Life on the Edge Network

Heidi Howard
University of Cambridge
first.last@cl.cam.ac.uk

ABSTRACT

The end-to-end principle of the internet is a fallacy. Modern distributed system rely on the cloud rather than deal with the complexity of the edge network. We propose to explore how to provide primitives such as consistency, integrity, accessibility and authentication in the context of edge network distributed systems.

Motivation

The internet has abandoned the end-to-end principles on which it was established [2]. With IPv4 addresses depleted and the transition to IPv6 yet to restore public identities, devices are left behind NATs and firewalls. Instead of dealing with the complexity of the edge network, users opt to use centralized cloud services, offering usability and high availability.

In this post-Snowden era, users are beginning to question their decision out of fear of censorship and mass-surveillance. Furthermore, a series of highly publicized data breaches and DoS attacks have shed light on the weak guarantees provided by opaque terms of service [8], which are engineered to minimize legal responsibility. Classes of applications such as multiplayer games and video conferencing can benefit from low latency characteristics of direct peer to peer connections whilst others such as local file sync and sharing can benefit from the high bandwidth and scalability. Even in this modern world, users need the ability to establish inter-device connectivity without a full internet connection, for example isolating processing of personal data from the Internet Of Things or connecting between personal devices on the go.

In response to this demand, developers are building new applications for the edge network. They are reimplementing solutions to establishing authenticated identities, consensus and availability in the face of mobile nodes, network partitions and asymmetric channels. Without a clear stack and layers of abstraction, systems fail to provide even the most basic safety guarantees. Protocols are layered on top of each other without formal agreement on the services provided at each layer. Even after this engineering effort by developers, systems still require intricate configuration to deal with the diversity of devices, middleboxes and network environments [6] on the edge network, if they are able to work at all.

Challenges

For this discussion we make the following distinction. *Data requirements* are needs specified by the application, for example a distributed file system may specify that file metadata must be strongly consistent whilst the files themselves need only be eventually consistent. In contrast, *environmental requirements* is the set of network environments that the application needs to operate in. For example, an application might specify that the nodes may be mobile and intermittently connected, however there will always be a cloud node which is highly reliable and publicly addressable but run on untrusted infrastructure.

Our focus on the edge network means we lose the data center assumptions, typical in distributed systems for decades. The environmental requirements now spans:

- **Heterogeneous network topologies** — Middleboxes plague the edge network, network topologies are complex, devices may have asymmetric reachability, there is a wide range of link characteristics and traffic can be treated differently depending on its class.
- **Mobile nodes** — We can no longer rely on IP addresses to identify nodes. Nodes may move between networks and have multiple network interfaces. Intermittent connectivity and network partitions are common.
- **Diverse hardware** — Devices can vary in the constraints of CPU, power supply or memory. Utilizing different networks may come at different costs.
- **New failure models** — We no longer assume homogeneous trust between nodes. Different nodes suffer with different failure models and expected failure patterns.

Developers make crude assumptions about their applications' requirements. The data requirement space is large, it includes some regions that have been proved impossible and others which may prove impossible.

Research Questions

The key research question is how can we provide services such as consistency, accessibility and authentication in the context of edge network distributed systems, this encompasses other questions such as:

- Which areas of the space of data and environmental requirements are covered by existing distributed algorithms, which areas are not yet covered and which areas are provably impossible to cover?

- How can we formally express the assumptions and guarantees of distributed algorithms and their trade-offs, data and environmental requirements such that our engine can resolve them?
- How can we evaluate such systems given the diversity of possible environmental requirements and combinations of data requirements?
- How can we ensure that the distributed algorithms provide the stated guarantees under the assumptions? How can we construct and reason about these algorithms such that they provide stronger guarantees then conventional systems?
- How can we combine the above to provide a stack of protocols which fulfils the data requirements, given the environmental requirements?

Approach

We propose a new common abstraction between applications and networked devices to form personal clouds. Programmers (and ultimately users) formally specify the data and environmental requirements, these requirements span domains in fault tolerance, replication, consistency, caching, accessibility, security levels and confidentiality. From a collection of distributed algorithms, each with their own set of formally specified assumptions and guarantees, an engine will stack the protocols to provide the data requirements in the environmental requirements. From this foundation, we can build new distributed systems including new systems for personal data [4]. We are currently considering building upon a suite of existing tools in this domain including a unikernel operating system [10], TLS implementation [11], a git-style distributed data store [3] and Raft consensus implementation [5].

State of the Art

Sapphire [13] is a programming platform to separate application and deployment logic in cloud and mobile applications. Whilst Sapphire's motivation is similar to ours, it covers a limited space of data requirements and environmental requirementss and doesn't provide any guarantees to applications running on the platform.

The systems community is beginning to design distributed protocols specifically to tolerate the edge network, such as achieving consistency in an environment of heterogeneous trust [12, 7]. But quantifying the environmental requirements of such protocols requires a much richer abstraction than those currently used. For example, it's no longer sufficient to state that a protocol tolerates $\frac{n-1}{2}$ fail-stop faults for a cluster of n nodes. Some authors [1, 9] suggest we can provide stronger guarantees for distributed protocols by changing the basic programming constructs and languages used, this is something we intend to explore further.

1. REFERENCES

[1] Peter Alvaro, Tyson Condie, Neil Conway, Joseph M. Hellerstein, and Russell Sears. I do declare: Consensus in a logic language. *SIGOPS Oper. Syst. Rev.*, 43(4), January 2010.

[2] Marjory S. Blumenthal and David D. Clark. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology*, August 2001.

[3] Thomas Gazagnaire. Irminsule; a branch-consistent distributed library database. *OCaml 2014 Workshop*, 2014.

[4] Hamed Haddadi, Heidi Howard, Amir Chaudhry, Jon Crowcroft, Anil Madhavapeddy, and Richard Mortier. Personal data: Thinking inside the box. *arXiv preprint arXiv:1501.04737*, 2015.

[5] Heidi Howard, Malte Schwarzkopf, Anil Madhavapeddy, and Jon Crowcroft. Raft refloated: Do we have consensus? *SIGOPS Oper. Syst. Rev.*, 49(1), January 2015.

[6] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 246–259. ACM, 2010.

[7] Jed Liu, Michael D. George, K. Vikram, Xin Qi, Lucas Waye, and Andrew C. Myers. Fabric: A platform for secure distributed computation and storage. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, 2009.

[8] Ewa Luger, Stuart Moran, and Tom Rodden. Consent for all: Revealing the hidden complexity of terms and conditions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2687–2696. ACM, 2013.

[9] Anil Madhavapeddy. Combining static model checking with dynamic enforcement using the statecall policy language. In *Proceedings of the 11th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, pages 446–465. Springer-Verlag, 2009.

[10] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. Unikernels: Library operating systems for the cloud. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, 2013.

[11] Hannes Mehnert and David Kaloper Mersinjak. Transport layer security purely in ocaml. In *OCaml Workshop*, 2012.

[12] Isaac C Sheff, Robbert van Renesse, and Andrew C Myers. Distributed protocols and heterogeneous trust: Technical report. *arXiv preprint arXiv:1412.3136*, 2014.

[13] Irene Zhang, Adriana Szekeres, Dana Van Aken, Isaac Ackerman, Steven D Gribble, Arvind Krishnamurthy, and Henry M Levy. Customizable and extensible deployment for mobile/cloud applications. In *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*, pages 97–112. USENIX Association, 2014.