The End-to-End Argument: Saltzer, Reed, Clark [Saltzer81a]

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

Key ideas

- end-to-end argument:
  - given a service S, where should S be implemented?
  - "push S up as high as possible"?
  - argument: if you do S at a lower layer, you will *also* have to do it at higher layer, and so the lower layer version of S will be redudnant
    - why do we have to do at at the higher layer?
      - app may have very specific requirements
      - if done at lower layer, then it must be in ALL lower layers (and this is hard—see [Clark88a])
      - responsibility for correcteness falls back on the app in the end
    - why is redundancy bad?
      - overhead in lower layers (sometimes)
  - one other important point to consider…see later
- keep with the end-to-end principle or not?

Example: Reliability

- Consider copying a file
  - want an *end-to-end* checksum
- steps:
  - A reads from disk to mem; sends to network
  - network moves data from A to B
  - B gets data from network; writes to disk
- possible faults:
  - disk IO errors, buffer overruns in NIC, memory errors, network corruption or congestion, computer crashes

Other examples in paper

- encrypted data transfer
- duplicate message suppression
- guaranteed FIFO message delivery
- transactions in a DB

Other examples?

- encryption
  - wireless and WEP
    - not good enough for "full security"
      - ex: end host could be compromised
      - ex: other hops in the network aren't encrypted
- reliability in the net?
  - sending long-distnace messages in the Internet: if they're corrupted, detect and discard them early
  - maybe somethings can *only* be detected in the net
    - ex. Denial-of-service attacks
  - be careful: e2e argument doesn't say NEVER do things in the middle, it says if you *must* do it at the end, ask yourself if *also* doing it in the middle is helpful

Caveat: performance

- Consider file copy again
- Reliability at physical, link, network, transport, application layers
  - minimial link-layer checksums to discard corrupted packets
  - link-layer ARQ could retx single packets
    - still have to do end-to-end retx
    - but over very long connections, hop-by-hop repair could be helpful
    - could be very important if you have a link layer with high drop rates (maybe in wireless nets, or satellite)
- multiple levels may be needed for *performance*

Challenge: Defining the "End"

- ex. security: what is the protocol and end for each app?
  - with WEP, ends are wireless host and base-station
  - VPN (Virtual Private Network): ends are your computer and USC's VPN server
  - buy something from amazon, your browser to their web server
    - vulnerable to snoopers over my shoulders, or amazon's database may be compromised

Why is End-to-End Important to Network Design?

smart vs. dumb?	**telephone**	**Internet**
network	*smart*	*dumb*
terminal	*dumb*	*smart*

- e2e is about efficiency
  - but smart networks can become very application specific
  - therefore become less able to adapt to new applications

Other questions/observations?

- moving away from e2e? (Blumenthal summarizers?)
  - maybe security forces us to move away for e2e?
- xxx