## Advanced Networking

# P2P Voice Applications

Renato Lo Cigno
Renato.LoCigno@disi.unitn.it

**Credits for part of the original material to Saverio Niccolini NEC Heidelberg**

# The Client/Server model in conversationsl communications

- User-plan communication is nearly always in direct mode, i.e., logically it is P2P
- C/S is used in traditional telephone networks (also over IP) for look-up and signaling
- This requires a very expensive network of servers and resources …
- … it's just like calling always a 12** number instead of looking in your address book!

# From POTS to VoP2P: Step 1

- H.323
  - Tries to reproduce the traditional telephony over a packet, IP-based network
  - Adds services that are not conceivable in traditional telephony (e.g. voice-web integration)

# From POTS to VoP2P: Step 2

- SIP
  - Has the standard “internet” philosophy
  - Move service logic and intelligence in terminals
  - Distributed and flexible (indeed, SIP goes far beyond telephony)
  - Security problems

# From POTS to VoP2P: Step 3

- (Nearly) Server-Less systems
  - Are a transition from the C/S model, which is still rooted in H.323 and SIP) to a service model where each terminal tries to be as autonomous as possible with a flexible hierarchy for seracing services and userse (DNS like) ... The P2P paradigm is the ending point.
  - Without a service provider P2P systems are open and closed at the same time: anybody can be part of a system, but different systems do not talk one another
  - However a communication service either has a monopoly (protocols, formats, syntax – like IP) or there are gateways to cross different service domains

# VoP2P Standardization

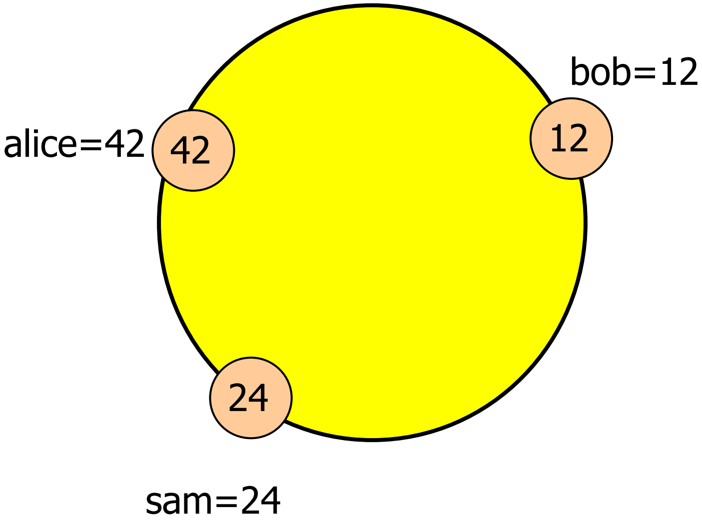
- Many approaches:
  - An interest forum …
  - Pilot products …
  - A Task Force IETF talking to an ITU group talking with …
- But indeed looking into the problem …
  - SIP is compatible with a P2P model
    - Just substitute Registrars and Proxys with distributed data-bases and distributed search engines
  - If the idea is a win-win scenario some idea will be the winner … not necessarily SIPeer

# SIPeer

- Goals
  - P2P standardization project based on SIP primitives
  - 0-configuration
  - Audio and messaging
  - Backward compatible sith standard SIP systems
- Use existing DHT (Distributed Hash Tables) systems
  - Key=hash(user@domain)

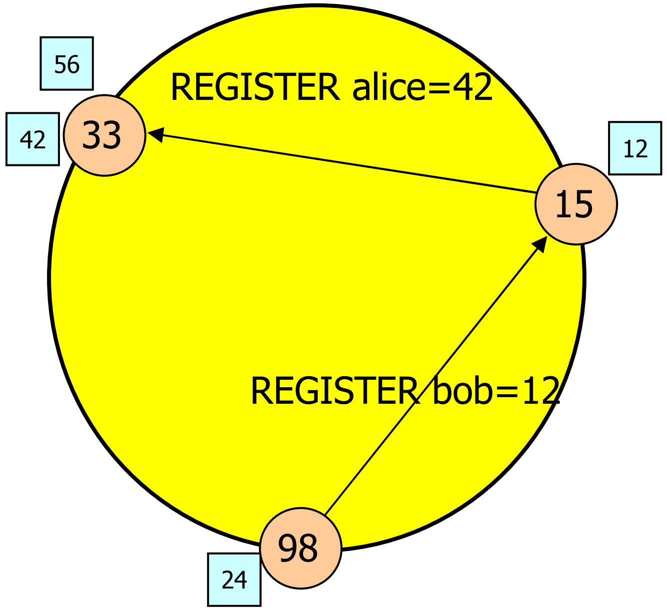
# Users' Search

- Without "REGISTER"
  - The key is computed based on the user ID
  - Nodes enter the P2P overlay with their user ID
  - One node ⇔ One user

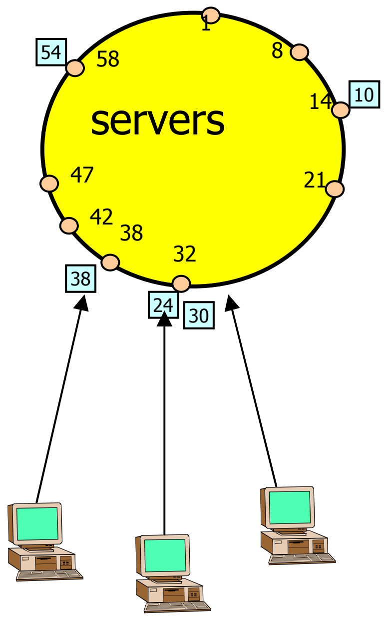



With "REGISTER"

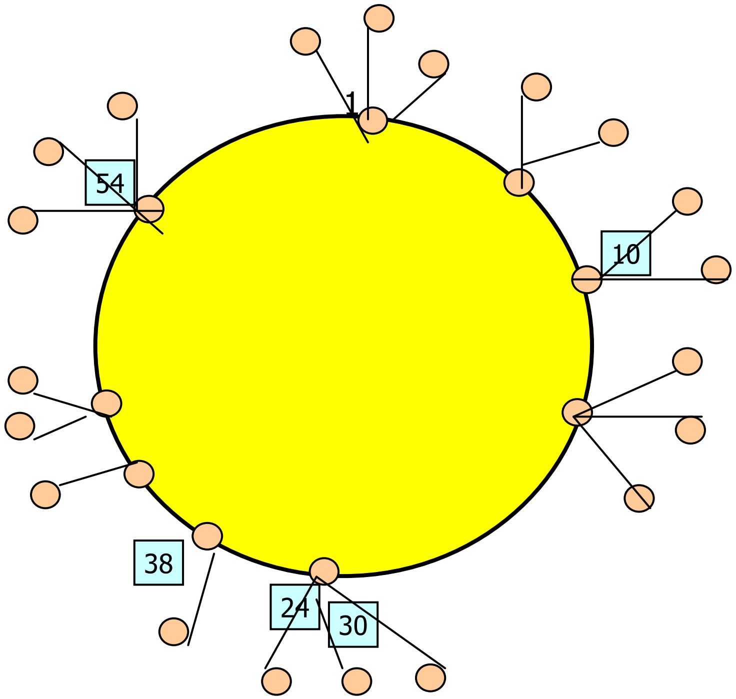
- The users "REGISTER" with some nodes responsible for its key
- Periodic refresh
- Enabled off-line message exchange



# Design alternatives




DHTs are located on distributed, low-cost servers




Hierarchical approach: standard, but stable and powerful nodes maintain the DHT – similar to skype

# P2P real-time: Users perspective

- Ease of usage
- No user configuration required
- Working across all networking environments
  - Network Address Translators (NATs)
  - Firewalls (FWs)
- P2P real-time applications are not standard-based but they “just work”
- Different user experience with respect to standard-based real-time applications
  - e.g. H.323-based or SIP-based

# Identification of issues with P2P SIP

- Goal
  - Identify potential issues of SIP-based P2P communication related to Middleboxes (NAT and firewall) traversal
    - to be considered when designing standards for a SIP-based P2P infrastructure
- Non-Goals
  - Constrain a future P2P SIP architecture in any way
  - Still we need to list potential communication steps that might raise issues
  - Those steps are not necessary part of the final SIP-based P2P solution
  - Suggest NAT traversal methods to be selected for P2P solution

# Potential Communication Steps

- Steps considered
  - middlebox detection
  - registration
  - search for relays
  - address lookup
  - call setup
  - call termination
- Not all steps might be necessary
- Several steps may be combined into one

# Middlebox Detection

- Detect Middleboxes
  - on the signaling path
  - on the data path
- Communication means detection for
  - registration
  - incoming / outgoing signaling
  - data streaming to and from other terminals or relays
- Checks to be performed
  - sending and receiving UDP packets
  - opening incoming and outgoing TCP connections
  - use of certain fixed port numbers
  - the option to relay or tunnel signaling messages and streamed data
- NAT parameter detection
  - full cone, half cone, etc…

# Registration

- Authentication of the user
- Notification of communication capability and willingness
- Registration of contact parameters
- Notification of service provisioning capability and willingness

# Further Steps

- Search and Connect Relay
  - Candidate relays may be suggested by infrastructure
- Address Lookup
  - Per-call lookup
  - Buddy list lookup
- Connection Establishment and Termination

# Middlebox Traversal Methods

- **Tunneling**
  - in highly restricted environments only
  - controversial:
    - HTTP and DNS tunneling are not legitimate
    - TURN could be OK
- **Network-initiated Middlebox Signaling**
  - not the right choice for P2P SIP
- **Terminal-initiated Middlebox Signaling**
  - several methods known

# Terminal-initiated Middlebox Signaling

- **Standards**
  - **STUN (IETF RFC3489)**
  - **UPnP (UPnP Forum)**
  - **SOCKS (IETF RFC 1928)**
  - **RSIP (IETF RFC 3103)**
- **Under development**
  - **STUN update (IETF behave WG)**
  - **ICE (IETF mmusic WG)**
  - **NSIS (IETF nsis WG)**
- **Middlebox traversal using relays**
  - **STUN relay (previously TURN) (IETF mmusic WG)**

# Open Issues for SIP-based P2P

- SIP-unrelated
  - middlebox detection beyond UDP
- SIP-related
  - terminal reachability
  - communication service requirements
  - communication service offers
- The relevance of these issues strongly depends on the choice of P2P architecture

# Middlebox Detection Beyond UDP

- Limited or no middlebox detection for TCP and DCCP (Datagram Congestion Control Protocol) available
  - Middlebox signaling for TCP is covered by UPnP, SOCKS, RSIP, NSIS
- TCP considered for signaling and for data
  - Several SIP-signaled services use TCP
  - RTP over TCP used when UDP is blocked
- Might get solved partially by ICE TCP
  - still in early state

# Terminal Reachability

- **Relevance depends on registration and relay detection process**
- **Terminal might need to register first and then find and connect to a relay in order to be reachable**
- **In between these two steps it would be reachable for signaling but unreachable for data transmission and should be registered as such**
- **Currently, the SIP protocol does not provide explicit means for signaling such a state**

# Communication Service Requirement

- The terminal might need to express its needs for relaying
  - signaling messages
  - lookup requests
  - data streams
- Infrastructure nodes might need to suggest relays to be used by terminal
- For both, request and suggestion, signaling means are required
  - Extension Header Field for Service Route Discovery During Registration (RFC 3608) might offer means

# Communication Service Offering

- A terminal in an unrestricted (or just slightly restricted) environment might be able (and the user willing) to offer services to other peers, such as relay services and lookup services
- Currently, the SIP protocol does not provide explicit means for signaling such offers

# P2P SIP: how to locate peers?

- **Basic idea is that what you are looking for has an identifier**
  - **Locate items in the overlay based on the identifier**
  - **Distributed Hash Table (DHT), Content Addressable Networks (CAN)**
  - **Since “everything has its place”, eliminate false negatives**
  - **Since you can go (close to) directly to the item you want, more efficient**

# Applying this to SIP

- **Use pure Distributed Hash Tables (DHT) to find the other UAs**
  - **Problems**
    - **currently no DHT standardized**
    - **some firewalls block DHT traffic as “file sharing”**
- **Use DHT for location, but implemented as SIP messages**
  - **Essentially, use DHT as another registration/location mechanism**
- **Use standard SIP to signal once resources are located**

# Problems with P2P SIP

- **Like most things SIP, NATs**
  - **Same problems, plus some new ones**
  - **Super nodes?**
- **Security**
  - **Sybil attacks**
  - **DoS (through traffic and true denial)**
  - **Encryption**
  - **Information “leakage”**
  - **Choosing node locations to divert/block**

# Advanced Networking

# Skype

Renato Lo Cigno
Renato.LoCigno@disi.unitn.it

**Credits for part of the original material to Saverio Niccolini NEC Heidelberg**

# Skype characteristics

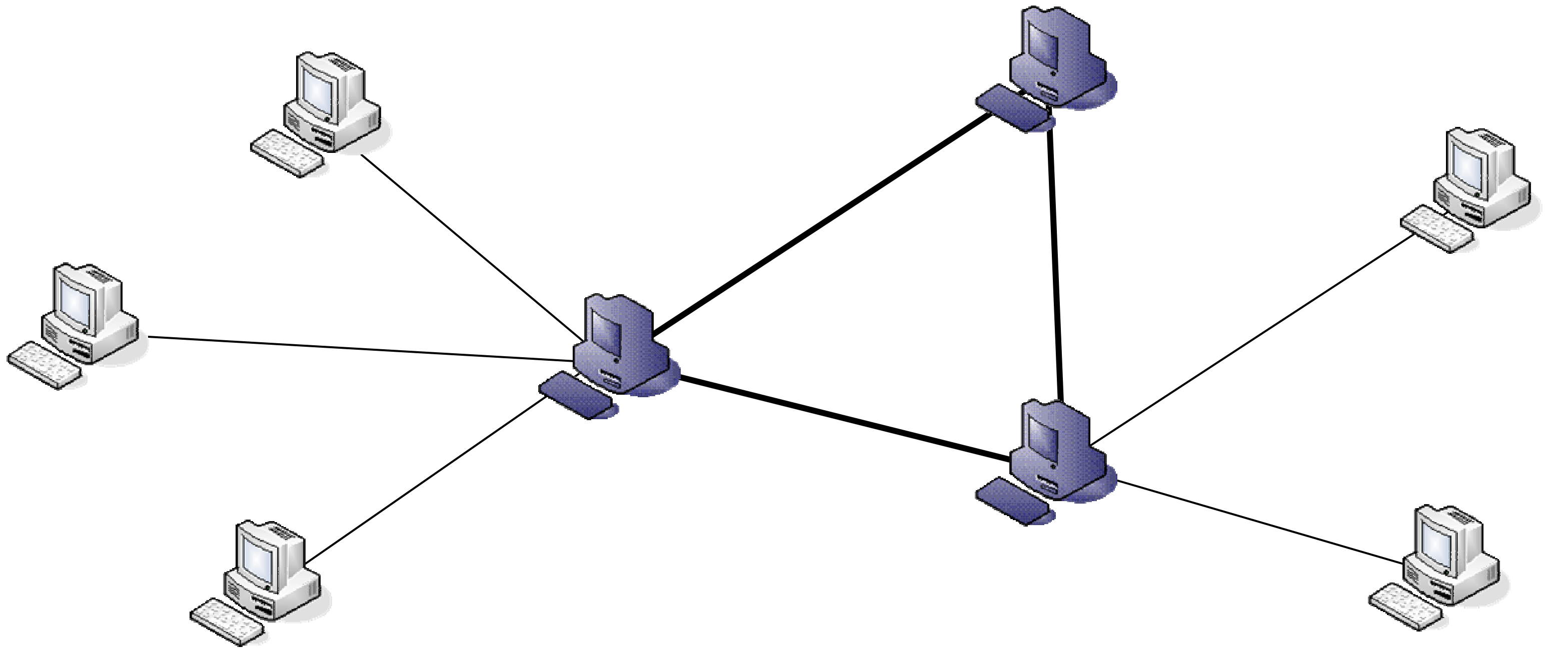
- Skype is a well known P2P program for real time communications
  - Voice calls
  - Video (from version 2.0)
  - File sharing and instant messaging when in a call
- Seems to work with no problems in all network conditions compared to similar P2P applications
- One of the reasons of its success is its ability to work in network scenarios with middleboxes
  - such as firewalls and Network Address Translators (NATs)
  - usually, this is a problem for P2P applications

# How Skype works

- **Skype overlay network**
  - network structure
  - entities involved
- **Skype function analysis**
- **Lesson learned**
- **Skype security analysis**
  - Binary
  - Network protocol
  - Skype authentication
  - Traffic encryption

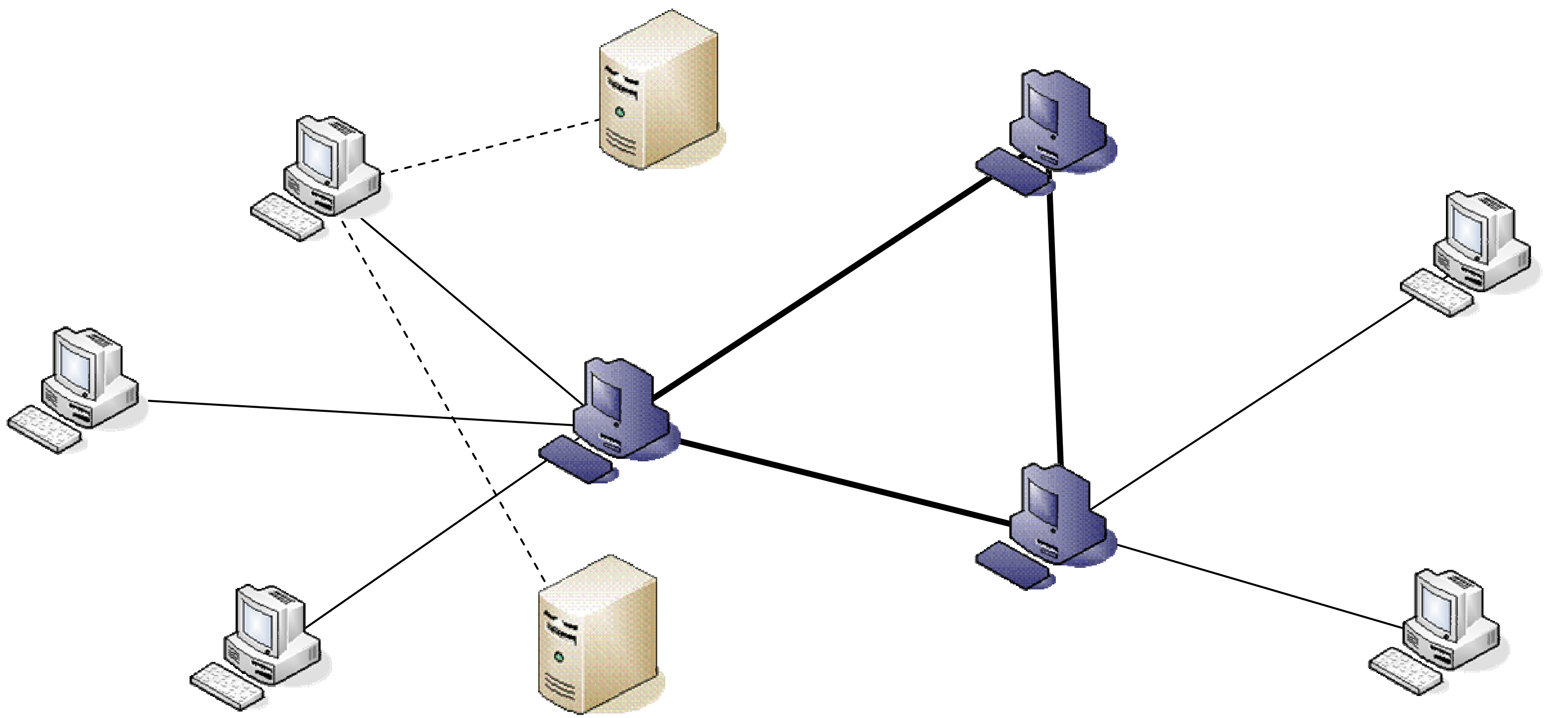
# Skype overlay network (I)

- **Skype network relies on distributed nodes:**
  - **Skype Clients (SCs)**
  - **Supernodes (SNs)**



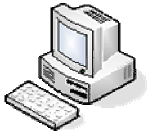
# Skype overlay network (II)

- **Although there are also centralized entities:**
  - **HTTP Server**
  - **Login Server**



Renato.LoCigno@disi.unitn.it

# Skype overlay network (III)



## Skype Client

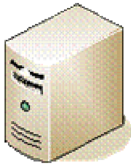
- used to place voice calls and send instant messages
- connection to skype network possible through a supernode (SN)
- connection with the SN (via TCP) maintained for the whole time the client is on-line
- client configuration and SN addresses are stored locally and refreshed periodically to maintain a coherent view of Skype network

# Skype overlay network (IV)

## Supernode

- Normal Skype Client that can accepts incoming TCP connections, with enough CPU, memory and bandwidth
- There are also a number of "default" Supernodes, used to increase network robustness and stability

# Skype overlay network (V)

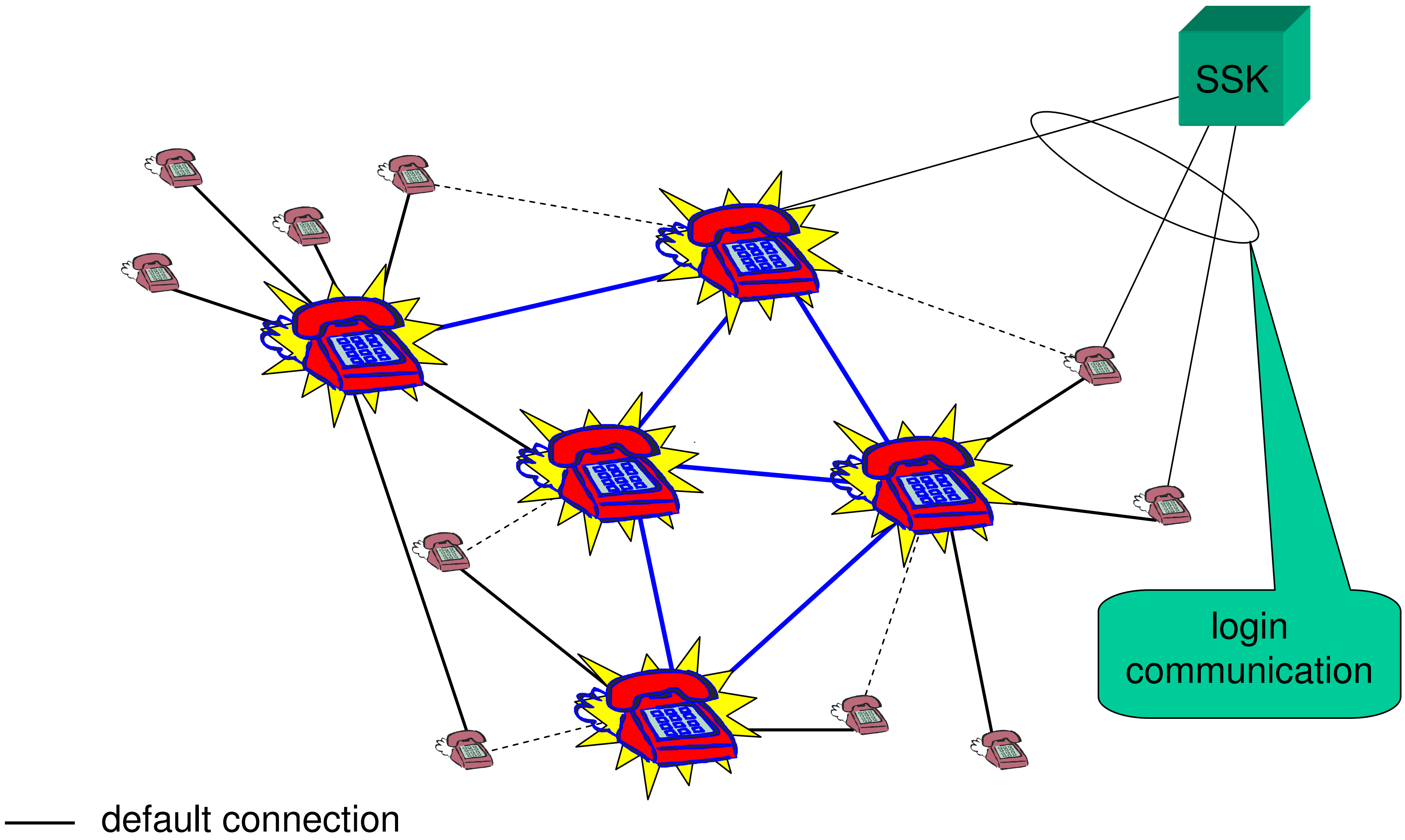


## Servers

- Login server ensures that names are unique across Skype namespace. Also central point for authentication
- HTTP Server used by clients to check for updates

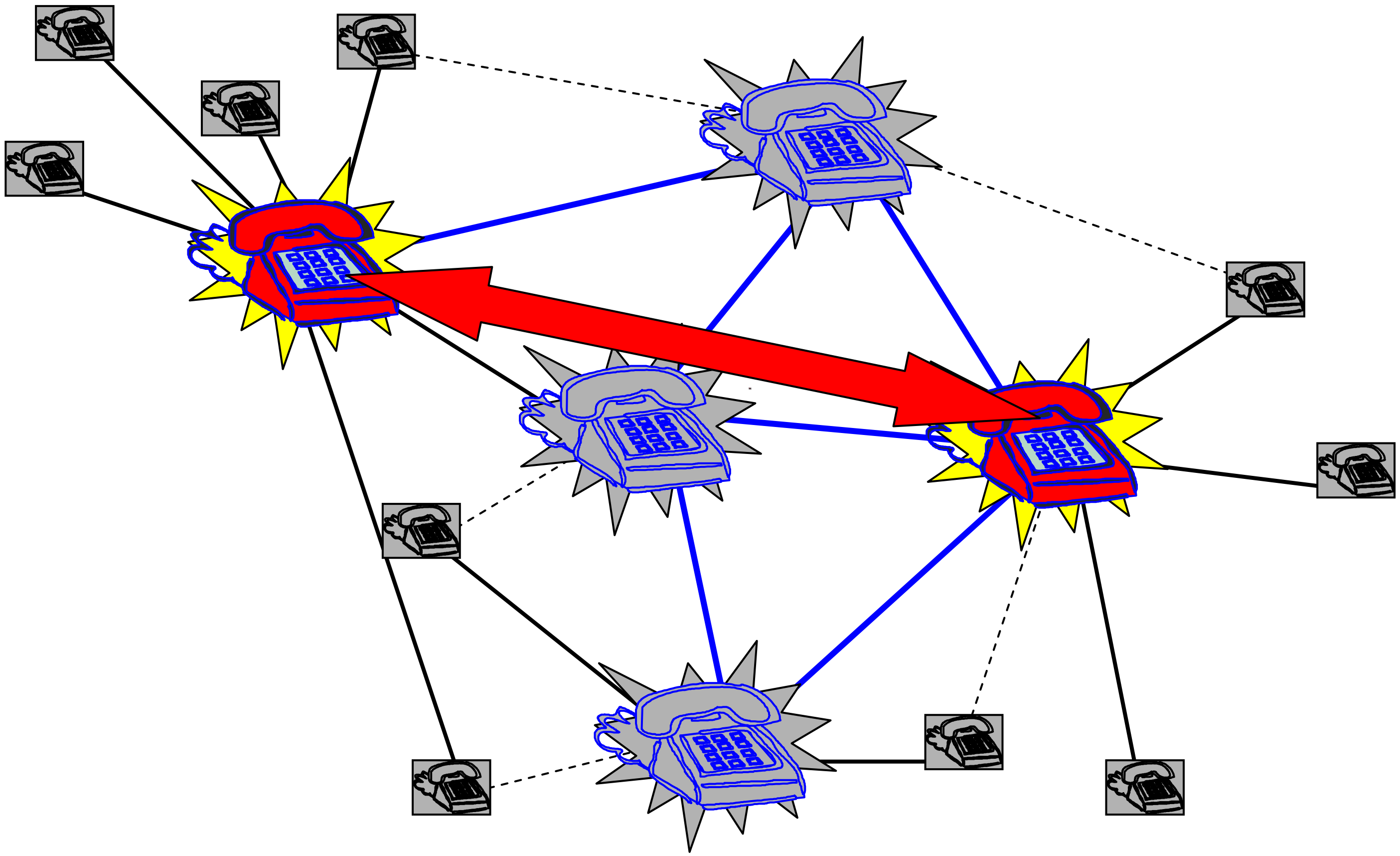
Renato.LoCigno@disi.unitn.it

# Topology



# Topology: calls

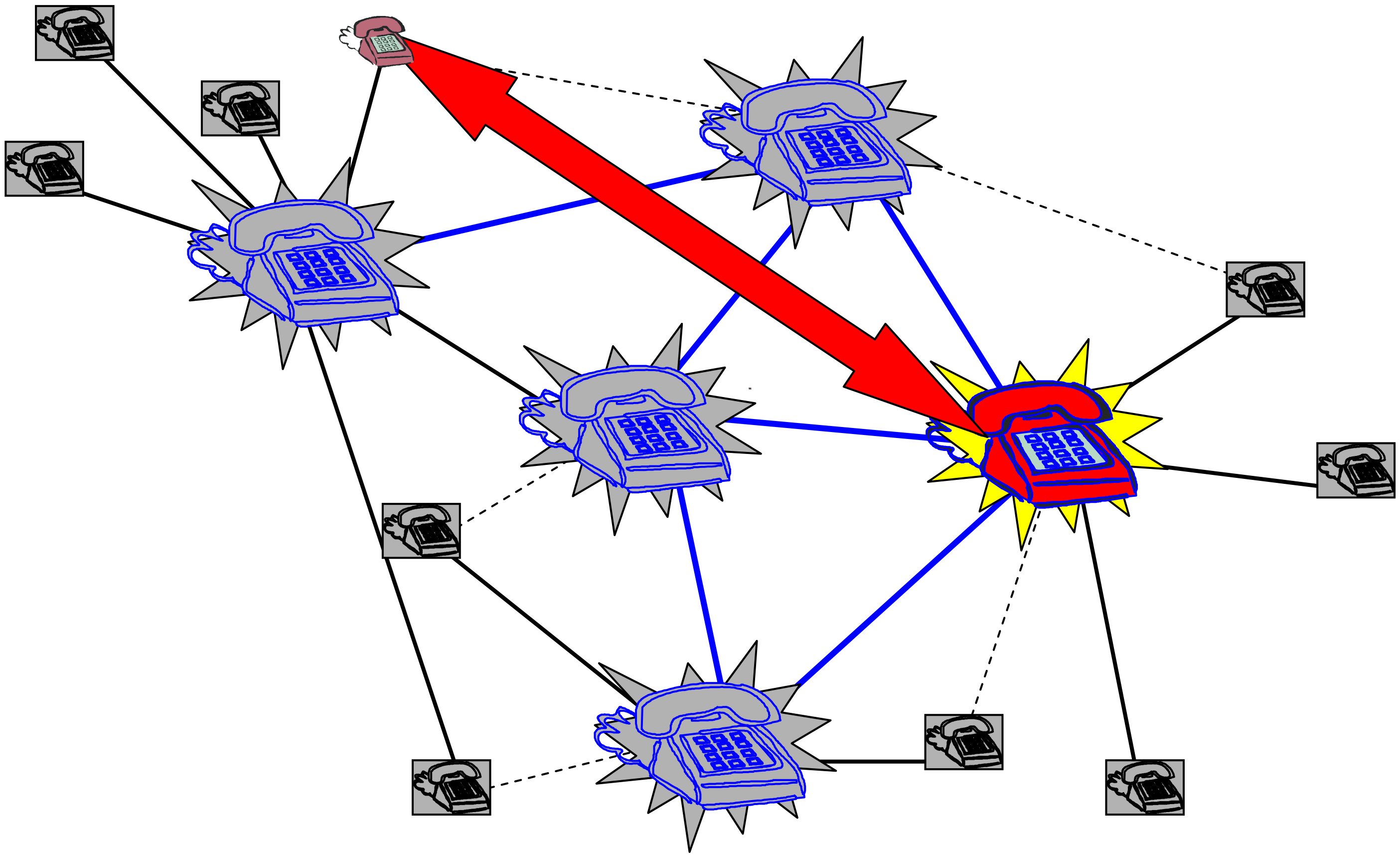
**Supernodes communicate directly ...**



Renato.LoCigno@disi.unitn.it

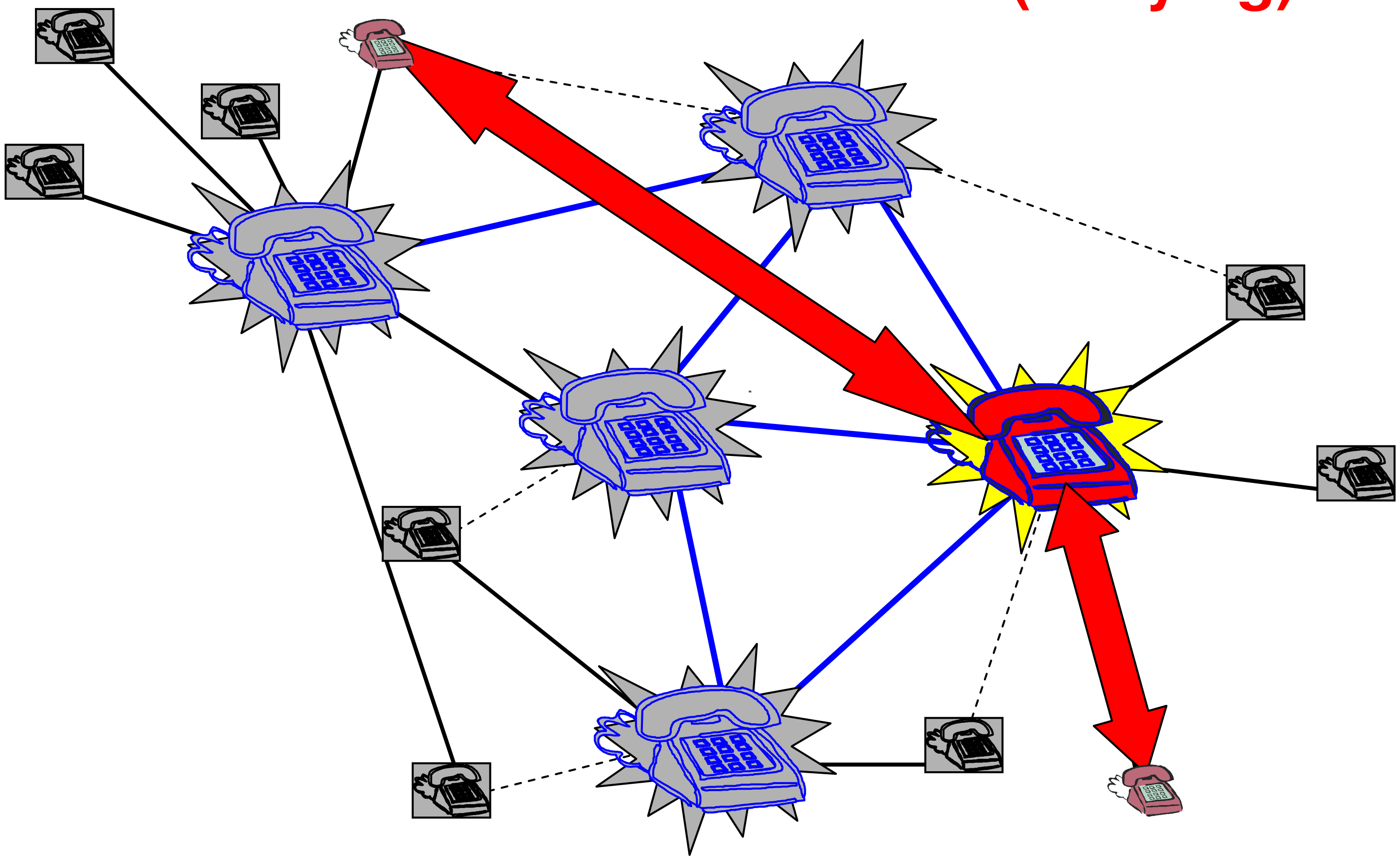
# Topology: calls

**... also with normal peers**



# Topology: calls

normal nodes require a supernode intermediation (relaying)



Renato.LoCigno@disi.unitn.it

# Some caratteristics

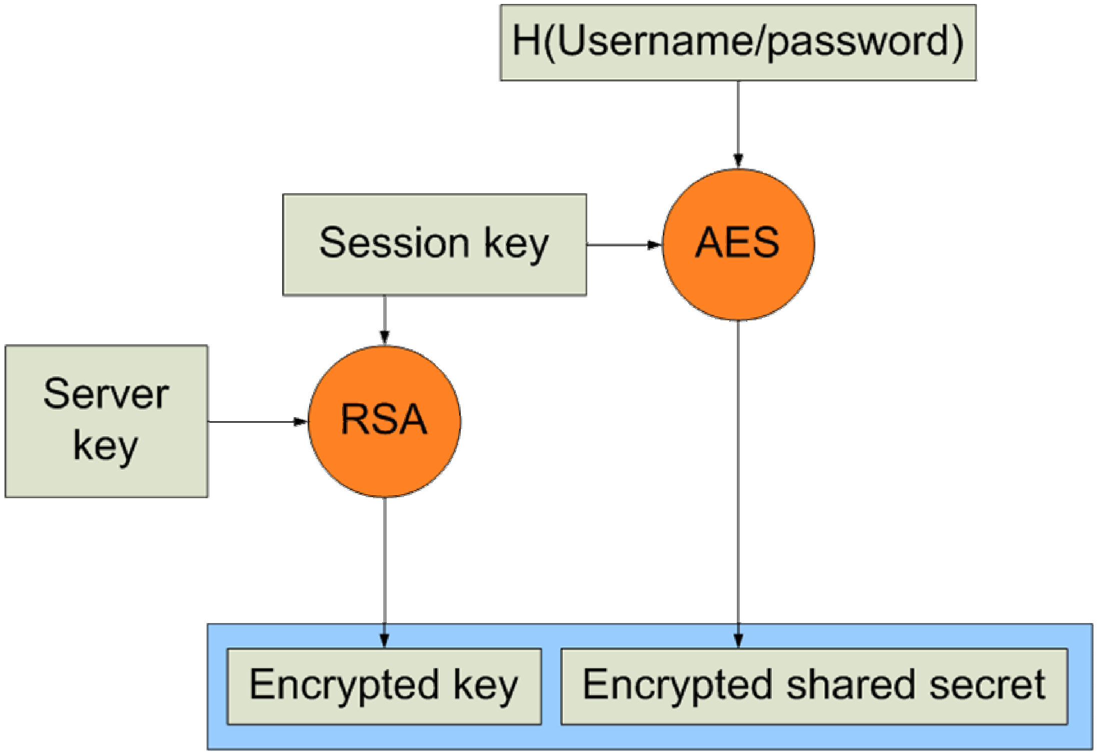
- CODECs
  - Default is a wideband (8 kHz-16kHz sampling) resulting in a transmission rate of 40 kbit/s in each direction (140 pck/s with payload of 67 bytes)
  - Quality in normal conditions is very good, much better than PCM telephony
  - No narrowband coding is provided, congestion is not considered a problem generated by skype
  - Under lab conditions over UDP the system works well even with only 16--20 kbit/s; below 12 kbit/s the system cannot work

# Some Characteristics

- Ports
  - 80 (HTTP) e 443 (HTTPS) on TCP for signalin, random choice on UDP or TCP for voice
  - Ports are announced on the P2P network
- Encryption
  - All communications are AES (Advanced Encryption Standard) encoded

# Skype Encryption

- **Authentication**
  - **At login time the client generate a RSA session key and uses it to encrypt his credentials.**
  - **Then encrypts the session key using the server's public key**
  - **and sends this information to the login server**



# Some Characteristics

- Host Cache
  - List of supernodes (IP, Port) used to make the search phase faster
  - Roughly 200 entries dynamically updated
  - If the host cache is void skype does not work (some defaults entry are there from the beginning)
  - Une of the critical points for skype functioning
  - The idea is not new to P2P networks and answer to the bootstrap problem … albeit in a naive way

# Skype functions analysis

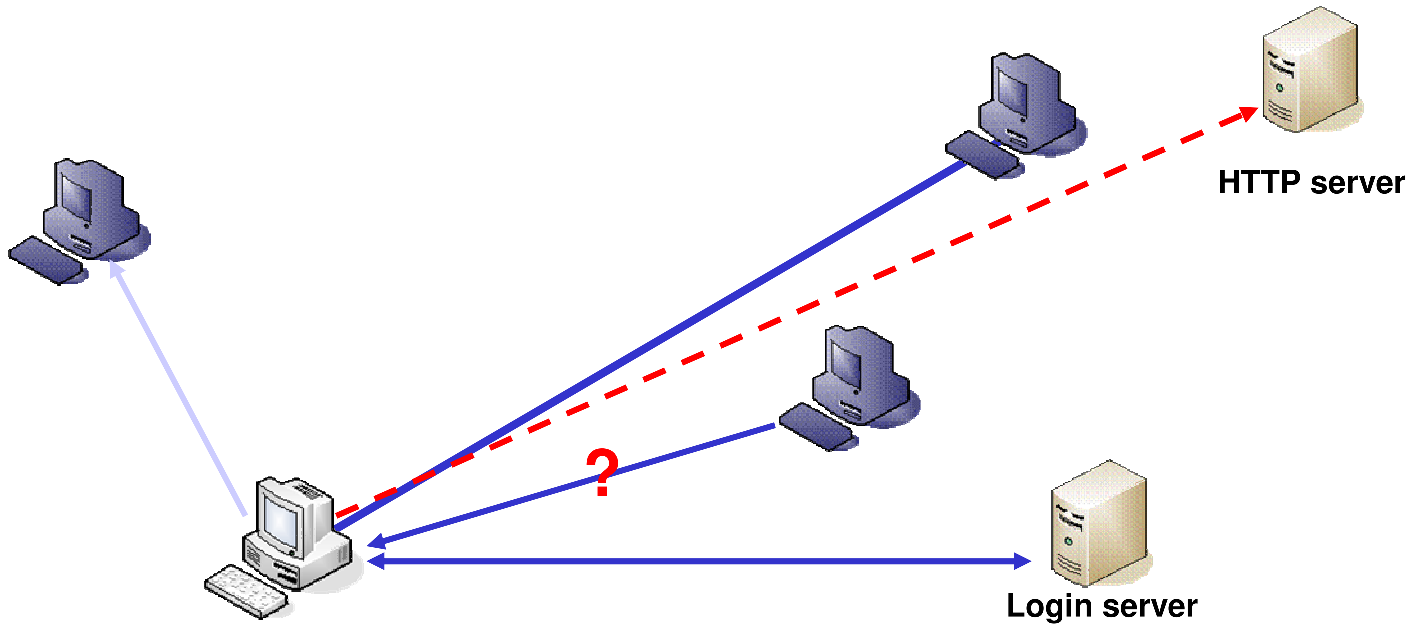
- Essentials
  - Login
  - Search
  - Buddy list signaling
  - Call establishment

# Login function

- **Join and maintain overlay network:**
  - **Interaction with central servers**
    - **login server manage authentication and ensures unique names**
    - **HTTP server ensures client software updates**
  - **Refresh of shared.xml**
    - **file stored on the client containing SNs list and parameters identifying middlebox**
  - **Network tests if joining client can act as a SN**

# Login procedure

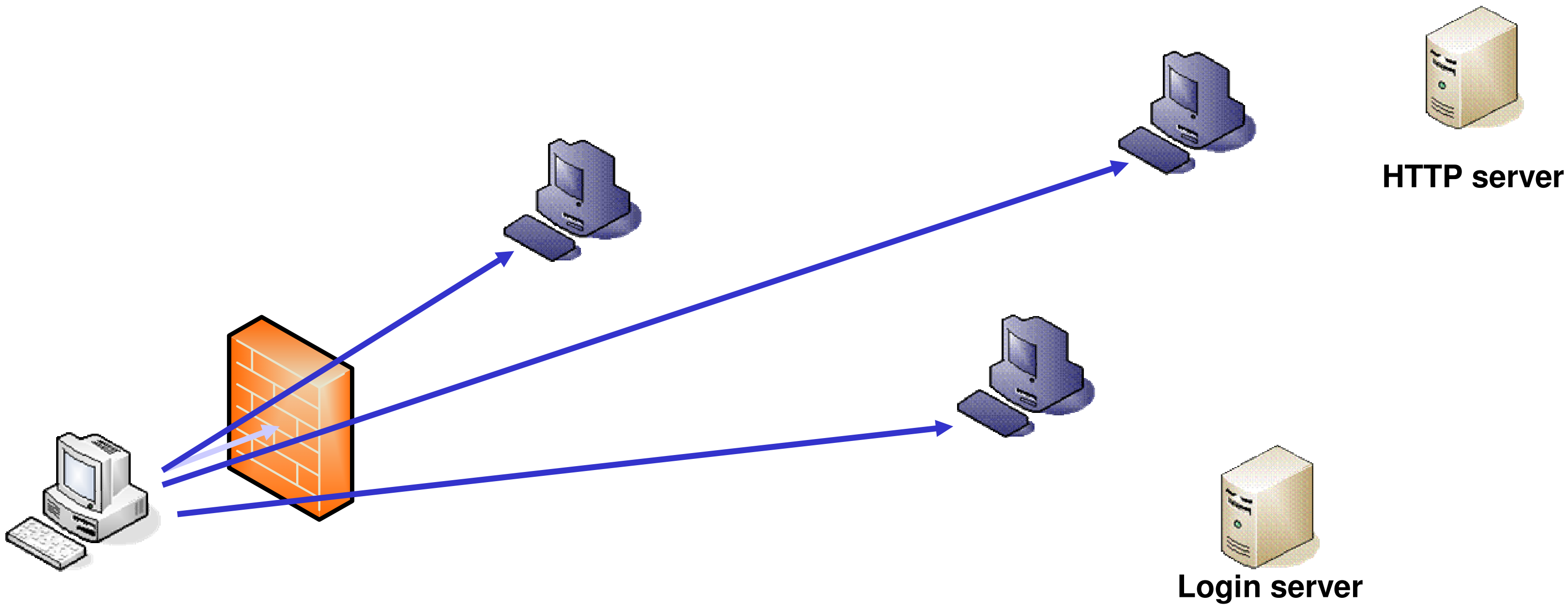
- At startup the client contacts the HTTP server to check for updates
- Sends UDP datagram to a -default SN- to refresh the list of supernodes
- Connects via TCP to a SN (connection maintained throughout Skype session) and exchanges info on online nodes
- Verify username and password via TCP with the Login server
- Another SN tests if client can act as a SN




Renato.LoCigno@disi.unitn.it

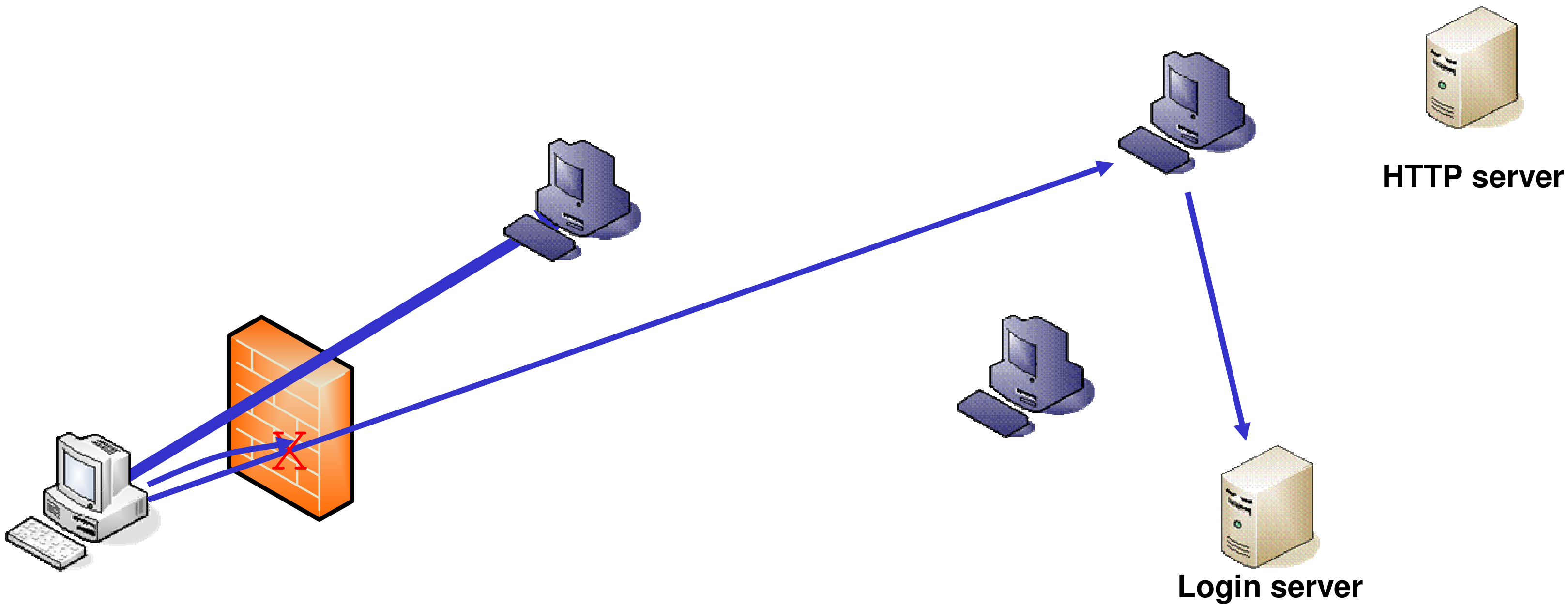
# Login: Firewall blocks UDP

- Firewall prevents UDP exchange for SN list refreshing
- Client establishes several TCP connections with SNs to gather information, when finished all but one are torn down



# Login: Firewall blocks Login sever

- After connection with the SN, attempt to connect with the Login server fails
- Client connect to the Login using a SN as a relay



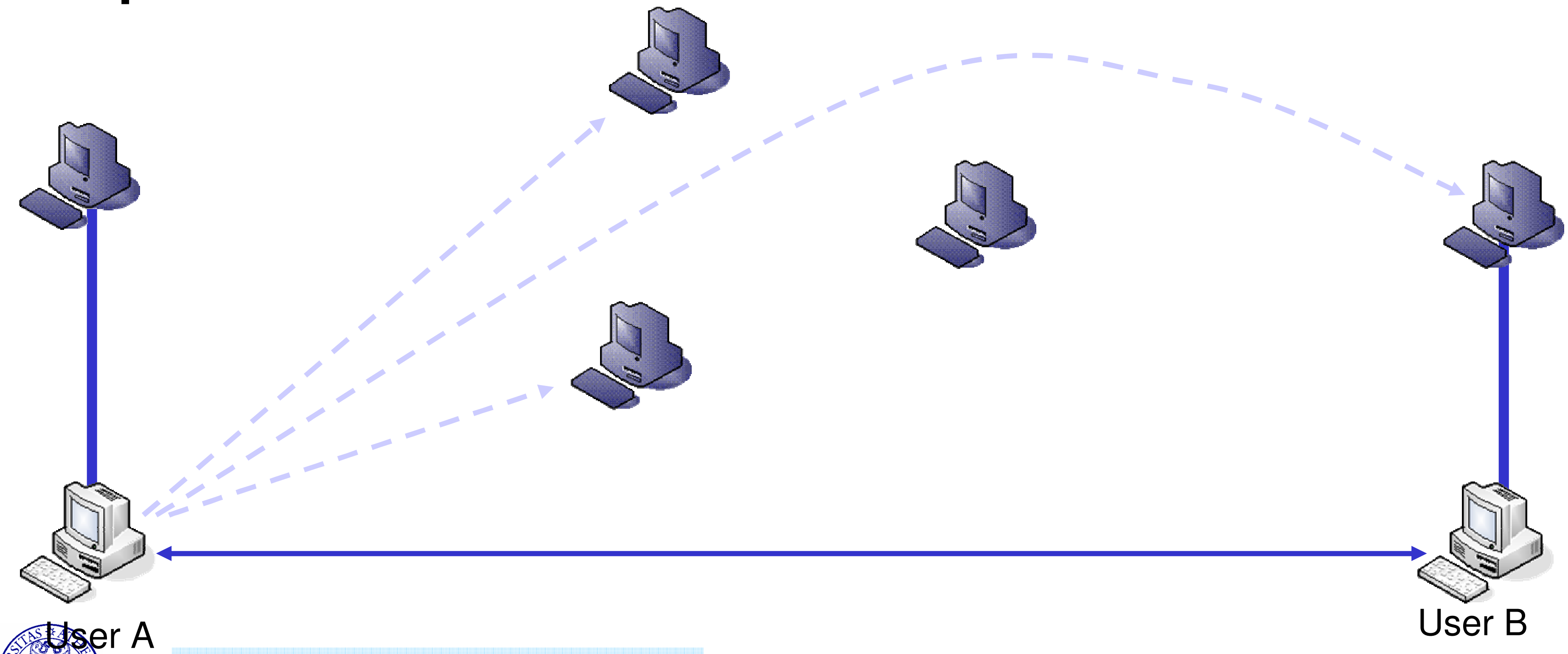
# Search function

- Procedure performed when a user wants to add someone to his buddy’s list and communicate for the first time

- Search is performed using username as key
  - possible since names are unique
  - this is why there is the need for central servers

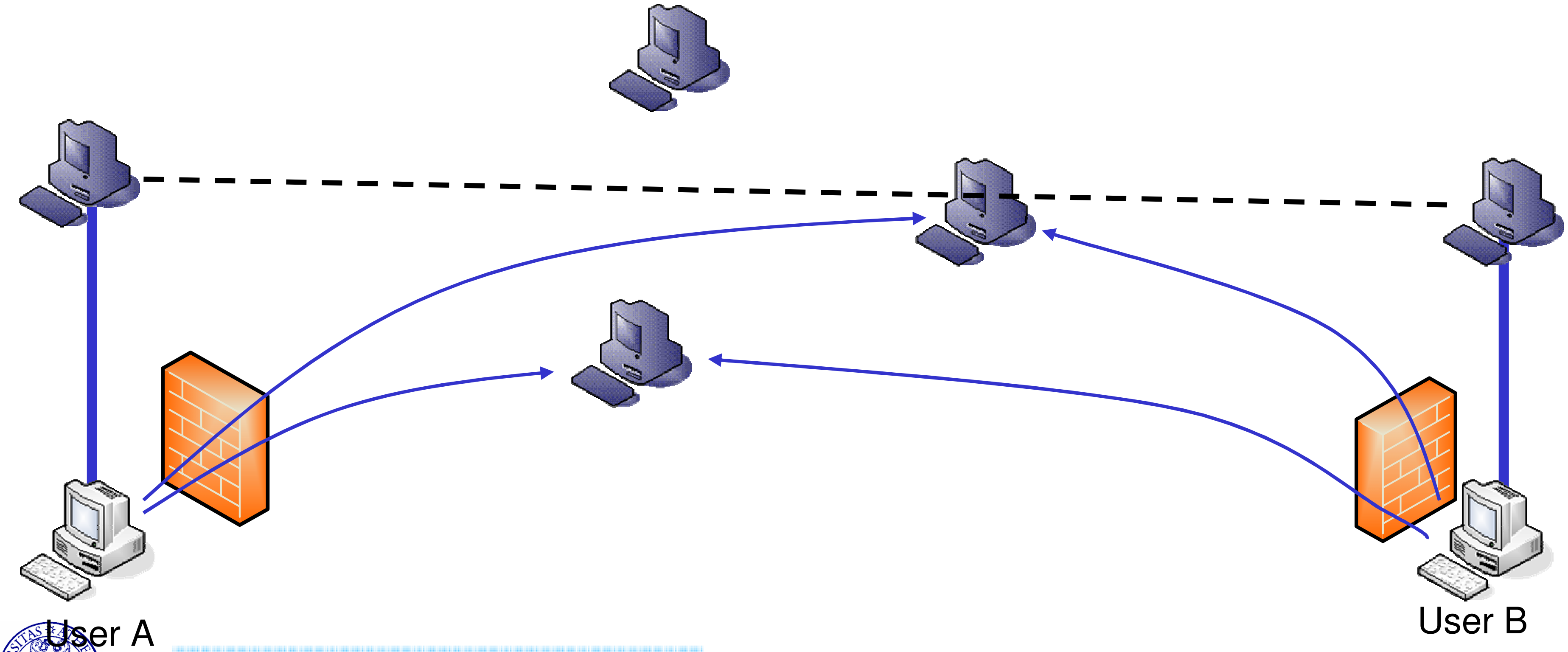
# Search procedure

- User A exchanges info with its SN and gather 3 SNs addresses
- A query the 3 SNs via UDP asking if they know the public IP of B
- Once A gets the address of B authorization exchange is performed



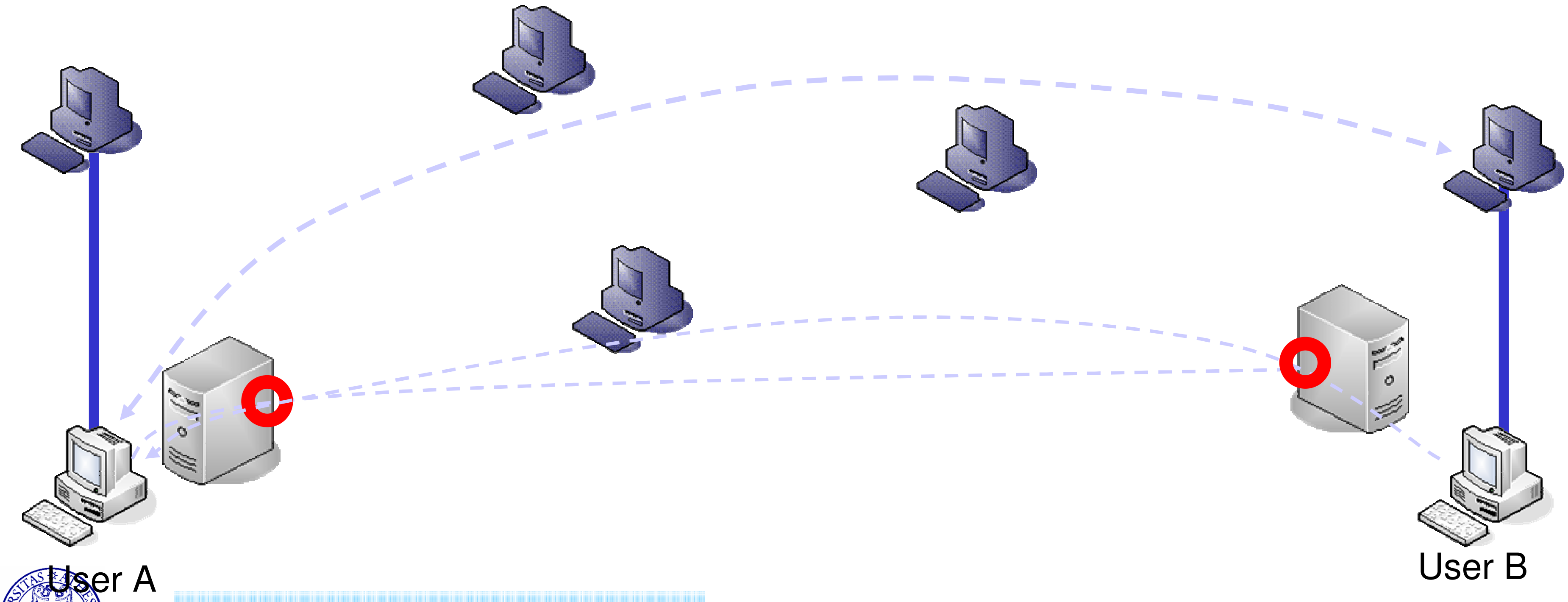
# Search: Firewall blocks UDP

- **Firewall blocks UDP**
  - preventing direct connection w/ the SNs or another user
  - the SN of A communicate to B (via his SN) the address of A
- **Both A and B establish TCP connections with the same 2 SN to exchange authorization**



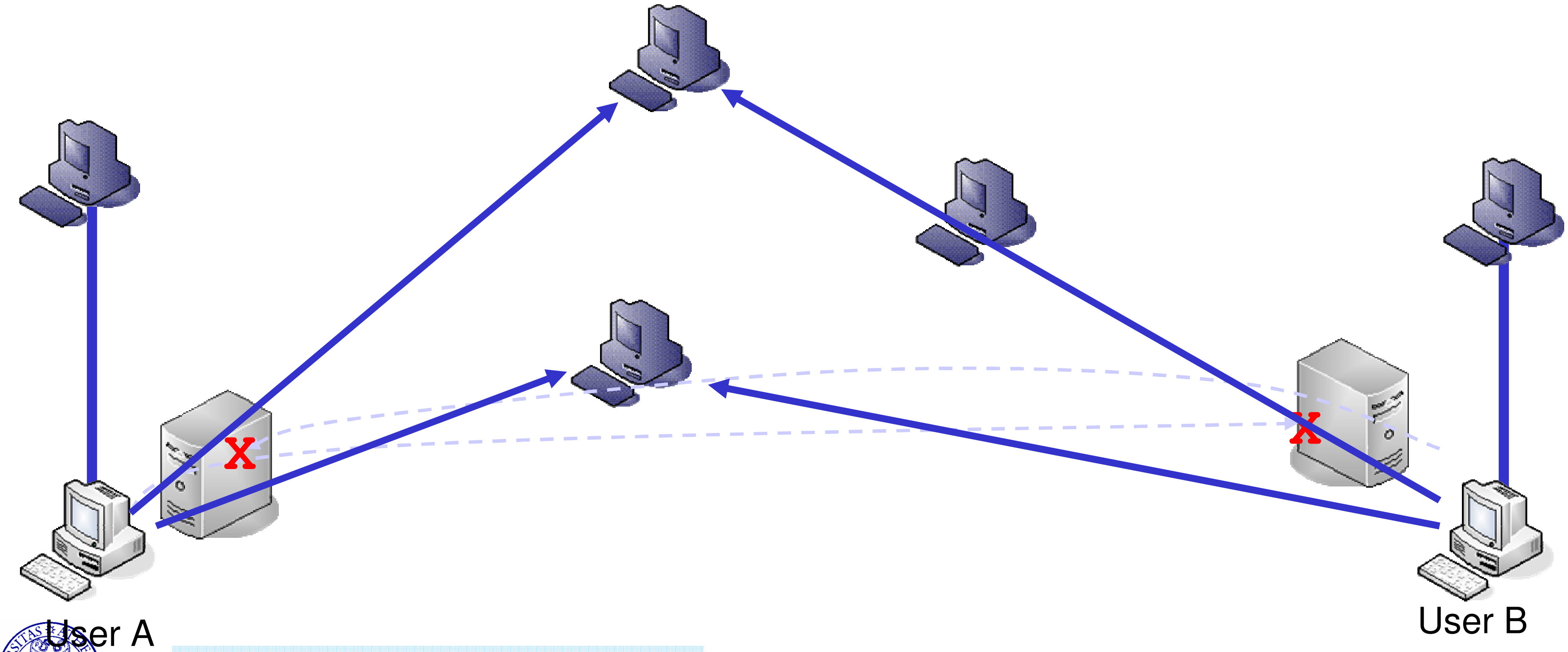
# Search: Port restricted NAT

- Once user A gather the address of SN of B, sends a UDP query containing his external address. SN of B replies with user B external address.
- User A send an UDP datagram to user B external address in order to create a mapping in his NAT, anyway packet will be filtered by NAT of B
- User B does the same but this datagram reaches user A
- Once exchanged authorization a TCP connection via 2 SNs as relay is established, as depicted in previous slide



# Search: Symmetric NAT

- **Clients try the technique depicted for Port restricted NAT**
  - **but it fails due to symmetric NAT behvior**
- **Clients exchange authorization via TCP using 2 SNs as relay**




Renato.LoCigno@disi.unitn.it

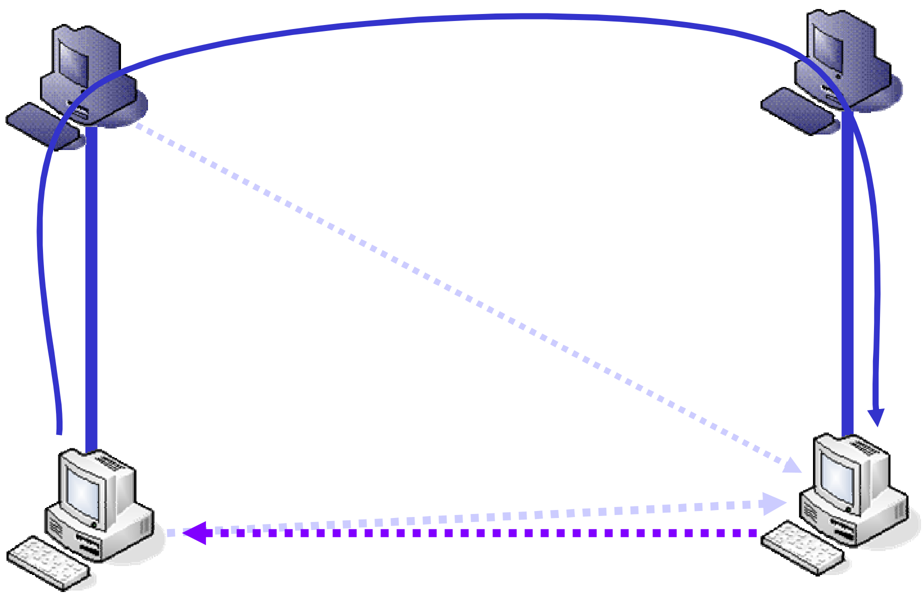
# Buddy list signaling

- Buddy list is a list of “friend” users
- Skype allow a user to know if buddies are online/offline
  - overlay network informs buddies when user change status

Renato.LoCigno@disi.unitn.it

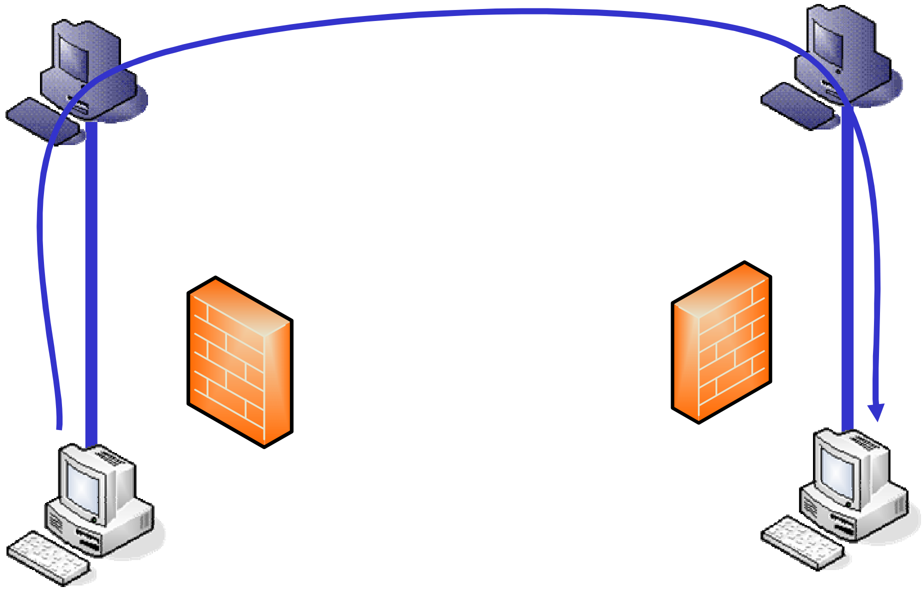
# Buddy List signaling procedure

- A user going on-line informs his buddies either directly using UDP or via the SNs.
- When going off-line, a user tear down the TCP connection with the SN.
- The SN informs via UDP the buddies that the user is going off-line
- To have a confirmation buddies try to ping the user.



# Buddy List signalling: Firewall blocking UDP

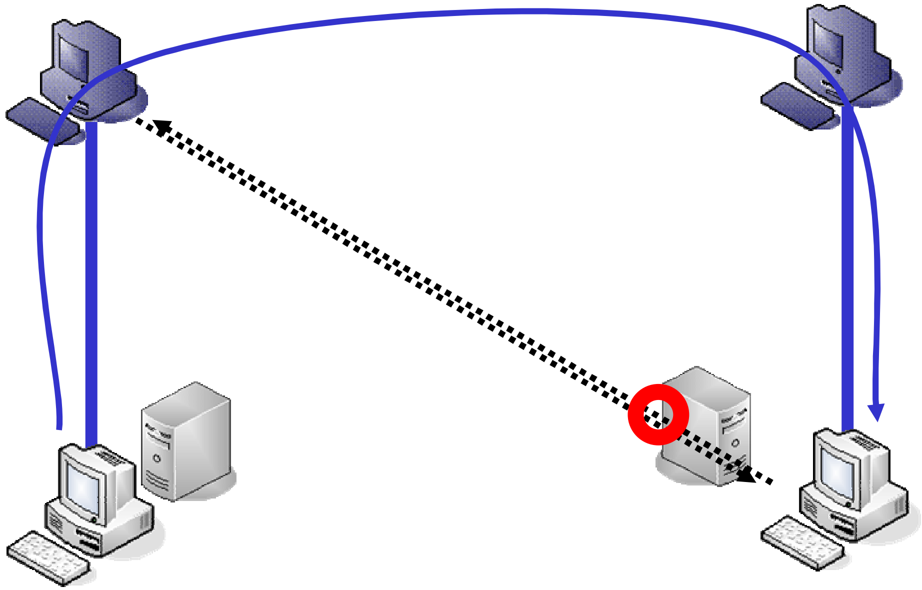
- **Since UDP traffic is blocked, on-line/off-line signalling is performed via the SNs**



Renato.LoCigno@disi.unitn.it

# Buddy List signaling: Port restricted NAT

- On-line/off-line signaling is performed in a way similar to that depicted in previous slide.
- As a difference after the change of status, buddies query the SN of the user for confirmation.

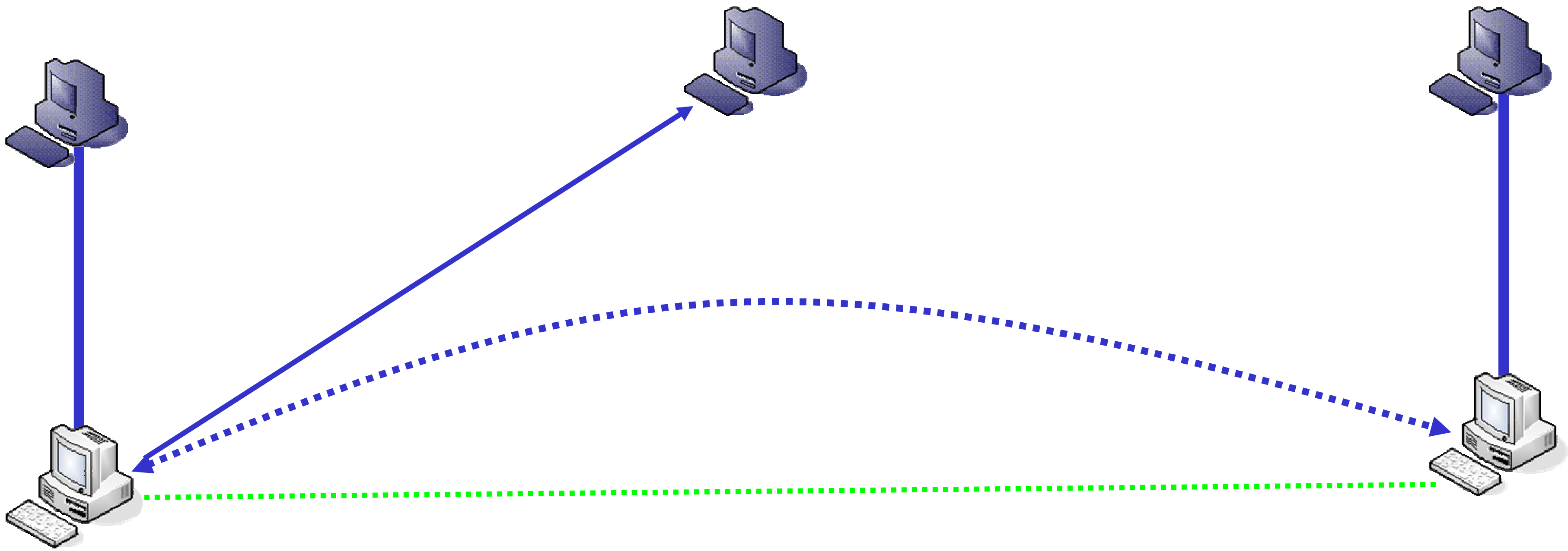


# Call establishment function

- **Signaling performed using TCP connection**
  - **overlay network used only if otherwise impossible**
- **Media carried over UDP when possible**
  - **in case relay servers are used**

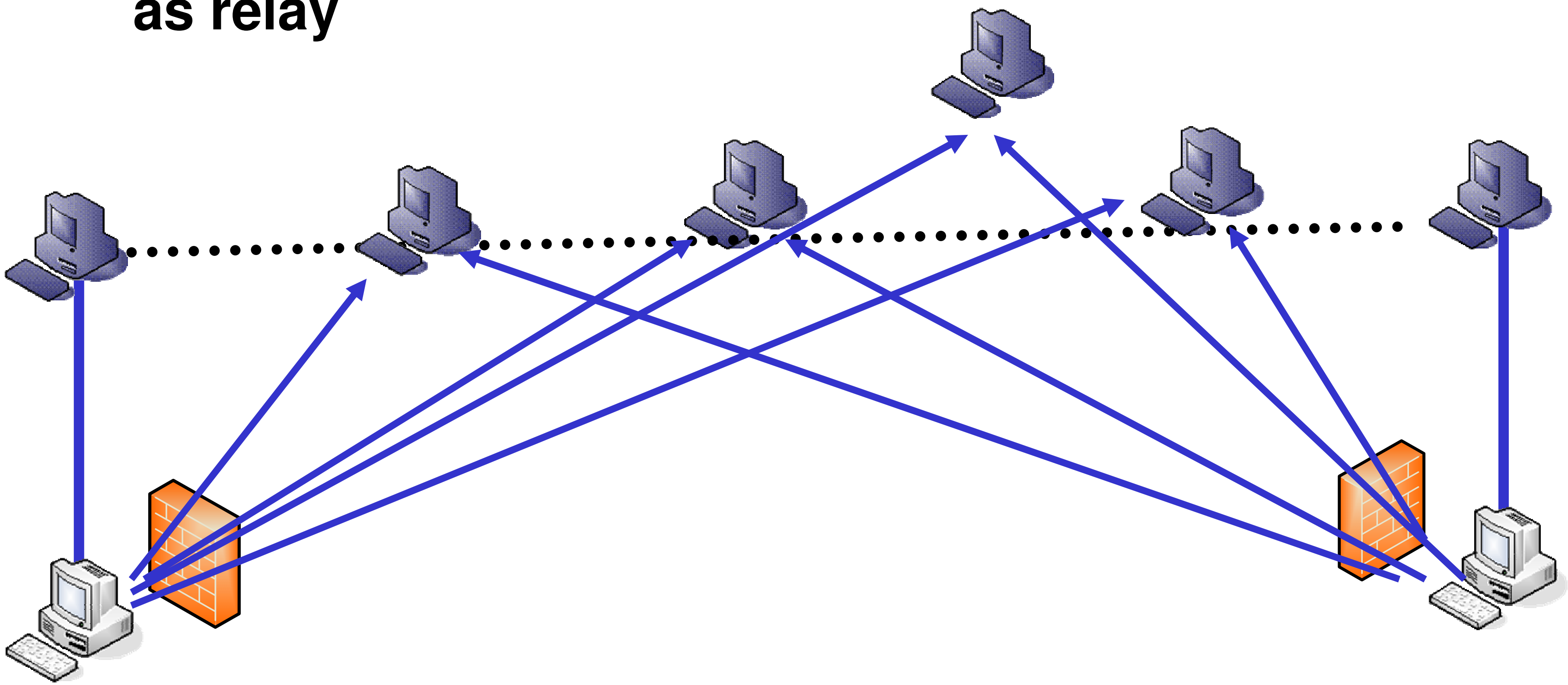
# Call establishment procedure

- User A wants to call user B, so he query some SNs for user B address.
- Once he gets user B address they exchange signaling over TCP
- Voice traffic carried via UDP



# Call establishment: firewall blocks UDP

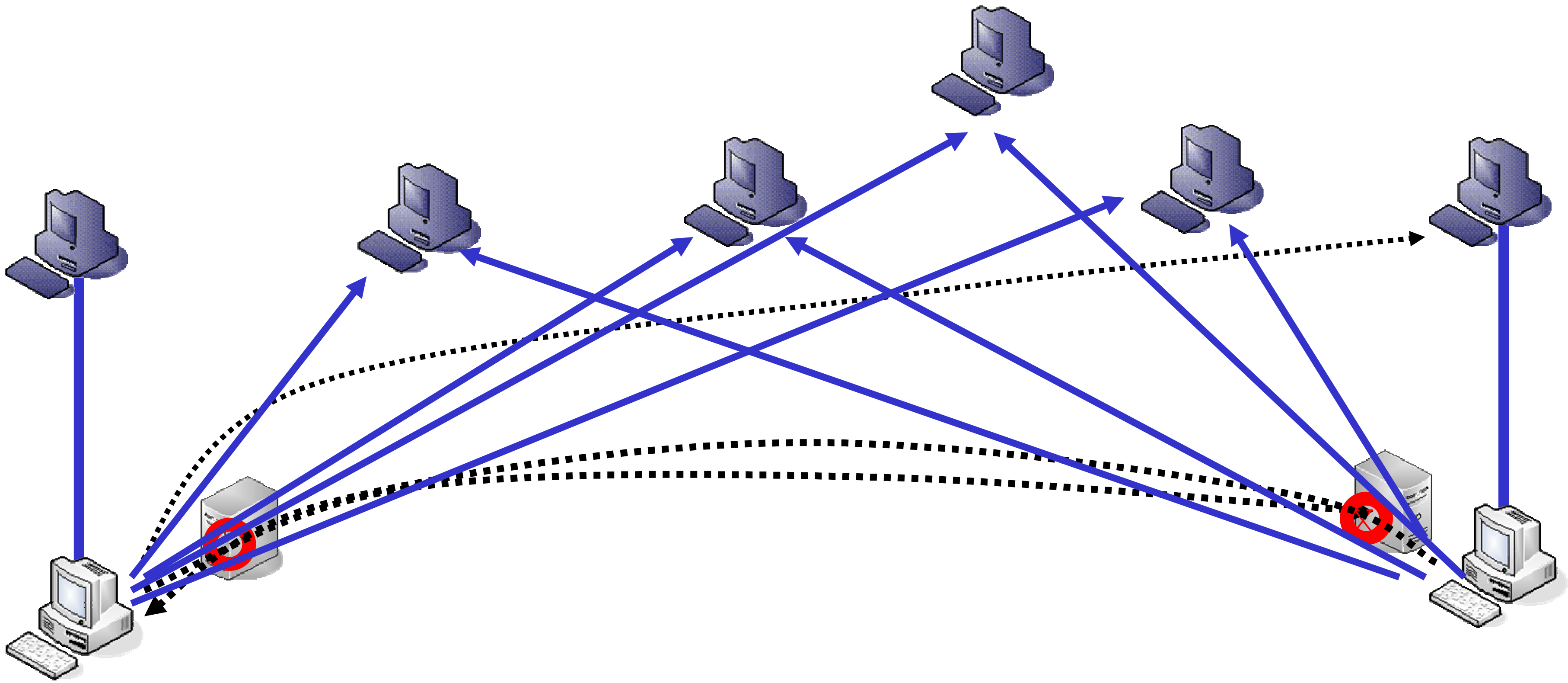
- Signaling exchanges are performed by the SNs on behalf of the users
- Media exchange is performed via TCP using 4 SNs as relay



Renato.LoCigno@disi.unitn.it

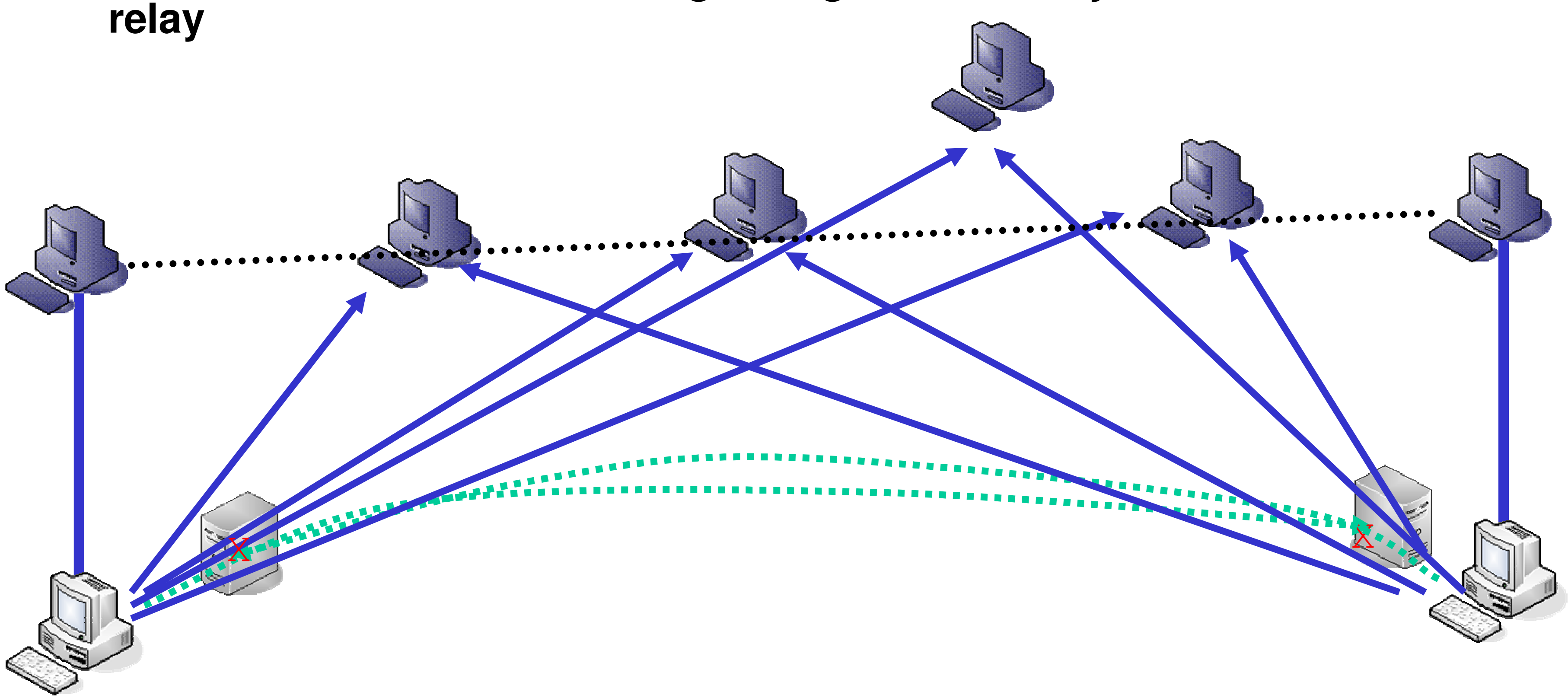
# Call establishment: Port restricted NAT

- Once User A gets the address of the SN responsible for user B he queries for his address. SN informs B that user A wants to call him, and tells external address of B to A.
- A and B establish UDP flow using reverse hole punching
- They also establish TCP connection using 4 SNs as relay



# Call establishment: Symmetric NAT

- User A and B communicate their addresses via their SNs
- They try reverse hole punching but it won't work because of NATs restrictions
- To establish the media and signalling channel they will use 4 SNs as relay



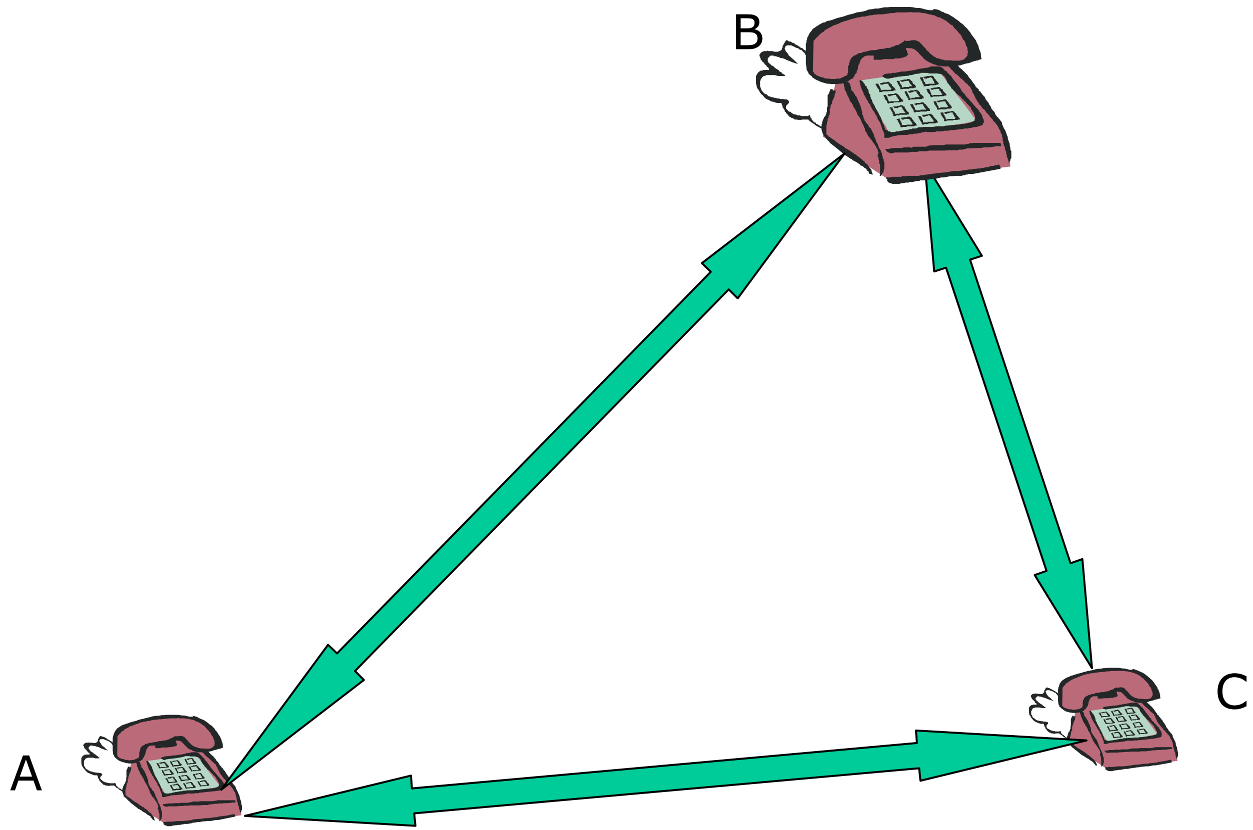
# Lesson learned

- Traversal is well possible in many cases without explicit signaling to the middlebox
  - open public access network
  - protected enterprise networks
- Reverse hole punching and tunneling techniques workarounds allow Peer-to-peer communications in almost every scenario
  - Skype only fails completely if firewall blocks TCP but in fact that is a very uncommon case
- Explicit middlebox signaling protocols (like IETF MIDCOM MIB, CheckPoint OPSEC, NEC's SIMCO) are still required for
  - highly protected access network
  - applying security policies by network operator
  - anyway Skype will undermine many of these policies
- Skype tries to use IP network instead of overlay
  - SNs can't assure constant presence
  - avoid overlay congestion

# Audio Conference

- Based on traffic mixing in one of the nodes
- Limited to few nodes (5-6)
- Works also with some nodes behind NAT/FW
- The mix node is elected based on it elaboration capabilities, since mixing is CPU intensive
- It does not need to be the conference initiator

# Audio Conference: signaling



# Audio Conference: audio flows