Reliability from unreliability
What the Internet provides

● “Best-effort” delivery of datagrams
  ○ up to about 1,500 bytes
  ○ from one computer to another

● “Best-effort” means datagram might be:
  ○ lost completely
  ○ delivered more than once
  ○ delivered after another datagram that was sent later
  ○ delivered with some bytes changed
  ○ delivered but truncated
What most users and applications want

- Reliable retrieval of a short piece of data
  - “What’s the IP address that corresponds to cs144.keithw.org?”
- Reliable action
  - The text of Keith’s message #7 is: “Fire a torpedo!”
- Reliable byte stream
  - Sequence of bytes (in each direction) delivered in order, correctly
- Reliable delivery of a large file (FTP, SMTP, HTTP)
  - “Subject: Homework. Dear Professor McKeown: Here is my 20 MB file.”
  - “Subject: Re: Homework. Thank you! Please call me Nick.”
- Reliable remote procedure call (RPC) (HTTP/1, HTTP/2, HTTP/3, gRPC, Thrift)
  - POST /bankaccounts/checking/billpay HTTP/1.1
    amount=270,000&payee=StanfordSailing&memo=admitmychildplz
A module behaves **reliably** when it:
- provides **some** stated abstraction/interface
- even in the face of underlying faults (e.g. packet loss)
- and when it can’t do that, the module signals failure.
The big question

How to provide these abstractions reliably on top of an unreliable system?
TCP in a nutshell

» datagram that says \texttt{bytes 0..49 of the byte stream have the contents: “MAIL FROM: \texttt{<thepope@vatican.va>”}}

» datagram that says \texttt{bytes 50..99 of the byte stream have the contents: “DATA\nHi Keith here is your ordination.”}

« “The next byte of the stream that I need from you is \#0.”

» datagram that says \texttt{bytes 0..49 of the byte stream have the contents: “MAIL FROM: \texttt{<thepope@vatican.va>”}}

« “The next byte of the stream that I need from you is \#100.”