CSx55: Distributed Systems [P2P Systems]

P2P Systems

Here's what oft missed in the tale
It's all about the scale
Robust services from unreliable nodes
While coping with extreme loads

Each peer balancing its turn
Systems cleaning up after the churn
Peers putting the network in a squeeze
Joining and leaving as they please

Shrideep Pallickara
Computer Science
Colorado State University

Frequently asked questions from the previous class survey

- Can multiple peers talk simultaneously to the *same* super-put peer?

Topics covered in this lecture

- Peer-to-Peer systems
- Characteristics of P2P systems
- P2P Middleware
- P2P Generations
- Napster

Peer-to-Peer (P2P) Systems

P2P systems

- Supports the construction of large-scale distributed systems
- Data and computational resources are contributed by many hosts
  - All participate in the provisioning of a uniform service

P2P systems

- Ability to share computing resources, storage, and data
  - Present in computers at the "**edges of the internet**"
- Have been used in several applications such as
  - File sharing, web caching, information distribution
  - 10s of thousands of machines harnessed by these applications

Goals

- Demand for Internet Services continues to grow
  - Scope for expanding popular services is limited when all hosts must be owned and managed by a single provider
- P2P systems aim to enable sharing of data and resources at a very large scale
  - They do so by eliminating requirements for separately managed servers and their associated infrastructure

P2P systems provide access to information resources

- Information located on computers throughout a network
- Algorithms for **placement** and **retrieval** of objects are a key aspect of system design

Traditional client-server systems

- Single computer or a cluster of tightly-coupled servers
- Simple decisions relating to the placement of resources
- Scale of service is limited by:
  - Server hardware capacity
  - Network connectivity

However, to be effective the delivered service must be

- Fully **decentralized**
- **Self-organizing**
- **Dynamically balance** storage and processing loads between all participating computers
  - Even as computers join and leave the service

CHARACTERISTICS OF P2P SYSTEMS

P2P characteristics

- Each node **contributes** resources to the system
- Each node may **differ in the quality** of the resource that they contribute
  - But every node has the **same** *functional capabilities* and *responsibilities*
- Correct operation does not depend on the existence of any centrally administered systems
- Can be designed to provide a *limited degree of anonymity* to providers and users of resources

Key issue for the efficient operation of P2P systems

- Choice of algorithm for the **placement of data** across many hosts
- **Subsequent access** to the data in a manner that *balances* workload
  - Ensure availability without adding undue overheads

Coping with volatile resources in P2P systems

- Computers and network connections in P2P systems are owned by different entities
  - A single node can become unavailable at any time
- P2P systems *do not rely* on guaranteed access to individual resources
- They are designed to make probability of *failure to access a copy* of a replicated object arbitrarily small
  - Degree of resistance to tampering by malicious nodes

Realizing the potential of P2P systems

- Emerged when significant number of users had acquired **always-on, broadband** connections
  - Made their desktops suitable for resource sharing

- TIMELINES
  - In the U.S., this occurred around 1999
  - By mid-2004, worldwide number of broadband connections exceeded 100 million


Each generation imagines itself to be more intelligent than the one that went before it, and wiser than the one that comes after it.
George Orwell
P2P GENERATIONS

P2P Generations

- 1st Generation
  - Napster music exchange service
- 2nd Generation
  - Offered greater scalability, anonymity, and fault tolerance
    - Freenet, Gnutella, and BitTorrent

The 3rd Generation of P2P systems

- Emergence of middleware layers for application independent management of distributed resources
- Examples
  - Pastry
  - Chord
  - Tapestry
  - Khademlia

The 3rd Generation of P2P systems

- Designed to place resources (data objects or files) on a set of widely distributed computers
- Routes messages to these resources on behalf of clients
- Clients
  - Do not make decisions about placement of resources
  - Do not hold information about resource whereabouts

Unlike 2nd generation systems, 3rd generation P2P systems …

- Provide **guarantees of delivery** for requests in a *bounded* number of network hops
- Place replicas of resources on hosts in a **structured** manner taking account of their:
  - Volatile availability
  - Variable trust worthiness
  - Requirements for load balancing
  - Locality of information storage and use

3rd Generation P2P systems: Resources are identified by globally unique identifiers (GUIDs)

- Derived as a secure hash from some or all of the resource's state
- Make a resource **self-certifying**
  - Clients receiving a resource can check the validity of the hash
  - Protects it against tampering by untrusted nodes on which it might be stored
  - Requires that states of the resource are immutable
    - Change to the state will result in a different hash value

Use of objects with changing values

- Is much more challenging
- Requires addition of trusted servers to manage sequence of versions
  - Use this to identify the most current version

Availability

- Must avoid situations in which all replicas of an object are simultaneously unavailable
- Use of randomly generated GUIDs (Globally Unique Identifier) assists by distributing object replicas
  - To randomly located nodes
  - If the underlying network spans multiple domains?
    - Risk of simultaneous unavailability is reduced significantly

Our bodies get bigger but our hearts get torn up
We're just a million little gods causing rain storms
Turning every good thing to rust
I guess we'll just have to adjust

With my lightning bolts a-glowing
I can see where I am going to be

Wake Up, Arcade Fire

P2P MIDDLEWARE

P2P middleware is designed to orchestrate

- Automatic **placement** of resources (data items, objects, files, etc.)
- Subsequent location (**discovery**) of distributed resources

How different P2P generations cope with this issue

- 1st Generation
  - Maintain a **centralized index** of available files
  - Files are stored at the peers
- 2nd Generation
  - Systems such a Gnutella & Freenet employ **partitioned distributed indexes**
- 3rd Generation
  - Rely on **Overlays**

Requirements for P2P systems

- Functional
  - Specific behaviors or functions that must be supported
- Non-functional (or evaluation metrics)
  - Criteria that can be used to judge the operation of a system

Functional requirements for P2P middleware

- **Locate and communicate** with any resource made available to the system
  - Even though resources are dispersed over a large number of nodes
- The ability to **add and remove** both resources and nodes at will

Non-functional requirements for P2P systems

- Scalability
- Load balancing
- Dynamic host availability

Non-functional requirements: Load balancing

- Achieved via **random placement** of resources
- **Replicas** of heavily used resources are created

Accommodate highly dynamic host availability

- Host computers are free to join or leave at any time
- Provide a dependable service, from unreliable nodes
- As nodes join the system
  - Must be integrated into the system
  - Load must be redistributed to exploit their capabilities
- As nodes leave the system (voluntarily or involuntarily)?
  - Redistribute their load and resources
    - Replication levels for some resources must be preserved

Systems that we will look at

Systems that we will observe closely

- 1st Generation
  - Napster
- 3rd Generation
  - Pastry
  - Chord
  - Tapestry
- Unstructured P2P or 2nd Generation
  - Gnutella and BitTorrent

Sleep thoughts
Are spreading
Throughout the whole land.
The time for nigh-brushing of teeth is at hand
The Sleep Book, Dr. Seuss.

NAPSTER

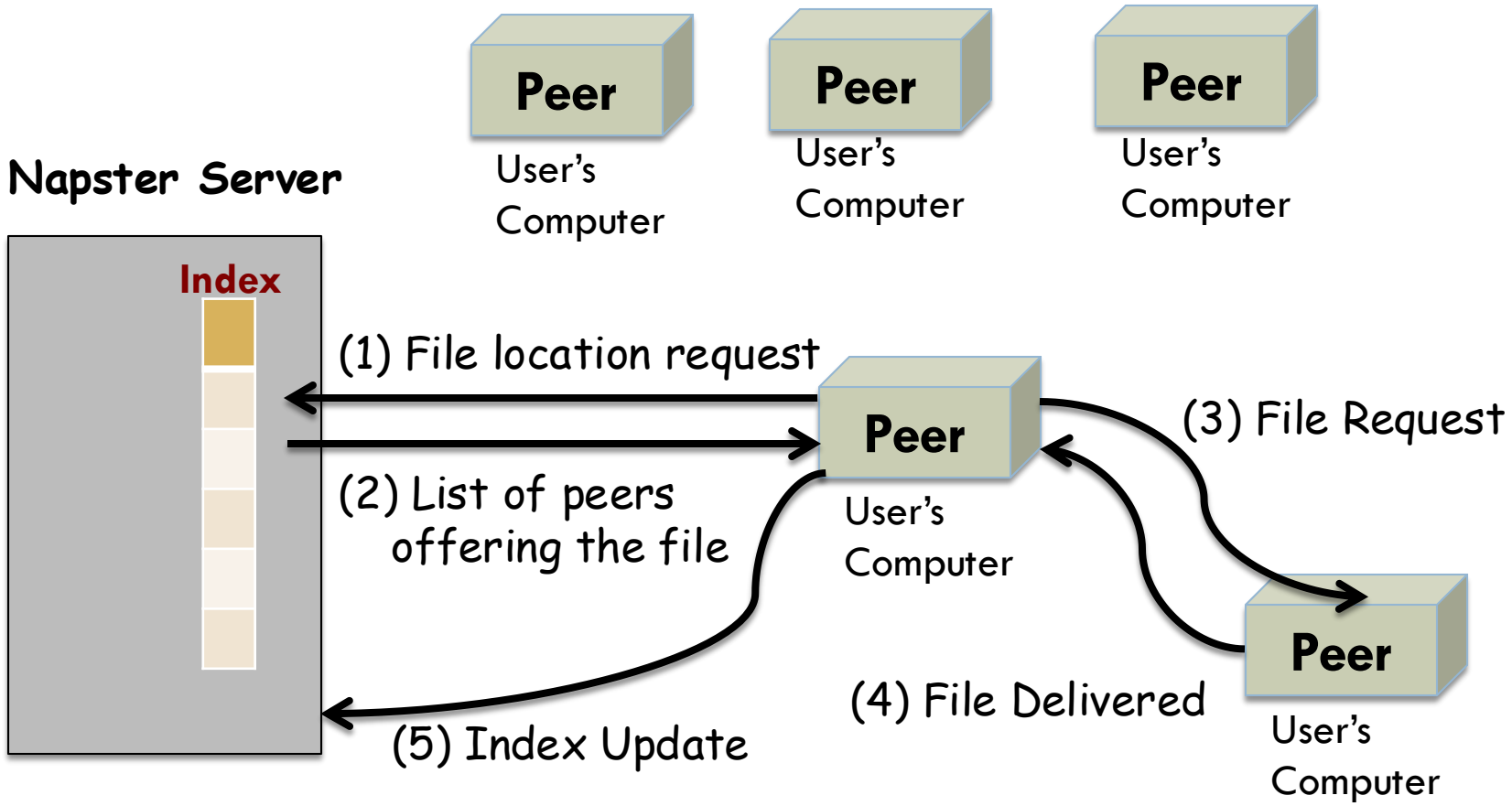
Napster

- First application in which demand for massively scalable storage and retrieval arose
  - Downloading of digital music files
- Became very popular soon after its launch
- At its peak
  - **Several million** users
  - Thousands swapped music files *simultaneously*

Key features of the architecture

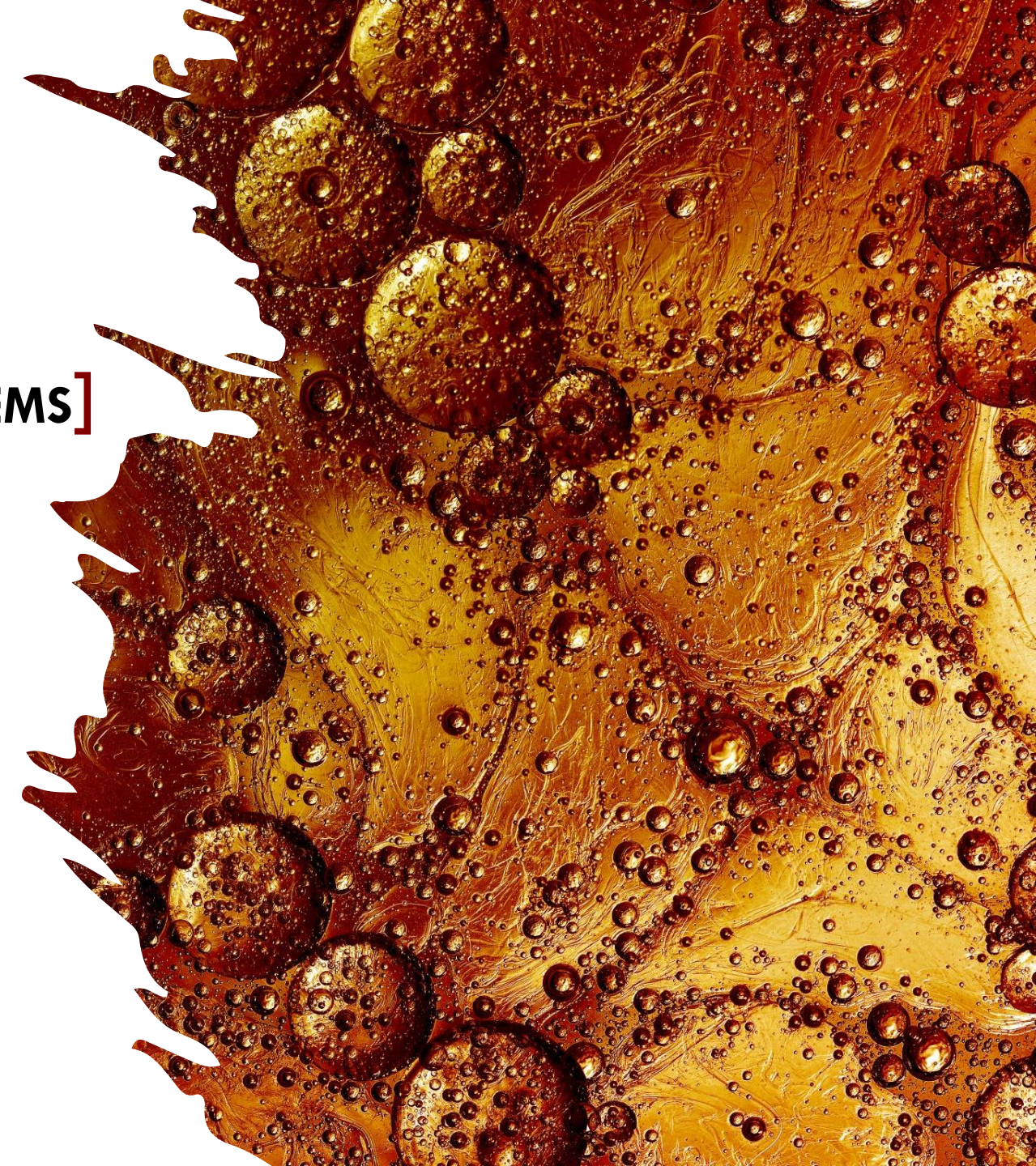
- Centralized indexes
- Users supplied the files
  - Stored and accessed on their personal computers
- Clients add their own music files to the pool of shared resources
  - Transmit a link to Napster's indexing service for each available file
  - Shared resources at the "*edge of the internet*"

Napster Architecture



OVERLAYS

[USED IN 3RD GENERATION P2P SYSTEMS]



Overlays

- **A distributed algorithm** takes the responsibility of locating nodes and objects
  - This is the *routing overlay*
- Denotes that the **middleware** is a layer that is responsible for routing requests
  - From a client to the host that holds the requested object

But why call it an overlay?

- Denotes that it implements a *routing mechanism* in the **application layer**
  - This is different from routing mechanisms deployed at the network level, e.g., IP
- A logical hop in the routing overlay, encompasses multiple underlying router hops

What does the routing overlay do?

- Ensures any node can access any object
- Routes requests through a **sequence** of nodes
  - Exploits (local) knowledge at each of the intermediate nodes to locate the destination object
- If there are multiple replicas of objects?
  - Overlay maintains knowledge of all available replicas, and then delivers request to the *nearest* "live" node

Overlay vs IP Routing

Oh, let the sun beat down upon my face, with stars to fill my dream.
I am a traveler of both time and space to be where I have been.

Kashmir; Robert Plant, John Bonham, Jimmy Page; Led Zeppelin

Overlay Routing vs. IP Routing

- There are several similarities between the two
- Why have a separate mechanism?
  - The legacy nature of IP
  - The legacy's impact is too strong for it to be overcome
    - Hard to support P2P applications directly

IP Routing vs. Overlay routing: Scale

- IP
  - IPv4 is limited to 2^{32} nodes
  - IPv6 is 2^{128}
  - But addresses are **hierarchical**ly structured
    - Much of the space is pre-allocated to meet administrative requirements
- Overlay
  - Globally Unique Identifiers (GUID) namespace is very large (2^{128} or 2^{160})
  - The namespace is also **flat** allowing for it to be much more fully occupied

IP Routing vs. Overlay routing: Load Balancing

- IP
  - Loads are determined by network topology and associated network patterns
- Overlays
  - Object locations can be randomized, so …
  - Traffic patterns can be divorced from the network topology

IP Routing vs. Overlay routing: Network dynamics

- IP
  - Routing tables are *updated asynchronously* on a best-effort basis
  - Typically, on the order of an hour
- Overlays
  - Can be updated synchronously or asynchronously
  - Fractions of seconds

IP Routing vs. Overlay routing: Fault tolerance

- IP
  - Redundancy provided by network managers
    - To handle router or network connectivity failure
  - N-fold replication is costly

- Overlay
  - Routes and object references can be replicated n-fold
    - Tolerance of (n -1) failures of nodes or connections

IP Routing vs. Overlay routing: Target identification

- IP
  - Each IP address maps to exactly one node
- Overlay
  - Message can be routed to the nearest replica of a target object

Main task of a routing overlay

1. Routing of requests to objects
2. Insertion of objects
3. Deletion of objects
4. Node additions and removals

Calculation of Globally Unique Identifiers (GUIDs)

- This is computed from all or part of the state of the object
- Function delivers a value that is, with a *very high probability*, unique
  - One way hash functions, such as SHA-1 or MD5 are often used

Why are overlay systems also called Distributed Hash Tables (DHTs)?

- Randomly distributed identifiers are used to determine resource
  - Placements
  - Retrievals

In the DHT model, a data item with GUID $\mathbf{X}$

- Is stored at the node whose GUID is ***numerically close*** to X
- If the replication factor is r?
  - Then it is stored at the r hosts whose GUIDs are next-closest to it numerically

A quick tour of how different P2P systems solve this

- Prefix routing
- Exploiting distance measures

Prefix routing

- Routes for delivery of messages based on values of GUIDs to which they are addressed
- Narrow search for the next node along the route by applying a **binary mask**
  - Selects an increasing number of hexadecimal digits from the destination GUID after each hop
- Used in Pastry and Tapestry

Exploiting different measures of distance to narrow search for next hop destination

- Chord
  - Numerical difference between GUIDs of the selected node and the destination node
- CAN
  - Uses distance in a d-dimensional hyperspace into which nodes are placed
- Kadmelia
  - Uses XOR of pairs of GUIDs as a metric for distance between nodes

A final note about GUIDs

- These are not human readable
- Client applications must obtain GUIDs for resources of interest through some indexing service
  - Human readable names or search requests
- For e.g., BitTorrent
  - Web index search leads to a sub file containing details of desired resource
    - GUID
    - URL of tracker: Host that holds up to date list of network providers willing to supply the file

The contents of this slide-set are based on the following references

- *Distributed Systems: Principles and Paradigms. Andrew S. Tanenbaum and Maarten Van der Steen. 2nd Edition. Prentice Hall. ISBN: 0132392275/978-0132392273.* [Chapter 5]
- *Distributed Systems: Concepts and Design. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. 5th Edition. Addison Wesley. ISBN: 978-0132143011.* [Chapter 10]
- *http://en.wikipedia.org/wiki/Non-functional_requirement*