eecs 489 COMPUTER NETWORKS

Lecture 4: PA1 Walk-through

Lab2 and PA1: P2P Search

Lab2: implement a peer node
- the first one listens for connection
- subsequent ones try to join the network by connecting to a known peer
- return 1 known peer to joining peer
- if peer table (= 2) is full, decline (redirect) joining peer
- if join declined, try other peers (manually)

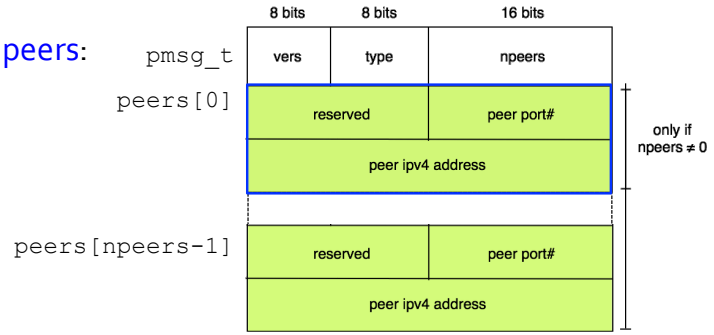
PA1: extend Lab2, integrate with Lab1
- make peer table size a user-defined run-time variable, default to `NETIMG_MAXPEERS` (6)
- return up to `NETIMG_MAXPEERS` peers
- automate peer join on receiving peer list
- search for an image on the p2p network
- remote display image if found

PA1: More Peers

Larger peer table: linear insert/search is ok, may use STL, e.g., hashmap allows for $O(1)$ insert/search

Assume no peer departure, except not to crash a peer when network is being taken down

More peers:



PA1: Automatic Join

Continue to attempt joining until peer table full or known peers exhausted

Four cases to watch out for:
1. peer already in peer table
2. peer "recently" rejected join request (peer declined table), remember only the last `NETIMG_MAXPEERS` rejections
3. pending peer upon successful `connect()` before receiving acknowledgement
4. simultaneous join (`connect()` returns -1 with `errno` set to `EADDRNOTAVAIL`; peer not added to peer table if `connect()` fails, peer will be added by the successful `accept()` instead; peering relationship is bidirectional)

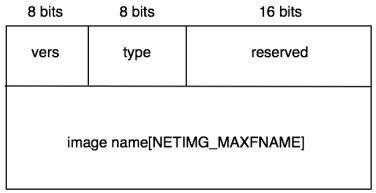
PA1: Image Search

A node on the p2p network includes both `peer` (Lab2) and `imgdb` (Lab1) objects

It listens on 2 sockets: peer socket for p2p network management, image socket for image query and reply

Client connects to the image socket and sends an `iqry_t` message to query for an image

Node simply calls `imgdb::readimg()` to "search" for an image locally



PA1: P2P Search

If image not found locally, node sends out a search packet on the p2p network

- node maintains only one outstanding search at any one time
- if there's already an outstanding search, node returns `NETIMG_EBUSY` to clientreturns `NETIMG_NFOUND` to client if times out waiting for search reply

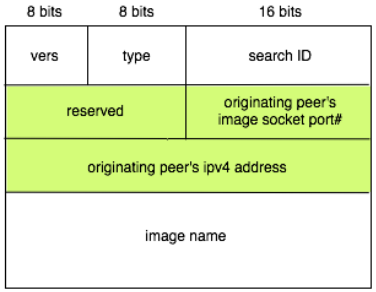
If image not found locally, node floods a search packet:

- search packet sent out to all peers, except incoming peer

P2P search packet is sent to another peer's peer socket

PA1: P2P Search

Each search is given a `searchID` by its originating peer (can be a simple monotonically increasing number)



Prevent loop by storing `NETIMG_MAXPEERS` number of search packets in a circular buffer

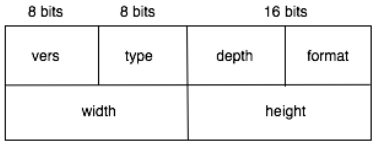
Don't forward a search packet if:

- search packet recently seen and forwarded already
- peer is incoming peer

PA1: P2P Search

If a peer has the queried image, it connects directly to the originating peer (not client) at the originating peer's image socket

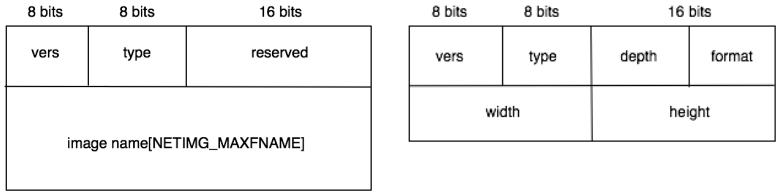
The returned image must be preceded by an `imsg_t` packet, with type field set to `NETIMG_FOUND`



To test, run your `p2pdb` (or `refp2pdb`) on a folder with only one image file so that each peer has only one, unique local image file

PA1: Demultiplexing

On an image socket, a peer may receive a `NETIMG_QUERY` or a `NETIMG_FOUND` packet



Demultiplexing by the packet's `type` field may use:

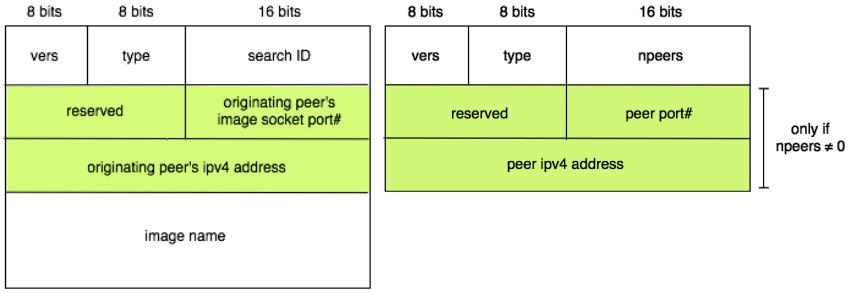
- `MSG_PEEK` the first two bytes of packet

Check version number:

- use `socks_clear()` to clear "pipe" of unrecognized bits

PA1: Demultiplexing

On a peer socket, a peer may receive a `PM_WLCM`, `PM_RDRT`, or `PM_SRCH` packet



Demultiplexing by the packet's `type` field may use

- `recv()` the first four bytes of packet first

PA1: P2P Search

May "purchase" solutions to the labs:

- each lab costs 20 points

You DON'T have to build off the support code

- you may build your own from scratch

BUT must interoperate with `refp2pdb` and the `netimg` client and NOT RECOMMENDED

Turn in your implementation of both `p2pdb` and `netimg`

Hygiene

Keep a back up of all your submitted files (as individual files) on a private third party repository

- local file modification dates can be easily modified −10pts

Don't turn in support code you haven't modified −4pts

Don't turn in binary (exe, obj, dll, image) files −4pts

Don't use library or compiler option not used in the provided `Makefile` −10pts

Please do NOT share support code with others not taking the course

NEVER share access to your accounts