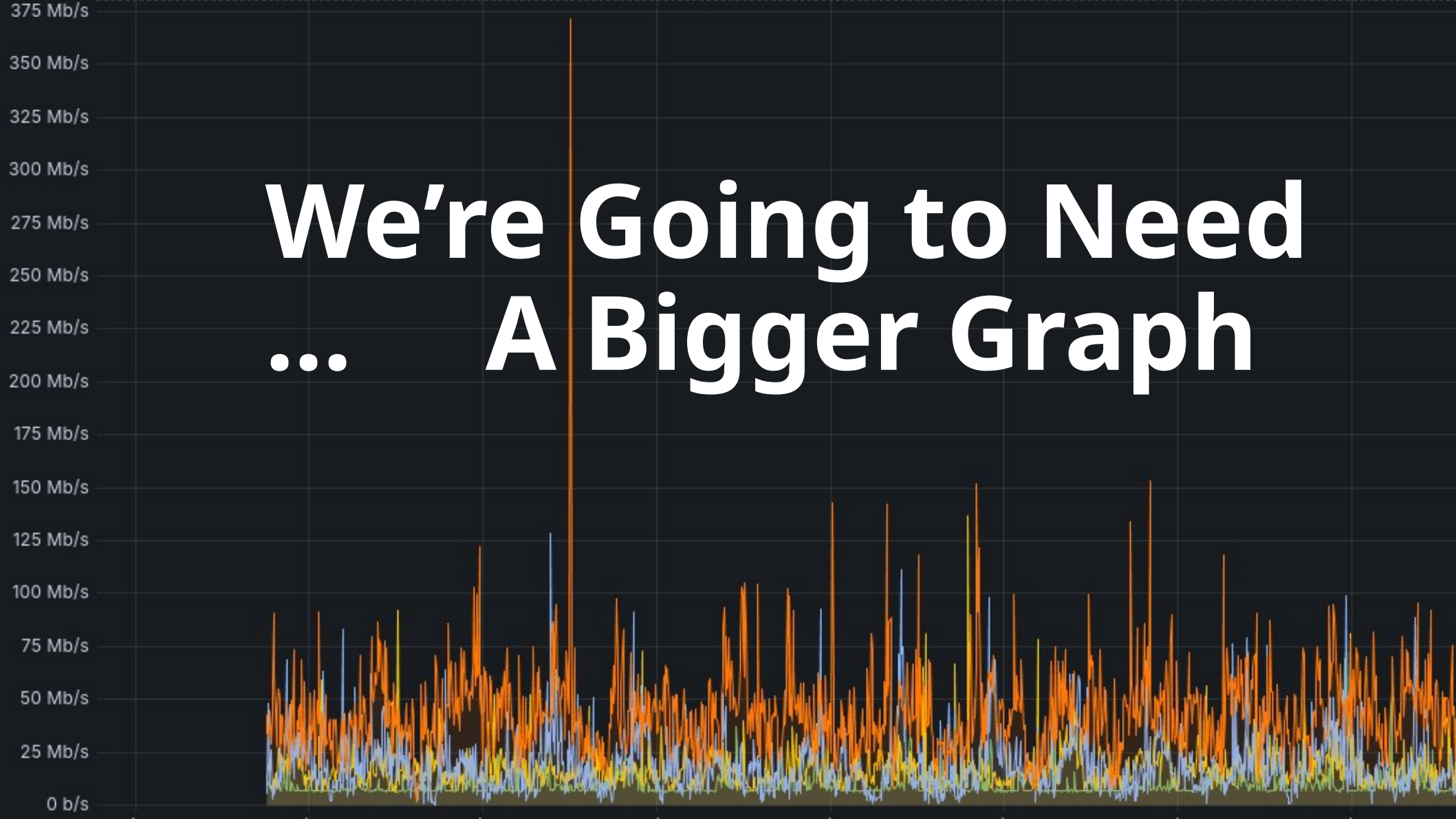
streaming elixir

dch@skunkwerks.at


375 Mb/s
350 Mb/s
325 Mb/s
300 Mb/s
275 Mb/s
250 Mb/s
225 Mb/s
200 Mb/s
175 Mb/s
150 Mb/s
125 Mb/s
100 Mb/s
75 Mb/s
50 Mb/s
25 Mb/s
0 b/s
We're Going to Need
... A Bigger Graph

constraints & deployment

- no downtime allowed
- no reboots allowed
- no info (# connections, data volume)
- lots of live debugging in the wild
- every endpoint is slightly different

basics

env

- ~ 200 servers running FreeBSD, mostly bare metal
- firewall & security related services
- shifting ~ 20TB / month

app

- classic 2 legged tcp proxy
- custom auth & routing: wrote our own

elixir FTW

- 930 LoC, 330 LoT
- OTP27 for process_labels
- thousand_island handles inbound TCP req
- some fancy pattern-matching
- create outbound leg of proxy

interesting bits

- runtime config
- TCP proxy
- TLS sniffing
- Live Debugging
- Future Work

runtime config

- init with env vars
- load into app env
- could use persistent_term
- change everything without restart

```elixir
config :envoy,
  country: System.get_env("ENVOY_COUNTRY", "US") |> String.upcase(),
  proxy: System.get_env("ENVOY_UPSTREAM", "localhost:10000"),
  port: System.get_env("ENVOY_PORT", "8000") |> String.to_integer(),
  ip: System.get_env("ENVOY_IP", "127.0.0.1"),
  acceptors: System.get_env("ENVOY_ACCEPTORS", "5") |> String.to_integer(),
  headers: System.get_env("ENVOY_HEADERS", ""),
  region_key: System.get_env("ENVOY_REGION_KEY", "x-region-key"),
  session_key: System.get_env("ENVOY_SESSION_KEY", "x-session-key")
```

```elixir
  def proxy_ip, do: get_env(:envoy, :proxy_ip)
  def proxy_port, do: get_env(:envoy, :proxy_port)
  def acceptors, do: get_env(:envoy, :acceptors)
  def country, do: get_env(:envoy, :country)
  def server_name, do: get_env(:envoy, :server_name)
  def listen_port, do: get_env(:envoy, :port)
  def session_key, do: get_env(:envoy, :session_key)
  def server_hash, do: get_env(:envoy, :server_hash)
```

tcp proxy

- thousand_island GenServer handler
- 2 connections per session (up & down)
- sniff header to validate & auth
- create outbound gen_tcp/3
- post auth just shuffle the packets

```elixir
@impl ThousandIsland.Handler

def handle_connection(%Socket{socket: socket}, _args) do
  peer = peer_info(socket)
  Logger.info("init < #{peer} from #{inspect(socket)}")
  {:continue, %Envoy{mode: :opened, peer: peer}}
end

# handle proxyv2 protocol if present
def handle_data(frame = @proxyv2_signature <> _rest, socket, envoy = %Envoy{mode: :opened}) do
  Logger.debug("hav2 < raw #{dump(frame)}")
  {:ipv4, src, buffer} = proxyv2_parse(frame)

  peer =
    to_string(:inet.ntoa(src.address)) <>
      ":" <> Integer.to_string(src.port)

  # open upstream connection or die trying
  upstream = connect!(Envoy.proxy_ip(), Envoy.proxy_port(), @opts)
  Logger.info("init > #{peer} for #{src.host} to #{inspect(upstream)}")

  handle_data(buffer, socket, %Envoy{
    envoy
    | client_ip: src.address,
      client_port: src.port,
      host: src.host,
      peer: peer,
      upstream: upstream,
      client_session_id: proxyv2_ip_as_hex(src.address),
      mode: :connected
  })
```

tls sniffing

- most network protocols are TLV

```
<< @type::size(8),
   length::size(16),
   value::binary-size(length),
   rest::binary >>
```

- sometimes recursive & nested
- read RFC5246, RFC8446, RFC9460 very closely

```elixir
@proxyv2_authority 0x02

@doc """
https://www.haproxy.org/download/3.0/doc/proxy-protocol.txt@proxyv2_auth
Section 2.2 0x02 authority TLV
"""
def proxyv2_parse_tlv(""), do: nil

def proxyv2_parse_tlv(
      <<@proxyv2_authority, length::size(16), authority::binary-size(len
    ) do
  Logger.warning("hav2 | found authority #{authority}")
  authority
end

def proxyv2_parse_tlv(<<_type, length::size(16), _value::binary-size(len
  Logger.error("hav2 | unexpected tlv #{dump(rest)}")
  proxyv2_parse_tlv(rest)
end
```

```elixir
def sni_from_client_hello(data) do
  case resp = parse_tls_record(data) do
    {:ok, _} ->
      Logger.debug("tls_record: #{dump(data)}")
      resp

    _ ->
      Logger.debug("parser error " <> dump(data))
      resp
  end
end

def parse_tls_record(
      <<@handshake::size(8), _tls_protocol::size(16), _length::size(16)
    ) do
  parse_client_hello(handshake)
end
```

```elixir
@client_hello 0x01
def parse_client_hello(
      <<@client_hello::size(8), _length::size(24), _tls_version::
        _random::binary-size(32), session_id_length::size(8),
        _session_id::binary-size(session_id_length), ciphers::bina
    ) do
  # Logger.debug("TLS version: #{inspect(tls_version(tls_version)
  # Logger.debug("session id length: #{session_id_length}")
  # Logger.debug("session id: #{Base.encode16(session_id, case: :
  parse_cipher_suite(ciphers)
end
```

```elixir
def parse_cipher_suite(<<length::size(16), suites::binary-size(length
    when byte_size(suites) == length do
  # Logger.debug("cipher_suite: #{dump(suites)}")
  parse_compression_method(compression)
end

@no_compression 0x0100
def parse_compression_method(<<@no_compression::size(16), extensions
  # Logger.debug("compression_method: none")
  parse_extensions(extensions)
end

def parse_compression_method(
      frame = <<length::size(8), _methods::binary-size(length), exte
    )
    when byte_size(frame) == length + 1 do
  # Logger.debug("compression_method: #{dump(frame)}")
  parse_extensions(extensions)
```

live debugging

- dump frames in debug, or on error
- use tracing FTW
- re-use frames as test fixtures

```
13:34:33.948 [info] init < 127.0.0.1:28198 from #Port<0.15>

13:34:33.949 [debug] hav2 < raw
00000000:  0d 0a 0d 0a 00 0d 0a 51 55 49 54 0a 21 11 00 1a     .......QUIT.!...
00000010:  64 40 00 00 64 40 00 00 6e 25 01 bb 02 00 0b 68     d@..d@..n%.....h
00000020:  74 74 70 62 69 6e 2e 6f 72 67 16 03 01 02 00 01     ttpbin.org......
00000030:  00 01 fc 03 03 df dc a1 2a a8 e8 19 99 6a 6a af     ........*....jj.
00000040:  42 2a e9 da 67 ab ef 5f 2a f8 0e 0d b2 51 fb ca     B*..g.._*....Q..
00000050:  0a ec 28 bf 93 20 37 58 21 02 57 0b ff 69 0e 54     ..(...7X!.W..i.T
00000060:  9e 57 36 89 46 4a 95 93 12 82 29 fc 1a 14 1a ee     .W6.FJ....).....
00000070:  7b 26 e7 52 f8 5a 00 3e 13 02 13 03 13 01 c0 2c     {&.R.Z.>.......,
00000080:  c0 30 00 9f cc a9 cc a8 cc aa c0 2b c0 2f 00 9e     .0.........+./..
00000090:  c0 24 c0 28 00 6b c0 23 c0 27 00 67 c0 0a c0 14     .$.(.k.#.'.g....
000000a0:  00 39 c0 09 c0 13 00 33 00 9d 00 9c 00 3d 00 3c     .9.....3.....=.<
000000b0:  00 35 00 2f 00 ff 01 00 01 75 00 00 00 10 00 0e     .5./.....u......
000000c0:  00 00 0b 68 74 74 70 62 69 6e 2e 6f 72 67 00 0b     ...httpbin.org..
000000d0:  00 04 03 00 01 02 00 0a 00 16 00 14 00 1d 00 17     ................
000000e0:  00 1e 00 19 00 18 01 00 01 01 01 02 01 03 01 04     ................
000000f0:  00 10 00 0e 00 0c 02 68 32 08 68 74 74 70 2f 31     .......h2.http/1
00000100:  2e 31 00 16 00 00 00 17 00 00 00 31 00 00 00 0d     .1.........1....
00000110:  00 2a 00 28 04 03 05 03 06 03 08 07 08 08 08 09     .*.(............
```

```
13:34:33.949 [warning] hav2 | found authority httpbin.org

13:34:34.066 [info] init > 100.64.0.0:28197 for httpbin.org to #Port<0.16>

13:34:34.067 [debug] helo < raw #Port<0.15> with data
00000000:  16 03 01 02 00 01 00 01 fc 03 03 df dc a1 2a a8     ..............*.
00000010:  e8 19 99 6a 6a af 42 2a e9 da 67 ab ef 5f 2a f8     ...jj.B*..g.._*.
00000020:  0e 0d b2 51 fb ca 0a ec 28 bf 93 20 37 58 21 02     ...Q....(...7X!.
00000030:  57 0b ff 69 0e 54 9e 57 36 89 46 4a 95 93 12 82     W..i.T.W6.FJ....
00000040:  29 fc 1a 14 1a ee 7b 26 e7 52 f8 5a 00 3e 13 02     ).....{&.R.Z.>..
00000050:  13 03 13 01 c0 2c c0 30 00 9f cc a9 cc a8 cc aa     .....,.0........
00000060:  c0 2b c0 2f 00 9e c0 24 c0 28 00 6b c0 23 c0 27     .+./...$.(.k.#.'
00000070:  00 67 c0 0a c0 14 00 39 c0 09 c0 13 00 33 00 9d     .g.....9.....3..
00000080:  00 9c 00 3d 00 3c 00 35 00 2f 00 ff 01 00 01 75     ...=.<.5./.....u
00000090:  00 00 00 10 00 0e 00 00 0b 68 74 74 70 62 69 6e     .........httpbin
000000a0:  2e 6f 72 67 00 0b 00 04 03 00 01 02 00 0a 00 16     .org............
000000b0:  00 14 00 1d 00 17 00 1e 00 19 00 18 01 00 01 01     ................
000000c0:  01 02 01 03 01 04 00 10 00 0e 00 0c 02 68 32 08     .............h2.
000000d0:  68 74 74 70 2f 31 2e 31 00 16 00 00 00 17 00 00     http/1.1........
000000e0:  00 31 00 00 00 0d 00 2a 00 28 04 03 05 03 06 03     .1.....*.(......
```

```
13:34:34.068 [debug] next_extension data type: 00

13:34:34.068 [debug] next_extension data length: 16

13:34:34.068 [debug] next_extension data frame:
00000000:  00 0e 00 00 0b 68 74 74 70 62 69 6e 2e 6f 72 67     .....htt


13:34:34.068 [debug] next_extension leftovers:
00000000:  00 0b 00 04 03 00 01 02 00 0a 00 16 00 14 00 1d     ........
00000010:  00 17 00 1e 00 19 00 18 01 00 01 01 01 02 01 03     ........
00000020:  01 04 00 10 00 0e 00 0c 02 68 32 08 68 74 74 70     ........
00000030:  2f 31 2e 31 00 16 00 00 00 17 00 00 00 31 00 00     /1.1....
00000040:  00 0d 00 2a 00 28 04 03 05 03 06 03 08 07 08 08     ...*.(..
00000050:  08 09 08 0a 08 0b 08 04 08 05 08 06 04 01 05 01     ........
00000060:  06 01 03 03 03 01 03 02 04 02 05 02 06 02 00 2b     ........
00000070:  00 09 08 03 04 03 03 03 02 03 01 00 2d 00 02 01     ........
00000080:  01 00 33 00 26 00 24 00 1d 00 20 a8 51 c8 12 22     ..3.&.$.
```

live debugging & recon_trace

```elixir
defmodule I do
  def cls, do: IO.puts("\ec")

  def qt(m, f \\ :_) do
    l(m)
    :recon_trace.calls(
      {m, f, :return_trace},
      {1000, 10000},
      pid: :all,
      scope: :local,
      stack: :return
    )
  end
end

import_if_available I
```

https://gist.github.com/dch/e458748e2bcfde038f711ca5b3bd1f90

```
13:42:20.540094 <0.309.0> 'Elixir.Envoy.Proxy':child_spec({[],[]})

13:42:20.545586 <0.309.0> 'Elixir.Envoy.Proxy':child_spec/1 -->
  {id=>'Elixir.Envoy.Proxy',
    restart=>temporary,
    start=>{'Elixir.Envoy.Proxy',start_link,[{[],[]}]}}

13:42:20.545703 <0.308.0> 'Elixir.Envoy.Proxy':start_link({[],[]})

13:42:20.545787 <0.517.0> 'Elixir.Envoy.Proxy':init([])

13:42:20.545831 <0.517.0> 'Elixir.Envoy.Proxy':init/1 --> {ok,{nil,[]}}

13:42:20.545910 <0.308.0> 'Elixir.Envoy.Proxy':start_link/1 --> {ok,<0.517.0>}

13:42:20.545970 <0.517.0> 'Elixir.Envoy.Proxy':handle_info({thousand_island_ready, Port<0.12>,
                    {port => 8000,shutdown_timeout => 15000,
                      num_acceptors => 100,
                      '__struct__' => 'Elixir.ThousandIsland.ServerConfig',
                      transport_options =>
                          [{nodelay,true},
                           {ip,{0,0,0,0}},
                           {backlog,100},
                           {reuseaddr,true}],
                      handler_module => 'Elixir.Envoy.Proxy',
```

cURL is awesome

```
dch@wintermute ~> curl --trace - --head https://httpbin.org/
== Info: Host httpbin.org:443 was resolved.
== Info: IPv6: (none)
== Info: IPv4: 100.64.0.0
== Info:   Trying 100.64.0.0:443...
== Info: ALPN: curl offers h2,http/1.1
=> Send SSL data, 5 bytes (0x5)
0000: 16 03 01 02 00                                      .....
== Info: TLSv1.3 (OUT), TLS handshake, Client hello (1):
=> Send SSL data, 512 bytes (0x200)
0000: 01 00 01 fc 03 03 ef b9 8f 30 91 4d 37 13 8a 25 .........0.M7..%
0010: f8 cb 3f d1 f7 3d 5e 62 36 89 1c 4e ef fa 08 a5 ..?..=^b6..N....
0020: 98 5d 22 97 e0 f8 20 10 2a be 44 24 53 24 72 43 .]"... .*.D$S$rC
0030: 67 30 37 8e 4d a3 53 bf cd 4d 0b 64 49 71 e5 83 g07.M.S..M.dIq..
0040: c1 7b 2e 44 5c 3f 5e 00 3e 13 02 13 03 13 01 c0 .{.D\?^.>.......
0050: 2c c0 30 00 9f cc a9 cc a8 cc aa c0 2b c0 2f 00 ,.0.........+./.
0060: 9e c0 24 c0 28 00 6b c0 23 c0 27 00 67 c0 0a c0 ..$.(.k.#.'.g...
0070: 14 00 39 c0 09 c0 13 00 33 00 9d 00 9c 00 3d 00 ..9.....3.....=.
0080: 3c 00 35 00 2f 00 ff 01 00 01 75 00 00 00 10 00 <.5./.....u.....
0090: 0e 00 00 0b 68 74 74 70 62 69 6e 2e 6f 72 67 00 ....httpbin.org.
```

cURL \o/ also ngrep, tshark

```
dch@wintermute ~> curl --trace - --head https://httpbin.org/
== Info: Host httpbin.org:443 was resolved.
== Info: IPv6: (none)
== Info: IPv4: 100.64.0.0
== Info:   Trying 100.64.0.0:443...
== Info: ALPN: curl offers h2,http/1.1
=> Send SSL data, 5 bytes (0x5)
0000: 16 03 01 02 00                                      .....
== Info: TLSv1.3 (OUT), TLS handshake, Client hello (1):
=> Send SSL data, 512 bytes (0x200)
0000: 01 00 01 fc 03 03 ef b9 8f 30 91 4d 37 13 8a 25 .........0.M7..%
0010: f8 cb 3f d1 f7 3d 5e 62 36 89 1c 4e ef fa 08 a5 ..?..=^b6..N....
0020: 98 5d 22 97 e0 f8 20 10 2a be 44 24 53 24 72 43 .]"... .*.D$S$rC
0030: 67 30 37 8e 4d a3 53 bf cd 4d 0b 64 49 71 e5 83 g07.M.S..M.dIq..
0040: c1 7b 2e 44 5c 3f 5e 00 3e 13 02 13 03 13 01 c0 .{.D\?^.>.......
0050: 2c c0 30 00 9f cc a9 cc a8 cc aa c0 2b c0 2f 00 ,.0.........+./.
0060: 9e c0 24 c0 28 00 6b c0 23 c0 27 00 67 c0 0a c0 ..$.(.k.#.'.g...
0070: 14 00 39 c0 09 c0 13 00 33 00 9d 00 9c 00 3d 00 ..9.....3.....=.
0080: 3c 00 35 00 2f 00 ff 01 00 01 75 00 00 00 10 00 <.5./.....u.....
0090: 0e 00 00 0b 68 74 74 70 62 69 6e 2e 6f 72 67 00 ....httpbin.org.
```

future work

- add hot code loading
- some metrics & telemetry
- some punsch