Chord (CAN,Tapestry, Pastry) [Stoica01a]

Stoica, R. Morris, D. Karger, F. Kaashoek, and H.Balakrishnan.

Chord: A peer-to-peer lookup service for Internet applications. Proceedings of *ACM SIGCOMM* Conference, pages 149-160, August 2001

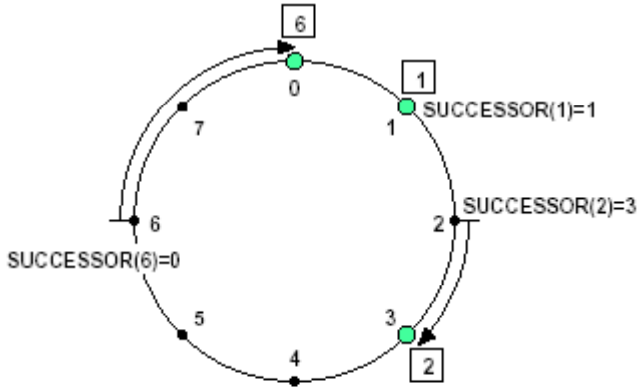
Chord

- A structured peer-to-peer system
- Map key to value
- Emphasis on good algorithmic performance
  - Use consistent hashing
  - O(logN) route storage, O(logN) lookup cost, $O(\log^2 N)$ cost to join/leava
- Easy if static, but must deal with node arrivals and departures

Compare Search in Several Peer-to-Peer Systems

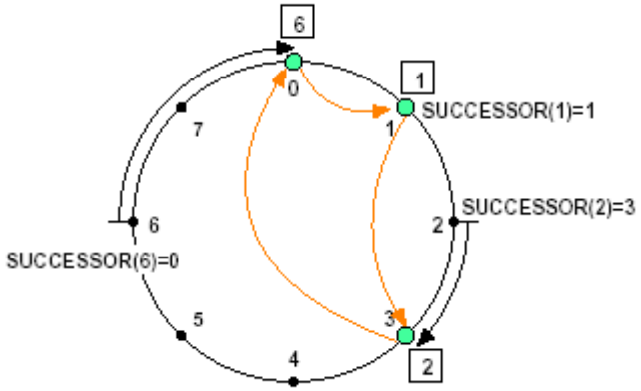
- Napster: central search engine
- Freenet: search towards keys, but no guarantees
- Chord:
  - Map keys to linear search space
  - Keep pointers (fingers) into exponential places around space
  - Probabilistic (depends on hashing)

Hashing Nodes and Data



- Nodes hash IP addresses to key space
  - Because this hashing is random, can expect nodes to be evenly distributed in key space
- Store data in the successor of the data item's key
- Property:
  - If each node maintains successor, can find any data item

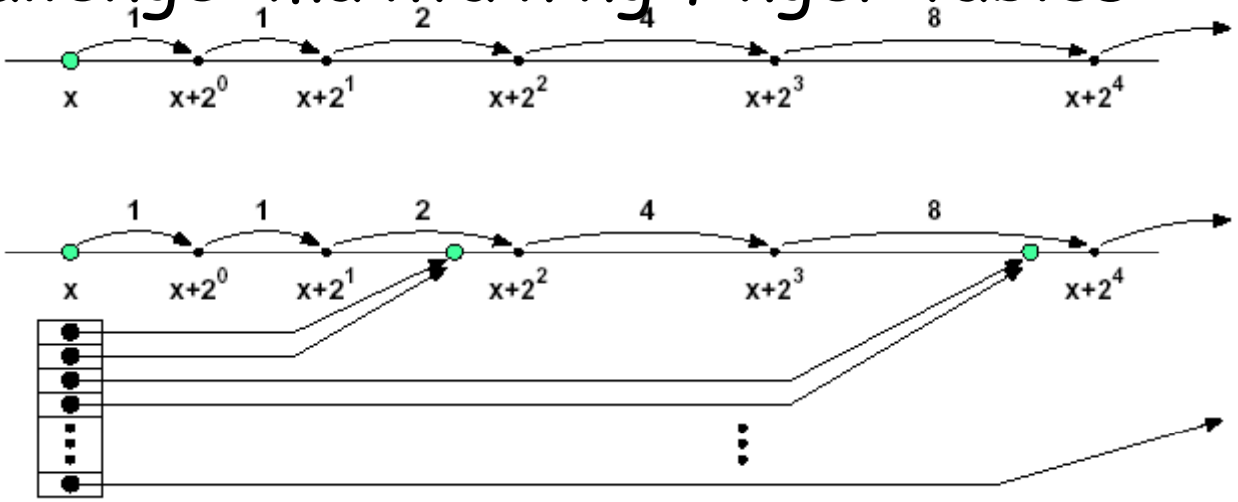
Hashing Nodes and Data (cont.)



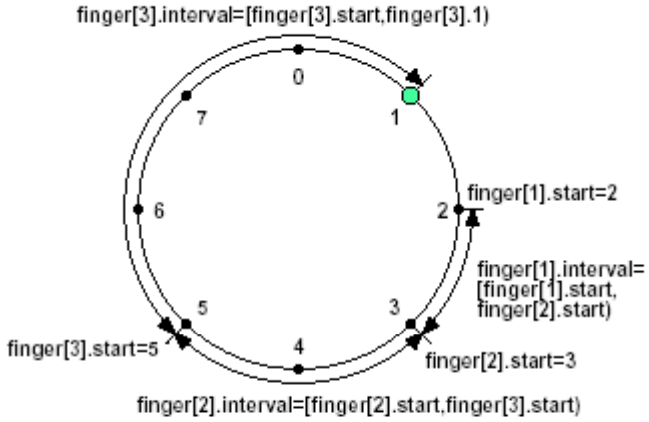
- Nodes hash IP addresses to key space
  - Because this hashing is random, can expect nodes to be evenly distributed in key space
- Store data in the successor of the data item's key
- Property:
  - If each node maintains successor, can find any data item
  - But O(n) performance

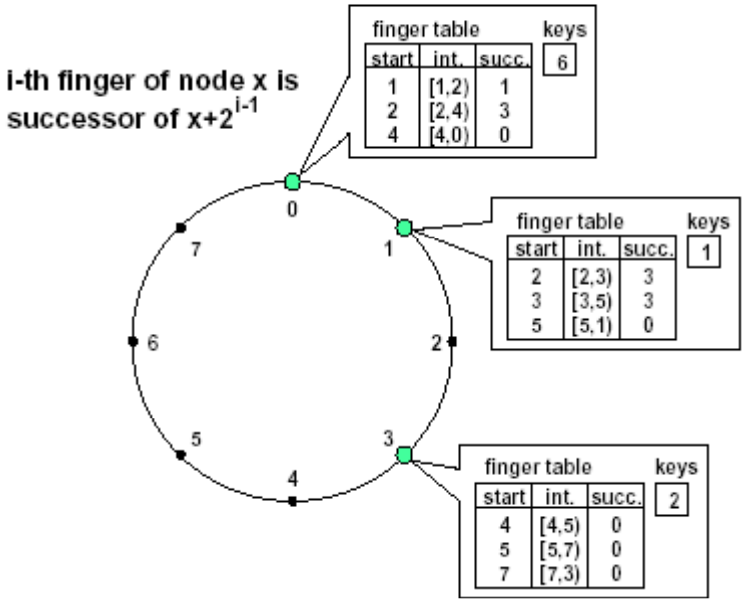
Improving Search Performance with Finger Tables

- Finger tables enable logarithmic lookup
  - i-th finger of node x is successor of $x+2^{i-1}$
  - At each step, we halve the remaining distance (in key space) to the target
- Challenge: maintaining finger tables



Finger Tables Example





Node Join

- Must keep successors and finger table current
- Use successors for correctness
  - Can always fall back on them to find a key
- Use finger table for performance
  - Must update it, but can tolerate temporary errors
- Keep successor and predecessor so we can update our neighbors
- Key observations: can find successors and fingers by doing a lookup on the existing Chord ring!

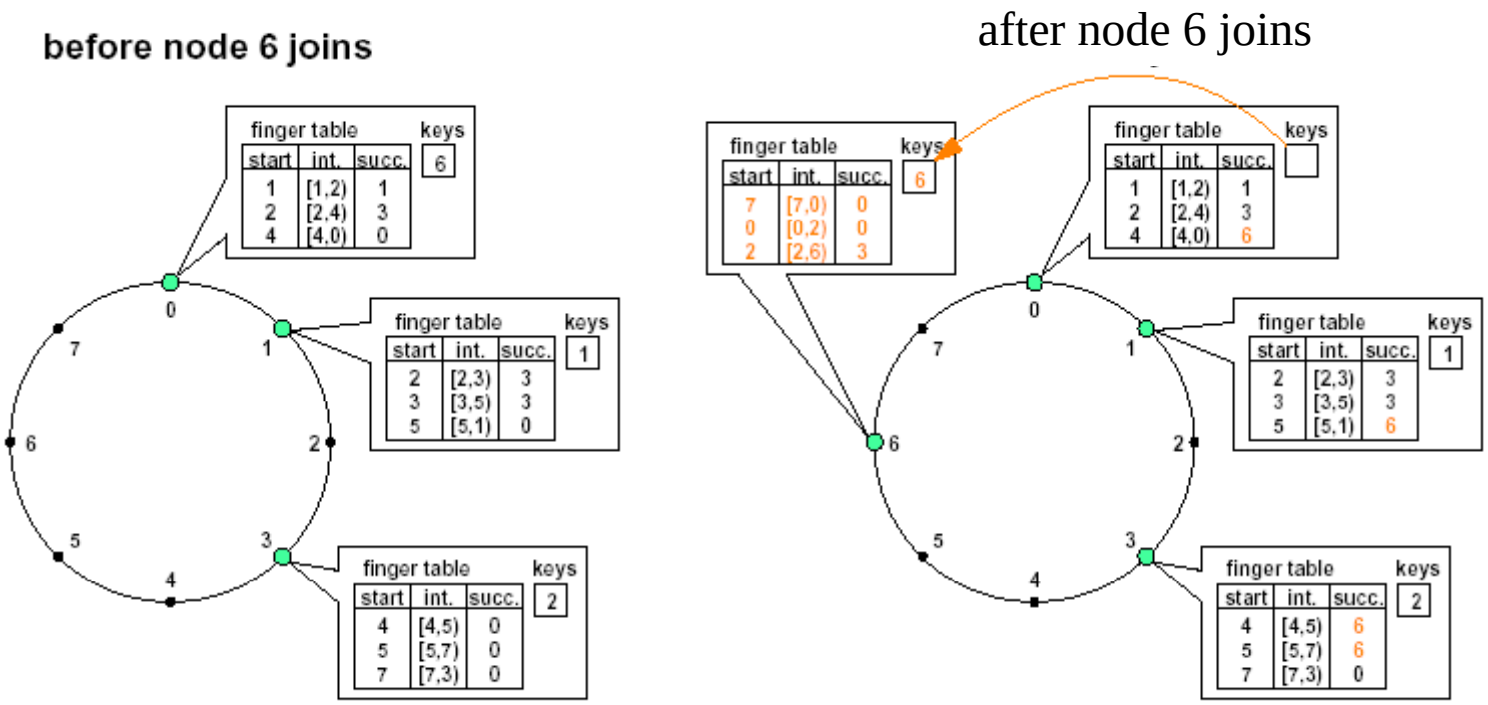
Finding Predecessor and Successor

```
node.find_successor(key)
        n = find_predecessor(key);
        return n.successor;

node.find_predecessor(key)
        n = node;
        while (key ∉ (n,n.successor])
                n = n.closest_preceding_finger(key);
        return n;

node.closest_preceding_finger(key)
        for (i=m; i > 0; i--)
                if (finger[i].node ∈ (node,key))
                        return finger[i].node;
        return node;
```

Join Example



- When new node enters, it establishes its successor and predecessor and then builds its finger table, and moves and keys it now "owns"

Robustness

- Stabilization algorithm to confirm ring is correct
  - Every 30s, ask successor for its predecessor
    - Fix your own successor based on this
    - Successor fixes its predecessor if necessary
  - Also, pick and verify a random finger table entry
    - Rebuild finger table entries this way
    - Important observation: finger tables can be incorrect for some time (between network size of N and 2N)
- Dealing with unexpected failures:
  - Keep successor list of r successors
  - Can use these to replicate data

Chord Performance

- Performance dominated by lookup cost
  - How long does it take to get to the node that stores a key?
- Chord promises few O(logN) hops on the overlay
  - But, on the physical network, this can be quite far
    - This is often the problem with overlay networks

Roadmap

- Background on P2P
- DHT and Unstructured Systems
- Chord
- Skype
- Coolstreaming
- Network Coding for P2P
- Q&A

An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol

Salman A. Baset and Henning Schunllzrinne

INFOCOM 2006.
25th IEEE International Conference on Computer Communications. Proceedings
(2006), pp. 1-11.

Outline

- **Introduction**
- Key components of Skype
- Experiment Setup
- Skype Functions
- Conferencing
- Comparisons between IM applications
- Conclusion

Introduction

- Skype is a peer-to-peer VoIP client developed by KaZaa in 2003.
- Skype claims that it can work almost seamlessly across NATs and firewalls and has better voice quality than the MSN and Yahoo IM application.
- It encrypts calls end-to-end, and stores user information in a decentralized fashion.
- Skypes also supports instant messaging and conferencing.

Skypes Network

- Ordinary host
  - A Skype application
  - place voice calls and send text messages
- Super nodes (SN)
  - An ordinary host's end-point
  - have a public IP address, sufficient CPU, memory, and network bandwidth.
- Skype login server
  - An ordinary host must register itself with login server.
  - User authentication at login is done here.

Skypes Network

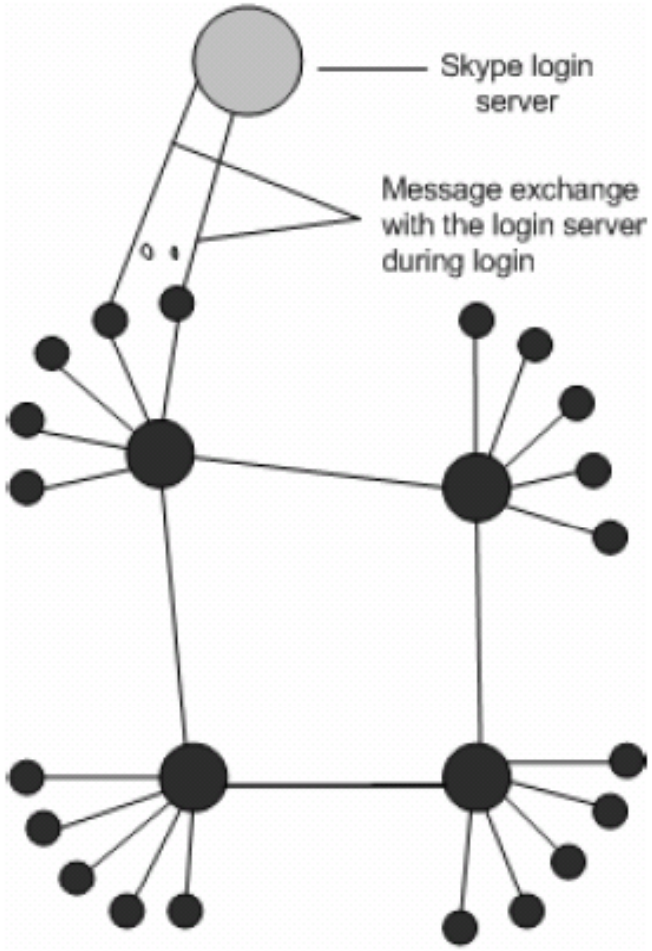


Figure 1. Skype Network. There are three main entities: supernodes, ordinary nodes, and the login server.

Outline

- Introduction
- **Key components of Skype**
- Experiment Setup
- Skype Functions
- Conferencing
- Comparisons between IM applications
- Conclusion

Key Components

- Ports
  - Skype Client (SC) opens a TCP and a UDP listening port configured in its connection dialog box.
  - Also opens TCP listening ports at port 80 (HTTP port), and port 443 (HTTPS port)

- Host Cache (HC)
  - A list of super node IP address and port pairs that Skype Client (SC) builds and refreshed regularly.
  - At least one valid entry must be present.
  - At most 200 entries in HC.

Key Components (cont.)

- Codecs
  - Uses iLBC, iSAC ,iPCM codec.
  - Skype codecs allows frequencies between 50~8000Hz to pass through.→ wideband codec.

- Buddy list
  - The buddy list is local to one machine and stored on a central server.

Key Components (cont.)

- Encryption
  - AES (Advanced Encryption Standard)
- NAT and Firewall
  - Use a variant of STUN and TURN protocol
  - No global NAT and firewall traversal server from experiments.

Outline

- Introduction
- Key components of Skype
- **Experiment Setup**
- Skype Functions
- Conferencing
- Comparisons between IM applications
- Conclusion

Experiment Setup

- Test Environment
  - Windows Skype version 1.4.0.84
  - Linux Skype version 1.2.0.18

- Machine
  - 3 GHz Pentium 4 CPU with 1GB of RAM
  - 10/100 Mb/s Ethernet card
  - Connect to 100Mb/s network

Experiment Setup

- Network Setups
  - Both Skype users with public IP addresses
  - One Skype user behind a port-restricted NAT
  - Both behind a port-restricted NAT and UDP-restricted firewall

- NAT and firewall machines run Mandriva Linux 10.2

Outline

- Introduction
- Key components of Skype
- Experiment Setup
- **Skype Functions**
- Conferencing
- Comparisons between IM applications
- Conclusion

Skype Function

- Startup
- Login
- User Search
- Call Establishment and Teardown
- Media Transfer and Codecs

Skype Functions - Startup

- When SC was run for the first time after installation, it sent a HTTP 1.1 GET request to the Skype server.

- The first line of this request contained the keyword "installed"

Skype Functions - Login

- SC authenticates its user name and password with the login server
- Advertises its presence to other peers and buddies
- Determine the type of NAT and firewall it is behind
- Discover online super nodes with public IP address.
- Checks the availability of latest Skype version
  - Send HTTP 1.1 GET request keyword: "getlastestversion"

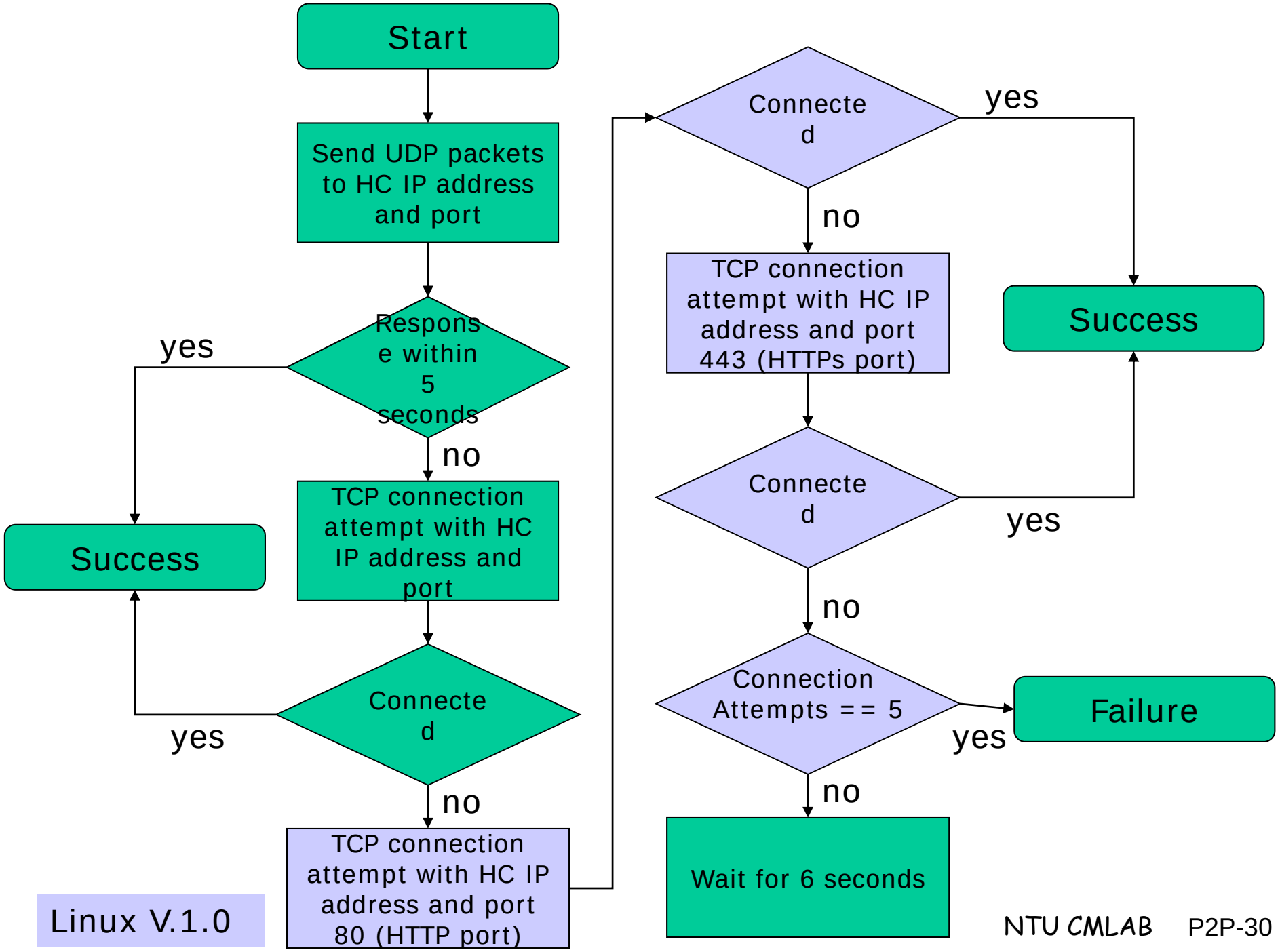
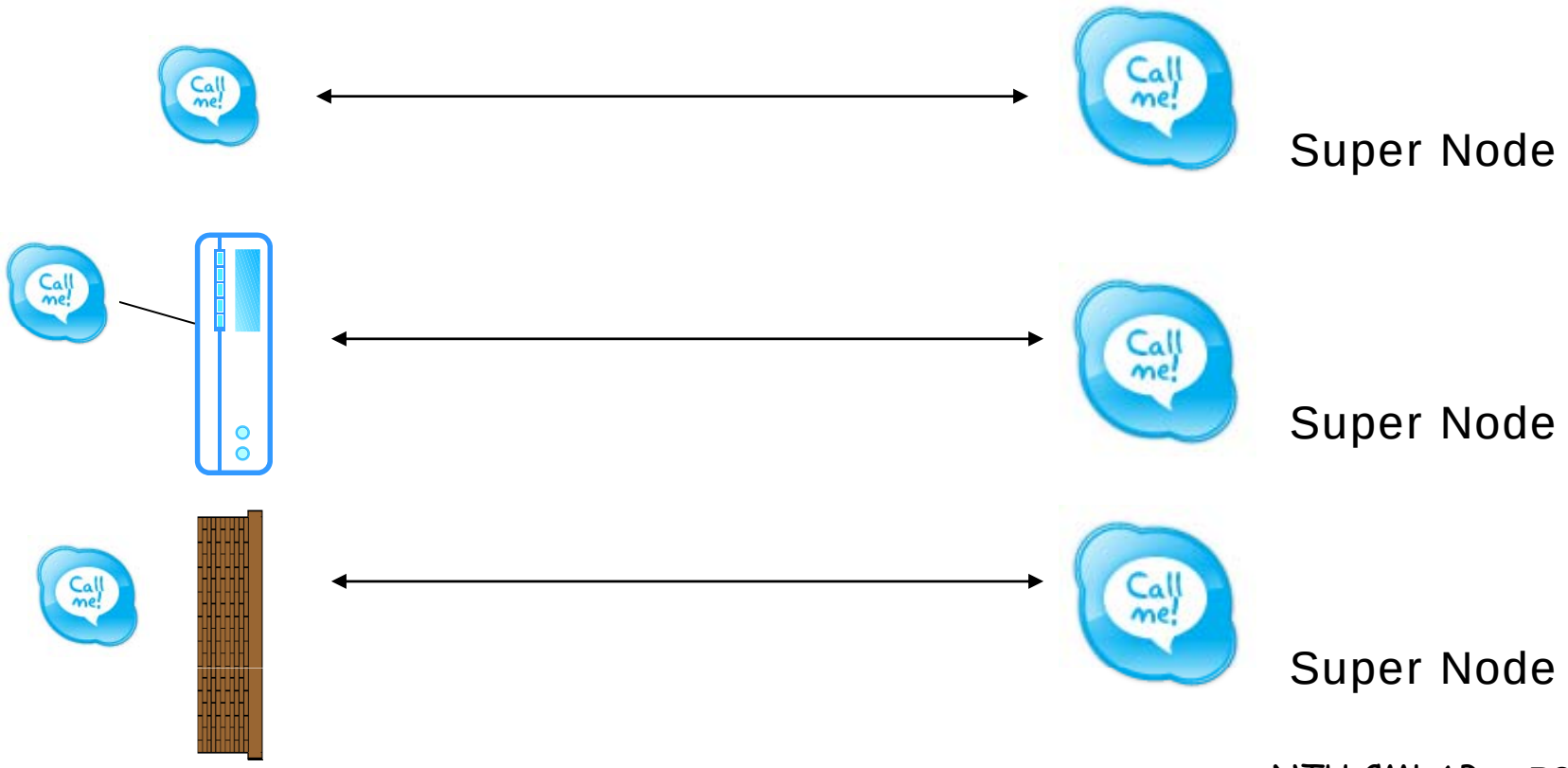

Start
Send UDP packets to HC IP address and port
Respons e within 5 seconds
yes
no
Success
TCP connection attempt with HC IP address and port
Connecte d
yes
no
TCP connection attempt with HC IP address and port 80 (HTTP port)
Connecte d
yes
no
TCP connection attempt with HC IP address and port 443 (HTTPs port)
Success
Connecte d
yes
no
Connection Attempts == 5
yes
Failure
no
Wait for 6 seconds
Linux V.1.0

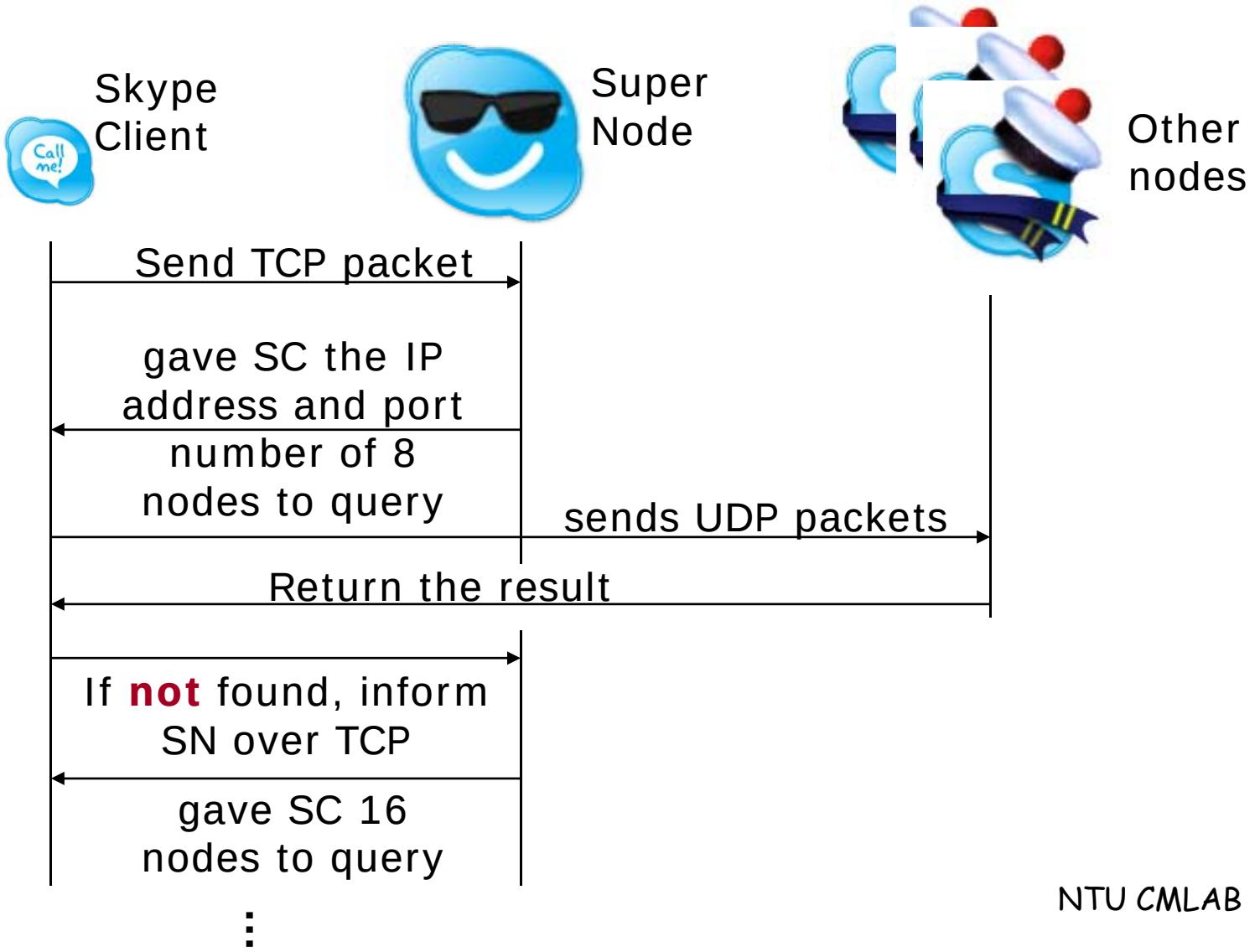
TABLE I

SKYPE (VER 1.4) LOGIN EXPERIMENT SUMMARY

Skype on a Machine with/behind	Data Exchanged	Time to Login
Public IP address	10 KB	3-7 s
Port-restricted NAT	11 KB	3-7 s
UDP-restricted firewall	7 KB	35 s



Skype Functions- User Search

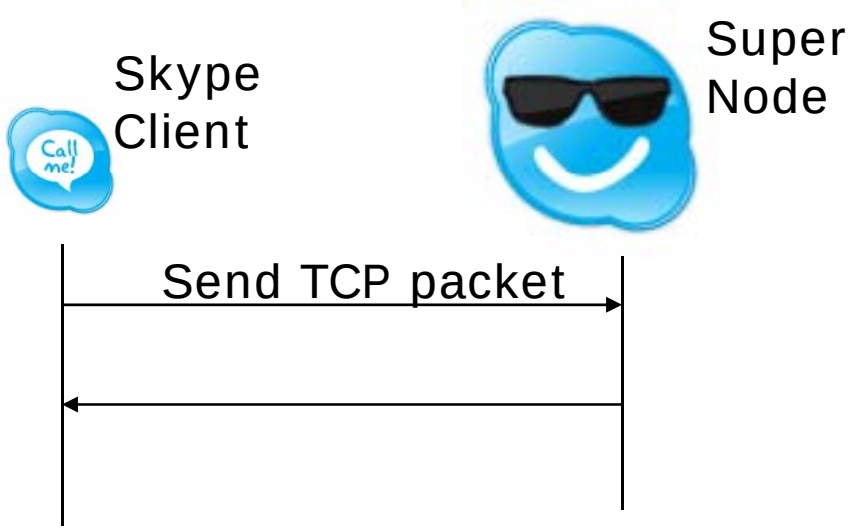


Skype Functions- User Search

- Search results are cached at intermediate nodes.
- Setup1: On average, SC contacted more than 24 nodes. The search took three to four seconds.
- Setup2: This search took about five to six seconds.

Skype Functions- User Search

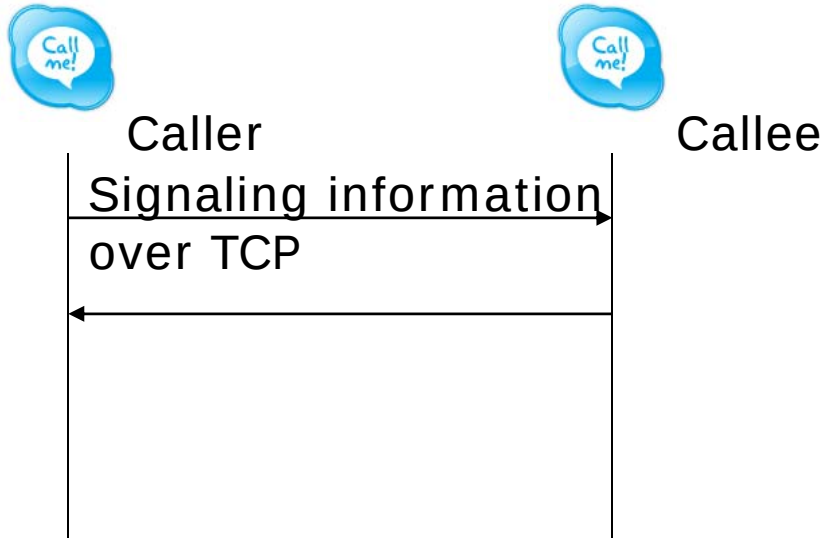
- Setup3: SC did not contact any other nodes. The search took about 10-15 seconds.



- Skype is using the login server as a fall back option in case the search is unsuccessful

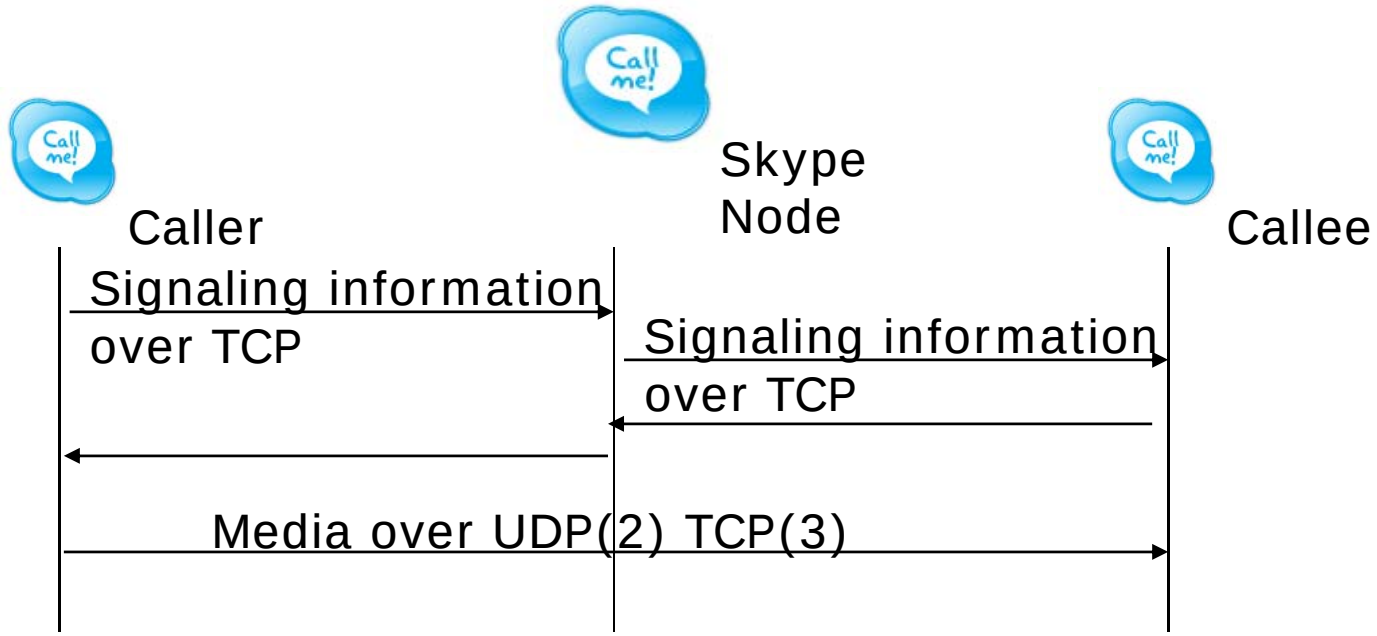
Skype Functions- Call Establishment and Teardown

- Call Establishment
  - Users in the buddy list: call signaling
  - Users not in the buddy list: user search + call signaling
- Caller and callee on public addresses



Skype Functions- Call Establishment and Teardown

- Second Setup: approximately 8 kilobytes of data was exchange.
- Third Setup: approximately 10 kilobytes of data was exchange.



Skype Functions- Call Establishment and Teardown (cont.)

- Having a node route the voice packets from caller to callee.
  - Advantages:
    - provides a mechanism for users behind NAT or firewall
    - Serves as a mixer and broadcasts the conferencing traffic
  - Disadvantages:
    - Lots of traffic flowing
    - Users generally do not want that arbitrary traffic should flow across their machine
- Call tear down signaling was sent over TCP

Skype Functions- Media Transfer and Codecs

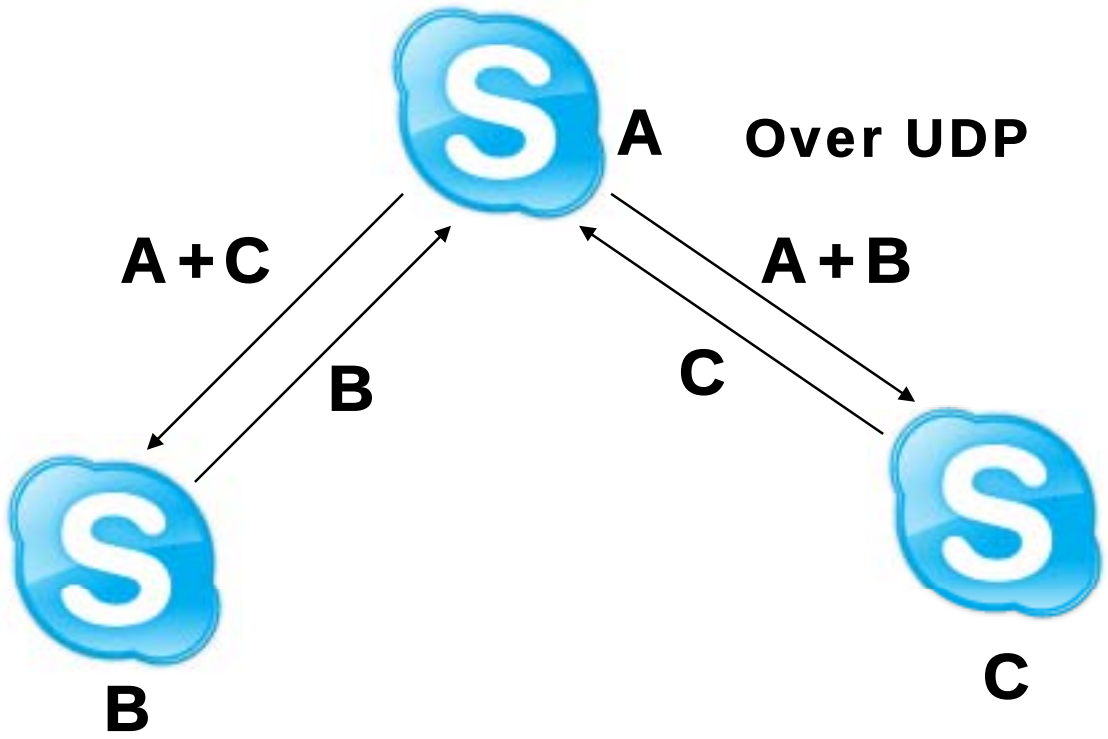
- No silence suppression
  - Maintain the UDP bindings at NAT
  - Used to play some background noise at the peer
  - Purpose: avoid the drop in TCP congestion window size.
- Codecs allow to pass through are 50~8000Hz
- In all three cases, the codec used was iSAC

Outline

- Introduction
- Key components of Skype
- Experiment Setup
- Skype Functions
- **Conferencing**
- Comparisons between IM applications
- Conclusion

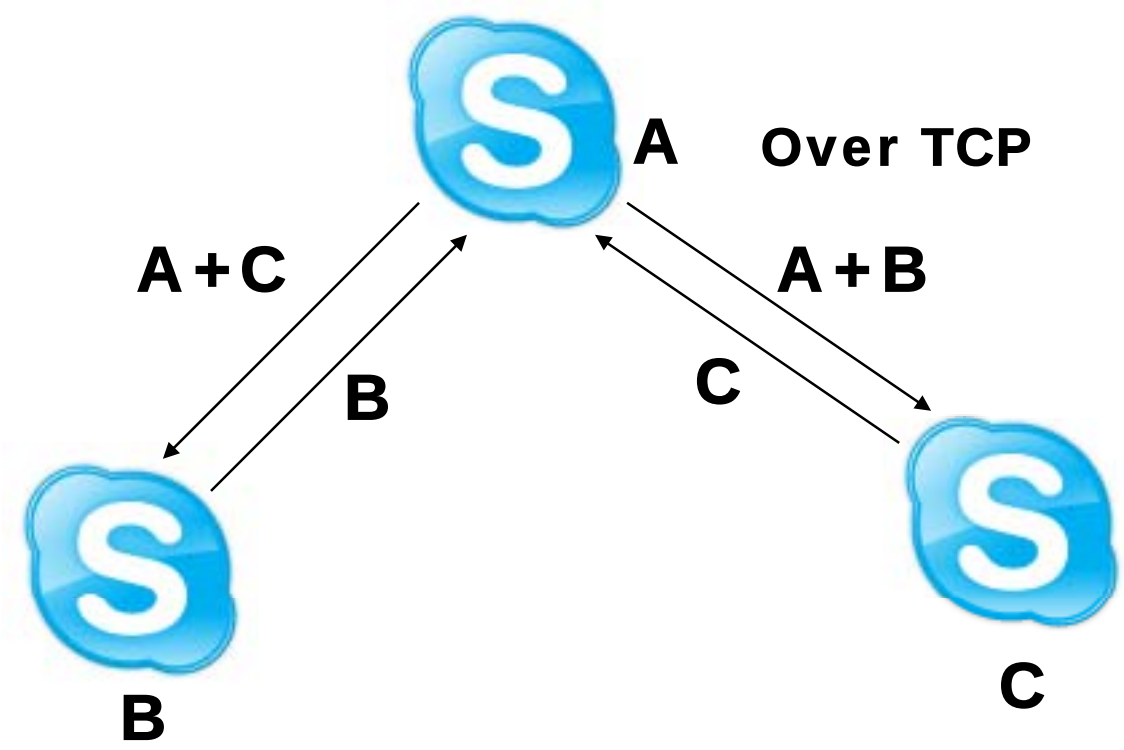
Conferencing

1. Three machines had public IP addresses
2. B and C were behind port-restricted NAT.
   A was on public Internet



Conferencing (cont.)

1. B and C were behind port-restricted NAT and UDP-restricted firewall. A was on the public Internet



Outline

- Introduction
- Key components of Skype
- Experiment Setup
- Skype Functions
- Conferencing
- **Comparisons between IM applications**
- Conclusion

Comparisons between IM applications

TABLE III
SKYPE, YAHOO, MSN AND GOOGLE TALK COMPARISON

	Application version	Memory Usage before call (caller, callee)	Memory Usage during call (caller, callee)	Process priority before call	Process priority during call	Mouth-to-ear latency	Latency Standard Deviation
Skype	1.4.0.84	19 MB, 19 MB	21 MB, 27 MB	Normal	High	96 ms	4
Yahoo	7.0.0.437	38 MB, 34 MB	43 MB, 42 MB	Normal	Normal	152 ms	12
MSN	7.5	25 MB, 22 MB	34 MB, 31 MB	Normal	Normal	184 ms	16
G-Talk	1.0.0.80	9 MB, 9 MB	13 MB, 13 MB	Normal	Normal	109 ms	10

Why skype has fewer latency?

The skype is based on P2P network, so the voice packet transmission is point to point without central server.

Skype is more scalable.

Outline

- Introduction
- Key components of Skype
- Experiment Setup
- Skype Functions
- Conferencing
- Comparisons between IM applications
- **Conclusion**

Conclusion

- Analyze various aspects of the Skype protocol by an analyzing the Skype network traffic and by intercepting the shared library and system calls of Skype.
- Skype can work almost seamlessly across NATs and firewalls.
- Compared to other Talk applications, Skype reported the best mouth-to-ear latency.

Conclusion (cont.)

- Skype relies on SNs. Skype does not allow a user to prevent its machine from becoming a SN.

- Classical packet sniffing tools such as Ethereal are less useful. Shared library and system call interception techniques can be useful.