Experiences with Pip: Finding Unexpected Behavior in Distributed Systems

Overview

- Many bugs reflect discrepancies between actual and expected system behavior
  - Structural – incorrect order or placement of processing and/or communication
  - Performance – resource consumption or delays higher or lower than expected
- Pip reports behavior that violates expectations – potential bugs

To use Pip, the programmer writes:
- **Expectations** – external, declarative description of expected system behavior
- **Annotations** – additions to system source code to trace relevant activity

Expectations

- Expectations describe how to recognize and classify paths
- Separate from application source code
- Expectations can be generated automatically
- Concise description of system behavior
- Reading it helps you understand the system, find bugs

SplitStream

- 13 structural bugs found
- 11 found using expectations
- 2 found in the GUI
- 7 were many months old

FAB

- 4 protocol implementations checked
- 1 performance bug found

RanSub

- 2 bugs found
- One performance, one structural
- One found using expectations, one in the GUI

---

Language design

- **Goals**
  - Match every valid path
  - Reject (almost) every invalid path
  - Keep life simple for the programmer

Expectation for the FAB read protocol

```c
validator Read3Others
limit (VOL_CS, 0);
thread Client(*, 1)
  send (Coordinator) limit (SIZE, {=44b});
recv (Coordinator);
thread Coordinator(*, 1)
  recv (Client) limit (SIZE, {=44b});
task ("fabrpc::Read")
  repeat 3 send (Peer);
  repeat 2
    recv (Peer);
  task ("quorumrpc::ReadReply");
future // happens now or later
  recv (Peer);
  task ("quorumrpc::ReadReply");
send (Client);
thread Peer(*, 3)
  recv (Coordinator);
task ("quorumrpc::ReadReq")
send (Coordinator)
```

Structural visualization: FAB Paxos protocol