TCP Over Email

Venkat Arun, Akshay Narayan, and Mercury

Abstract

In response to recent trends in e-mail usage patterns, we propose "Reliable e-mail", or *remail*. *remail* implements reliable delivery of email messages end-to-end, in accordance with the end-to-end principle. This is a timely contribution, in light of increasing software complexity, and Software 2.0 [3]

1. Introduction

Long-time trends indicate a shift toward higher-level abstractions in computing. What began as paleolithic counting devices to track physical goods, lacking even an operating system or standard instruction set architecture, has slowly evolved into the widespread use of seemingly magical light-boxes which can handle almost all problems. As these magical light-boxes evolve, they insist on acquiring an essential trait of their creators; fallibility.

In the old days, before abstractions, the designers of these devices[1] created a *layered architecture* for computers to communicate, based on the principle of separation of concerns. The lowest layer of this hierarchy is the physical layer, which operates on similar principles as the aforementioned paleolithic counting devices. Above this, we must ascend into abstraction: the link layer, the IP layer, the transport layer, etc. After some experimentation [1], a consensus design emerged with the IP layer as a "narrow waist"; that is, a globally deployed and understood abstraction used to inter-network the communication fiefdoms of the time. Above this IP layer, the transport layer implements reliable datastream delivery to applications.

Modern society, however, has developed another narrow-waist: e-mail (seen in Figure 1). E-mail is the common interface to civilized life: it either is or is a crucial component of a social network, a method of commerce, a newspaper, a travel agent, and more.

Therefore, just as the internet (that is, IP) was too important to be entrusted to reliably deliver packets, email systems are too complex to be entrusted with the reliable delivery of email. Signs of this increasing complexity can be witnessed in the failing email system of CSAIL, MIT. This failure is a testament to MIT's quest for intelligence, a decades long struggle to make machines more human-like.

Our approach comes at a crucial time, with large-scale reliability issues plaguing a significant fraction of the IPv4 address space[2] in recent months. We stand at a crossroads in society: either we can start the slow backslide into unreliable communication and give up on email reliability, or we can decide, as the creators of the Internet did, to add more abstraction to hide our problems, passing the buck down to future generations just as our ancestors have taught us. We choose the latter option, and propose to make email more reliable using a classic reliability mechanism: the transmission control protocol (TCP [2]), which implements reliable, in-order bytestream delivery. Our contributions are:

1. A summary of existing human-centric approaches to email reliability (§2).
2. A survey of the future of autonomous email systems (§4.1).
3. A discussion of related work and historical context (§3).

[1] Now known as "computers" or "computer systems"

[2] Except the part MIT sold to Amazon.

DOI: http://dx.doi.org/10.1145/(to come)

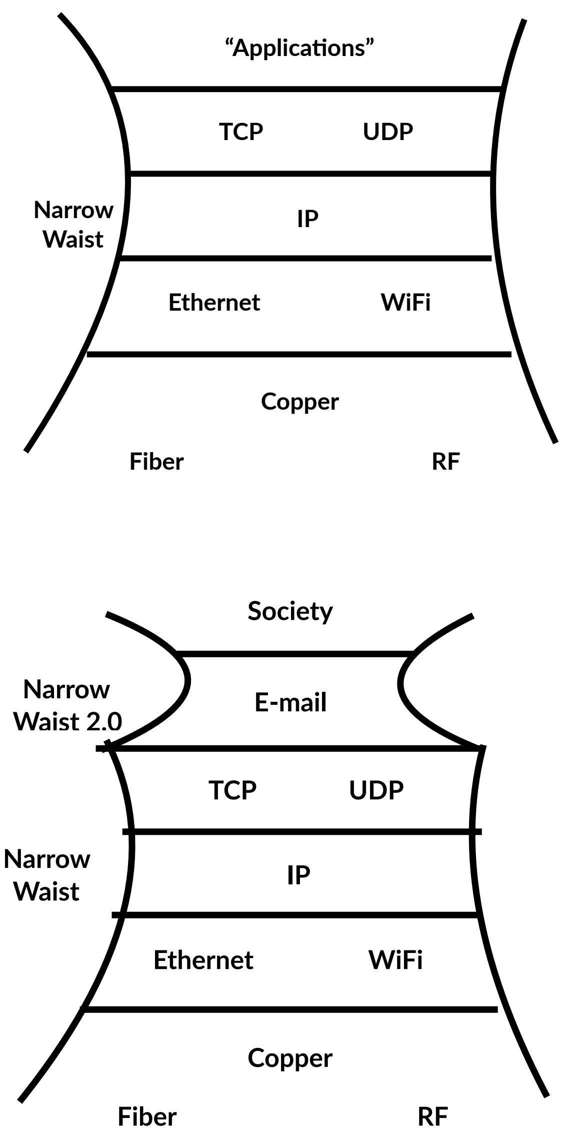

Figure 1: A narrow-waist view of the Internet. Recently, a new narrow-waist has emerged.

4. An evaluation of our approach (§5).

2. Human Assisted Out-of-Band Reliability

A naive protocol would require the recipient of every email to acknowledge receipt. The sender must retransmit until they get an acknowledgement. This guarantees that the message reached the recepient's brain. The brain may then proceed to forget this message, however building reliability protocols for the 'brain layer' is left for future work.

This protocol is entirely in-band, and relies purely on email. The challenge is that acknowledgements can also fail. This problem has been dealt with by TCP; it simply keeps retransmitting packets and ACKs until something gets through. However, email is more complex than a simple packet delivery system, and is hence more prone to failure. Therefore, we propose an out-of-band reliability system. In addition to sending emails, users are expected to drop by each others' offices/homes to discuss what emails were sent and which were actually received. If physical distances make this impractical, they can fall back to the phone system. Telephone calls are more reliable than email. This is because the technology has become rather stagnant, suffering from an acute lack of innovation. The alarming reliability of phone calls is reminiscent of face-to-face conversations, which like phone calls, are an ancient technology in rapid decline.

In addition to increasing reliability, out-of-band reliability mechanisms confer two additional *social* advantages. First, it increases the probability that two people will have a face-to-face or voice-based conversation. Second, since the burden of reliable email delivery is no longer solely on the shoulders of the email systems, recipients of incessant dog/cat/baby pictures can claim they thought the dog/cat/baby was really cute, but the response email singing its praises got lost in transit. It was really the dog/cat/baby admirer's fault for not having dropped-by/called-in to check whether the recipient thought the dog/cat/baby was cute. This does however increases the risk that the dog/cat/baby admirer would actually drop-by/call-in. We leave the investigation of mechanisms to mitigate such risks to future work.

We depict this out-of-band reliability protocol in Figure 2.

3. Historical Context

Some early readers of our paper reported indigestion regarding this idea. They think email reliability is a fixable problem, and adding another layer of reliability is entirely unnecessary. To put an end to such delusions, we provide some historical context. Electrical engineers building the first communication systems, thought they had it all figured out until they realized mother nature would always mess with their bits. Then Shannon proved that you could always add enough extra bits to keep mother nature at bay. However he didn't account for human stupidity. We humans kept tripping over wires, and designed protocols that kept talking over each other. What could poor Shannon do when en-

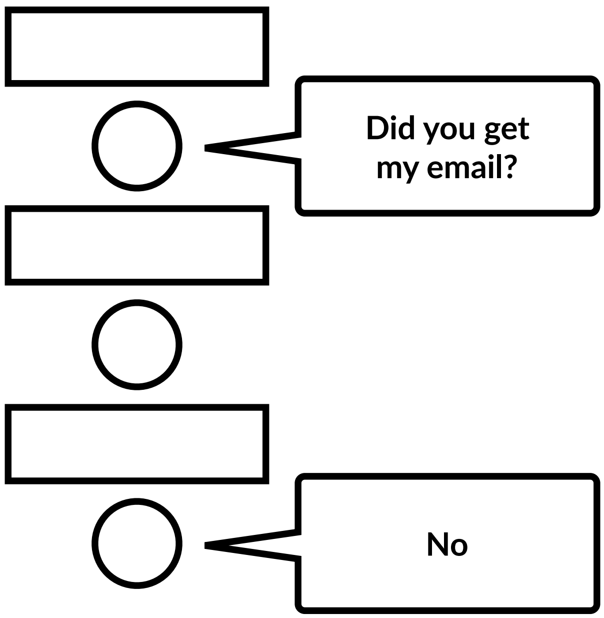

Figure 2: Realistic, to-scale depiction of human-centric out-of-band email reliability protocol. Upon receipt of negative email acknowledgements, participants wallow in despair and send email rants to organization-wide mailing lists. Predictably, these rants are also not received.

tire batches of bits get lost? So we introduced retransmission in the link layer, thereby making many of our link layer protocols reliable. Aha! So a network composed purely of reliable links must be reliable, right? Wrong. People never know how fast to send packets. When they send too much, the network is obliged to ignore them (this is analogous to how we ignore dog/-cat/baby pictures). Ok, so we'll have reliability in the transport layer (TCP), and since this is end-to-end, we should be all set. Nope. Different link-layer designers decided to have different MTU (maximum transmission unit) sizes, do the IP layer must handle fragmentation and reconstruction. Further, just like their human creators, network layers also mistrust each other. Hence IP adds another level of checksums to ensure the bits got transmitted correctly. Then TLS operating on top of TCP, authenticates all messages, just in case someone tries to maliciously change bits in transit.

In summary: physical layer has some forward error correction. Then the link layer have have additional forward error correction and detection, in addition to retransmission. IP has an additional checksum, and handles fragmentation and consequent reconstruction. Then TCP implements end-to-end reliability, which TLS checks to look for malicious changes. Given that we have reliability *every step of the way* in the computer world, is it really outlandish to add reliability in the human-layer as well?

4. Autonomous Email

As we push toward a better separation of layers in the network stack, future systems will achieve a near-complete separation of the 'human-layer' from the 'email-layer'. Humans will no longer need to be concerned with the contents of email. Email systems are already adept at telling users what emails to read, which emails to respond to and what to write in said responses. Future systems will go a step further and automatically draft emails and their responses. They will autonomously schedule meetings and politely decline invitations to meetings, advertise talks and forget to attend advertised talks, generate spam emails for hypothetical products and delete spam emails for hypothetical products, send cat pictures and produce 'ooh' and 'aaw' sounds in response. We humans will be free to ponder over such essential questions as the meaning of life, and why the pizza we ordered over 2 minutes ago isn't at our table yet.

4.1 Autonomous Reliability

In a world where most emails are written and read by autonomous systems, human-based reliability mechanisms will fail, since most emails would never reach a human. Therefore we propose an automatic reliability layer, in addition to the email reliability layer. As discussed above (§2), a purely in-band mechanism is not guaranteed to work. While this may appear to be a key limitation of our work, we argue that the added confusion will keep our robots busy, and prevent them from plotting the end of humanity.

4.2 Implementation of Autonomous Reliability

Our system, *remail*, paves the way to reliable email by re-implementing the mechanisms of TCP at the layer above email. When the user sends an e-mail, an email with the text "SYN" is sent to the recipient along with a randomly chosen sequence number. Upon receipt of this email, the recipient responds with the text "SYNACK", also with a randomly chosen initial sequence number. For all subsequent emails, the beginning of the email contains a cumulative ACK of the most recent consecutive email received. Using well-known TCP mechanisms, we can then ensure that the appropriate emails are re-sent they are not acknowledged. Recognizing the parallels between email and TCP reliability is a key contribution of our paper.

In addition to providing reliability, our approach further allows for new email features. For example, because messages contain increasing sequence numbers, it is possible to order messages within a conversation, a feature currently implemented only by heuristics and copying message bodies unnecessarily. Furthermore, because TCP is a one-to-one protocol, our approach disallows the "reply-all" functionality, thus improving the email experience.

5. Evaluation

As our proposal is clearly correct, an evaluation is unnecessary.

References

[1] D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM*, 1988.

[2] IETF. TRANSMISSION CONTROL PROTOCOL, 1981. RFC 793.

[3] A. Karpathy. Software 2.0, November 2017. [Online; accessed 1-April-2019].