



Overlay Networks

Jennifer Rexford

COS 461: Computer Networks

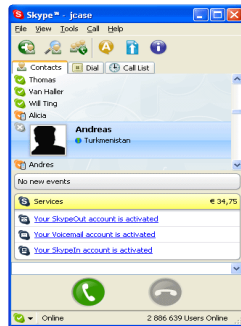
Lectures: MW 10-10:50am in Architecture N101

<http://www.cs.princeton.edu/courses/archive/spr12/cos461/>

Skype

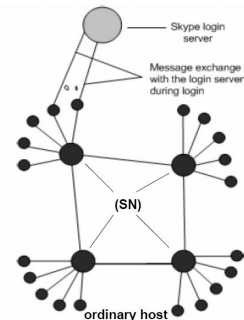
Skype

- Niklas Zennström and Janus Friis in 2003
- Developed by KaZaA
- Instant Messenger (IM) with voice support
- Based on peer-to-peer (P2P) networking technology



Skype Network Architecture

- Login server is the only central server
- Both ordinary host and super nodes are clients
- Any node with public IP address and resources can become a super node



Challenges of Firewalls and NATs

- **Firewalls**
 - Often block UDP traffic
 - Usually allow hosts to initiate connections on port 80 (HTTP) and 443 (HTTPS)
- **Network Address Translation (NAT)**
 - Cannot easily initiate traffic to a host behind a NAT
- **Skype must deal with these problems**
 - Discovery: client exchanges messages with super node
 - Traversal: sending data through an intermediate peer

Data Transfer

- **UDP directly between the two hosts**
 - Both hosts have public IP addresses, no UDP blocks
 - **Easy:** the hosts can exchange UDP packets directly
- **UDP between an intermediate peer**
 - One or both hosts with a NAT
 - Neither host's network blocks UDP traffic
 - **Solution:** direct UDP packets through another node
- **TCP between an intermediate peer**
 - Hosts behind NAT and UDP-restricted firewall
 - **Solution:** TCP connections through another node

Silence Suppression

- **What to transfer during quiet periods?**
 - Could save bandwidth by reducing transmissions
- **Skype does not appear to do silence suppression**
 - Maintain the UDP bindings in the NAT boxes
 - Provide background noise to play at the receiver
 - Avoid drop in the TCP window size
- **Skype sends data when call is “on hold”**
 - Send periodic messages as a sort of heartbeat
 - Maintain the UDP bindings in the NAT boxes

7

Skype Data Transfer

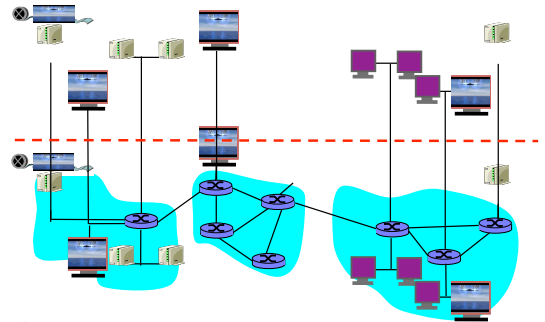
- **Audio compression**
 - Voice packets around 67 bytes
 - Up to 140 packets per second
 - Around 5 KB/sec (40 kbps) in each direction
- **Encryption**
 - Data packets are encrypted in both directions
 - To prevent snooping on the phone call
 - ... by someone snooping on the network
 - ... or by the intermediate peers forwarding data

8

Overlay Networks

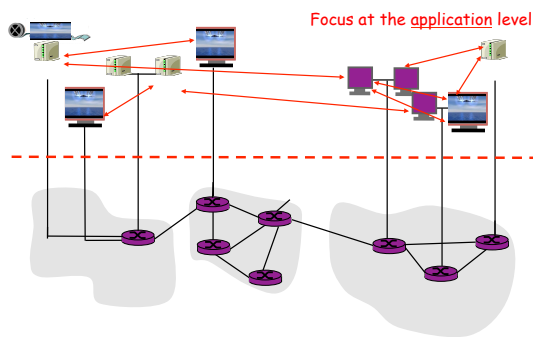
9

Overlay Networks




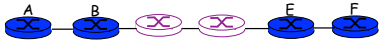
10

Overlay Networks

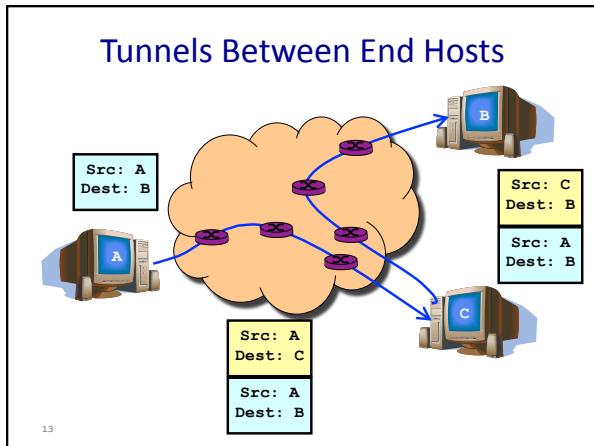


11

IP Tunneling to Build Overlay Links

- **IP tunnel is a virtual point-to-point link**
 - Illusion of direct link between two separated nodes
- Logical view: 
- Physical view: 
- **Encapsulation of packet inside an IP datagram**
 - Node B sends a packet to node E
 - ... containing another packet as the payload

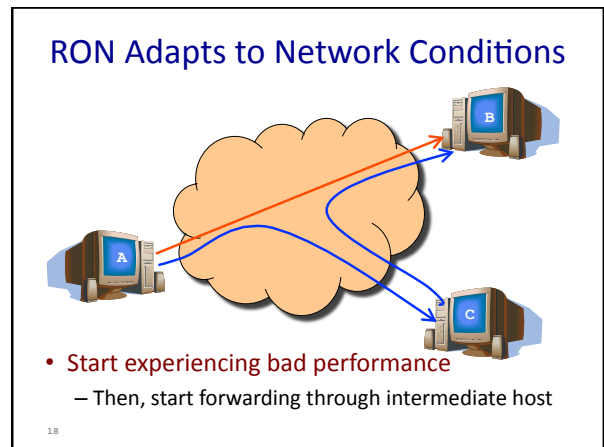
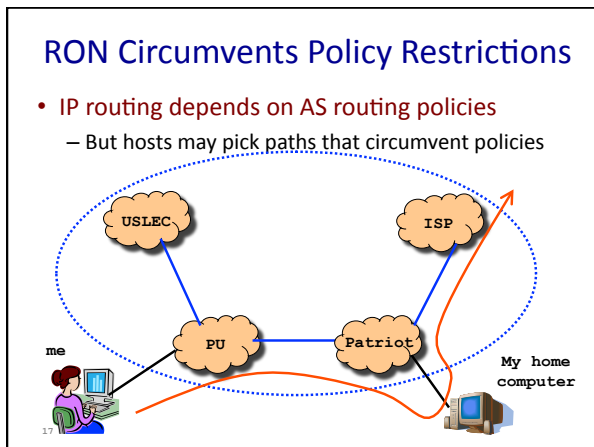
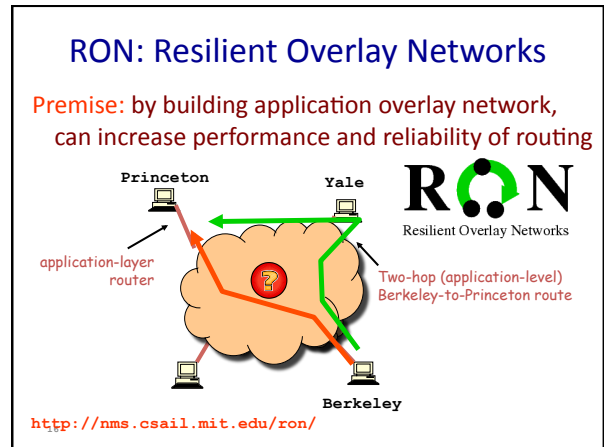
12



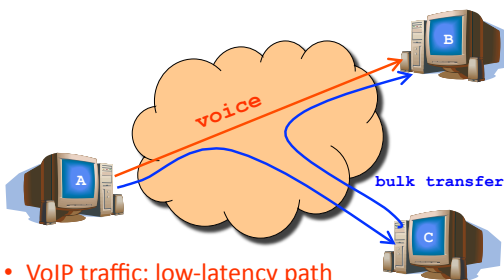
- ### Overlay Networks
- Logical network built on top of physical network
 - Overlay link is tunnel through underlying network
 - Many logical networks may coexist at once
 - Over the same underlying network
 - Nodes are often end hosts
 - Acting as intermediate nodes that forward traffic
 - Who controls the nodes providing service?
 - The party providing the service
 - Distributed collection of end users
- 14

Case Study: Resilient Overlay Networks

15



RON Customizes to Applications

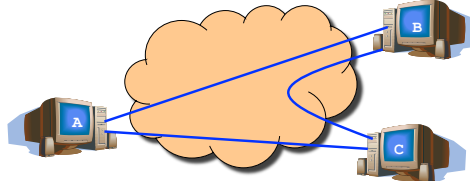


- VoIP traffic: low-latency path
- Bulk transfer: high-bandwidth path

19

How Does RON Work?

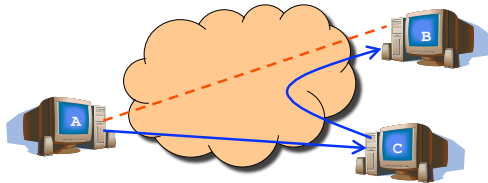
- Keeping it small to avoid scaling problems
 - A few friends who want better service
 - Just for their communication with each other
 - E.g., VoIP, gaming, collaborative work, etc.
- Send probes between each pair of hosts



20

How Does RON Work?

- Exchange the results of the probes
 - Each host shares results with every other host
 - Essentially running a link-state protocol!
 - So, every host knows the performance properties
- Forward via intermediate host when needed



21

RON Works in Practice

- Faster reaction to failure
 - RON reacts in a few seconds
 - BGP sometimes takes a few minutes
- Single-hop indirect routing
 - No need to go through many intermediate hosts
 - One extra hop circumvents the problems
- Better end-to-end paths
 - Circumventing routing policy restrictions
 - Sometimes the RON paths are actually shorter

22

RON Limited to Small Deployments

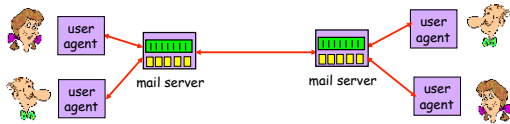
- Extra latency through intermediate hops
 - Software and propagation delays for forwarding
- Overhead on the intermediate node
 - Imposing CPU and I/O load on the host
- Overhead for probing the virtual links
 - Bandwidth consumed by frequent probes
 - Trade-off between probe overhead & detection speed
- Possibility of causing instability
 - Moving traffic in response to poor performance
 - May lead to congestion on the new paths

23

Electronic Mail

24

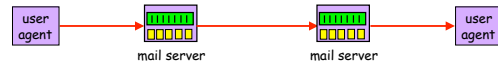
Mail Servers and User Agents



- **Mail servers**
 - Always on and always accessible
 - Transferring e-mail to and from other servers
- **User agents**
 - Sometimes on and sometimes accessible
 - Intuitive interface for the user

25

Store-and-Forward Model

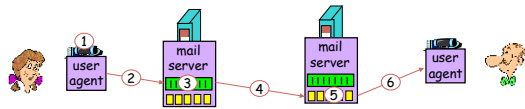


- **Messages sent through a series of servers**
 - A server stores incoming messages in a queue
 - ... to await attempts to transmit them to next hop
- **If the next hop is not reachable**
 - The server stores the message and tries again later
- **Each server adds a Received header**
 - To aid in diagnosis of problems

26

Scenario: Alice Sends Message to Bob

- 1) Alice uses UA to compose message "to" bob@someschool.edu
- 2) Alice's UA sends message to her mail server; message placed in message queue
- 3) Alice's mail server opens TCP connection with Bob's mail server
- 4) Alice's mail server sends message over the TCP connection
- 5) Bob's mail server places the message in Bob's mailbox
- 6) Bob invokes his user agent to read message



27

Identifying the Mail Server

- **Alice identifying her mail server**
 - User-agent configuration (e.g., smtp.cs.princeton.edu)
- **Alice's mail server identifying Bob's mail server**
 - From name in Bob's e-mail address (e.g., yale.edu)
- **Domain name is not necessarily the mail server**
 - Mail server may have longer/cryptic name
 - Multiple servers may exist to tolerate failures
- **Identifying the mail server for a domain**
 - DNS query asking for MX records (Mail eXchange)
 - Then, a regular DNS query to learn the IP address

28

Simple Mail Transfer Protocol



- **Client-server protocol**
 - Client is sender, server is receiver
- **Reliable data transfer**
 - Built on top of TCP (on port 25)
- **Push protocol**
 - Sending server pushes file to the receiving server
 - ... rather than waiting for the receiver to request it

29

Sample SMTP interaction

```

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
    
```

30

Try SMTP For Yourself

- **Running SMTP**
 - Run “telnet servername 25” at UNIX prompt
 - See 220 reply from server
 - Enter HELO, MAIL FROM, RCPT TO, DATA commands
- **Spoofing is easy!**
 - Just forge the argument of the “FROM” command
 - ... leading to all sorts of problems with spam
- **Spammers can be even more clever**
 - E.g., using open SMTP servers to send e-mail
 - E.g., forging the “Received” header

31

Multiple Server Hops

- **Typically at least two mail servers**
 - Sending and receiving email servers
- **May be more**
 - Separate servers for key functions
 - Spam filtering, virus scanning
 - Servers that redirect the message
 - From jrex@princeton.edu to jrex@cs.princeton.edu
 - Messages to princeton.edu go through extra hops
 - Electronic mailing lists
 - Mail delivered to the mailing list’s server
 - ... and then the list is expanded to each recipient

32

Example With Received Header

```
Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
  by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04MSR7Y023164
  for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from blobbox.CS.Princeton.EDU ([128.112.136.38])
  by ribavirin.CS.Princeton.EDU (SMTPSMTF 4.1.0.19) with SMTP id M2006010417053607946
  for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
  by blobbox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04MSW9Q005204
  for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pitn13.pacbell.net [69.107.78.147])
  (authenticated bits=0)
  by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04MSW92018875
  (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NO);
  Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC046F.3D30306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit
```

33

Retrieving E-Mail From the Server

- **Server stores incoming e-mail by mailbox**
 - Based on the “From” field in the message
- **Users need to retrieve e-mail**
 - Asynchronous from when the message was sent
 - With a way to view and organize messages
- **In the olden days...**
 - User logged on to machine where mail was delivered
 - Users received e-mail on their main work machine
- **Now, user agent typically on a separate machine**
 - And sometimes on more than one such machine

34

Influence of PCs on E-Mail Retrieval

- **Separate machine for personal use**
 - Users did not want to log in to remote machines
- **Resource limitations**
 - Most PCs did not have enough resources to act as a full-fledged e-mail server
- **Intermittent connectivity**
 - PCs only sporadically connected to the network
 - Too unwieldy to have sending server keep trying
- **Led to the creation of new e-mail agents**
 - POP, IMAP, and Web-based e-mail

35

Post Office Protocol (POP)

- **POP goals**
 - Support users with intermittent connectivity
 - Retrieve e-mail messages when connected
- **Typical user-agent interaction with a POP server**
 - Connect to the server
 - Retrieve all e-mail messages
 - Store messages on the user’s PCs as new messages
 - Delete the messages from the server
 - Disconnect from the server

36

Limitations of POP

- **Does not handle multiple mailboxes easily**
 - Designed to put user's incoming e-mail in one folder
- **Not designed to keep messages on the server**
 - Instead, designed to download messages to client
- **Poor handling of multi-client access to mailbox**
 - Increasingly important as users have home PC, work PC, laptop, cyber café computer, PDA, etc.
- **High network bandwidth overhead**
 - Transfers all of e-mail messages, often well before they are read (and they might not be read at all!)

37

Interactive Mail Access Protocol (IMAP)

- **Supports connected and disconnected operation**
 - Users can download message contents on demand
- **Multiple clients can connect to mailbox at once**
 - Detects changes made to mailbox by other clients
 - Server keeps message state (e.g., read, replied to)
- **Access to parts of messages and partial fetch**
 - Clients can retrieve individual parts separately
 - E.g., message text without attachments
- **Multiple mailboxes on the server**
- **Server-side searches**

38

Web-Based E-Mail

- **User agent is an ordinary Web browser**
 - User communicates with server via HTTP
 - E.g., Gmail, Yahoo mail, and Hotmail
- **Reading e-mail**
 - Web pages display the contents of folders
 - “GET” request to retrieve the various Web pages
- **Sending e-mail**
 - User types text into a form and submits to server
 - “POST” request to upload data to the server
 - Server uses SMTP to deliver message to other servers

39

Conclusions

- **Overlay networks**
 - Tunnels between host computers
 - Build networks “on top” of the Internet
 - Deploy new protocols and services
- **Benefits of overlay networks**
 - Customization to the applications and users
 - Incremental deployment of new technologies
 - May perform better than the underlying network
- **Precept: Distributed Hash Tables**

40