# HTTP/2 時代の
# コネクションの切り方

2019.9.20

山本和彦

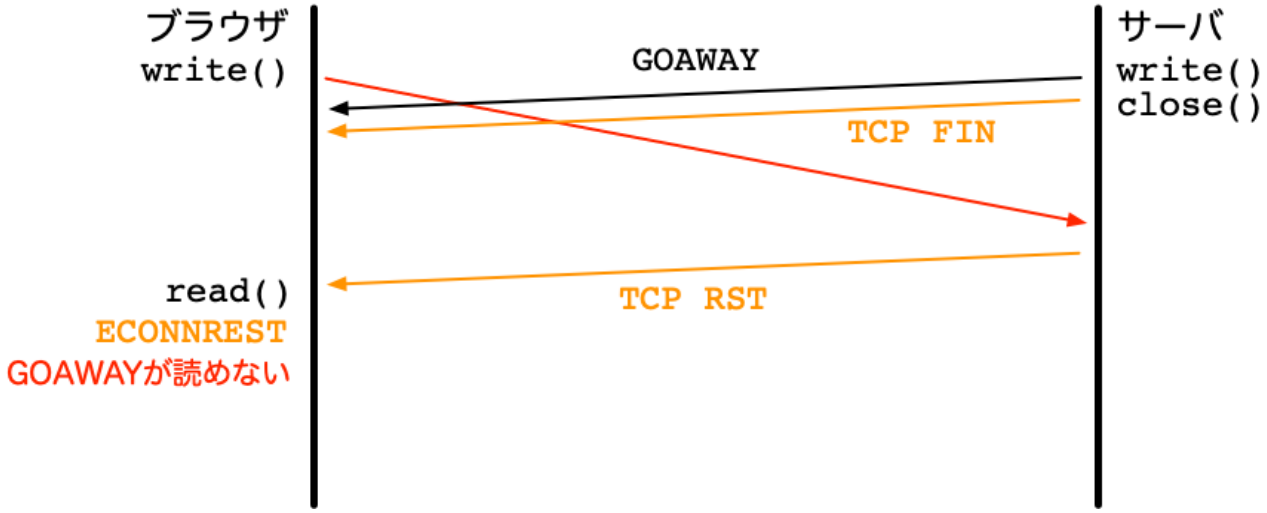
とあるサイトは
nginx で運営している

あるページにはたくさんの
サムネイルがある

nginx で HTTP/2 を有効にしたら
ブラウザがページを正しく
表示しなくなった

# 水面下で起こっていること

- HTTP/2 の GOAWAY が失われる
  - HTTP/2の非同期性が原因
  - 現時点で図を理解する必要はありません



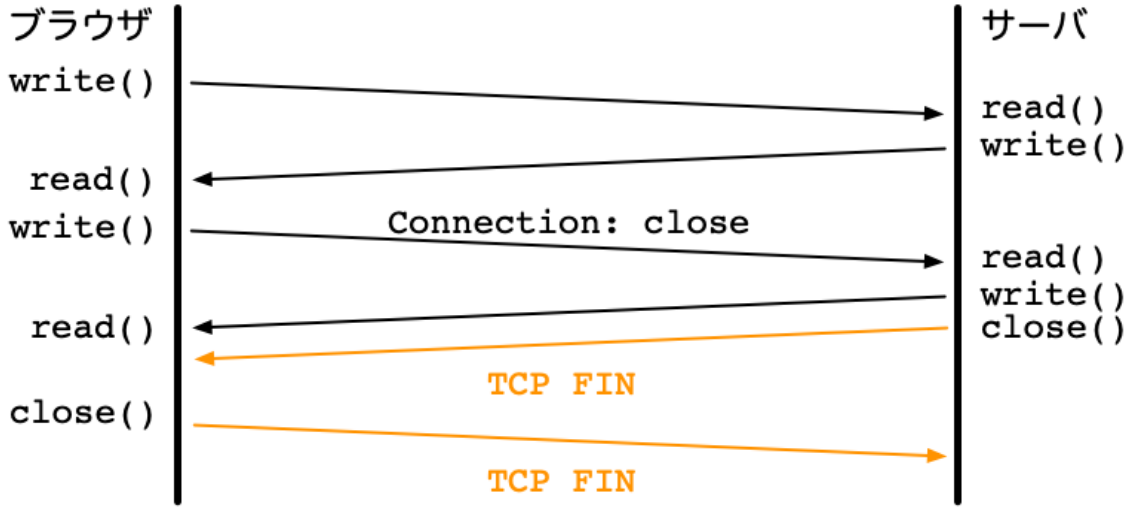
HTTP/1.1 ではなぜうまくいくのか？

HTTP/1.1は同期プロトコル！

`write()` と `read()` の繰り返し
だから異常時に回復できる

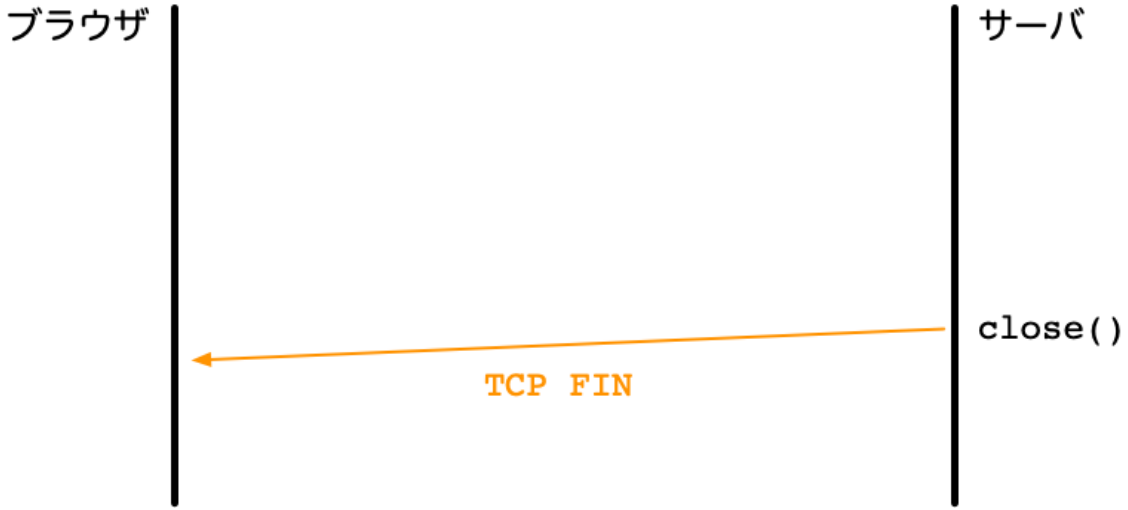
# HTTP/1.1の正常ケース

- デフォルトではコネクションが持続
  - コネクションを切るには、`Connection: close`



# HTTP/1.1の異常ケース

- サーバがコネクションを切る
  - 一定時間使われないコネクションは切る
  - リクエスト数が上限に達するとコネクションを切る
  - 注：HTTP/2 でも同様



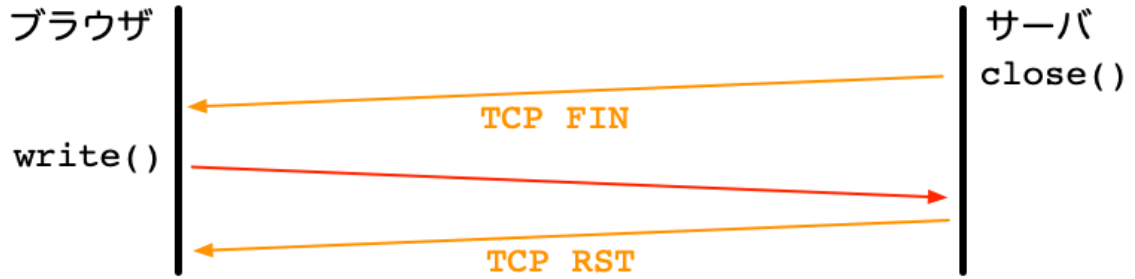
`write()` する前に
ソケットが生きているか
調べられればよい？

ソケットが生きているか調べる方法：

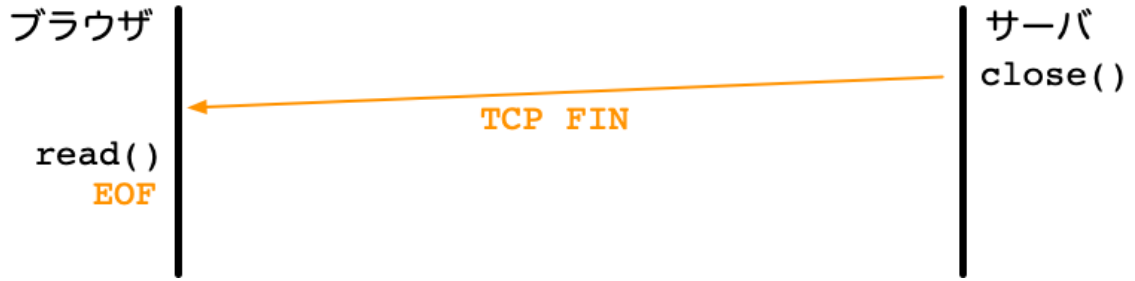
`write()` するか `read()` するしかない！

# TCP FIN を受け取ったソケット

- `write()` は成功する
  - TCP は RST を受け取る
  - アプリケーションには通知なし

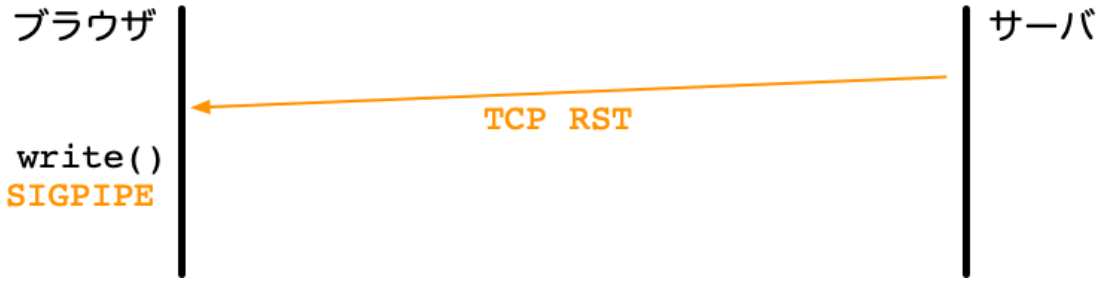



- `read()` は 0 (EOF) を返す



# TCP RST を受け取ったソケット

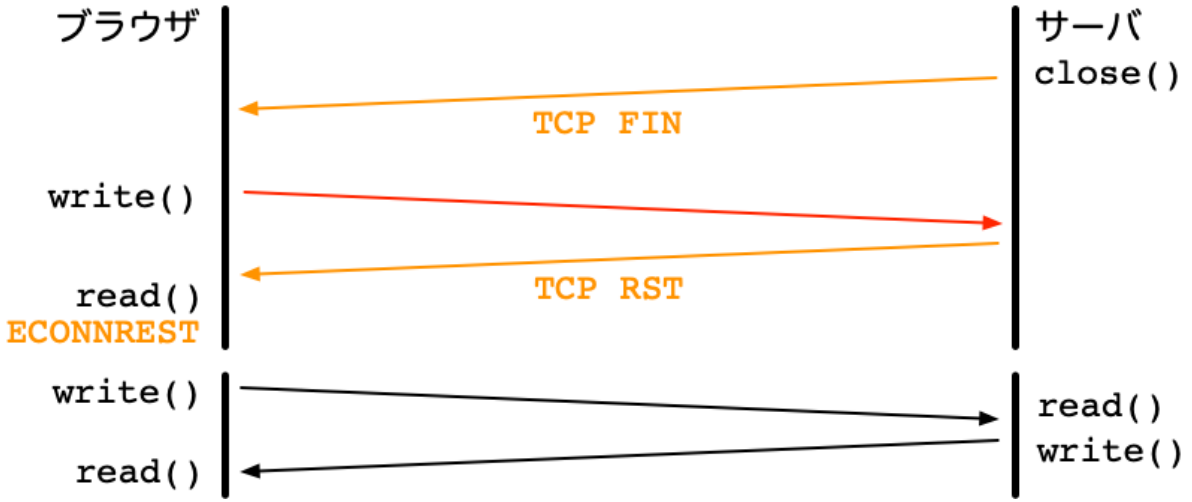
- `write()` は `SIGPIPE` / `EPIPE`




- `read()` は ECONNRESET

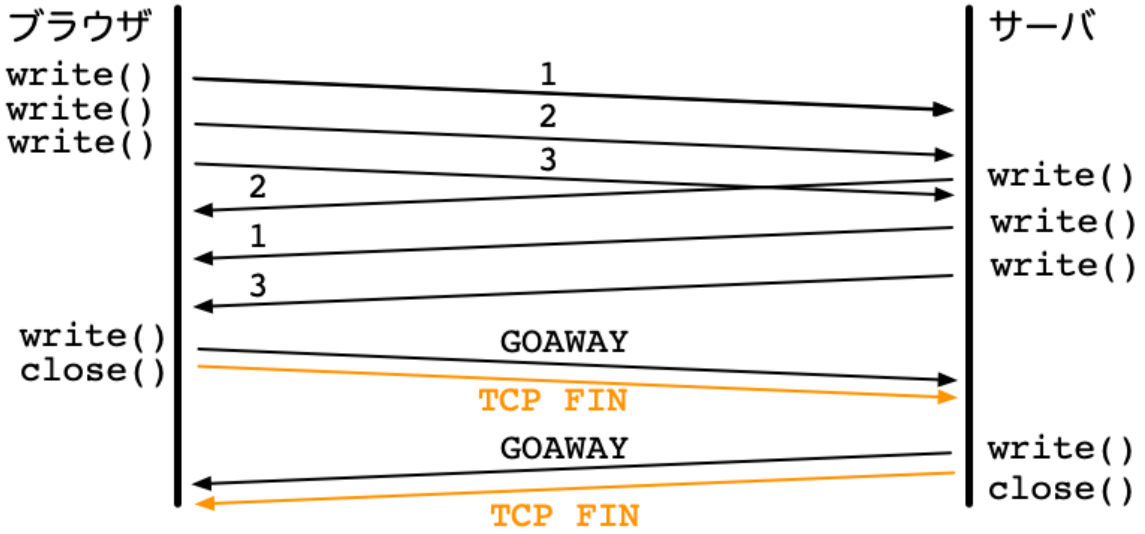
# HTTP/1.1 なら回復できる

- write() と read() の繰り返し
  - レスポンスを読もうとすると ECONNRESET が返ってくる
  - コネクションを張り直し、同じリクエストを送ればよい



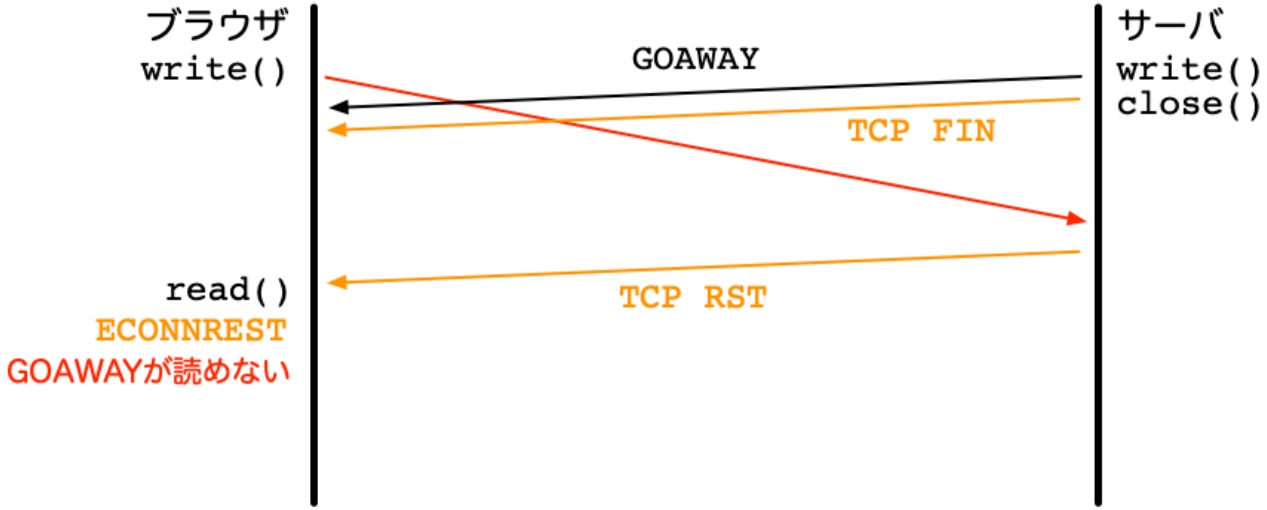
# HTTP/2 の正常ケース

- 1コネクション上に、非同期に複数のリクエスト＆レスポンスが行き交う
  - レスポンスの順番は変わることがある
  - コネクションを切るには GOAWAY を送る



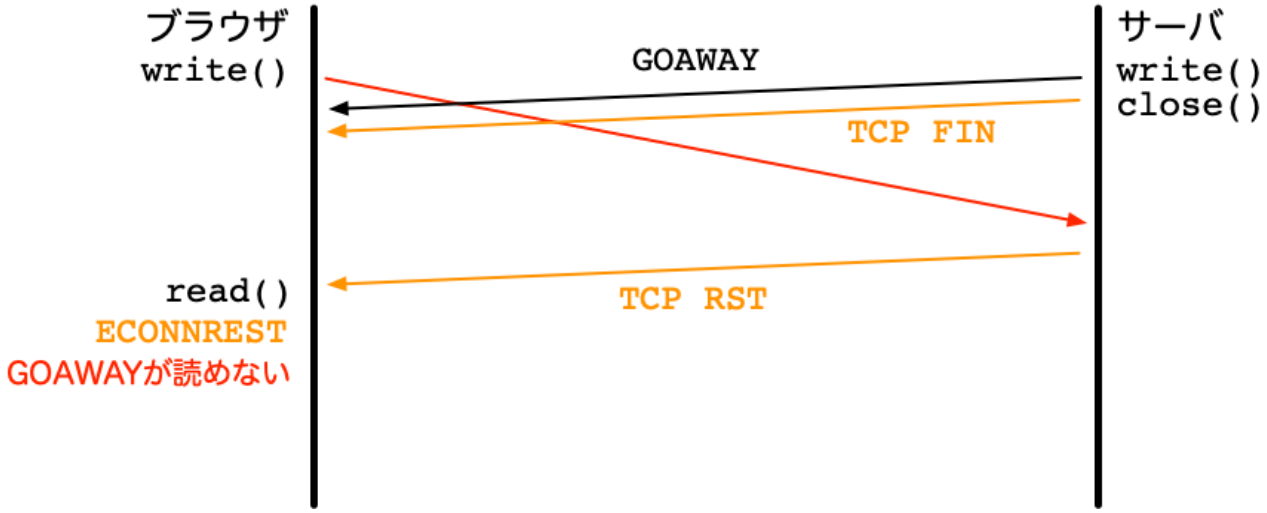
# HTTP/2 の異常ケース (1)

- サーバがコネクションを切る
- ブラウザは非同期に `wirte()` するかも



# HTTP/2 の異常ケース (2)

- TCP RST の受信
  - 読まれるのを待っているデータが破棄される
  - GOAWAY がなくなる
- GOAWAY の喪失
  - GOAWAYにはサーバが最後に処理したストーリムIDが入っている
  - GOAWAYが失われるとブラウザはどこから再開すべきかわからない



graceful close は古くからある問題

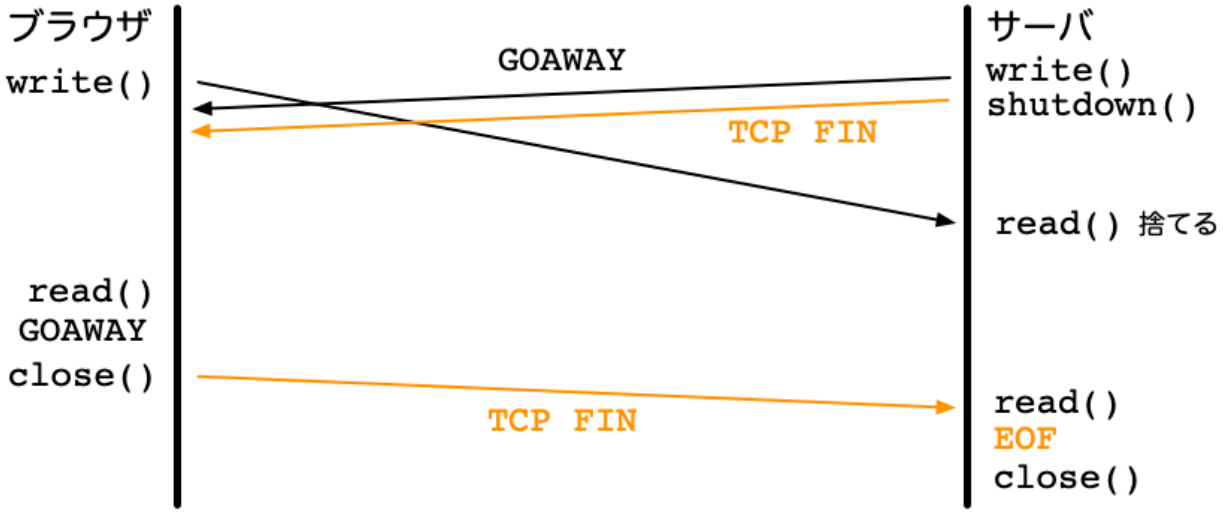
名著を読み直そう！

"Unix Network Programming"
Volume 1: The Sockets Networking API
(3rd Edition)

W. Richard Stevens, et al.

# 解決方法

- close() の代わりに shutdown()
  - TCP FIN を送るが、ソケットの受信側は閉じない
- read() で TCP FIN (EOF)を待つ
  - 一定時間でタイムアウトさせる必要がある



おまけ

# Haskell の network ライブラリでの実装 (1)

1) timeout 関数を使う
   - 利点：実装が容易
   - 欠点：余分な軽量スレッドを消費する

2) `SO_RCVTIMEO` を使う
   - 利点：なし
   - 欠点：blockingソケットにしか利用できず GHC と相性が悪い
     余分なネイティブスレッドを消費する

# Haskell の network ライブラリでの実装 (2)

3) threadDelay を使う
- 利点：IOマネージャがなくても利用できる
- 欠点：タイムアウトの時間を正確にできない
- Windows と non-threaded RTS で採用

4) IOマネージャのコールバックを使う
- 利点：オーバヘッドがなく、タイムアウトの時間が正確
- 欠点：IOマネージャがないと利用できない
- threaded RTS で採用