End-to-end Arguments in System Design

CS380L: Emmett Witchel

January 30, 2025

> "Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer.
>
> Thus the amount of effort to put into reliability measures within the data communication system is seen to be an engineering tradeoff based on performance, rather than a requirement for correctness.
>
> The argument... provides a rationale for moving function upward in a layered system, closer to the appliction that uses the function.
>
> The end-to-end argument is not an absolute rule, but rather a guideline that helps in application and protocol design analysis; one must use some care to identify the end points to which the argument should be applied."
> *Saltzer, Reed, & Clark, "End-to-end Arguments in System Design*

1 File transfer

1. At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk-format independent.
2. Also at host A the file transfer program asks the data communication system to transmit the file using some communication protocol that involves splitting the data into packets. The packet size is typically different from the file block size and the disk track size.
3. The data communication network moves the packets from computer A to computer B.
4. At host B a data communication program removes the packets from the data communication protocol and hands the contained data on to a second part of the file transfer application, the part that operates within host B.
5. At host B, the file transfer program asks the file system to write the received data on the disk of host B.

The authors enumerate the following threats to this process

1. The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.
2. The software of the file system, the file transfer program, or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.

3. The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.

4. The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.

5. Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

Are there additional failure points? Which of the five are most likely today? Was that the most likely failure point when the paper was written?

Only an end-to-end check would result in a file transfer program whose probability of failure was proportional to the file size. Where does the retry go?

- Making the communications system more reliable allows all applications that use it to benefit, lowering overall system cost.

- Making the file transfer program do the retry means it can be more efficient.

"Thus the argument: in order to achieve careful file transfer, the application program that performs the transfer must supply a file-transfer-specific, end-to-end reliability guarantee...for the data communication system to go out of its way to be extraordinarily reliable doe not reduce the burden on the application program to ensure reliability."

"...there may be some application that finds the cost of the enhancement is not worth the result but it now has no choice in the matter."

What happens in reality? It depends on the application.

- TCP and UDP co-exist. The Network Appliance company sells a souped up NFS server. They recommend using TCP sometimes and UDP somethimes as NFS's transport protocol depending on the properties of your network.

- TCP provides reliable transmission in the communication layer. In-order, reliable packet delivery simplifies applications. But that is not the right trade-off for some applications. The low-bandwidth filesystem (LBFS SOSP '01) has a custom retransmission scheme for files to save bandwidth.

- Many research projects consist of pushing functionality toward the application and reaping performance improvements that are possible when reducing complexity (e.g., exokernel).

- Multicast is supported directly in the IP layer, but no one uses it. It doesn't scale, and alternatives implemented at a higher level, like internet relay chat (IRC), are available. One problem with multicast is security, and the IP security standard IPsec won't work for multicast (oops).

- Wireless networks often have variable reliability to the point where complex functionality is built into lower layers.

- Who should integrity check files? An application (e.g., tripwire)? For security, must run as root. To guard against corruption, maybe the file system (which can do so periodically).

2 Communications

Acknowledgement of a message is nearly useless. What if host gets message, but fails to act on it?

What if host were structured that if it got the message, it would (atomically) act on it? Is that an end-to-end guarantee?

Should the network do duplicate detection? Authors argue that there are application level duplicates, e.g., multiple login attempts and host failure detection.

"Eventually, it became standard practice for every application to provide its own application-dependent checks and recovery strategy" Google file system requires applications to link a library that detects some file system errors.

2.1 Encryption in the network

- `ssh`, the Linux standard for encrypted login and creating encrypted network connections, does not rely on network-level encryption. IPsec is not widely used.
- Encryption is the best way to secure your wireless router from your neighbor.
- DNSSEC is greatly needed, but deployment is difficult.

Why does the end-to-end argument apply to `ssh`, but not apply to wireless routers?
Remember, retry extends right up to humans.

- "What did you say?" is how we deal with the abysmal cell phone service in America.
- Landlines occasionally fail to give a dialtone.
- Compare programs to old listing manually.
- Keep multiple copies of files on multiple systems to avoid catatrophic failure.
- Airline ticket agent will retry a reservation, no need to hold onto pending reservations during a system crash.
- Duplicate data in the Google file system is allowed. A user-level library filters them.
- Torvalds interview - avoid system interfaces.

Does the end-to-end argument reinforce or contradict Gabriel's "Worse-is-better" argument?