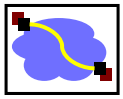
15-441 15-641 Computer Networking

Delivering Content: Peer to Peer
Peter Steenkiste

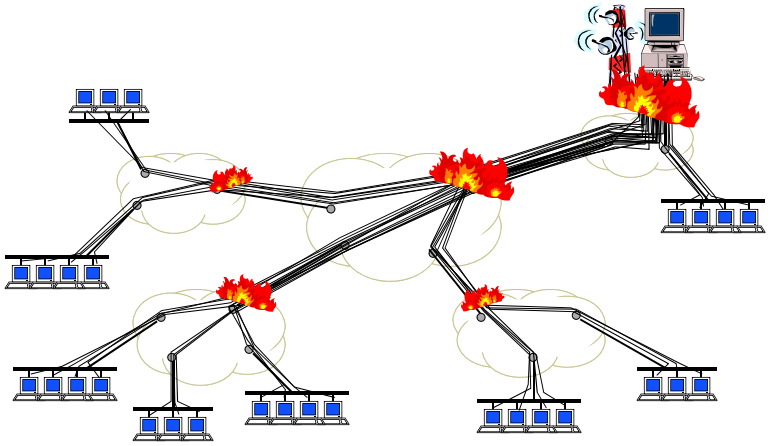
Fall 2015
www.cs.cmu.edu/~prs/15-441-F15

Overview



- Web
- Peer-to-peer
  - Motivation
  - Architectures
  - BitTorrent
  - TOR
  - Skype
- CDN
- Video

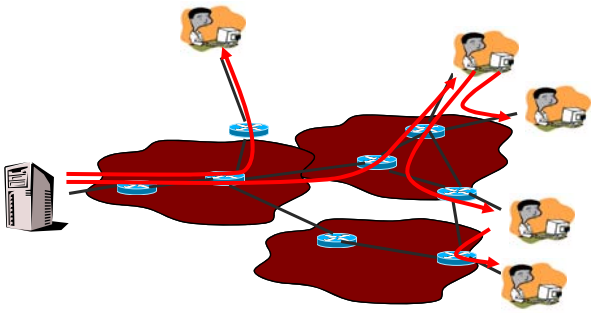
Scaling Problem



- Millions of clients ⇒ server and network meltdown

3

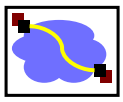
P2P System



- Leverage the resources of client machines (peers)
  - Computation, storage, bandwidth

4

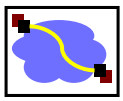
Why p2p?

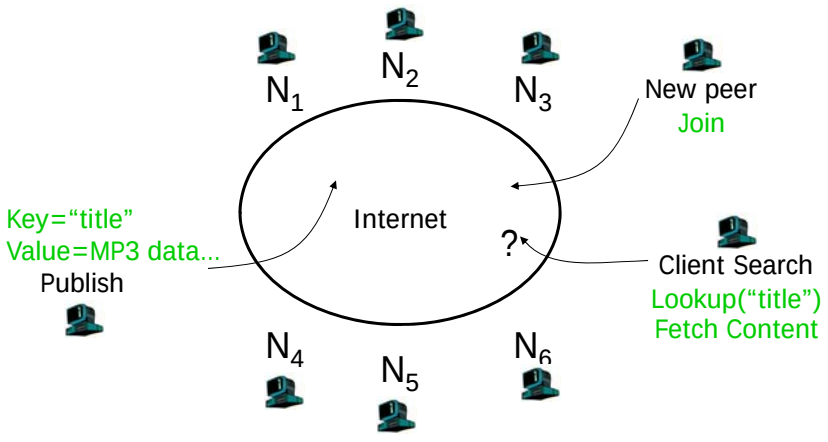


- Harness lots of spare capacity
  - 1 Big Fast Server: 1Gbit/s, $10k/month++
  - 2,000 cable modems: 1Gbit/s, $??
  - 1M end-hosts: Uh, wow.
  - Capacity grows with the number of users!
- Build very large-scale, self-managing systems
  - Same techniques useful for companies and p2p apps
    - E.g., Akamai's 14,000+ nodes, Google's 100,000+ nodes
  - Many differences to consider
    - Servers versus arbitrary nodes
    - Hard state (backups!) versus soft state (caches)
    - Security, fairness, freeloading, ..

5

Common P2P Framework





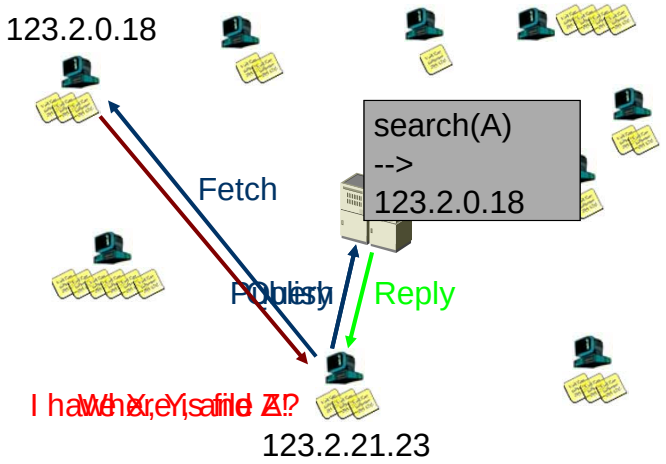
6

What is (was) out there?

	Central	Flood	Super-node flood	Route
Whole File	Napster	Gnutella		Freenet
Chunk Based	BitTorrent		KaZaA (bytes, not chunks)	DHTs eDonkey 2000

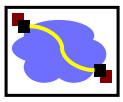
7

Napster: Central Database



8

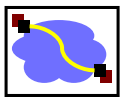
Napster: Discussion

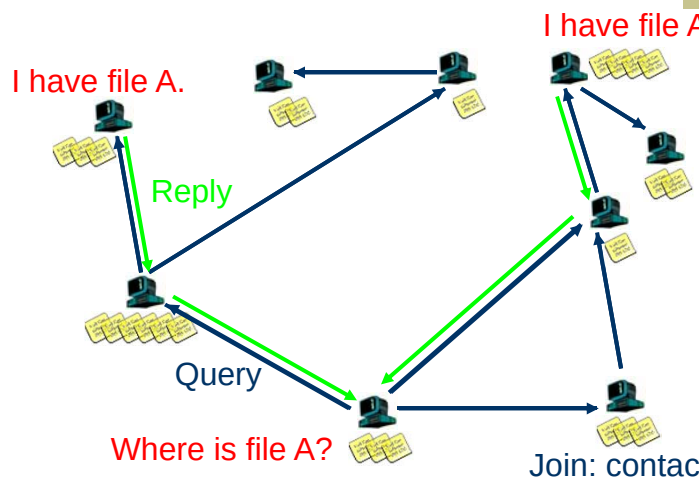


- Pros:
  - Simple
  - Search scope is O(1)
  - Controllable (pro or con?)
- Cons:
  - Server maintains O(N) State
  - Server does all processing
  - Single point of failure

9

Gnutella: Flooding

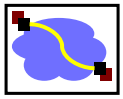




Join: contact peers
Publish: noop
Fetch: direct p2p

10

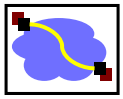
Gnutella: Discussion



- Pros:
  - Fully de-centralized
  - Search cost distributed
  - Processing @ each node permits powerful search semantics
- Cons:
  - Search scope is O(*N*)
  - Search time is O(???)
  - Nodes leave often, network unstable
- TTL-limited search works well for haystacks.
  - For scalability, does NOT search every node.
  - May have to re-issue query later

11

KaZaA: Query Flooding

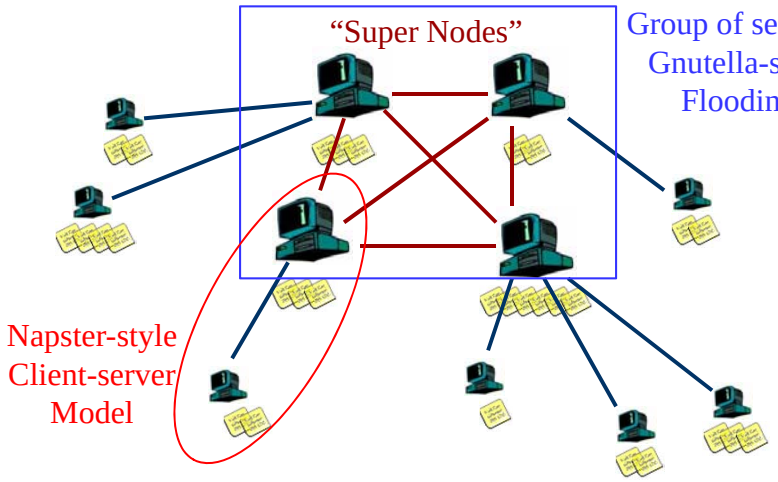


- First released in 2001 and still used today
  - Also very popular

- **Join**: on startup, client contacts a “supernode” ... may at some point become one itself
- **Publish**: send list of files to supernode
- **Search**: send query to supernode, supernodes flood query amongst themselves.
- **Fetch**: get the file directly from peer(s); can fetch simultaneously from multiple peers

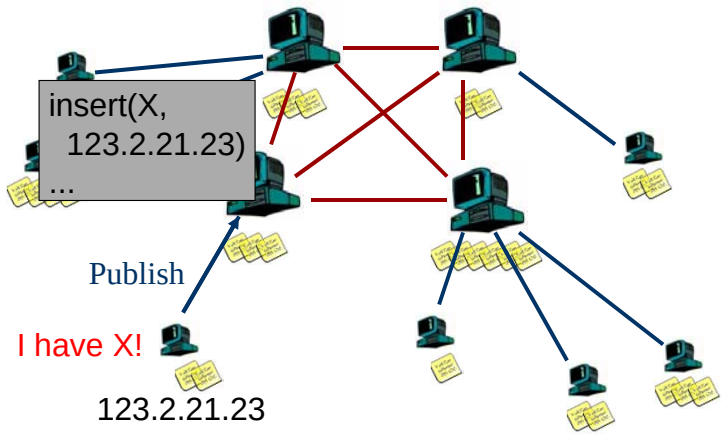
12

KaZaA: Intelligent Query Flooding



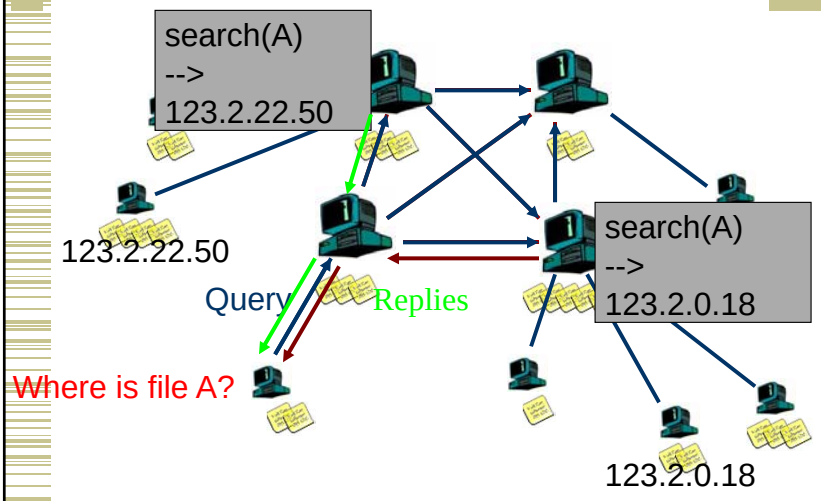
13

KaZaA: File Insert and Query



14

KaZaA: File Search



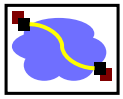
15

KaZaA: Discussion

- Works better than Gnutella because of query consolidation
- Several nodes may have requested file... How to tell?
  - Must be able to distinguish identical files
  - Not necessarily same filename
  - Same filename not necessarily same file...
- Use Hash of file
  - Can fetch bytes [0..1000] from A, [1001...2000] from B
- Pros: Tries to take into account node heterogeneity:
  - Bandwidth, computational resources, …
- Cons: Still no guarantees on search scope or time
- Challenge: want stable superpeers – good prediction
  - Must also be capable platforms

16

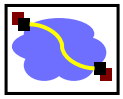
KaZaA: Discussion



- Pros:
  - Tries to take into account node heterogeneity:
    - Bandwidth
    - Host Computational Resources
    - Host Availability (?)
  - Rumored to take into account network locality
- Cons:
  - Mechanisms easy to circumvent
  - Still no real guarantees on search scope or search time
- Similar behavior to gnutella, but better.

17

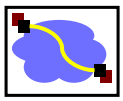
Overview



- Web
- Peer-to-peer
  - Motivation
  - Architectures
  - BitTorrent
  - TOR
  - Skype
- CDN
- Video

18

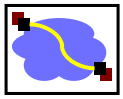
BitTorrent: Swarming

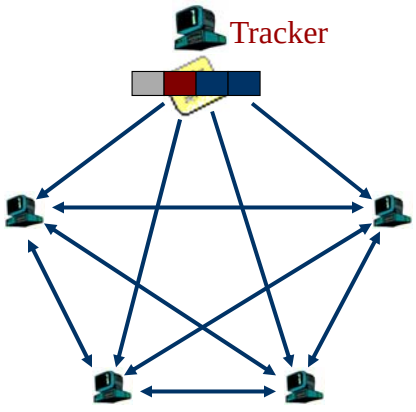


- Started in 2001 to efficiently support flash crowds
  - Focus is on fetching, not searching
- **Publish**: Run a tracker server.
- **Search**: Find a tracker out-of-band for a file, e.g., Google
- **Join**: contact central “tracker” server for list of peers.
- **Fetch**: Download chunks of the file from your peers. Upload chunks you have to them.
- Comparison with earlier architectures:
  - Focus on fetching of “few large files”
  - Chunk based downloading
  - Anti-freeloading mechanisms

19

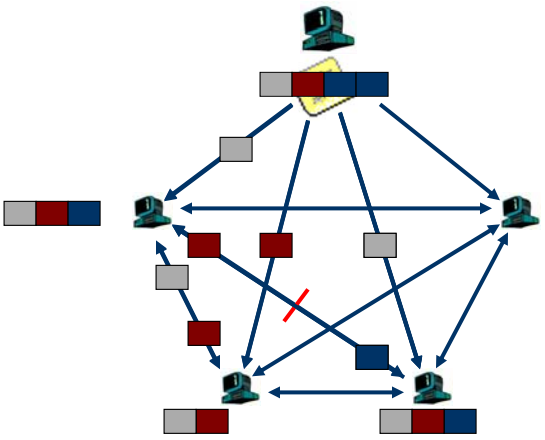
BitTorrent: Publish/Join





20

BitTorrent: Fetch



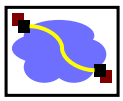
21

BitTorrent: Sharing Strategy

- Employ “Tit-for-tat” sharing strategy
  - A is downloading from some other people
    - A will let the fastest N of those download from him
  - Be optimistic: occasionally let freeloaders download
    - Otherwise no one would ever start!
    - Also allows you to discover better peers to download from when they reciprocate
- Goal: Pareto Efficiency
  - Game Theory: “No change can make anyone better off without making others worse off”
  - Does it work? (don’t know!)

22

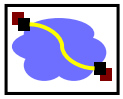
BitTorrent: Summary



- Pros:
  - Works reasonably well in practice
  - Gives peers incentive to share resources; avoids freeloaders
- Cons:
  - Pareto Efficiency relative weak condition
  - Central tracker server needed to bootstrap swarm
  - (Tracker is a design choice, not a requirement, as you know from your projects. Could easily combine with other approaches.)

23

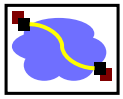
When are p2p Useful?



- Works well for caching and “soft-state”, read-only data
  - Works well! BitTorrent, KaZaA, etc., all use peers as caches for hot data
- Difficult to extend to persistent data
  - Nodes come and go: need to create multiple copies for availability and replicate more as nodes leave
- Not appropriate for search engine styles searches
  - Complex intersection queries (“the” + “who”): billions of hits for each term alone
  - Sophisticated ranking: Must compare many results before returning a subset to user
  - Need massive compute power

24

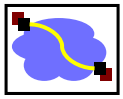
Overview



- Web
- Peer-to-peer
  - Motivation
  - Architectures
  - BitTorrent
  - TOR
  - Skype
- CDN
- Video

25

Tor Anonymity Network

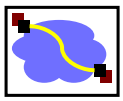


- Deployed onion routing network
  - http://torproject.org
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
  - Thousands of relay nodes, 100K-500K? of users
- Easy-to-use client proxy, integrated Web browser
  - Not like FreeNet – no data “in” TOR
- Really an overlay – not pure peer-to-peer

Based on slides by Vitaly Shmatikov

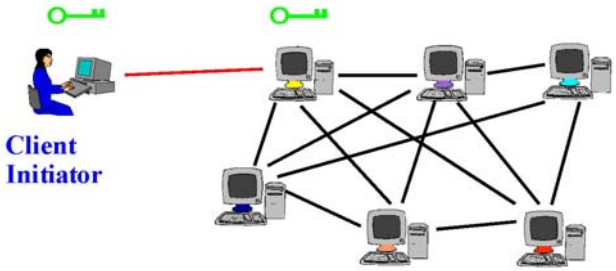
slide 26

Tor Circuit Setup (1)

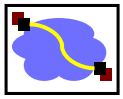


- Client proxy establish a symmetric session key and circuit with relay node #1
- All data sent over the circuit is encrypted

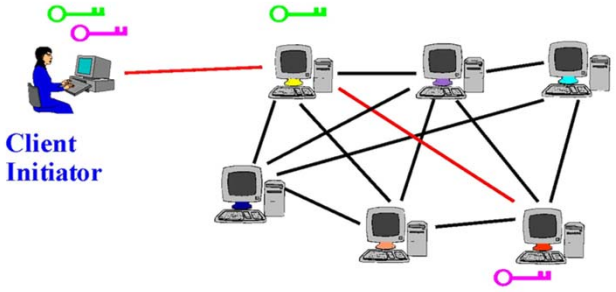
$A = K(B)_k$



Tor Circuit Setup (2)

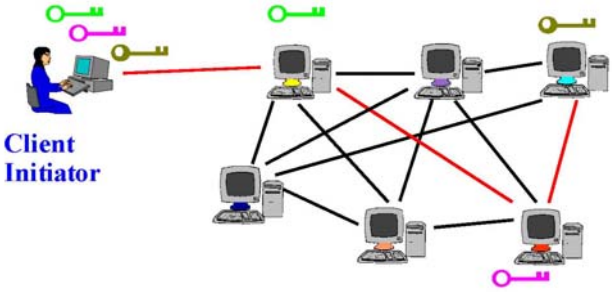


- Client proxy extends the circuit by establishing a symmetric session key with relay node #2
  - Tunnel through relay node #1



Tor Circuit Setup (3)

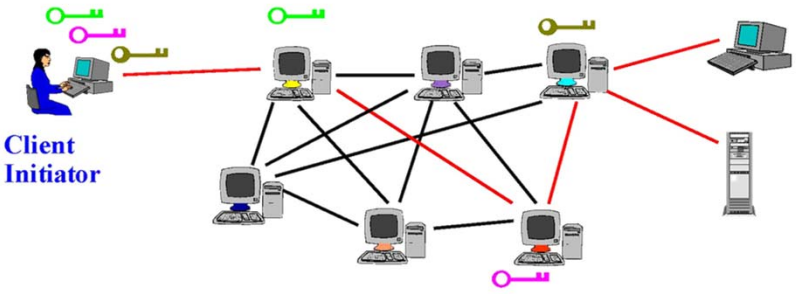
- Client proxy extends the circuit by establishing a symmetric session key with relay node #3
  - Tunnel through relay nodes #1 and #2



slide 29

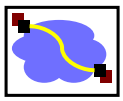
Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit
  - Datagrams decrypted at each link
- Also want end-to-end encryption – not done by Tor



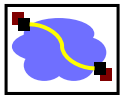
slide 30

Using Tor



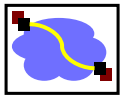
- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active relay nodes, their locations, current public keys, etc.
  - Control how new nodes join the network
    - "Sybil attack": attacker creates a large number of relays
  - Directory servers' keys ship with Tor code

Overview



- Web
- Peer-to-peer
  - Motivation
  - Architectures
  - BitTorrent
  - TOR
  - Skype
- CDN
- Video

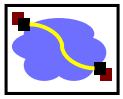
What is Skype?



- Support pc-to-pc, pc-to-phone, phone-to-pc VoIP and IM client communication
  - Also: conference calls, video, …
- Developed by people who created KaZaa
  - Has peer-to-peer features that will look familiar
- Supported OS: Windows, Linux, MacOS, PocketPC
- A p2p illusion
  - Login server
  - Buddy-list server
  - Servers for SkypeOut and SkypeIn
  - Anonymous call minutes statistic gathering

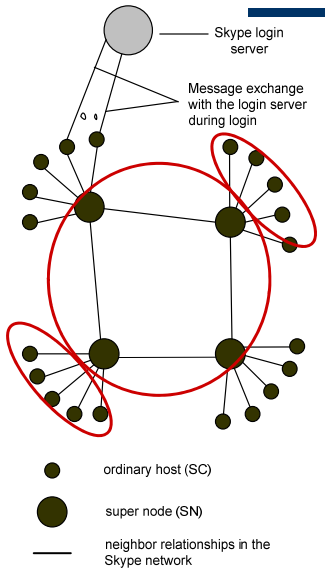
Based on slides by Baset and Schulzrinne (Infocom 06)

What problems does it solve?



- NAT and firewall traversal
  - Nielsen September 2005 ratings
    - 61.3% of US home internet users use broadband (http://www.nielsen-netratings.com/pr/pr_050928.pdf)
    - 'Most' users have some kind of NAT
- Calls between traditional telephone and internet devicese
  - SkypeOut (pc-to-phone)
    - Terms of service: governed by the laws of Luxembourg
  - SkypeIn (phone-to-pc), voicemail
- Configuration-less connectivity
- Scalability for member data and call bandwidth

The Skype Network

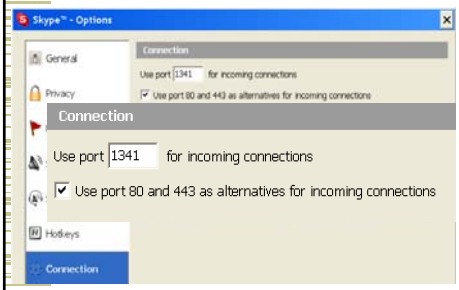


- Ordinary host (OH)
  - A Skype client (SC)
- Super nodes (SN)
  - A Skype client (SC)
  - Has public IP address, 'sufficient' bandwidth, CPU and memory
- Login server
  - Stores Skype id's, passwords, and buddy lists
  - Used at login for authentication

35

Ports used by Clients

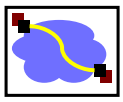
- No default listening port
- Randomly chooses a port (P1) on installation
- Opens TCP and UDP listener sockets at P1
- Opens TCP listener sockets at port 80 (HTTP) and port 443 (HTTPS)



- Clients also use a cache:
  - IP addresses and port numbers of supernodes
  - Login server IP address and port number

36

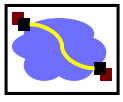
Skype Functions: LOGIN



- Public, NAT
  - Establishes a TCP connection with the supernode
    - Keep connection alive by sending refresh message every 2 min.
  - Authenticates with the login server
  - Announces arrival on the network (controlled flooding)
  - Determines NAT type
- Firewall
  - Establishes a TCP connection with the supernodes
  - Authenticates with the login server

37

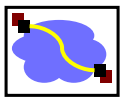
Skype Functions: USER SEARCH



- From the Skype website
  - Guaranteed to find a user it exists and logged in the last 72 hours
- Search results are cached at intermediate nodes
- Cannot force a node to become a SN
  - Host cache is used for connection establishment and not for SN selection
- User does not exist. How does search terminate?
  - Skype contacts login server for failed searches

38

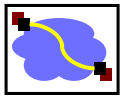
Skype Functions: CALL ESTABLISHMENT



- Call signaling always carried over TCP and goes e2e
- Calls to non buddies=search+call
- Public-public call
  - Caller establishes a TCP connection with callee
- Public-NAT
  - Caller is behind port-restricted NAT
    - Different solutions based on the nature of the NAT
  - Caller----> supernode ----> Callee
  - TCP connections established between caller, callee, and more than one Skype nodes
- Firewall-firewall call
  - Same as public-NAT but no in-UDP packets

39

Skype Functions: MEDIA TRANSFER



- No silence suppression
- Silence packets are used to
  - play background noise at the peer
  - maintain UDP NAT binding
  - avoid drop in the TCP congestion window
- Putting a call on hold
  - 1 packet/3 seconds to call-peer or Skype node
  - same reasons as above
- Codec frequency range
  - 50-8,000 Hz (total bw of 3 kilobytes/s)
- Reasonable call quality at (4-5 kilobytes/s)

40