CSCI-351
Data communication and Networks

Lecture 17: Peer-to-Peer System and BitTorrent

(I swear I only use it for Linux ISOs)

The slide is built with the help of Prof. Alan Mislove, Christo Wilson, and David Choffnes's class

Outline

- Peer-to-Peer Overview
- Example: Bittorrent

Traditional Internet Services Model

- Client-server
  - Many clients, 1 (or more) server(s)
  - Web servers, DNS, file downloads, video streaming
- Problems
  - Scalability: how many users can a server support?
    - What happens when user traffic overload servers?
    - Limited resources (bandwidth, CPU, storage)
  - Reliability: if # of servers is small, what happens when they break, fail, get disconnected, are mismanaged by humans?
  - Efficiency: if your users are spread across the entire globe, how do you make sure you answer their requests quickly?

The Alternative: Peer-to-Peer

- A simple idea
  - Users bring their own resources to the table
  - A cooperative model: clients = peers = servers
- The benefits
  - Scalability: # of "servers" grows with users
    - BYOR: bring your own resources (storage, CPU, B/W)
  - Reliability: load spread across many peers
    - Probability of them all failing is **very** low…
  - Efficiency: peers are distributed
    - Peers can try and get service from nearby peers

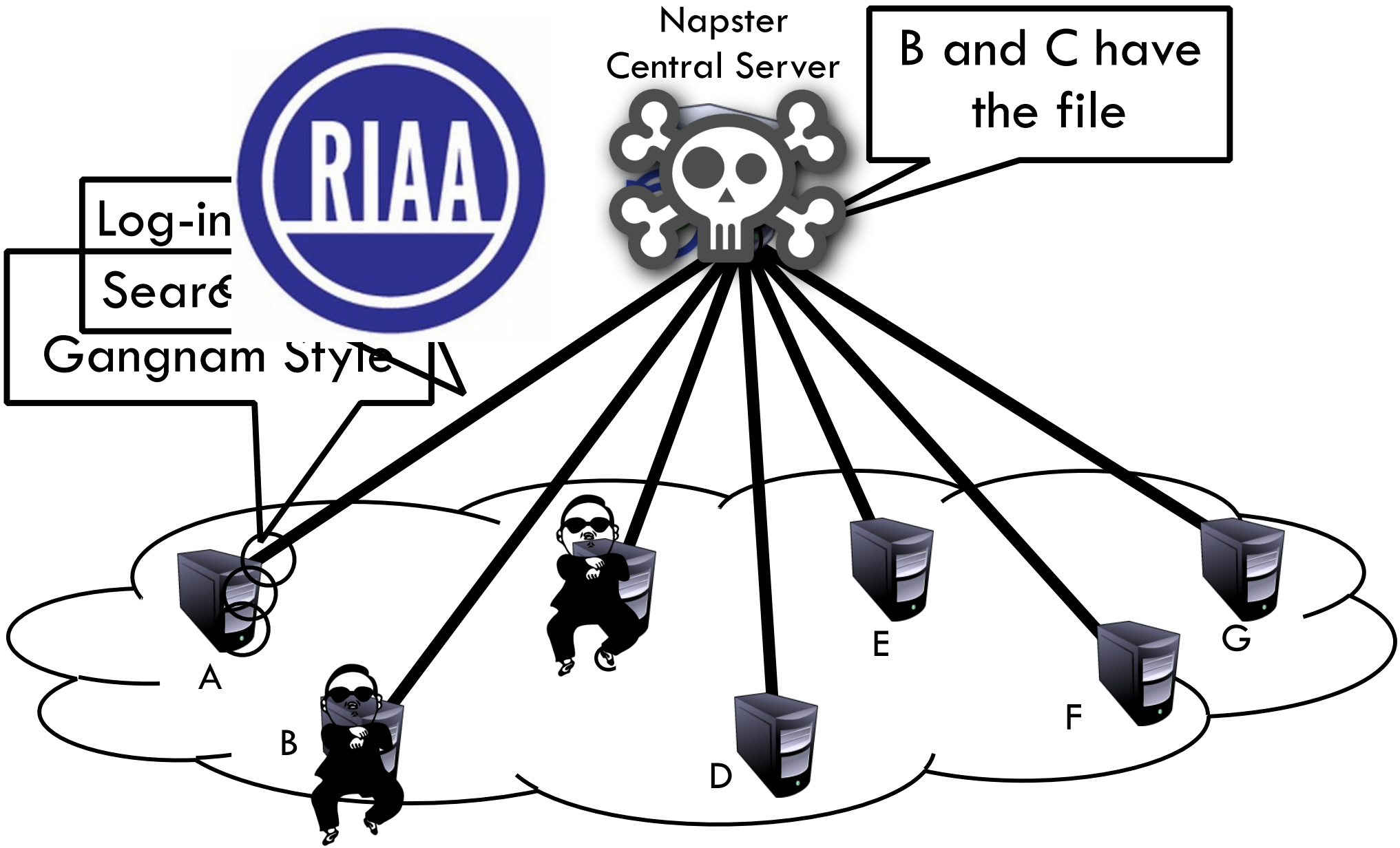
The Peer-to-Peer Challenge

- What are the key components for leveraging P2P?
  - Communication: how do peers talk to each other
  - Service/data location: how do peers know who to talk to
- New reliability challenges
  - Network reachability, i.e. dealing with NATs
  - Dealing with churn, i.e. short peer uptimes
- What about security?
  - Malicious peers and cheating
  - The Sybil attack

Centralized Approach

- The original: Napster
  - 1999-2001
  - Shawn Fanning, Sean Parker
  - Specialized in MP3s (but not for long)
- Centralized index server(s)
  - Supported all queries
- What caused its downfall?
  - Not scalable
  - Centralization of liability

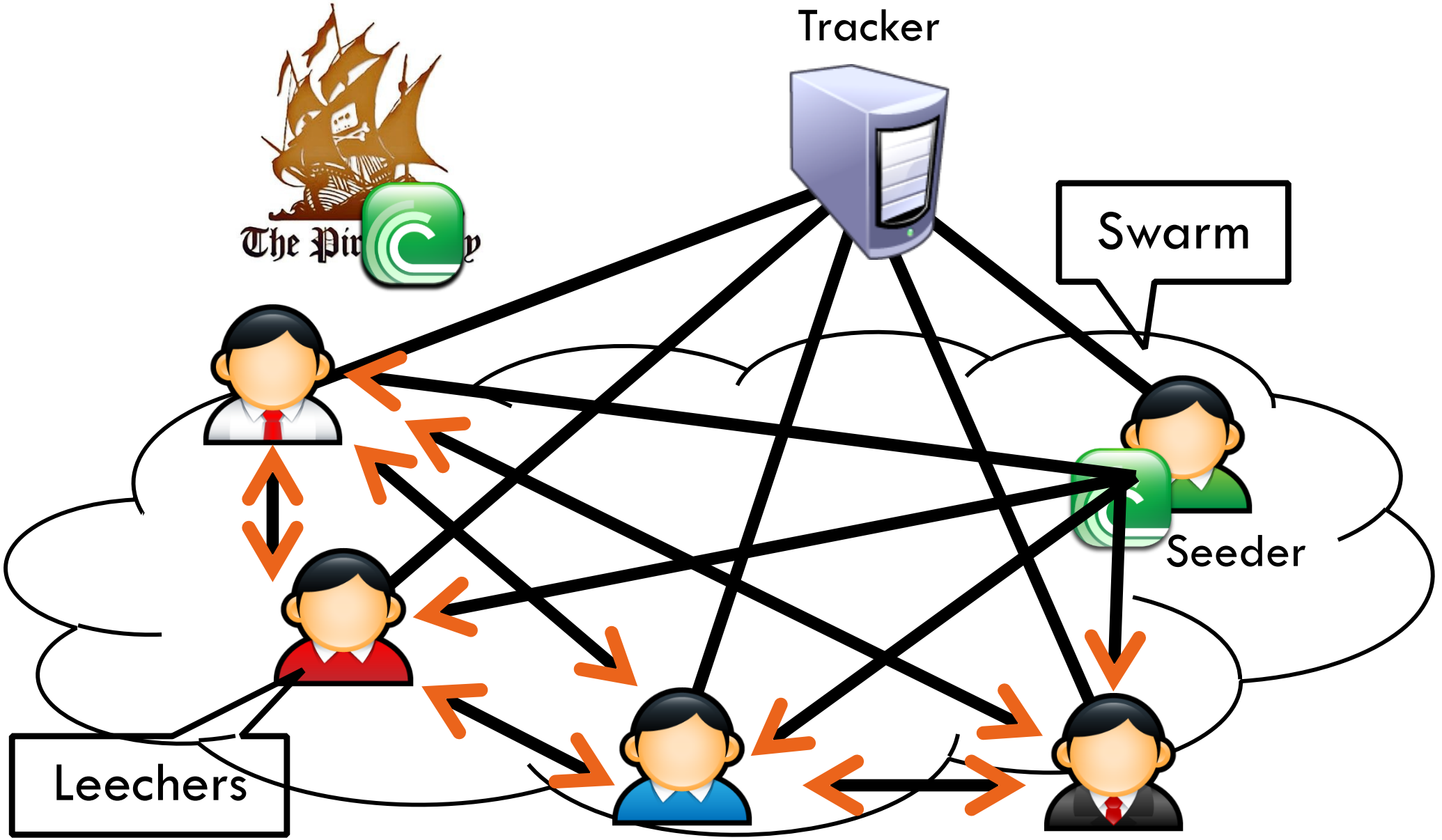
Napster Architecture



What is BitTorrent

- Designed for fast, efficient content distribution
  - Ideal for large files, e.g. movies, DVDs, ISOs, etc.
  - Uses P2P file swarming
- Not a full fledged P2P system
  - Does not support searching for files
  - File swarms must be located out-of-band
  - Trackers acts a centralized swarm coordinators
    - Fully P2P, trackerless torrents are now possible
- Insanely popular
  - 35-70% of all Internet traffic in early 2010

BitTorrent Overview



.torrent File

- Contains all meta-data related to a torrent
  - File name(s), sizes
  - Torrent hash: hash of the whole file
  - URL of tracker(s)
- BitTorrent breaks files into pieces
  - 64 KB – 1 MB per piece
  - .torrent contains the size and SHA-1 hash of each piece
- Basically, a .torrent tells you
  - Everything about a given file
  - Where to go to start downloading

Torrent Sites

- Just standard web servers
  - Allow users to upload .torrent files
  - Search, ratings, comments, etc.
- Some also host trackers
- Many famous ones
  - Mostly because they host illegal content
- Legitimate .torrents
  - Linux distribution
  - World of Warcraft patches

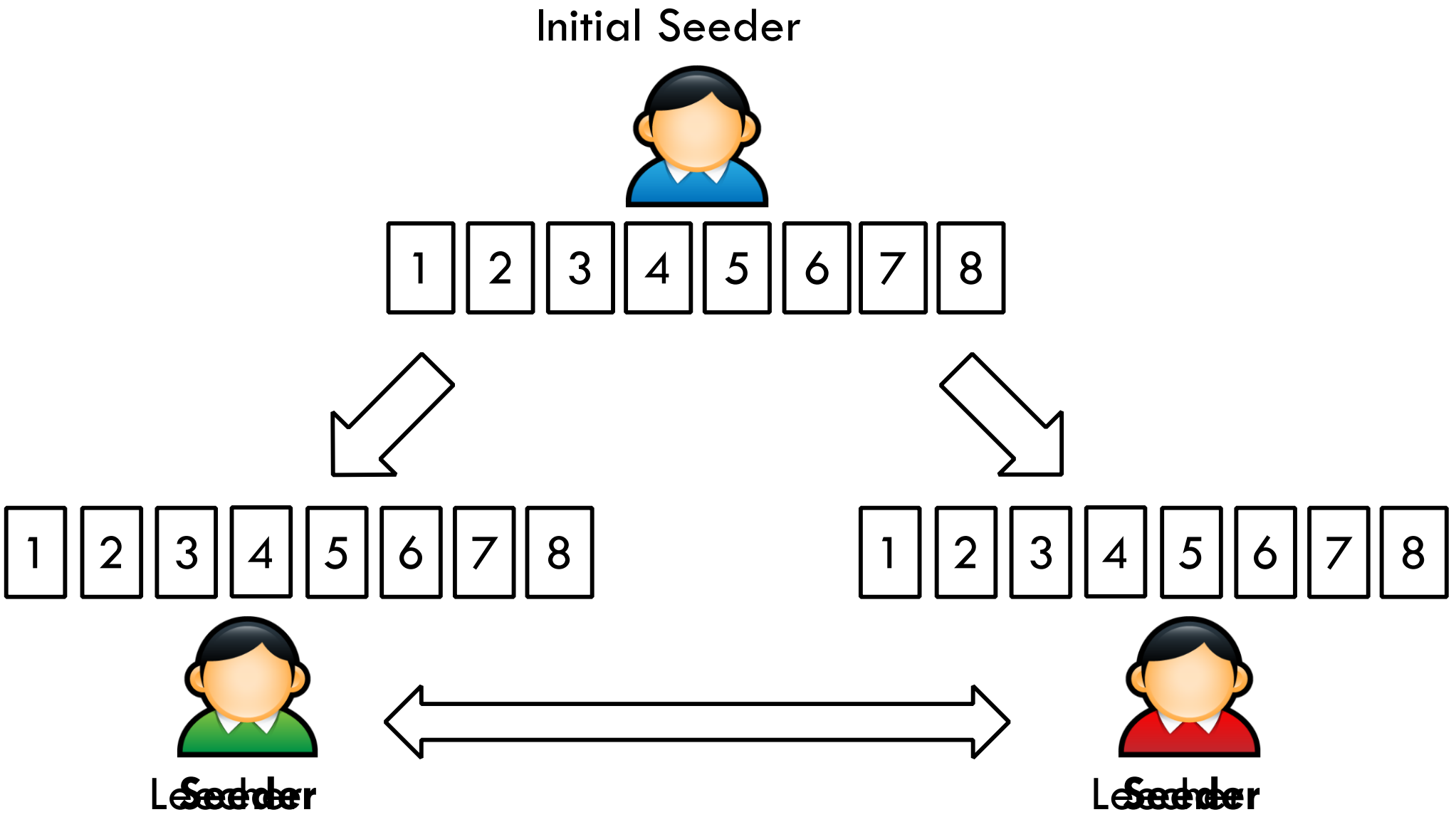
Torrent Trackers

- Really, just a highly specialized webserver
  - BitTorrent protocol is built on top of HTTP
- Keeps a database of swarms
  - Swarms identified by torrent hash
  - State of each peer in each swarm
    - IP address, port, peer ID, TTL
    - Status: leeching or seeding
    - Optional: upload/download stats (to track fairness)
  - Returns a random list of peers to new leechers

Peer Selection

- Tracker provides each client with a list of peers
  - Which peers are best?
    - Fastest bandwidth
- Option 1: learn dynamically
  - Try downloading from many peers
  - Keep only the best peers
  - Strategy used by BitTorrent
- Option 2: use external information
  - E.g. Some torrent clients prefer peers in the same ISP

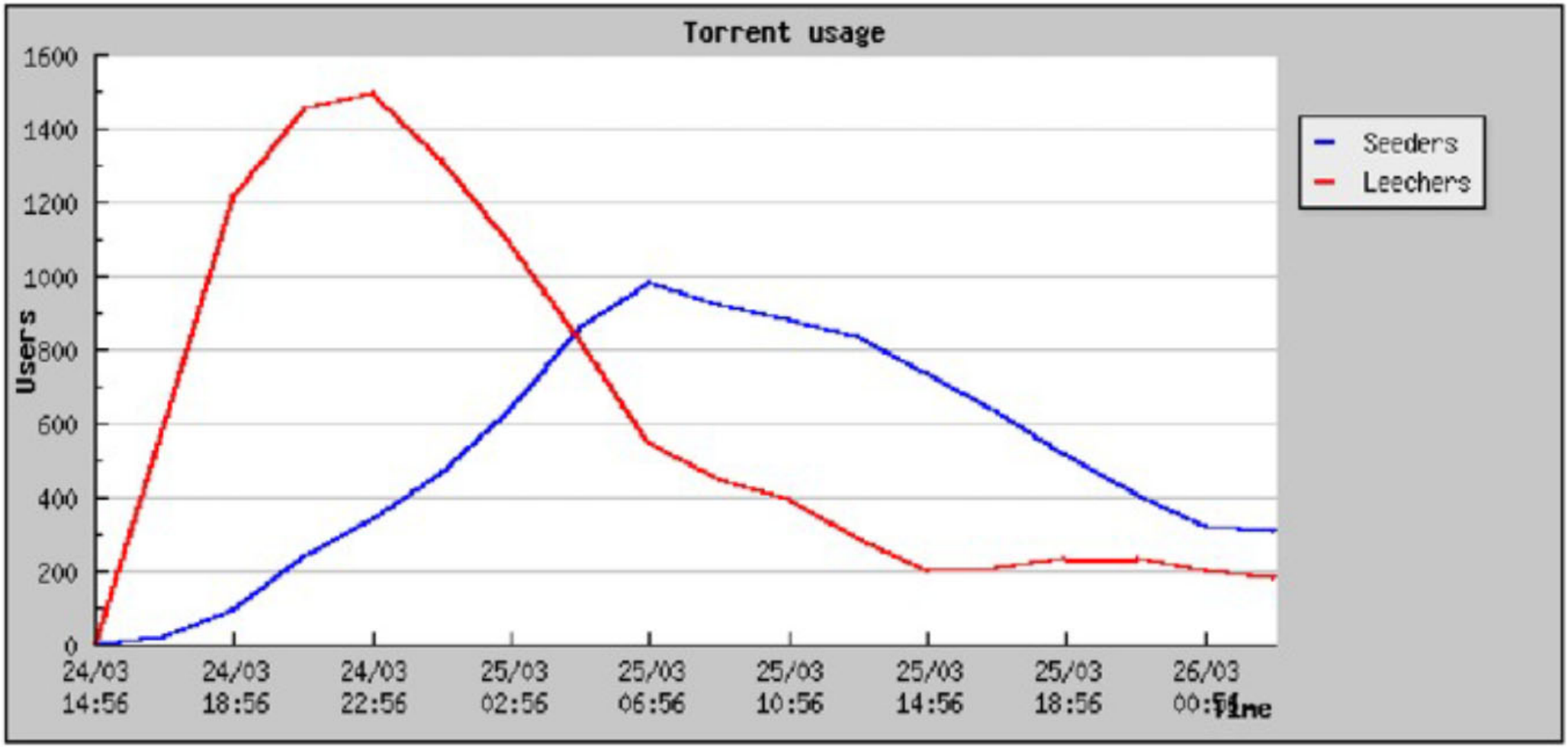
Sharing Pieces



The Beauty of BitTorrent

- More leechers = more replicas of pieces
- More replicas = faster downloads
  - Multiple, redundant sources for each piece
- Even while downloading, leechers take load off the seed(s)
  - Great for content distribution
  - Cost is shared among the swarm

Typical Swarm Behavior



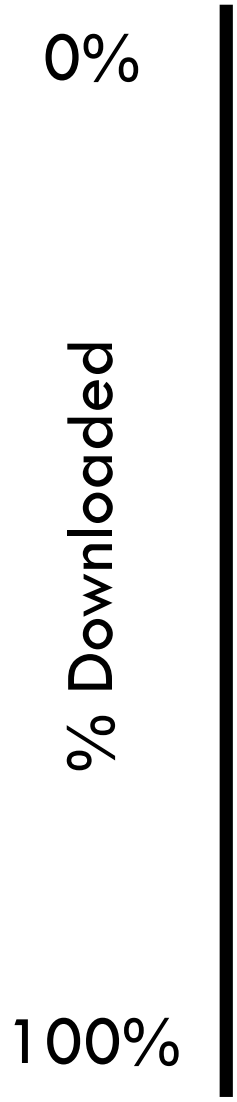
Sub-Pieces and Pipelining

- Each piece is broken into sub-pieces
  - ~16 KB in size
- TCP Pipelining
  - For performance, you want long lived TCP connections (to get out of slow start)
  - Peers generally request 5 sub-pieces at a time
  - When one finished, immediately request another
  - Don't start a new piece until previous is complete
    - Prioritizes complete pieces
    - Only complete pieces can be shared with other peers

Piece Selection

- Piece download order is critical
  - Worst-case scenario: all leeches have identical pieces
    - Nobody can share anything :(
  - Worst-case scenario: the initial seed disappears
    - If a piece is missing from the swarm, the torrent is broken
- What is the best strategy for selecting pieces?
  - Trick question
  - It depends on how many pieces you already have

Download Phases



- Bootstrap: random selection
  - Initially, you have no pieces to trade
  - Essentially, beg for free pieces at random
- Steady-state: rarest piece first
  - Ensures that common pieces are saved for last
- Endgame
  - Simultaneously request final pieces from multiple peers
  - Cancel connections to slow peers
  - Ensures that final pieces arrive quickly

Upload and Download Control

- How does each peer decide who to trade with?
- Incentive mechanism
  - Based on tit-for-tat, game theory
  - "If you give a piece to me, I'll give a piece to you"
  - "If you screw me over, you get nothing"
  - Two mechanisms: choking and optimistic unchoke

A Bit of Game Theory

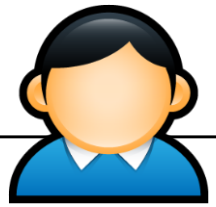
- Iterated prisoner's dilemma
- Very simple game, two players, multiple rounds
  - Both players agree: +2 points each
  - One player defects: +5 for defector, +0 to other
  - Both players defect: +0 for each
- Maps well to trading pieces in BitTorrent
  - Both peers trade, they both get useful data
  - If both peers do nothing, they both get nothing
  - If one peer defects, he gets a free piece, other peer gets nothing
- What is the best strategy for this game?

Tit-for-Tat

- Best general strategy for iterated prisoner's dilemma
- Meaning: "Equivalent Retaliation"

Rules

1. Initially: cooperate
2. If opponent cooperates, cooperate next round
3. If opponent defects, defect next round




Round			Points
1	*Cooperate*	*Cooperate*	*+2 / +2*
2	*Cooperate*	*Defect*	*+0 / +5*
3	*Defect*	*Cooperate*	*+5 / +0*
4	*Cooperate*	*Cooperate*	*+2 / +2*
5	*Cooperate*	*Defect*	*+0 / +5*
6	*Defect*	*Defect*	*+0 / +0*
7	*Defect*	*Cooperate*	*+5 / +0*
		Totals:	*+14 / +14*

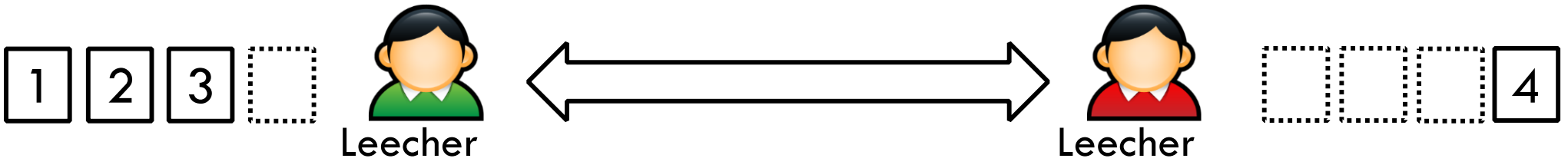
Choking

- Choke is a temporary refusal to upload
  - Tit-for-tat: choke free riders
  - Cap the number of simultaneous uploads
    - Too many connections congests your network
  - Periodically unchoke to test the network connection
    - Choked peer might have better bandwidth

Optimistic Unchoke

- Each peer has one optimistic unchoke slot
  - Uploads to one random peer
  - Peer rotates every 30 seconds
- Reasons for optimistic unchoke
  - Help to bootstrap peers without pieces
  - Discover new peers with fast connections

BitTorrent Protocol Fundamentals



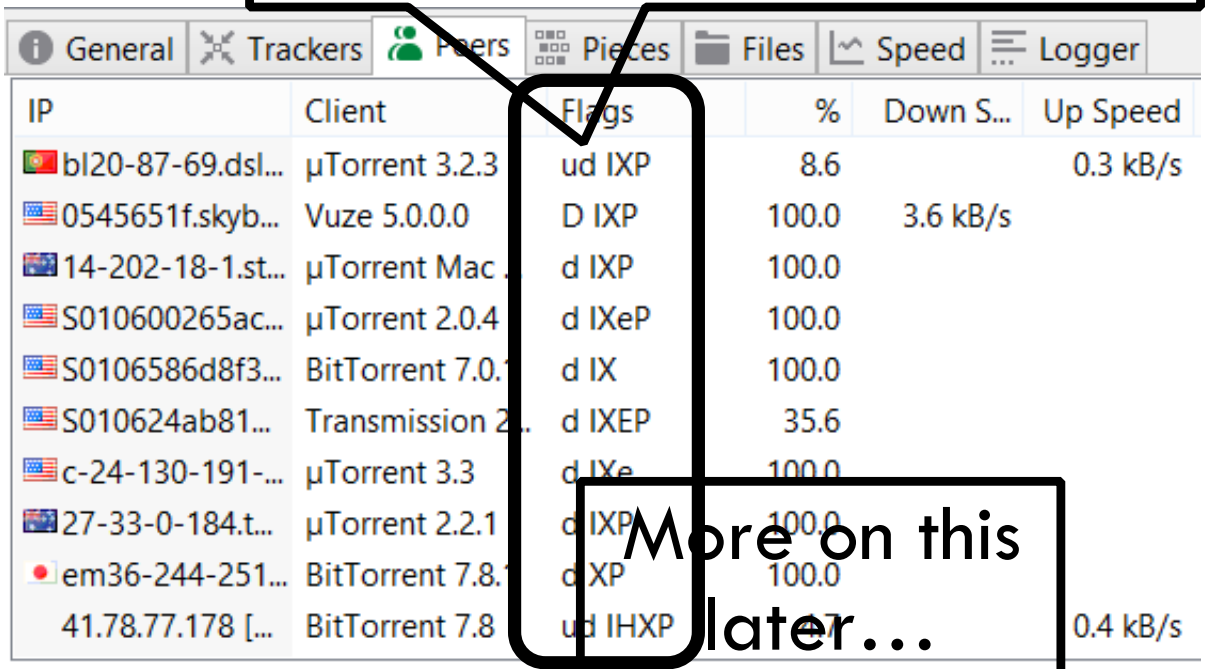
- BitTorrent divides time into rounds
  - Each round, decide who to upload to/download from
  - Rounds are typically 30 seconds
- Each connection to a peer is controlled by four states
  - Interested / uninterested – do I want a piece from you?
  - Choked / unchoked – am I currently downloading from you?
- Connections are bidirectional
  - You decide interest/choking on each peer
  - Each peer decides interest/chocking on you

Connection States

Error states. Connection should be closed

Most peers are d or D. No need to connect with uninteresting peers.

- Download control
  - d – interested and choked
  - D – interested and unchoked
  - K – uninterested and unchoked
  - S – snubbed (no data received in 60 seconds)
  - F – piece(s) failed to hash
- Upload control
  - u – interested and choked
  - U – interested and unchoked
  - O – optimistic unchoke
  - ? – uninterested and unchoked
- Connection information
  - I – incoming connection
  - E/e – Using protocol encryption



More on this later…

  - h – used UDP hole punching
  - P – connection uses µTP
- How was this peer located?
  - H – DHT (distributed hash table)
  - L – local peer discovery (multicast)
  - X – peer exchange

Upload-Only Mode

- Once a peer completes a torrent, it becomes a seed
  - No downloads, no tit-for-tat
  - Who to upload to first?
- BitTorrent policy
  - Upload to the fastest known peer
  - Why?
  - Faster uploads = more available pieces
  - More available pieces helps the swarm

BitTorrent and TCP

- BitTorrent accounts for 35-70% of all Internet traffic
- Thus, BitTorrent's behavior impacts everyone
- BitTorrent's use of TCP causes problems
  - Long lived, BitTorrent TCP flows are "elephants"
    - Ramp up past slow start, dominate router queues
  - Many applications are "mice," get trampled by elephants
    - Short lived flows (e.g. HTTP traffic)
    - Delay sensitive apps (i.e. VoIP, SSH, online games)
- Have you ever tried using SSH while using BitTorrent?

Conclusions

- BitTorrent is an extremely efficient tool for content distribution
  - Strong incentive system based on game theory
  - Most popular file sharing client since 2001
  - More active users than YouTube and Facebook combined
- However, BitTorrent is a large system with many different mechanisms
  - Ample room to modify the client, alter behavior