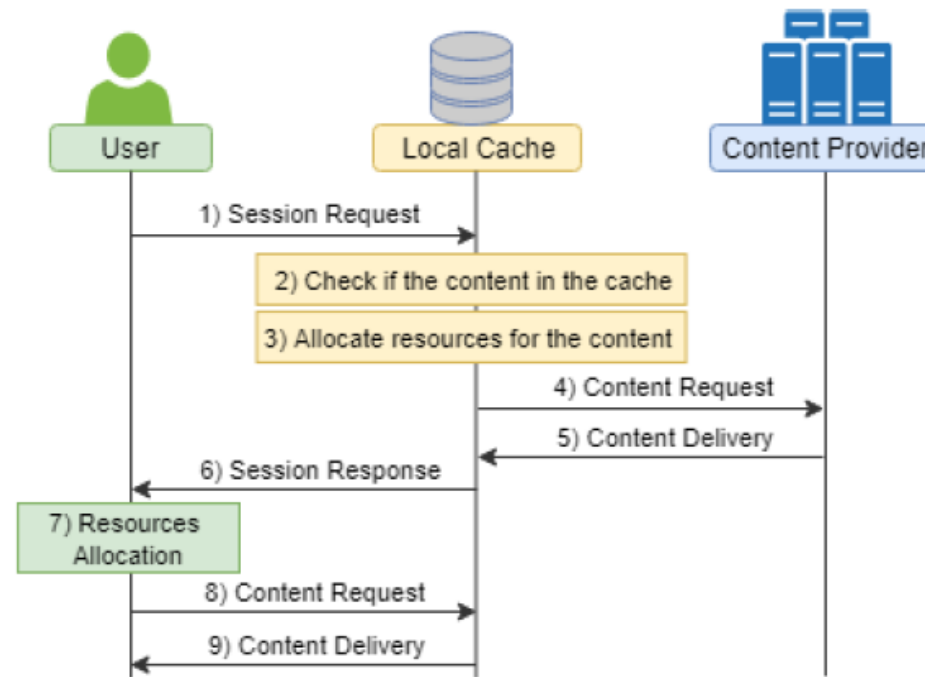
When a user requests a specific content, the application sends this request to the cache.

1. If the content is in the cache, the content is delivered to the user directly from the cache.



Content delivery from the origin server

2. If the content is not in the cache, the request is forwarded "upwards" in the CDN system. All content can always be found in the root server, that is the origin server.



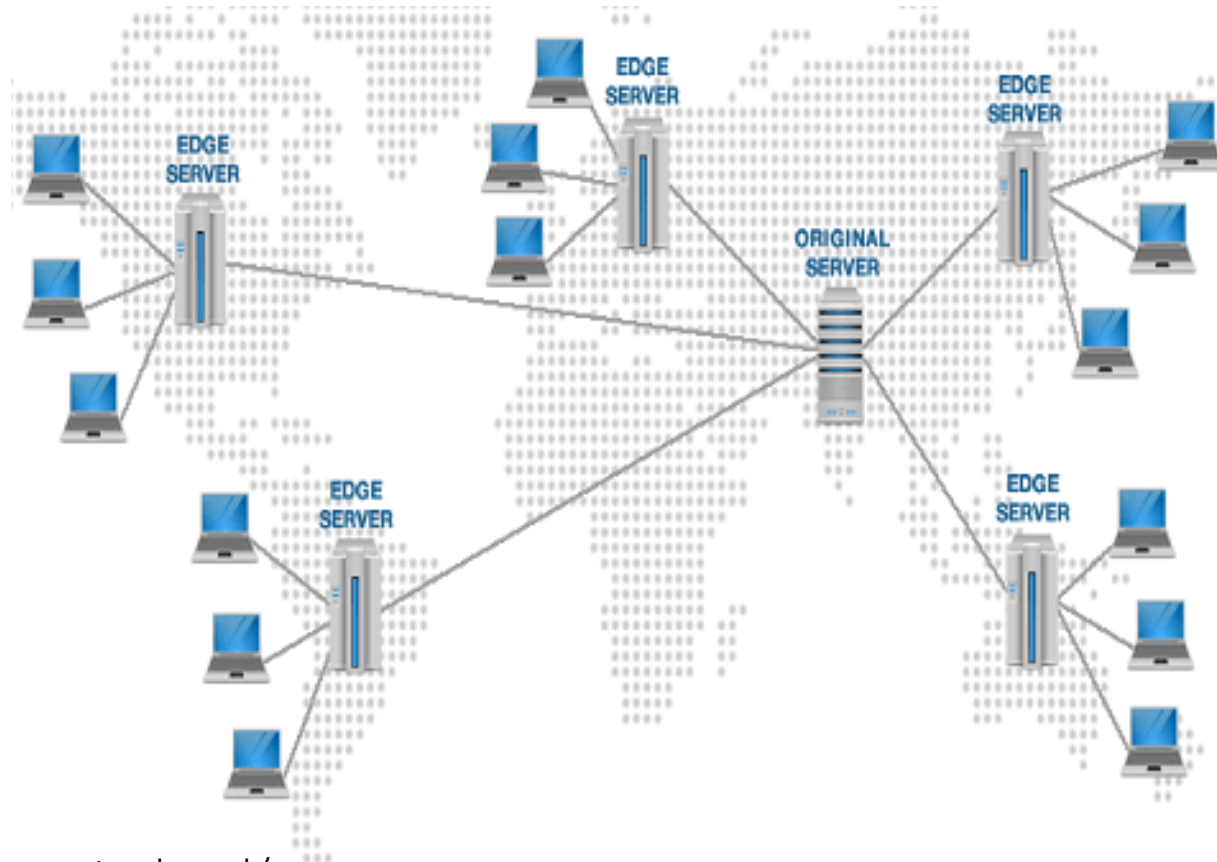
WebFilmz – Solution for content delivery?

Webfilmz is a video streaming company. Users pay a monthly subscription fee to gain access to the company's catalogue of TV shows and movies, which they can then stream via the internet. The catalogue currently has about 5000 videos in total. WebFilmz is a relatively new company still trying to build market share, so it is very important that new users are provided with a consistently high user perceived quality so that they will continue to subscribe. WebFilmz currently has 15 million subscribers, located in all EU countries (it is not available outside the EU yet), and their subscriber distribution approximately follows the general population distribution.

WebFilmz has so far been serving all their content from a central data center located in Copenhagen. However, they now wish to investigate a more sophisticated solution. The solution should deliver video streaming content to WebFilmz subscribers with high user perceived quality.

Solution suggestion

The solution can be some type of content delivery network. Cache servers can be deployed at suitable locations within the EU, or capacity can be bought from one of the world-wide CDN operators. The origin server can be located in Copenhagen.



Performance modelling and analysis of server systems

Server systems

For optimal design of server systems, we first need to estimate their expected performance. (This is also valid if we buy a cloud service).

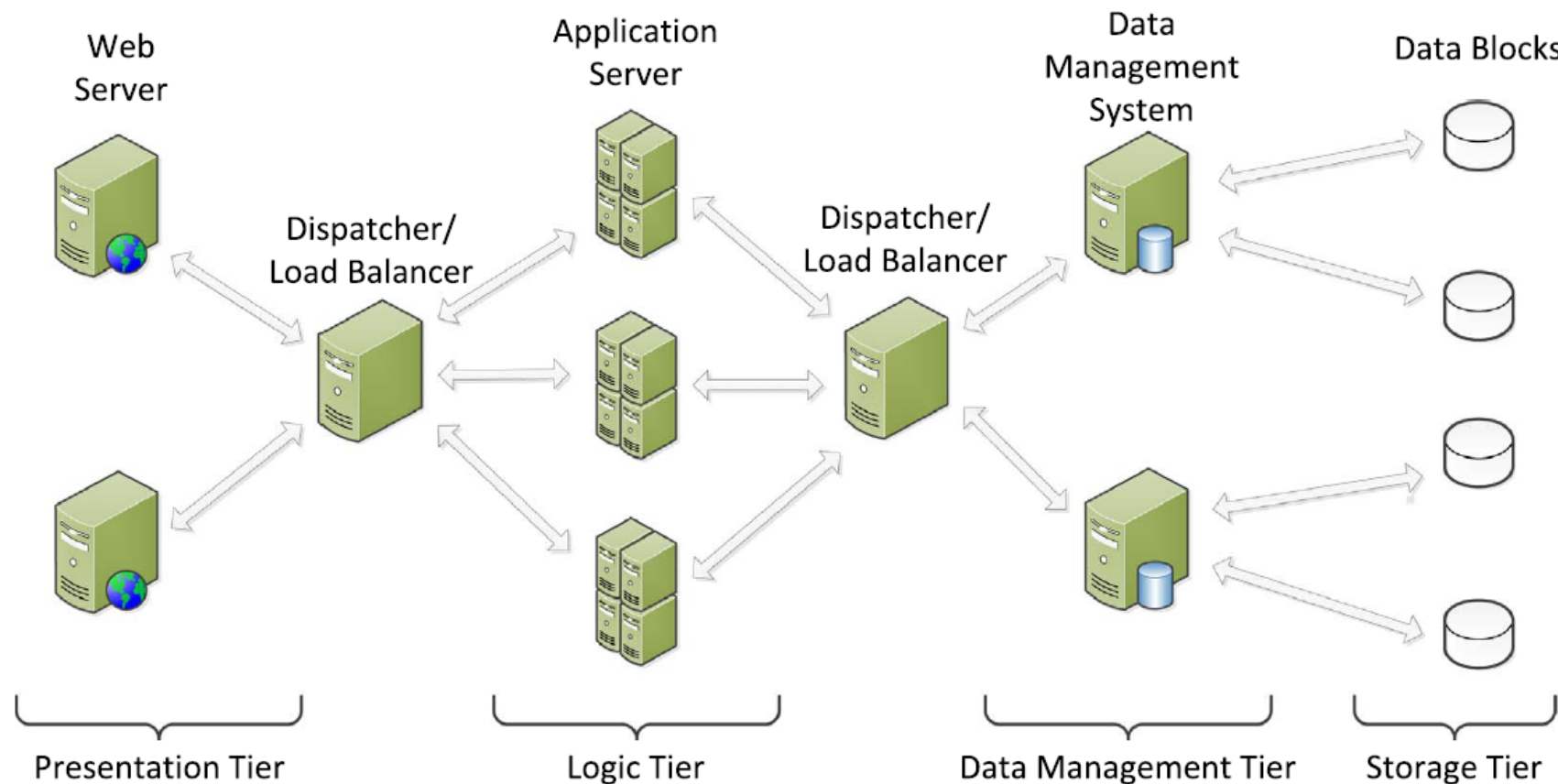
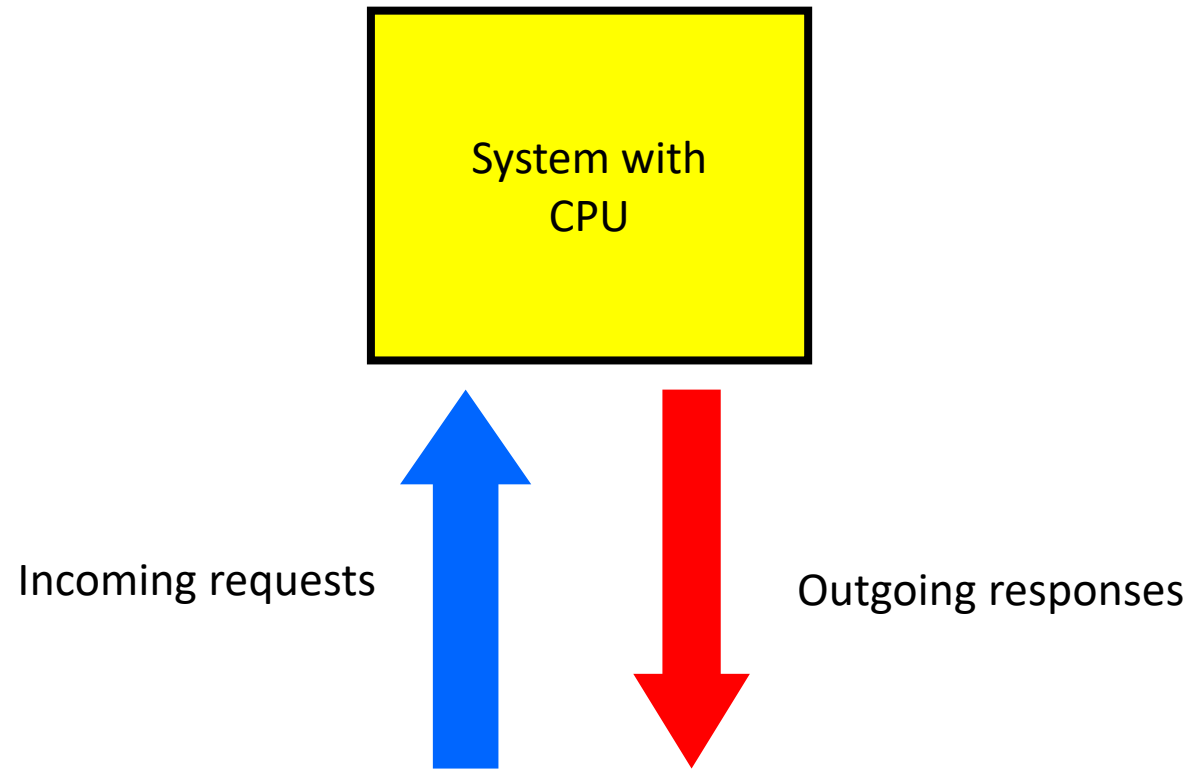


Fig. 2. Multi-tier application architecture.

General structure of a computing system



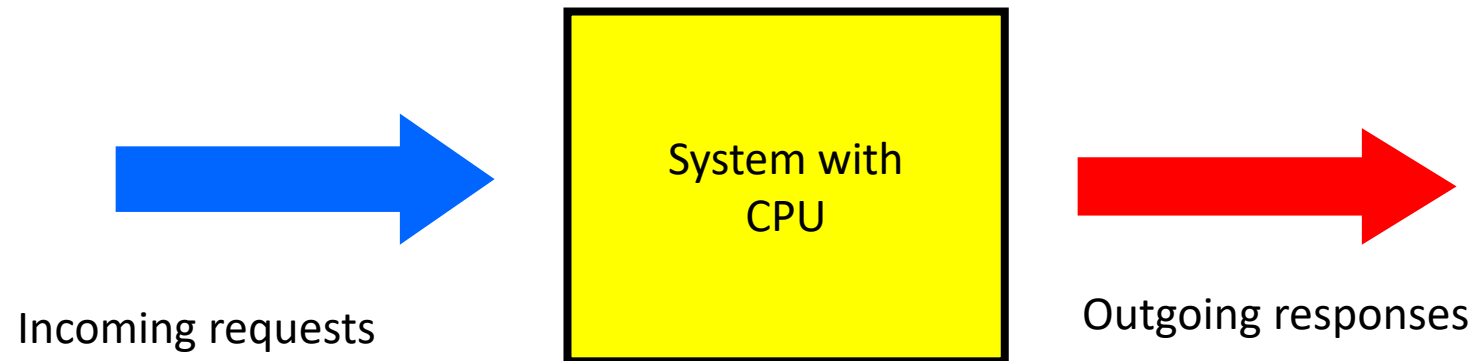
Computing system performance

When designing a computing system, there are several **performance metrics** that can be used:

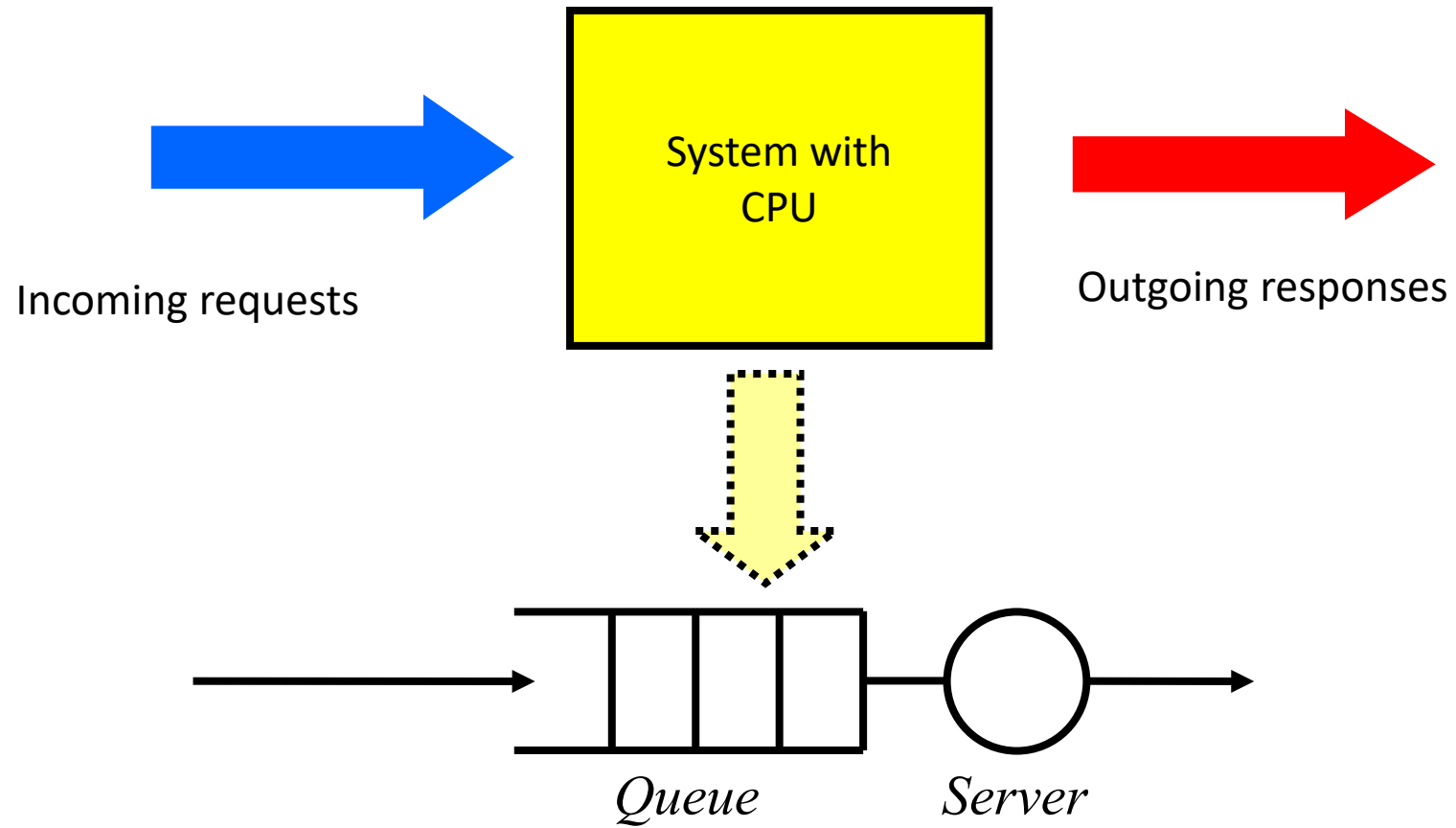
- Processing times
- Waiting times
- Loss probability
- Availability

To design a computing system

How should a computing system be dimensioned so all performance requirements can be fulfilled?



Queuing systems



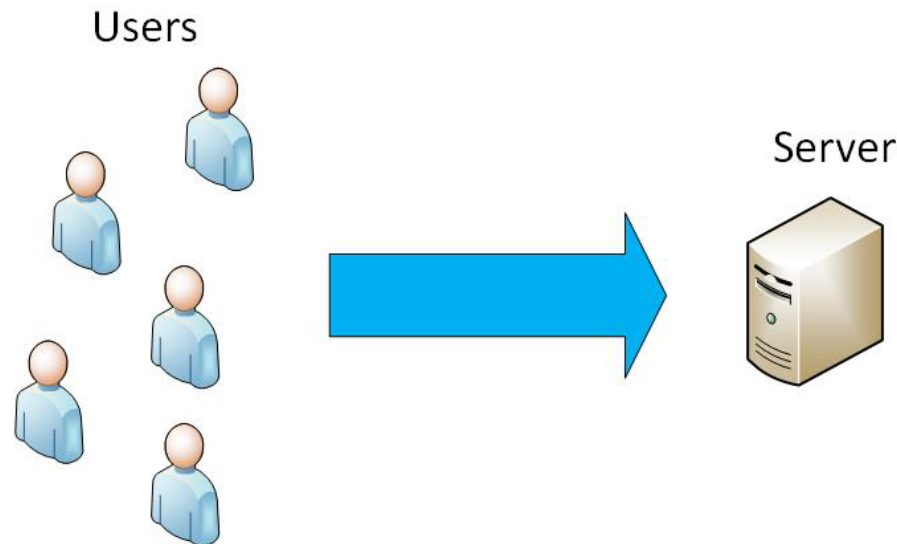
Queuing theoretic metrics?

- Stability requirements
- CPU load
- Response times
- Throughput
- Loss probability
- Average number of jobs in the system
- Etc...

Server systems

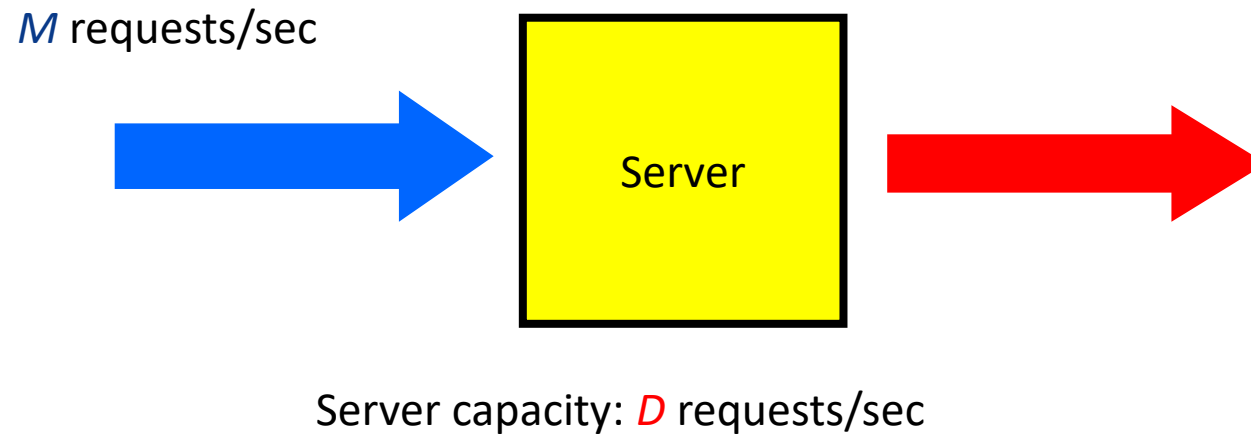
In reality we have several users that request web pages from the same server system. Two cases:

1. "Many users" means that their combined request patterns can be modelled as one arrival process.
2. "Limited amount of users" means that each user's request pattern needs to be modelled separately.



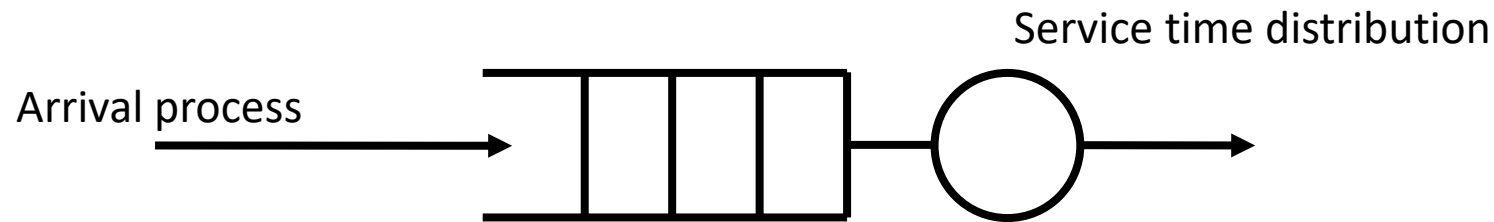
Stability requirement

- If $M \geq D$, the server will be overloaded.
- Stability requirement: $M < D$



Design a queuing model

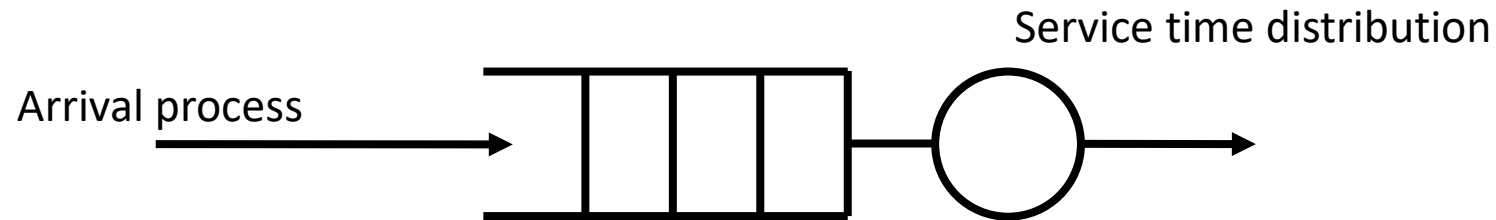
- Request arrivals are modelled as a stochastic process.
- Each job is processed by the server. The service time is modelled as a random distribution.



A queuing model is a mathematical description of a computing system, where the **arrival process** and **service time distribution** are described together with other important metrics, for example maximum **queue size** and the **number of servers**.

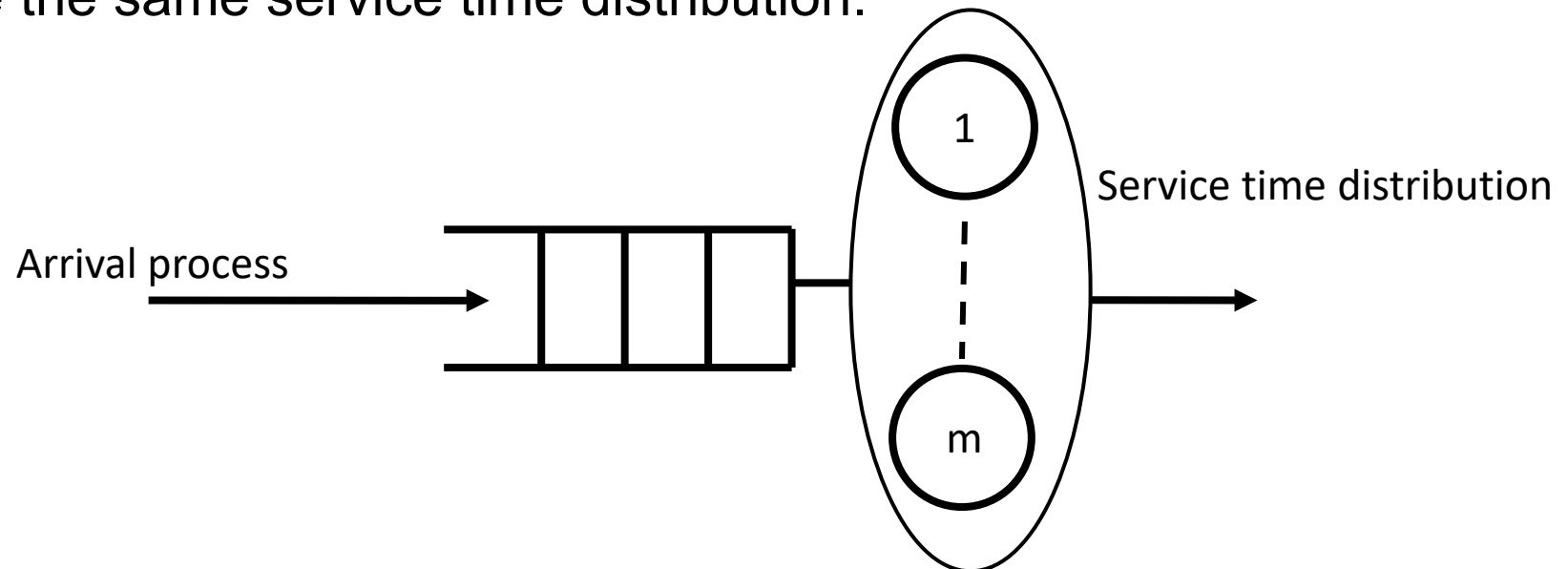
Maximum queue size

- The maximum queue size describes the maximum number of requests that can be in the system at the same time.
- In this course, we work with two main cases:
 - Unlimited queue
 - No queue (a so called Erlang system)



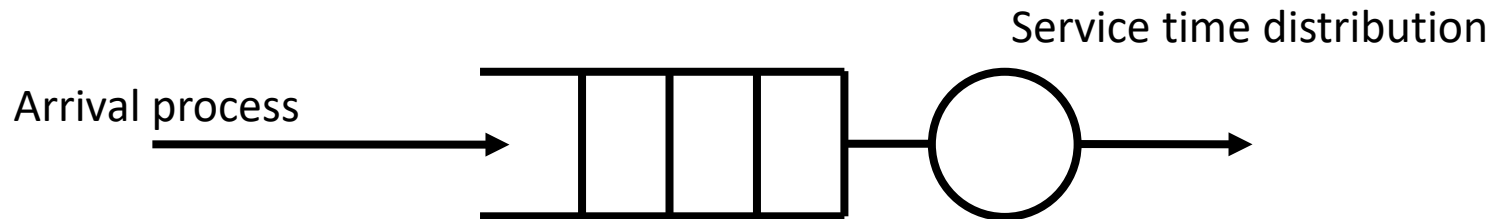
Number of servers

- The queuing systems in this course can have m servers.
- The servers (CPU:s) in a queueing system used in this course are always considered to work independent of each other.
- A server can either be idle or busy. It is never "in between".
- When a server becomes idle, it starts to process the next request in the queue.
- All servers have the same service time distribution.



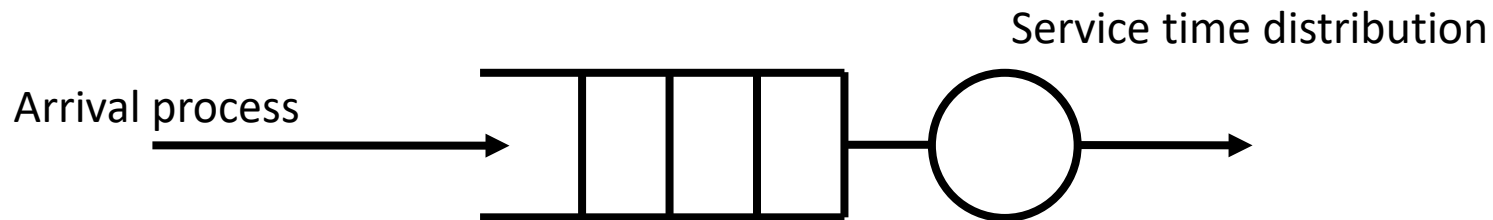
Service time distribution

- When a request comes to the CPU (server), it must be processed before it is completed.
- When a request is processed in the server, there is a service time (also called processing time).
- In queueing theory, the service times for different jobs are usually considered to be independent of each other (otherwise the math becomes too difficult).
- The service time for a request can be described with a service time distribution.



Service time distributions

- Deterministic distribution
 - All requests have the same service time
 - If the server capacity is D requests/sec, the service time for a request, $x = 1/D$ seconds.
- Exponential distribution
 - The service time for a request, x , is exponentially distributed with mean $1/D$ seconds.



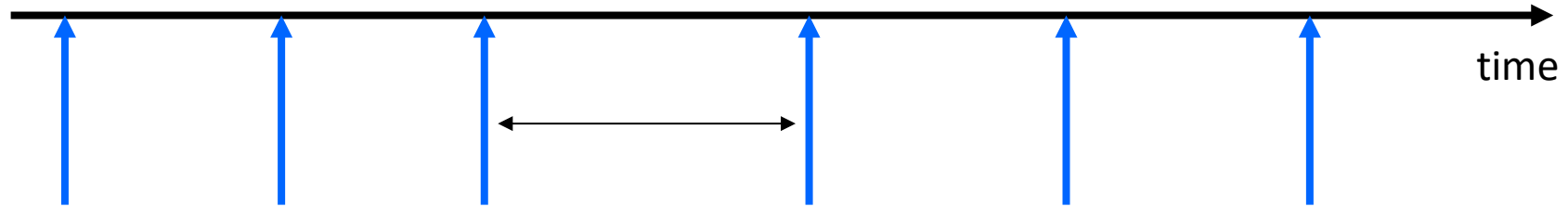
Exponential distribution

The exponential distribution is commonly used to model time between events in telecommunication system.

- The exponential distribution is **memoryless**. This means that it does not matter how long you have waited for an event so far. If you have not observed an event until time a , the distribution of waiting time (from time a) until the next event is the same as when you started at time zero.
- **If the number of events in a time interval is Poisson distributed, the time between events is exponentially distributed.**
- A continuous random variable X is said to have an exponential distribution with parameter $\lambda > 0$, shown as $X \sim \text{Exponential}(\lambda)$, if its probability density function (PDF) is given by:

$$f_X(x) = \lambda \cdot e^{-\lambda x}, \text{ when } x > 0$$

Stochastic arrival process



New requests arrive "randomly". The time between two arrivals can be modelled as a random distribution $A(t)$.

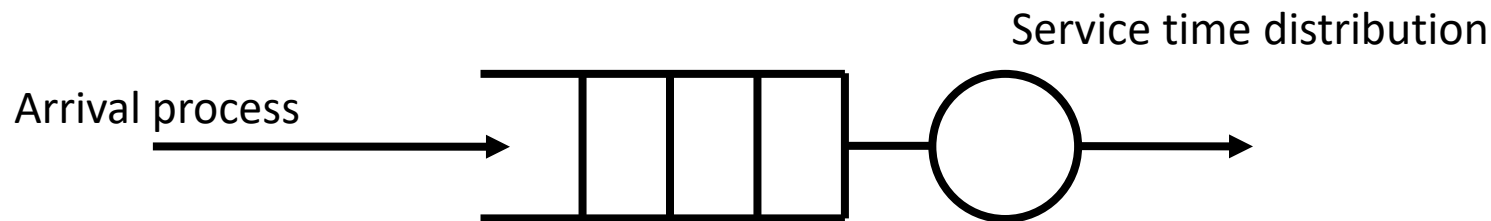
Two arrival processes

- **Deterministic process**

- The time between two arrivals is constant.
- M requests/sec means that the time between two arrivals always is $1/M$ seconds.

- **Poisson process**

- The time between two arrivals, T_a , is exponentially distributed with mean $1/M$ seconds.



Poisson distribution

The Poisson distribution is an appropriate model if the following assumptions are true:

- k is the number of times an event occurs in an interval and k can take values 0, 1, 2, ...
- The occurrence of one event does not affect the probability that a second event will occur. That is, **events occur independently**.
- The **average rate at which events occur is independent of any occurrences**. For simplicity, this is usually assumed to be constant but may in practice vary with time.
- **Two events cannot occur at exactly the same instant**; instead, at each very small sub-interval exactly one event either occurs or does not occur.
- If these conditions are true, then **k is a Poisson random variable, and the distribution of k is a Poisson distribution**.

Poisson distribution

The Poisson distribution can often be applied to systems with a **large number of possible events, each of which is rare**. The number of such events that occur during a fixed time interval is, under the right circumstances, a random number with a Poisson distribution.

Let a discrete random variable X represent the number of events during a time interval t . If X follows a Poisson distribution with mean $E[X] = \lambda t$ events, then the probability mass function for X is:

$$P(X = k \text{ during } t) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

Exercise: Slotted ALOHA

Assume that we have a wireless system where the MAC protocol uses slotted ALOHA. In Slotted ALOHA, users can only submit a data packet at the start of a time slot. Each user submits maximum one packet per time slot.

The number of users that want to submit a data packet follows a Poisson distribution with mean $\lambda=1$ per time slot. For modelling reasons we can assume that these data packets (maximum one per user) have arrived during the previous time slot.

What is the probability that more than one user submits a data packet in a time slot?

Solution: Slotted ALOHA

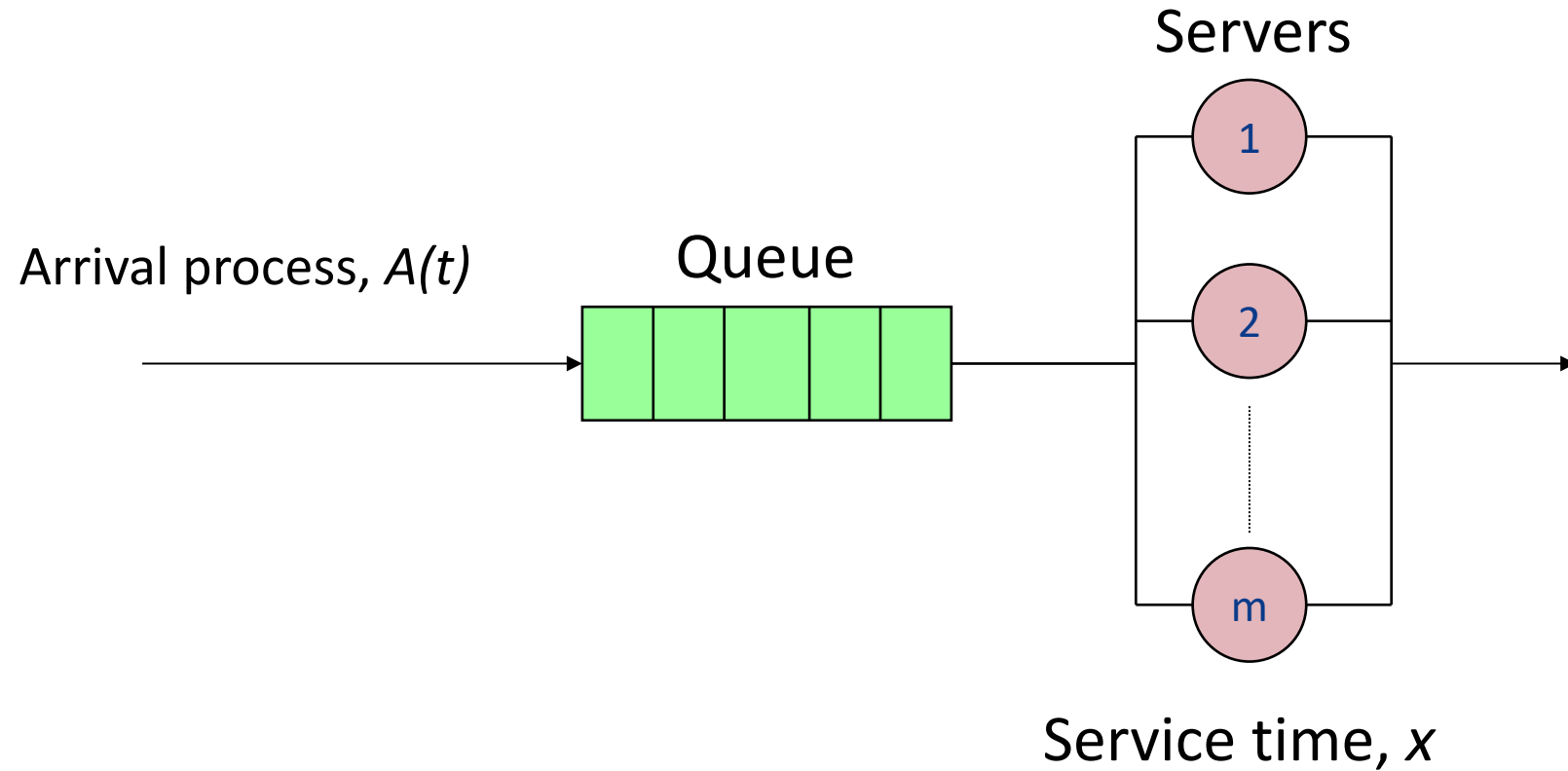
Let X be a random variable for the number of users that submit a data packet in the time slot (here, we let t = a time slot).

$$P(X = k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!} = \frac{e^{-1}}{k!}$$

$$P(X > 1) = 1 - P(X \leq 1) = 1 - P(X = 0) - P(X = 1) = 1 - e^{-1} - e^{-1} \approx 0.26$$

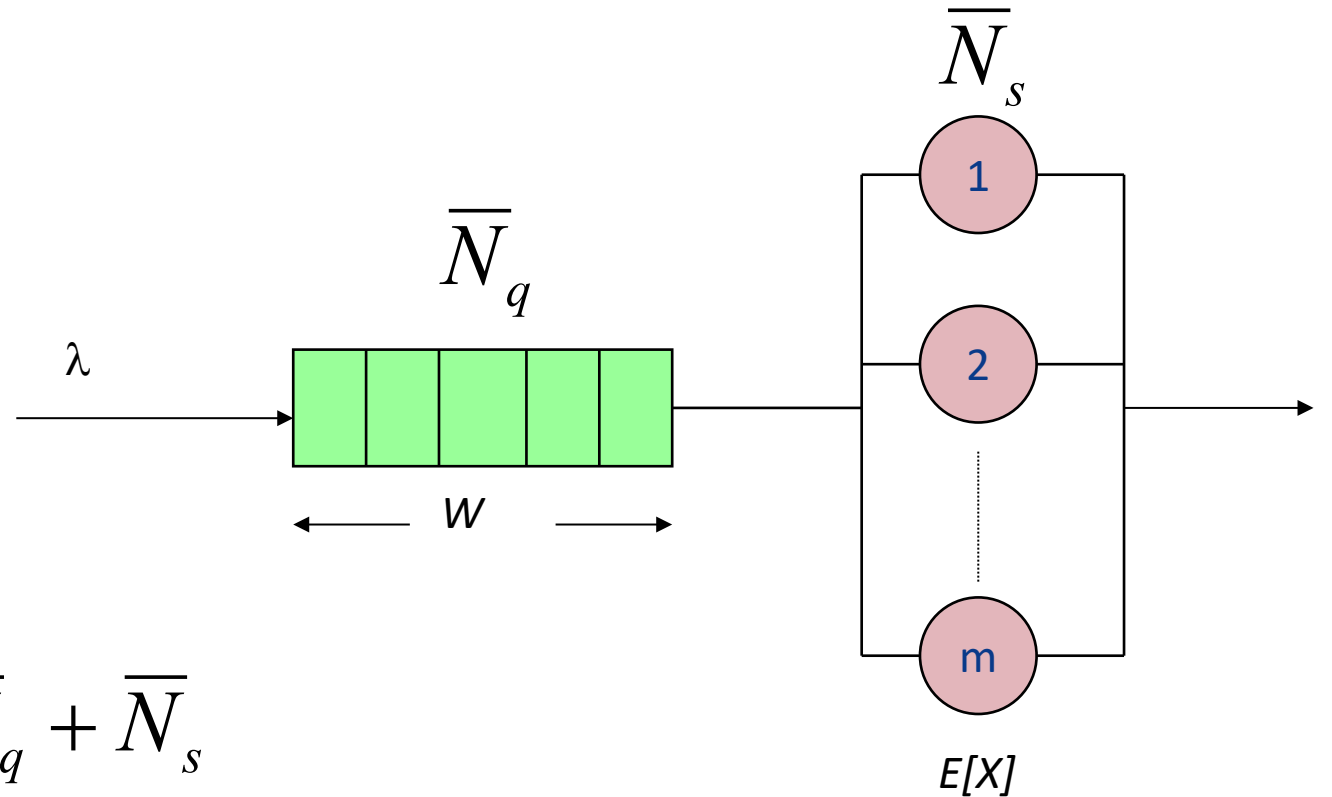
The probability that there is a data collision in a time slot is 0.26.

A general queuing system with unlimited queue



Main system variables

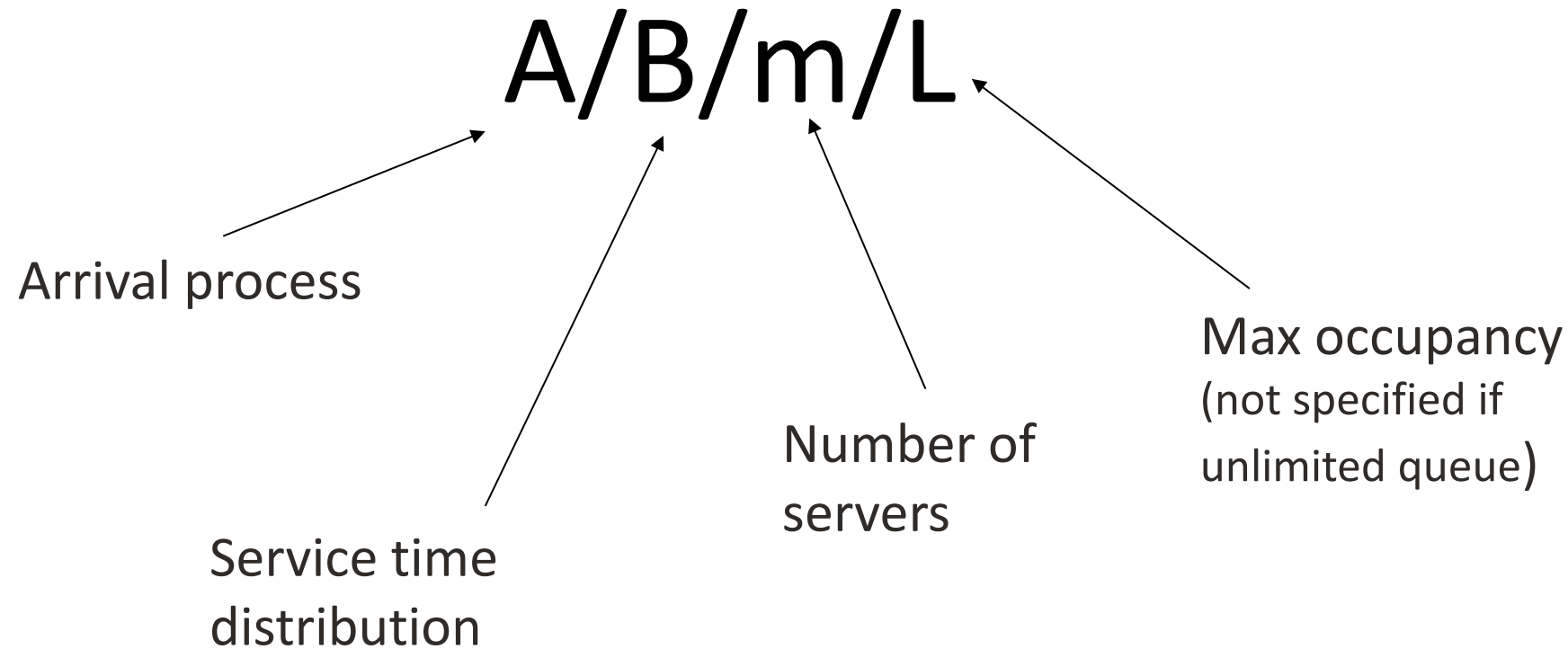
- Average number of requests
 - In the whole system, $\bar{N}$
 - In the queue, $\bar{N}_q$
 - In the servers, $\bar{N}_s$
- Average time for a request
 - In the system, T
 - In the queue, W
 - In the server (service time), $E[X]$



$$\bar{N} = \bar{N}_q + \bar{N}_s$$

$$T = W + E[X]$$

General classification for queuing systems



Some arrival processes

- M = Markovian
 - The interarrival time is exponentially distributed (that means that the arrival process is a Poisson process)
- D = Deterministic
 - The interarrival time is deterministic.
- G = General
 - The interarrival times follow another (specified) distribution.

Some service time distributions

- M = Exponential distribution
- D = Deterministic distribution
- G = General distribution (according to a specified distribution)

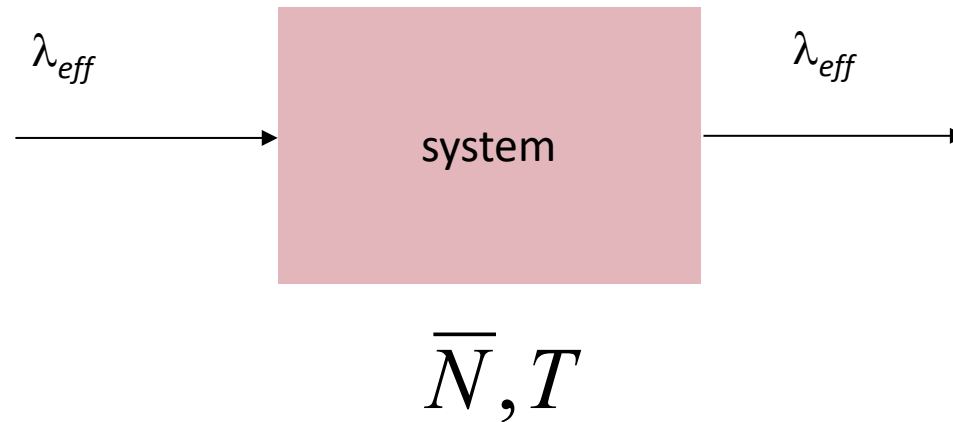
Queueing models in this course

In this course, we will use the following models:

- **$M/M/1$ with unlimited queue**
 - 1 server
 - Arrival process is Poisson
 - Service times are exponentially distributed
- **$M/G/1$ with unlimited queue**
 - 1 server
 - Arrival process is Poisson
 - Service times have a specified distribution (general)
- **$M/M/m/m$ (Erlang systems)**
 - No queue
 - m servers
 - Arrival process is Poisson
 - Service times are exponentially distributed

Little's theorem

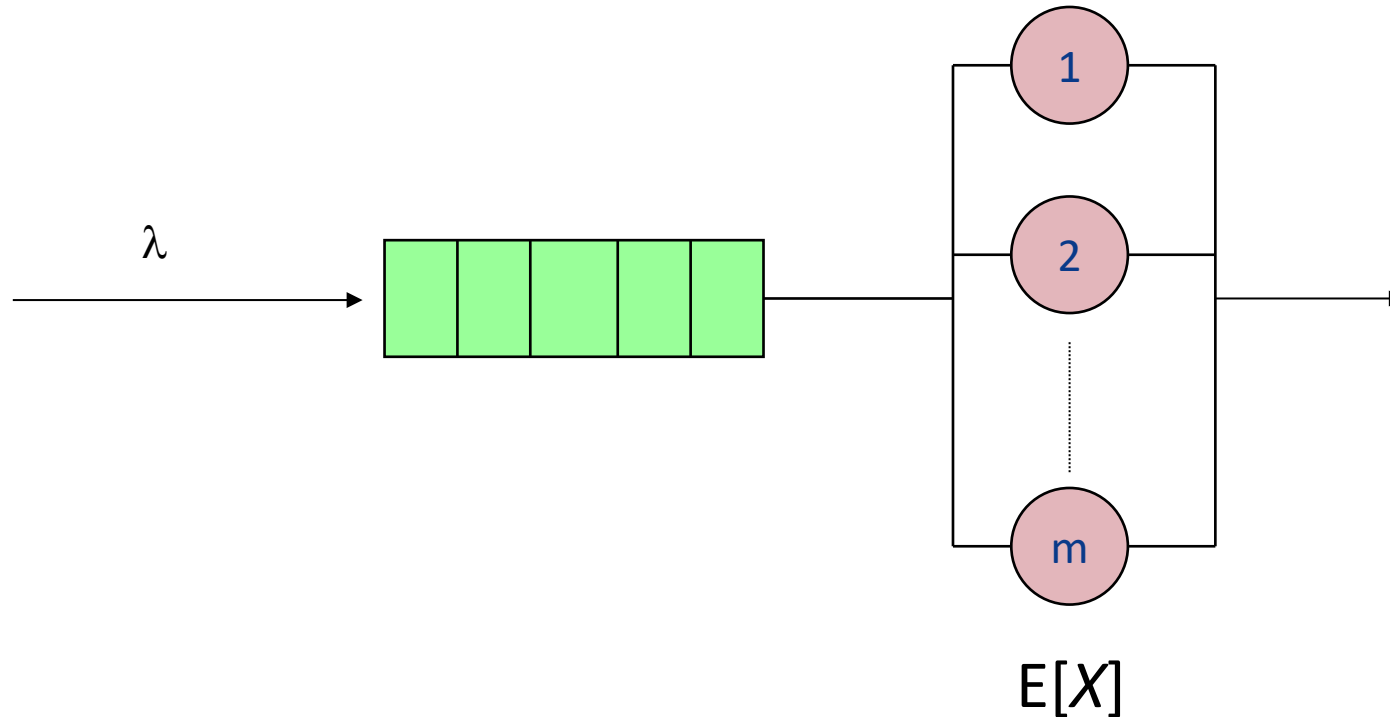
- Little's theorem is valid for **all** stable conservative systems.
- Conservative system = no jobs are destroyed or created in the system



$$\bar{N} = \lambda_{eff} \cdot T$$

Stability requirement

A stable queuing system has a limited average response time, which means that a system with unlimited queue is stable if $\lambda < m/E[X]$ ($= m\mu$)



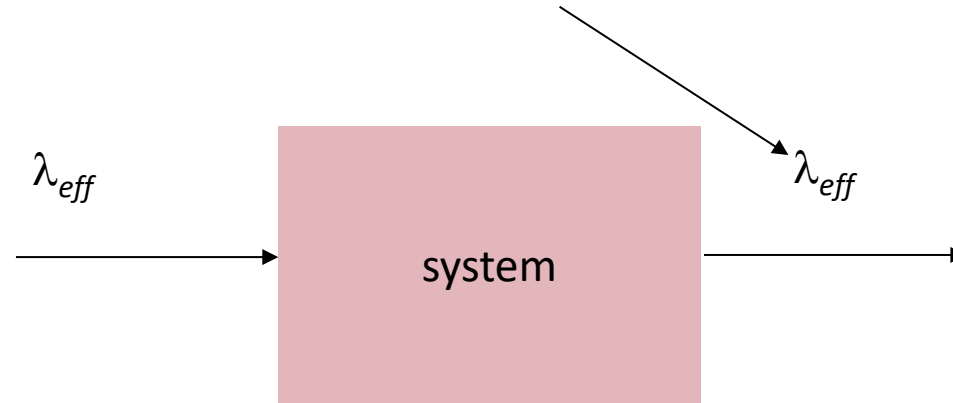
Utilisation

The **utilisation** of a computing system (or system load) is defined as the average fraction (ratio) of **occupied servers**.

$$U = \frac{\bar{N}_s}{m} = \frac{\lambda_{eff} \cdot E[X]}{m}$$

Throughput

The **throughput** of a computing system is defined as the **average departure rate**, that is the **average rate of processed requests**.

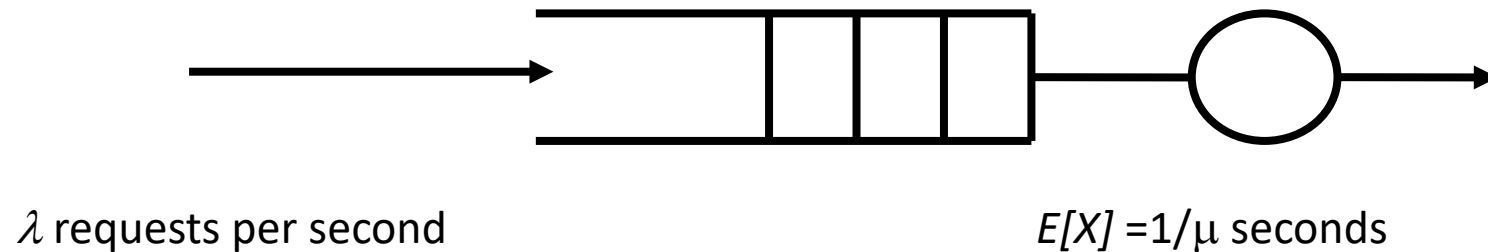


"Everything that goes in, also comes out"

Some queueing theoretic formulas
(the math behind them is part of
the course Queueing Systems)

M/M/1 system

- Utilisation = Probability that the server is busy = $\lambda/\mu = \rho$
- Throughput, $\lambda_{\text{eff}} = \lambda$ (no requests are blocked)
- Average number of requests in the system, $\bar{N} = \frac{\rho}{1-\rho}$
- Average response time, $T = \frac{1}{\mu(1-\rho)}$
(use Little's theorem)



M/G/1 systems: Pollaczek-Kinchine formula

The Pollaczek-Kinchine formula is valid for M/G/1 systems

$$\bar{N} = \rho + \frac{\rho^2 + \lambda^2 \cdot Var(X)}{2 \cdot (1 - \rho)}$$

$$\rho = \lambda \cdot E[x]$$

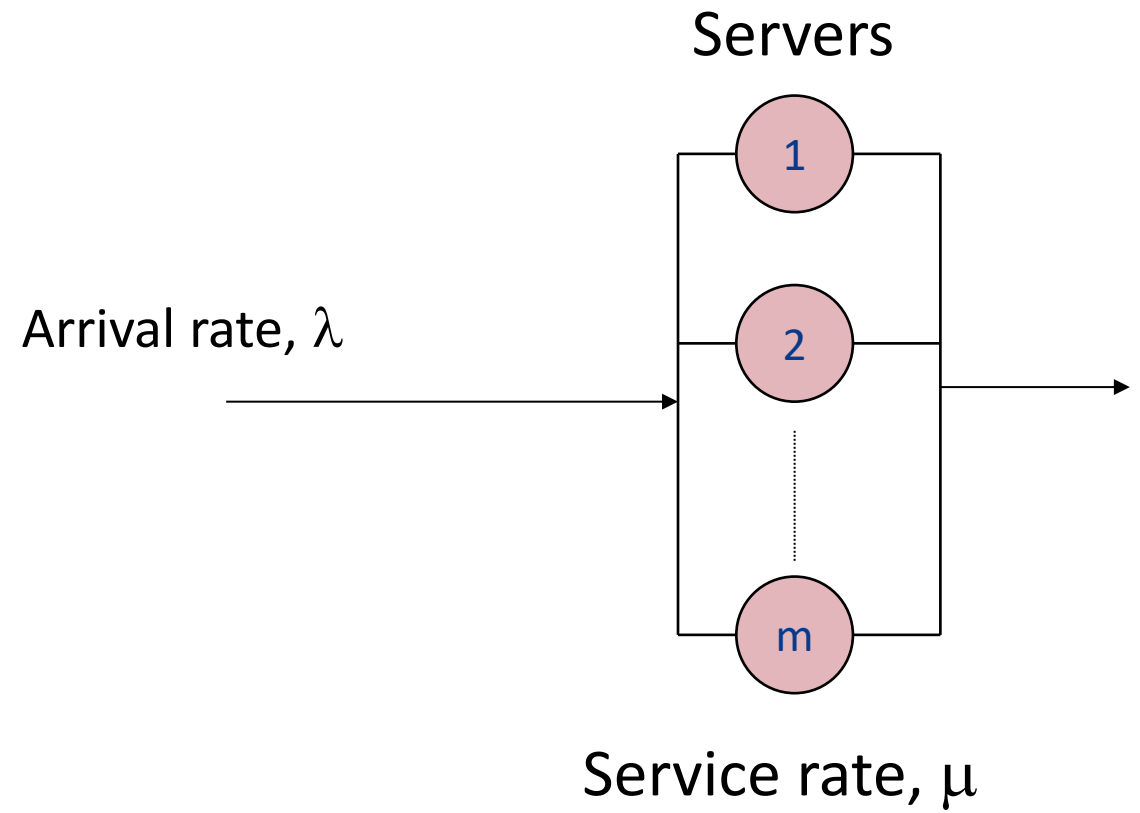
Erlang systems: M/M/m/m (without queue)

Models of server systems without queue are sometimes useful.

The probability that a request is blocked = **Erlang loss formula:**

$$E_m(\rho) = \frac{\rho^m / m!}{\sum_{i=0}^m \frac{\rho^i}{i!}}$$

Where $\rho = \lambda/\mu$



Exercise (ETSN11 231025):

Web server system

You are doing performance modeling of a web server system. You measure the arrival rate of new requests and find that the arrivals can be modelled as a Poisson process with an average rate of 10 requests per second. Also, you measure the service times of request, and find that the service times are independent and that they can be modelled with an exponential distribution with an average of 0.06 seconds. Also, you see that the web server system can be modelled as a conservative single server system with unlimited queue.

- a) What is the **average utilization** of the server?
- b) What is the **average throughput** of the server system?
- c) What is the **average total number of requests** in the server system?
- d) Assume instead that the service times can be modelled as a **deterministic distribution**. What is the **average total number of requests** in the system in this case?

Solution a) – c)

Requests arrive according to a Poisson process and the service times are following a (negative) exponential distribution. There is one server and an unlimited queue.

Therefore, the system can be modelled as an **M/M/1 system** with $\lambda = 10$ requests per second and $\mu = 1/0.06 = 16.67$ requests per second. You find required formulas in the formula sheet.

- a) The average utilization of the server, ρ , can be calculated as $\rho = \lambda/\mu = 0.6$.
- b) The average throughput of the system, λ_{eff} , is the same as the average arrival rate, since we have an unlimited queue. Therefore, $\lambda_{\text{eff}} = 10$ requests per second.
- c) The average total number of requests in the system, $\bar{N} = \frac{\rho}{1-\rho} = 1.5$

Solution d)

d) In this case, Requests arrive according to a Poisson process, but the service times are deterministic. This means that we have an **M/D/1** system with unlimited queue. Therefore, we should use the **Pollaczek-Kintchine formula** to find the average total number of requests in the system.

Pollaczek-Kintchine formula:

$$\bar{N} = \rho + \frac{\rho^2 + \lambda^2 \cdot \text{Var}(X)}{2 \cdot (1 - \rho)}$$

A deterministic distribution has zero variance, i.e $\text{Var}(X) = 0$.

This means that $\bar{N} = \rho + \frac{\rho^2}{2 \cdot (1 - \rho)} = 1.05$

WebFilmz

Webfilmz is a video streaming company. Users pay a monthly subscription fee to gain access to the company's catalogue of TV shows and movies, which they can then stream via the internet. The catalogue currently has about 5000 videos in total. WebFilmz is a relatively new company still trying to build market share, so it is very important that new users are provided with a consistently high Quality of Experience (QoE) so that they will continue to subscribe. WebFilmz currently has 15 million subscribers, located in all EU countries (it is not available outside the EU yet), and their subscriber distribution approximately follows the general population distribution.

WebFilmz has so far been serving all their content from a central data center located in Copenhagen. However, they now wish to investigate a more sophisticated solution. The solution should deliver video streaming content to WebFilmz subscribers with low latency and high Quality of Service (QoS).

User-perceived performance

For simplification, we use a very simple model of a server system.

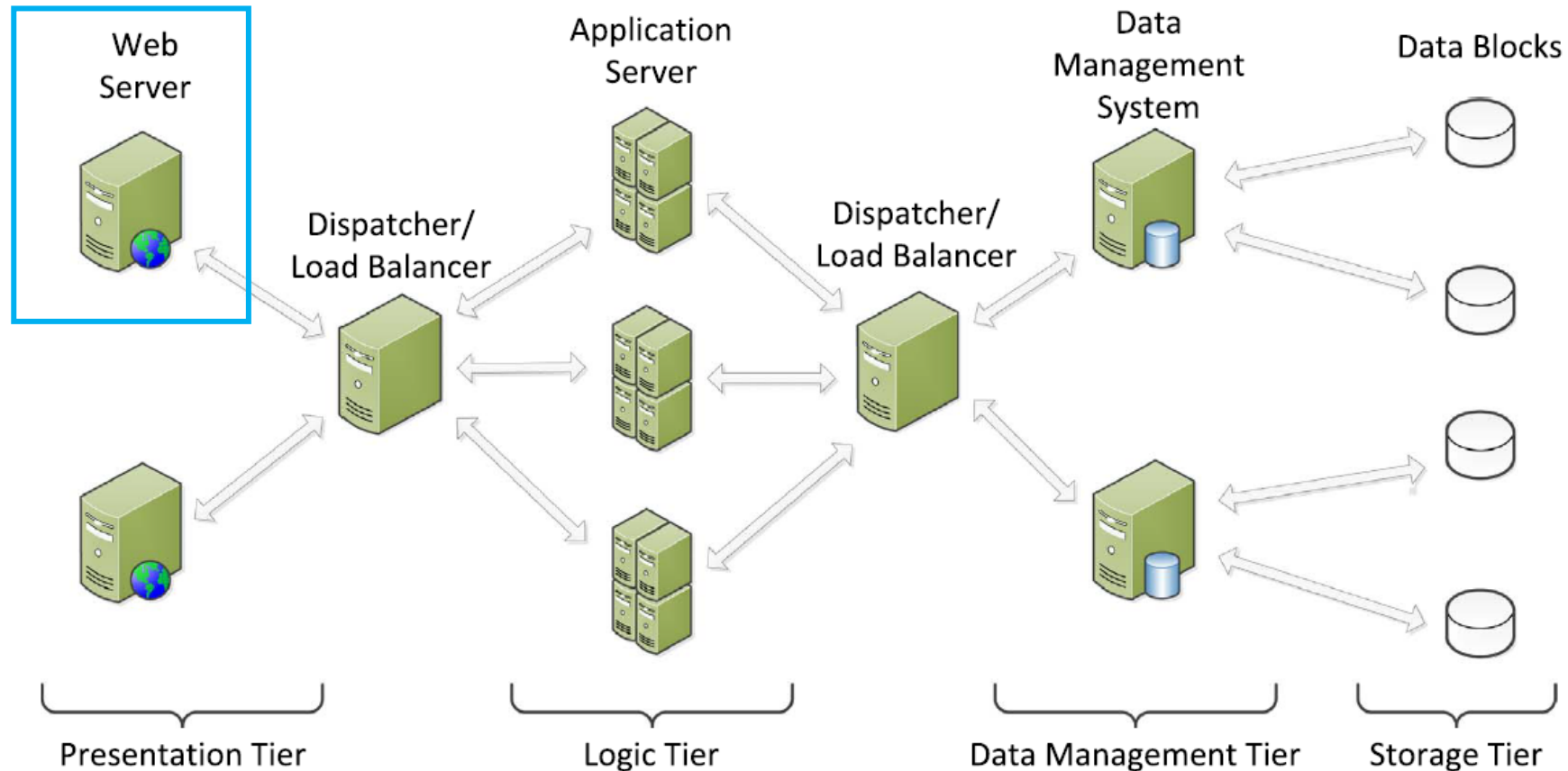
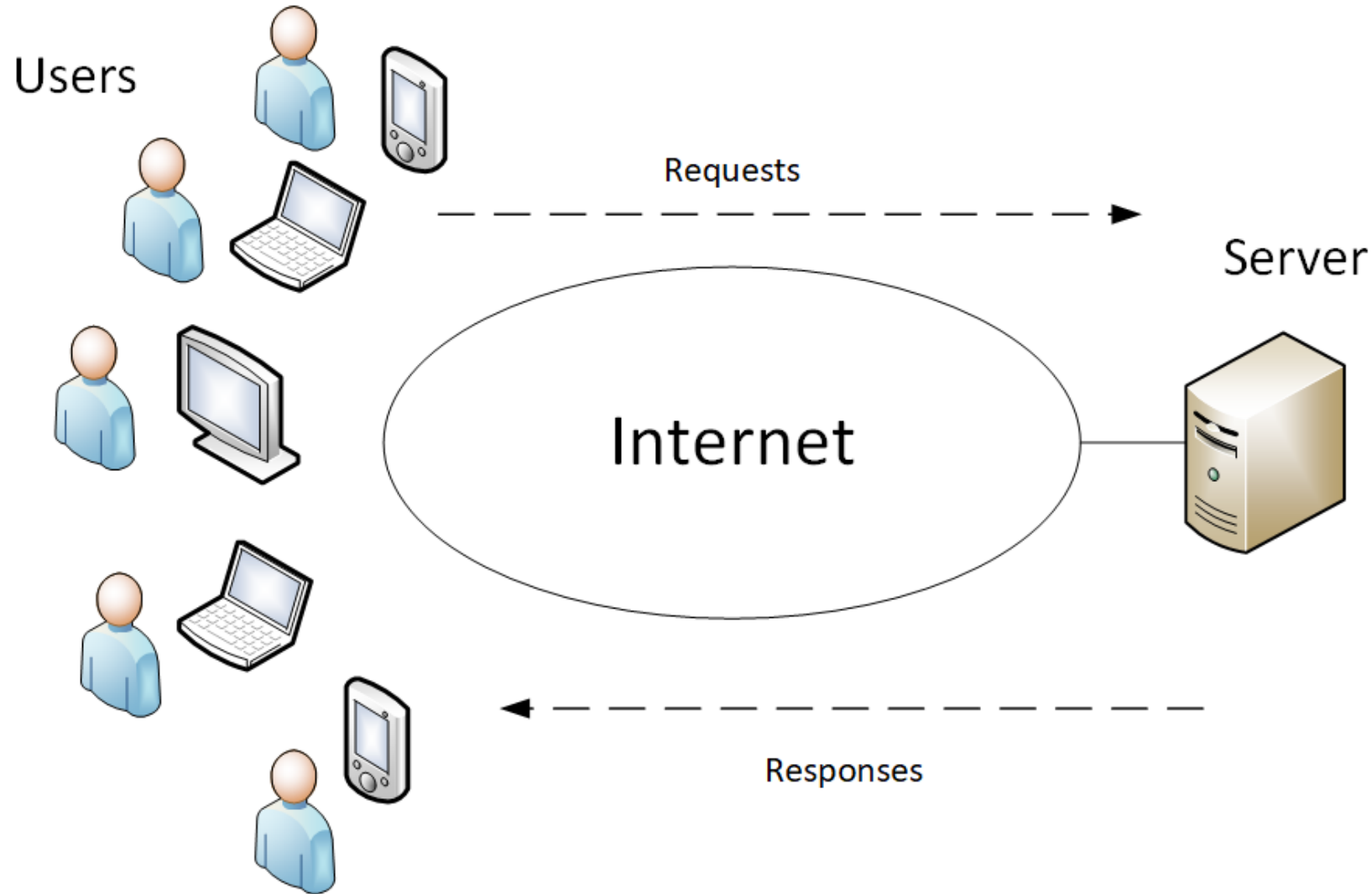


Fig. 2. Multi-tier application architecture.

What do we need to know before using queueing theory on a server system?



Important system model parameters

- **Arrival process:**

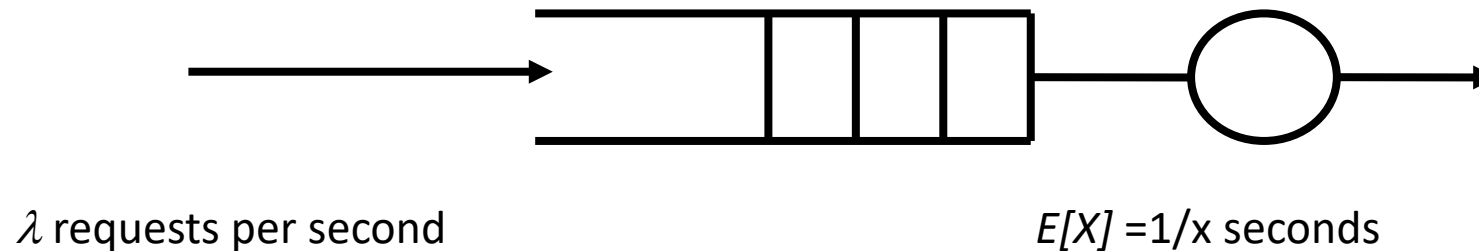
- "Infinite" number of users?
- Independent requests?
- Stationary arrival process?

- **Server:**

- Conservative system?
- Independent processing of requests?
- When multiple servers: random load distribution?

Model of a server system

If the arrival process can be modelled as a Poisson process, and the service times can be modelled according to an exponential distribution:



M/M/1 system

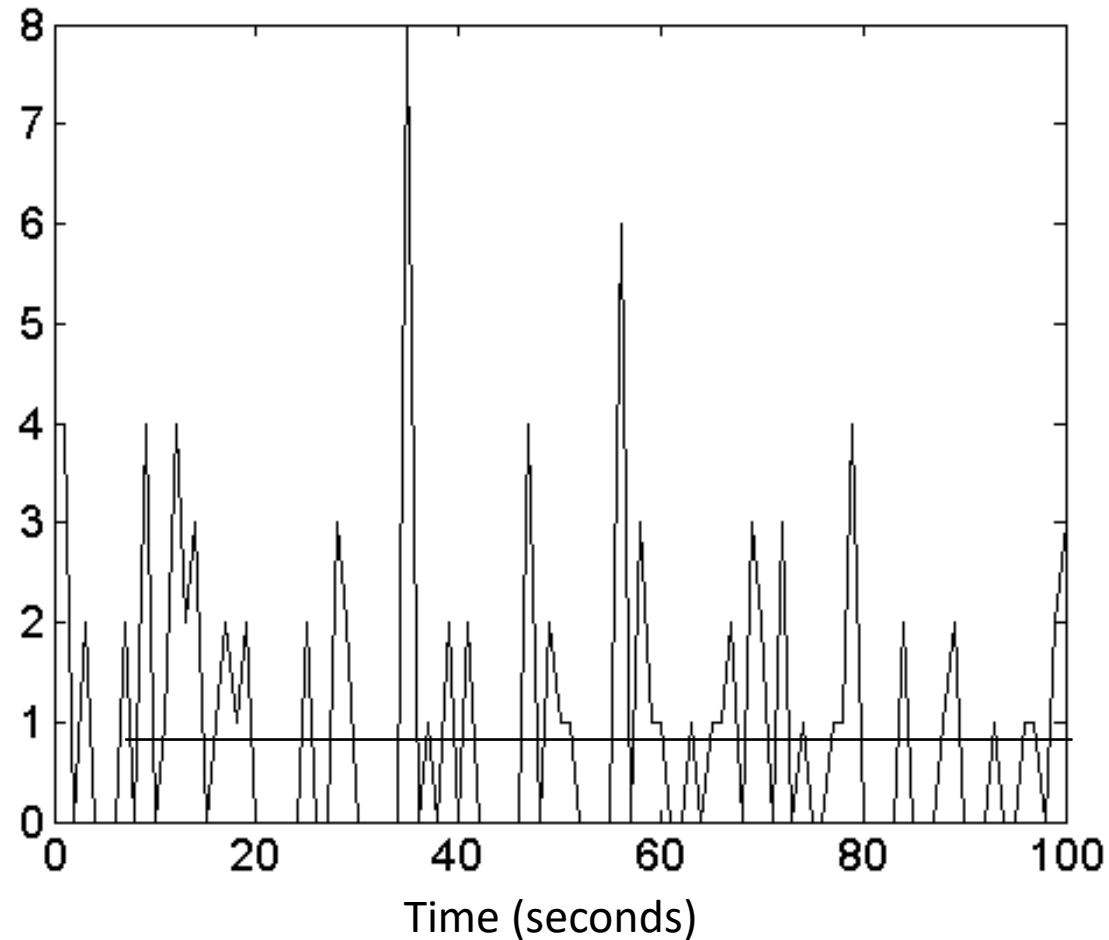
System behavior

- To analyse the system behavior, we need to choose values for the system variables:
 - Average arrival rate, λ
 - Average service time, $E[X] = 1/\mu$

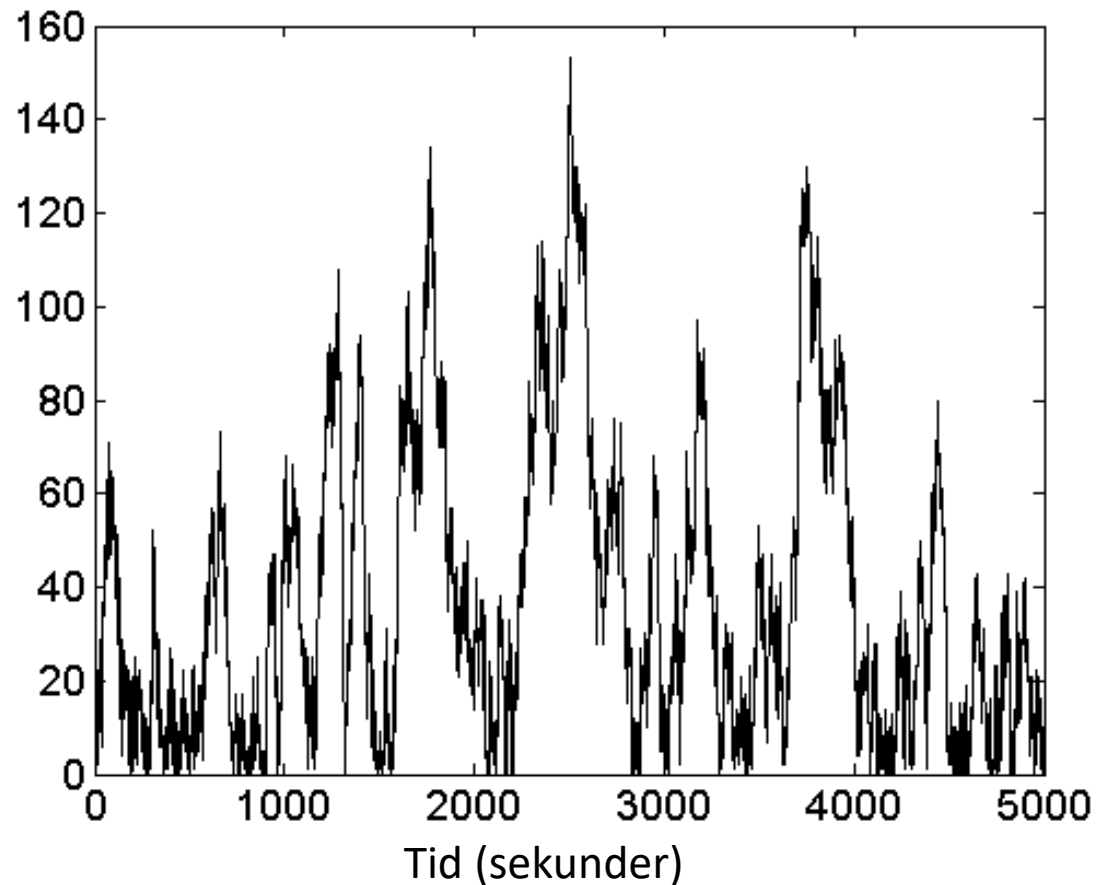
Let us analyse the system when varying λ when $\mu = 10$ requests/second

(rather unrealistic value for our use case, but multiplying this number with for example 10^6 will give the same general dynamics)

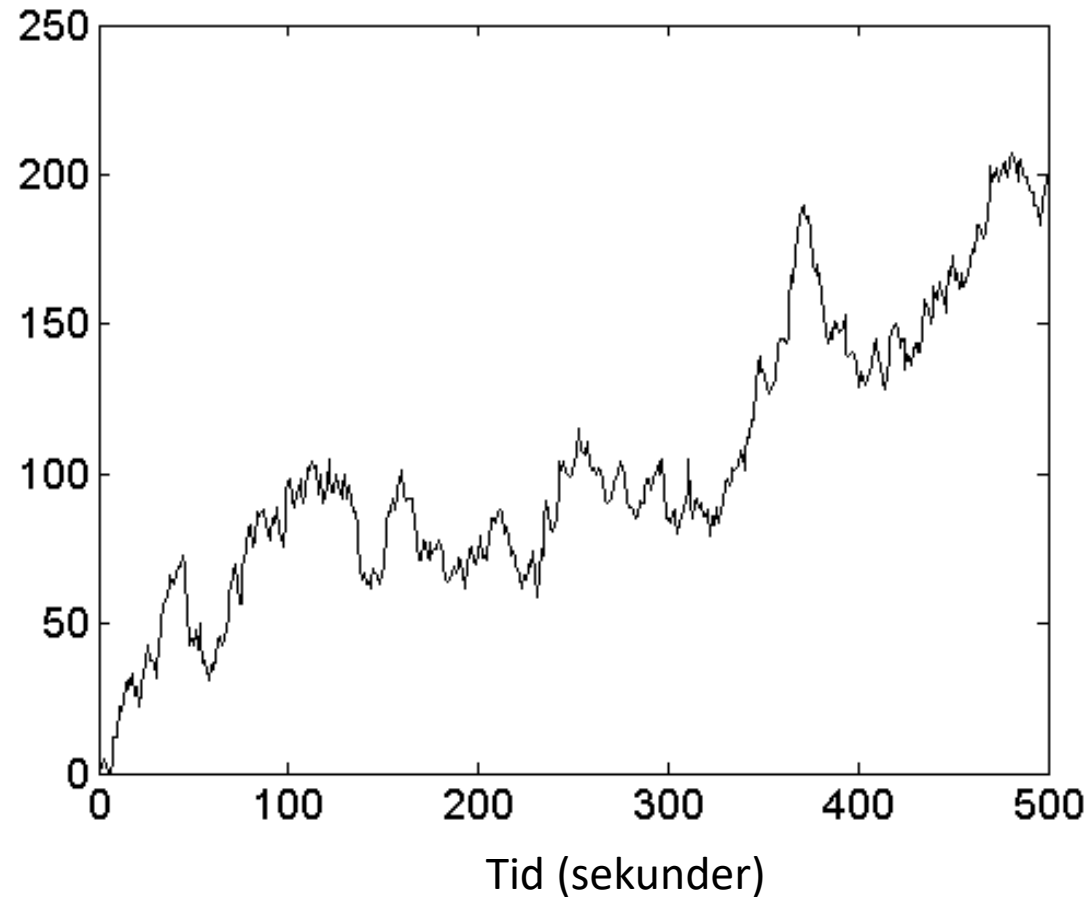
Number of requests in the system when $\lambda=5$
requests per second



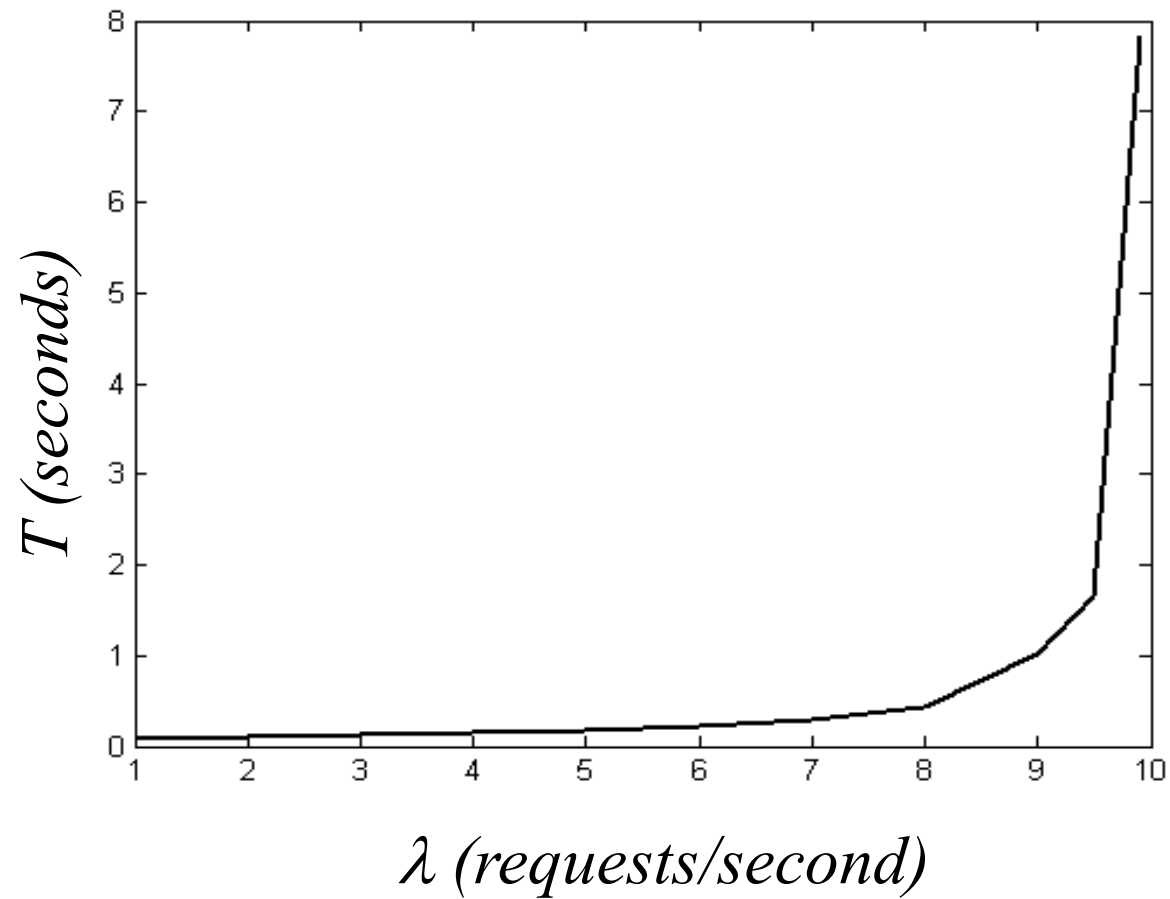
Number of requests in the system when $\lambda=9.8$ req/sec.



Number of requests in the system when $\lambda=10$ req/sec.



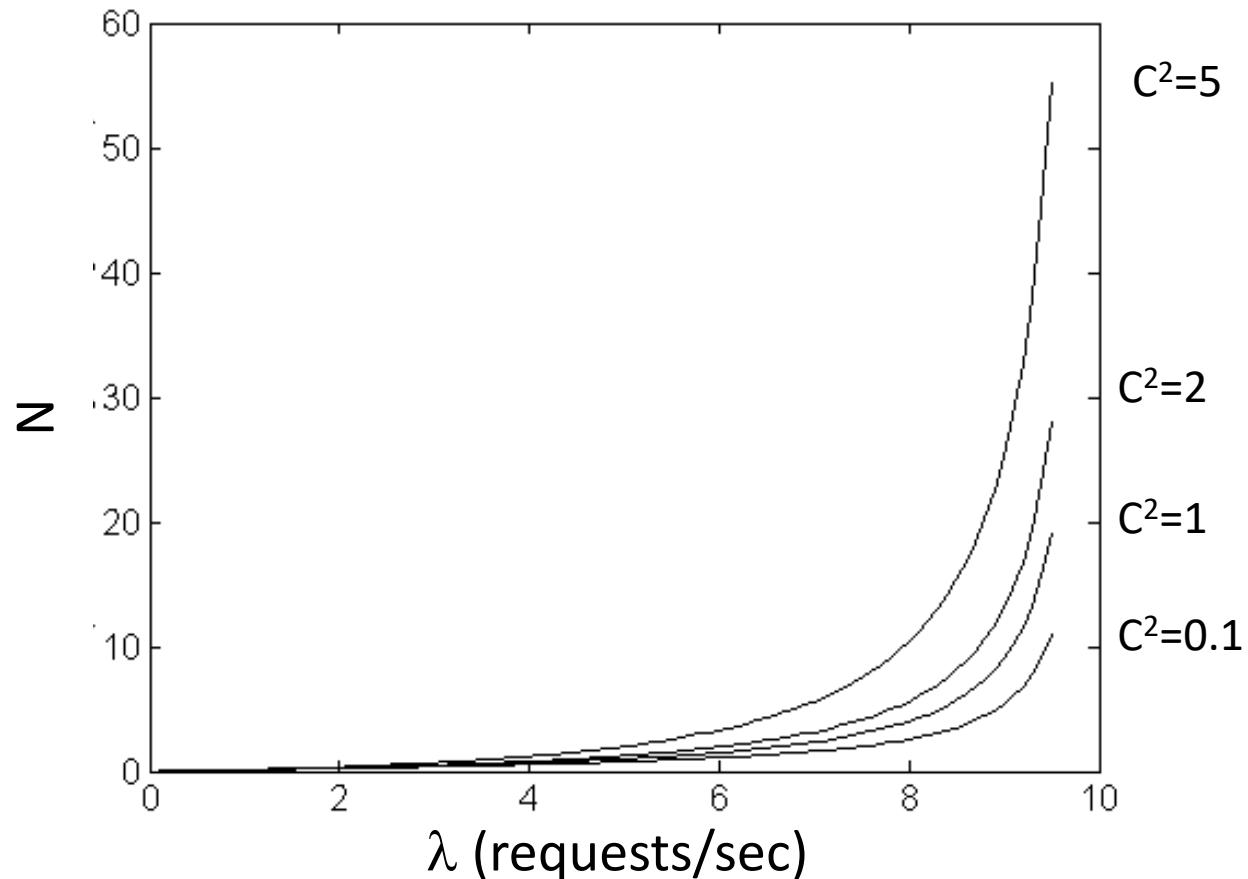
Average response time, T



M/G/1: Average number of requests in the system, $\bar{N}$

Similar behavior can be found for arrival processes!

A more bursty arrival process gives worse behavior of the system.





LUND
UNIVERSITY