Distributed Systems
Consistency & Replication II

Olaf Landsiedel

Notes

- If you like distributed systems
  - Distributed Systems II (TDA297), in LP3
- Project courses in the area of distributed systems
  - ICT Support for Adaptiveness and (Cyber)security in the Smart Grid
    - DAT300, in LP1, around 15 to 20 students
  - Autonomous and Cooperative Vehicular Systems
    - DAT295, in LP2, max 15 students

Labs

- Many ways to solve the tasks
  - We do not exactly tell you which approach to take
  - Some are better than others:
    - Very good ones, good ones and some that are not good
    - We want you to choose the (very) good ones
      - Justify / explain your choices in the video / documentation
  - Why do we do this?
    - This is a master level course
    - We want you to
      - Reason about different approaches
      - Learn to explain / justify your design choices

Last Time

- Consistency & Replication I
  - Introduction
  - Metrics for consistency
  - Data-centric consistency models
    - Total ordering
    - Sequential ordering
    - Causal ordering

Today…

- Consistency and Replication II
  - Client-Centric Consistency Models
  - Replica Management
  - Case Study: Content Delivery Networks (CDNs)

Recap: Trade-offs in Maintaining Consistency

- Maintaining consistency should balance between the strictness of c… cy versus efficiency
  - …uch consistency is “good-enough” depends on the … n

?

?

Loose Consistency

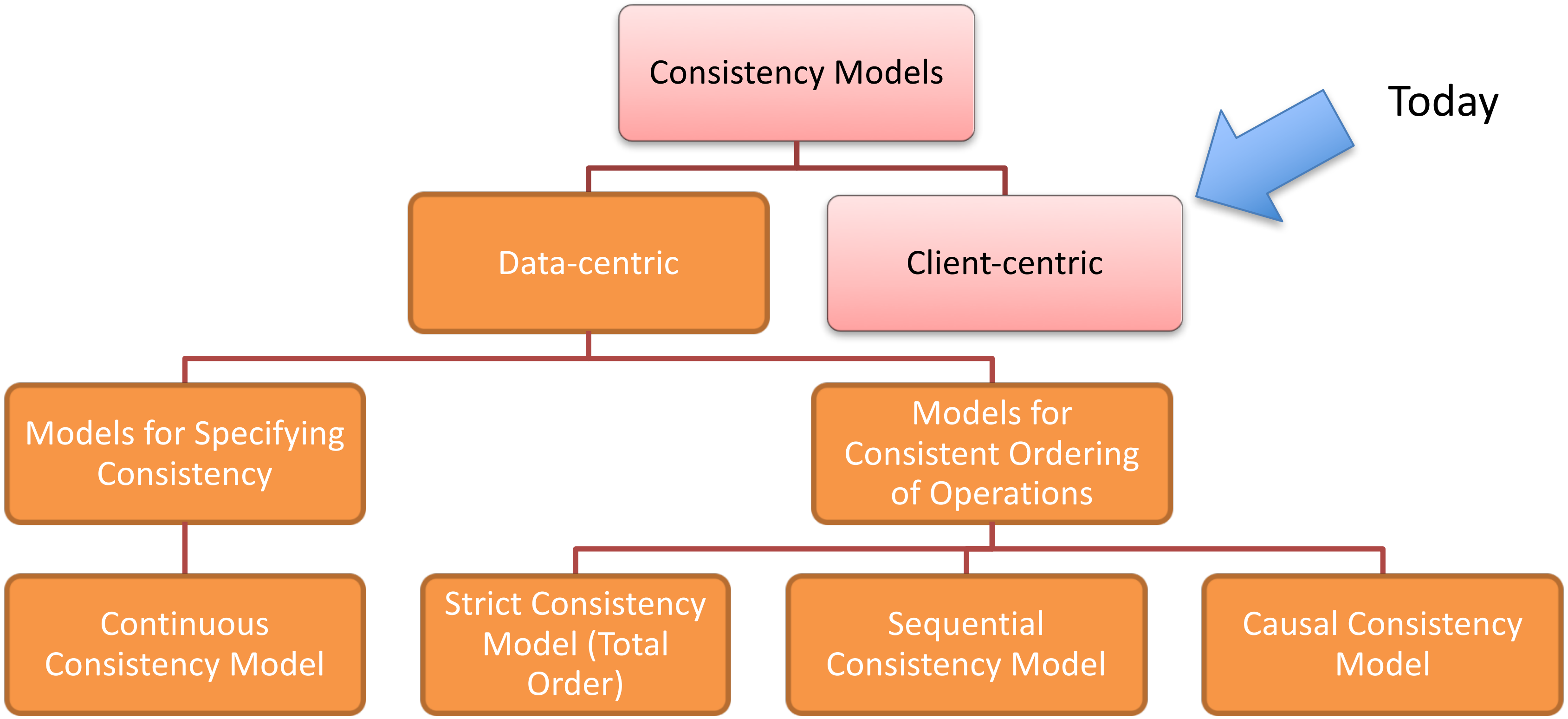
Strict Consistency

Easier to implement,
and is efficient

Generally hard to implement,
and is inefficient

Consistency Models

- A consistency model states the level of consistency provided by the *data-store* to the processes while reading and writing the data



Client-Centric Consistency Models

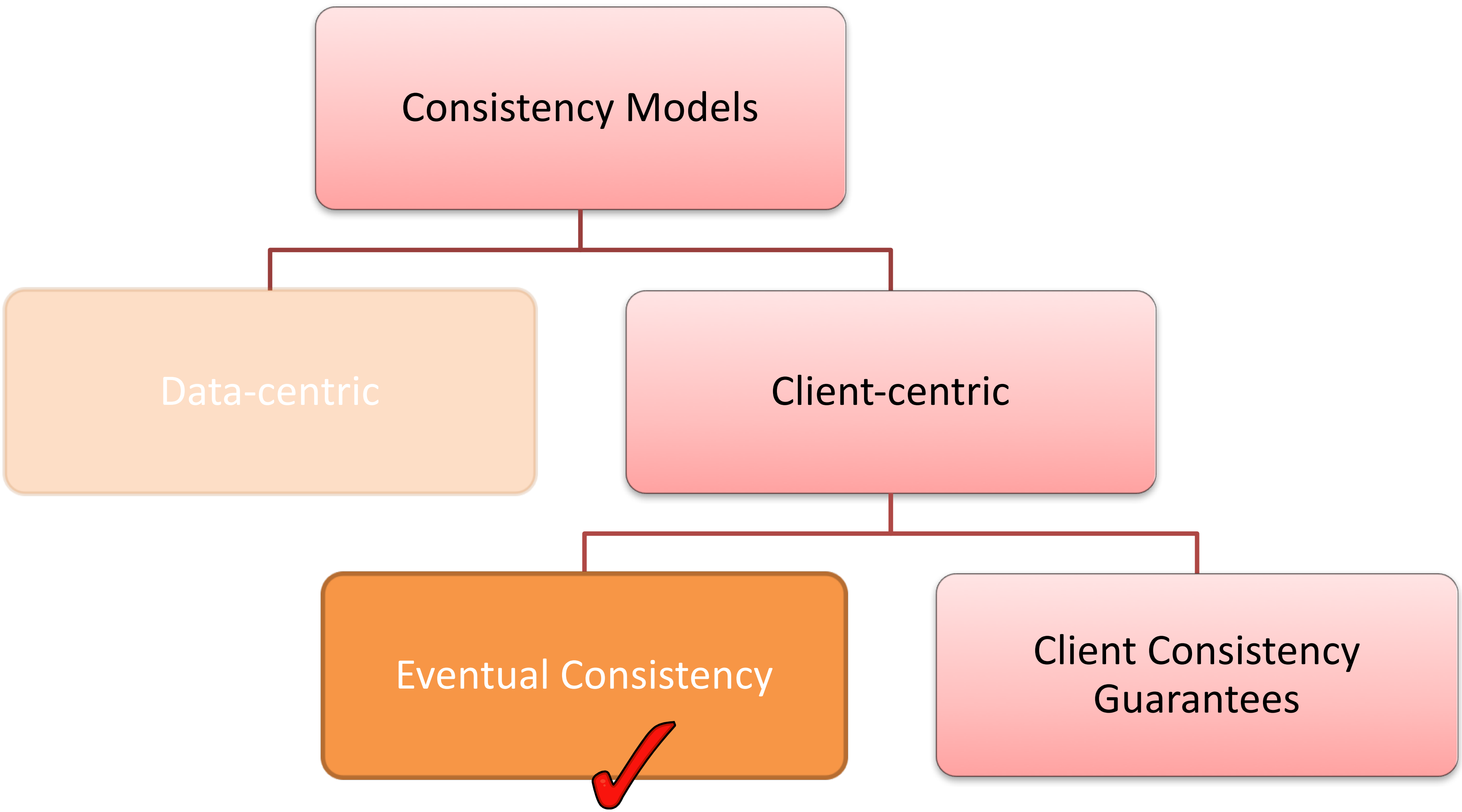
- Data-centric models lead to excessive overheads in applications where:
  - a majority operations are reads, and
  - updates occur frequently, and are often from one client process

- For such applications, a weaker form of consistency called Client-centric Consistency is employed for improving efficiency

Client-Centric Consistency Models

- Client-centric consistency models specify two requirements:
  - **Eventual Consistency**
    - All the replicas should eventually converge on a final value
  - **Client Consistency Guarantees**
    - Each client processes should be guaranteed some level of consistency while accessing the data value from different replicas

Overview


Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees

Eventual Consistency

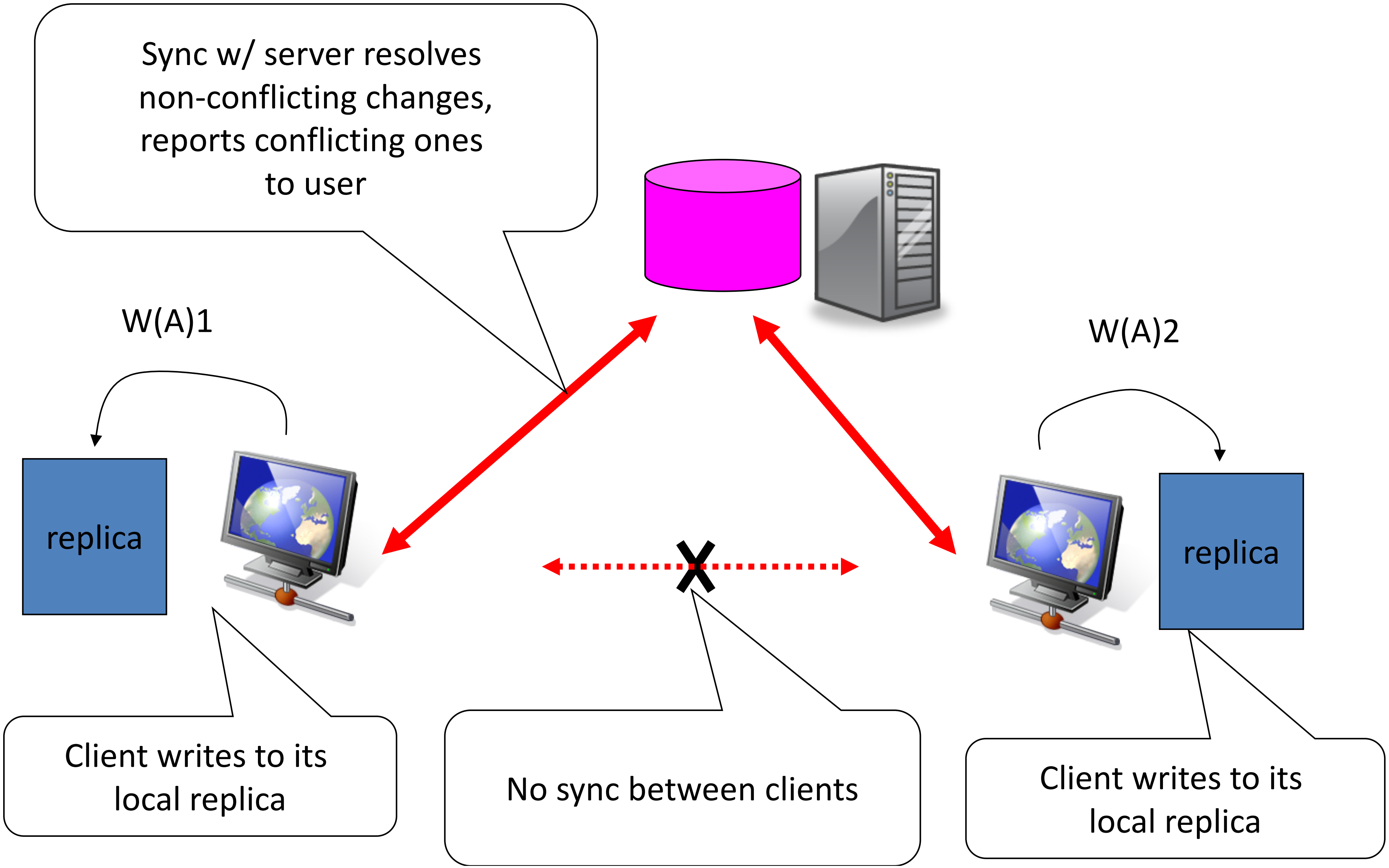
- Many applications can tolerate inconsistency for a long time
  - Webpage updates, Web Search – Crawling, indexing and ranking, Updates to DNS Server
  - Why? What others do you know?
- In such applications, it is acceptable and efficient if replicas in the data-store rarely exchange updates
- A data-store is termed as *Eventually Consistent* if:
  - All replicas will gradually become consistent in the absence of updates
- Typically, updates are propagated infrequently in eventually consistent data-stores

Designing Eventual Consistency

- In eventually consistent data-stores,
  - Write-write conflicts are rare
    - Two processes that write the same value are rare
    - Generally, one client updates the data value
      - e.g., One DNS server updates the name to IP mapping
    - Such rare conflicts can be handled through simple mechanisms, such as mutual exclusion

  - Read-write conflict are more frequent
    - Conflicts where one process is reading a value, while another process is writing a value to the same variable
    - Eventual Consistency Design has to focus on efficiently resolving such conflicts

Operating w/o total connectivity

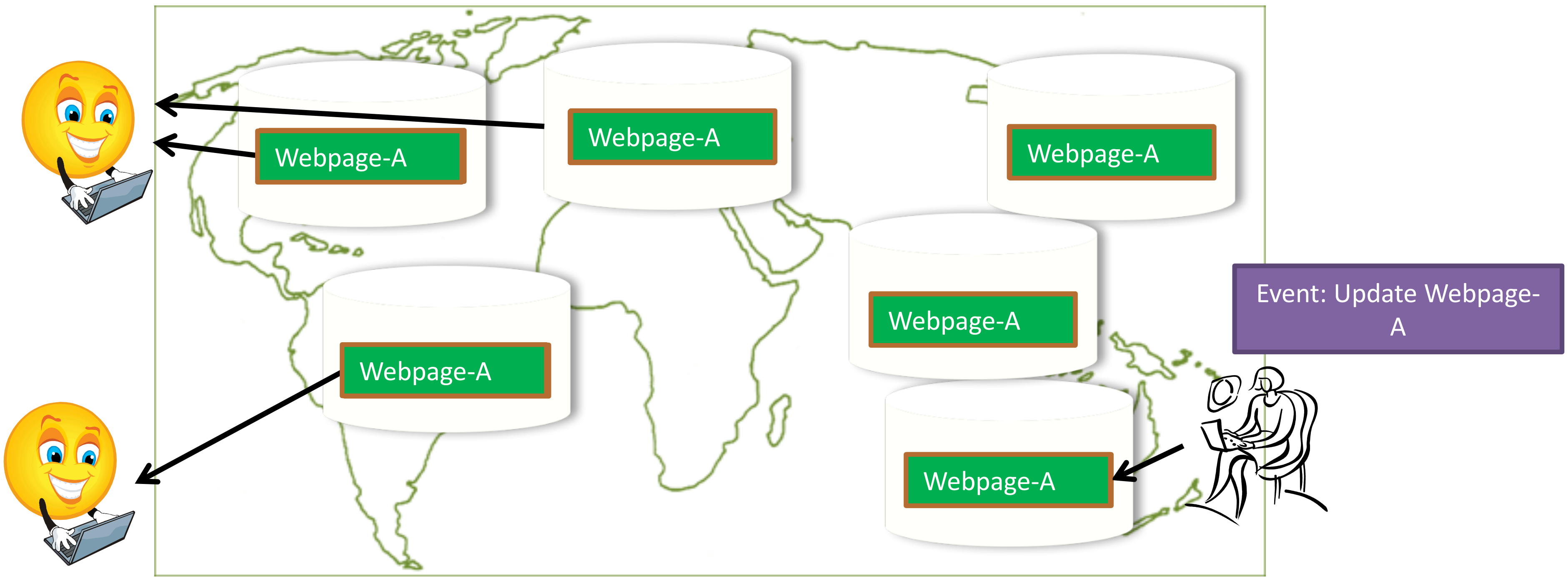


Eventual Consistency

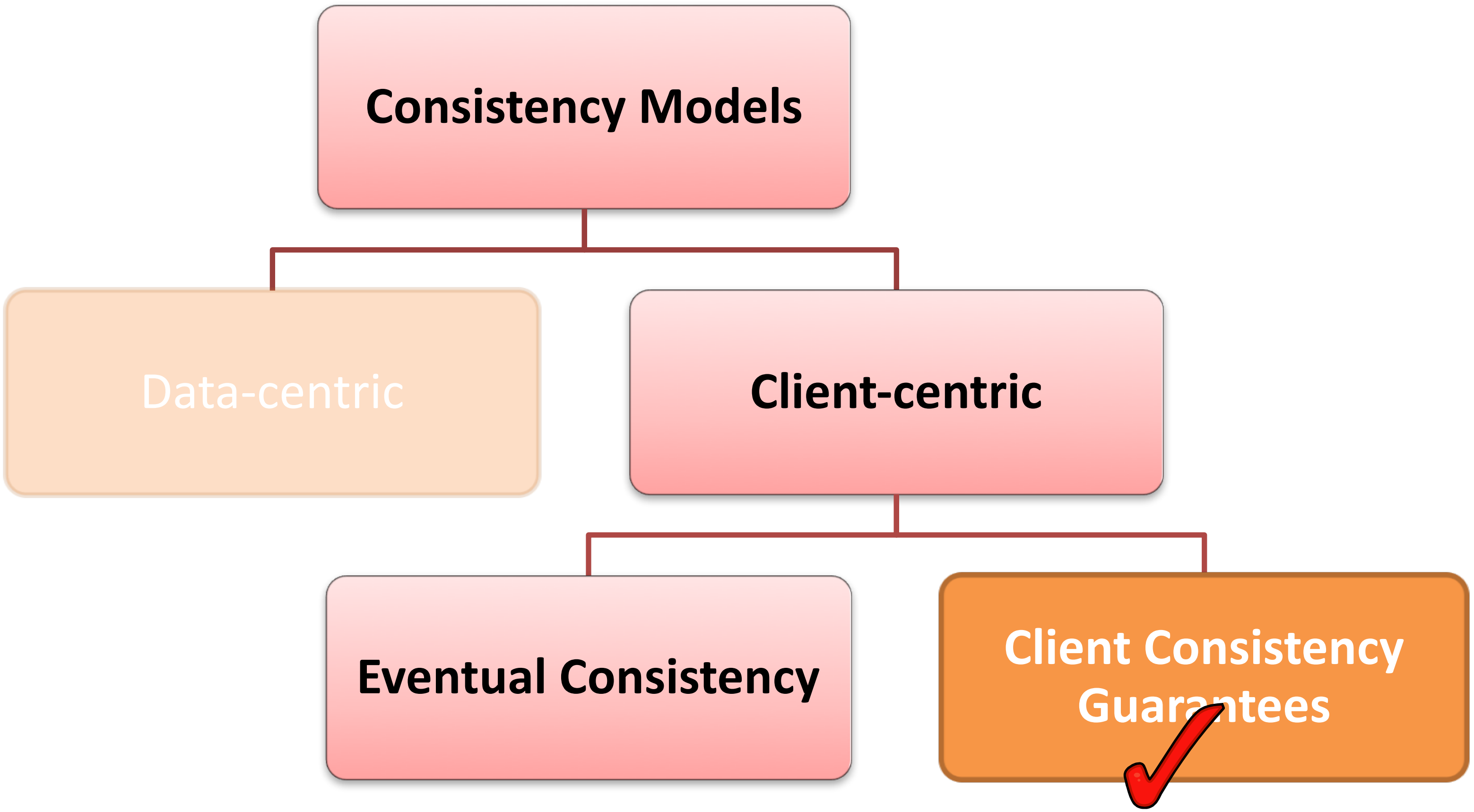
- Can operate with limited connectivity
  - Mobile environments
  - Handles networks failures well
- But
  - Very simple form of consistency
- Widespread in today's cloud
  - Google File System (GFS)
  - Facebook's Cassandra
  - Amazon's Dynamo
  - Your lab 3: Eventually Consistent Blackboard ;-)

Challenges in Eventual Consistency

- Eventual Consistency is not good-enough when the client process accesses data from different replicas
  - We need consistency guarantees for a single client while accessing the data-store: next topic



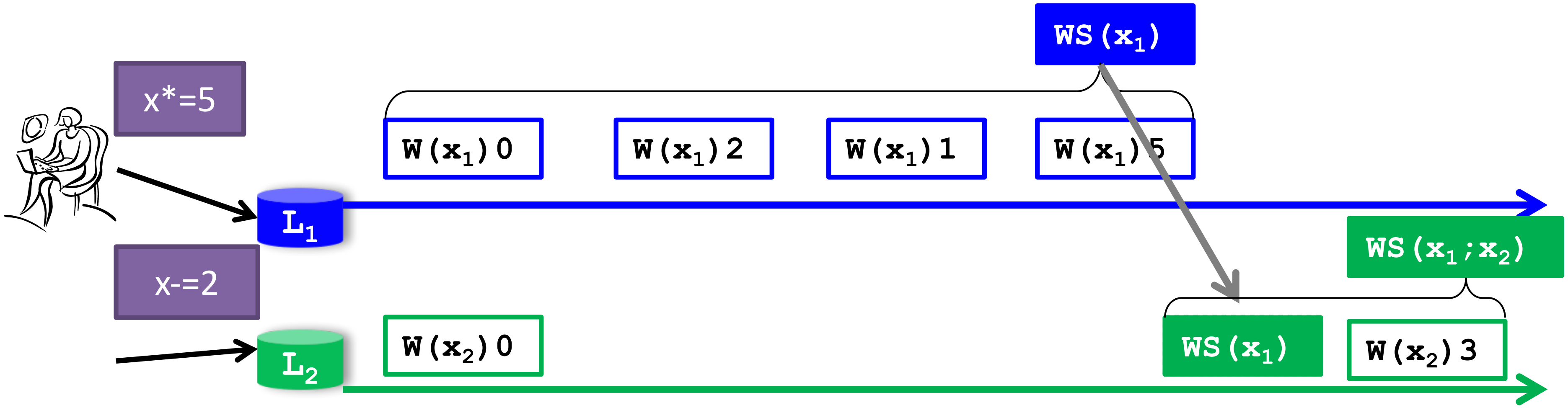
Overview



Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees

Client Consistency Guarantees

- Client-centric consistency provides guarantees for a single client for its accesses to a data-store
- Example: Providing consistency guarantee to a client process for data $\mathbf{x}$ replicated on two replicas. Let $\mathbf{x_i}$ be the local copy of a data $\mathbf{x}$ at replica $\mathbf{L_i}$.



$\mathtt{WS(x_1)}$ = **Write Set for $\mathbf{x_1}$** = Series of ops being done at some replica that reflects how $\mathbf{L_1}$ updated $\mathbf{x_1}$ till this time

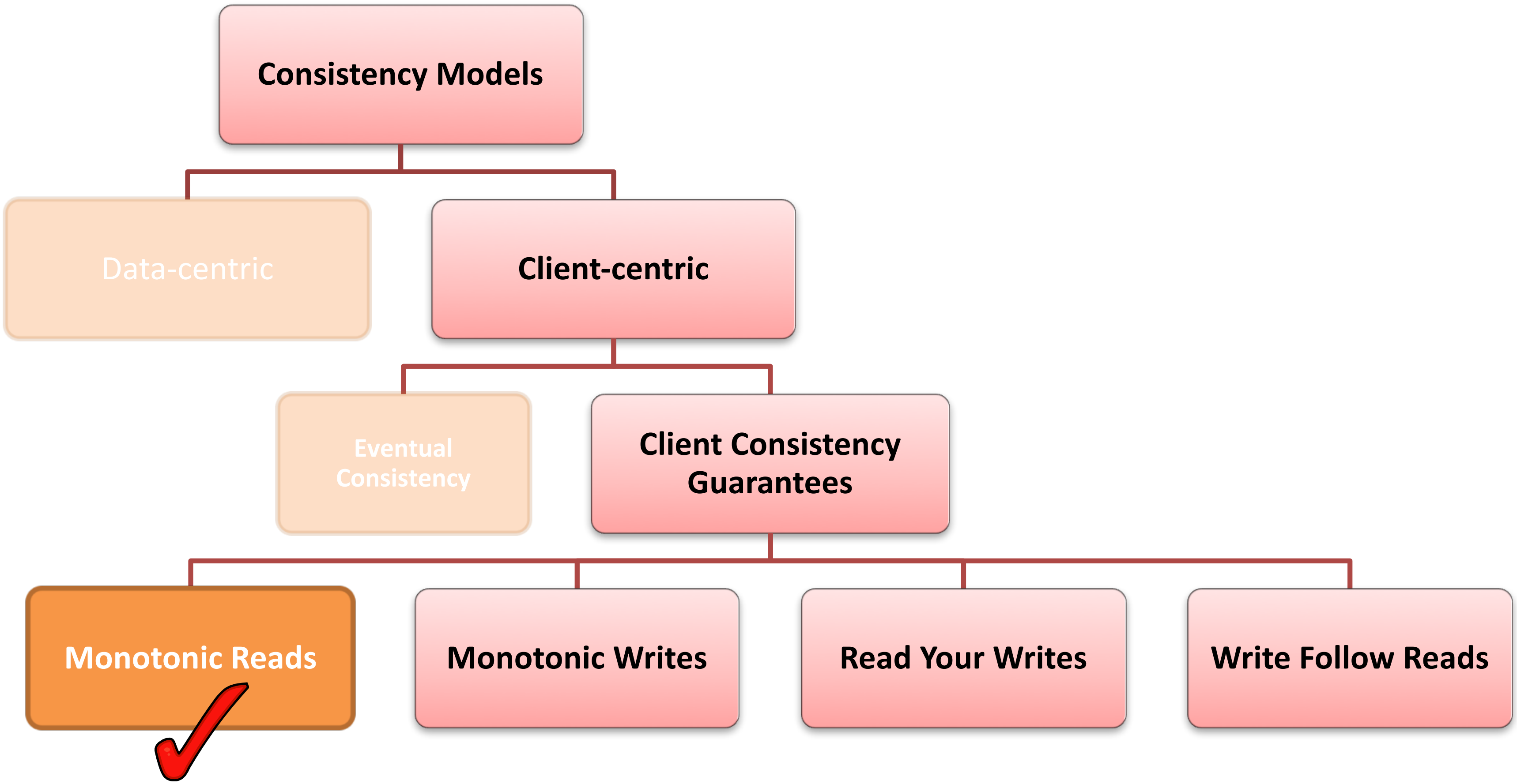
$\mathtt{WS(x_1;x_2)}$ = **Write Set for $\mathbf{x_1}$ and $\mathbf{x_2}$** = Series of ops being done at some replica that reflects how $\mathbf{L_1}$ updated $\mathbf{x_1}$ and, later on, how $\mathbf{x_2}$ is updated on $\mathbf{L_2}$

$\mathbf{L_i}$ = Replica i | $\mathtt{R(x_i)b}$ = Read variable x at replica i; Result is b | $\mathtt{W(x)b}$ = Write variable x at replica i; Result is b | $\mathtt{WS(x_i)}$ = Write Set

Client Consistency Guarantees

- We will study four types of client-centric consistency models
  1. Monotonic Reads
  2. Monotonic Writes
  3. Read Your Writes
  4. Write Follow Reads

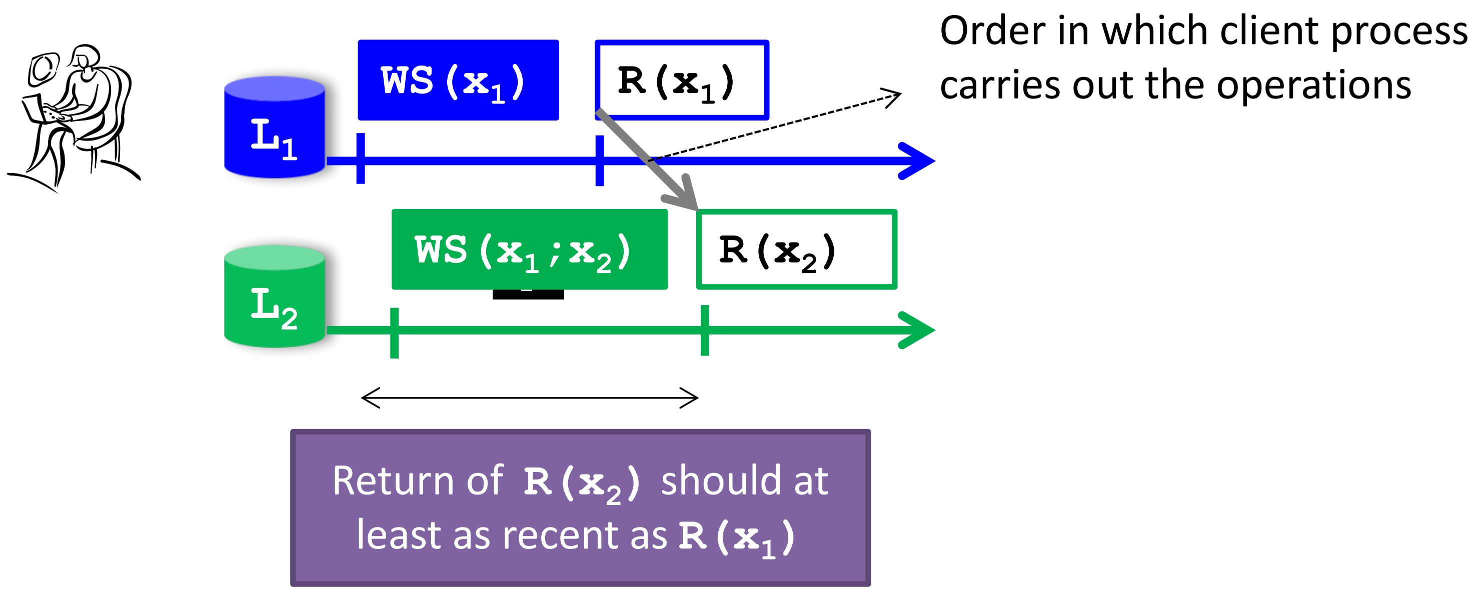
Overview



Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Monotonic Reads

- The model provides guarantees on successive reads
- If a client process reads the value of data item **x**, then any successive read operation by that process should return the <u>same</u> or a <u>more recent value</u> for **x**



Monotonic Reads – Puzzle

Recognize data-stores that provide monotonic read guarantees

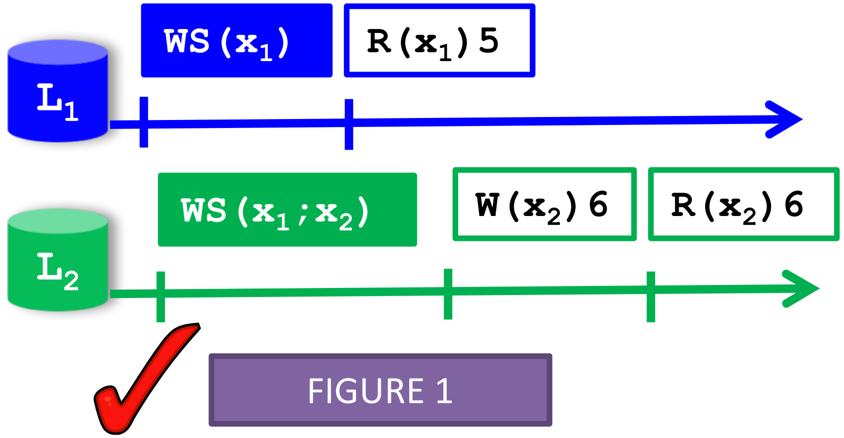


FIGURE 1

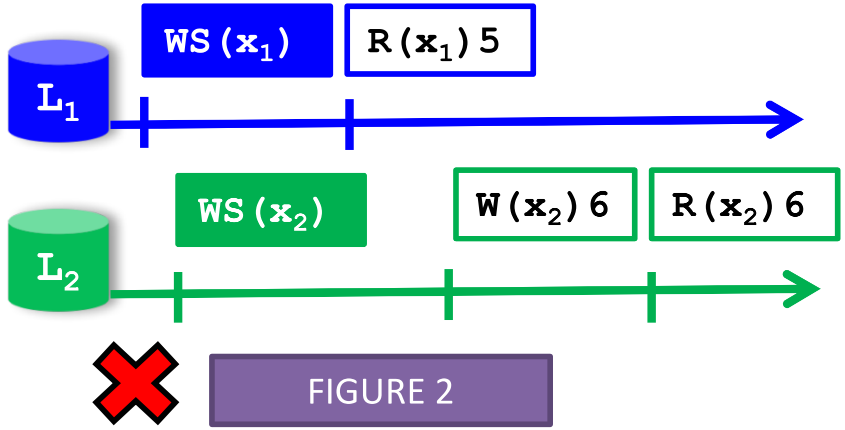


FIGURE 2

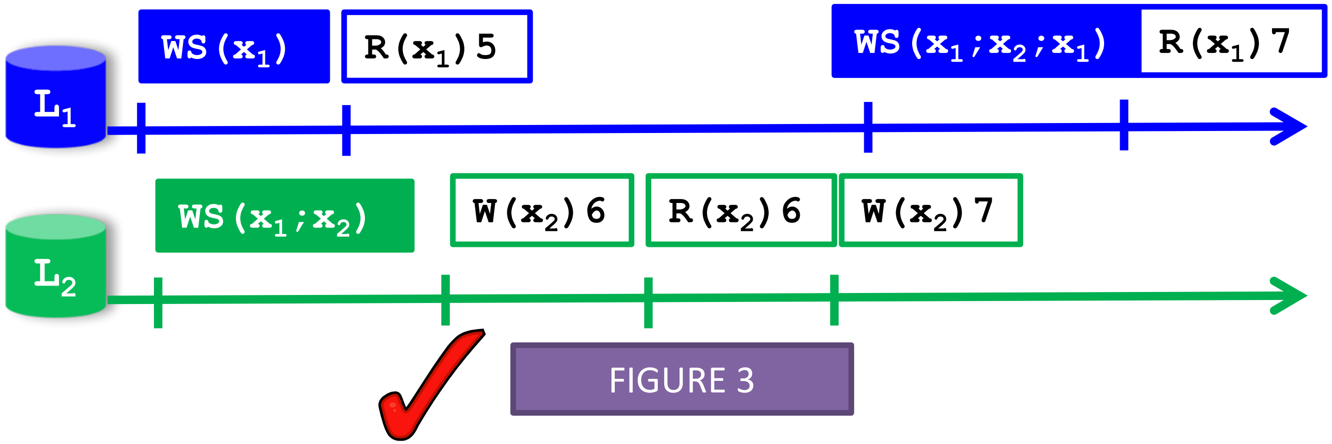
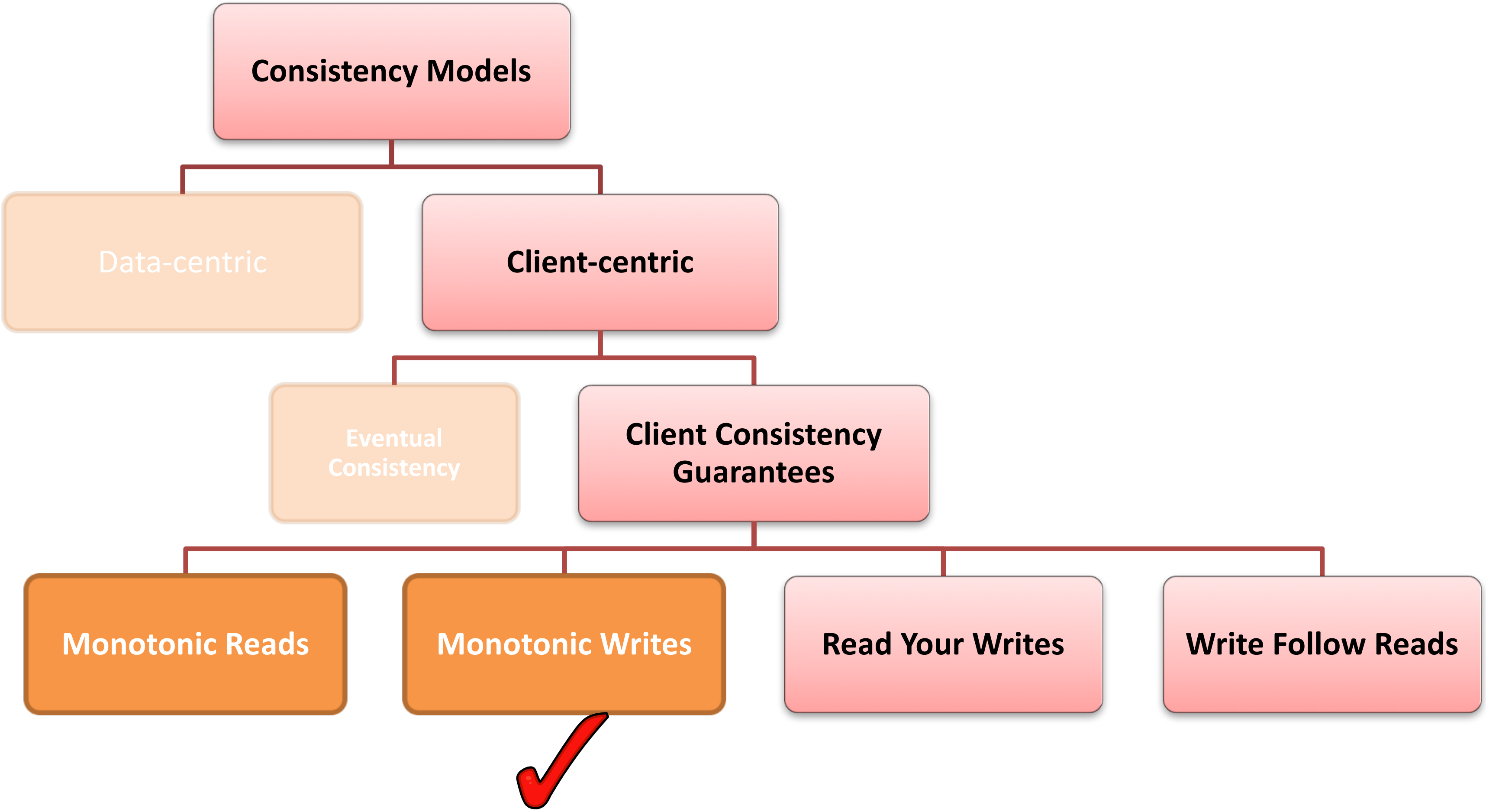


FIGURE 3

Monotonic Reads

- How to implement?
  - On read: check with all other replicas
    - To exchange write sets (WS)
    - Problem: inefficient
  - Alternative?
    - Ask client: from which replica did you read last
- Similar approach
  - Also for other consistency models

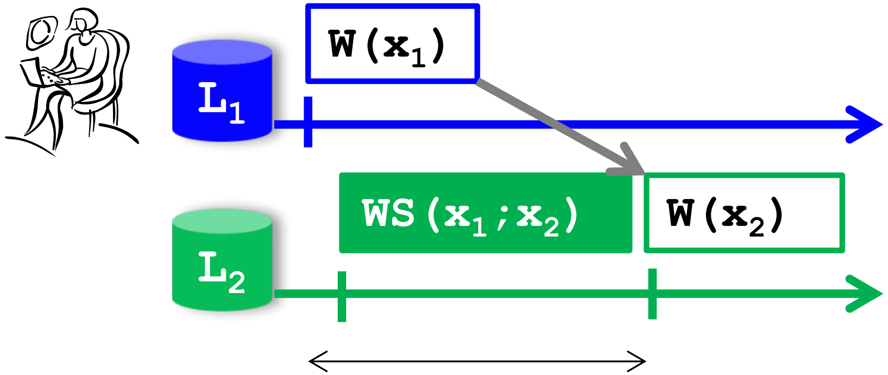
Overview



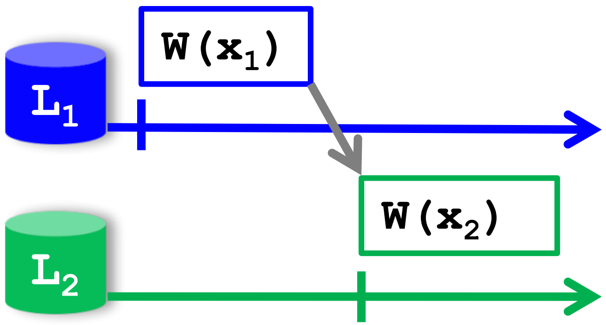
Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Monotonic Writes

- This consistency model assures that writes are monotonic
- A write operation by a client process on a data item **x** is completed before any successive write operation on **x** by the same process
  - A new write on a replica should wait for all old writes on any replica



$W(x_2)$ operation should be performed only after the result of $W(x_1)$ has been updated at L_2



The data-store does not provide monotonic write consistency

Monotonic Writes – Puzzle

Recognize data-stores that provide monotonic write guarantees

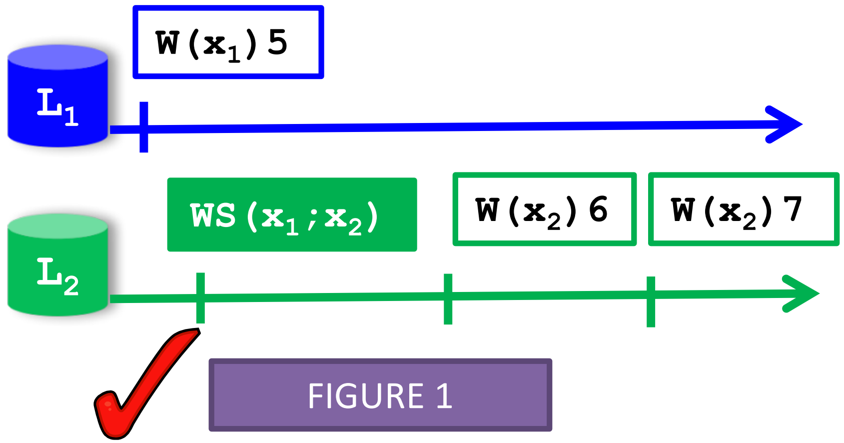


FIGURE 1

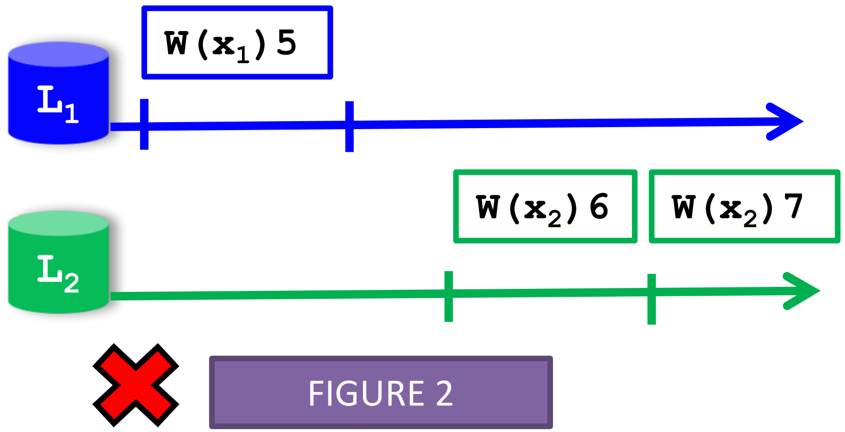


FIGURE 2

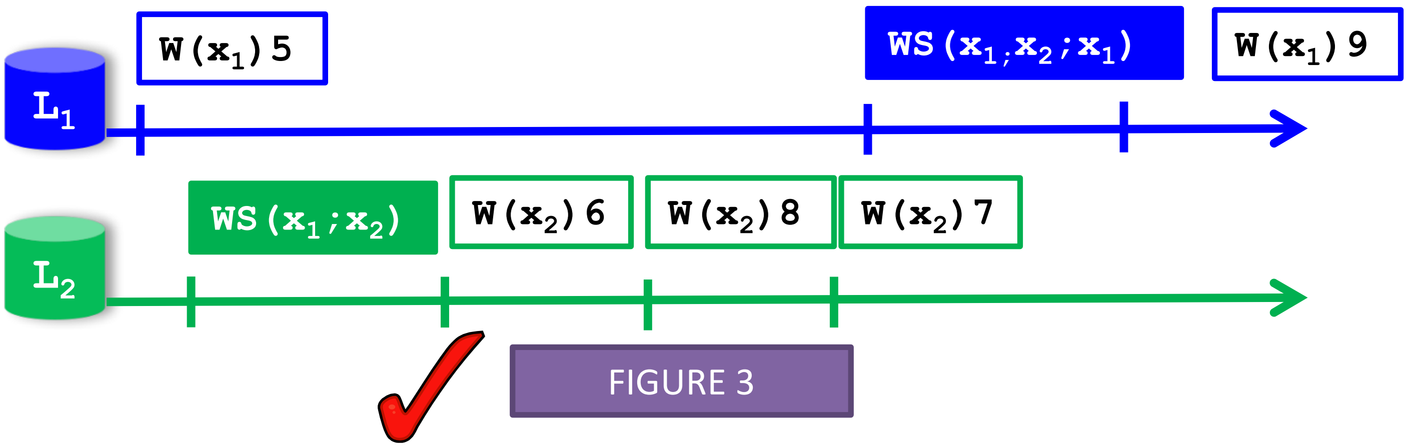
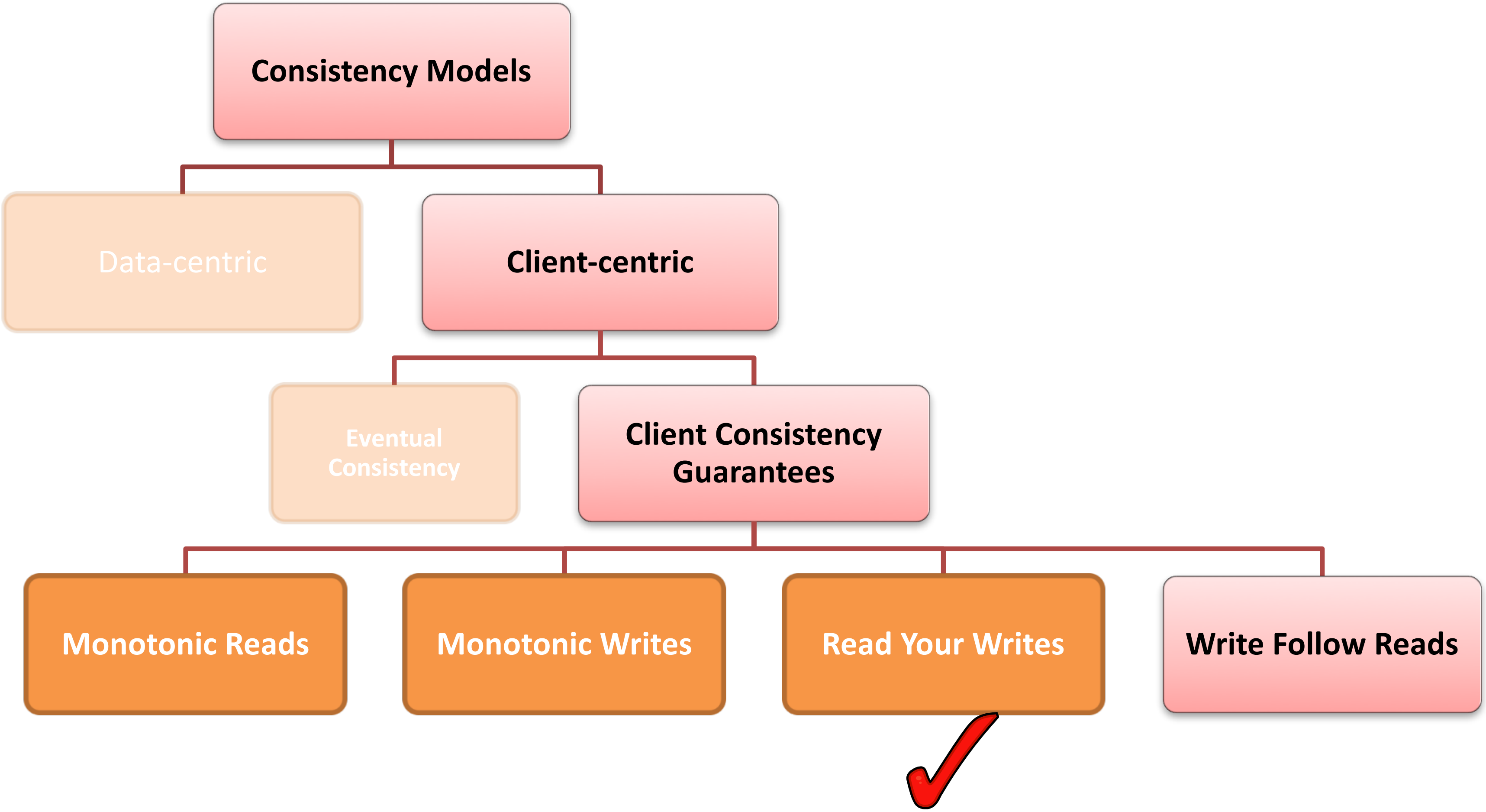


FIGURE 3

Monotonic Writes – An Example

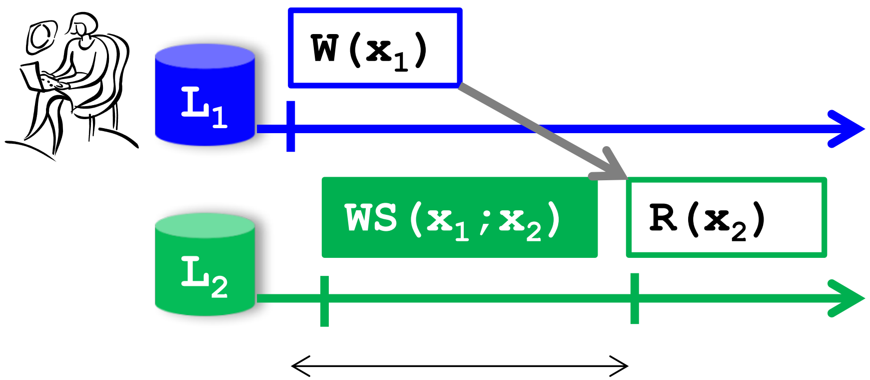
- Example: Updating individual libraries in a large software source code which is replicated
  - Updates can be propagated in a lazy fashion
  - Updates are performed on a part of the data item
    - Some functions in an individual library is often modified and updated
  - Monotonic writes: If an update is performed on a library, then all preceding updates on the same library are first updated

Overview

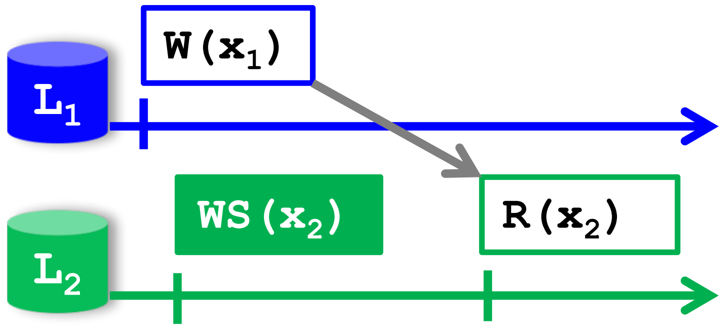

Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Read Your Writes

- The <u>effect of a write</u> operation on a data item **x** by a process will <u>always be seen by a successive read</u> operation on **x** by the same process
- Example scenario:
  - In systems where password is stored in a replicated data-base, the password change should be seen immediately

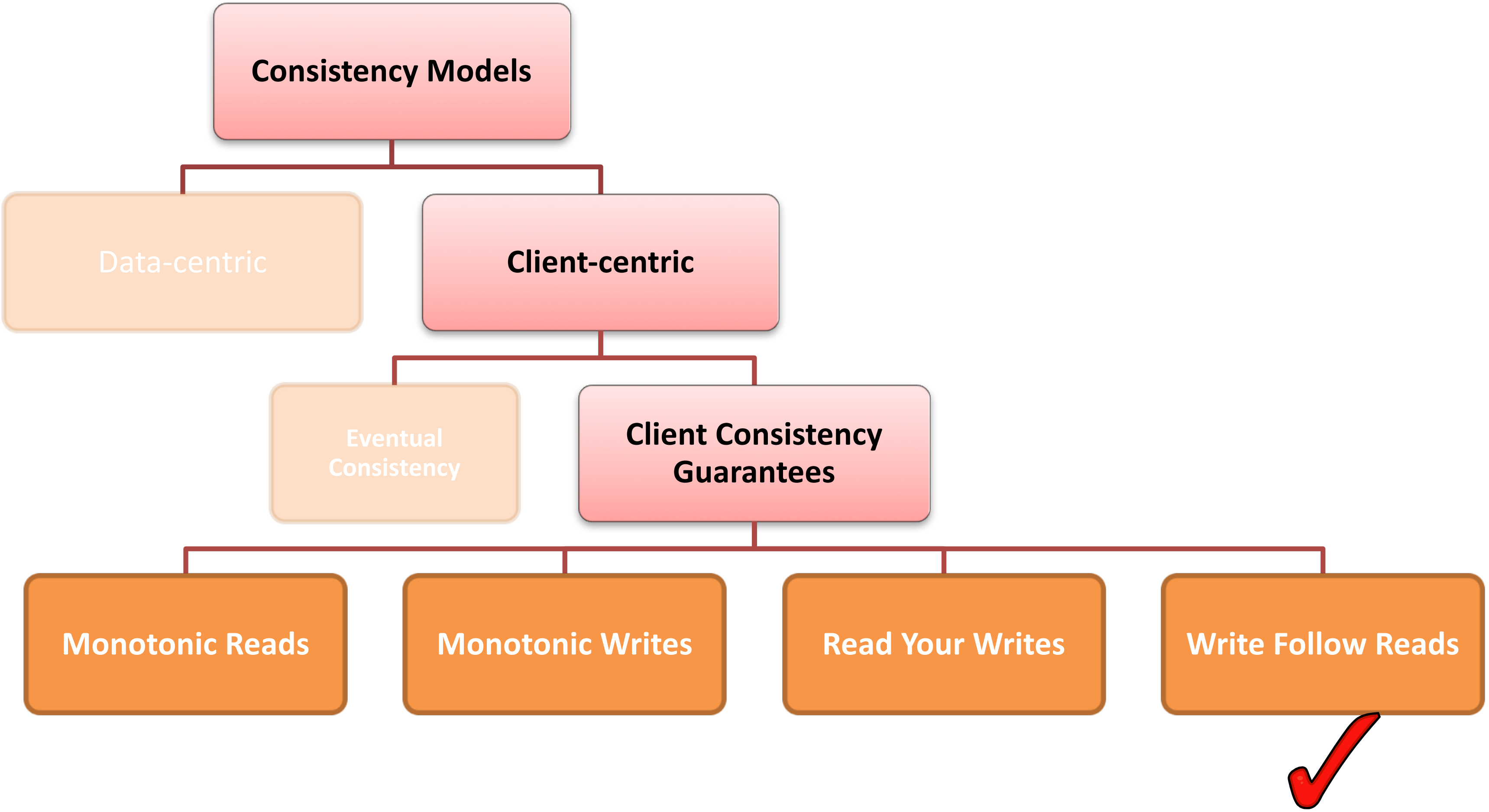


$R(x_2)$ operation should be performed only after the updating the Write Set $WS(x_1)$ at L_2



A data-store that does not provide *Read Your Write* consistency

Overview

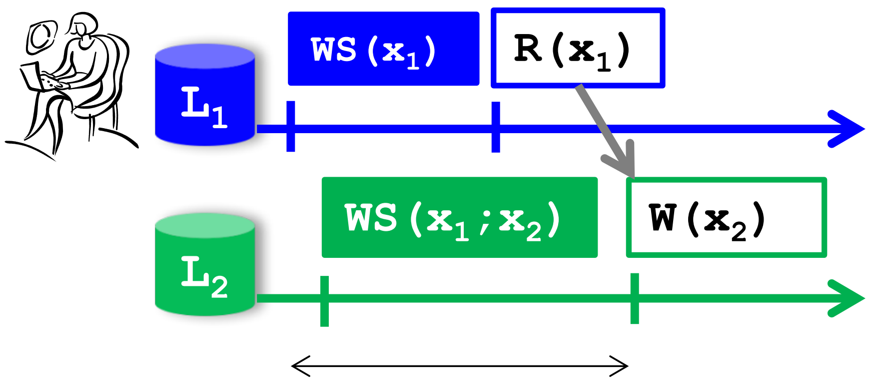


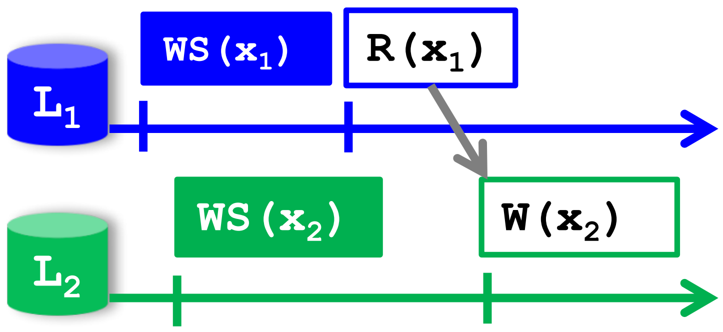
Consistency Models
Data-centric
Client-centric
Eventual Consistency
Client Consistency Guarantees
Monotonic Reads
Monotonic Writes
Read Your Writes
Write Follow Reads

Write Follow Reads

- A write operation by a process on a data item **x** following a previous read operation on **x** by the same process is guaranteed to take place on the same or a more recent value of **x** that was read

- Example scenario:
  - Users of a newsgroup should post their comments only after they have read all previous comments





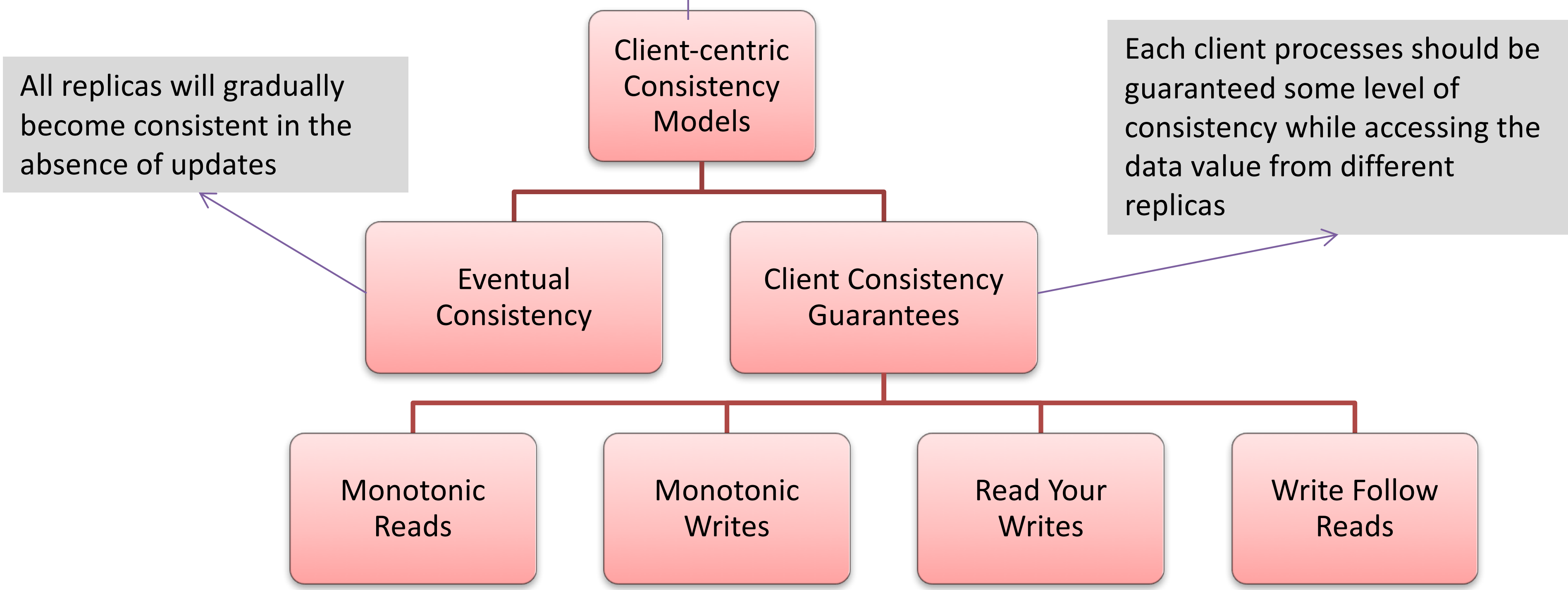
$W(x_2)$ operation should be performed only after the all previous writes have been seen

A data-store that does not guarantee Write Follow Read Consistency Model

Summary of Client-centric Consistency Models

Client-centric Consistency Model defines how a data-store presents the data value to an individual client when the client process accesses the data value across different replicas.
It is generally useful in applications where:

- one client always updates the data-store.
- read-to-write ratio is high



Your turn!

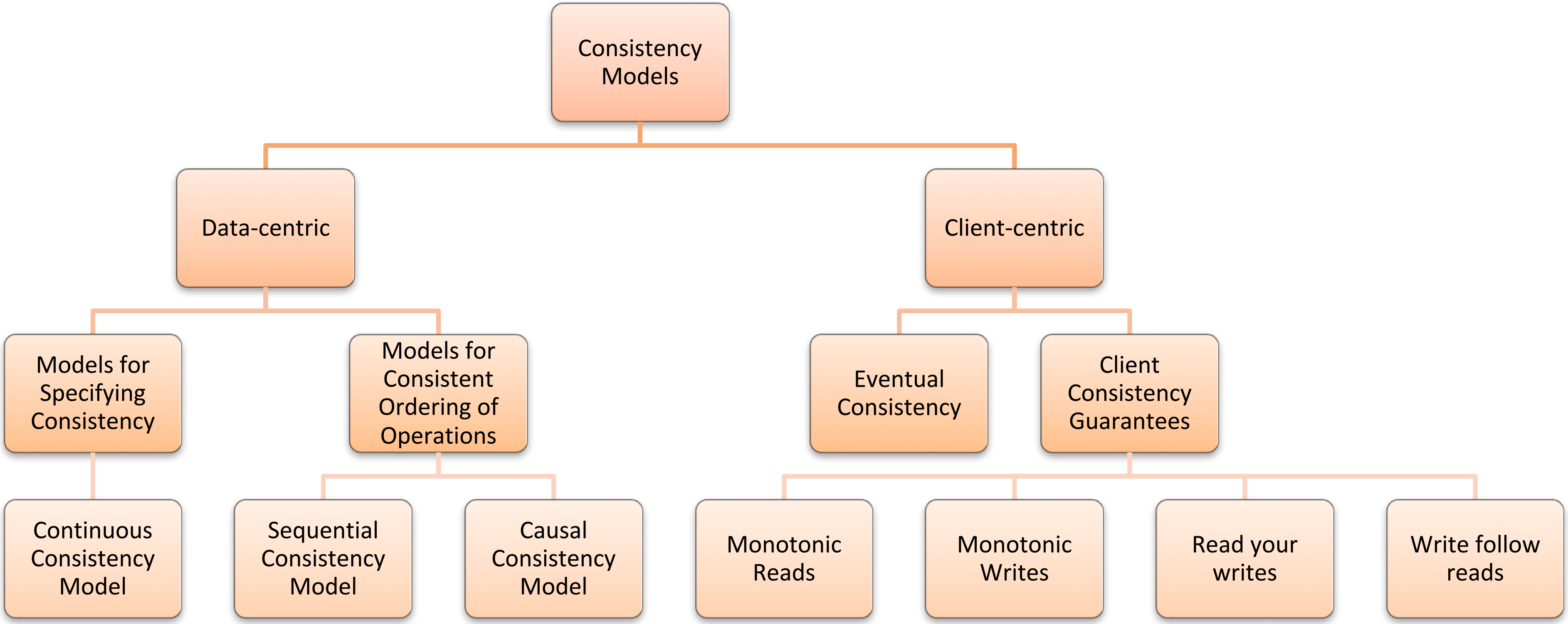
- Eventual Consistency is easy to implement (when compared to other consistency models)
- Eventual Consistency is very common in distributed systems
- Eventual Consistency is a weak consistency model
- Which consistency models to use for a Inbox of a email program?

http://olafland.polldaddy.com/s/cons

Last question: no clear answer

- Q1 to Q3: answer is yes
- Q4: Which consistency models to use for a Inbox of a email program?
  - Monotonic Reads: on reconnect: should also see all emails I have seen before: yes
  - Write follow reads: should reply to emails only when I have seen all previous ones: yes
  - The other two ones: we can argue

Topics covered in Consistency Models



Summary of Consistency Models

- Different applications require different levels of consistency
  - Data-centric consistency models
    - Define how replicas in a data-store maintain consistency
  - Client-centric consistency models
    - Provide an efficient, but weaker form of consistency when
    - Here, one client process updates the data item, and many processes read the replica

Today…

- Consistency and Replication II
  - Client-Centric Consistency Models
  - Replica Management
  - Case Study: Content Delivery Networks (CDNs)

Replica Management

- Replica management describes <u>where</u>, <u>when</u> and <u>by whom</u> replicas should be placed

- We will study two problems under replica management
  1. Replica-Server Placement
     - Decides the best locations to place the replica server that can host data-stores

  2. Content Replication and Placement
     - Finds the best server for placing the contents

Today…

- Consistency and Replication II
  - Client-Centric Consistency Models
  - Replica Management
    - Replica Server Placement
    - Content Replication and Placement
  - Case Study: Content Delivery Networks (CDNs)

Replica Server Placement

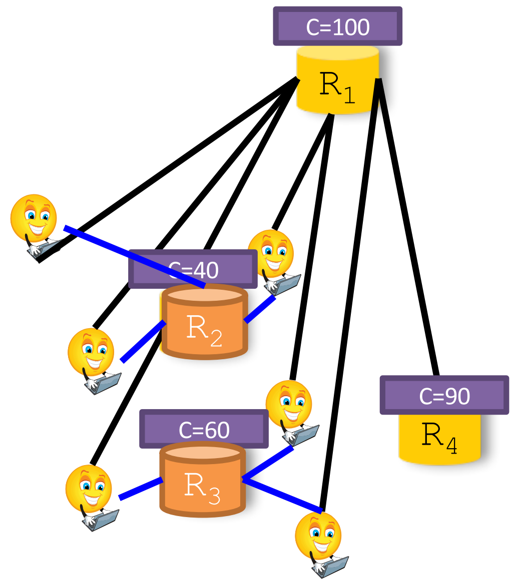
- Factors that affect placement of replica servers:
  - What are the possible locations where servers can be placed?
    - Should we place replica servers close-by or distribute it uniformly?
  - How many replica servers can be placed?
    - What are the trade-offs between placing many replica servers vs. few?
  - How many clients are accessing the data from a location?
    - More replicas at locations where most clients access improves performance and fault-tolerance
- If **K** replicas have to be placed out of **N** possible locations, find the *best **K** out of **N*** locations **(K<N)**

Replica Server Placement – An Example Approach

- Problem: **`K`** replica servers should be placed on some of the **`N`** possible replica sites such that
  – Clients have low-latency/high-bandwidth connections

- Example: Greedy Approach

1. Evaluate the cost of placing a replica on each of the **`N`** potential sites
   + Examining the *cost* of **`C`** clients connecting to the replica
   + Cost of a link can be **`1/bandwidth`** or **`latency`**
2. Choose the lowest-cost site
3. In the second iteration, search for a second replica site which, in conjunction with the already selected site, yields the lowest cost
4. Iterate steps 2,3 and 4 until **`K`** replicas are chosen



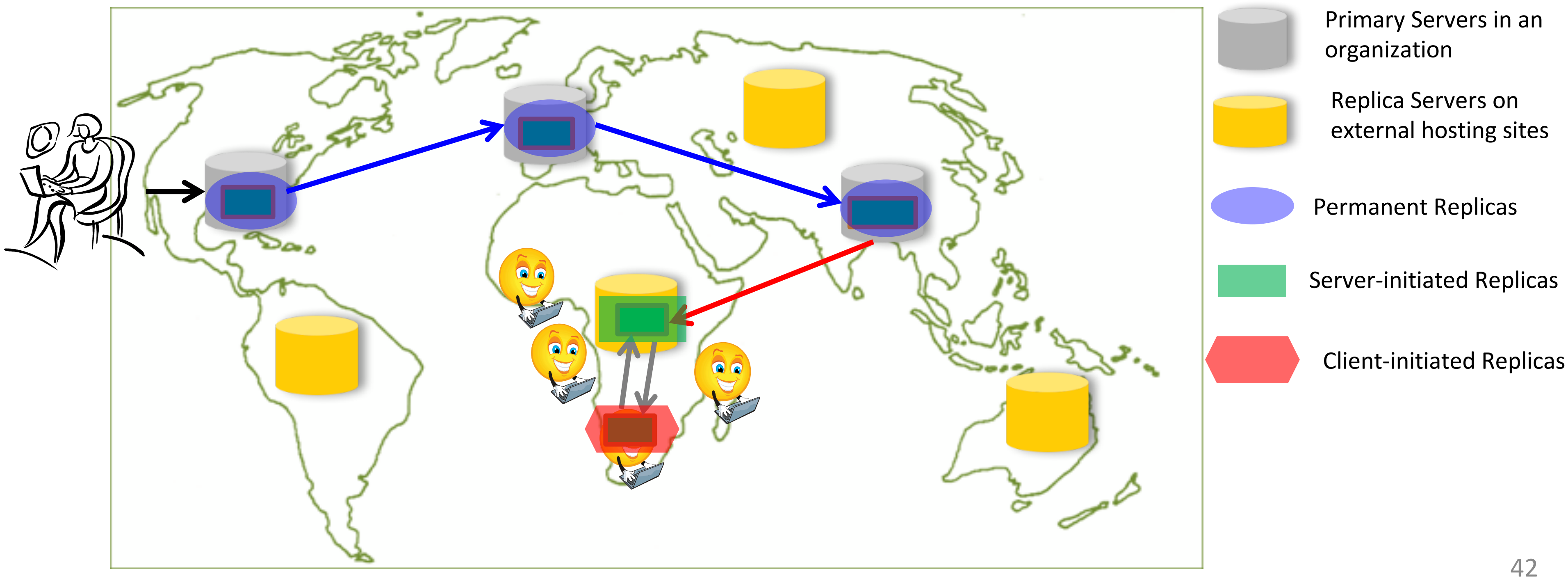
Today…

- Consistency and Replication II
  - Client-Centric Consistency Models
  - Replica Management
    - Replica Server Placement
    - Content Replication and Placement
  - Case Study: Content Delivery Networks (CDNs)

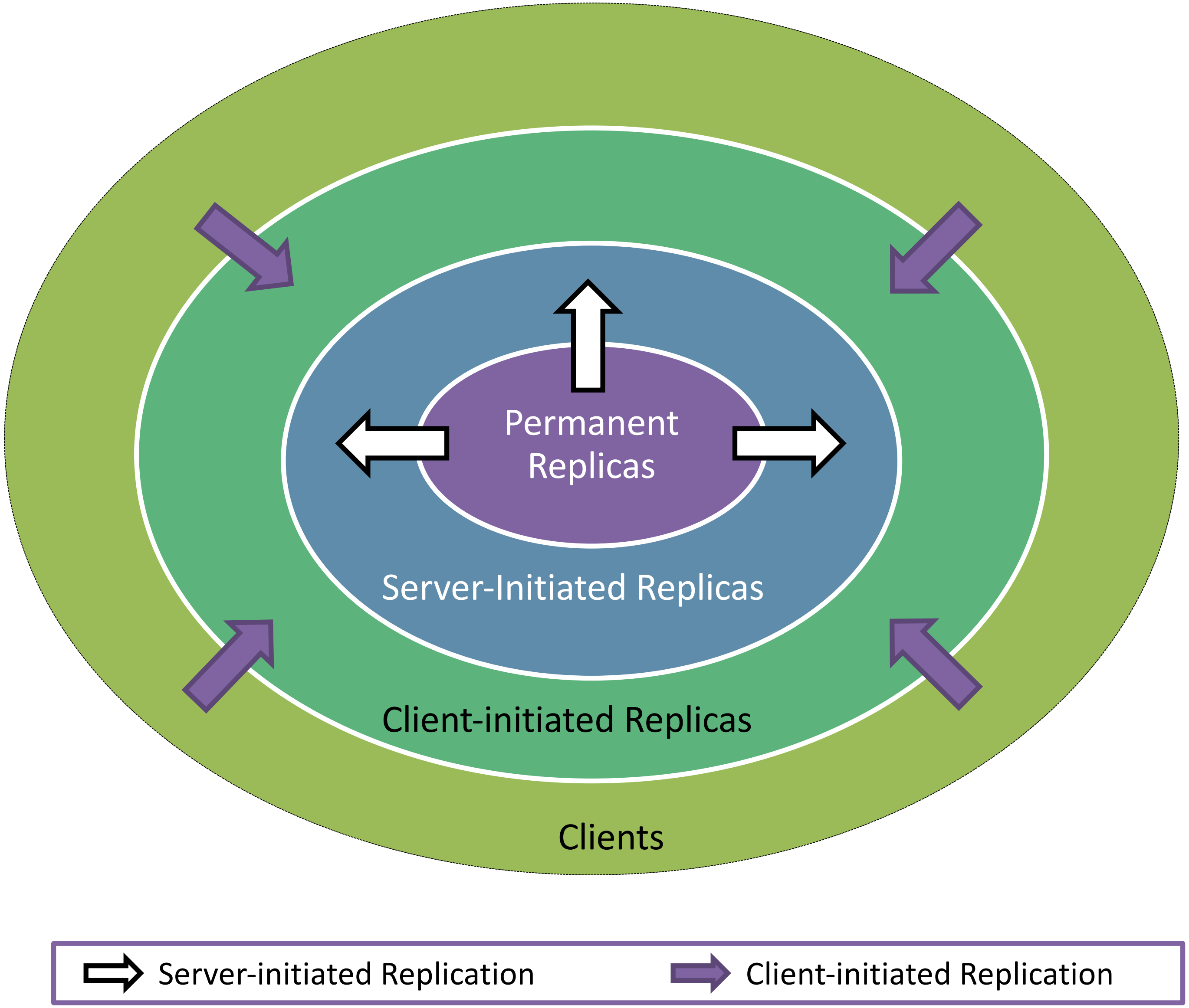
Content Replication and Placement

- In addition to the server placement, it is important:
  - how, when and by whom different data items (contents) are placed on possible replica servers

- Identify how webpage replicas are replicated:



Logical Organization of Replicas



1. Permanent Replicas

- Permanent replicas are the initial set of replicas that constitute a distributed data-store
- Typically, small in number
- There can be two types of permanent replicas:
  - Primary servers
    - One or more servers in an organization
    - Whenever a request arrives, it is forwarded into one of the primary servers
  - Mirror sites
    - Geographically spread, and replicas are generally statically configured
    - Clients pick one of the mirror sites to download the data

2. Server-initiated Replicas

- A third party (*provider*) owns the *secondary replica servers*, and they provide *hosting service*
  - The provider has a collection of servers across the Internet
  - The hosting service dynamically replicates files on different servers
    - Based on the popularity of the file in a region
- The permanent server chooses to host the data item on different *secondary replica servers*
- The scheme is efficient when updates are rare
- Examples of Server-initiated Replicas
  - Replicas in Content Delivery Networks (CDNs)

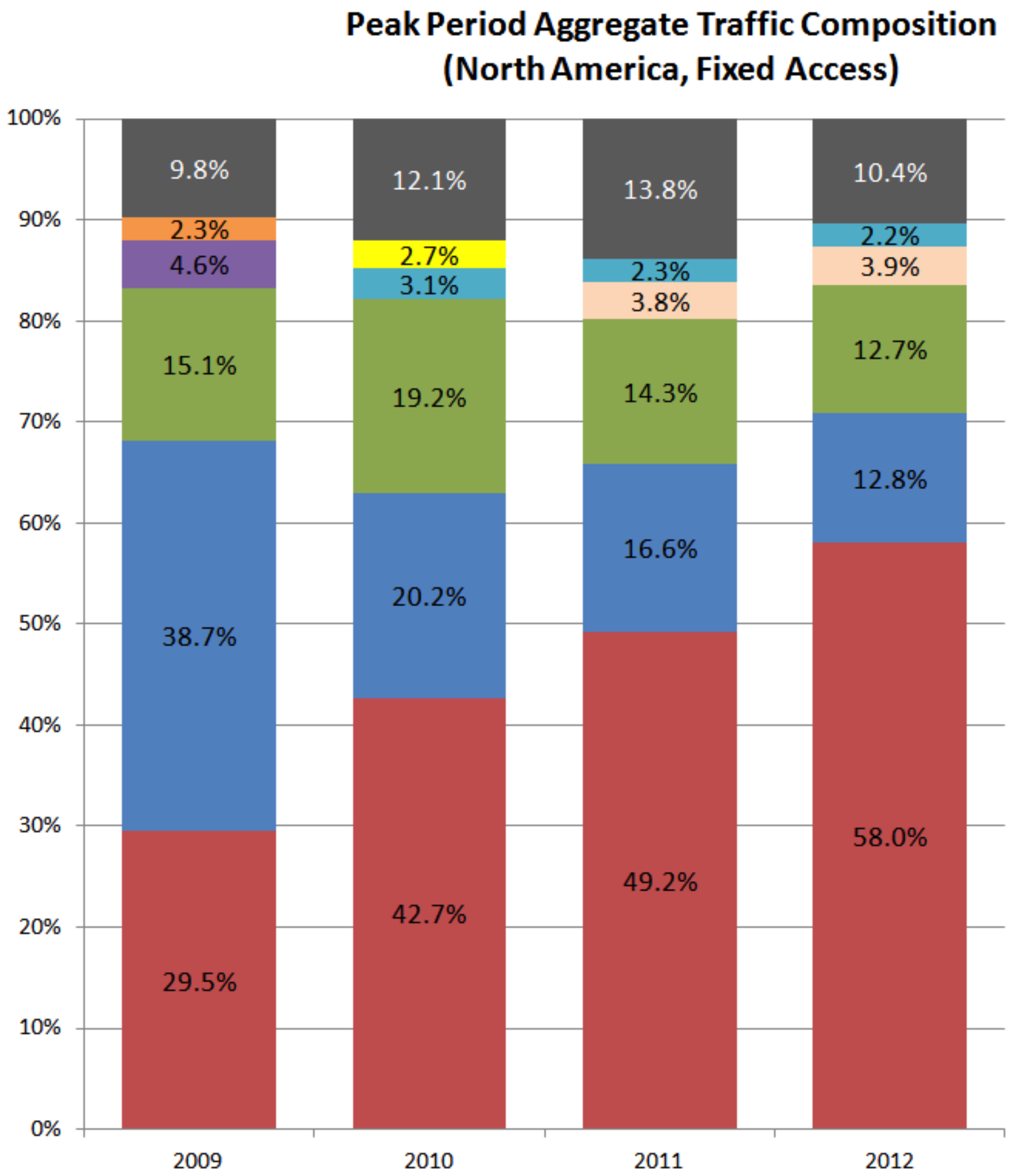
3. Client-initiated Replicas

- Client-initiated replicas are known as *client caches*
- Client caches are used only to reduce the access latency of data
  - e.g., Browser caching a web-page locally
- Typically, managing a cache is entirely the responsibility of a client
  - Occasionally, data-store may inform client when the replica has become stale

Today…

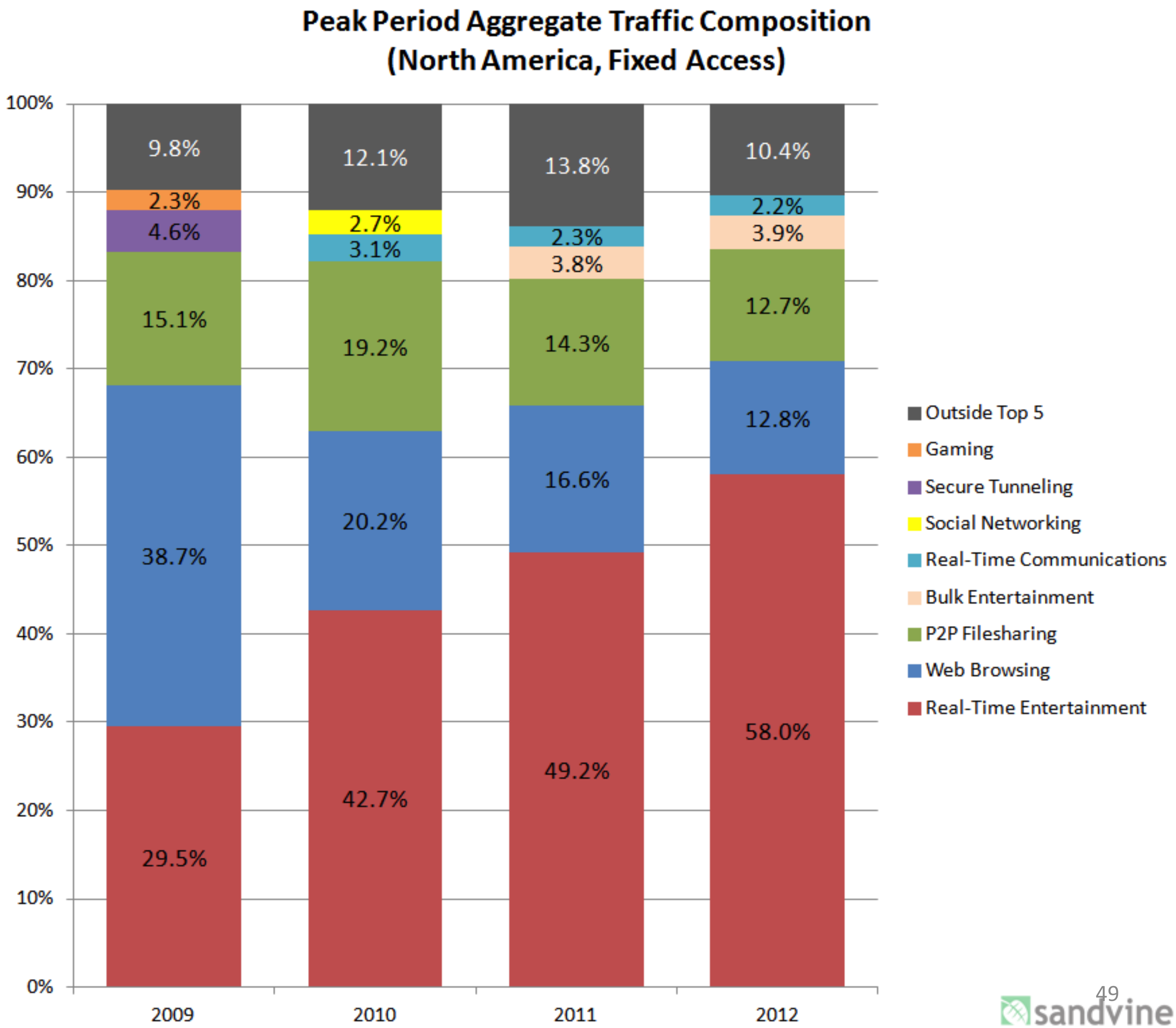
- Consistency and Replication II
  - Client-Centric Consistency Models
  - Replica Management
    - Replica Server Placement
    - Content Replication and Placement
  - Case Study: Content Delivery Networks (CDNs)

Internet Traffic Today



Guess!

Internet Traffic Today

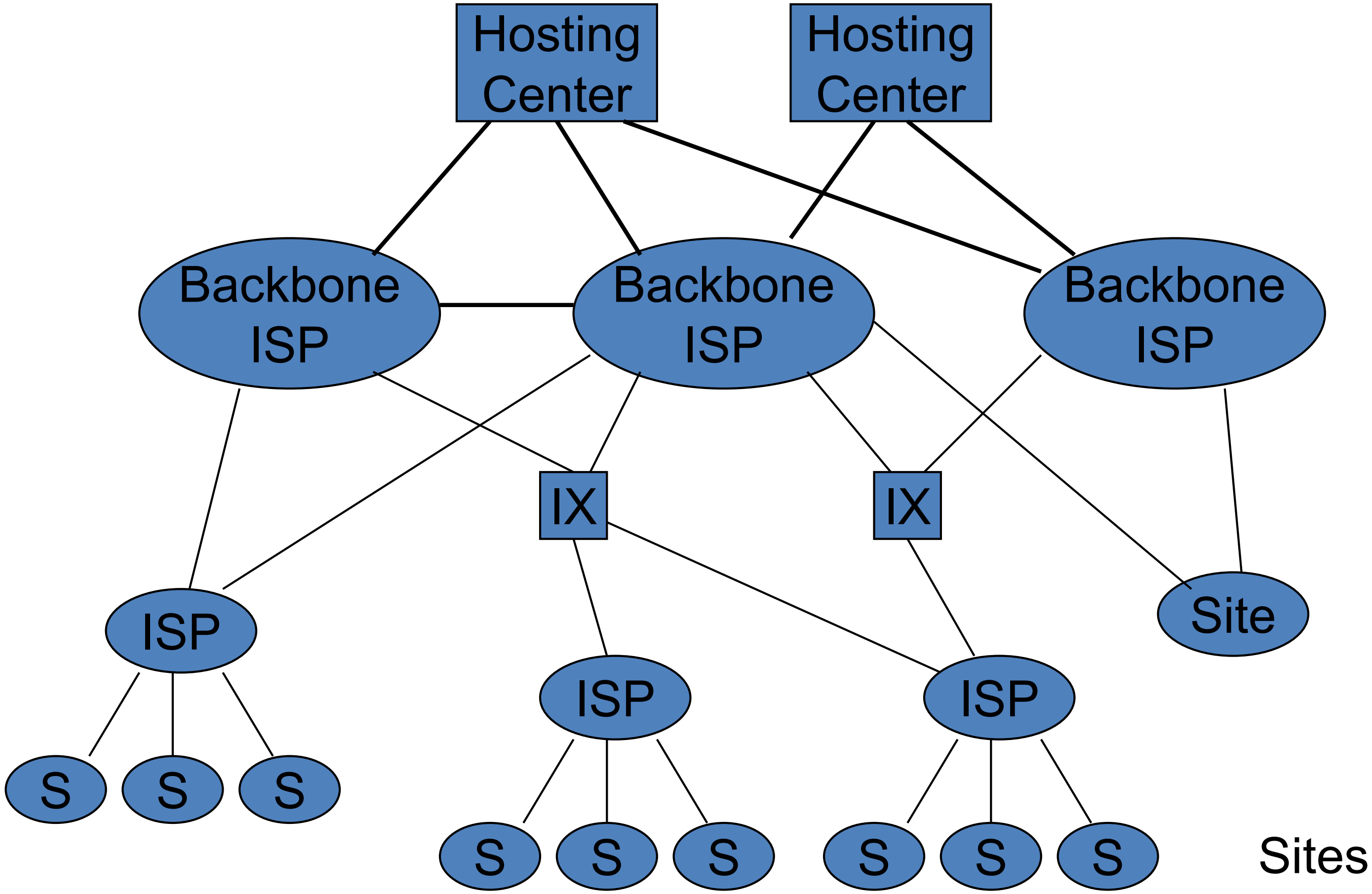


sandvine

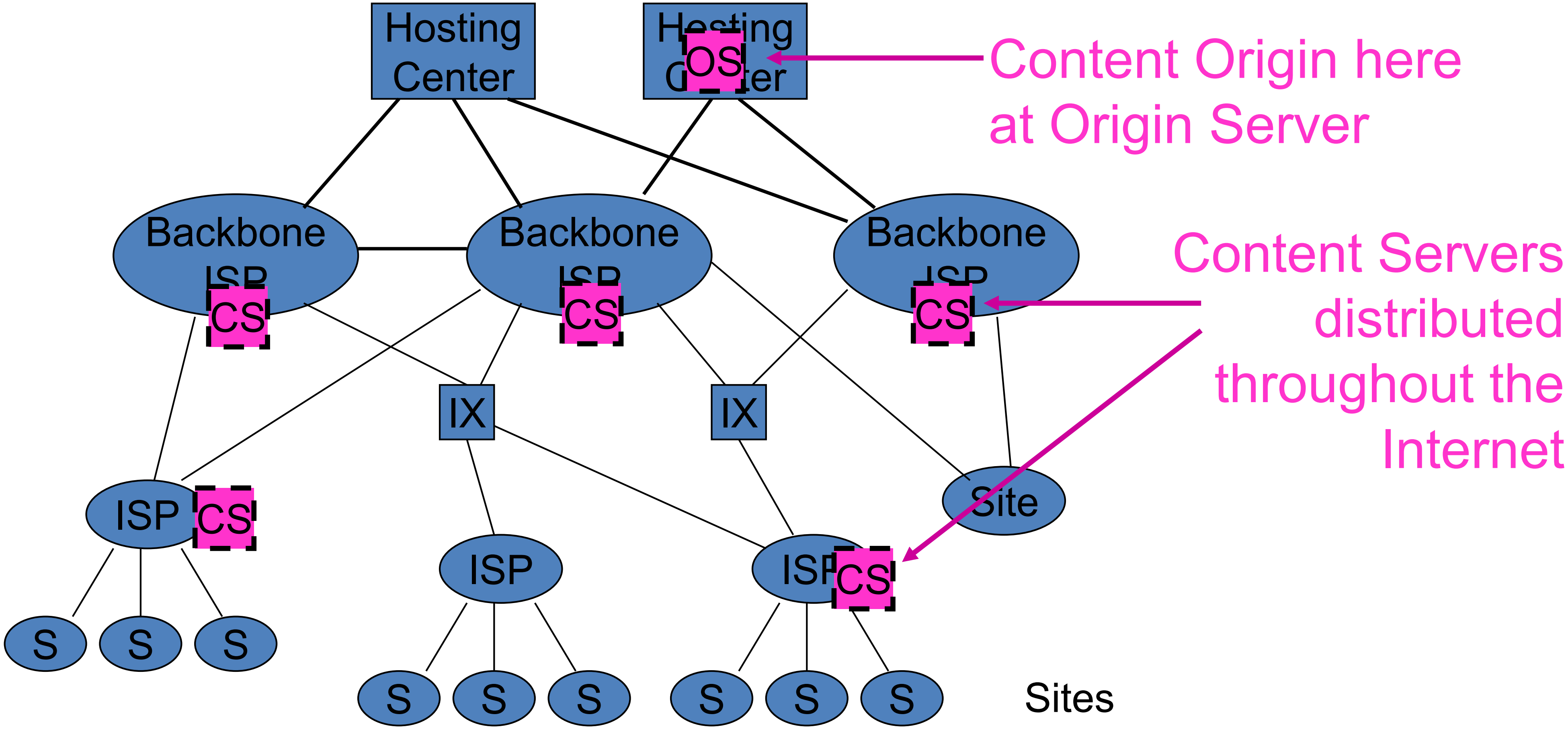
When Content is Popular

- Many downloads
  - Popular items: Web sites, videos etc.
  - Popular services: Facebook, Amazon, …
- How to make them scale?
  - Add replication
  - > Content Delivery Networks

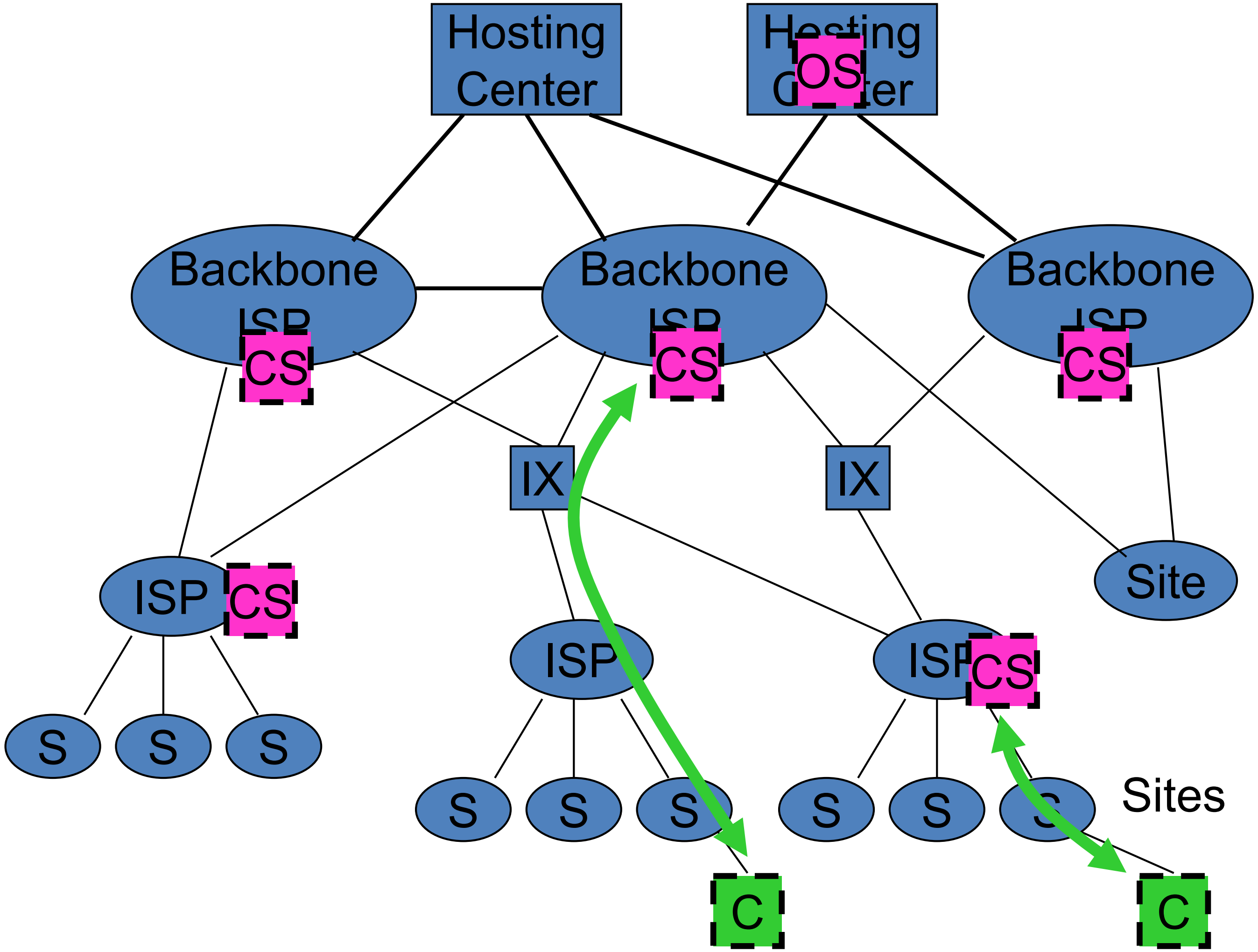
Content Routing Principle
(a.k.a. Content Distribution Network)



Content Routing Principle
(a.k.a. Content Distribution Network)

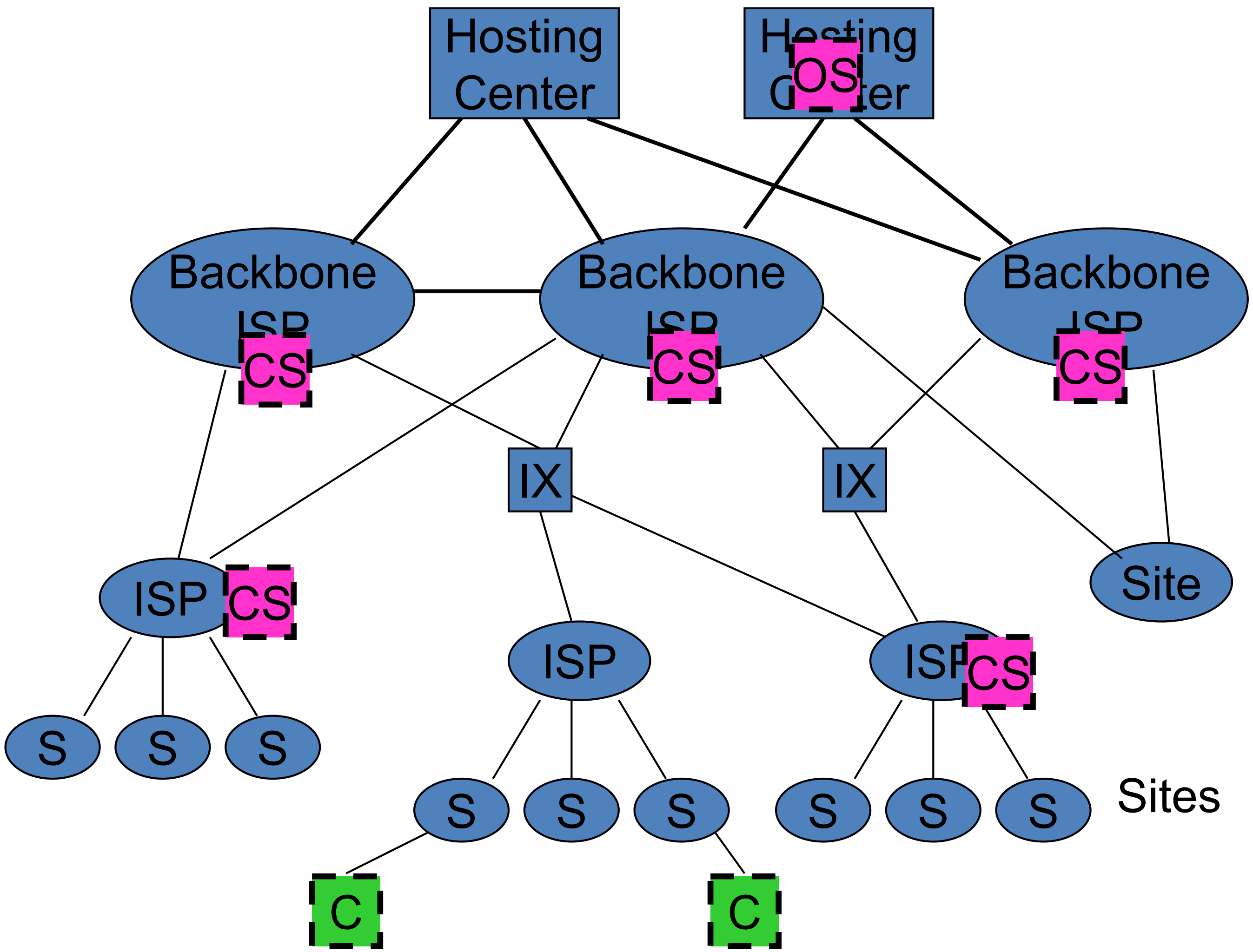


Content Routing Principle
(a.k.a. Content Distribution Network)

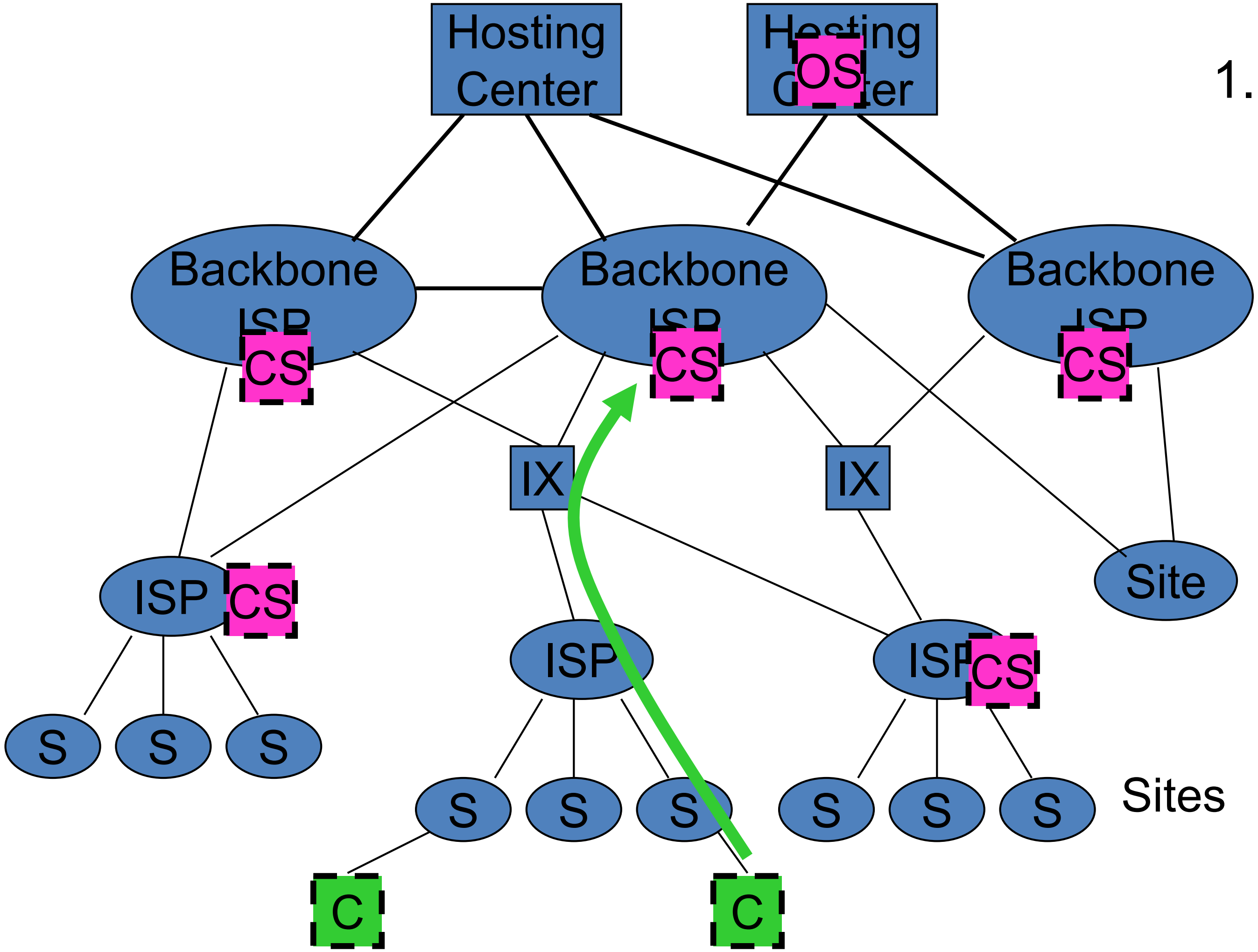


Content is served from content servers nearer to the client

Two basic types of CDN: cached and pushed

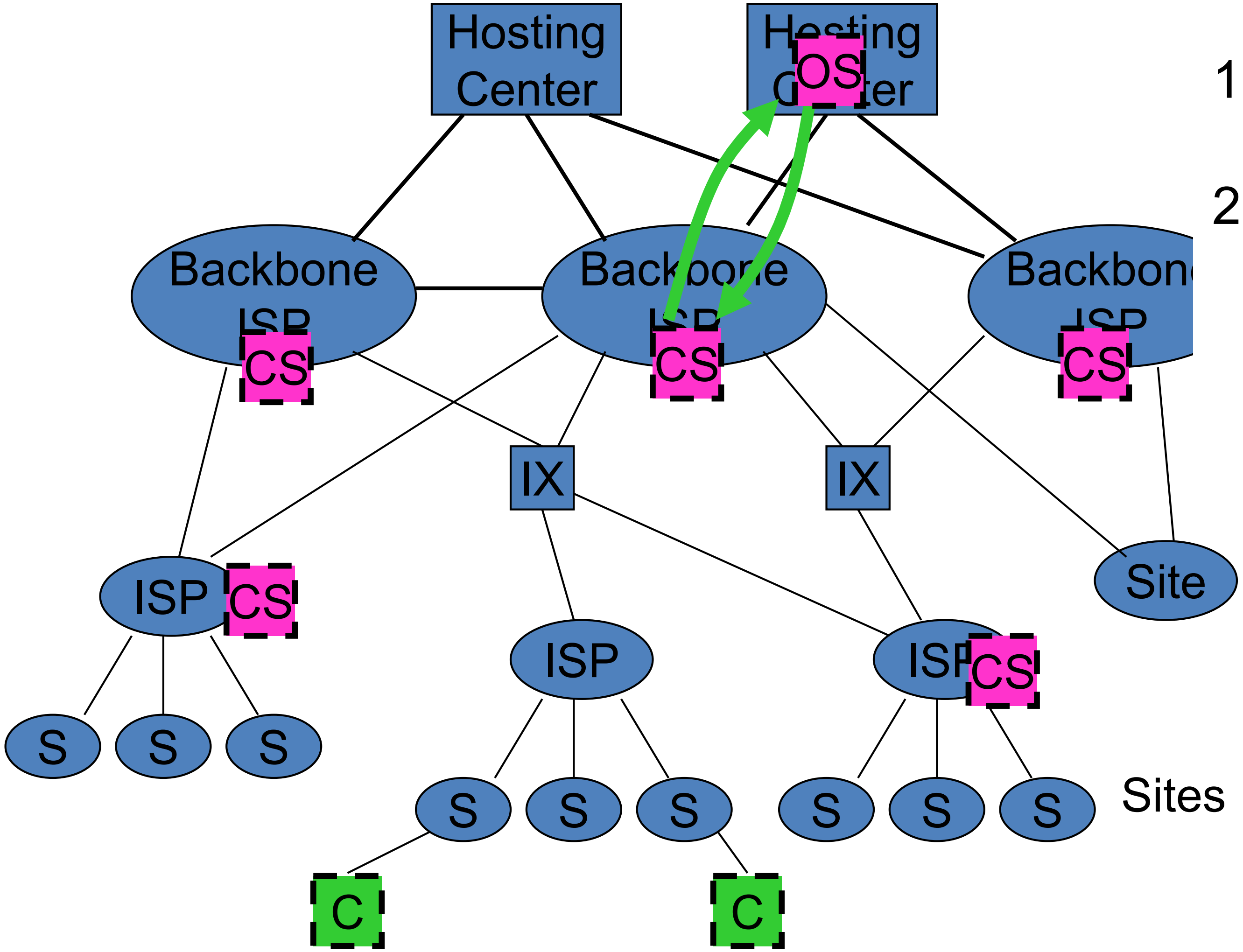


Cached CDN



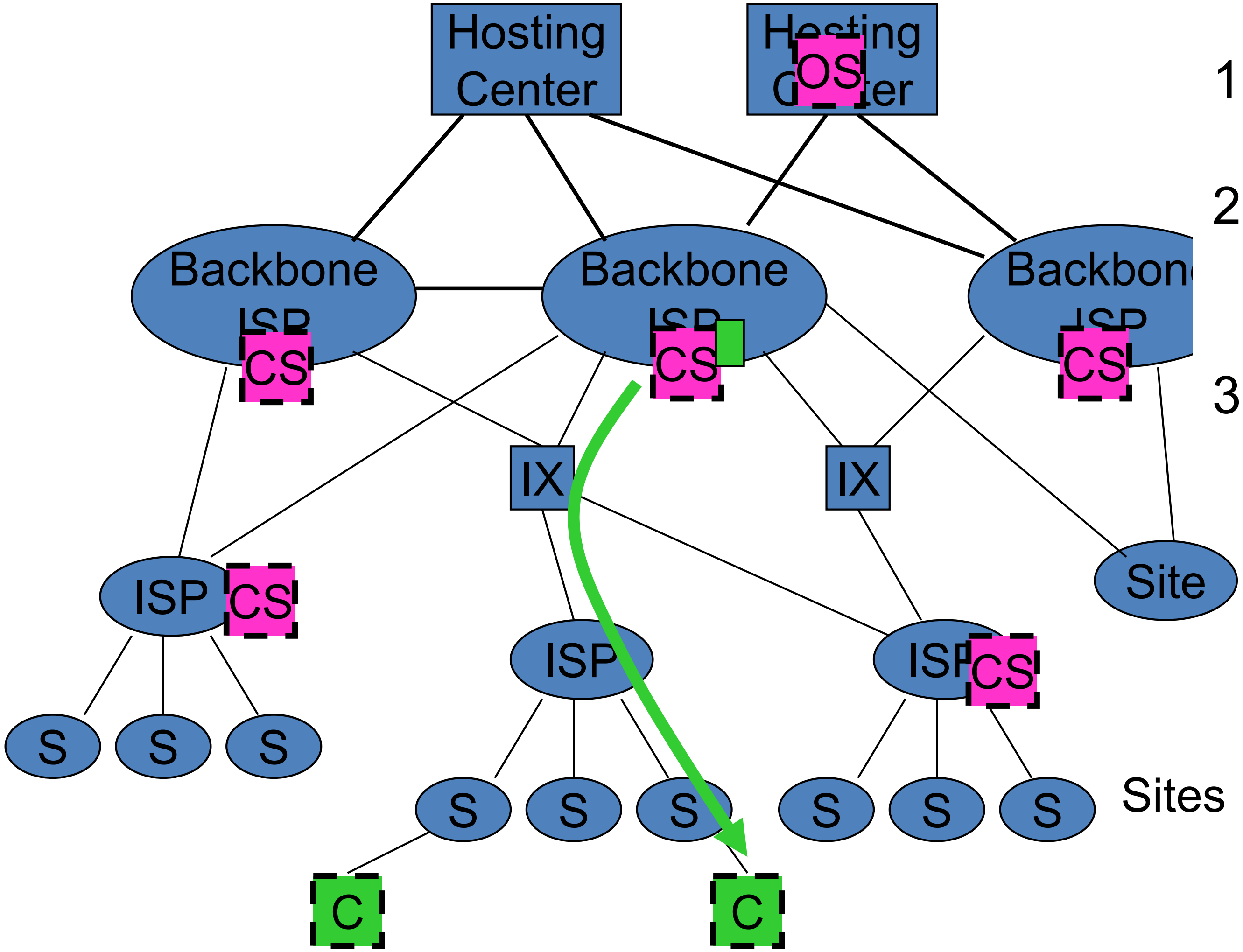
1. Client requests content.

Cached CDN



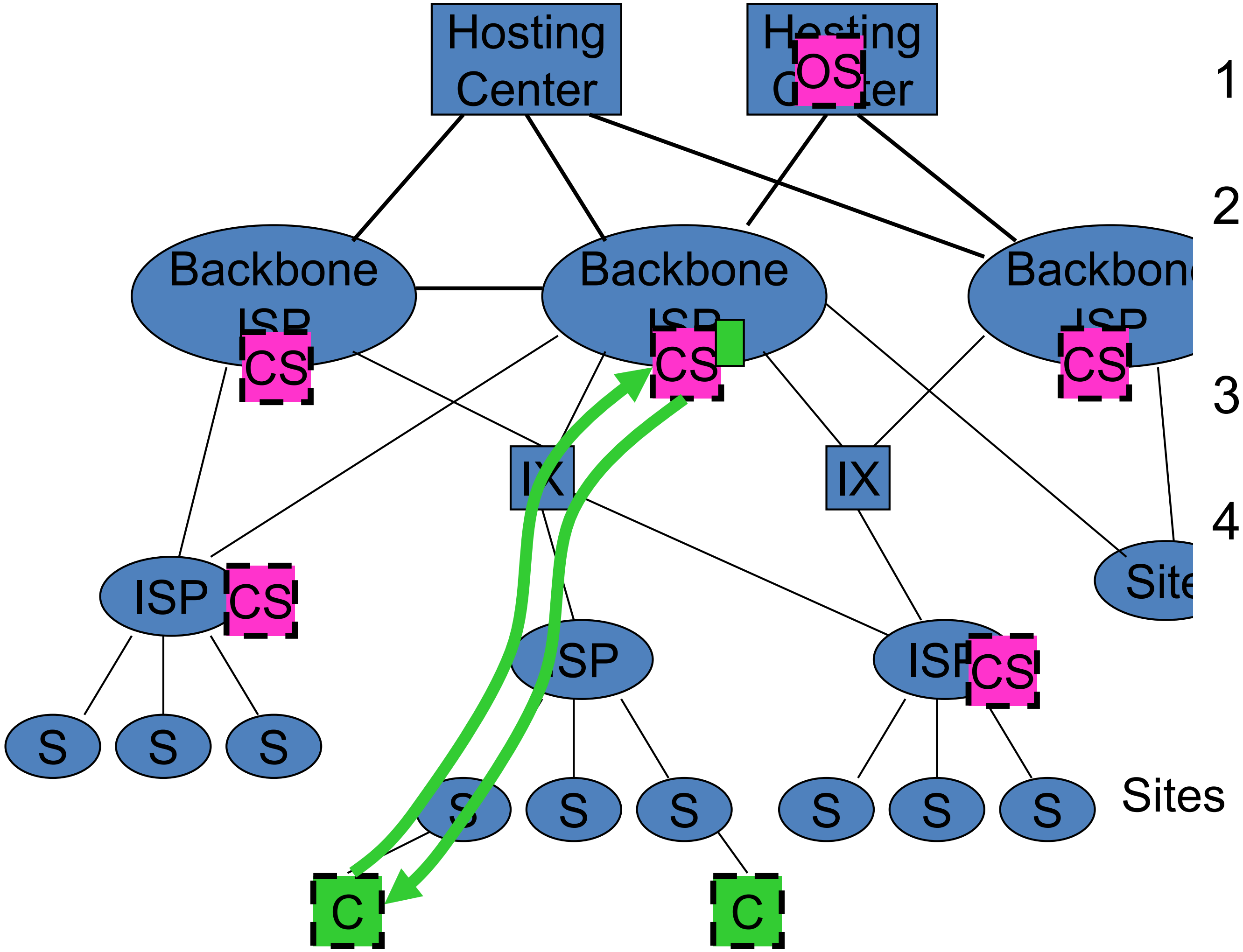
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.

Cached CDN



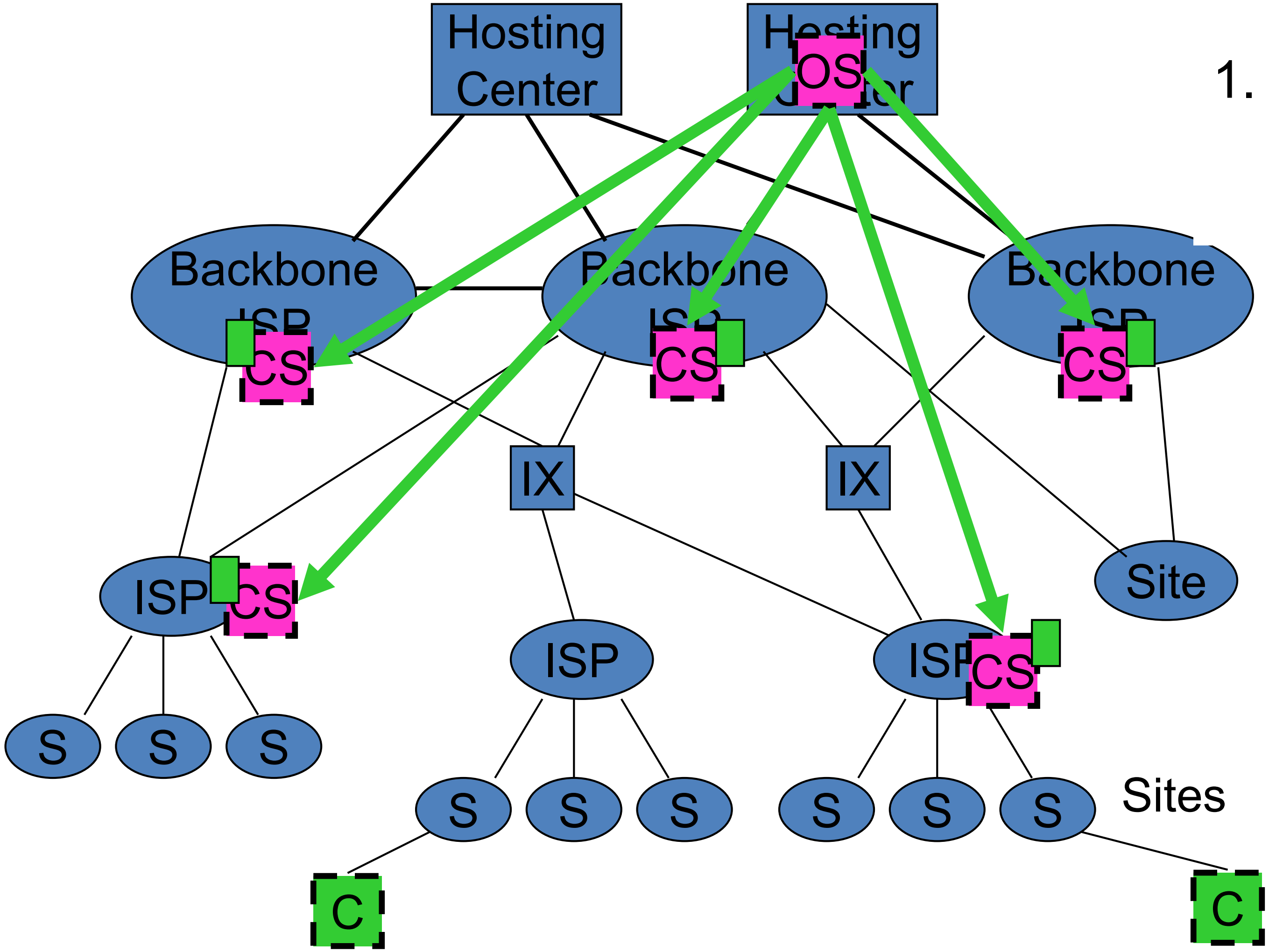
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.
3. CS caches content, delivers to client.

Cached CDN



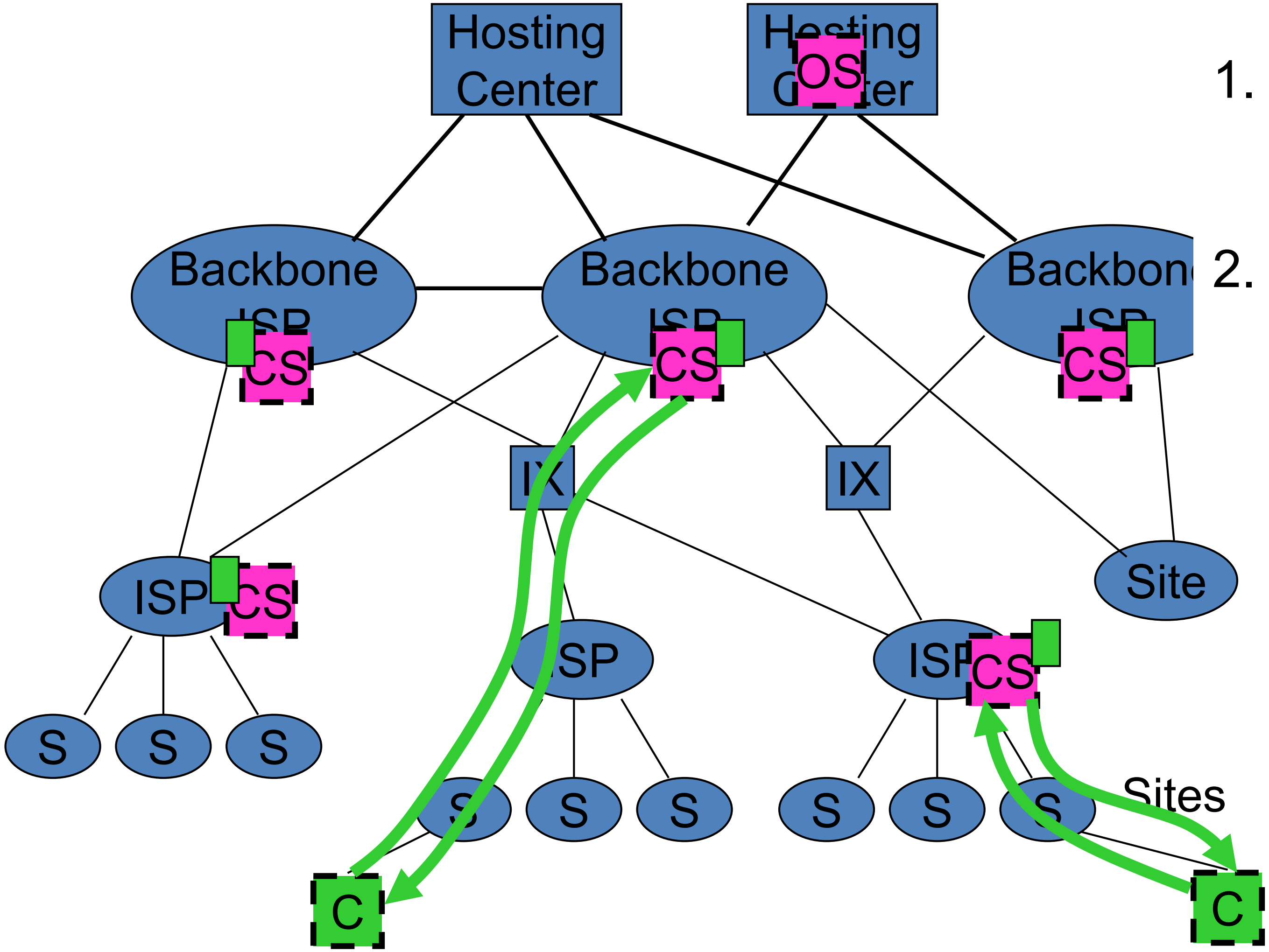
1. Client requests content.
2. CS checks cache, if miss gets content from origin server.
3. CS caches content, delivers to client.
4. Delivers content out of cache on subsequent requests.

Pushed CDN



1. Origin Server pushes content out to all CSs.

Pushed CDN



1. Origin Server pushes content out to all CSs.
2. Request served from CSs.

CDN Details

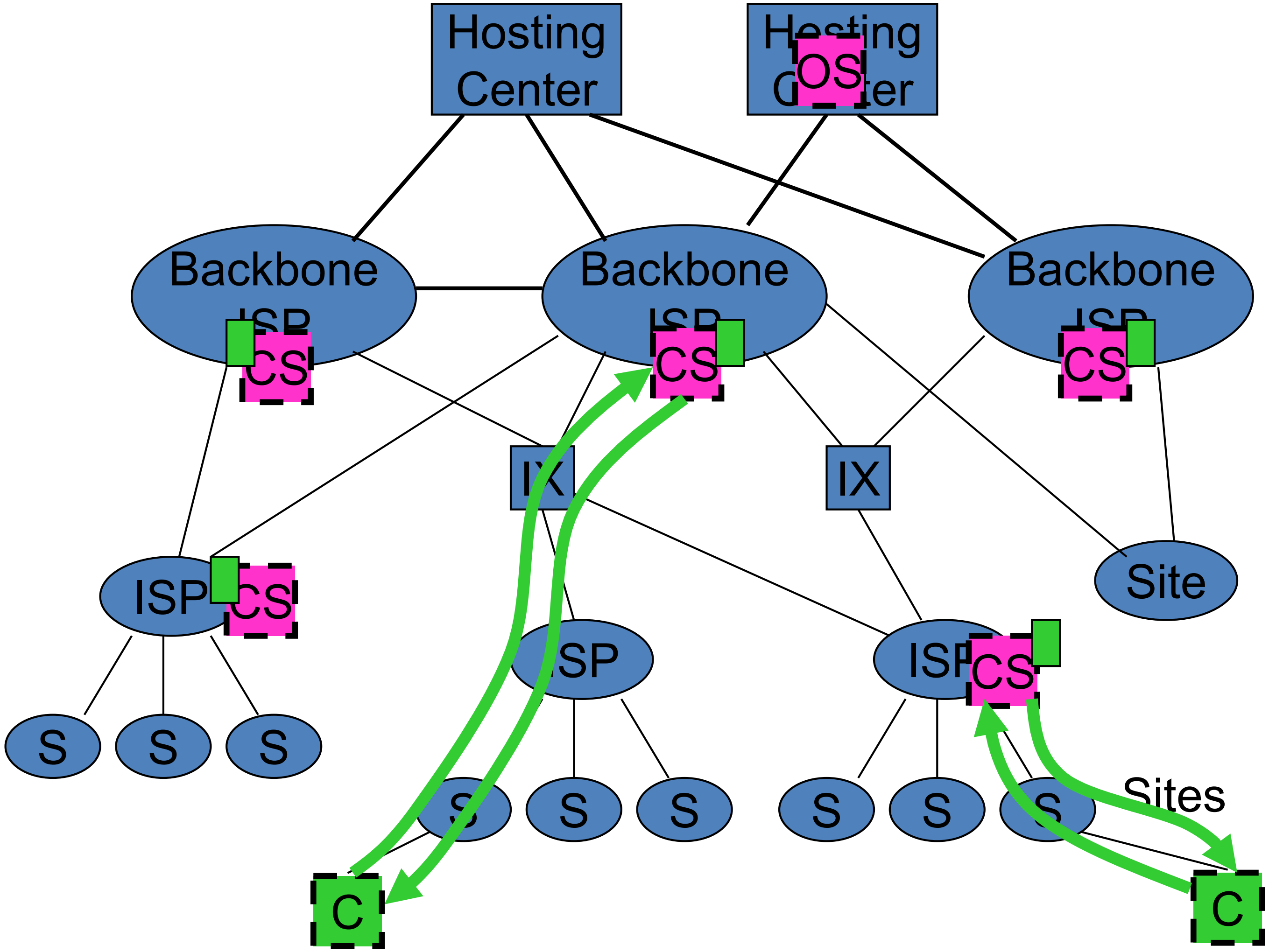
- How to ensure that nodes retrieve data from nearest (or best) replica?
  - http-redirect on login to a service
  - Change DNS replies
    - Based on locality: www.my-service.com
      - Resolves to different IP address based on client location / ISP
  - IP routing
    - Route to different cache based on client location / ISP

Dynamics

COURTESY: AKAMAI

- Watch the dynamics
  - Generate replicas on demand
    - Push to new replicas
  - Try to predict popularity
    - Time of day
    - Previous experience
    - …

CDN Benefits



As content provider:
How do I convince an ISP to buffer my content?

Buffering reduces cost for ISP: less outgoing traffic: to other ISPs

Win-win situation for both content provider and ISPs

CDN Benefits

- Content served closer to client
  - Less latency, better performance
- Load spread over multiple distributed CSs
  - More robust (to ISP failure as well as other failures)
  - Handle flashes better (load spread over ISPs)
  - *But well-connected, replicated Hosting Centers can do this too*

- *Often: Win-win for both ISP and content provider*

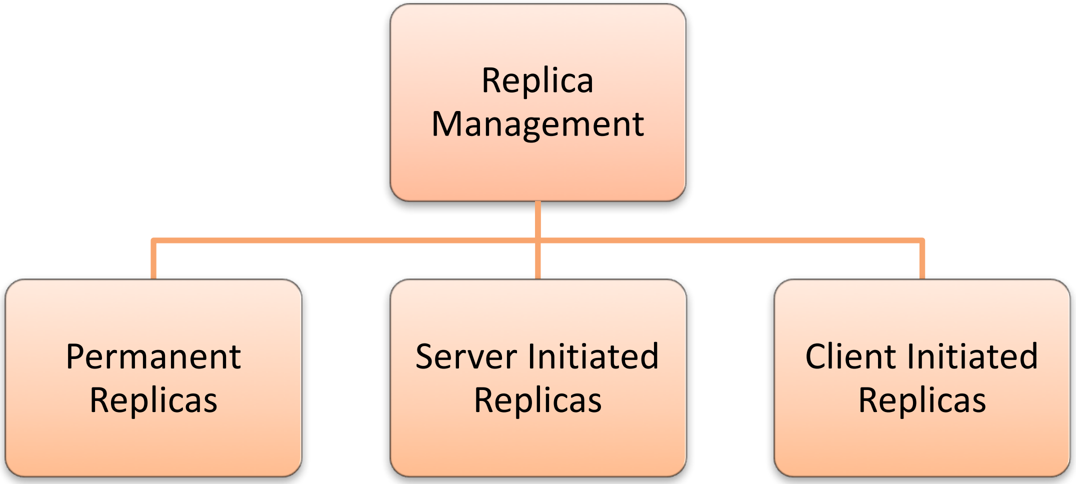
CDN Costs and Limitations

- More and more content is dynamic
  - “Classic” CDNs: caching driven
    - Cached CDNs can’t deal with dynamic/personalized content
    - Classic CDNs limited to images, videos, files, …
  - “Modern” CDNs
    - Service provider runs own machine at ISP
    - Can also serve dynamic content, even encrypted

- Managing content distribution is non-trivial
  - Tension between content lifetimes and cache performance
  - Dynamic cache invalidation
  - Keeping pushed content synchronized and current
  - Huge business has evolved around CDNs

Summary of Replica Management

- Replica management deals with placement of servers and content for improving performance and fault-tolerance



Till now, we know:

- how to place replica servers and content
- the required consistency models for applications

What else do we need to provide consistency in a distributed system?

Next Class

- Consistency Protocols
  - We study “how” consistency is ensured in distributed systems

Questions?

In part, inspired from / based on slides from

– Vinay Kolar
– Many others