Programmable Peer-to-Peer Systems

Dimitris Vyzovitis

vyzo@media.mit.edu

Viral Communications

MIT Media Laboratory

Overview

- Motivation
- The P2P system design problem
- The case of BitTorrent
- P2P protocol design at the program level
- Peers and VidTorrent

Motivation

- The Problem: How to build robust, extensible and scalable real-time distribution systems.
- Not just to entertainment or best-effort static data distribution. Real-time coverage of events, ad-hoc journalism, access to public cameras . . .
- P2P is the natural architecture: open and scalable.

The Design Problem

- How do we design P2P protocols for real-time distribution?
- Lots of academic proposals, none seen or heard of in the wild...
- The root cause: Broken unimplementable design.
- This work is about P2P protocol design that can be implemented in the real world.

Why P2P System Design is Hard

- Global scale.
- High-level of concurrency.
- Loose timing and unpredictable network conditions.
- Failures and turnover are the norm.
- Malicious and antisocial peer behavior. Defections cannot be detected in general.
- Huge gap from design to implementation.

Design approaches

- Academic approach: Top-down design, aimed at some abstract optimality metric. The result is usually an unimplementable, maximally brittle design.
- Contrast BitTorrent, an organically evolved hack which dominates network traffic.
- What can we learn from BitTorrent? How can we apply to design and implementation of new P2P protocols?

The case of BitTorrent

- BitTorrent is not an academic protocol!
- The protocol evolved as a bottom-up program, by trial and error in the wild.
- The code is the design, written in a high-level language (Python)

BitTorrent: Protocol Attributes

- High level of concurrency, typically interact with 30-40 peers.
- Robust and Adaptive to varying network conditions.
- Tolerate free-riding and antisocial peer behavior.

BitTorrent: Facts

- BitTorrent is by far the most popular P2P protocol in use today, exhibiting unprecedented scalability.
- Measurements have shown that over 50% of Internet edge traffic is BitTorrent.

Why BitTorrent scales

- Design principle. Do not design for optimality. Design for robustness and adaptability.
- No assumptions. Do not rely on extraneous assumptions about peer behavior. React on perceived behavior and performance.
- Tit-for-tat. Embed a strategy that penalizes sluggish behavior directly into the protocol

P2P System Design, BitTorrent style

- Design for approximate correctness. Tolerate an only approximately correct view of the global state. Explicitly take timing and malicious peer behavior into account.
- Protocol design as code. The design should be runnable, not the byproduct of simulation.
- Programming language and environment matters. High level language for algorithms, low level language only for performance critical bit-banging.

The Peers Programming Environment

- Event-driven kernel with continuations.
- High performance, low overhead RPC. Allows raw data interleaving and reverse channel calls.
- Both synchronous and asynchronous RPC interface, maintaining order of events.
- Errors are fi rst class events, propagated across process boundaries.
- Distributed continuations and tail-recursion as concurrency control and protocol implementation primitives.

Peers Implementation Details

- Thread-safety from the ground up. Programmer can freely mix user space concurrency with native threads.
- Kernel written in C++, exported to Python with Boost.Python.
- Interface specification DSL, compiled to C++ for the heavy lifting and Python bindings for algorithm implementation.

VidTorrent

- Live stream single-source overlay multicast.
- BitTorrent-style adaptive behavior.
- Allow distribution of stream components over different trees.
- Tit-for-tat for penalizing malicious peer behavior.
- Applications: Expat-TV, ad-hoc journalism, personal broadcast...
- Implemented with Peers, written almost entirely in Python.

VidTorrent: First generation

- Tree construction and maintenance algorithm. Creates and maintains a good enough tree with low overhead.
- Fast join and recovery, resilience to node failures and varying network conditions.

First generation protocol demo...

VidTorrent: Second generation

- Improved heuristics in tree construction and maintenance. Choking of sink leaves, multiple tree synchronization.
- Cryptographic primitives for peer identifi cation, stream data integrity, and establishment of trust.
- Web-of-trust for computing shared history of peer behavior.
- Tit-for-Tat in peer selection, admission, and tree maintenance strategies.

Code Availability

- Peers codebase is ready for initial public release under the GNU GPL.
- Currently working on documentation.
- The second generation VidTorrent implementation is work in progress. Code will be released in sync with Peers.

Conclusion

- The gap from design to implementation is a fundamental reason for the hardness of P2P system design.
- BitTorrent demonstrates the power bottom-up protocol programming.
- Peers as a hybrid language environment for designing programmable P2P systems.
- Work on VidTorrent, a real-time live P2P multicast protocol based on Peers.