IT441: Network Servers & Infrastructure

CLASS 10: 04 Apr 2005
13:30 - 16:15

This Time

- sample services:
  - mail
  - VoIP

A Familiar Content Service

- electronic mail service

MTA to MTA

- mail transfer agent (MTA) moves messages to their destination
- from a queue to some (recipient) MTA
- to a user agent from some (sender) MTA
- MTAs speak simple mail transfer protocol (SMTP) to each other
  - originally in RFC821
  - currently RFC2821
- SMTP is a TCP service
- runs on port 25

MTA's view: sending mail

220 something.org Sendmail 8.12 ready
HELO mynode.subdomain.com
250 OK
MAIL From:<scarter@mynode.subdomain.com>
250 <scarter@mynode.subdomain.com> OK
RCPT To: <joneil@nodex.sgc.mil> 250 OK mail accepted
DATA
354 Enter mail, end with \n\n mail message inserted here
.
250 OK mail accepted
QUIT
221 something.org closing connection

Last Time:

- TCP
- some basic protocols for getting started
  - RARP
  - BOOTP
  - DHCP
- Domain Name Service
Server Response Codes

- first defined for SMTP in RFC822
- now widely used in other services
- based on 3-digit xyz values:
  - 1yz positive preliminary reply
  - 2yz positive completion reply
  - 3yz positive intermediate reply
  - 4yz transient negative completion reply
  - 5yz permanent completion reply

- and z is context-specific code providing more detail

Server Commands

1. HELO, EHLO:
   - identify the SMTP client to the SMTP server
   - client sends its fully-qualified domain name
   - HELO used by older clients
   - all servers support HELO
   - newer ones support extended services, hence EHLO

2. MAIL
   - start mail transaction where mail is sent to server

3. RCPT
   - identifies a single recipient of the mail message
   - use multiple instances of RCPT for multiple recipients

4. DATA
   - server replies with 354 line then copies everything sent to it by client up to the '.
   - data should consist only of 7-bit ASCII characters
   - and avoid ctrl chars other than SP, HT, CR, and LF
Server Commands

1. HELO, EHLO:
2. MAIL
3. RCPT
4. DATA
5. RSET
   - aborts current mail operation in progress
   - any data received as part of this operation is deleted from server
   - has no effect if appears immediately after EHLO
6. VRFY
   - asks receiver to confirm that argument identifies a user or mailbox
7. EXPN
   - asks receiver to confirm that argument identifies a mailing list and, if so, return list membership
8. HELP
   - causes server to send helpful information to client
9. NOOP
   - has no effect on server other than to cause it to respond with OK
10. QUIT
   - server must send OK and close connection
email messages

- simple structure —
  - envelope: used by MTAs for delivery
    - consists of the 2 SMTP commands MAIL and RCPT
  - header: contains non-message information like addressee, sender, date, etc.
    - has a standard syntax
  - body: contains actual content of message
    - can be ‘anything’

Mail Headers

- used by user agents
- simple syntax: attribute/value
  - attributes may contain ASCII chars with codes 0x21 to 0x7E except 0x20
  - values may contain any ASCII chars except CR and LF
  - e.g., “From: user@example.com”
    - some start with +
      - are user-defined
        - e.g., X-Charset, X-Mailer

Mail Headers

- From: mailbox-list
  - specifies the author(s) of the message, that is, the mailbox(s) of the person(s) or system(s) responsible for the writing of the message
    - must appear
  - Sender: mailbox
    - specifies the mailbox of the agent responsible for the actual transmission of the message
      - can appear, but must appear if mailbox-list in From: line has \n  - Reply-To: address-list
    - list to which replies to a message should be sent
      - is optional

Mail Headers

- To: address-list
  - contains the address(es) of the primary recipient(s) of the message
  - must appear
- CC: address-list
  - contains the addresses of others who are to receive the message
- BCC: address-list
  - contains addresses of recipients of the message whose addresses are not to be revealed to other recipients of the message

Mail Headers

- Message-ID: <ident>
  - not intended to be human-readable
    - e.g., <200405262028.120501620@argonauts.mit.edu>
    - provides unique identifier referring to particular version of particular message: uniqueness guaranteed by host generating it
  - In-Reply-To: <ident>
    - lists message-id of current msg (to which it is a reply)
  - References: <ident>
    - provides id’s appearing in current msg’s references field

Mail Headers

- informational fields, human readable:
  - Subject:
  - Comments:
  - Keywords:
Mail Headers

- user defined fields:
  - X-Charset: identify character set used to represent message
  - X-Mailer: user agent used to send message
  - X-Sender: duplicate of From:

email messages

- originally were only text
  - using 7-bit ASCII
  - protocols designed around text-only original version
  - still support 7-bit ASCII
  - some may support 8-bit character data (see RFC1652)
  - described in RFC822 (still often cited)
  - current version is RFC822
    - lines must be ≤ 1000 bytes long
    - but now want more than just text...

Different Content Types

- need mechanism to support:
  - different kinds of content to appear in a msg
  - multiple different kinds of content within one msg

Different Content Types

- need mechanism to support:
  - different kinds of content to appear in a msg
  - multiple different kinds of content within one msg
  - use Multipurpose Internet Mail Extensions (MIME)
    - RFC1521 (now: RFCs: 2045–2049)
    - provides way to encode binary data using only printable ASCII characters
    - inflates size of data

MIME

- used in email, adds lines to headers:
  - for standard text messages:
    - MIME-Version: 1.0
    - Content-type: text/plain;
      charset=US-ASCII
    - text of message follows
    - message ends at end-of-text

MIME

- used in email, adds lines to headers:
  - for multi-part messages:
    - Content-Type: multipart/related;
      boundary="-------------004d05050103407600000708"
    - Content-Type: text/plain;
      charset=ISO-8859-1;
      format=fixed
    - Content-Transfer-Encoding: 7bit
    - first part of message (text):
      "-------------004d05050103407600000708"
      Content-Type: image/jpeg;
      name="IMG_0801.jpg"
      Content-Transfer-Encoding: base64
      Content-Disposition: inline;
      filename="IMG_0801.jpg"
    - second part of message (jpeg image):
      "-------------004d05050103407600000708--"
### Non-ASCII in Header

- headers may also contain non-ASCII chars
- introduced using syntax:

```plaintext
charset ? encoding ? encoded-text
```

- start delimiter
- identify character set:
  e.g., ISO-8859-1
- encoding scheme:
  0 for quoted printable
  8 for base-64

### Transfer Encodings

ways to represent non-ASCII data as ASCII:

2. quoted-printable

- ASCII chars with codes 0x21 to 0x7E (except 0x20) appears as is
- all line breaks expressly appear as \n\n
- space appears as _
- unless at end of line ":-0"
- lines longer than 76 bytes have inserted line breaks
- everything else represented as sequence =xx
  - e.g., `a` appears as `~a`

### Quoted Printable Example

```plaintext
From: +755-867-1776@patrik.f@411.com+867+@patrik..kth.se
translated into:

patrik@411.com
```

element taken from RFC1522

### Transfer Encodings

ways to represent non-ASCII data as ASCII:

2. base64 encoding

- replace every 6 bits of binary data with a single ASCII character from special character table
- table contains:
  - A...Z
  - a...z
  - 0...9
  - + / =
- send string so generated

### Base-64 Example

```plaintext
From: +755-867-1776@patrik.moore+emoore@cs.utk.edu
To: +755-867-1776@patrik.f@411.com+867+@patrik..kth.se
Cc: +755-867-1776@patrik.f@411.com+867+@patrik..kth.se
Subject: +755-867-1776@patrik.moore+emoore@cs.utk.edu+emoore@cs.utk.edu+
+emoore@cs.utk.edu+emoore@cs.utk.edu+emoore@cs.utk.edu+
+emoore@cs.utk.edu+emoore@cs.utk.edu+emoore@cs.utk.edu+
+emoore@cs.utk.edu+emoore@cs.utk.edu+emoore@cs.utk.edu+
which a user agent renders as:

From: Keith Moore +emoore@cs.utk.edu
To: Walid Jarm Simensen <keld@kduug.dk>
Cc: André Picard <PERMAN@vel.ulg.ac.be>
Subject: If you can read this you understand the example.
```

### Address Services

- what's in a name?
  - user@subnode.subdomain.domain
  - where does "user" get its email?
  - what about
    - user@othernode.subdomain.domain

### Address Services

- what's in a name?
  - user@subnode.subdomain.domain
  - where does "user" get its email?
  - what about
    - user@othernode.subdomain.domain
Address Services

- many organizations now use email gateway
  - node dedicated to receiving, forwarding, sending email
- lets users have email addresses like
  - username@domain
- gateway receives all inbound email
  - either forwards to particular node, or,
  - keeps local and user’s user-agent retrieves the email therefrom

Address Services

- gateway receives all inbound email
  - either forwards to particular node
- gateway needs list of entries associating
  - username@domain with
    - username@subdomain.com
- must this association be 1:1?

Address Services

- must this association be 1:1?
  - no: can have one name: many names
    - so email gateway can match a listname with a 1 mailboxes and must generate individual copies of the message to each
  - such a gateway is hence known as a mail-exploder
    - see Comer fig. 32.5 for example
  - users must have their email address added to list in order to receive mailings

Address Services

- adding name to list can be automated procedure:
  - usually simply send email message to list
  - follow syntax rules specified by list, e.g., send subject line as ‘add me’
  - no body to msg (not being read by human)
  - list manager on email gateway watches for these special msgs
    - performs add when discovers one to do
    - analogously for removal

Getting Your Mail

- today, email usually is delivered to some server whence you must retrieve it
  - rather than being delivered to ‘your’ local machine
  - i.e., your email user agent may be local, but the actual mailbox isn’t
- how to retrieve?
**Post Office Protocol (POP)**

- so mail client runs
- software to send email (via SMTP)
- software to receive email (via POP)
- POP client interacts with server to,
  - copy accumulated email from mail server to user
  - delete copied messages from email server, or leave intact
- service runs on port 110

**Post Office Protocol (POP)**

- example client server interaction in POP3:
  - connect to port 110 on server...
  - OK POP3 server ready
  - USER fred
  - OK
  - PASS derf
  - OK
  - USER Logged on
  - LIST
  - OK 2 messages (120 octets)
  - 1 120
  - 2 200
  -

**Post Office Protocol (POP)**

- example client server interaction in POP3, cont’d:
  - RETR 1
    - OK 120 octets
    - the POP3 server sends the entire message here>
  - .
  - DELE 1
    - OK message 1 deleted
  - DELE 1
    - ERR message 1 already deleted
  - QUIT
    - OK POP3 server signing off

**POP3**

- available commands in the POP3 protocol include:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>APOP</td>
<td>PASS</td>
<td>STAT</td>
</tr>
<tr>
<td>DELE</td>
<td>QUIT</td>
<td>TOP</td>
</tr>
<tr>
<td>LIST</td>
<td>RETR</td>
<td>UDL</td>
</tr>
<tr>
<td>NOOP</td>
<td>RESET</td>
<td>USE</td>
</tr>
</tbody>
</table>

**Another Mail Fetching Protocol**

- instead of POP, a user agent may use Internet Mail Access Protocol (IMAP) RFC3501
- particular advantage: allows management of messages on a server
  - e.g., can organize messages into folders
  - also provides ability to retrieve only parts of a message
  - e.g., subject lines
  - e.g., specific part of MIME multipart message

**Servers**

- provide a service
- provide content
  - static
  - dynamic
    - generated on-demand
  - “streamed”
    - on-going flow of data
    - often real-time sensitive
    - may be pre-computed (e.g., mp3)
    - real-time, not pre-computable (e.g., telephony)
Internet Telephony

- real-time
- full duplex
- cannot be pre-computed, pre-compressed
- provide usual telephony services
  - call forwarding
  - call waiting
  - caller ID
  - PBX services
- interoperability with existing PSTN

Telephones

- what is a telephone?
  - activation control switch
- audio capture device (microphone)
  - audio reproduction device (speaker)
- activity alert device (ringer)
  - address selection device (keypad)
- any combination of hardware and software that performs these functions can be a telephone

Conventional Phone Network

Voice over IP

- end points of calls:
  - computer (with mic and spkr)
- 'phone call' can originate or end at either
  - need to have Internet interoperate with PSTN
  - signal standards
  - addressing
    - phone numbers
    - IP addresses

The New Layout

- need to use telephony stds for PSTN work
- need to use IP stds for Internet work
Special Gateways

- media gateway:
  - audio data xlate between IP and PSTN
- media gateway controller:
  - 'overall' coordination between IP phones
  - mediates access to callee info in PSTN for IP side
- signaling gateway:
  - signaling operation translation (e.g., between SIP and SS7)
  - xlates call requests from either 'side' into corresponding representation for other 'side'

The Standards

- VoIP standards proposed by
  - IETF
  - ITU-T

It Started Off Well...

- ITU-T and IETF agree on the following common ground:
  1. audio is digitized using PCM
     - 8 bits per sample
     - 8000 samples per second
     - 64 kbps
  2. use RTP to move data (Real-Time Transport Protocol) [RFC3550]
     - not a transport-layer protocol
  3. RTP message carried via standard UDP
     - further encapsulated as IP

Why UDP?

- UDP provides
  - out of order datagram delivery
  - duplicates, missing data
  - little or possibly no checking on data
- overhead of TCP too high for real-time needs
- better to have a gap from dropped or error pkt than delay waiting for re-xmit
- introduce real-time clock values into datagrams
- can eliminate duplicates
- can reassemble in correct sequence

Time Sensitive (Multimedia) Data

- voice data is carried in UDP datagrams
- needs (at least) timestamps
  - may need other information
  - need for a protocol that can carry time sensitive data via UDP
  - what other requirements do we have?
Time Sensitive (Multimedia) Data

- some additional requirements:
  - notification of missing packets
  - respond to congestion
  - control messages
    - e.g., so recipient can notify sender of congestion
  - frame boundary notification
  - strings for identification

RTP

- Real-time Transport Protocol (RTP) [RFC3550]
  - really a transport protocol?
  - Real-time Transport Control Protocol (RTCP)
  - supports wide range of 'real-time' data
    - audio, video
  - defines, for each class of real-time data:
    - profile: guides understanding of header data
    - 1 format: how data after hdr is represented

RTP Message

0 1 2 3
0 1 2 3
4 5 6 7
8 9 0 1

Packet: 0 1 2 3 4 5 6 7 8 9 0 1

Padding [1]

Extension header [1]

Payload Type [7]

version = 2

Payload: 0 1 2 3 4 5 6 7 8 9 0 1

Payload Type

CSRC Count [4]: number of contributing source identifiers

Sequence number [16]: count of packets

RTP Message

0 1 2 3
0 1 2 3
4 5 6 7
8 9 0 1

Packet: 0 1 2 3 4 5 6 7 8 9 0 1

Timestamp [12]

SSRC Id [32]

Synchronization source (SSRC) Identifier

Contributing source (CSRC) Identifiers

CSRC Idents [32 X 32]

RTCP

- main functions:
  - app performance, network feedback
    - for adaptive rate applications
  - synchronize different media streams from same sender
    - e.g., audio and video may be sent separately
    - streams may have different clocks, different drift
    - represents identity of sender for display in UI
RTP: an example of ALF

- ALF: application layer framing
  - use to design new protocols for multimedia content
  - notion: only the application really best knows its requirements

Moving Datagrams Is The Easy Part

- once endpoints are known, moving datagrams from src to dest is ‘easy’
  - signaling: work done to manage infrastructure:
    - set-up a call
    - tear-down a call
  - current PSTN uses “Signaling System 7” (SS7)
  - any IP elements in VoIP must interoperate with SS7 to use any PSTN elements

Standards, Again

- IETF proposes:
  - Session Initiation Protocol (SIP)
- ITU-T proposes:
  - H.323 group
  - jointly have proposed:
    - Megaco, MGCP

SIP Terminology

- user agent: entity originating or receiving a call
  - user agent client: originates a call (outgoing)
  - user agent server: receives a call (incoming)
- location server: provides user-specific info, e.g., services user subscribes to
- user preferences
- proxy server: handle routing, enforce policy
- redirect server: forwarding, toll-free connections
- registrar server: applies authenticated updates to db used by location server

The SIP Method(s)

- SIP defines methods (message types):

  INVITE: endpoint invited to join a session
  ACK: acknowledge INVITE request
  BYE: terminate session: end call
  CANCEL: cancel pending request
  REGISTER: URL where user can be reached
  OPTIONS: query callee’s capabilities

Sample SIP Session
**Who You Gonna Call?**

- IP targets identified by IP addresses
- telephones ‘identified’ by telephone numbers
- ITU-T E.164 telephone number standard
- different formats in different places
- how do you identify callee?
  - by phone number?
  - by IP address?

**Mapping Phone Numbers**

- ENUM protocol [RFC 2916]
  - reverse string holding phone number
  - treat each digit as segment of domain name
  - in special domain e164.arpa
  - e.g., phone number 1-800-555-1244 becomes: 4.4.2.1.5.5.0.0.8.1.e164.arpa
  - is 1:many mapping...DNS server replies with:
    - list of hosts with that phone number
    - protocol needed to reach each

**Finding How to Reach Users**

- TRIP: Telephone Routing over IP [RFC 3219]
  - location servers use to advertise known ‘routes’ to each other

**H.323 Terminology**

- terminal: provides IP telephone function
  - for voice
  - for video and other data as well
- gatekeeper: location and signaling functions + coordinates gateway access to PSTN
- gateway: interconnect IP to PSTN networks
- MCU: services including multipoint conferencing

**H.323 Family**

- H.323 is collection of protocols that collectively perform all telephone operation
- uses both TCP and UDP for transport
- not ASCII-based; uses BER to represent ASN.1 binary messages

**H.323 Family**

- Comer, Fig. 33.4 shows the family:
International Packet Communications Consortium

- formerly ISC: International Softswitch Consortium
- industry-formed group
- define standards that can be implemented and put into service
- attempt to embrace all situations...

IPCC View

- **MCC-F**: keeps state info in endpoints, provides call logic and call control
- **Ca-F**: maintains call state (e.g., SIP, H.323, Q.931)
- **W-F**: signaling between different nets (e.g., SS7 —> SIP)
- **RF/AF**: Routing of calls/Accounting info re. calls
- **SC-F**: signaling IP network —— PSTN
- **ACS-F**: signaling IP network —— circuit-sw access net (e.g., ISDN)
- **MG-F**: digital audio xlate

IPCC View

- **AS-F**: application services (e.g., voicemail)
  - **SC-F**: invoked when AS-F controls a service
  - **MS-F**: handles media pkt stream for AS-F app
- **MG-F**: digital audio xlate
  - may also recognize DTMF tones and phone device events (like OHI)