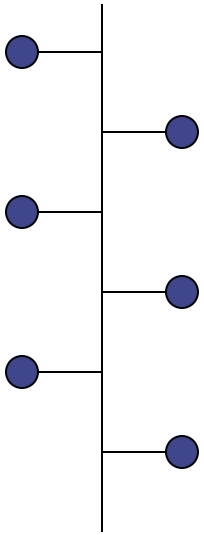
# Rechnernetze und Verteilte Systeme

## Introduction to Communication Networks and Distributed Systems



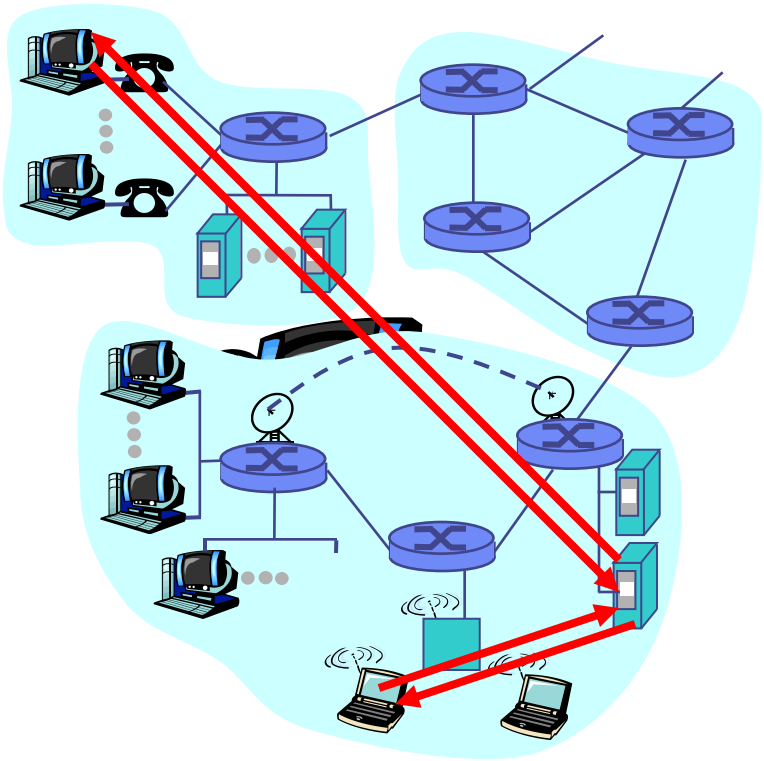
***Unit 6: P-t-P***

Prof. Dr.-Ing. Adam Wolisz

TKN Telecommunication Networks Group

# Client-Server Model

[Kurose, Ross]



server:

- always-on host
- permanent IP address
- server farms for scaling

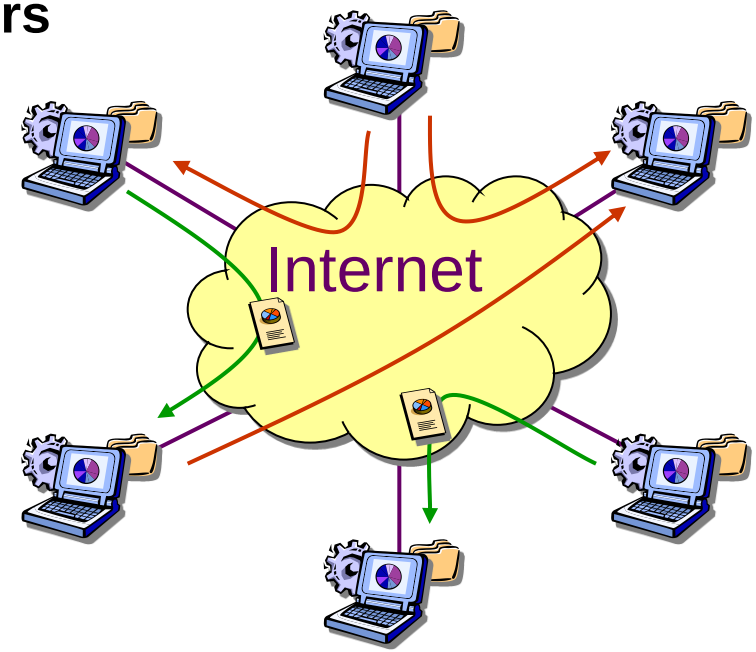
clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

# What is P2P?

- A distributed system architecture
  - Typically many nodes, but unreliable and heterogeneous
  - Nodes are equivalent in function -**peers**
  - Take advantage of distributed, shared resources (bandwidth, CPU, storage) on peer-nodes

    DATA can be at ANY node !
  - Fault-tolerant, self-organizing
  - Operate in dynamic environment, frequent join and leave is the norm



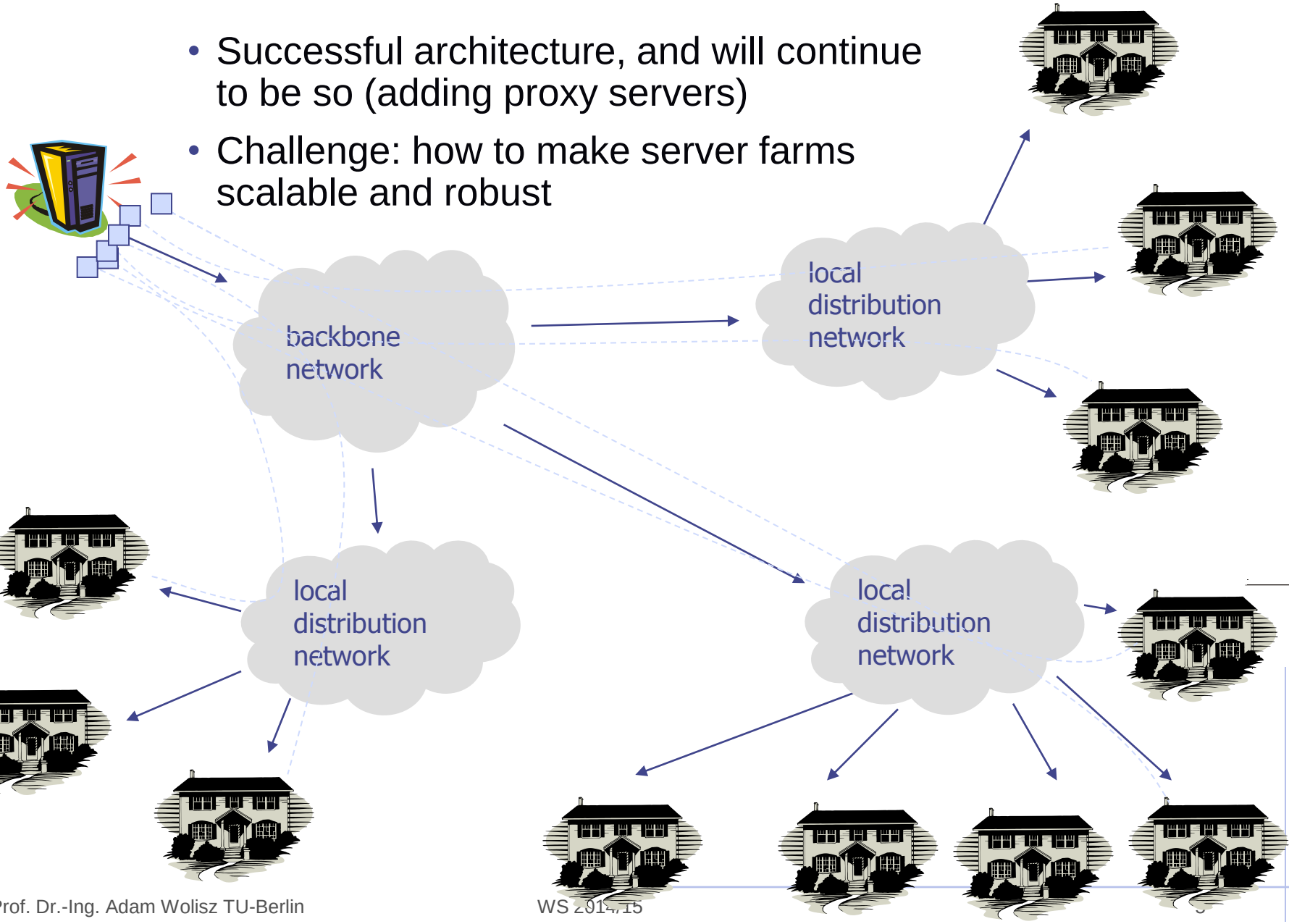
# How Did it start?

- A killer application: Free music over the Internet
- Key idea: share the **content**, storage *and* bandwidth of individual (home) users
- Each user stores a subset of files
- Each user has access (can download) files from all users in the system

# Client-Server Downloads

[Ana Preston; internet2]

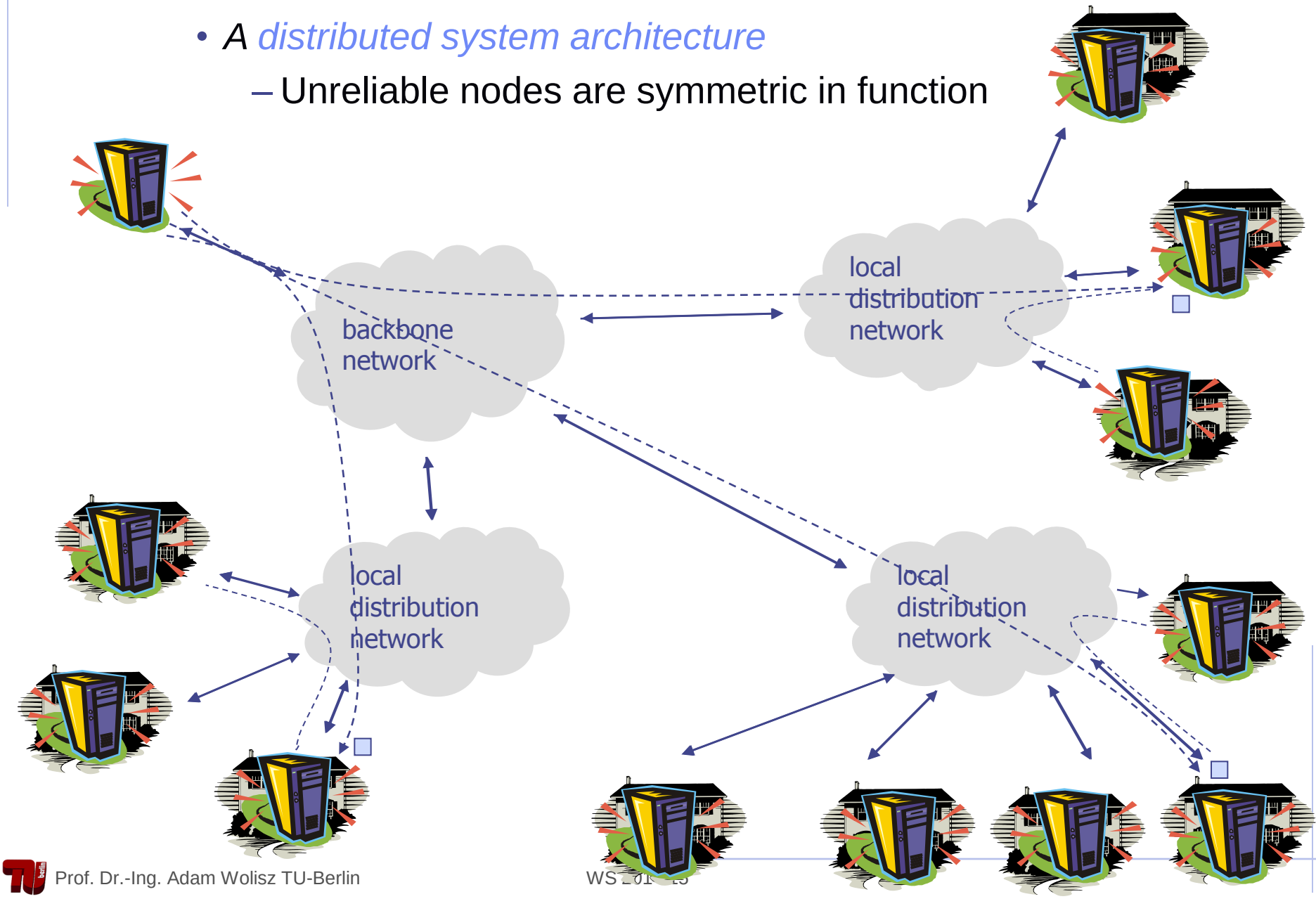
- Successful architecture, and will continue to be so (adding proxy servers)
- Challenge: how to make server farms scalable and robust



# Peer-to-Peer (P2P) support

[Ana Preston; internet2]

- *A distributed system architecture*
  - Unreliable nodes are symmetric in function



# Challenges

- Main Challenge: Find where a particular file is stored
- Scale: up to hundred of thousands or millions of machines
- Dynamicity: machines can come and go any time

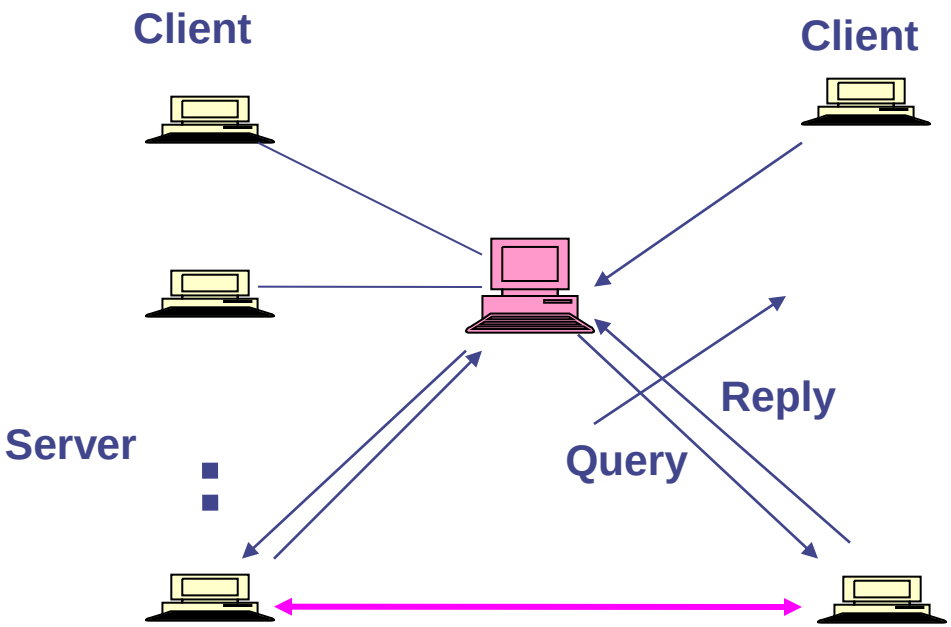
- “Quality” of Searches:
  - Recall (% of all relevant items retrieved)
  - Distinct Recall (% of all relevant distinct items retrieved)
  - Response Time (Latency) to 1st result

# Napster

[Shenker &Stoica, UCB]

- Assume a centralized index system that maps files (songs) to machines that are alive
- Nodes Register their contents with this index
- How to find a file (song)
  - Query the index system → return a machine that stores the file
    ! Ideally this is the closest/least-loaded machine !
  - ftp the file (directly form the holder!)
- Advantages:
  - Simplicity, easy to implement sophisticated search engines on top of the index system
  - Access Control possibilities
- Disadvantages:
  - Robustness, scalability (?)
- ***napster.com responsible for users' copyright violation***
  - ***"Indirect infringement"***

# Napster model




Server might return a LIST of candidates:

- Client Checks them in turn – might check the availability and connectivity (bit rates)
-Client decides where to download from…

**All Communication uses TCP!**

# Entirely Decentralized scheme GNUTELLA

- Every user acts as a client, a server or both (servent).
- User connects to a framework and becomes a member of the community, allowing others to connect through him/her
- Users speak directly to other users with no intermediate or central authority
- Not one entity controls the information that passes through the community

# Gnutella

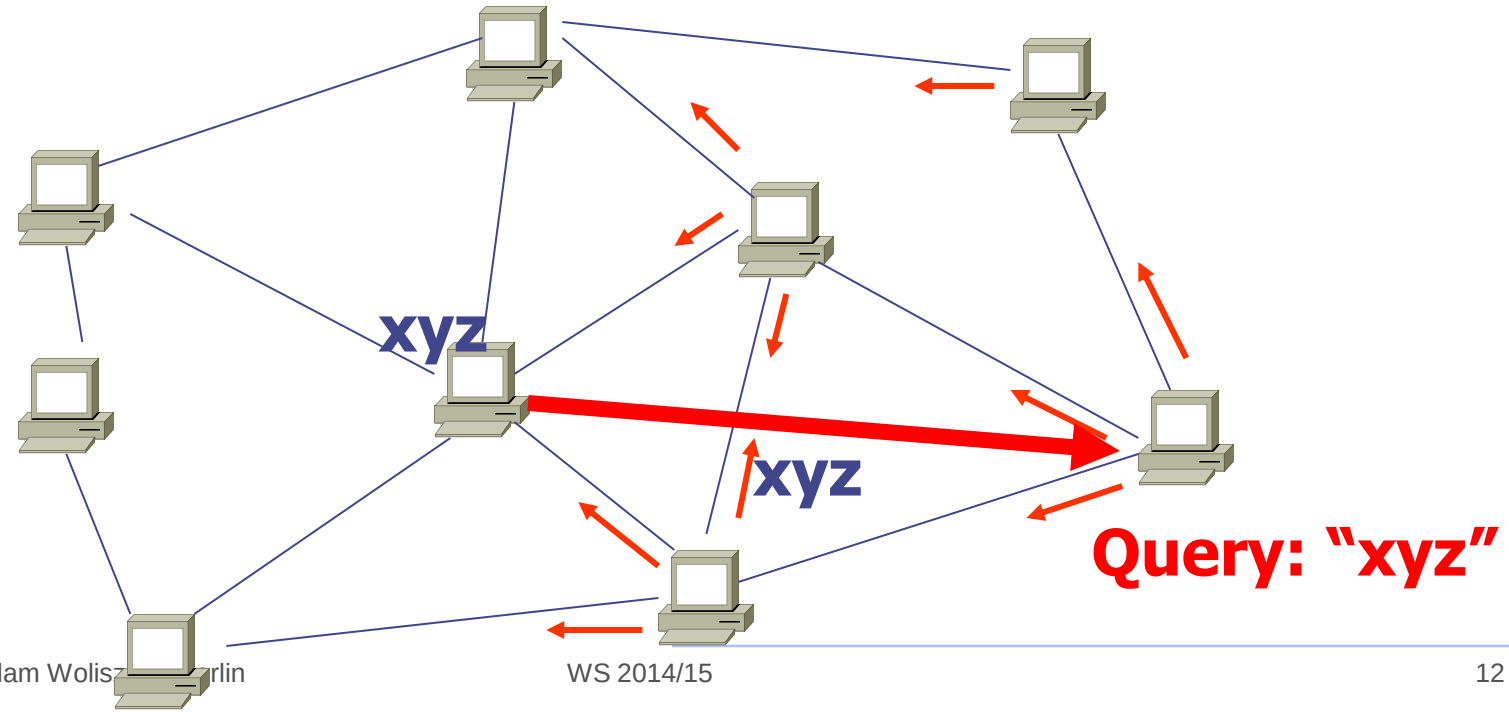
[Shenker &Stoica, UCB]

- How to find a file:
  - Send request to all neighbors
  - Neighbors recursively multicast the request
  - Eventually a machine that has the file receives the request, and it sends back the answer (reverse path – directly!)
- Advantages:
  - Totally decentralized, highly robust
- Disadvantages:
  - Not scalable; the entire network can be swamped with request
  - Bound the number of retransmissions of any query (TTL)

# Unstructured Networks

[B. T. Loo, Intel Res.]

- Gnutella protocol has 5 main message types
  - Query
  - QueryHit (response to query)
  - Ping (to probe network for other peers)
  - Pong (reply to ping, contains address of another peer)
  - Push (used to initiate file transfer)



# Details…

[I.Gupta,N.Vaidya, UIUC]

- Periodic Ping-pong to update neighbor lists in spite of peers joining/leaving
- Requestor chooses best QueryHit responder
  - Initiates HTTP request directly to responder's ip+port

    GET /get/<File Index>/<File Name>/HTTP/1.0\r\n

    Connection: Keep-Alive\r\n

    Range: bytes=0-\filesize\n

    User-Agent: Gnutella\r\n

    \r\n
- Responder then replies with file packets following:

  HTTP 200 OK\r\n

  Server:Gnutella\r\n

  Content-type:application/binary\r\n

  Content-length: 1024 \r\n

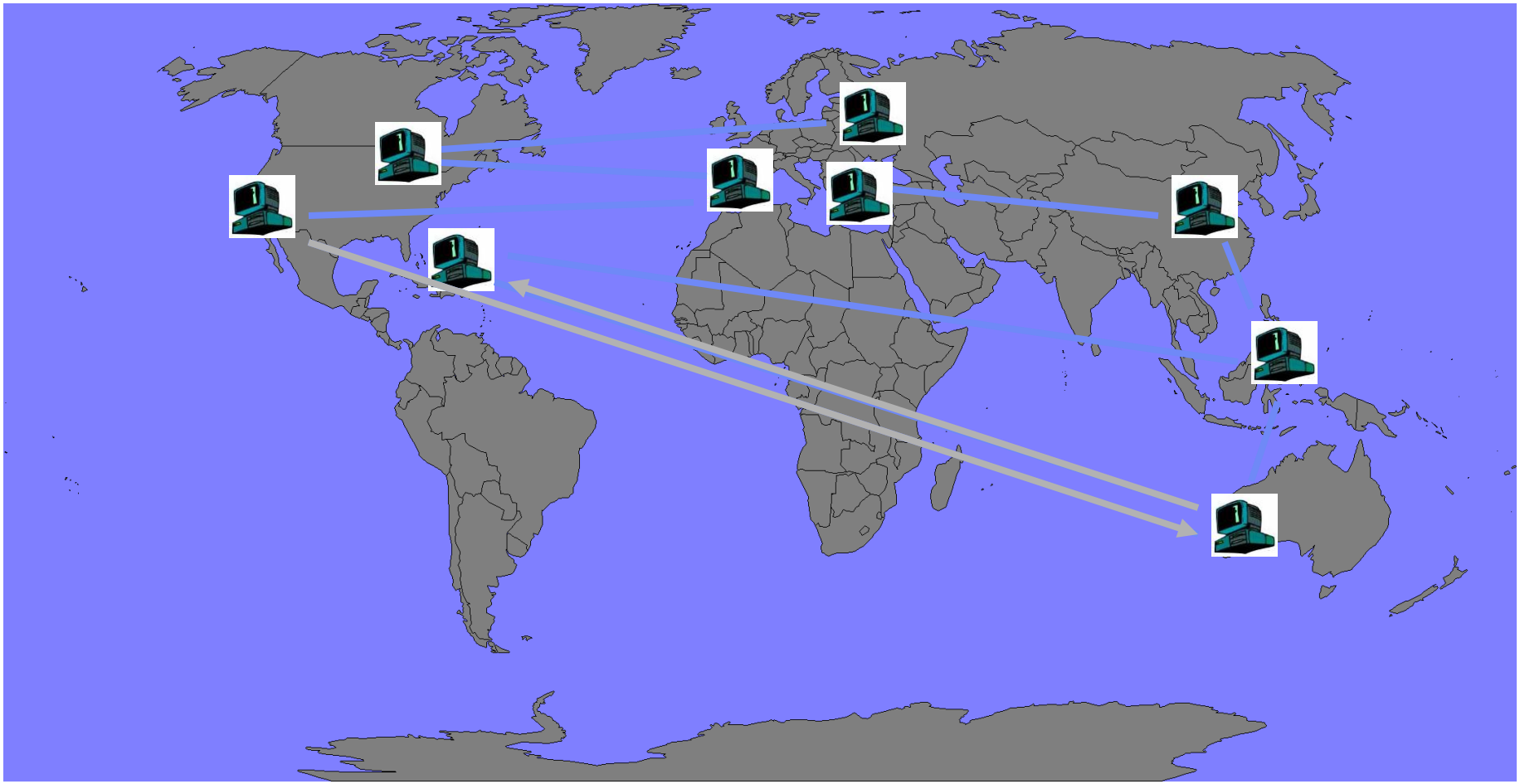
  \r\n

# Avoiding excessive traffic

- To avoid duplicate transmissions, each peer maintains a list of recently received messages
  - Query forwarded to all neighbors except peer from which received
  - Each Query (identified by DescriptorID) forwarded only once
  - QueryHit routed back only to peer from which Query received with same DescriptorID
  - Duplicates with same DescriptorID and Payload descriptor (msg type) are dropped
  - QueryHit with DescriptorID for which Query not seen is dropped
- Multiple ideas for further performance improvements
  - Forwarding Queries only to some randomly chosen directions
    - Reduced nzmber of messages, reduced

# Gnutella Topology Mismatch

[Yatin Chawathe]



# Gnutella: Free Riding:

[Yatin Chawathe]

- Does really  everybody contribute?
  - Some data − see fig.  - More than 25% of Gnutella clients share no files; 75% share 100 files or less
  - Others claim up to 75% user with marginal contributions...
- Gnutella has a high percentage of free riders
- If only a few individuals contribute to the public good, these few peers effectively act as centralized servers.

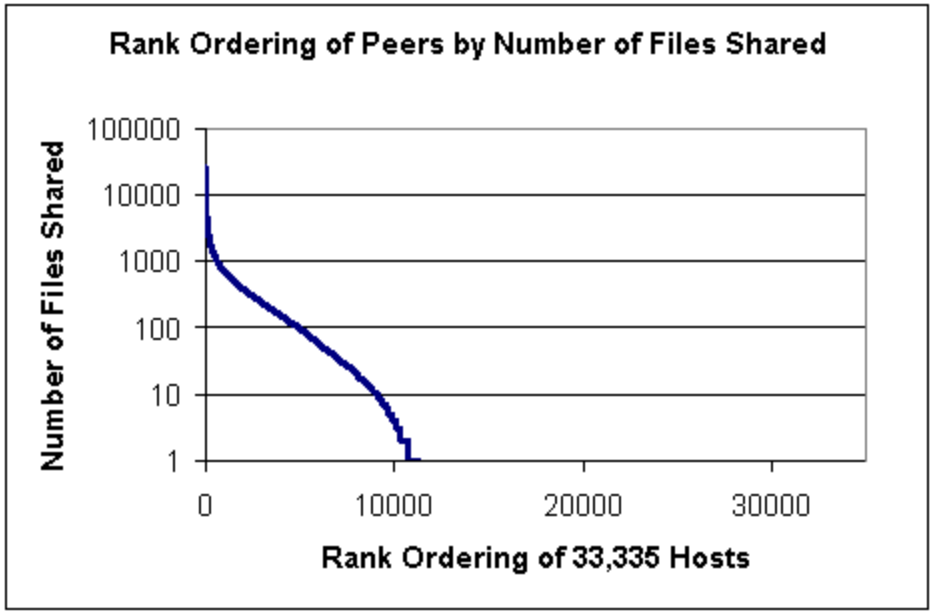
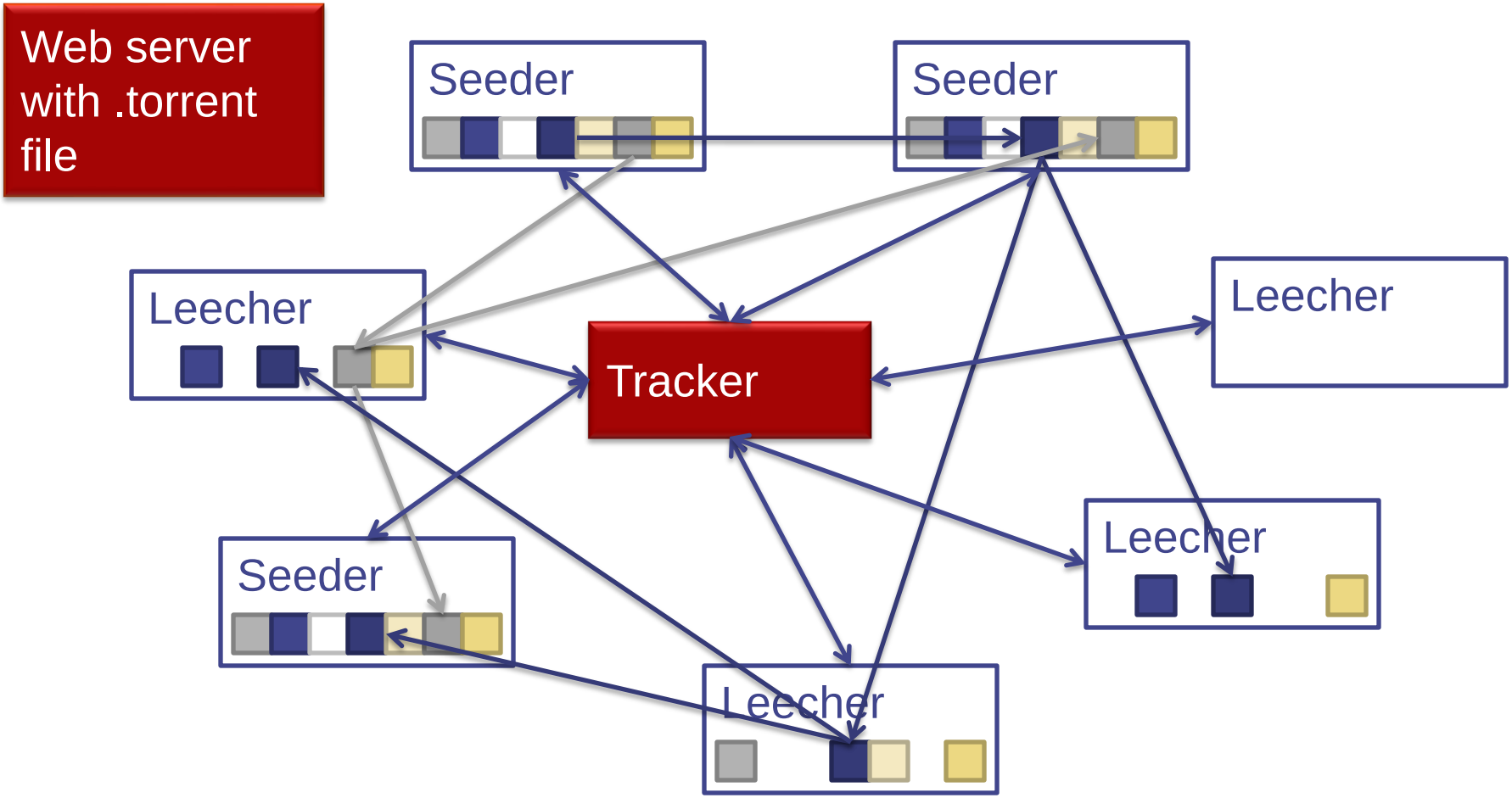



*Figure 1*

Adar and Huberman (Aug '00)

# BitTorrent


Web server with .torrent file
Seeder
Seeder
Leecher
Leecher
Tracker
Seeder
Leecher
Leecher

# BitTorrent: Approach

- Peer downloads the torrent file from a web server with following information
  - File data: name, length, etc.
  - Tracker URL
  - SHA1 value of the file blocks
- Assumed
  - At least one seeder exists
  - Peer registers with the tracker
- Approach
  - Tracker sends a randomly chosen selection of peers (approx. 40)
  - Peer creates direct connections to the submitted participants
  - Peer downloads file block-wise from the neighboring peers
  - If list of neighbors contains less than 20 neighbors, than new list is requested from the tracker

# BitTorrent: Details

- Block transmission via TCP connections
- Each block is divided into sub blocks of 16 KB
- Transmission in pipelining mode
- Before a new block is requested, the missing sub-blocks from old blocks are completed
- Which packet should be requested next?
  - "Rarest First": Packet with the least number of replications among the known peers
    - Coupon Collector Problem
  - "Random First Piece": First packet is randomly chosen
    - ⇒Gain significant number of packets so the peer is able to distribute packets too
  - "Endgame Mode": Request all packets from all connected peers

# BitTorrent: Tit for Tat

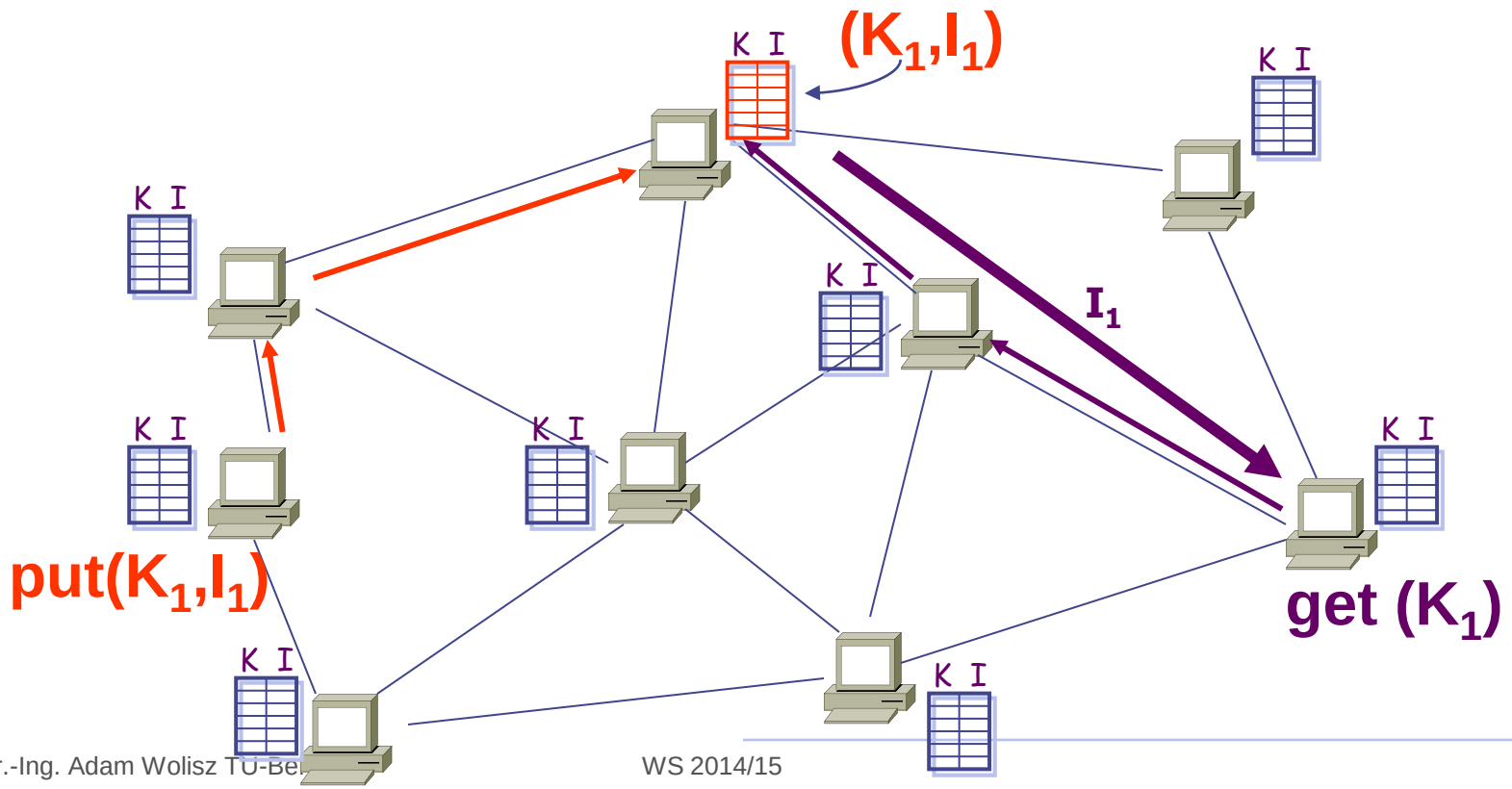
- How can we prevent that users consume but do not distribute content?
- Basic idea
  - Send content only to participants that also distribute content (Tit for Tat)
- Problems
  - Some peers do not have enough content to distribute
  - Some peers have poor network connectivity so no one selects those for download
  - How can we detect peers that do not distribute (enough)?

# Structured Networks

[B. T. Loo, Intel Res.]

- Is there some way of maintaining meta-information about peers that leads to more intelligent search?
- ➔ Distributed Hash Tables (DHTs)

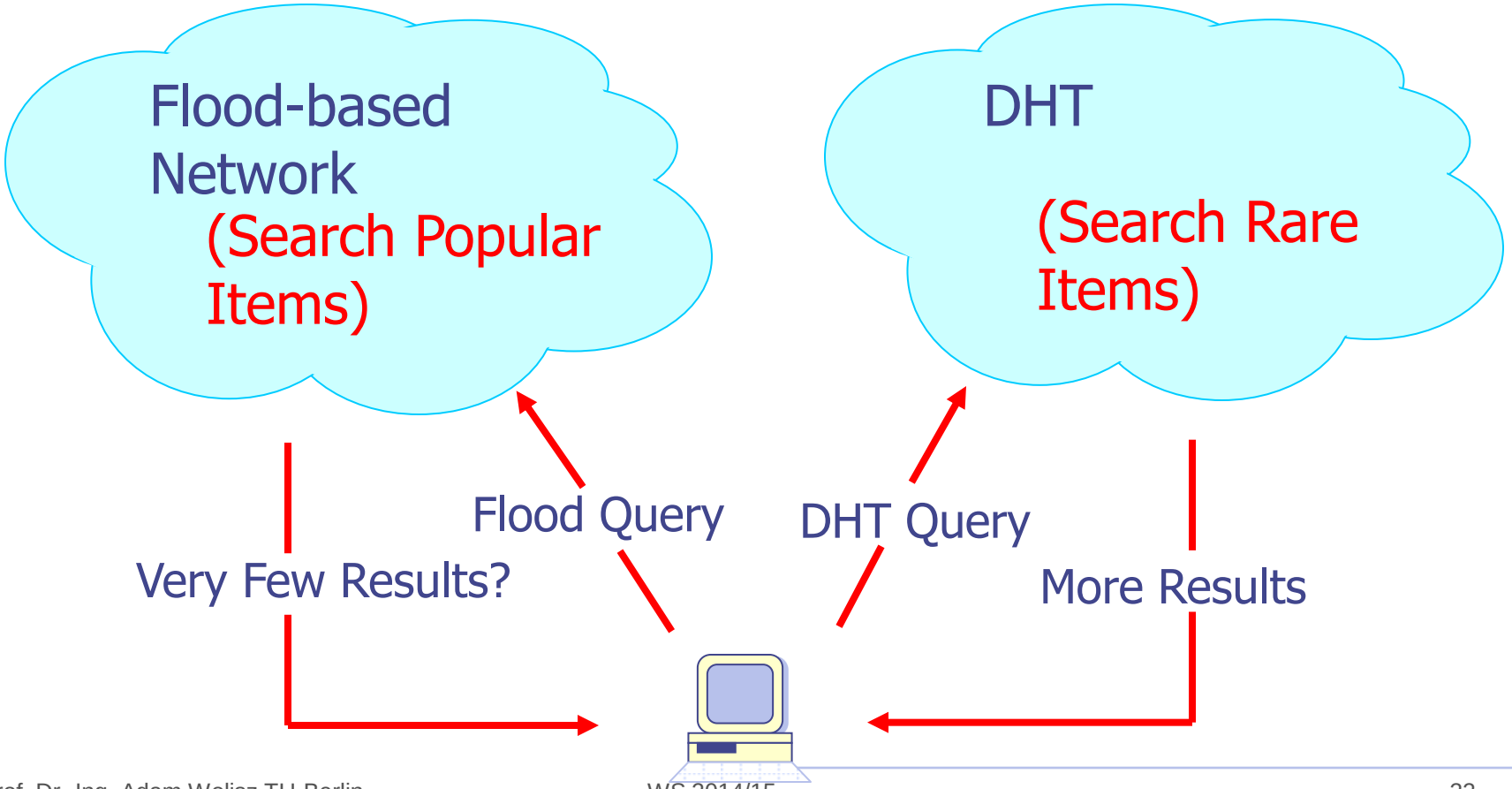
  interface: **put**(key,item), **get**(key), **delete**(key)
- Guarantees on recall – critical if few copies…



# Hybrid Search

[B. T. Loo, Intel Res.]

Hybrid = “Best of both worlds”



# Peer-to-Peer Applications…

- NUMEROUS!!!
- Well known example: SKYPE….