CSx55: Distributed Systems [Peer-to-Peer Systems]

Overlays

Traffic patterns divorced from
network topology

Breaking out of
hierarchical addressing shackles
With spaces that are flat
and more to go around

To route is to get closer
To what to you seek

Shrideep Pallickara
Computer Science
Colorado State University

Frequently asked questions from the previous class survey

- *Where* and *who* has access to the overlay?
- How can P2P systems have the same functionality if their quality differs?
- Info about log N nodes to route in log N hops? Really???
- Quiz-5 everyone gets a +1 without exceeding 10 i.e., no one gets a 11/10
  - We will have ≥15 quizzes, but will only take your top-10 scores
  - Embrace the low-stakes quizzes: they are the practice that makes the high-stakes exams go well

Some Upcoming Dates

- Term project pitches
  - Tuesday (October 14th) and Wednesday (October 15th)
- Midterm exam
  - Thursday (October 16th)
- Final Exam
  - On-campus students (Monday, December 15th from 2:00-4:00 pm)
  - On-line sections (Due on Sunday, December 14th @ 11:59 pm MT)
- Curving the midterm and final exam
  - Among the students who take the exam, the target average is 80%

Topics covered in this lecture

- 3^{rd} generation P2P Systems

In the DHT model, a data item with GUID 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

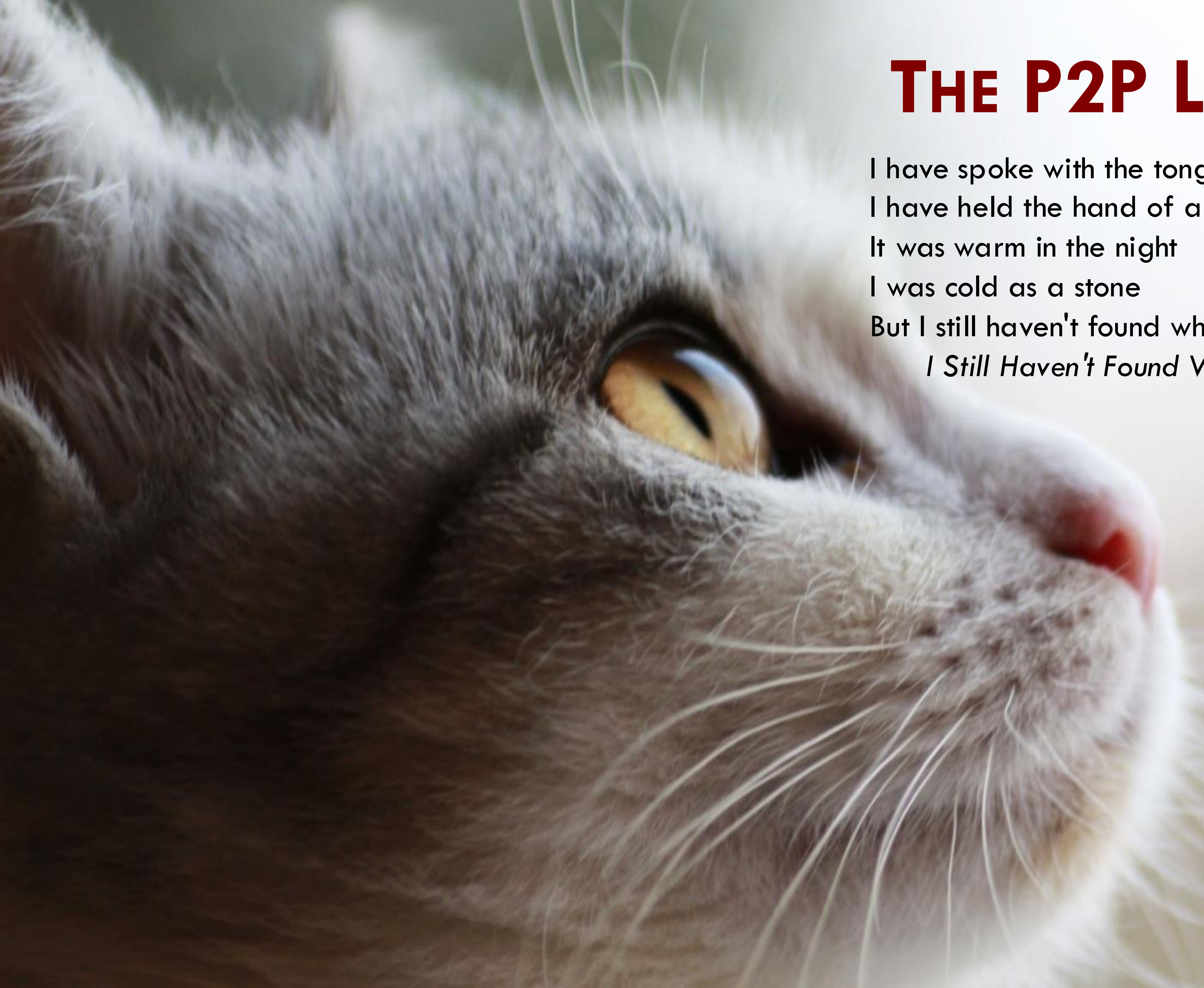
A final note about GUIDs

- 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 P2P Lookup Problem

I have spoke with the tongue of angels
I have held the hand of a devil
It was warm in the night
I was cold as a stone
But I still haven't found what I'm looking for

I Still Haven't Found What I'm Looking For, The Joshua Tree, U2



The peer-to-peer (P2P) lookup problem

- How do you **find** a data item in a large collection of peers?
- Lookup must be scalable and decentralized
  - Without hierarchy

The lookup problem: Centralized Approach

- Maintain central database
- Maintain table that maps file name to server(s) that holds content
  - NAPSTER
- Problems
  - Reliability — Single point of failure
  - Scalability — Database bottleneck for all requests
  - Vulnerability — Targeted denial of service attacks

The lookup problem: Broadcast

- **Flood** the network with requests looking for **X**
- When a node receives the request:
  - Check local repository
  - If it has **X**, node responds back with a message
- Scaling problems
  - ALL discovery requests sent to ALL nodes
  - ALL nodes process **every** discovery request

Broadcast costs can be reduced by organizing nodes into a hierarchy

- Searches start at the top
  - Traverse single path to the node that holds the desired data
- Directed traversal more frugal than broadcast
- Problems
  - Nodes at the **top** of the tree take **larger fraction** of load than leaf nodes
  - Requires expensive hardware
    - Loss of tree root (or node close to it) catastrophic

Distributed hash tables

- Few constraints on the structure of the keys

- REQUIREMENTS
  - Data identified using numeric **keys**
  - Nodes must be willing to store keys **for each other**

Storage and retrieval in distributed hash tables

- Data items are *inserted* and *found* by specifying a unique **key** for the data
- Underlying algorithm must determine ***which node*** is responsible for storing the data

Distributed Storage using DHTs: Publishing a file

- **Convert** file-name to numeric key
  - Using one-way hash functions like MD5 or SHA-1
- Call **`lookup(key)`**
  - Returns IP address of node responsible for key
- **Send file** to be stored at node returned by `lookup`

Distributed Storage using DHTs: Retrieving a file

① Obtain name of file

② Convert it to a key using one-way hash function

③ Call `lookup(key)`

④ Ask resulting node, from (3), for a copy of the file

IMPLEMENTING DHTs

Implementing DHTs: 3 core elements

- **Mapping** keys to nodes
- **Forwarding** a lookup for a key to the appropriate node
- Building **routing tables**

Implementing DHTs:
Mapping keys to nodes

- Must be load balanced
- Done using one-way hash functions
  - MD5 (128-bit) or SHA-1 (160-bit)
  - For e.g., lets look at hashing "Joshua Tree"
    - MD5: 69a20f4a82140b02b877ecad6a54881f
    - SHA-1: 3caf7b915c691763b2cb1ac9b74dcc788b33e249
- Ensures that content is distributed **uniformly**

Let's see how these hash values change with minor changes

String	MD5 [128-bits] 32 hex characters	SHA-1 [160-bits] 40 hex characters
Joshua Tree	69a20f4a82140b02b877ecad6a54881f	3caf7b915c691763b2cb1ac9b74dcc788b33e249
Joshua Tree1	f096b3623bc615555fe130c0686bd8d6	f1788baa923adcdf72a6b6a0e5bd36abd14ec69d
Joshua Tree *{N.B: I have put an extra space between "Joshua" and "Tree"}*	195cce3454aa54066c9f9a974bd8a04b	86a3dacb5e72171054f2cc6016065bbded48e82e
Joshua Tree [U2, 1987]	2515d32756cccb5b0d2d8a57e4683f2f	d40b072c58c264bc67982538db4e64666dcfbabf
The Joshua Tree [U2, 1987]	56fb7d6d3361a80661426add12e17b16	6f579cdc67a601b8ecf57b3d87f1ac69ac3b2a63

Implementing DHTs
Forwarding lookups

- Any node that receives query for key
  - Must forward it to a node whose ID is ***closer*** to the key
- Above rule guarantees that query ***eventually arrives*** at the closest node
- For e.g.:
  - Node has ID 346, and key has ID 542
  - Forwarding to node 495 gets it numerically closer

Implementing DHTs:
Building routing tables

- Multiple nodes participate in locating content
- Each node must know about **some other** nodes
  - To forward lookup requests
  - SUCCESSOR
    - The node with the ***closest succeeding*** ID
  - Other nodes
    - For efficiency in routing

Distributed hash tables: Identifiers

- Data items are assigned an identifier from a large random space
  - 128-bit UUIDs or 160-bit SHA1 digests
- **Nodes are also assigned a number from the same identifier space**

Crux of the DHT problem

- Implement an efficient, **deterministic** scheme to
  - Map data items to node
- When you **look up** a data item
  - Network address of node holding the data is returned

PASTRY

Pastry

- All nodes and objects are assigned 128-bit GUIDs
- Applies secure hash function to:
  - The public-key assigned to each node → Node GUID
  - The object's name or some part of the object's stored state

Resulting GUIDs have usual properties of secure hash values

- They are **randomly distributed** in the range $0 - (2^{128} - 1)$
- Provide no clue about the values from which they were computed
- **Collisions** in the GUID space (for nodes and objects) are *extremely unlikely*

The Pastry routing

- The number of nodes in the network, N
- The algorithm will correctly route messages addressed to any GUID in O(log N) steps
  - Delivered to an active node whose GUID is *numerically closest* to it
- **Active nodes** take responsibility for processing requests addressed to all objects in their *numerical neighborhood*

Pastry routing

- Routing transfers message to a node that is **closer** to its destination
- Closeness is in an *artificial* space
  - The space of GUIDs

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]