INF3190 – Data Communication
Application Layer

Carsten Griwodz

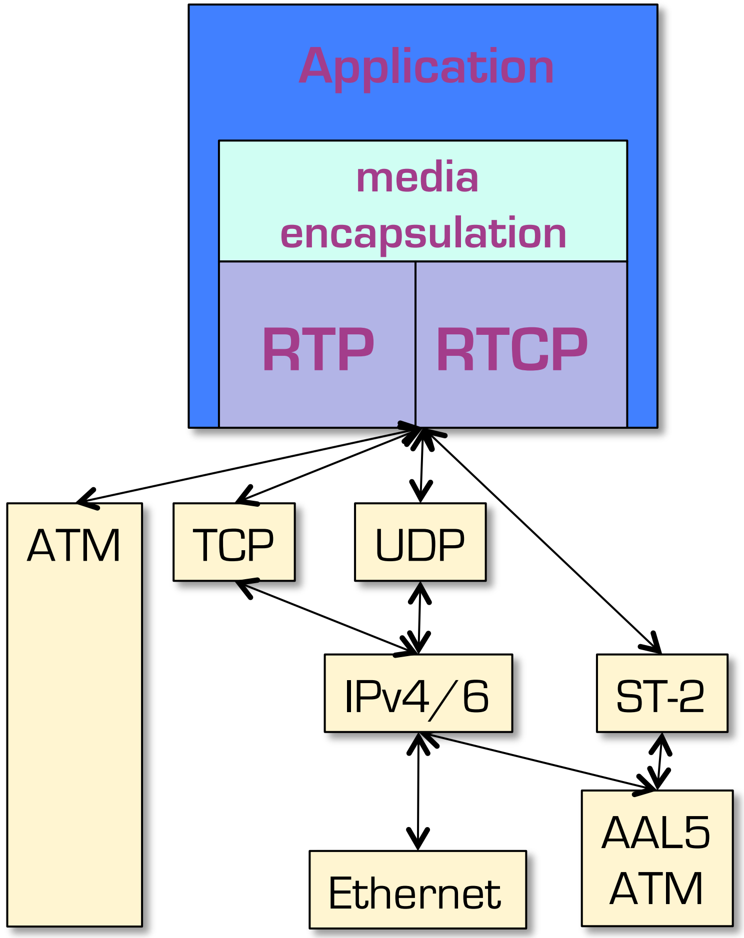
Email: griff@ifi.uio.no

Real-time Transport Protocol (RTP)

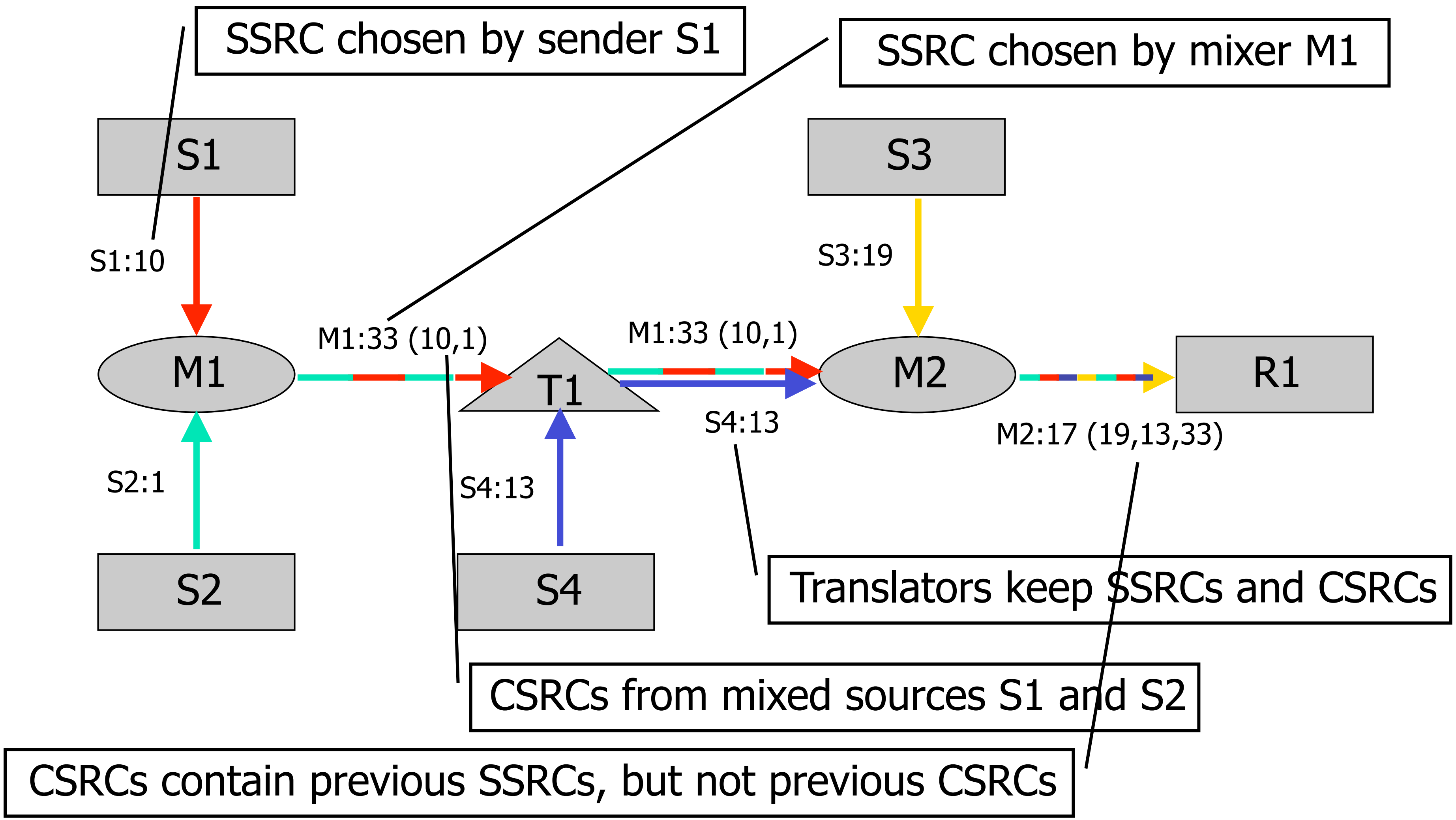
- Real-time Transport Protocol (RTP)
  - **NOT** real-time
  - **NOT** a transport protocol

- Two Components: RTP & RTCP
- Provides end-to-end transport functions
- No premise on underlying resources
- Integrated with applications

- RTP follows principles of
  - Application Level Framing and
  - Integrated Layer Processing



RTP Identifiers



Signalling Protocols: RTSP & SIP

Signaling Protocols

Applications differ

- Media delivery controlled by sender or receiver
- Sender and receiver "meet" before media delivery

Signaling should reflect different needs

- Media-on-demand
  - Receiver controlled delivery of content
  - Explicit session setup
- Internet broadcast
  - Sender announces multicast stream
  - No explicit session setup
- Internet telephony and conferences:
  - Bi-directional data flow, live sources
  - (mostly) explicit session setup, mostly persons at both ends

Media-on-demand and Internet broadcast: Real-Time Streaming Protocol (RTSP)

Internet broadcast and Internet telephony and conferences: Session Initiation Protocol (SIP)

Real-Time Streaming Protocol (RTSP)

Rough synchronization

- Media description in DESCRIBE response
- Timing description in SETUP response
- Fine-grained through RTP sender reports

Aggregate and separate control of streams possible

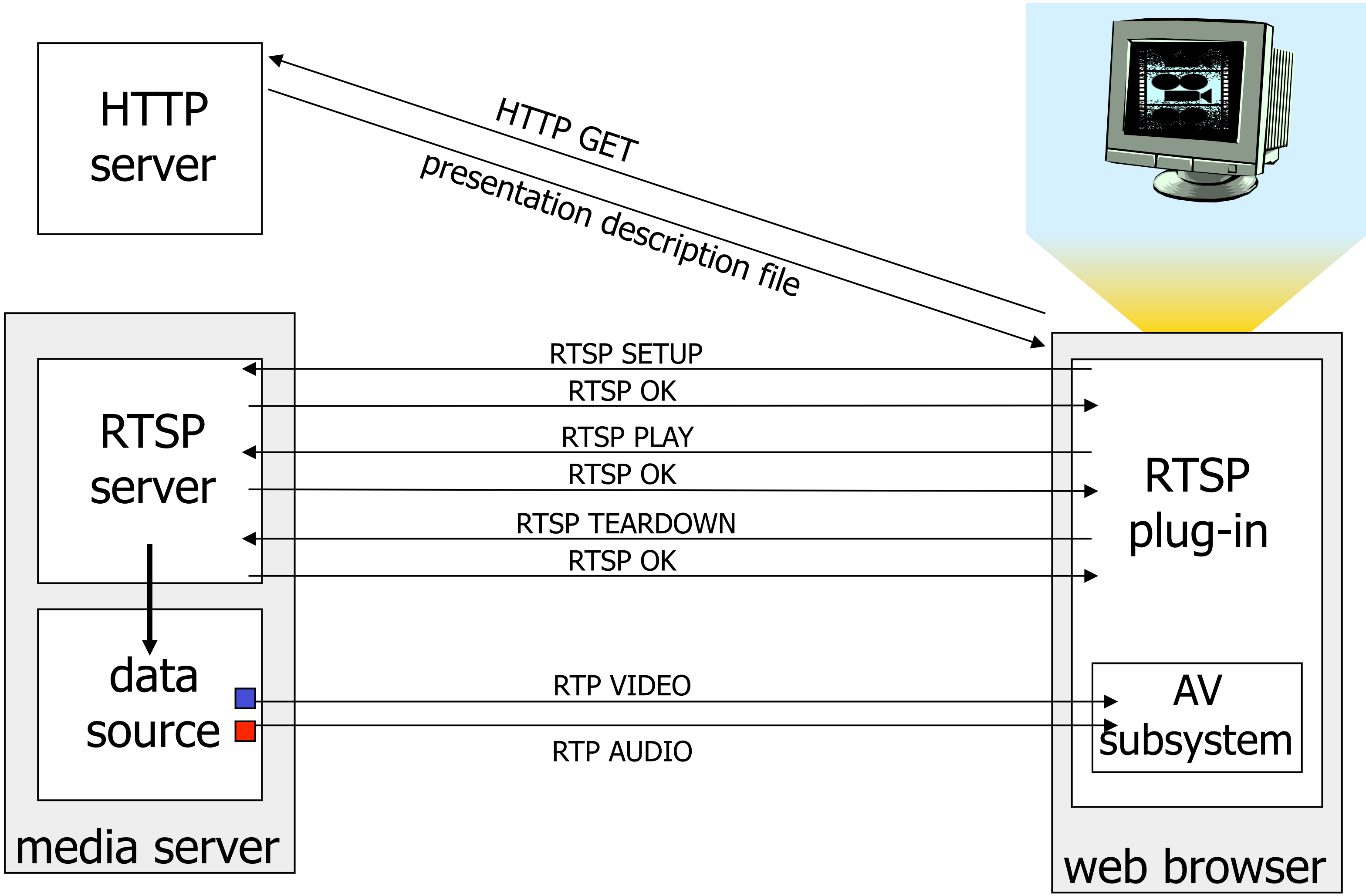
Combine several data (RTP) servers

Load balancing by REDIRECT at connect time

Caching

- Much more difficult than web caching
  - interpret RTSP
  - but cache several RTP flows
- Cache must act as an RTP translator
  - otherwise it cannot guarantee to receive packets

RTSP Integration



Session Initiation Protocol (SIP)

Lightweight generic signaling protocol

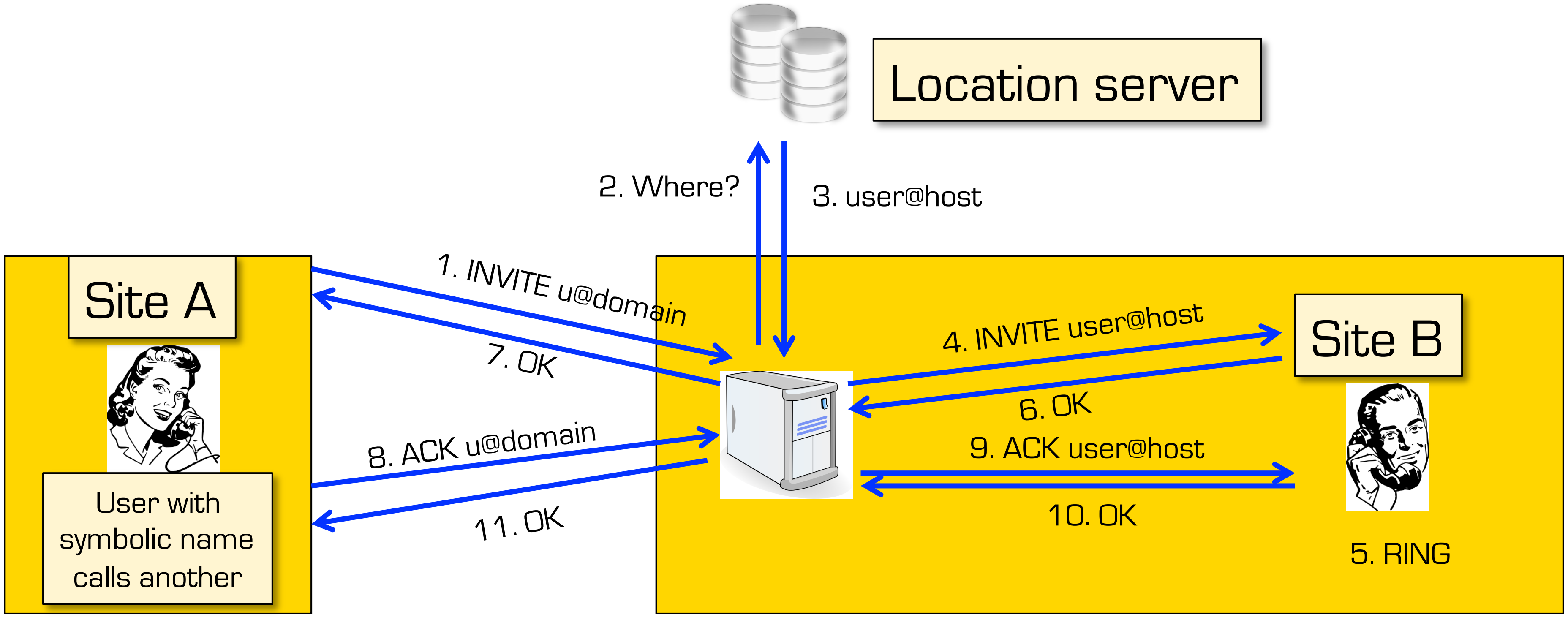
Internet telephony and conferencing

- call: association between number of participants
- signaling association as signaling state at endpoints (no network resources)

Several "services" needed

- Name translation
- User location
- Feature negotiation
- Call control
- Changing features

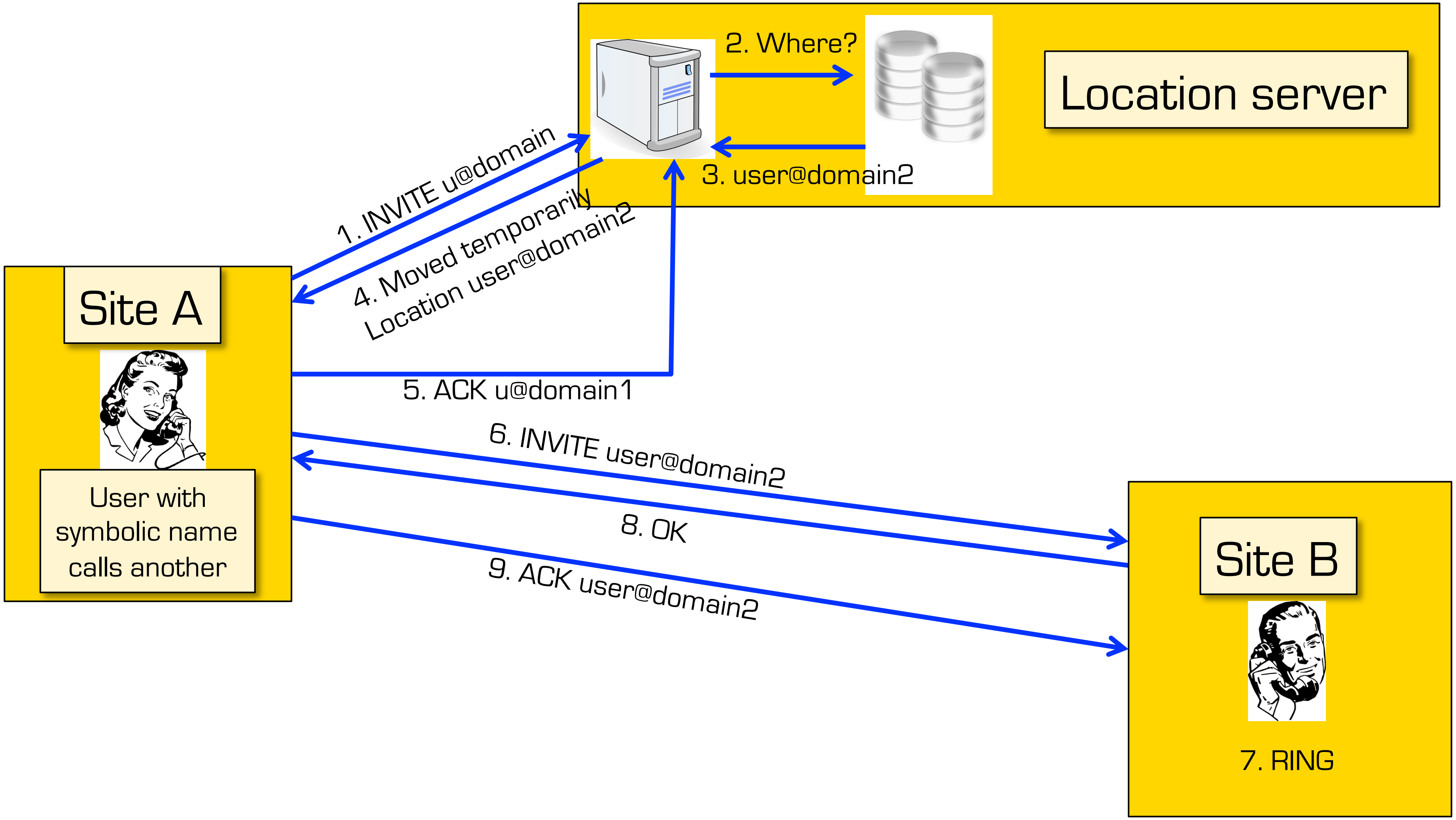
SIP Operation – Proxy Mode



Proxy forwards requests

- possibly in parallel to several hosts
- cannot accept or reject call
- useful to hide location of callee

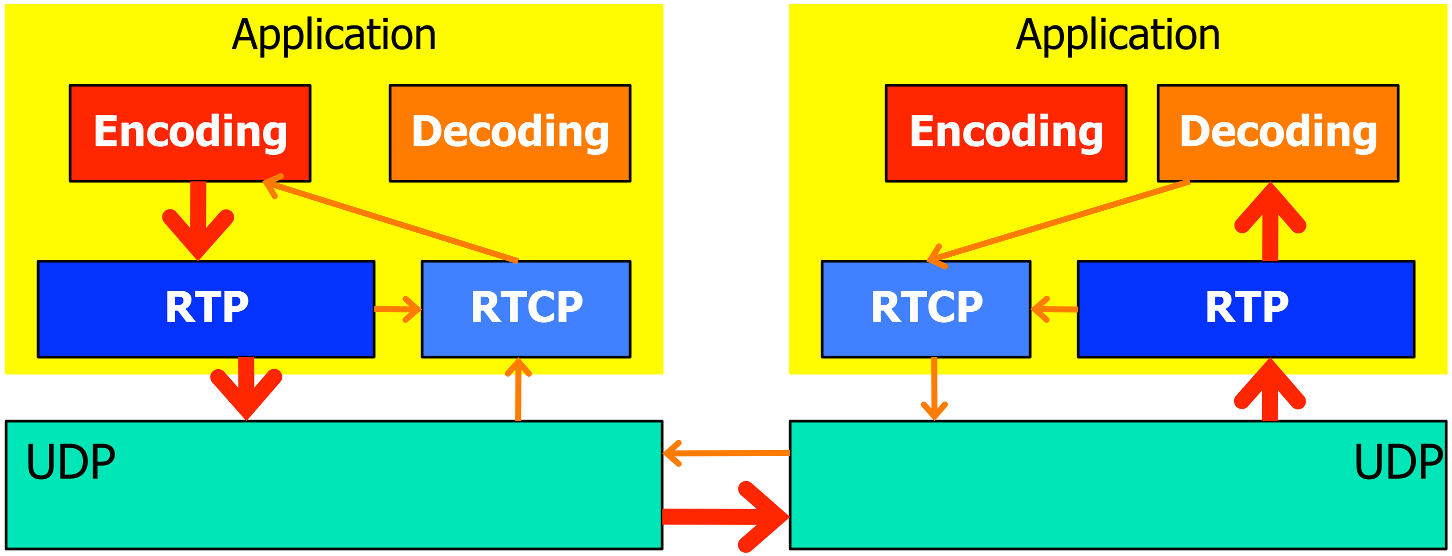
SIP Operation – Redirect Mode



Co-existing with TCP

Adapt audiovisual quality to your bandwidth share

RTP Quality Adaptation



Application level framing idea

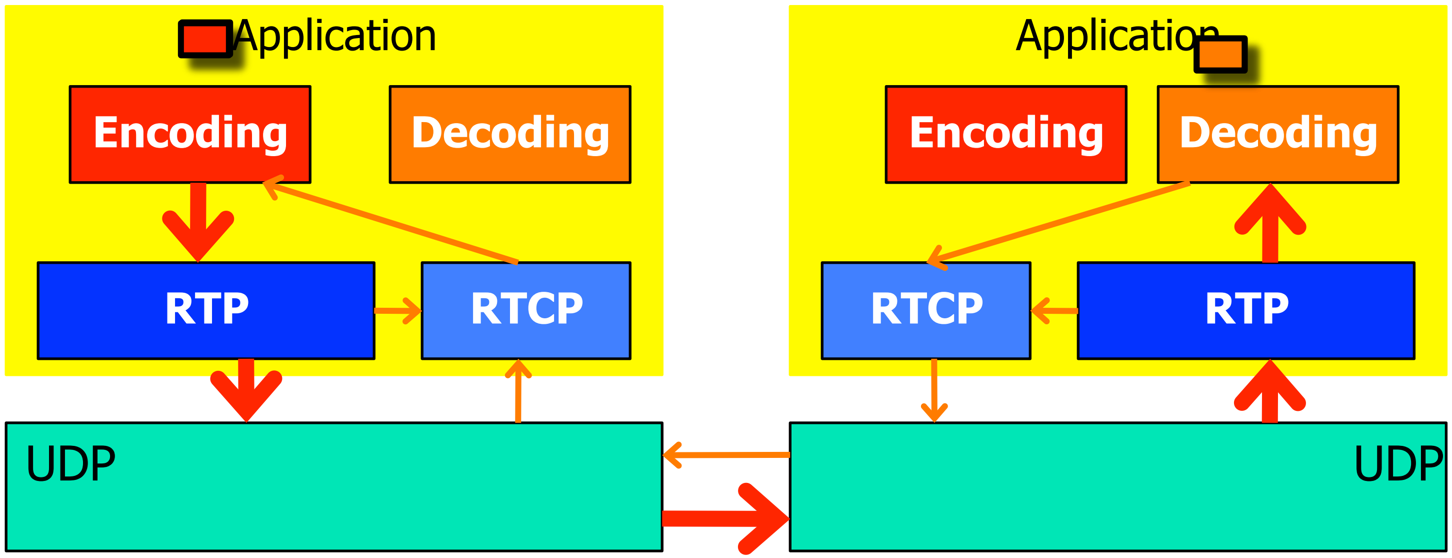
- application knows best how to adapt
- protocol (i.e. RTP) provides information about the network

Application can

- evaluate sender and receiver reports
- modify encoding schemes and parameters
- adapt its transmission rates

Loss-Delay Adjustment Algorithm

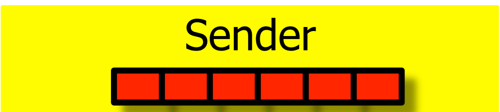
- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically



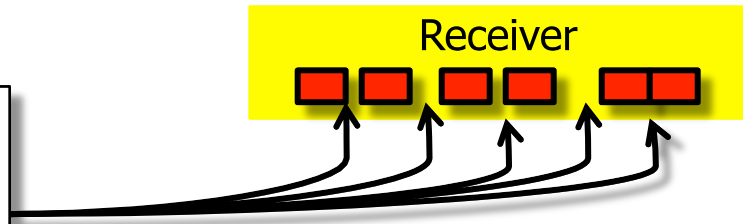
"The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme ",
D. Sisalem, H. Schulzrinne, NOSSDAV 1998

Loss-Delay Adjustment Algorithm

- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically
  - works like TCP's AIMD
    - but RRs are rare
    - can't adapt every time
  - step one: find the bottleneck bandwidth b



  - use packet size and gaps size



$$b = \frac{packetsize}{gapsize}$$

Loss-Delay Adjustment Algorithm

- LDA
  - An algorithm to stream with RTP in a TCP-friendly way
  - use RTCP receiver reports (RR)
    - RTCP sends RR periodically
  - works like TCP's AIMD
    - but RRs are rare
    - can't adapt every time
  - no loss:
    - use "AIR" – additive increase *rate*
    - but never more than 1 packet/RTT
  - loss:
    - RTCP counts losses, l is *fraction* of lost packets
    - *guess* 3 of those losses in one RTT

current rate

$$AIR_{i+1} = AIR_i * \left(1 - \frac{r_t}{b}\right)$$

$$r_{t+1} = r_t + AIR_{i+1}$$

newrate

$$r_{t+1} = r_t * (1 - l * 3)$$

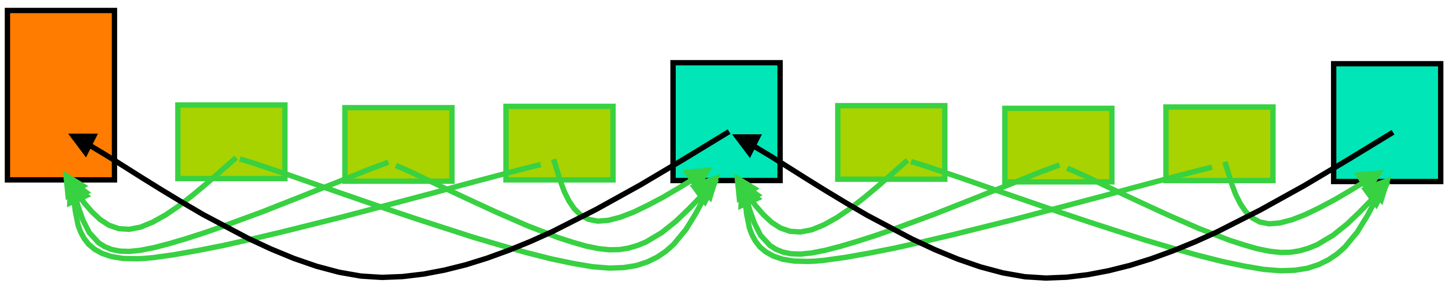
Coding for adaptation

Adapt audiovisual quality to your bandwidth share

Coding for Adaptive Streaming: MPEG-1

Frames can be dropped

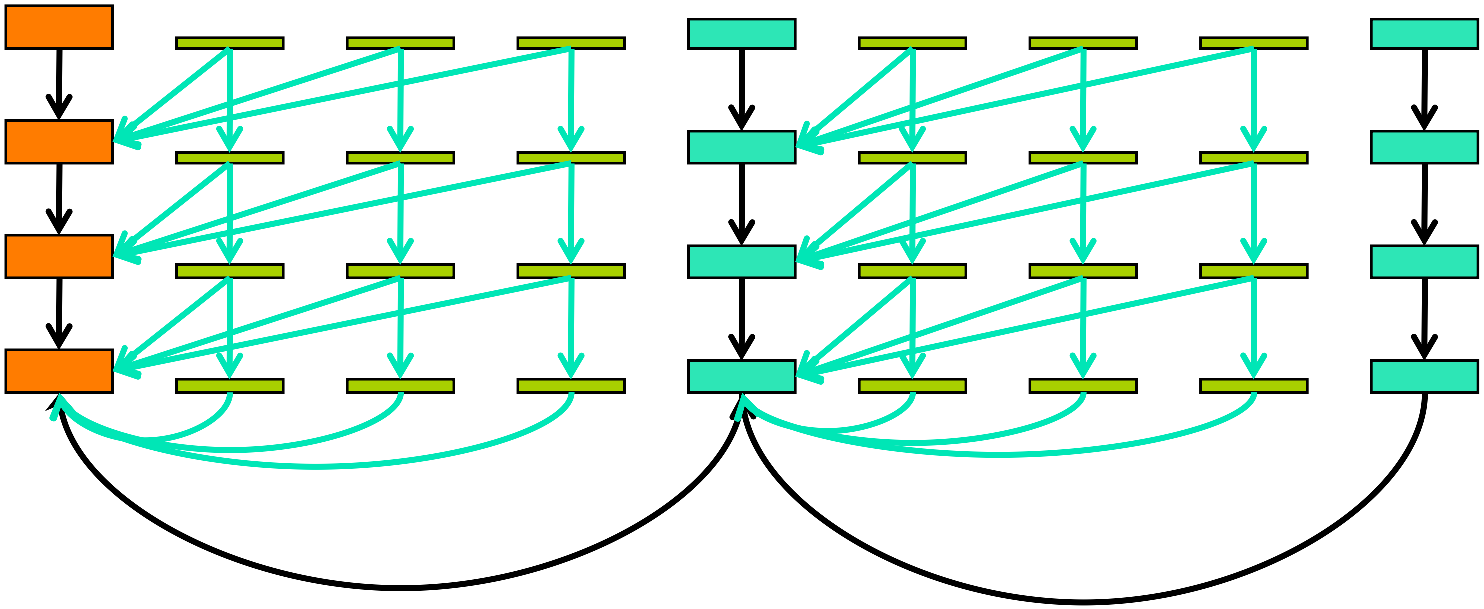
- In a controlled manner
- Frame dropping does not violate dependancies
- Example: B-frame dropping in MPEG-1



Coding ...: H.264 SVC extensions

H.264: most common codec for MPEG-4

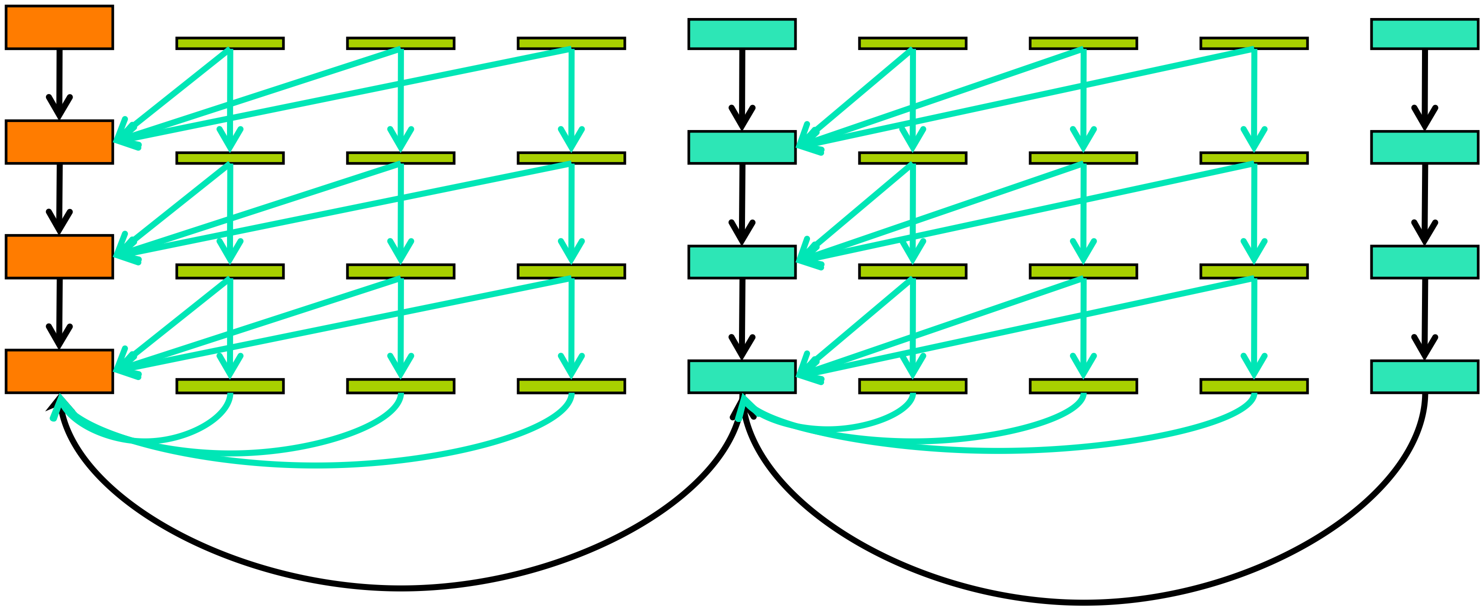
SVC: Scalable Video Codec



Simplified **representation of H.264/SVC**

- the H.264 motion vectors of a frame can point to 16 different frames
- motion vectors in H.264 can cross I-frames
- ...

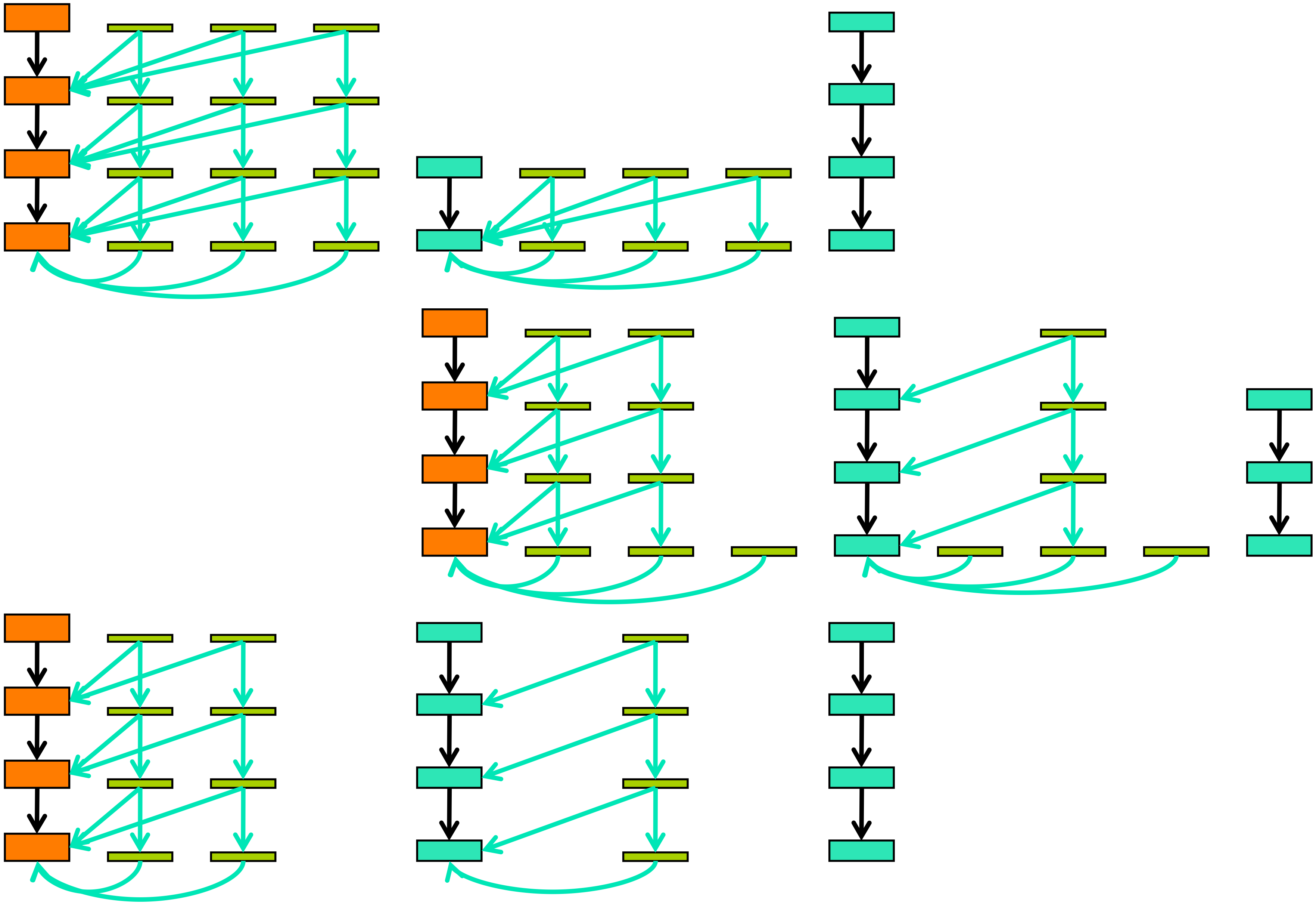
Coding …: H.264 SVC extensions



Simplified representation of H.264/SVC

- the H.264 motion vectors of a frame can point to 16 different frames
- motion vectors in H.264 can cross I-frames
- …

How to change quality?



The *BAD CHOICE: PSNR*

Peak Signal-to-Noise Ratio

- you find it everywhere
- but it is really, really bad

Example from
Prof. Touradj Ebrahimi,
ACM MM'09 keynote

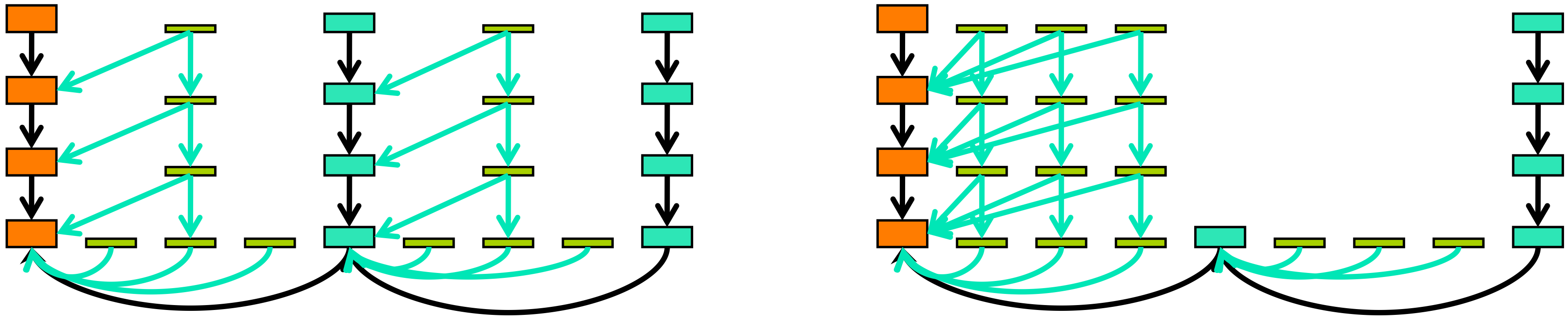
Reference

PSNR = 24.9 dB

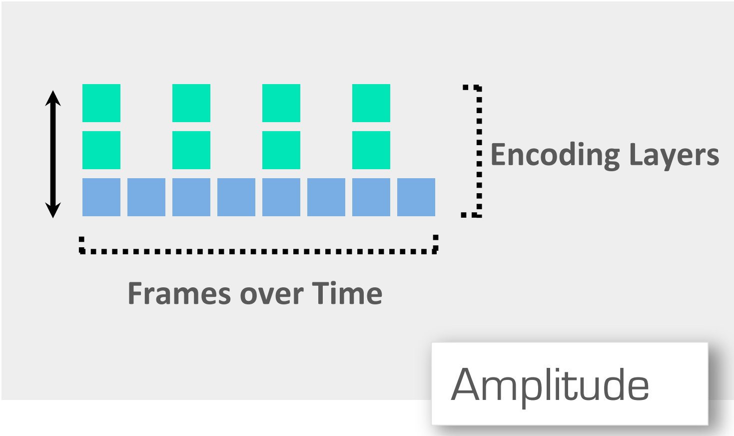
PSNR = 24.9 dB

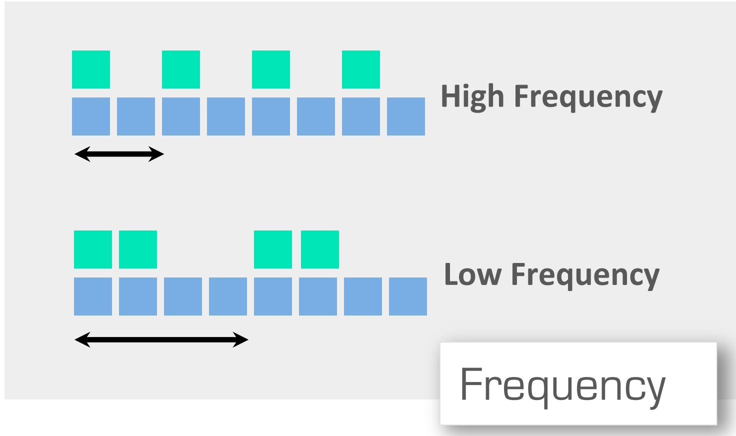
PSNR = 24.9 dB

Coding ...: hierarchical layer coding



Visual Perception of DASH Video

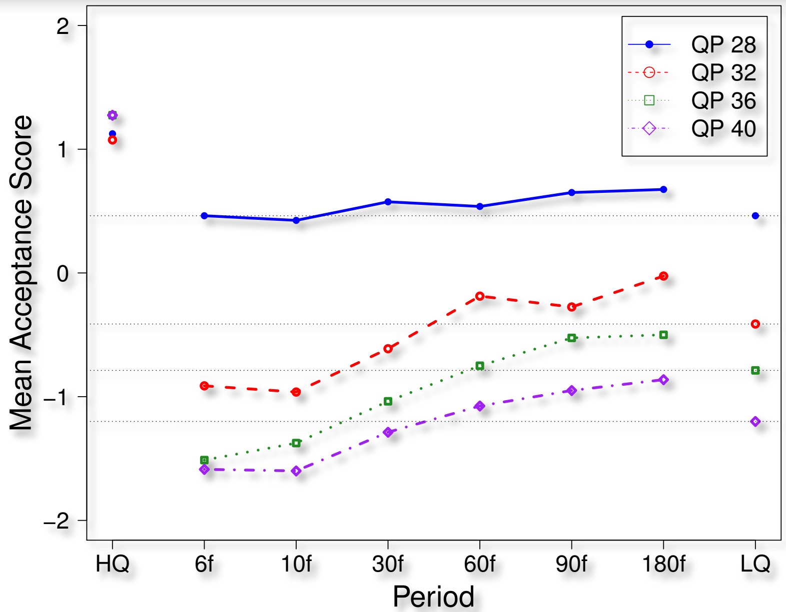




Field study

- mobile devices, free seating, resolution 480x320@30fps, no sunlight, lounge chairs

Blurriness, noise and motion flicker



Blurriness, noise and motion flicker

Three influential factors

Amplitude

Most dominant effect

Flicker is almost undetectable at amplitudes <8 QP and almost always detectable for larger amplitudes

Frequency

Major effect

Acceptance thresholds compared to constant low quality video: worse when above 1 Hz, often better when below 0.5 Hz

Content

Minor effect – within the class

But: content can influence flicker perception; low interaction for noise flicker and stronger for blur flicker

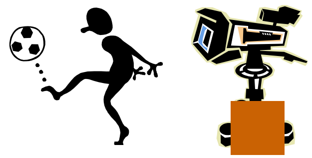
Remember for later:

- change at most once a second, better every two seconds

Dynamic Adaptive Streaming over HTTP

The State-of-the-Art

Dynamic Adaptive Streaming over HTTP



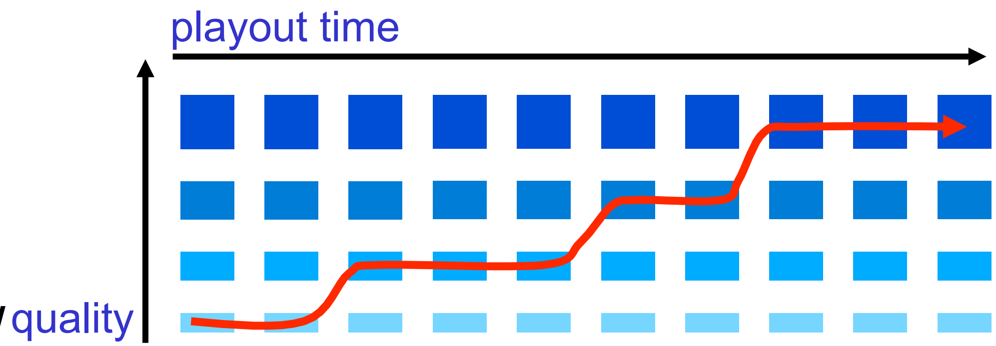
Divide video into segments at recording time
or convert later

Complete little movies

Choose the segment duration

Choose the number of layers

Choose the adaptation strategy



Typical segment lengths:
2-10 seconds
(2-hour movie → 3600++ small, indexed videos)

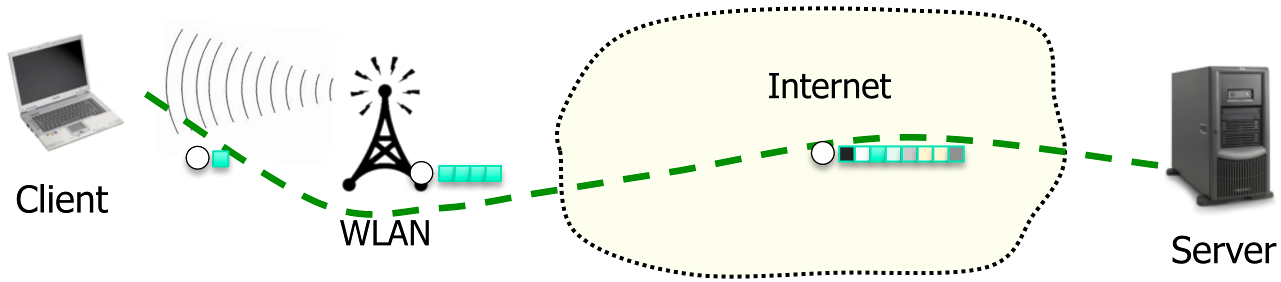
Dynamic Adaptive Streaming over HTTP

UDP-based streaming
- resists packets loss
- random loss

Applications
- IPTV
- DVB-H
- video conferencing
- classical RTSP/RTP servers

DASH & similar
- scales to available bandwidth
- congestion loss

Applications
- Commercial VoD: Netflix, Akamai, Microsoft, Apple, Comoyo, ...
- MPEG DASH
- Free VoD: Youtube, Metacafe, Dailymotion, Vimeo, Revver, Flixya ...



Dynamic Adaptive Streaming over HTTP

UDP-based streaming

Resilience to packet loss

Possibly resilience to bit errors

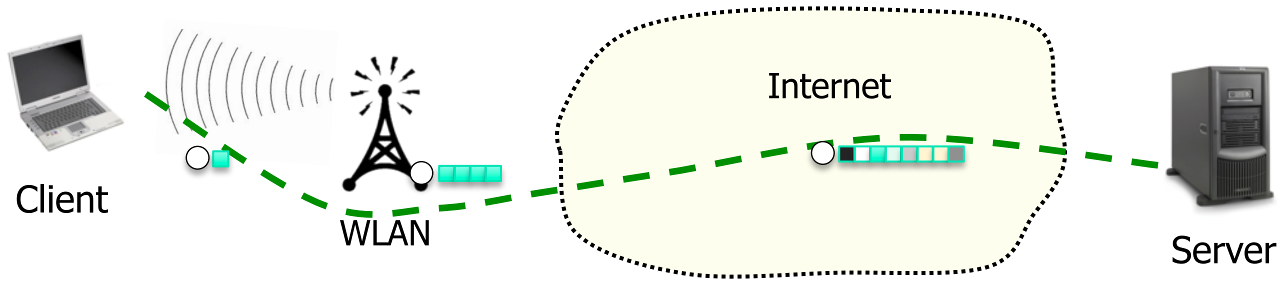
Possibly active adaptation (server-side decision)

DASH & similar

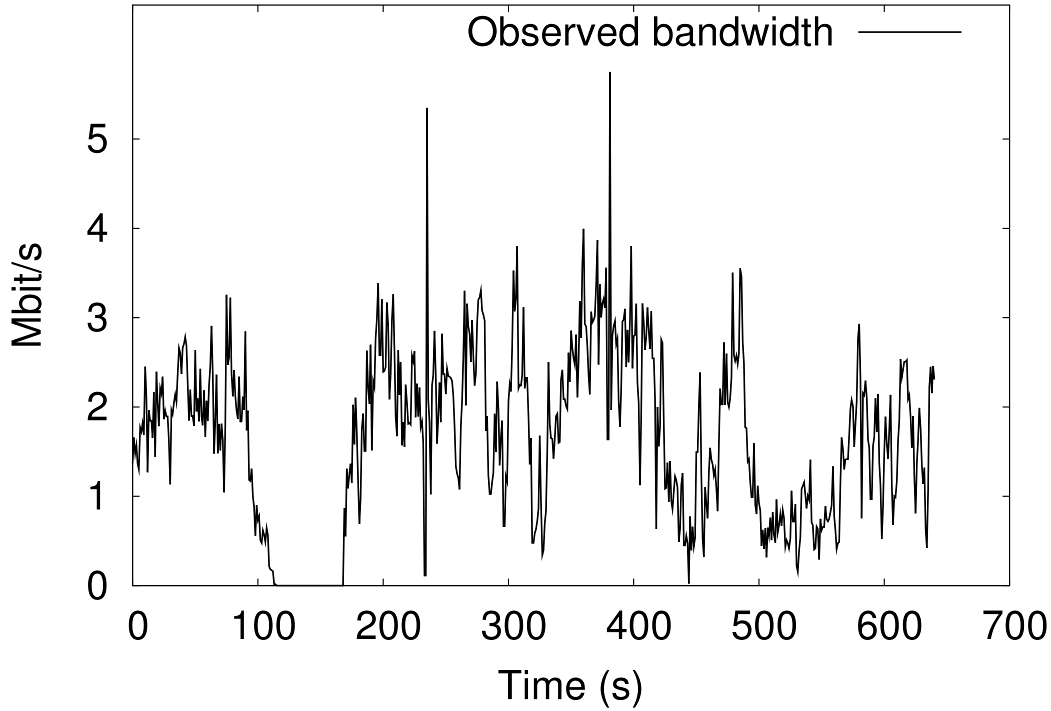
Resilience to buffer underruns

Active adaptation (client-side decision)

Works with web caches



Fluctuating Bandwidth Problem



Adaptive Delivery: Tested Systems

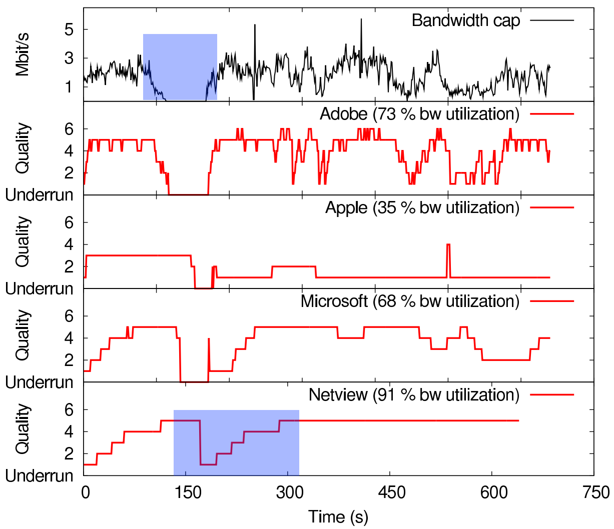
- Adobe Strobe Media Playback (v1.6.328 for Flash 10.1)
  using HTTP Dynamic Streaming Format
- Apple's native iPad player (iOS v4.3.3)
  using native HLS format
- Microsoft Silverlight/IIS Smooth (v4.0.60531.0 on Win7)
  using native Smooth format and default desktop scheduler
- Netview Media Client (v2011-10-10)
  using Apple HLS format (worst case) and Netview 3G scheduler

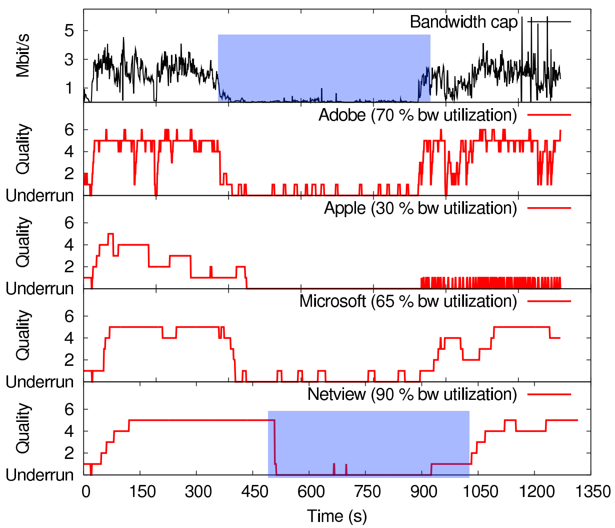
Comparison of Existing Quality Schedulers

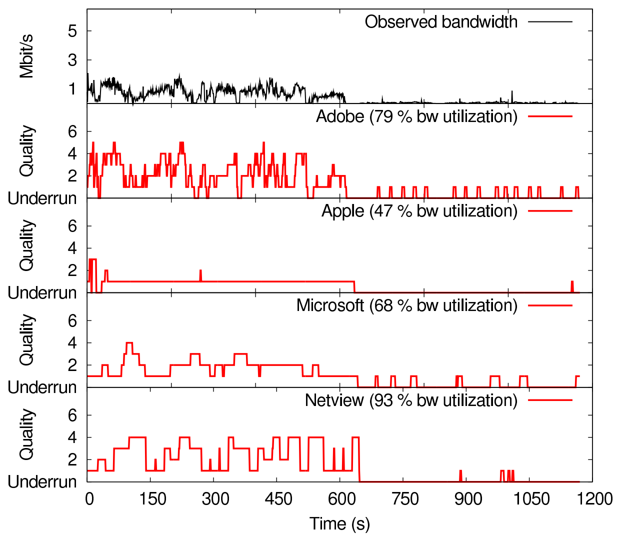
Bus:

Ferry:

Metro:







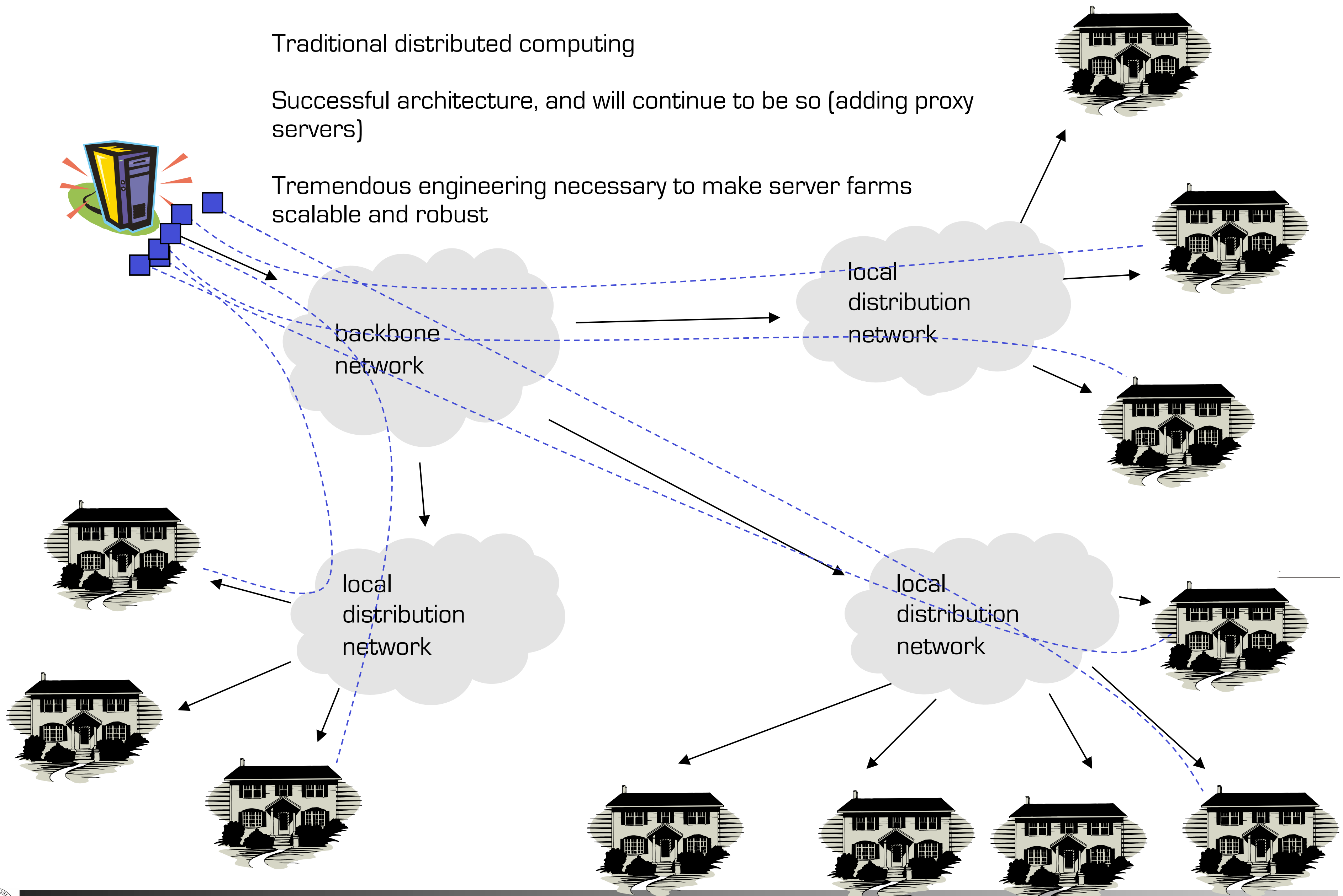
Distribution Architectures

Client-Server

Traditional distributed computing

Successful architecture, and will continue to be so (adding proxy servers)

Tremendous engineering necessary to make server farms scalable and robust



Distribution with proxies

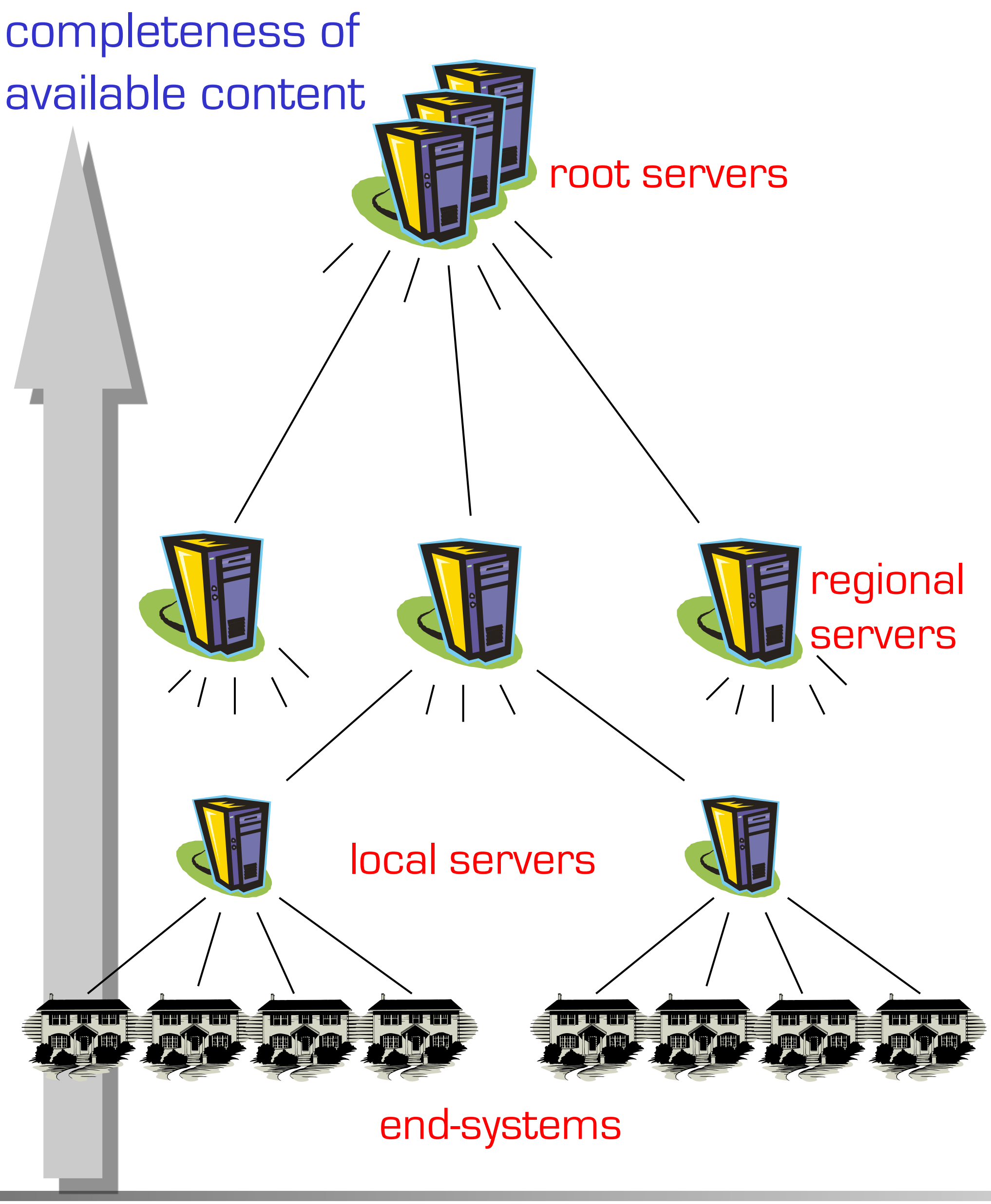
Hierarchical distribution system

- E.g. proxy caches that consider popularity

Popular data replicated more frequently and kept close to clients

Unpopular ones close to the root servers

➔ Where to keep copies?

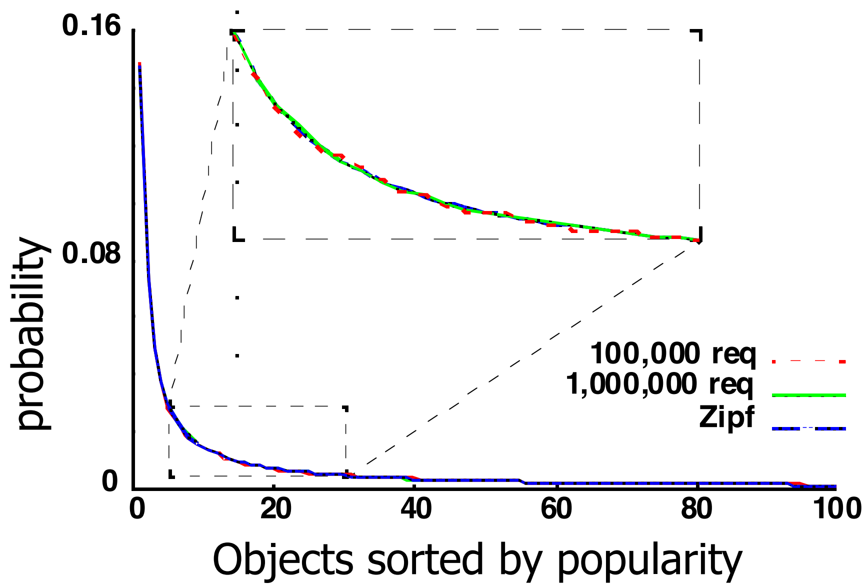


Zipf distribution and features

Popularity

- Estimate the popularity of movies (or any kind of product)
- Frequently used: Zipf distribution

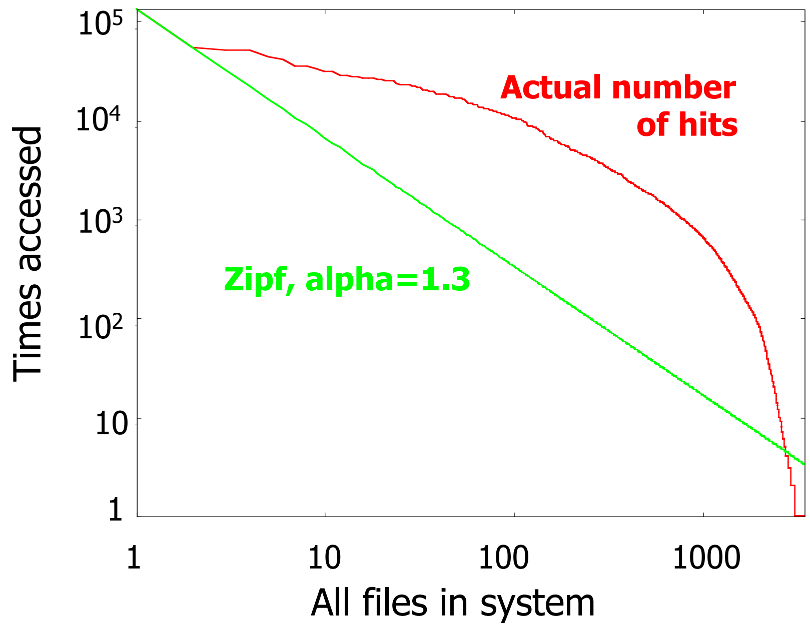
$$z(i) = \frac{C}{i^{\varsigma}} \qquad C = 1 / \sum_{n=1}^{N} \frac{1}{n^{\varsigma}}$$



Danger

- Zipf-distribution of a process
  - can only be applied while popularity doesn't change
  - is only an observed property
  - a subset of a Zipf-distributed dataset is no longer Zipf-distributed

Access probability distributions



Zipf-distribution

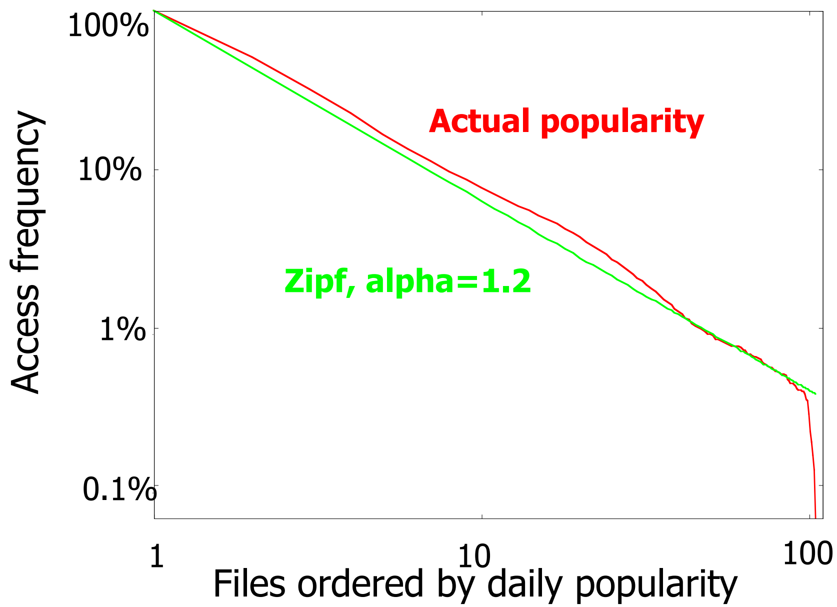
- often used to assign popularities to simulated files

Frequently observed

- Popularity over an entire log does not match a Zipf distribution

Why?

- Zipf-distribution models a snapshot in time
- Popularity of news changes daily
- Must not model according to Zipf-distribution with access counts for more than one interval of popularity change

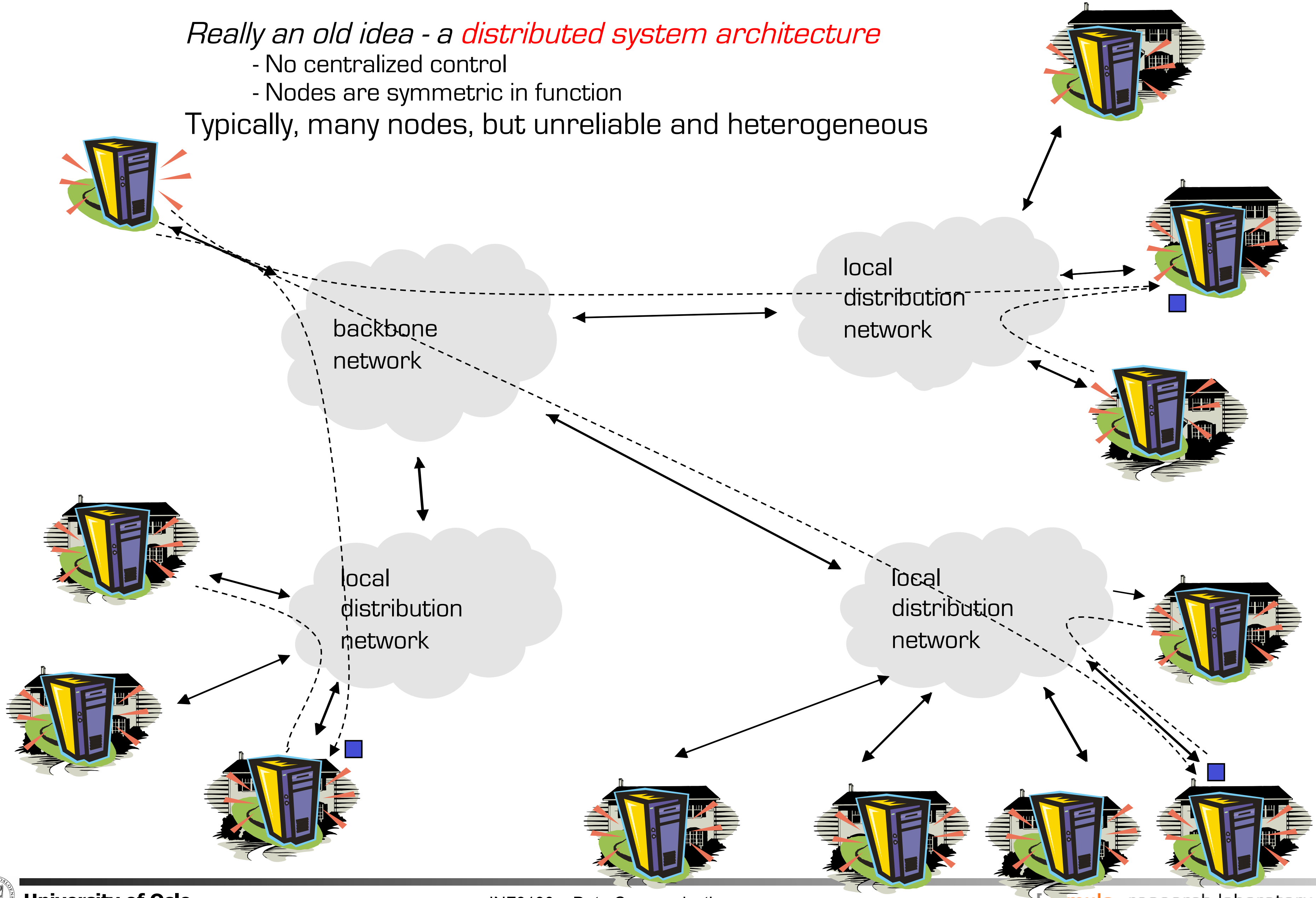


Peer-to-Peer (P2P)

Really an old idea - a distributed system architecture

- No centralized control
- Nodes are symmetric in function

Typically, many nodes, but unreliable and heterogeneous



P2P

Many aspects similar to proxy caches

- Nodes act as clients and servers
- Distributed storage
- Bring content closer to clients
- Storage limitation of each node
- Number of copies often related to content popularity
- Necessary to make replication and de-replication decisions
- Redirection

But

- No distinguished roles
- No generic hierarchical relationship: at most hierarchy per data item
- Clients do not know where the content is
  - May need a *discovery protocol*
- All clients may act as roots (origin servers)
- Members of the P2P network come and go (churn)

P2P

BitTorrent

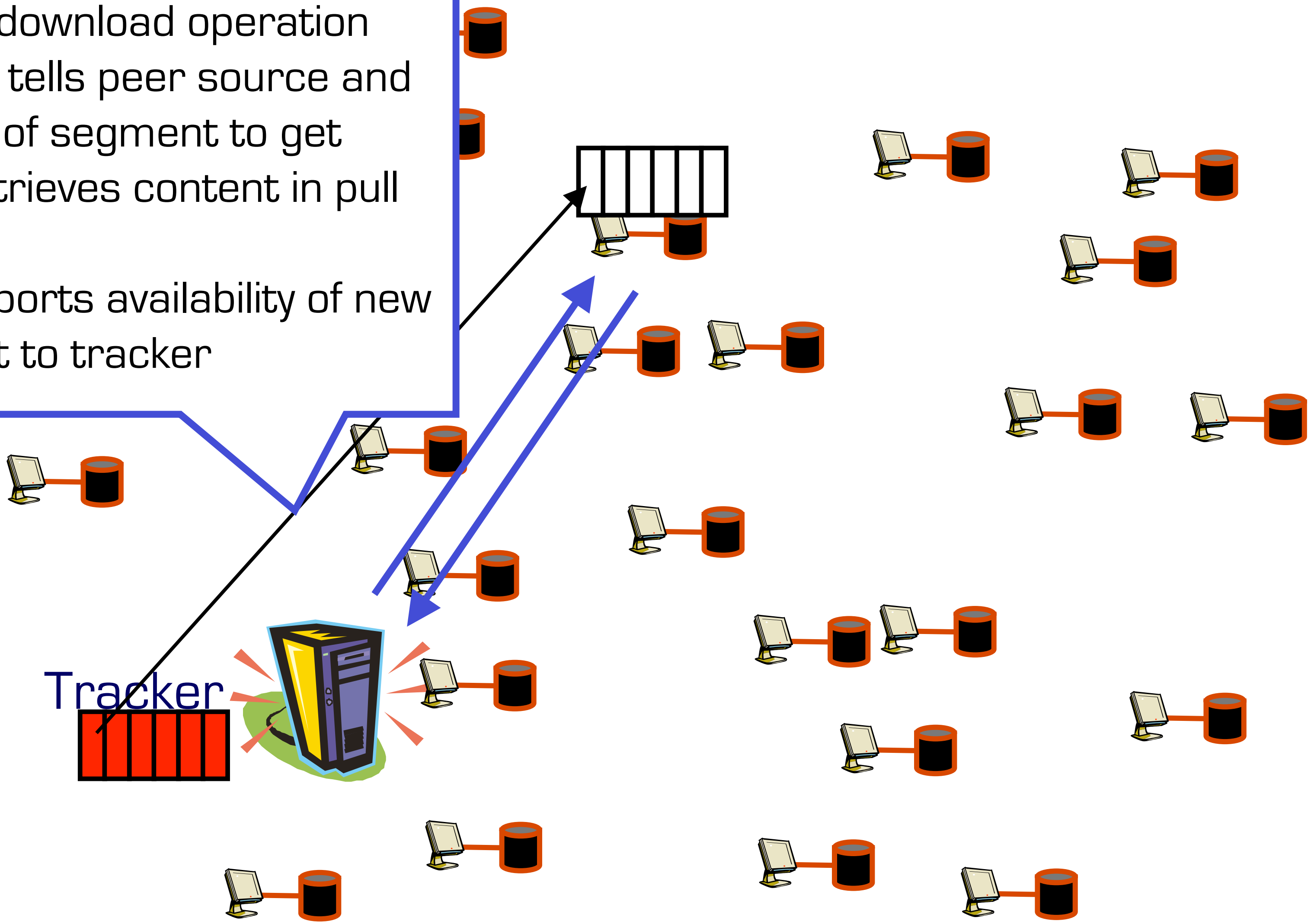
BitTorrent

- Distributed download system
- Content is distributed in segments
- Tracker
  - One central download server per content
  - Approach to fairness (tit-for-tat) per content
  - No approach for finding the tracker
- No content transfer protocol included

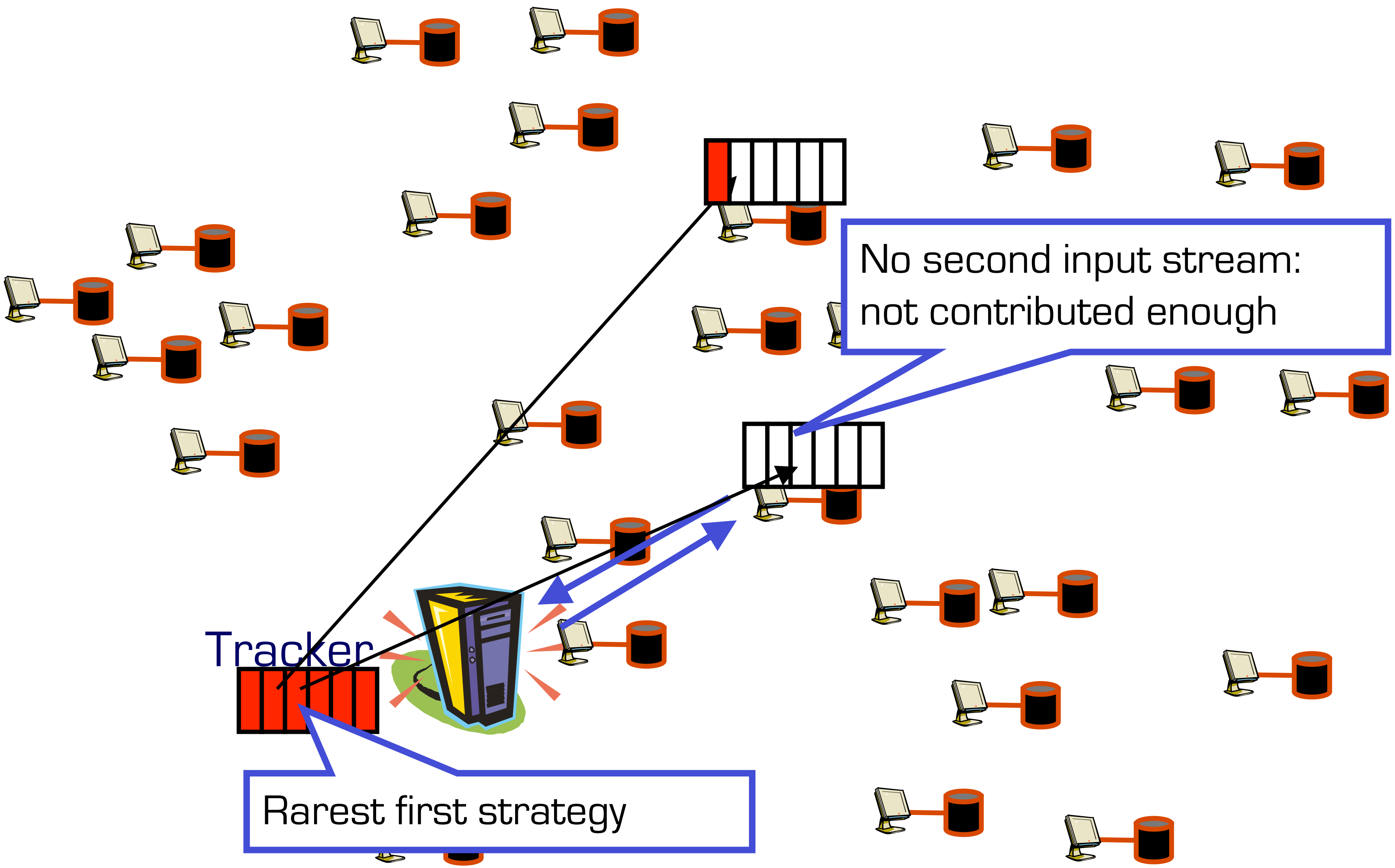
BitTorrent

Segment download operation

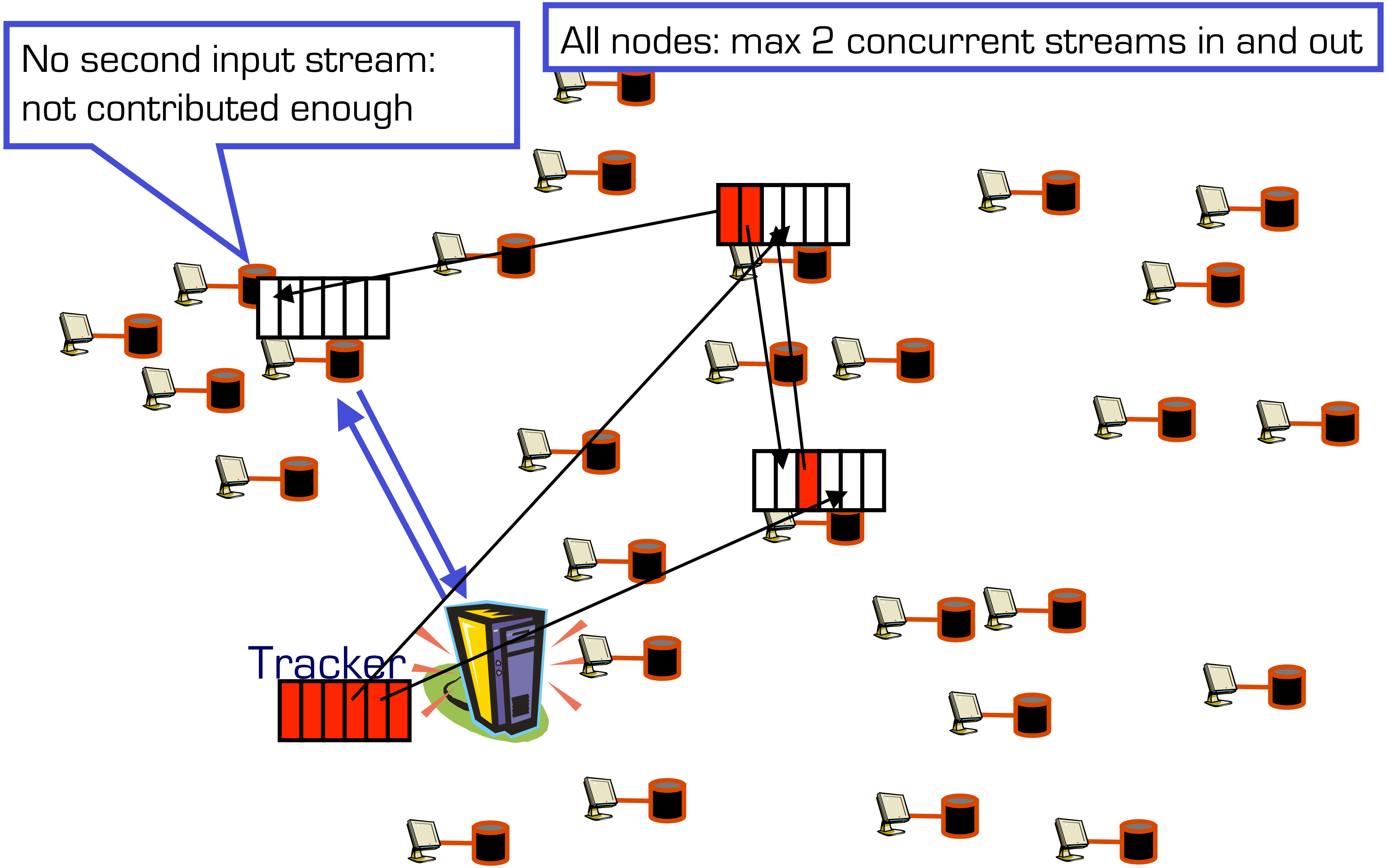
- Tracker tells peer source and number of segment to get
- Peer retrieves content in pull mode
- Peer reports availability of new segment to tracker



BitTorrent

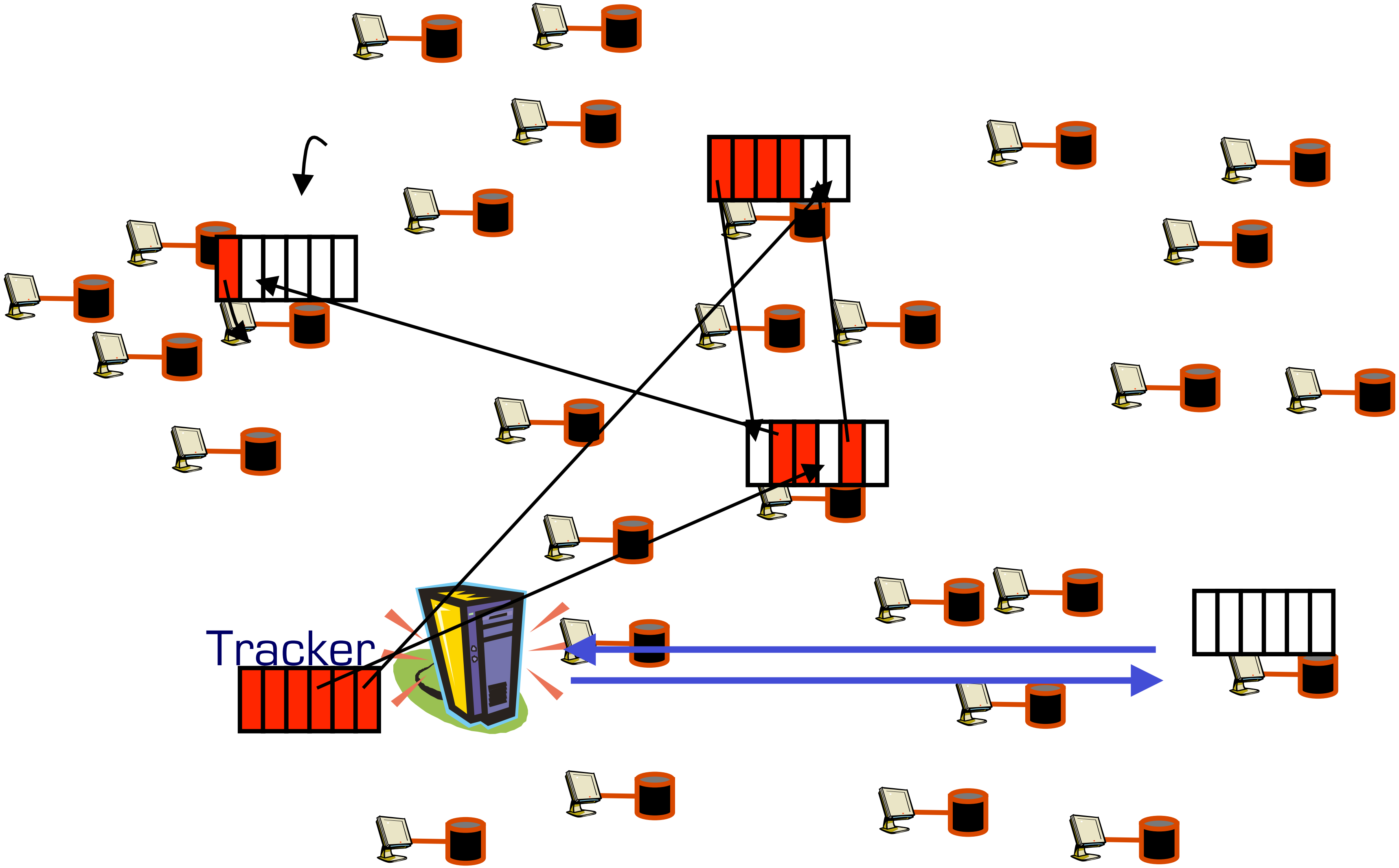

No second input stream:
not contributed enough
Tracker
Rarest first strategy

BitTorrent

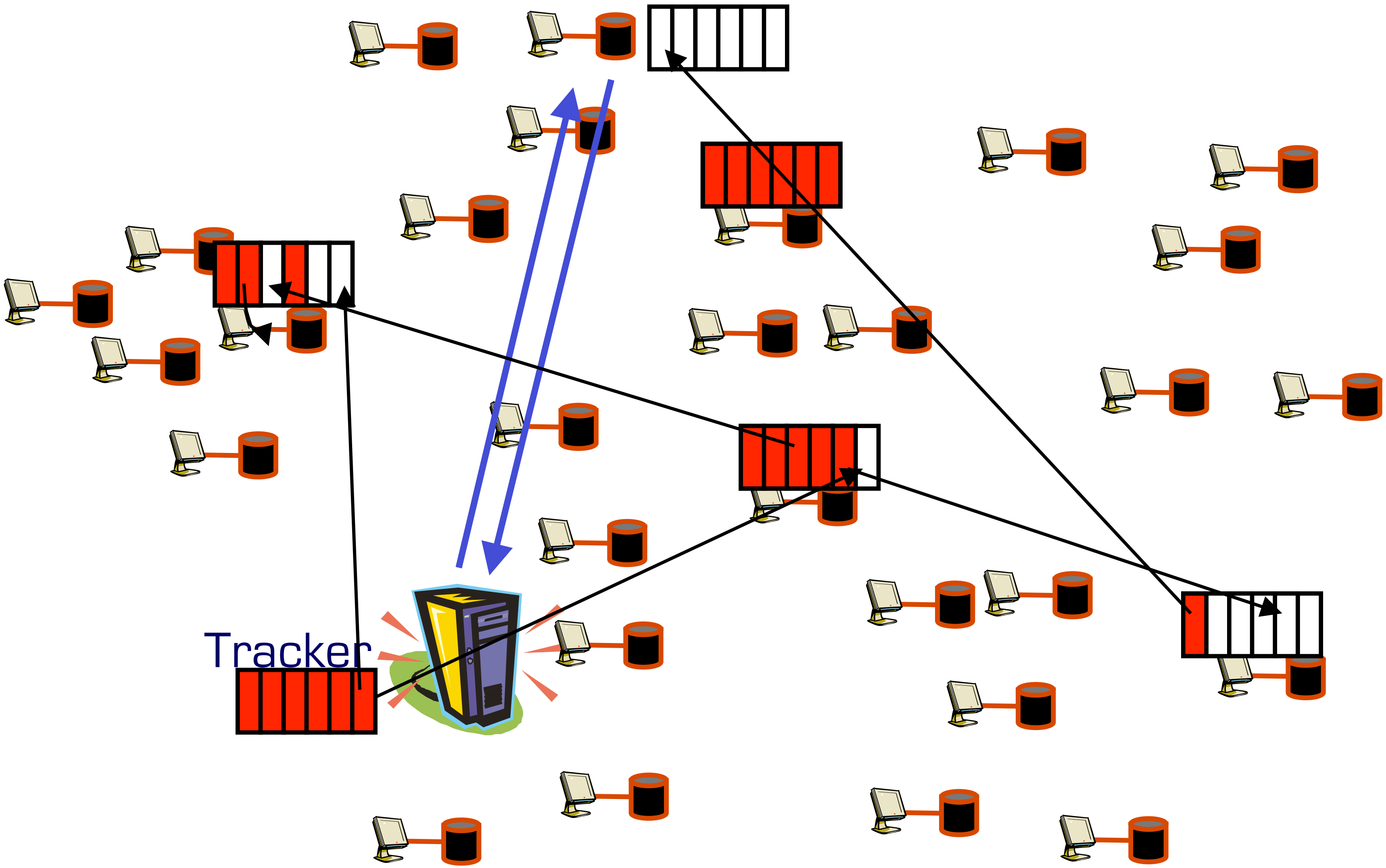


No second input stream: not contributed enough
All nodes: max 2 concurrent streams in and out
Tracker

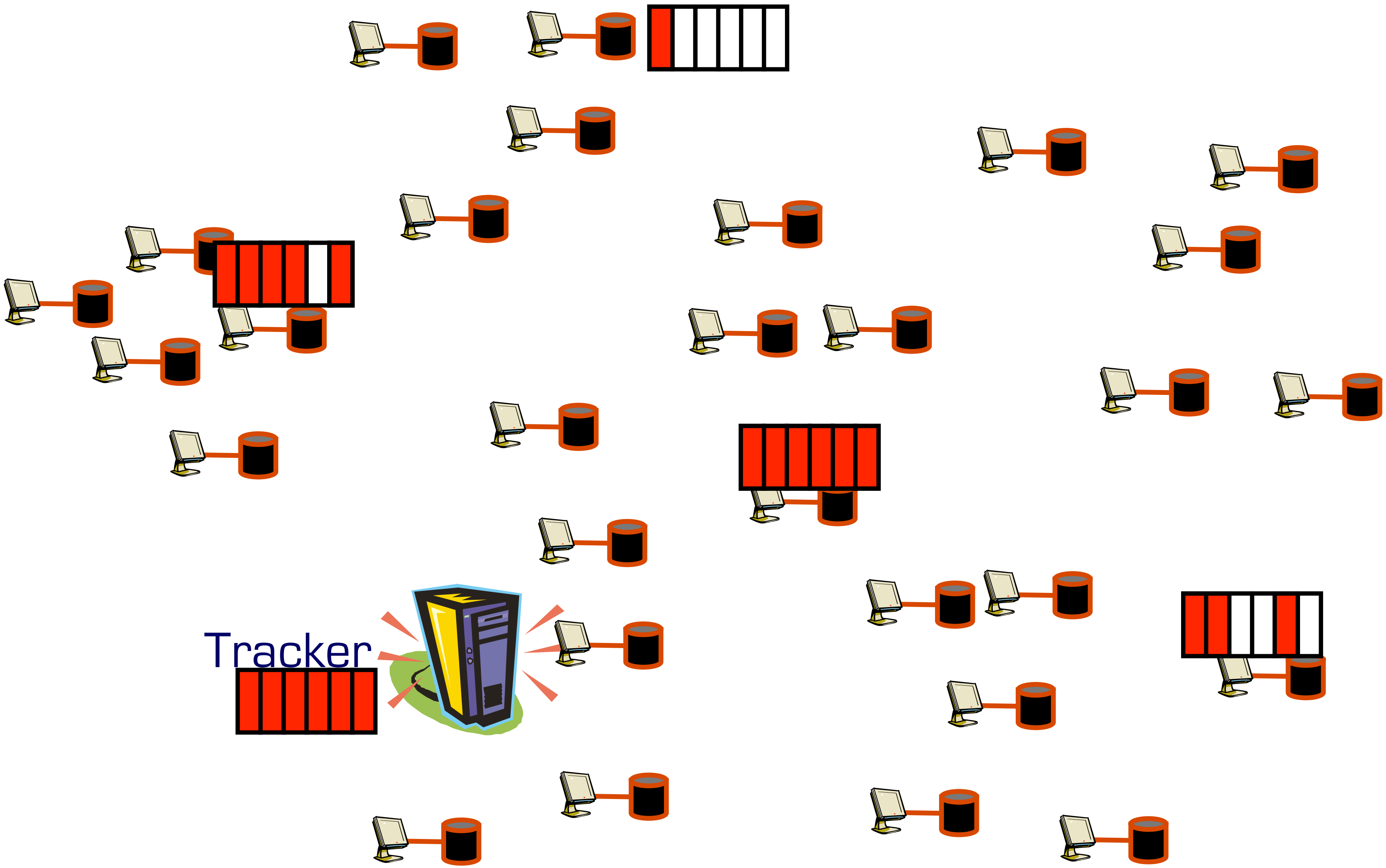
BitTorrent



BitTorrent



BitTorrent



P2P

Distributed Hash Tables (DHTs)

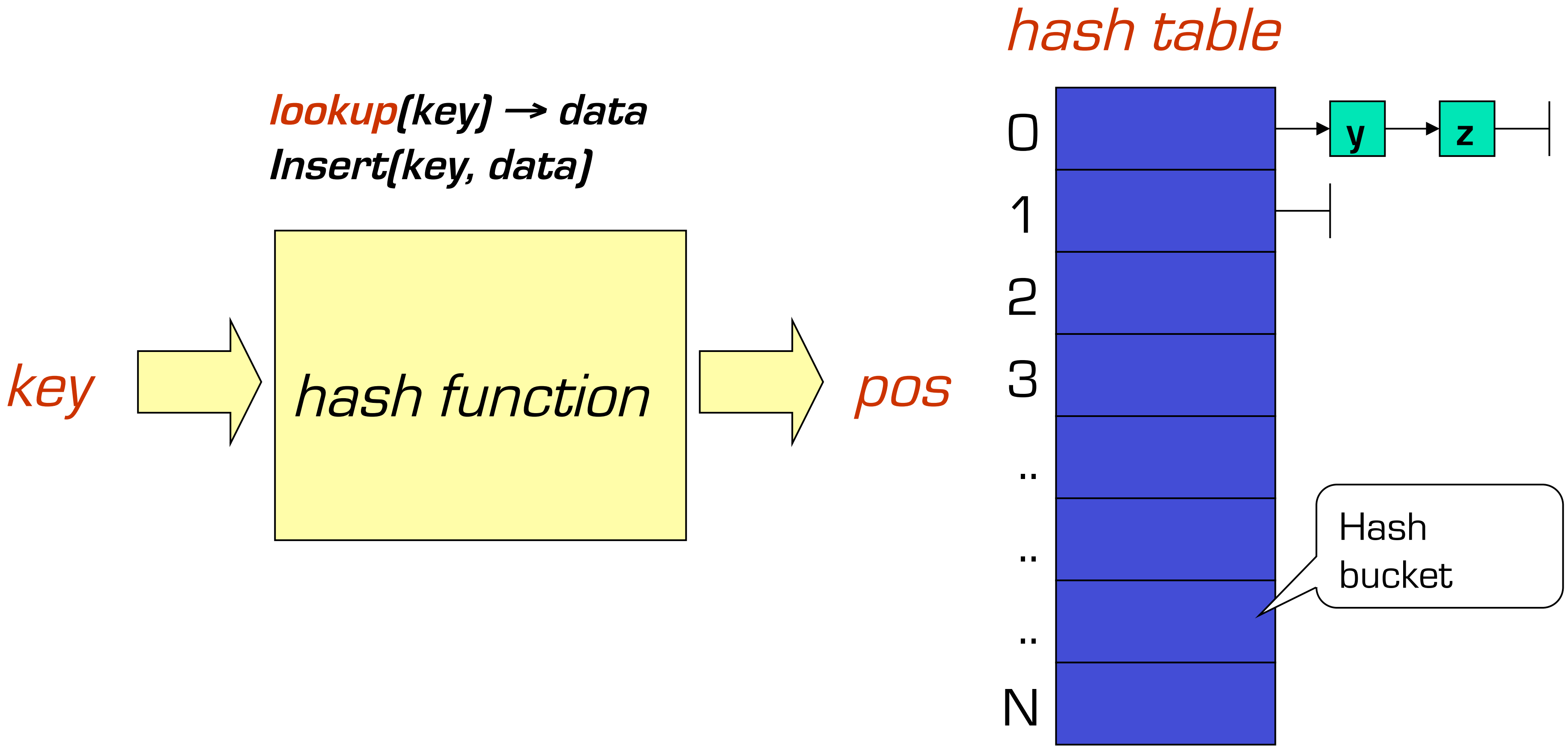
Challenge: Fast, efficient lookups

The BitTorrent tracker is a single point of failure

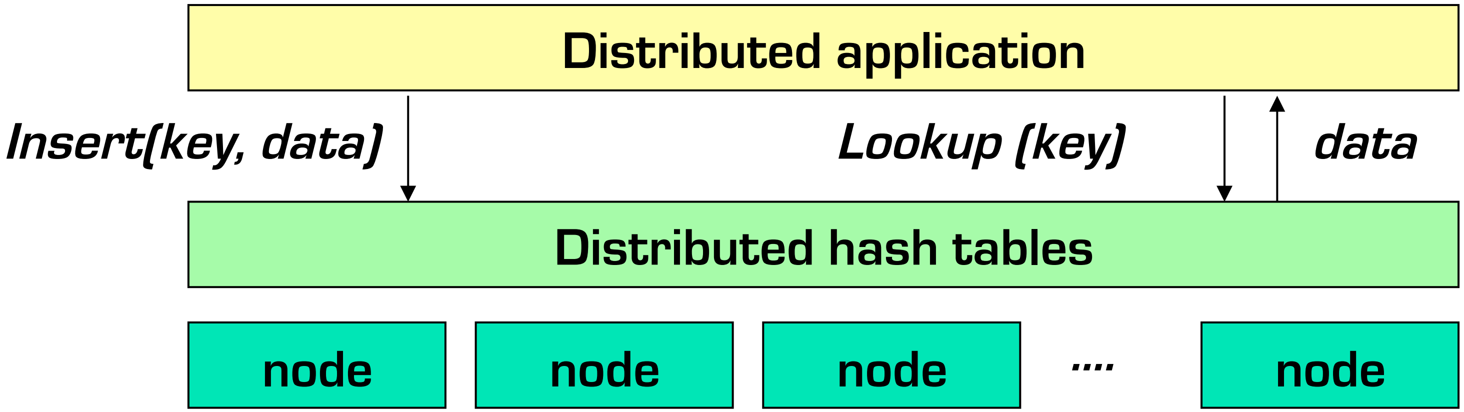
How to distribute tracker functions to many (all) machines?

➔ Distributed Hash Tables (DHTs) use a more structured key based routing

Lookup Based on Hash Tables



Distributed Hash Tables (DHTs)



Key identifies data uniquely

Nodes are the hash buckets

The keyspace is partitioned

- usually a node with ID = X has elements with keys close to X
- must define a useful ***key nearness metric***
- DHT should balances keys and data across nodes

Keep the hop count small

Keep the routing tables "right size"

Stay robust despite rapid changes in membership

Distributed Hash Tables

Chord

Chord

- Approach taken
  - Only concerned with efficient indexing
  - Distributed index - decentralized lookup service
  - Inspired by consistent hashing: SHA-1 hash
  - Content handling is an external problem entirely
    - No relation to content
    - No included replication or caching
- P2P aspects
  - Every node must maintain keys
  - Adaptive to membership changes
  - Client nodes act also as file servers

Chord IDs & Consistent Hashing

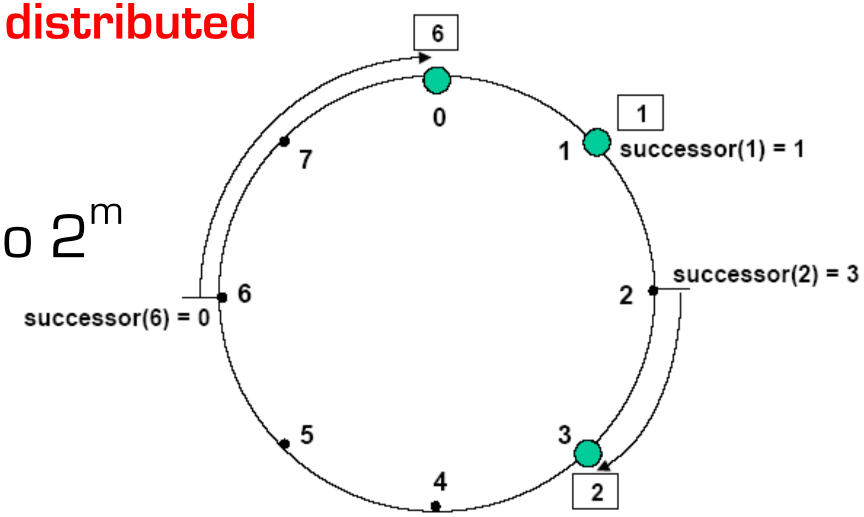
- *m*-bit identifier space for both keys and nodes
  - Key identifier = SHA-1(key)

Key="LetItBe" —SHA-1→ ID=54

  - Node identifier = SHA-1(IP address)

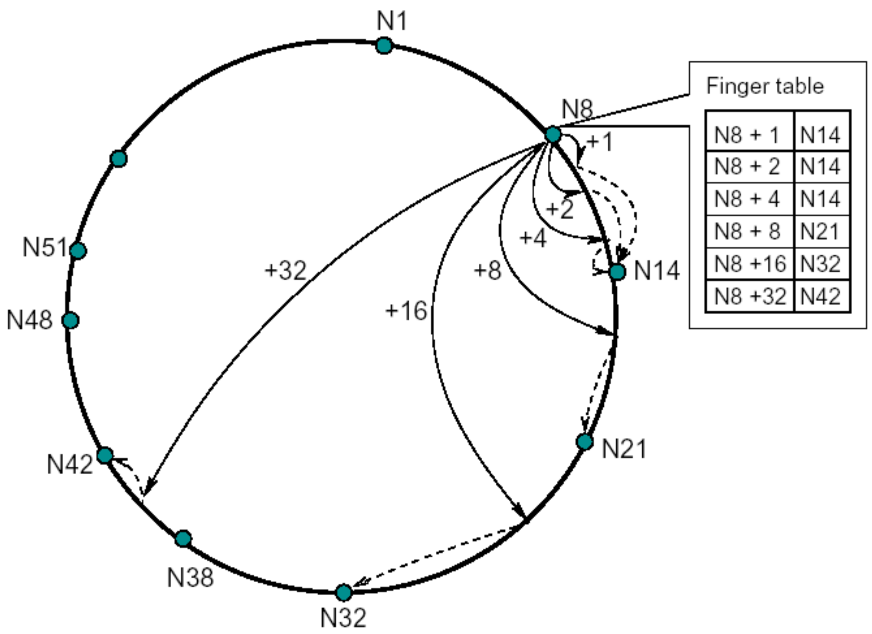
IP="198.10.10.1" —SHA-1→ ID=123

  - Both keys and identifiers are **uniformly distributed** in the space of m-bit numbers
- Identifiers ordered in a circle modulo 2^m
- A key is mapped to the first node whose node id ≥ key id



Routing: “Finger Tables”

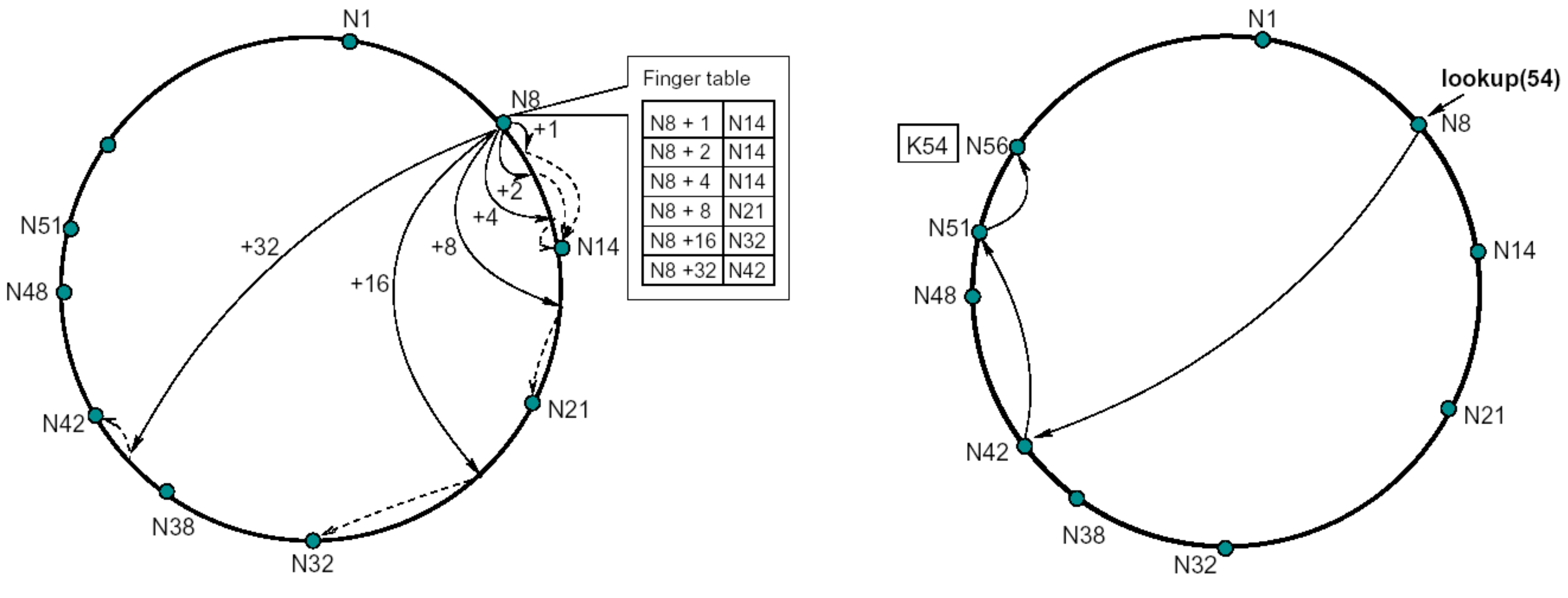
- Every node knows *m* other nodes in the ring
- Increase distance exponentially
- Finger *i* points to successor of $n+2^i$



(a)

Routing: "Finger Tables"

- Every node knows *m* other nodes in the ring
- Increase distance exponentially
- Finger *i* points to successor of $n+2^i$
- Lookup jumps to nodes with largest ID < hash(search term) in its lookup table



(a) (b)

Chord Assessment

Large distributed index

Scalability, fairness, load balancing

- Space complexity: routing tables are size ***O(log(#nodes))***
- Logarithmic insert effort
- Network topology is **not** accounted for
- Quick lookup in large systems, low variation in lookup costs

Content location

- Run-time complexity: ***O(log(#nodes))*** lookup steps
- Search by hash key: limited ways to formulate queries

No failure resilience in basic approach

- Easy fix
  - Successor lists allow use of neighbors to failed nodes
  - create several index with different has functions (***O(1)***)

Summary

1. Signaling protocols: RTSP and SIP
2. Adaptation with RTP: Loss Delay Adjustment Algorithm (LDA)
3. How to adapt video quality?
4. Dynamic Adaptive Streaming over HTTP (DASH) and related techniques
5. Distribution Architectures
   – using the Zipf distribution
6. P2P
   – DHTs
   – Chord