Approximate Semantics for Wirelessly Networked Applications

Benjamin Ransford and Adrian Sampson and Luis Ceze

University of Washington

Abstract

Approximation saves energy in computation and storage by allowing data to diverge from precise representations. In contrast, networks guarantee that transmissions are received precisely by enforcing integrity checks at multiple layers of the protocol stack; these checks result in onerous retransmissions and increased network contention especially in wireless networks. When energy is limited, such as on embedded systems, conservative network semantics can needlessly consume an application's energy budget. Why should approximate data be transmitted precisely?

1. Introduction

Low-level network layers abstract away details such as error handling to provide an interface for data to be received exactly as it was transmitted. This abstraction is not well matched to applications that deal with approximate information: both senders and receivers expend time and energy trying to ensure data integrity. Application designers cannot easily express a desire to relax integrity requirements.

A wealth of past research has proposed ways of recovering information from erroneous transmissions [1, 4], of reducing redundancy in transmissions [9], and of rearranging data to make decoding errors less damaging [8]. Our goal is to allow application designers *who are using approximation in their applications* to communicate approximate data efficiently. Relaxing integrity requirements for approximately data may allow for faster transmission, longer range [6], and reduced power requirements.

Our proposed approach is to allow for different treatment of precise and approximate data across the network stack. We provide applications with a configurable switch that toggles between precise and approximate behavior, making the necessary adjustments to lower layers of the network stack. It complements other approximation techniques for computation and storage, endowing approximate applications to naturally extend their capabilities to the network.

Applications that may benefit from approximate communication include distributed sensors that stream real-time readings, media playback, and even applications that require some transmissions to remain precise, such as HTTP or remote desktop.

2. Building Blocks for Approximate Networking

To retain many of the benefits of current networking techniques (straightforward APIs for sending and receiving data, well-tested network stacks, etc.), we propose working within the common layered network model, rather than, for example, creating new primitives atop software-defined radio.

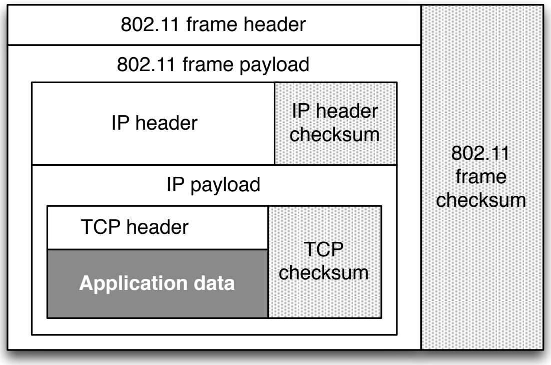

Figure 1. A TCP-over-IPv4 packet on WiFi carries three separate checksums (shaded). We propose to drop the outermost checksum and make the innermost checksum optional.

Figure 1 depicts the layered representation of application information in a modern network, using 802.11 (WiFi) for the lowest layers and TCP as the transport protocol. Even a single bit flip in the application's data payload will trigger a retransmission in the 802.11 MAC layer. The TCP packet's checksum computation is redundant with the 802.11 frame's.[1] A better match for approximate data transmission would have the following properties:

1. *Optional* integrity checks at multiple layers, to permit errors in approximate payloads.
2. *Partial* integrity checking to make sure control data (addresses, ports, etc.) are precise.
3. *Backward compatibility* with existing network stacks to avoid networks between the sender and receiver flagging the traffic as strange.

[1] IPv6 packets do not carry their own header checksums, depending instead on the surrounding layers (e.g., 802.11 and TCP).

4. *Simplicity* of integration with existing applications.
5. *Optional* TCP-like behavior such as exponential backoff.
6. *Quality metrics* to measure and control the correctness of the approximation, as in other approximation systems.

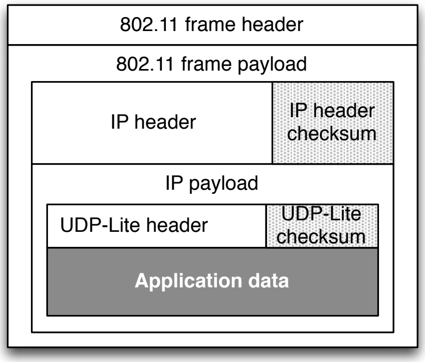

Figure 2. In our revised model, a single, variable-size checksum guards some fraction of the application payload.

UDP-Lite and selective approximation. Our approach to approximate networking on 802.11 is based on:

- UDP-Lite [3] with configurable *checksum coverage*—specifying how many bits are protected by a checksum—in place of TCP for application data. UDP-Lite is already implemented in mainstream network stacks, so routers will not discard it (unlike damaged TCP packets).
- A user-space socket library that allows applications to specify approximate or precise transmission (and switch at will) for a given socket.
- A kernel-space switch that toggles 802.11 frame integrity checking in the WiFi device driver.

Figure 2 depicts the application data of Figure 1 recast in a UDP-Lite packet under our scheme. The application can control how much of the IP payload is protected by a checksum, optionally protecting important header information or metadata. When the unprotected portion is corrupted, no checksum catches the error. An application may evaluate the payload and decide whether to request retransmission.

Our approach exhibits all but the last two desirable properties mentioned above.

Measuring and controlling quality. Previous approaches to approximation have resulted in application-specific quality metrics (e.g., PSNR for video). Our communication mechanism can adapt these metrics for use with transmitted payloads; bit error rate (BER), for example, is meaningful in many contexts. An API to specify maximum error tolerance is future work. One challenge endemic to this setting is that the sender and receiver are likely to be different entities, so obtaining ground truth against which to measure quality is difficult. A viable strategy may involve computing multiple partial checksums and requiring a quorum of "good" blocks, in the manner of PPR [1].

Another promising approach to bounding error is to add error correction at the application layer, giving applications the ultimate responsibility for quality measurement. Software libraries for forward error correction such as libfec [2] can allow applications to specify the error resilience of their transmissions. Moving error detection and correction up the stack has been shown to improve throughput in many cases [4].

Adding back connection state. Unlike TCP, UDP is stateless; the client and server have no notion of a *connected* socket that provides in-order delivery and survives a session. The traditional approach for applications using UDP is to incorporate state into higher layers. This aspect of our protocol is as yet unimplemented.

Wired networks. Approximate networking is most readily suited to wireless networks, where errors are the common case. Wired devices such as cable modems also use forward error correction to cope with long cable runs and line noise; approximate transmission may ease the burden for high-throughput applications such as video streaming. Other networks in which the last hop is the least reliable are likely to derive a similar benefit. Fewer errors at the last hop results in less backpressure on links along the entire path from the source.

3. Challenges for Approximate Networking

Approximate networking provides a "noisy channel" that poses challenges for the application layer. In general, these challenges are common to any approximate system that exposes random noise, including approximate memories [5, 7].

Precision granularity. Approximate applications use a mixture of error-resilient and error-tolerant data [5, 7]. To avoid excessive switching overhead, software should change precision levels at a coarse granularity, but application constraints may sometimes demand finer-grained interleaving. An approximate networking protocol will need to choose the finest granularity at which approximate and precise data transmission may interleave as a compromise between network-level efficiency and application-level needs.

Encryption. Encrypted network data is intolerant of small transmission errors. In fact, this is a desirable property for cryptosystems: for security, *any* change in the plaintext should produce completely different ciphertext. Therefore, even data that would otherwise be error resilient—images, audio, video—becomes error intolerant when it is encrypted. The feasibility of approximate communication and storage channels depends on either a large proportion of unencrypted approximate data or the use of encryption schemes that isolate errors in the ciphertext. For example, a scheme that encodes data blocks independently—as opposed to blockchaining schemes, for example—prevents errors from corrupting any data beyond the blocks in which they occur.

Compression and encoding. As with encryption, compression techniques threaten the independence of errors that approximate computing systems depend on. If a node transmits a raw stream of samples, then a single transmission error can corrupt at most one data element. Compressed formats like JPEG, however, allow errors to contaminate the entire image. Common compression techniques such as run-length encoding lead to storage of metadata that lead to catastrophically poor output even with very small errors. In some circumstances, it may be more efficient to transmit uncompressed data approximately than to transmit compressed data precisely. But to fully exploit approximate storage and transmission, the approximate-computing community should explore compression mechanisms that avoid this error-sensitive metadata or separate it from the error-tolerant data so that metadata can be transmitted precisely.

4. Related Work

No single technique from the extensive literature on error-resilient communication is a direct match for the scenario we consider.

Previous work has proposed to expose and exploit physical-layer errors for more efficient correction and retransmission. While these approaches do not expose errors to the end-user application, the techniques are potentially complementary to ours and share our overall performance goals.

Sen et al. improve 802.11's resilience by exploiting the predictable structure of errors [8]. Jamieson and Balakrishnan [1] propose to use additional physical-layer information to recover from more errors. Zhang et al. study frame retransmission in 802.11 networks and propose *micro-acks* [9], a method of retransmitting only the missing or damaged portions of a frame. Lin et al. push error correction to software and mix correction with retransmission in ZipTx [4], increasing throughput. Reimann and Winstein take a similar application-layer approach to error tolerance to extend an outdoor network's line-of-sight range by 70% [6].

5. Conclusion

Approximate computing research has primarily focused on computation and memory systems—the traditional "core" of computer systems. Networking hardware, however, can consume resources on par with these traditional targets of architecture research, especially in mobile devices where energy efficiency is paramount. Approximate wireless networking can exploit the error resilience in transmitted data to avoid paying the high costs of error correction and retransmission.

Acknowledgments

This work was supported in part by C-FAR, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] Kyle Jamieson and Hari Balakrishnan. PPR: Partial packet recovery for wireless networks. In *Proceedings of SIGCOMM*, August 2007.

[2] Phil Karn. DSP and FEC library. http://www.ka9q.net/code/fec/, 2007.

[3] L-A. Larzon, M. Degermark, S. Pink, L-E. Jonsson, and G. Fairhurst. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828 (Proposed Standard), July 2004. Updated by RFC 6335.

[4] Kate Ching-Ju Lin, Nate Kushman, and Dina Katabi. ZipTx: exploiting the gap between bit errors and packet loss. In *Proceedings of MOBICOM*, September 2008.

[5] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. Flikker: Saving refresh-power in mobile devices through critical data partitioning. In *Proceedings of ASPLOS*, March 2011.

[6] Reina Riemann and Keith Winstein. Improving 802.11 range with forward error correction. Technical Report MIT-CSAIL-TR-2005-011, Massachusetts Institute of Technology, February 2005.

[7] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. In *Proceedings of MICRO*, December 2013.

[8] Sayandeep Sen, Syed Gilani, Shreesha Srinath, Stephen Schmitt, and Suman Banerjee. Design and implementation of an "approximate" communication system for wireless media applications. *SIGCOMM Computer Communication Review*, 41(4), August 2011.

[9] Jiansong Zhang, Haichen Shen, Kun Tan, Ranveer Chandra, Yongguang Zhang, and Qian Zhang. Frame retransmissions considered harmful: Improving spectrum efficiency using micro-acks. In *Proceedings of ACM MobiCom*, August 2012.