Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU
kre@coe.psu.ac.th

http://fivedots.coe.psu.ac.th/~kre

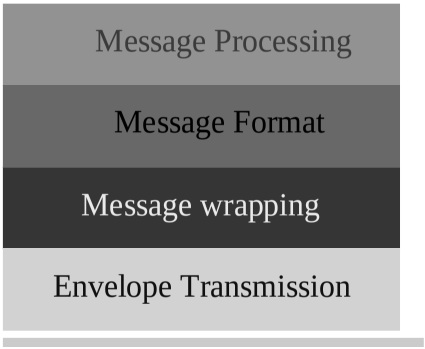
E-mail

- One of the oldest network applications
  - After Remote Login & File Transfer
- Network designed to allow remote access
  - Remote login a requirement
- To work on remote system
  - Need to transfer data
    - Input data for program
    - Output to be examined/printed
  - File Transfer a requirement
- With File Transfer Protocol (FTP)
  - Can send file to remote system
  - there deliver to user
    - rather than store in named location
  - This is e-mail
    - An offshoot of FTP

Design of E-Mail

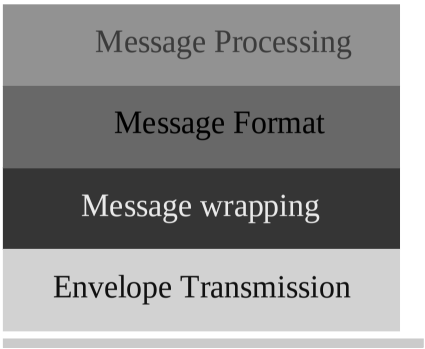
- Models Office memorandum
  - Message from one person to another
  - Semi-formal setting
  - Rules for format
  - Rules for processing
- Not a model of letters
    - Not ordinary postal mail
  - Not structured enough
  - Too hard to automate processing

E-Mail protocol layering



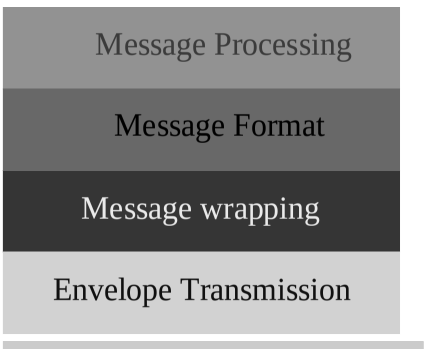
- Multiple layers of protocol
  - All in the application layer
  - Above the sockets interface

Message Processing



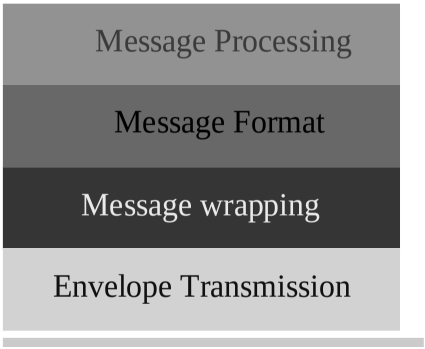
- Rules/Procedures how to handle messages
  - When to send replies
  - Where to send replies
  - How important message is
  - (and more)
- Largely implemented in the humans
  - With software assistance

Message Format



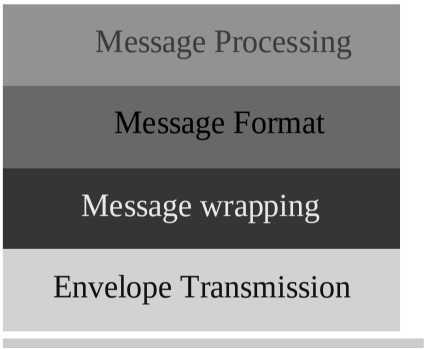
- Specification of message format
  - Useful so software can assist humans
  - Also helps humans
    - Less ambiguity as to meaning

Message Wrapping



- How to put message in envelope
  - Rules for writing envelopes
  - What kind of message fits
  - What envelope means when received
- Getting message out of envelope

Envelope Transmission

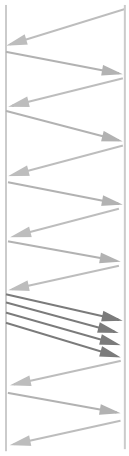


- How to send envelope to recipient
  - Envelope containing message
- What to do with messages that
  - cannot be delivered
- How to do that

Simple Mail Transfer Protocol

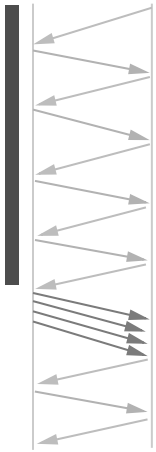
- Protocol between MTAs
    - Mail Transfer Agent
  - Also often used for message submission
    - From MUA to MTA
      - Mail User Agent
- Aim is reliable mail transfer
  - At least one MTA is always responsible
  - Whatever happens
- Very Simple protocol
  - Startup dialog
  - Identify sender
    - The envelope sender (becomes Return-Path)
  - Identify recipients
    - Envelope destination addresses
  - Send the message
    - Including all message headers
  - Terminate
    - Or repeat for a new message

Passing Responsibility



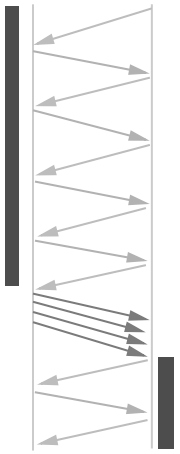
- Examine which system
  - is responsible for safety of e-mail

Sender responsible



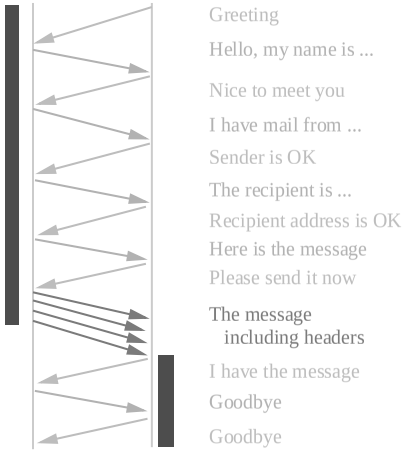
- Before message is sent
  - Sender must be responsible
  - Recipient has no data yet
    - Just the envelope
    - Or parts of it

Recipient Responsible



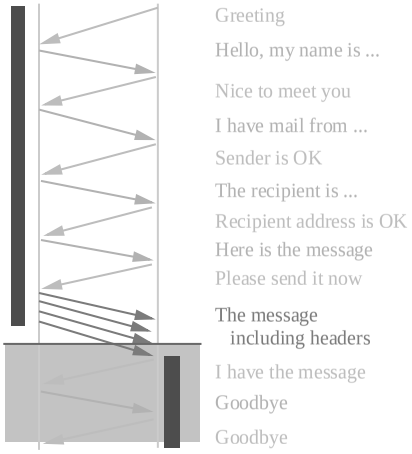
- After message is fully received
  - Recipient can become responsible
  - Now has complete message
    - plus complete envelope information

Sender letting go



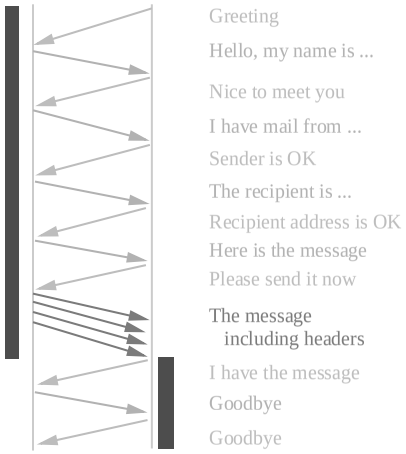
- But when does sender responsibility end?
  - When end of message is transmitted?

Sender letting go (2)



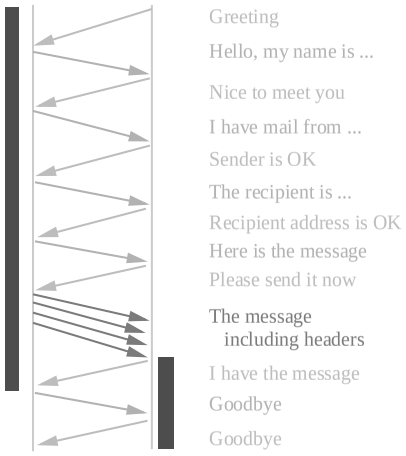
- What happens if system crashes here?
  - After end of message is transmitted
  - Before it arrives at recipient

Sender letting go (3)



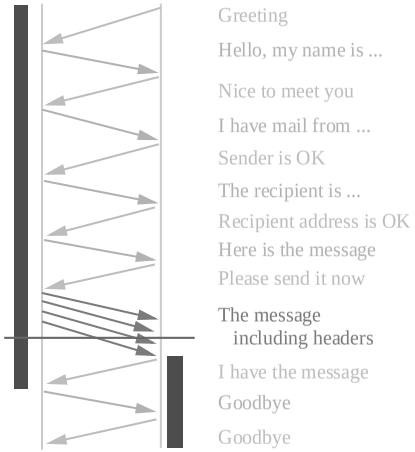
- Really want this
  - Sender responsible until recipient takes over
  - But how does sender know when that is?
    - Nothing happens at sender at this point!

Sender letting go (4)



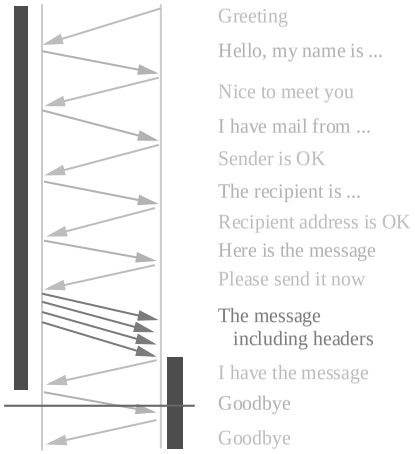
- Must wait until sender knows
  - That recipient is responsible for the data
  - After sender receives message
    - from recipient saying I Have The Message

System crashes



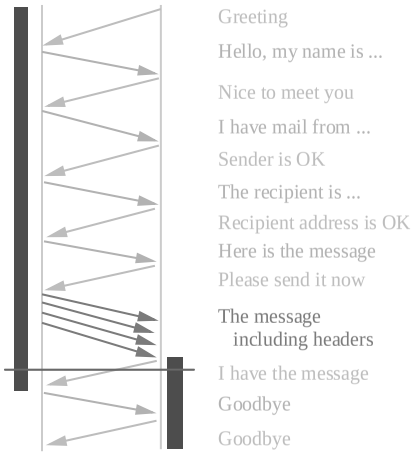
- If system crashes here
  - Sender still responsible
  - Retains the message
  - Can transmit it again later

System crashes (2)



- If system crashes here
  - Recipient has the message
    - Can deliver now or later
  - Sender has removed it
    - Everything good.

System crashes (3)



- If system crashes here
  - Both systems are responsible
    - Recipient delivers message
    - Sender retransmits later
    - 2 copies delivered