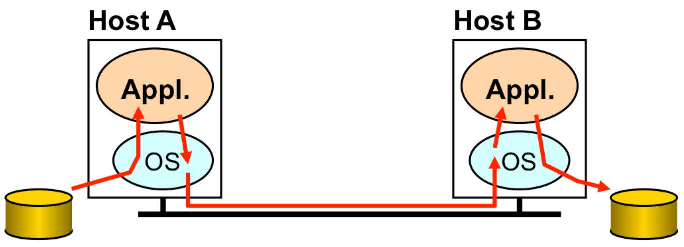
Sample Question: End-to-End Arguments in System Design (Please discuss)

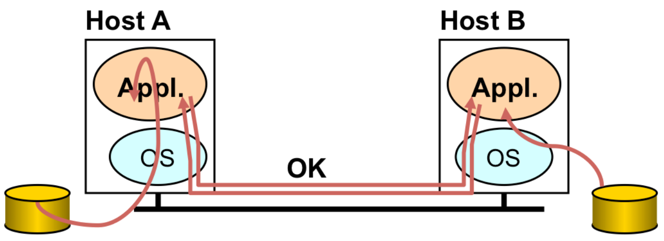
This question is centered around the fundamental paper "End-to-end Arguments in System Design" which is available here - Saltzer (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf)

Consider that we wish to implement a reliable file transfer protocol.

One approach is to make each step of the file transfer reliable and string them together to make the end-to-end process reliable. This is depicted in the figure below.



The other approach is to not worry about implementing reliability along each step but rather to make use of an end-to-end check (simple error detection at the receiver followed by feedback to the sender) and retransmit if necessary. This is depicted in the figure below.



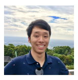
Discuss the pros and cons of each approach. Which approach would you pick?

Resource created about a month ago (Monday 16 July 2018, 02:50:37 PM), last modified 26 days ago (Friday 27 July 2018, 11:18:33 AM).

Comments

(/COMP3331/18s2/forums/search?forum_choice=resource/17226) (/COMP3331/18s2/forums/resource/17226)

Add a comment



Michael Yoo (/users/z5165635) 16 days ago (Mon Aug 06 2018 16:53:32 GMT+1000 (Australian Eastern Standard Time)), last modified 16 days ago (Mon Aug 06 2018 16:54:17 GMT+1000 (Australian Eastern Standard Time))

Pros of a reliable system at each layer and step:

- Consuming applications and upper layers do not need to worry about the lower level's reliability, which may simplify implementation.

Cons of a reliable system at each layer and step:

- Effort to make a reliable system at each layer increases exponentially for each step of increased reliability.
- Duplicated efforts to ensure reliability at each layer costs time and computing power.
- There are many ways a system can fail. Some of these failures are unrelated to the networking layer (i.e. disk buffering failure), thus no amount of effort expended to make networking layers more reliable will guard against these issues.
- Networking layers do not have domain-specific knowledge of correctness applied to the data being transferred.

Pros of an unreliable system and end-to-end check:

- Regardless of underlying conditions, the application can guarantee its own correct behaviour.
- Applications can use domain-specific knowledge to apply correctness. For example, video streaming does not require perfect correctness.

Cons of an unreliable system and end-to-end check:

- Every application will need to implement their own end-to-end checking system. This imposes the burden of buffering on individual applications.

  The conclusion of the paper is that network reliability does not fix non-networking corruption anyway, so it's better to use end-to-end check?

  Reply