POLITECNICO
MILANO 1863

POLITECNICO
MILANO 1863

Software Engineering 2

Software Qualities and Software Architecture

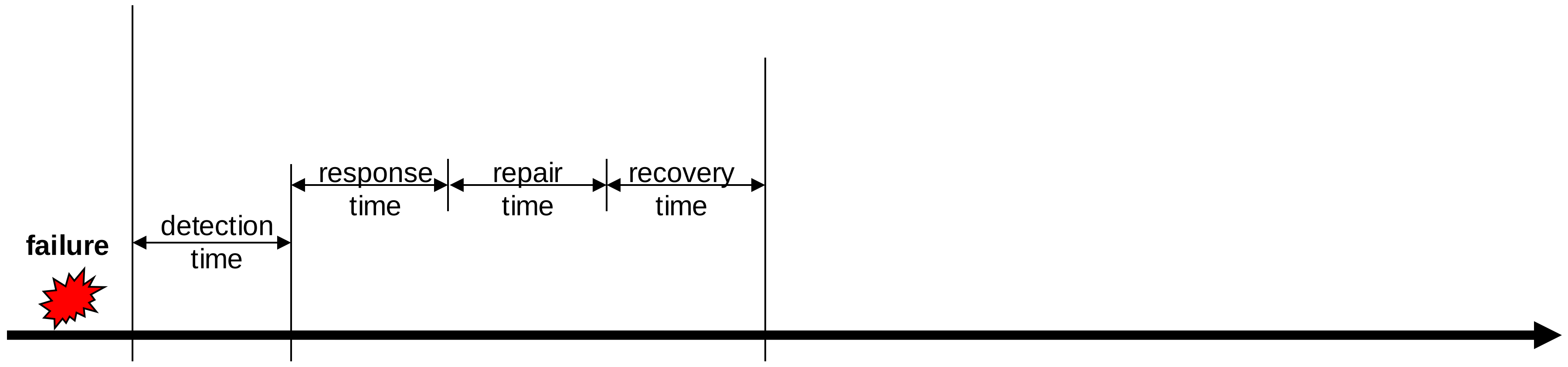
Software qualities and architectures

- Several software qualities are directly influenced by architectural choices
  - Scalability
  - Reliability
  - Availability
  - Usability
  - …
- How do we cope with this?
  - We need **metrics** to quantify qualities and specific **methodologies to analyze** the quantitative impact of architectural choices on these qualities
  - **Tactics** to address the issues

Availability

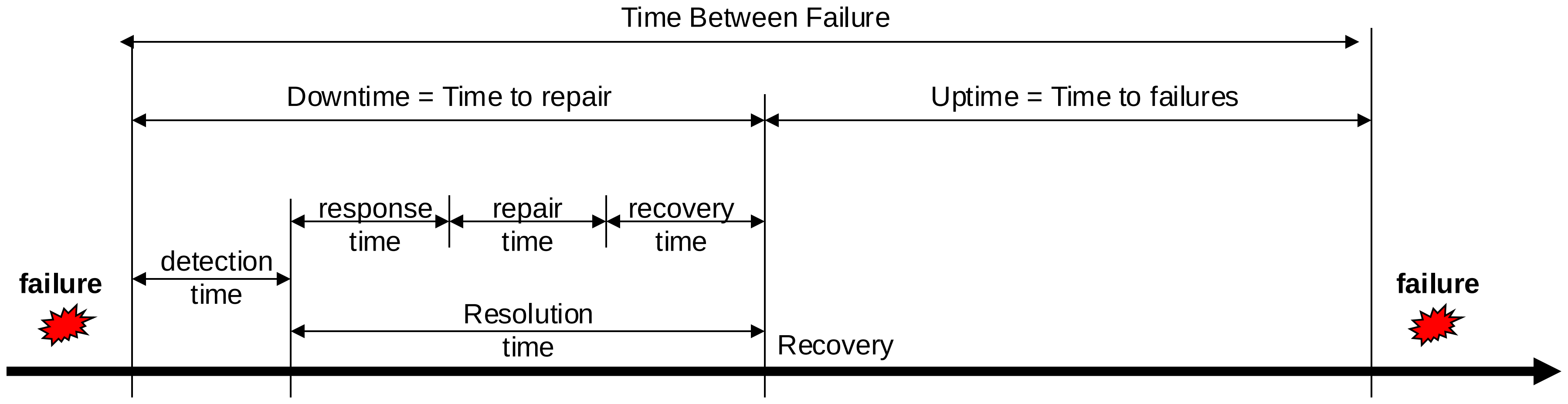
- A service shall be continuously available to the user
  - **Little downtime** and **rapid** service **recovery**
- The availability of a service depends on:
  - Complexity of the IT **infrastructure** architecture
  - **Reliability** of the individual components
  - **Ability to respond** quickly and effectively to faults
  - **Quality of the maintenance** by support organizations and suppliers
  - **Quality** and scope of the **operational management** processes

System life-cycle



- **Time of occurrence:** Time at which the user becomes aware of the failure
- **Detection time:** Time at which operators become aware of the failure
- **Response time:** Time required by operators to diagnose the issue and respond to users
- **Repair time:** Time required to fix the service/components that caused the failure
- **Recovery time:** Time required to restore the system (re-configuration, re-initialization,…)

System life-cycle



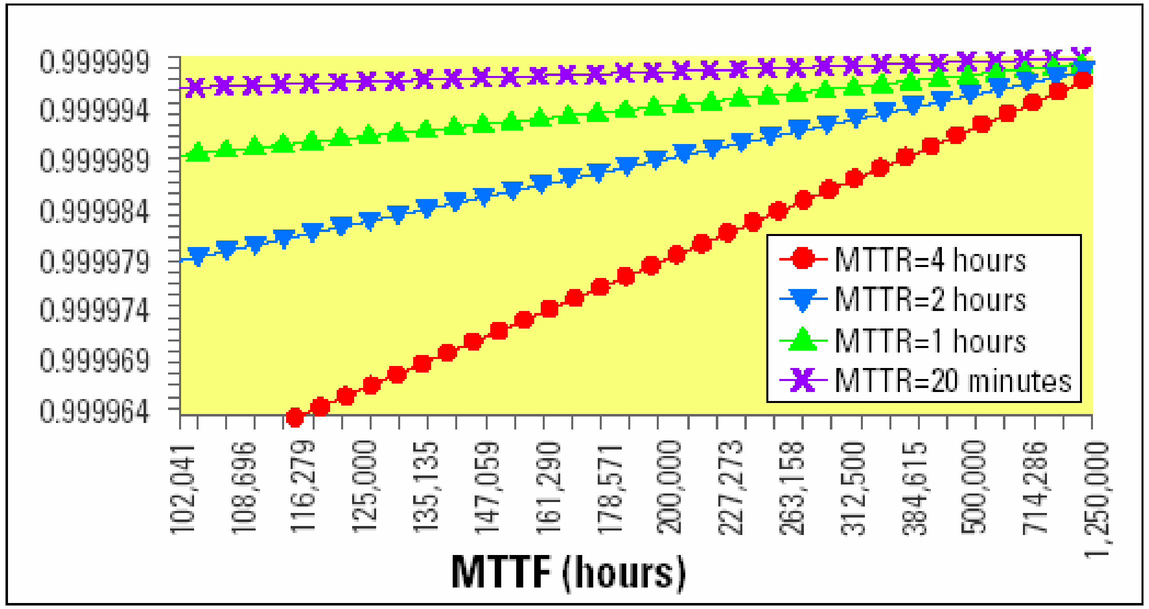
- **Mean Time to Repair (MTTR):** Average time between the occurrence of a failure and service recovery, also known as the downtime
- **Mean Time To Failures (MTTF):** Mean time between the recovery from one failure and the occurrence of the next failure, also known as uptime
- **Mean Time Between Failures (MTBF):** Mean time between the occurrences of two consecutive failures

Availability metric — definition

- Probability that a component is **working properly** at **time *t***

- $A = \frac{MTTF}{MTTF+MTTR}$

- if MTTR small, MTBF ≅ MTTF



Nines notation

- Availability is typically specified in **"nines notation"**
- For example, 3-nines availability corresponds to 99.9%, 5-nines availability corresponds to 99.999% availability

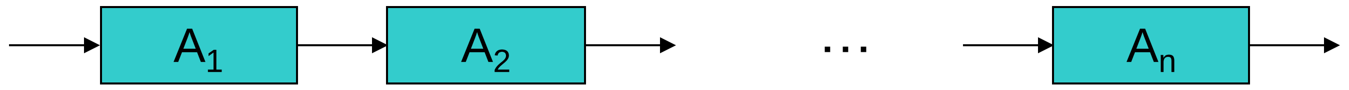
Availability	**Downtime**
90% (1-nine)	36.5 days/year
99% (2-nines)	3.65 days/year
99.9% (3-nines)	8.76 hours/year
99.99% (4-nines)	52 minutes/year
99.999% (5-nines)	5 minutes/year

Availability — analysis methodology

- Availability is calculated by **modeling the system** as an interconnection of **elements in series** and **parallel**
- Elements operating in series
  - Failure of an element in the series leads to a failure of the whole combination
- Elements operating in parallel
  - Failure of an element leads to the other elements taking over the operations of the failed element

Availability in series

- The combined system is operational only if every part is available
- The combined availability is the **product** of the availability of the component parts



$$A = \prod_{i=1}^{n} A_i$$

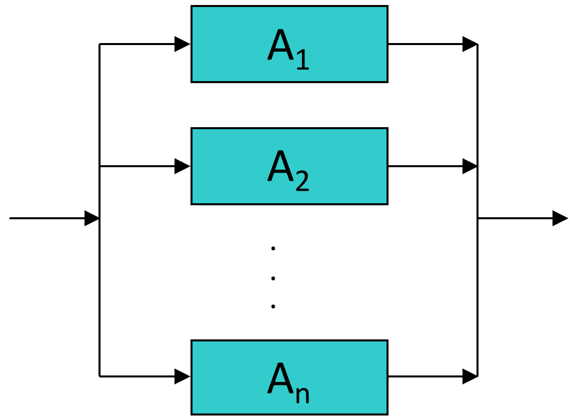
Availability in series – A numerical example

	Availability	Downtime
Component 1	99% (2-nines)	3.65 days/year
Component 2	99.999% (5-nines)	5 minutes/year
Combined	**98.999%**	**3.65 days/year**

- **Downtime = (1-A)*365 days/year**
- The availability of the entire system is negatively affected by the low availability of Component 1
- **A chain is as strong as the weakest link!**

Availability in parallel

- The combined system is operational if at least one part is available
- The combined availability is **1 - P(all parts are not available)**



$$A = 1 - \prod_{i=1}^{n} (1 - A_i)$$

Availability in parallel – A numerical example

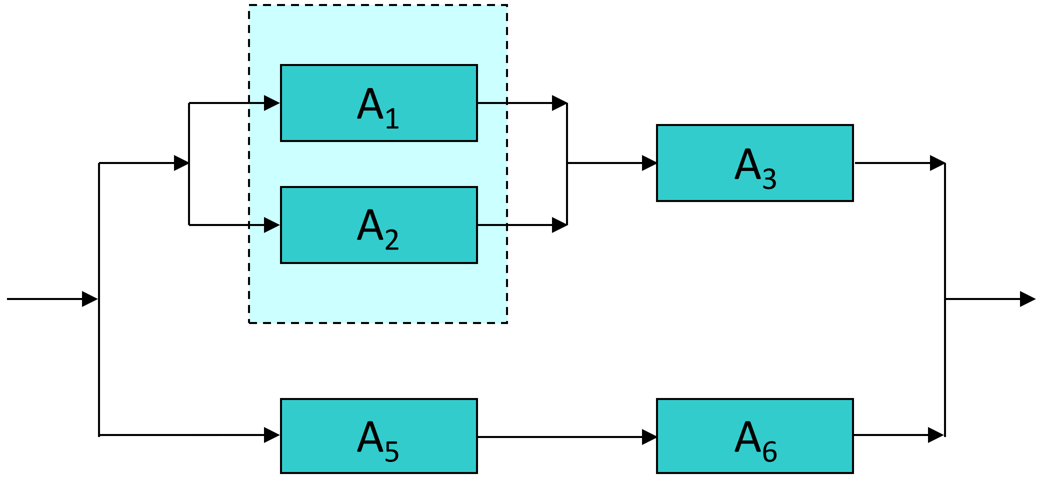
	Availability	Downtime
Component 1	99% (2-nines)	3.65 days/year
Component 2	99% (2-nines)	3.65 days/year
Combined	**99.99% (4-nines)**	**52 minutes/year**

- Downtime = (1-A)*365 days/year
- Even though components with very low availability are used, the overall availability of the system is much higher

- **Mission critical systems are designed with redundant components!**

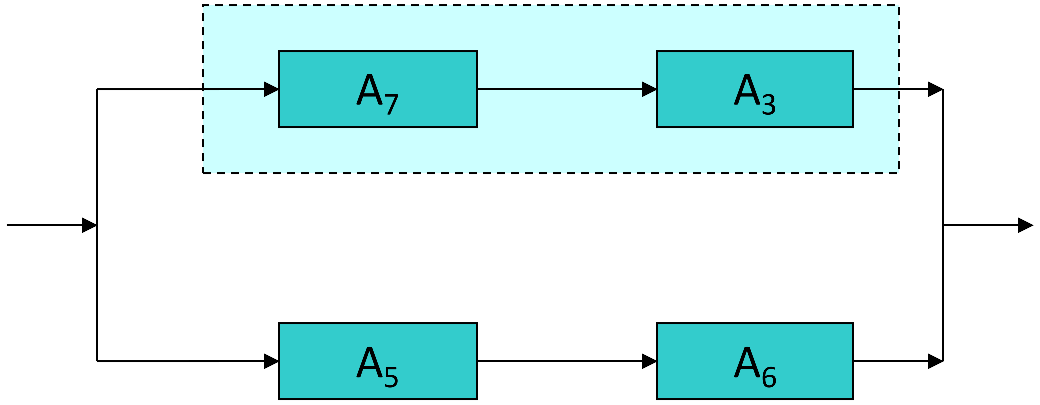
Availability of complex systems

$$A_7=1-(1-A_1)(1-A_2)$$

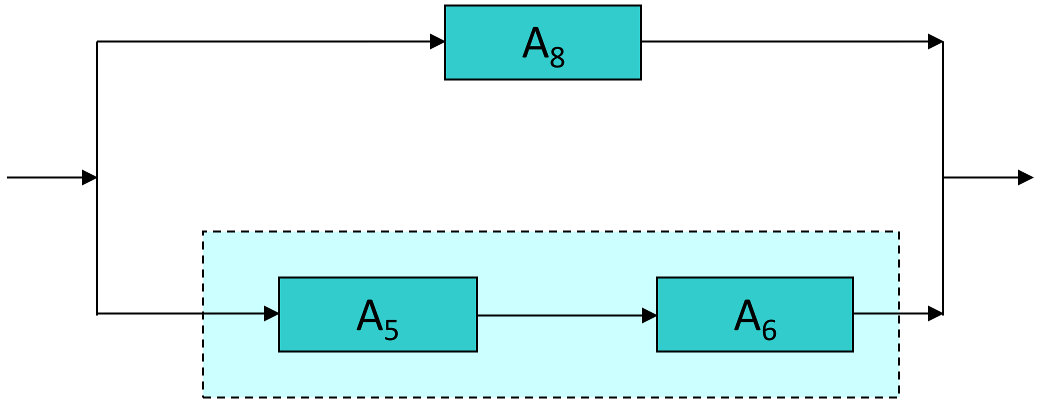


Availability of complex systems

$$A_8=A_7A_3$$



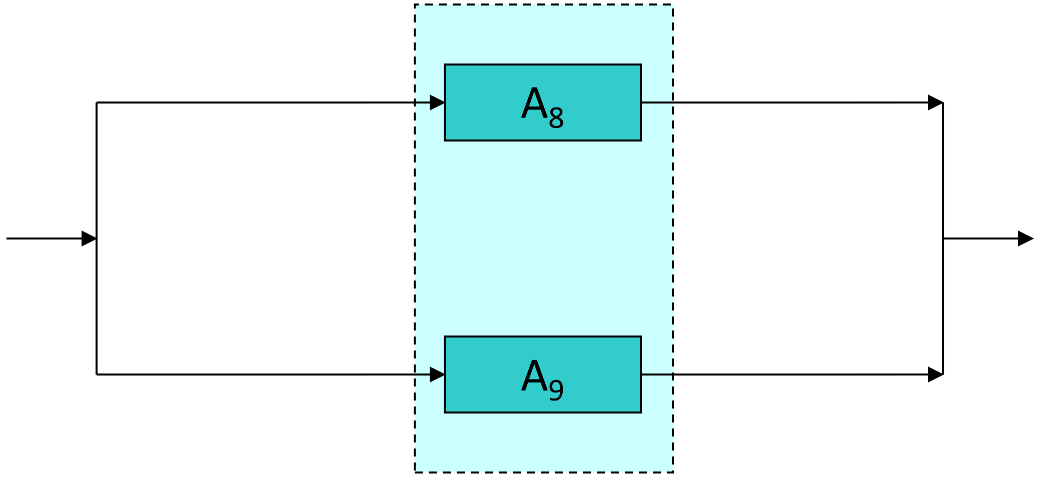
Availability of complex systems



$$A_9 = A_5 A_6$$

Availability of complex systems

$$A=1-(1-A_8)(1-A_9)$$

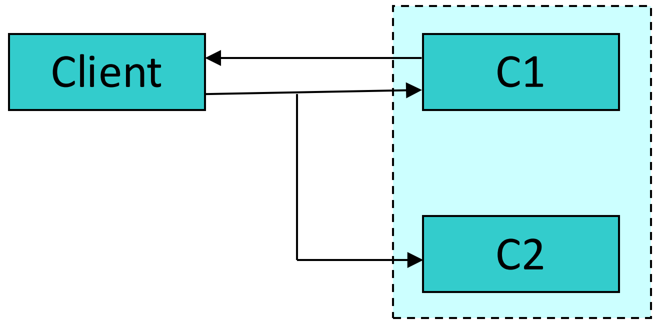


Tactics for Availability

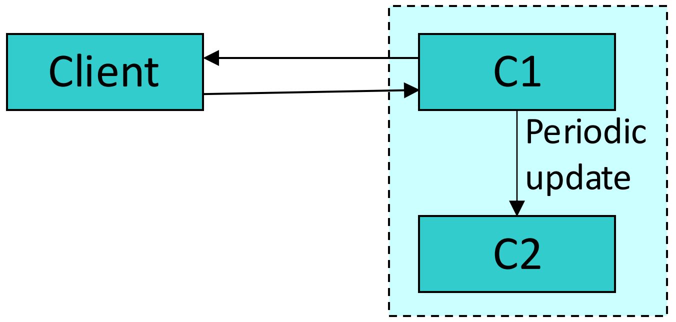
- Tactic = Design decisions that influence the control of one or more quality attributes

- **Replication**
  - Very simple to manage in case of stateless components
  - Various strategies in case of statefull components
- **Forward error recovery**
- **Circuit breaker** (see the patterns for microservices)

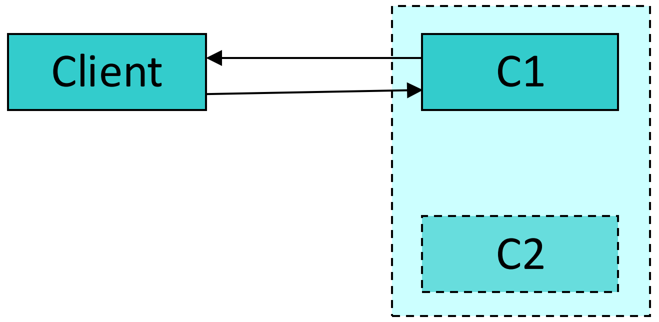
Replication approaches



Hot spare: C1 is leading, C2 is always ready to take over

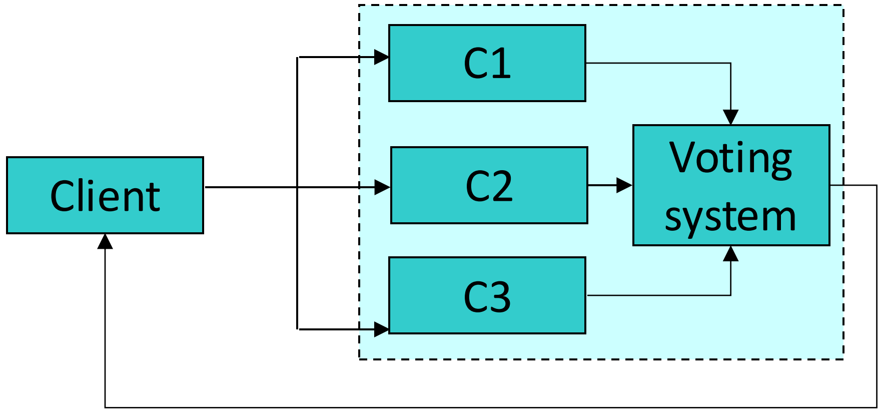


Warm spare: C1 is leading and periodically updating C2.
If C1 fails, some time might be needed to fully update C2

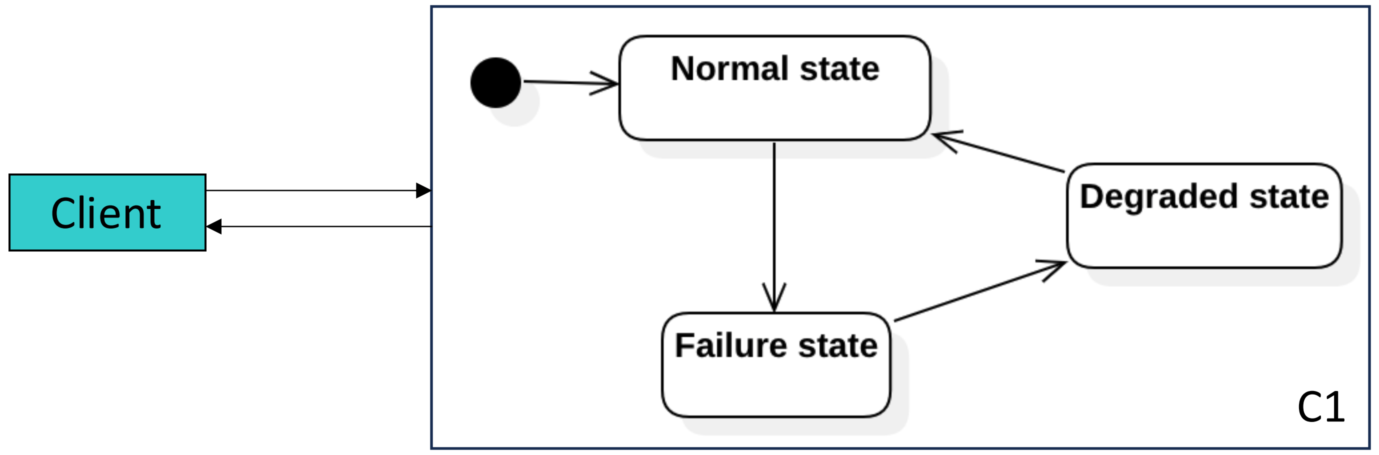


Cold spare: C2 is dormant and started and updated only when needed

Triple modular redundancy: C1, C2, and C3 are all active.
The produced result is the one produced by the majority.
Good when reliability is also important



Forward error recovery



- When C1 is in the failure state, a recovery mechanism moves it to the degraded state
- In the degraded state, C1 continues to be available even if not fully functional

EXTRA MATERIAL

Performance

- *Performance is an indicator of how well a software system or component meets its requirements for timeliness.*
  - Connie U. Smith, Lloyd G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, 2001, Addison-Wesley Professional.
- Sometimes, performance defined as efficient use of resources
- Performance is connected to scalability: ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases
- Multiple metrics available:
  - Response time
  - Throughput
  - CPU utilization
  - Memory utilization
  - I/O operations

EXTRA MATERIAL

Tactics for performance

POLITECNICO
MILANO 1863

- Control resource demand
  - Control input
    - Manage event arrival
    - Manage sampling rate
    - Bound event queues' size
    - Prioritize events
  - Improve efficiency of software
    - Reduce indirection
    - Co-locate interacting resources
    - Bound execution time
    - Improve algorithm efficiency
- Manage resources including computation and data
  - Increase resources
  - Introduce concurrency
  - Add multiple replicas and a load balancer
  - Add data replication and/or caching
  - Schedule resources
  - Split input and handle it

EXTRA MATERIAL

Introducing data replication - Twitter example

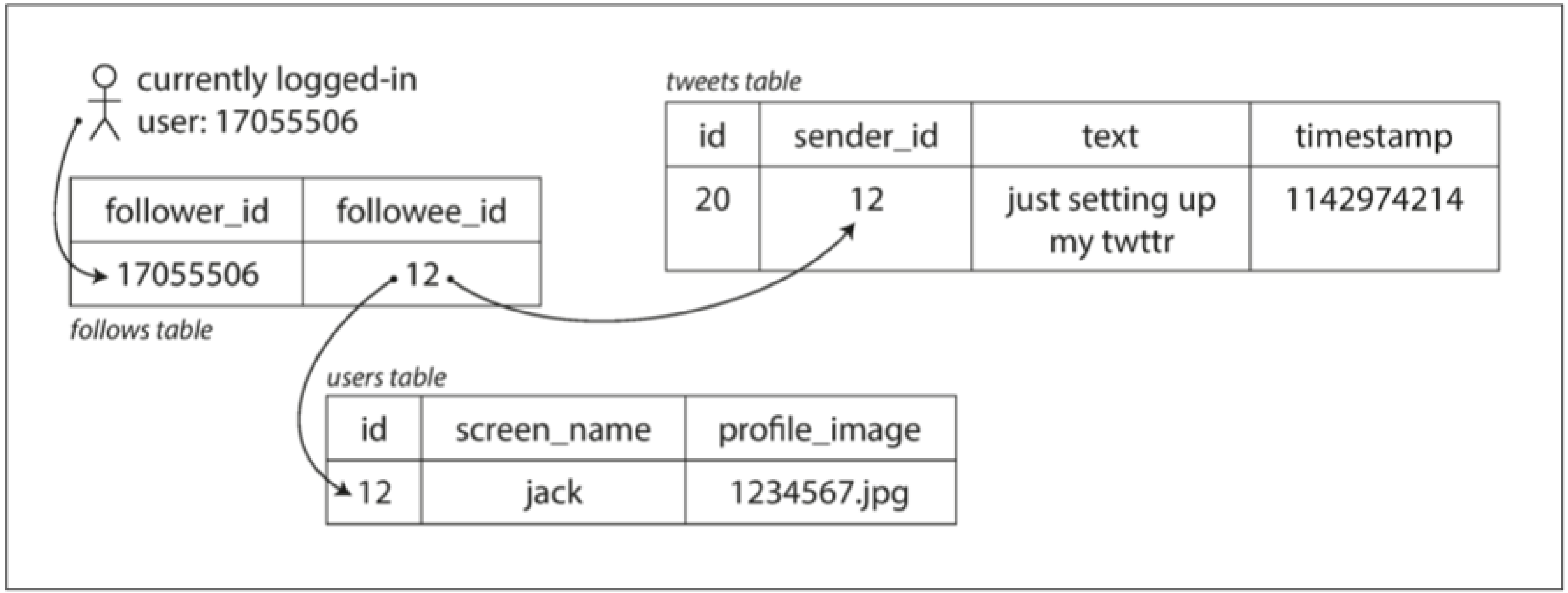
- Two main operations
  - Post tweet: a user can publish a message
    - Data from 2012: 4.6k requests/s on average, 12k requests/s at peak
  - Home timeline: a user can view the tweets posted by the people they follow
    - 300k requests/s
- Scalability challenge: cope with the number of connections between followers and followee

Martin Kleppmann, Designing Data-Intensive Applications : The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, Incorporated, Ed: 2017, ISBN: 9781449373320

EXTRA MATERIAL

Twitter example – design approach 1

- Tweets are stored in a tweets table
- Any time a user requests the home timeline, we look up at the followed people and retreve their tweets



follows table

follower_id	followee_id
17055506	12

tweets table

id	sender_id	text	timestamp
20	12	just setting up my twttr	1142974214

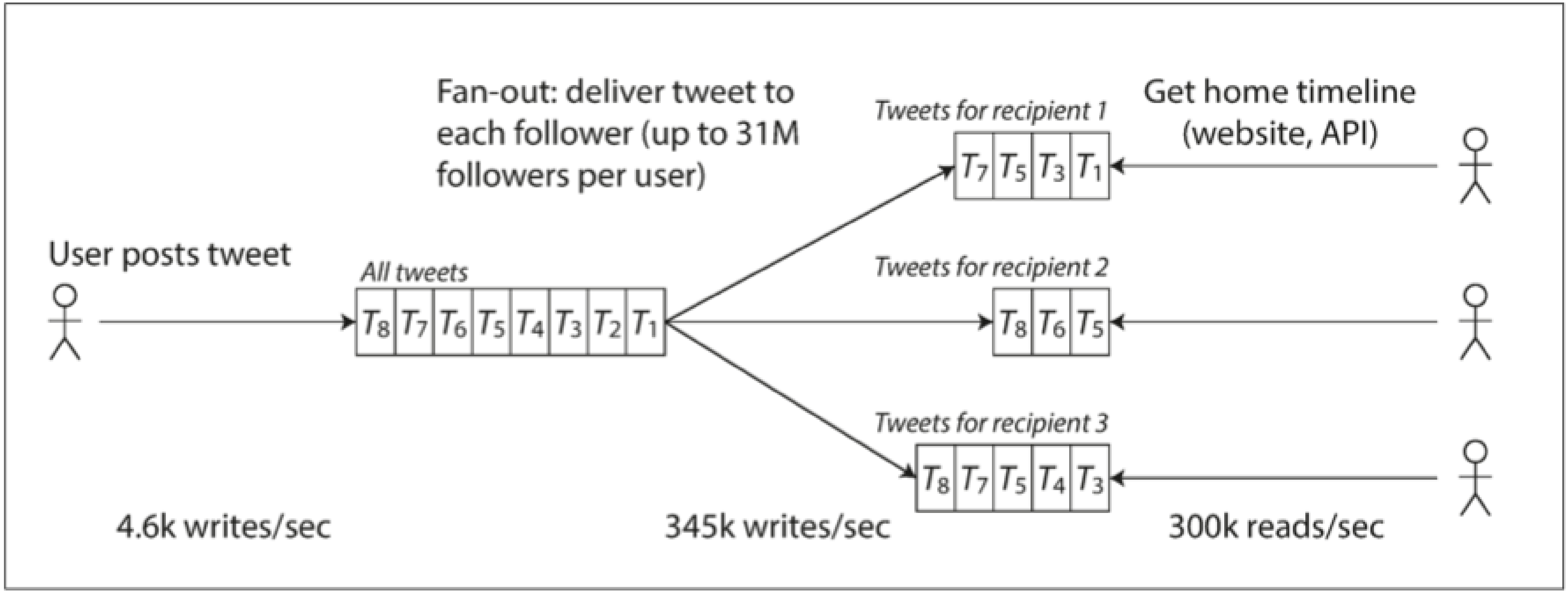
users table

id	screen_name	profile_image
12	jack	1234567.jpg

Martin Kleppmann, Designing Data-Intensive Applications : The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, Incorporated, Ed: 2017, ISBN: 9781449373320

EXTRA MATERIAL

Twitter example – design approach 2


Fan-out: deliver tweet to each follower (up to 31M followers per user)
Tweets for recipient 1
Get home timeline (website, API)
User posts tweet
All tweets
T_8 T_7 T_6 T_5 T_4 T_3 T_2 T_1
T_7 T_5 T_3 T_1
Tweets for recipient 2
T_8 T_6 T_5
Tweets for recipient 3
T_8 T_7 T_5 T_4 T_3
4.6k writes/sec
345k writes/sec
300k reads/sec

Martin Kleppmann, Designing Data-Intensive Applications : The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, Incorporated, Ed: 2017, ISBN: 9781449373320