Project 3
(contd. :)

cse461

Motivation

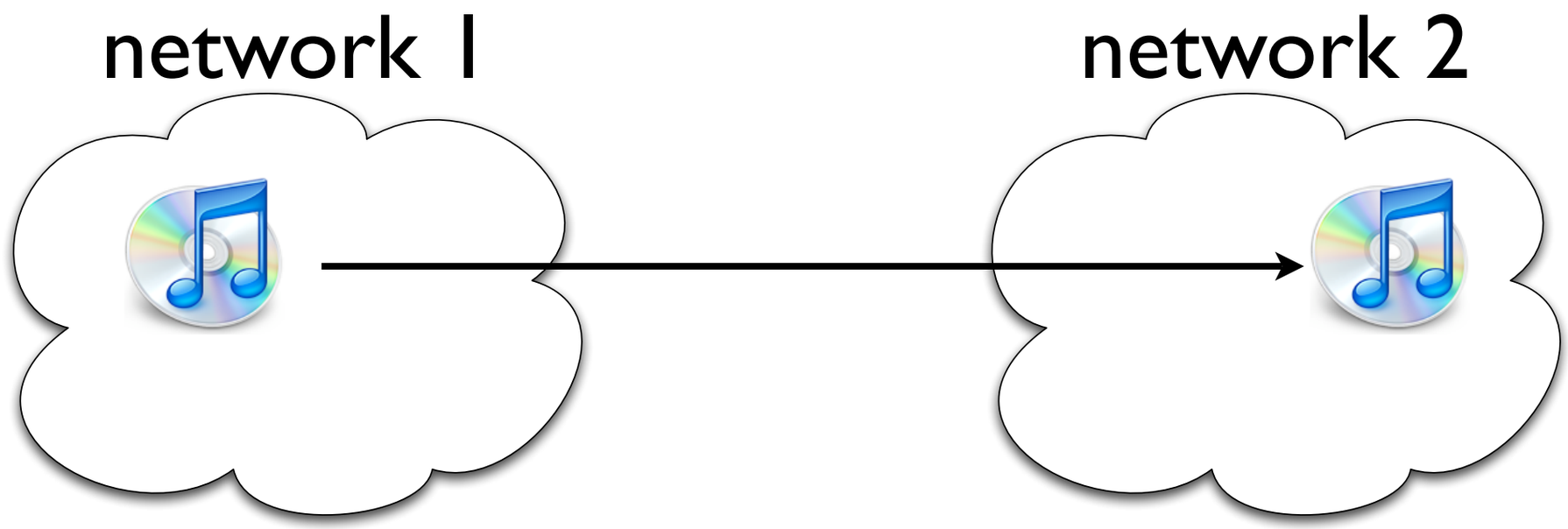
- Friend has an iTunes library that I would like to access, but unless we are on the same network, I can't "see" her shares.

Motivation

- Friend has an iTunes library that I would like to access, but unless we are on the same network, I can't "see" her shares.

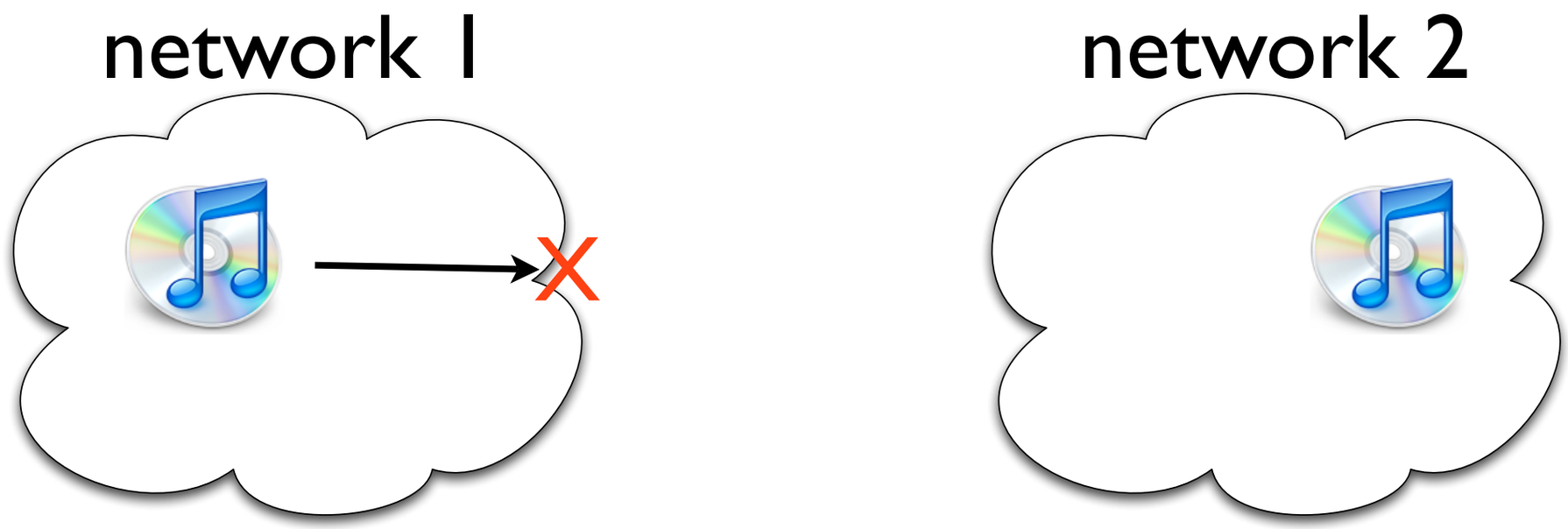
Motivation

- Friend has an iTunes library that I would like to access, but unless we are on the same network, I can't "see" her shares.



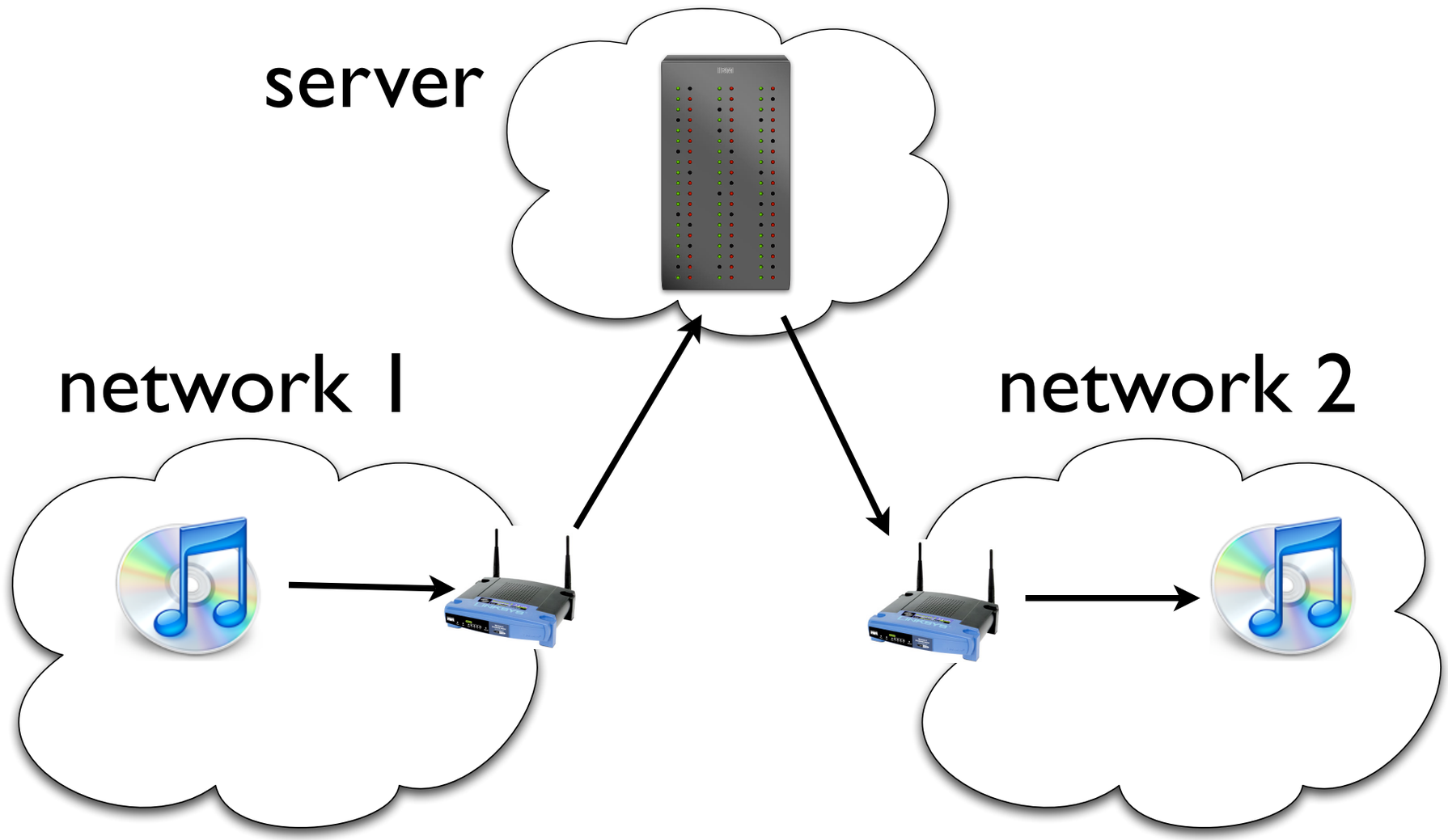
Motivation

- Friend has an iTunes library that I would like to access, but unless we are on the same network, I can't "see" her shares.



Set up

- In this project, you will explore a centralized solution to this problem

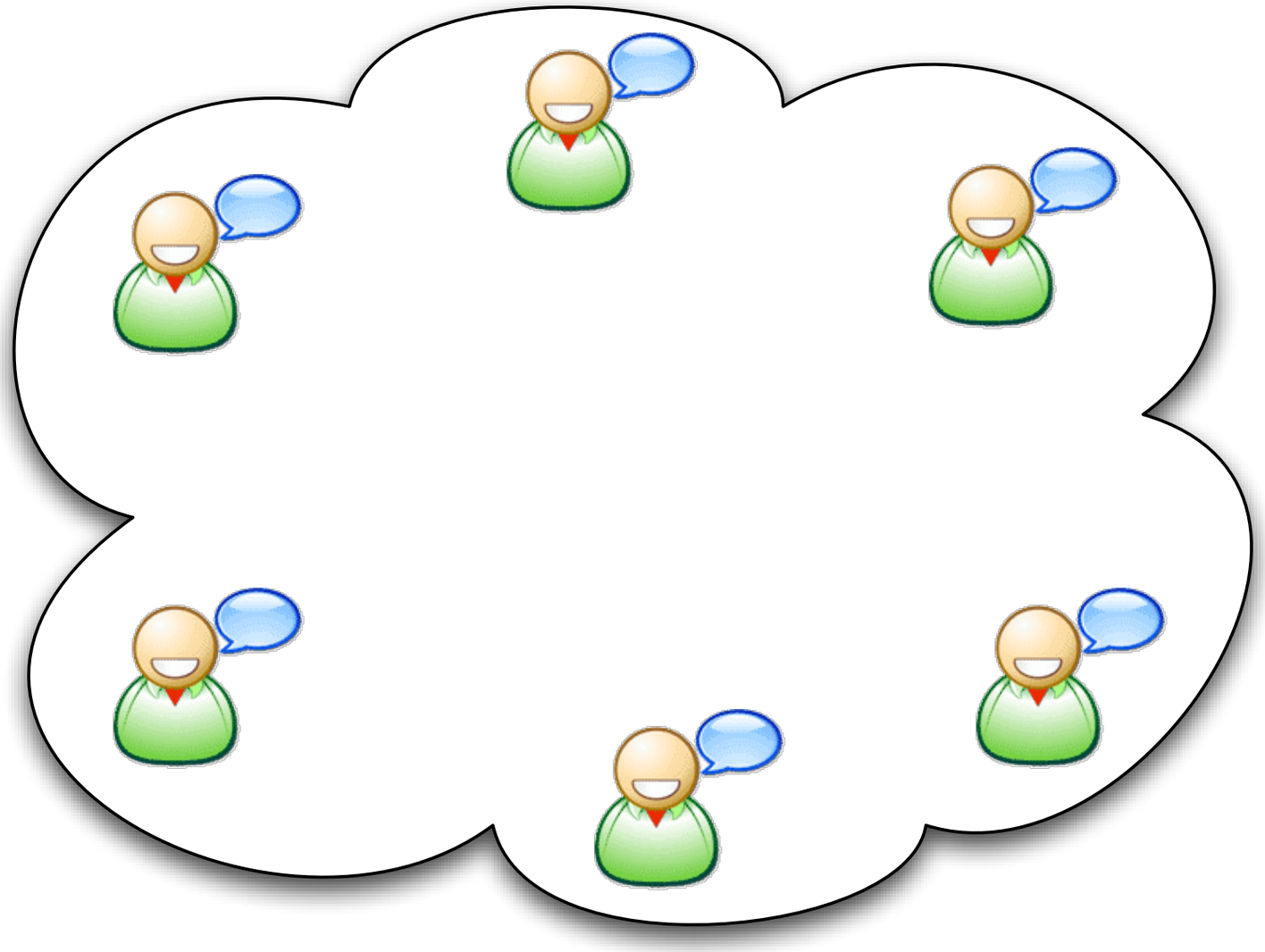


One small problem..

- iTunes uses mDNS (IP Multicast) for discovery and a TCP connection to exchange play lists
- mDNS is non-trivial and you have little time to learn to rewrite mDNS packets and tunnel TCP connections
- Instead, we (you) will use our (your) own application!

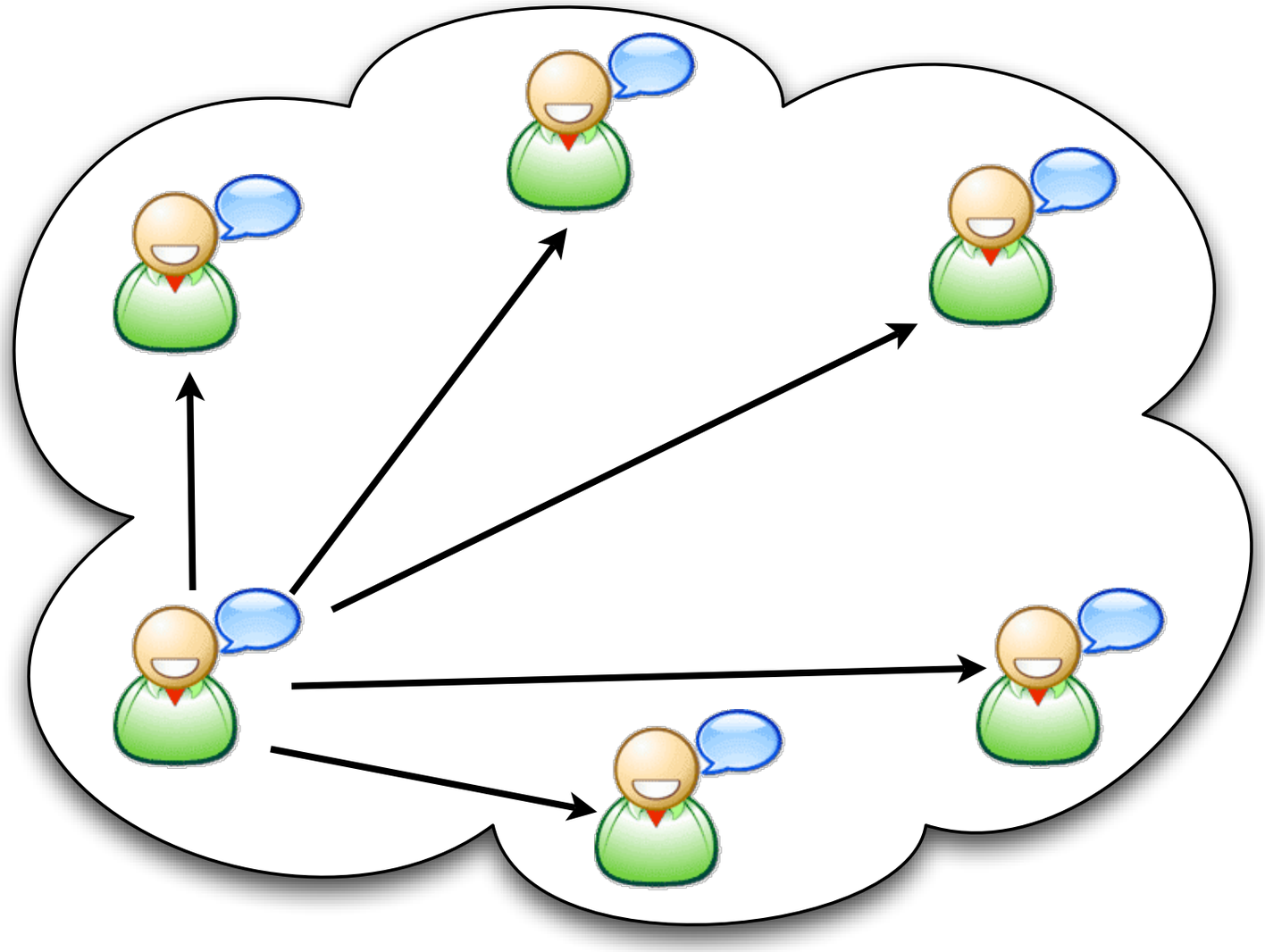
BChat : Broadcast Chat

- Chat messages are sent via subnet broadcast
- All participants see all messages



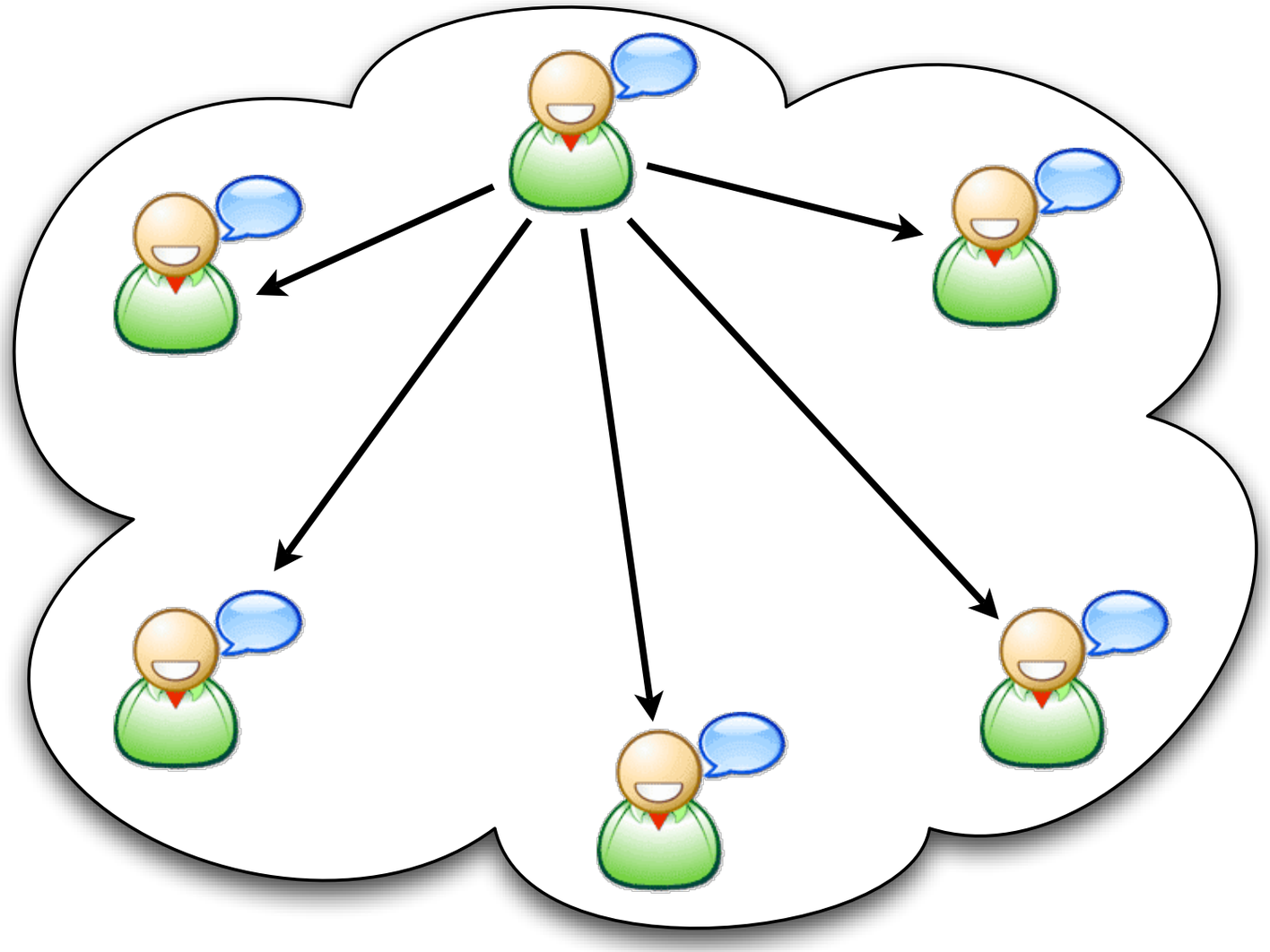
BChat : Broadcast Chat

- Chat messages are sent via subnet broadcast
- All participants see all messages



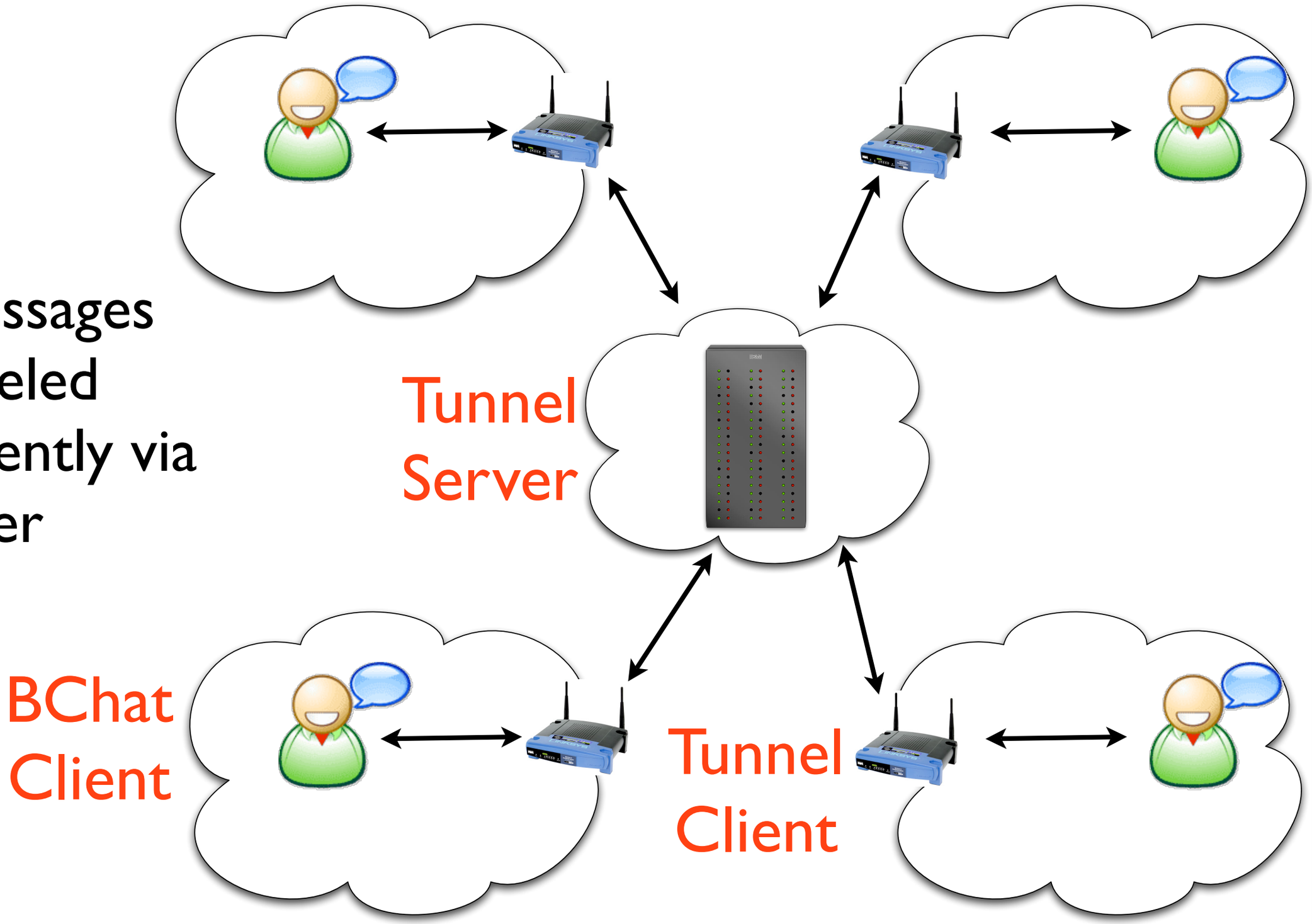
BChat : Broadcast Chat

- Chat messages are sent via subnet broadcast
- All participants see all messages



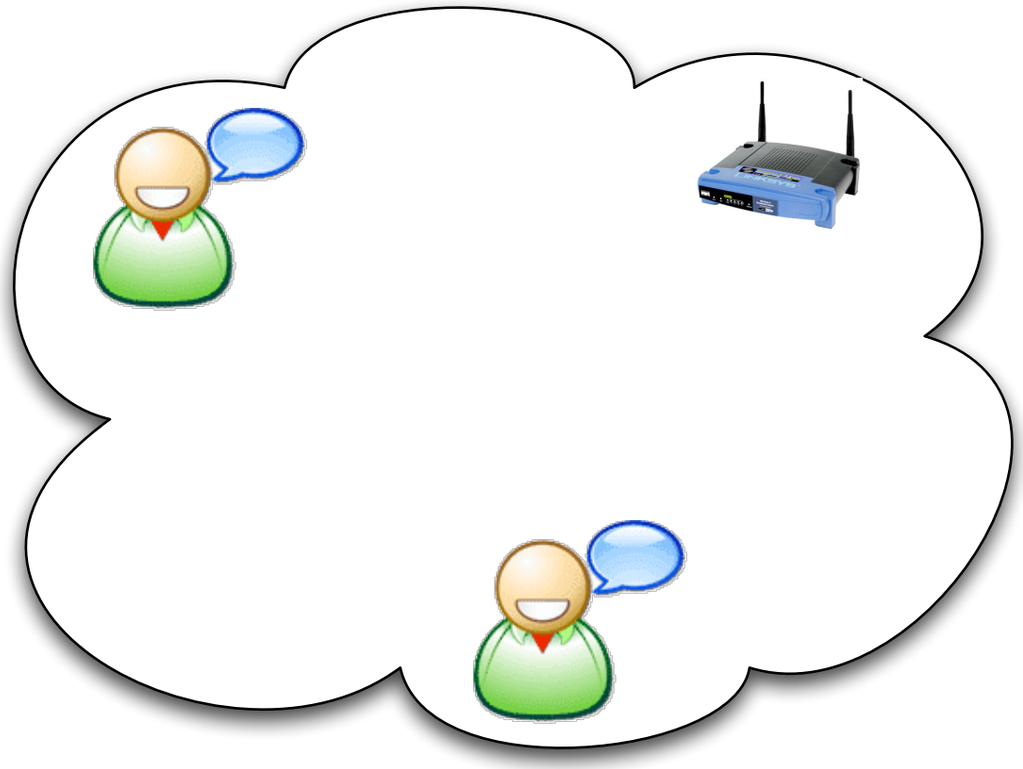
Tunneling BChat

- Chat messages are tunneled transparently via the server



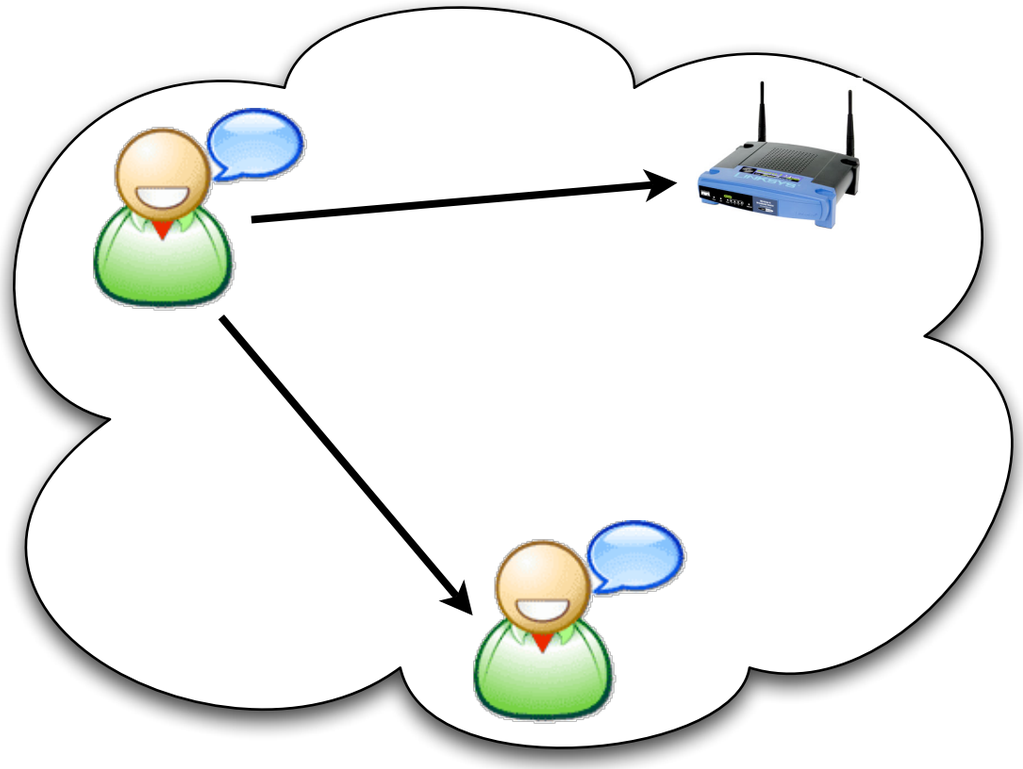
BChat Client protocol

- Send subnet broadcast UDP msgs on port 52367
- Receive subnet broadcast msgs on port 52368
- UDP payload should contain a string of format: “name: message”



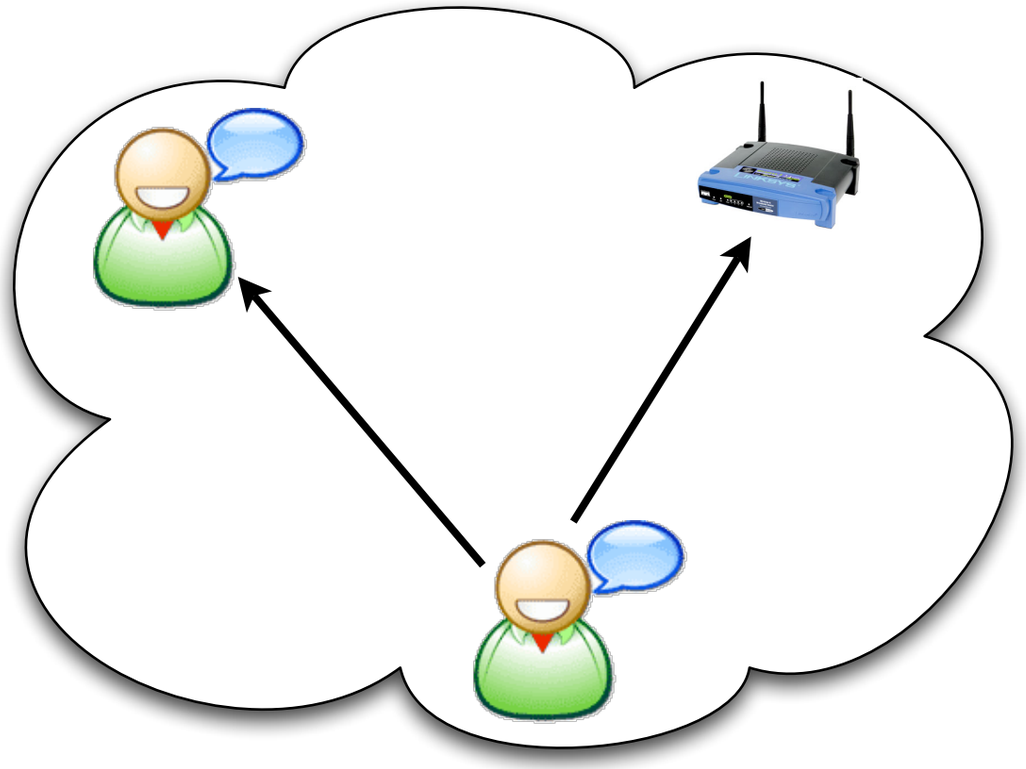
BChat Client protocol

- Send subnet broadcast UDP msgs on port 52367
- Receive subnet broadcast msgs on port 52368
- UDP payload should contain a string of format: “name: message”



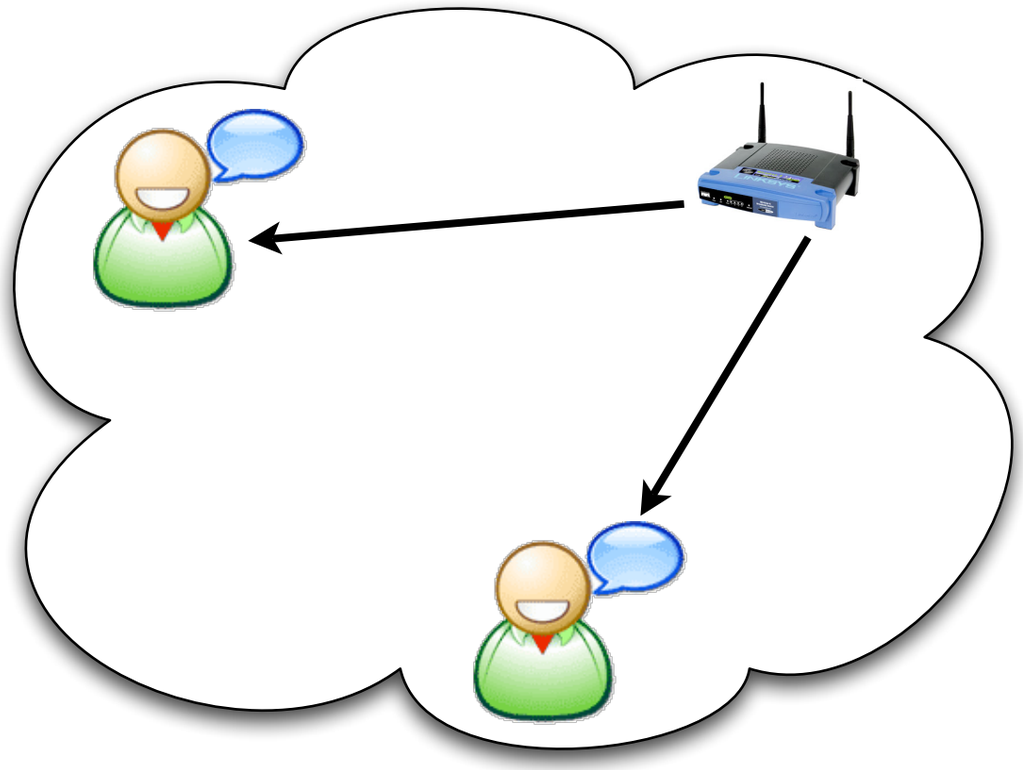
BChat Client protocol

- Send subnet broadcast UDP msgs on port 52367
- Receive subnet broadcast msgs on port 52368
- UDP payload should contain a string of format: “name: message”



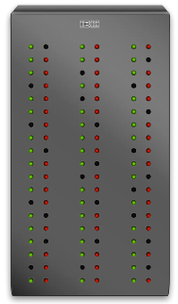
BChat Client protocol

- Send subnet broadcast UDP msgs on port 52367
- Receive subnet broadcast msgs on port 52368
- UDP payload should contain a string of format: “name: message”



Tunneling protocol

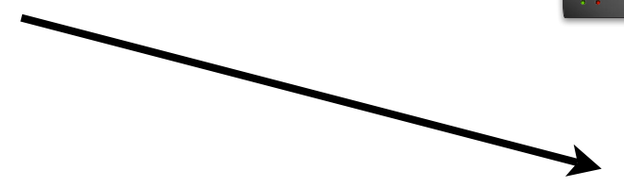
Tunnel Client



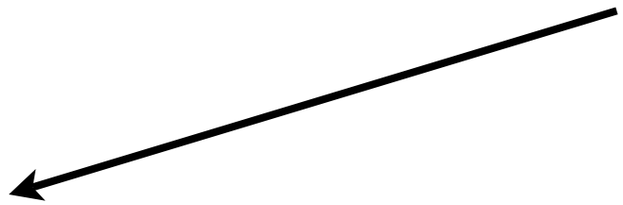
Tunnel Server

- Server and client exchange the same message type as in Step 1

```
typedef struct msg {
    uint32_t len;
    void* payload;
} msg_t;
```



len, payload



len', payload'

Re-broadcasting Strategy

- Packets received from the Tunnel server by the Tunnel client need to be re-broadcast on the local network:

1. Translate src IP to router's IP
2. Translate dst IP to local subnet bcast IP
3. Translate UDP src, and dst ports accordingly
4. Send translated packet on a raw socket

Requirements

- Tunneling client that interoperates with the tunneling server
- BChat client that interoperates with other BChat clients, and tunneling clients
- A BChat client interface that allows one to send new chat messages, and to see chat messages (e.g. stdin/stdout, curses, web..)

Extra Credit

- Use multicast instead of subnet broadcast for BChat messages
- Use your web-server from Project 2 to snoop, and show a log of all chat messages via a browser
- Use your web-server to provide a web interface to the BChat client
- Besides snooping, add a way to inject chat messages from the web-server on the router (the local subnet has to see these messages) via a web-interface
- Create a new application that uses the tunneling protocol in a novel manner

Due Date

- Everything due on Friday, 12/5 at 11:59 PM (last day of school)
- Include all source code, along with compilation/usage instructions