



Fall 2009


Dr. Kevin Hamlen



# CS 4485: CS PROJECT



# The Project

- build an advanced P2P file-sharing client
  - main features
    - keyword search
    - P2P file download
    - graphical user interface
    - automated test suite
    - user manual, technical manual
  - project organization
    - teams of 3-4 members each
    - you choose the programming language(s), development methodology, etc.
    - the customer (me) chooses the specs & deadlines
- 

# Grading


- All members of each team receive the same grade
- Grading breakdown
  - Stage 1 report & demo [due 9/18]: 10%
  - Stage 2 report & demo [due 10/16]: 10%
  - Stage 3 report & demo [due 11/13]: 10%
  - Stage 4 report & demo [ due 12/11]: 10%
  - Final Submission [due 12/18]
    - Low-level network implementation: 20%
    - GUI & user manual: 10%
    - Automated Test Suite & Technical Manual: 20%
    - Extra Feature(s) (TBA): 10%
- No exams, no homework assignments

# Teams

- You choose your teams
  - use class time today
  - use WebCT discussion board for this class
- Must be 3-4 members (no exceptions)
- By the end of the day next Friday, each team should email me:
  - the names and email addresses of all members
  - a 1-page plan of work (to be discussed today)



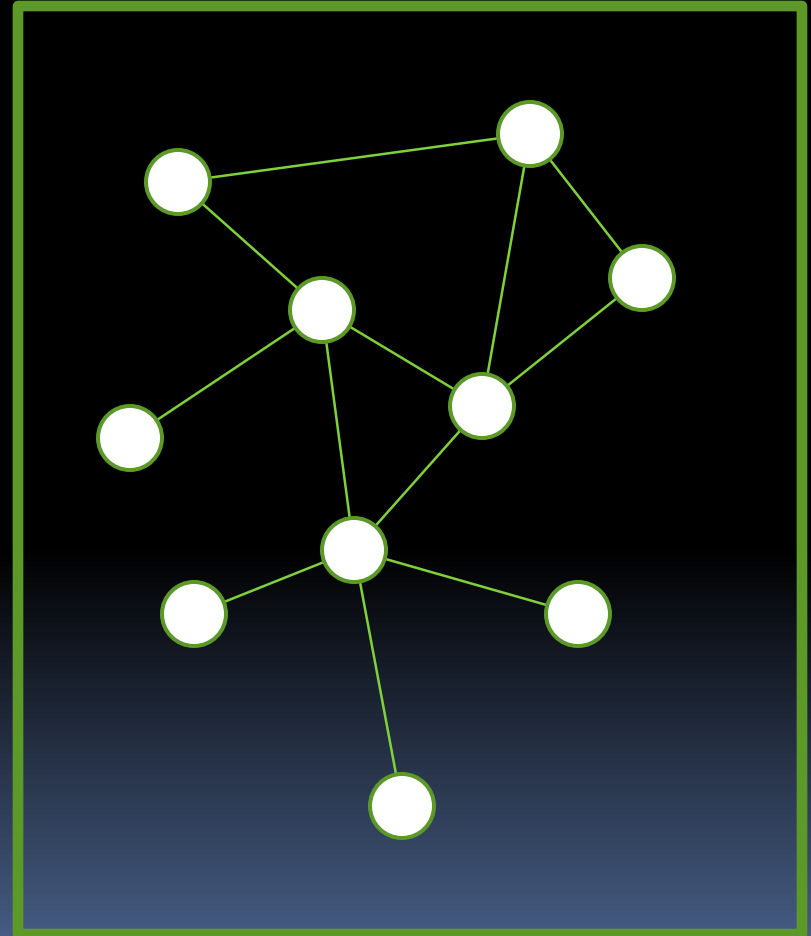
# Lectures

- I will lecture as needed
    - background info on P2P networks
    - high-level design issues
    - optional features (security)
  - Other class times will be used for...
    - weekly group meetings
    - software demo's
- 

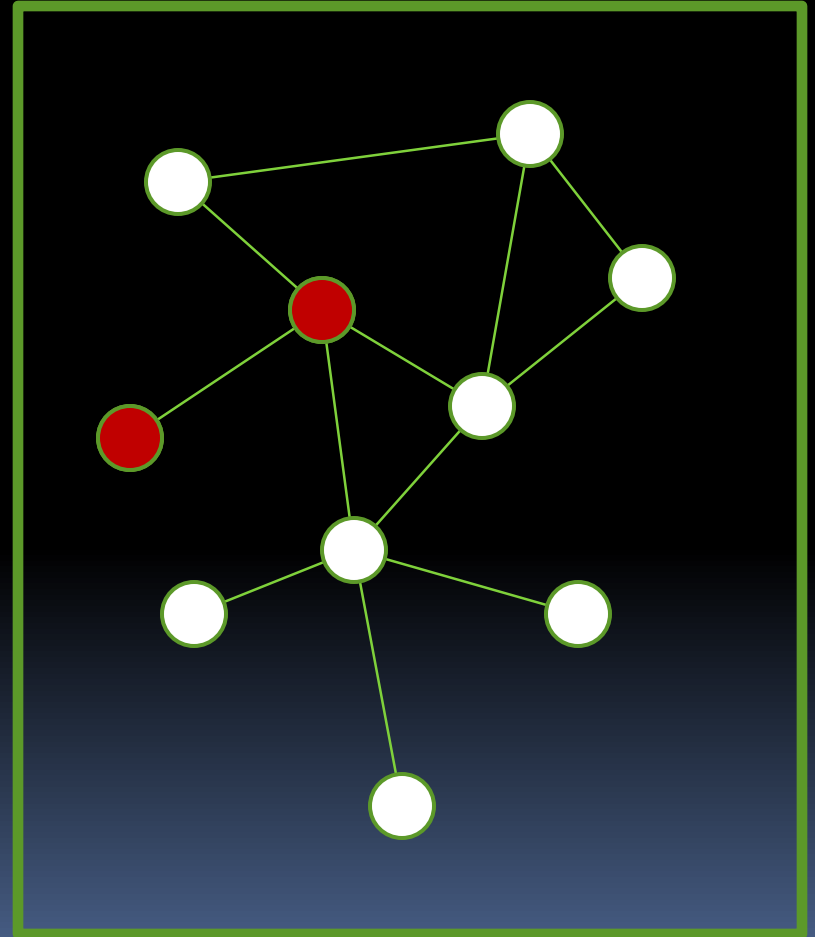
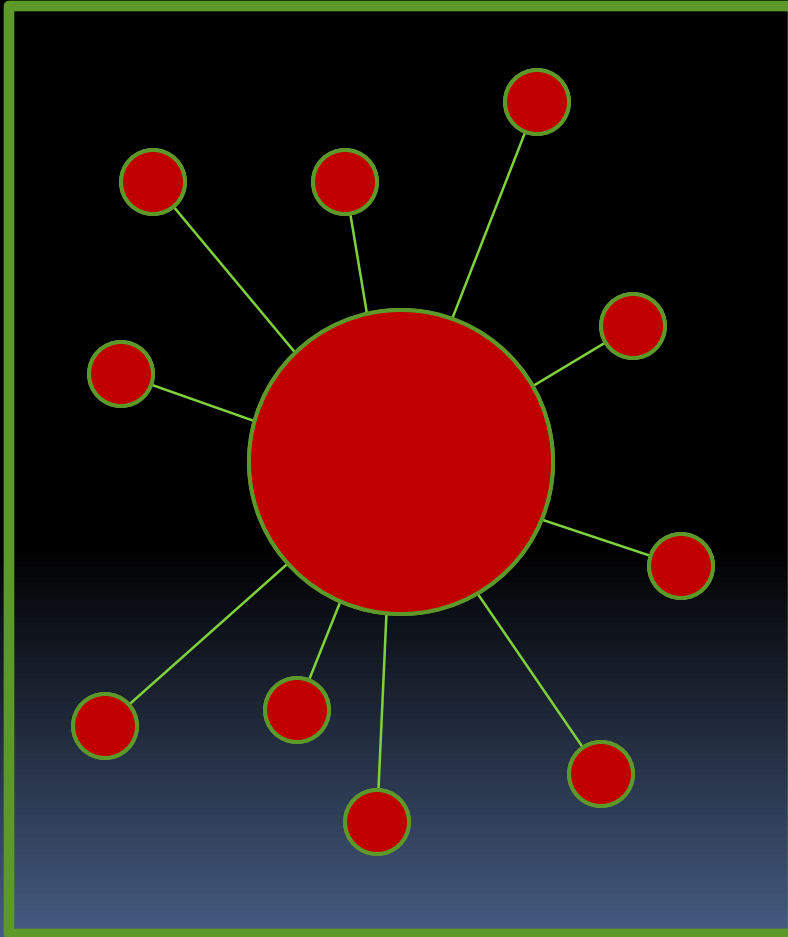


# WHIRLWIND OVERVIEW OF PEER-TO-PEER NETWORKS

# Centralized and Decentralized Networks

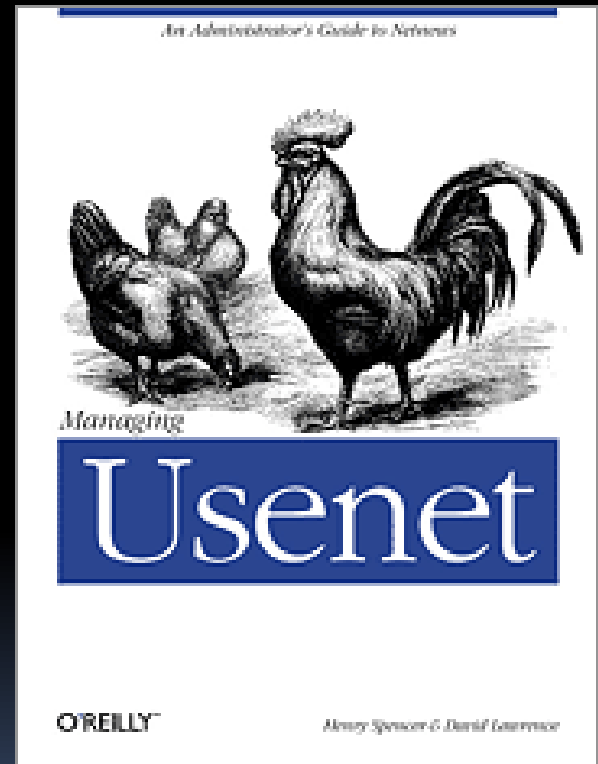


# Centralized and Decentralized Networks



# Usenet

- First widely used P2P Network
  - Truscott and Ellis [1979]
  - implemented at Duke and UNC
- Actually a hybrid structure
  - news servers form P2P network
  - clients access local servers
- Built for a slow (dial-up!) net
  - massive data replication
  - multicast protocol
  - sender-initiated transfers
- Still in widespread use today!

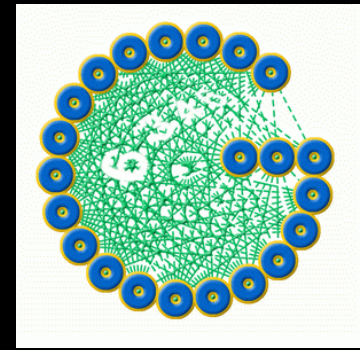


# Napster



- First commercially successful P2P Network
  - Fanning [1999]
  - digital music exchange
  - revenue via advertising
- Not really a P2P Network
  - queries go to central servers
  - servers respond with object location
  - actual download is P2P
- The Controversy
  - users share mostly pirated music
  - RIAA sues Napster for copyright infringement [2000]
  - U.S. 9<sup>th</sup> circuit rules against Napster [2001]
  - Napster reinvents itself as a standard music download site

# Gnutella

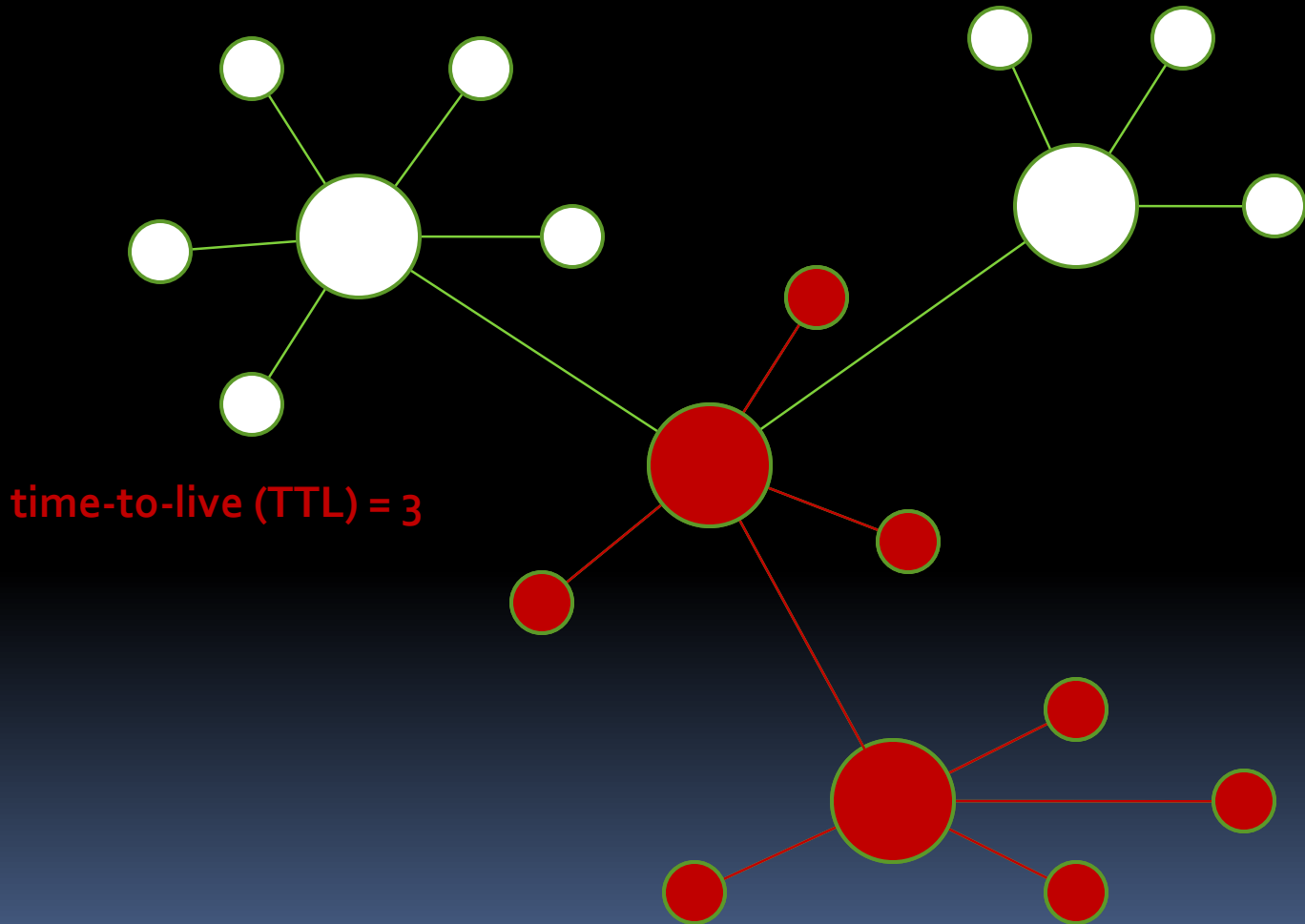


- First true P2P network
  - Frankel and Pepper [2000]
  - released by Nullsoft 3/14, retracted 3/15
  - reverse engineered within days
  - numerous home-made clients released
  - activity grows to millions of nodes by 2001
- Pure P2P network topology
  - fully decentralized
  - unstructured topology
  - multicast (flooding) query protocol
  - some issues raised about scalability
- Other Gnutella-like hybrids: Grokster, Kazaa, Limewire, Bearshare, etc.

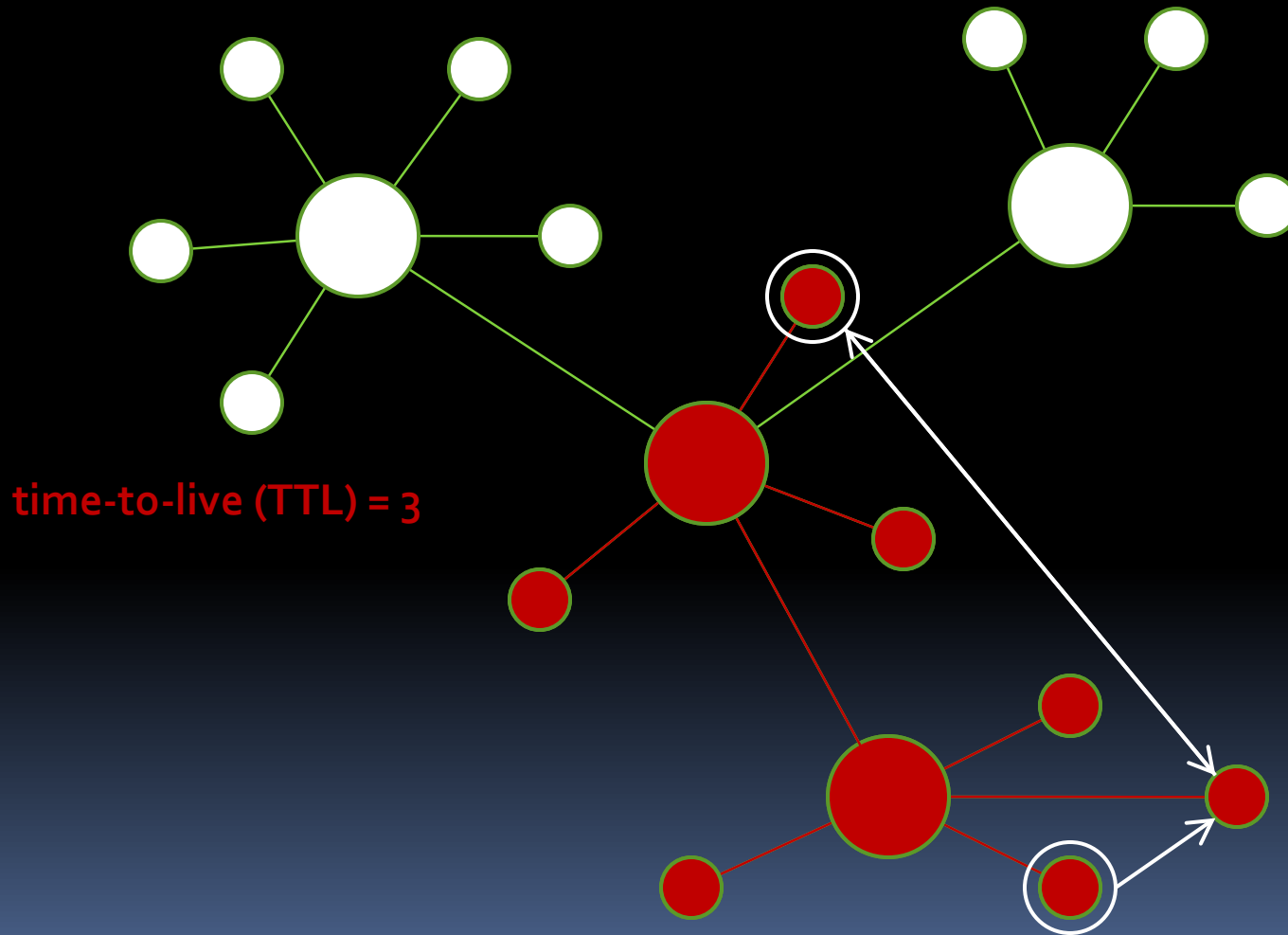
# Gnutella Basics: Peer Join



# Gnutella Basics: Querying



# Gnutella Basics: Query Hits




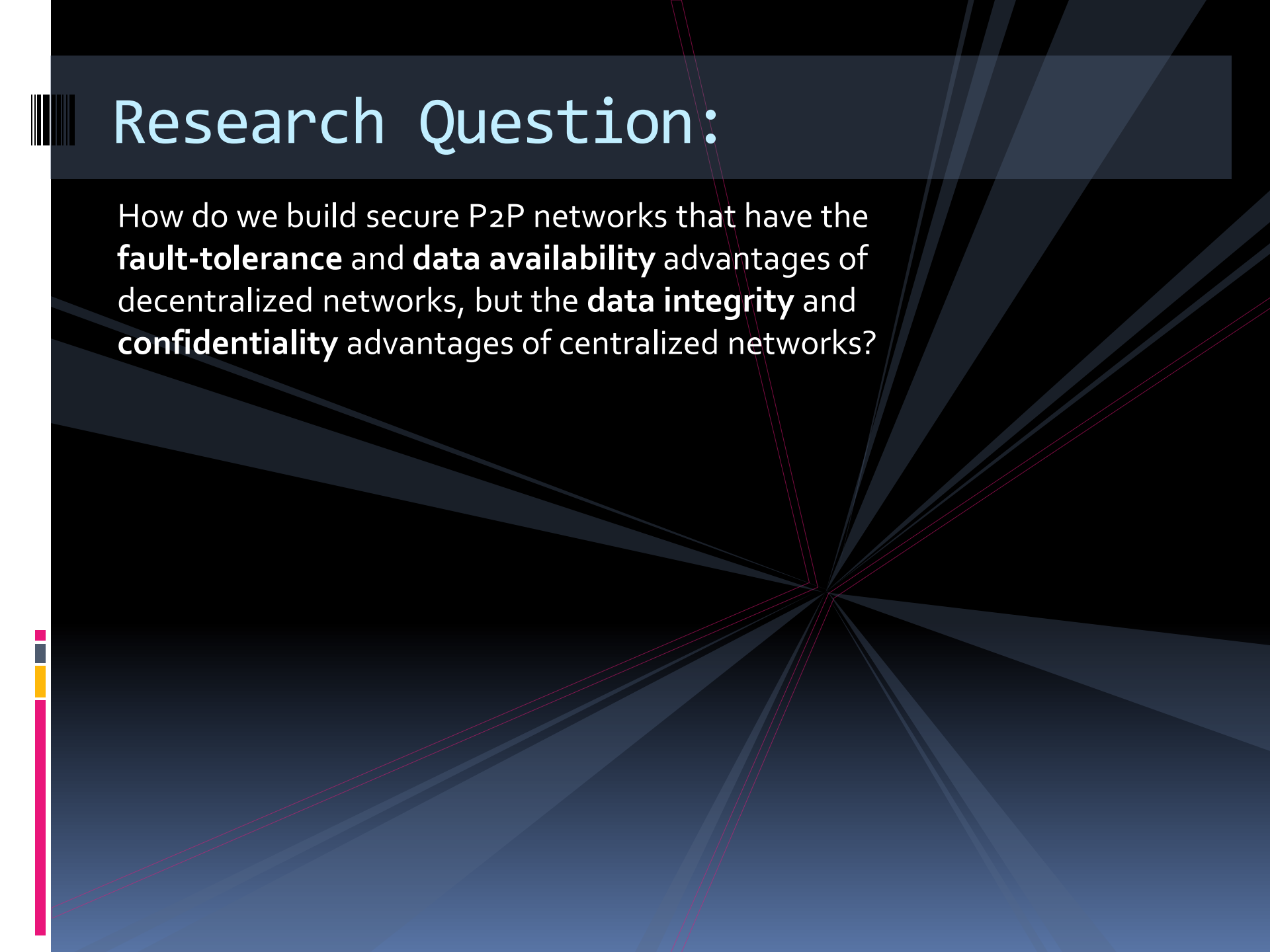
# Gnutella Shortcomings

- Scalability Issues
  - multicast (flooding) is bandwidth-intensive!
  - query hits are not convergent
- Attacks
  - false query hits
  - spoofing
  - free-riding
  - Sybil attacks [Douceur, IPTPS 2002]
  - confidentiality violations
  - denial-of-service (DoS) attacks
    - message-dropping attacks
    - selfish routing



# Research Question:

How do we build secure P2P networks that have the **fault-tolerance** and **data availability** advantages of decentralized networks, but the **data integrity** and **confidentiality** advantages of centralized networks?





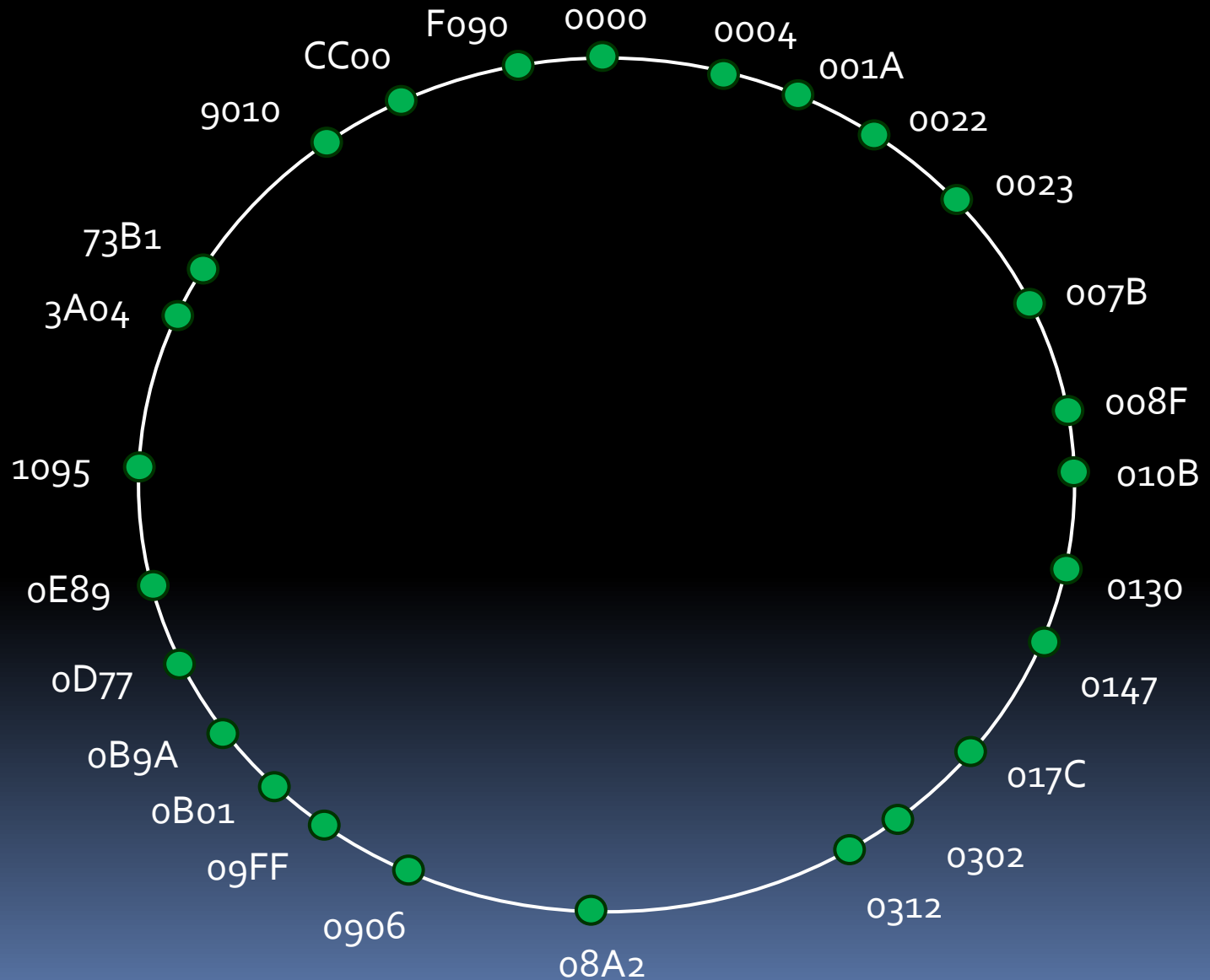
# Structured P2P Networks

- Solution: Add more structure to the overlay topology
- Chord [Stoica, Morris, Karger, Kaashoek and Balakrishnan, SIGCOMM 2001]
- Other topologies
  - CAN [Ratnasamy, Francis, Handley, Karp and Schenker, SIGCOMM 2001]
  - Pastry [Rowstron and Druschel, Middleware 2001]
  - Tapestry [Zhao, Huang, Stribling, Rhea, Joseph and Kubiawicz, JSAC 2004]

# Chord Overlay Structure

- Every client gets a unique number
  - called the client's "identifier"
  - computed by applying a **secure hash function** to the client's IP number & port number
  - Example:  $H("127.0.0.1:10035") = \text{0xF382E26A}$
- secure hash functions
  - map values from a large domain (e.g., strings) to a smaller domain (e.g., 128-bit integers)
  - one-way functions (hard to reverse)
  - collisions very rare (astronomically improbable)
  - NEVER TRY TO INVENT YOUR OWN!

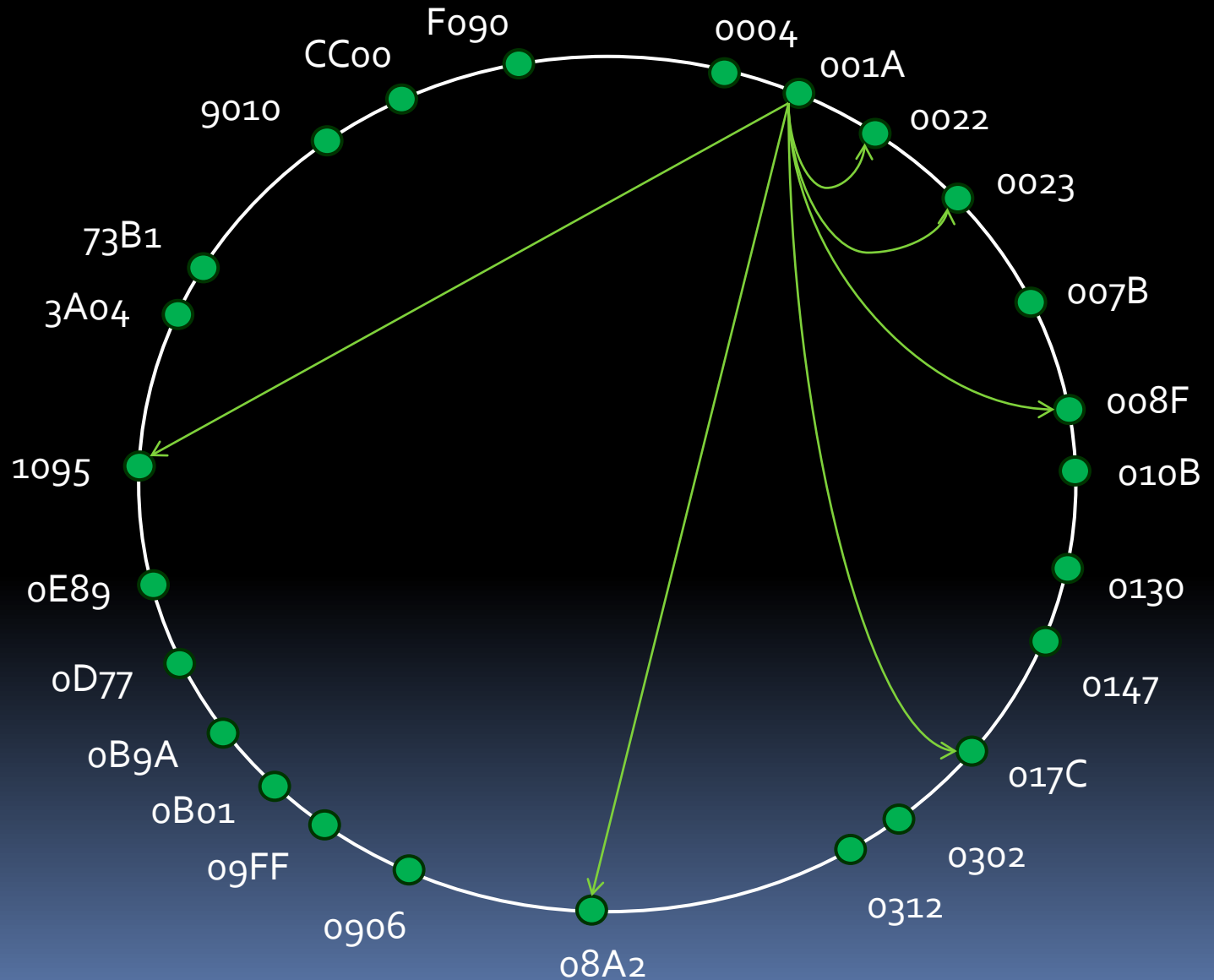
# Identifiers form a Ring



# Chord Finger Tables

- Each peer keeps track of a table of 128 other peers
  - Let  $d$  be my identifier
  - The  $n$ th entry in my table ( $0 \leq n \leq 127$ ) is the IP and port number of the peer whose identifier is closest to but no less than  $(d + 2^n) \bmod 2^{128}$
  - So the first entry is the next peer in the ring (my successor)
  - Each peer in the table is about twice as far away from me as the previous one in the table
  - Special case: If no peer has an identifier less than  $(d + 2^n) \bmod 2^{128}$ , then use the peer with greatest identifier

# Example

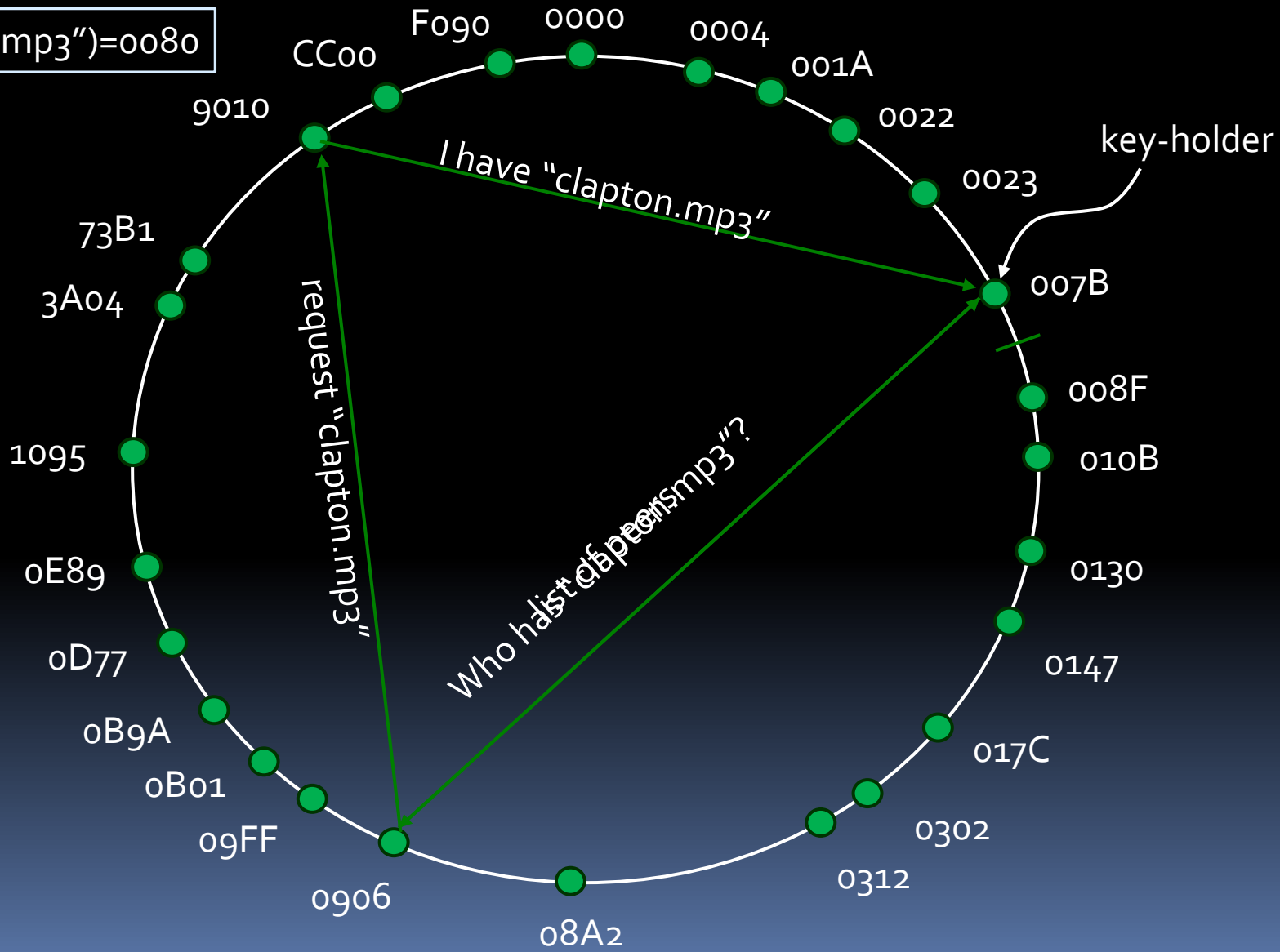


# Message Send/Receive

- Each peer can send a message to the peer whose identifier is closest to but no less than  $x$ .
  - Choose the peer in my finger table whose identifier is closest to but no greater than  $x$ .
  - Send the message to that peer and ask him to forward it to  $x$ .
  - Process continues until the desired peer is reached.
  - Guaranteed to take no more than  $O(\log_2 n)$  hops, where  $n$  is the size of the identifier space (e.g.,  $2^{128}$ ).

# Filesharing in Chord

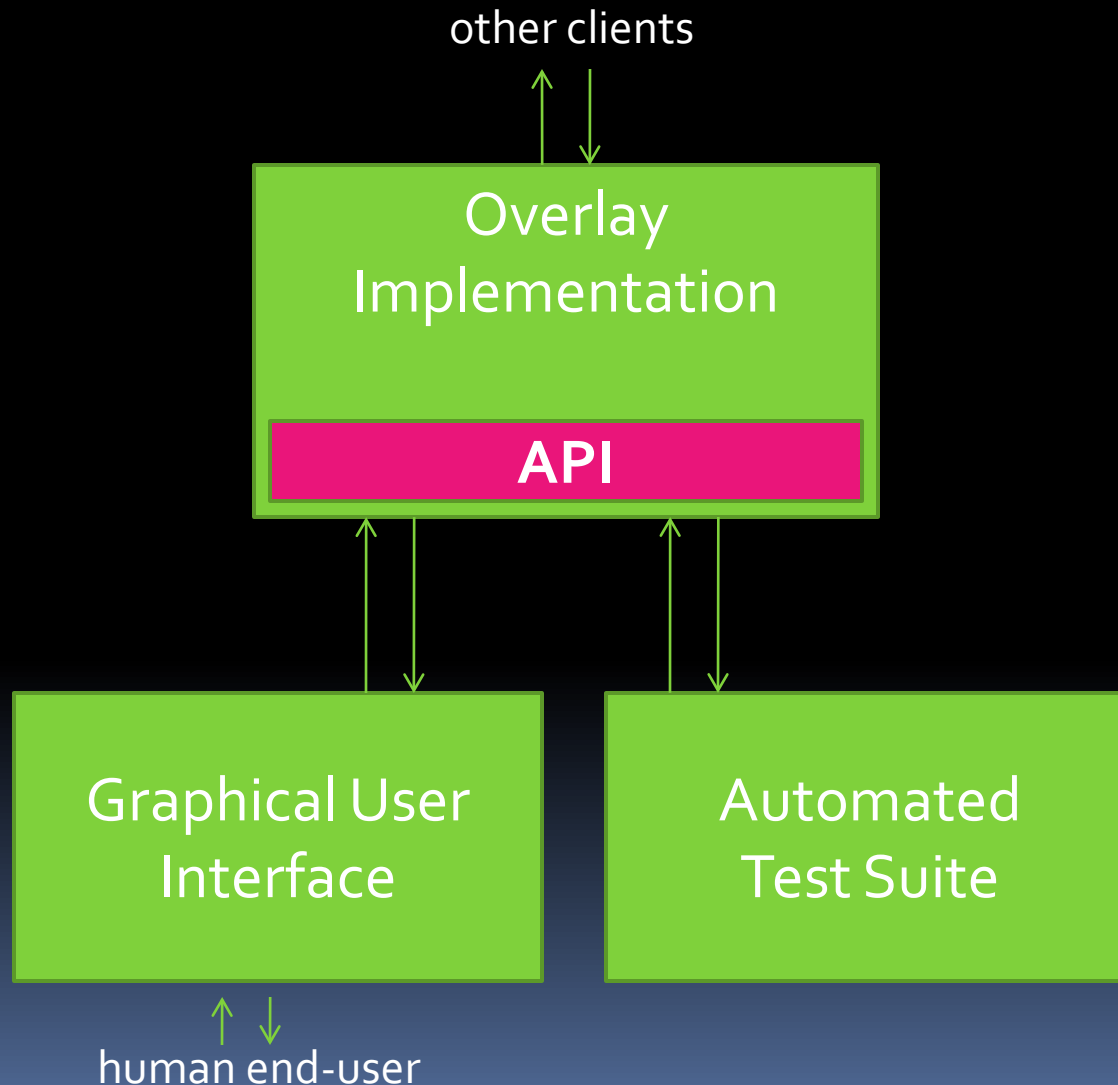
`h("clapton.mp3")=0080`



# Stage 1 Objectives (due September 18<sup>th</sup>)

- Implement a basic P2P overlay in the language of your choice
  - list of all peers fixed at start time (no join/leave)
  - each peer can send a message (e.g., a string) to the peer whose identifier is closest to but no greater than x
- Two interfaces supported
  - a graphical user interface (human user)
  - an API (to use the client as a service)
- Implement a test suite
  - uses the API to make clients on the same machine (listening on different ports) send messages
  - tests to see if messages were correctly received and no spurious messages received

# High-level Structure



# Automated Testing

- Test suite drives MULTIPLE clients on the same machine
  - (the clients listen on different ports)
- It instructs clients to send messages
  - pseudo-random (but choose the seed value deterministically so test can be repeated!)
  - messages sent at random times (don't wait for each to be delivered to send the next)
  - test that each message was received by the correct client, and no spurious messages received

# Objectives for Next Week

- Form teams of 3-4 people each
  - use remainder of class time to organize
  - use class discussion board on eLearning if desired
- Choose programming language(s)
  - suggestion: Java or C#
- Set up a version-control system
  - see links on course website
- Decide on a tentative allocation of work amongst team members
  - Example: X is in charge of writing the low-level overlay code, Y is in charge of the GUI, and Z is in charge of the testing suite



# Next Class

You should meet as a group (here)

- Email me the following by the end of the day on 8/28:
  - the names and email addresses of all team members
  - a 1-page description of your work plan, including language(s) chosen, IDE and version-control systems chosen, and allocation of work between members