Fault tolerant TCP state

with Riak Core

Magnus and Luca

Use case

Problem: Some enterprise systems require keeping long-lived TCP connections

Solution: Separate TCP state from application and from the OS

- Enables application (or even OS!) restart without interrupting TCP connection

Problem: Storage of TCP state may be single point of failure

Solution: Replicate it - using Riak Core

Achievements

On TCP/IP handling:

- Design
  - One node per IP at a time
  - Minimal TCP segments handling
    - Received data synchronously passed to rest of application before storing updated state
- Sending raw IP packet

On replicated TCP state:

- Design

Apply a display filter ... <⌘/>

Expression...

No.	Time	Source	Destination	Protocol	Dest port	Src pc
1	0.000000	127.0.0.1	127.0.0.1	IPv4		

```
▶ Frame 1: 40 bytes on wire (320 bits), 40 bytes captured (320 bits)
  Raw packet data
▼ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 20
    Identification: 0x0000 (0)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▼ Header checksum: 0x0000 [validation disabled]
      [Good: False]
      [Bad: False]
    Source: localhost (127.0.0.1)
    Destination: localhost (127.0.0.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

Software used

Riak Core - for replication of data

Tunctl - for sending and receiving raw IP packets (containing TCP segments)

Pkt - codec for IPv4 and TCP