DNSフルリゾルバ bowline を支える検証技術について

IIJ
Internet Initiative Japan

株式会社 インターネットイニシアティブ
技術研究所　日比野 啓

Ongoing Innovation

自己紹介

- 日比野 啓 / @khibino
- 以前はISPでRadius認証サーバの開発、検証、導入
  2022年から DNSフルリゾルバ bowlineの研究開発実装

本日の話

- DNSのVirtual Circuit
- bowlineのセッション状態管理
  - 軽量スレッド
  - STM(Software Transactional Memory)
- bowlineのセッション状態管理の検証

DNSのVirtual Circuit

DNS の virtual circuit

- DNS over TCP53, DNS over TLS, DNS over HTTP, DNS over QUIC のようなセッションのあるトランスポート

- 1つのセッション 内 で 複 数 の DNSリクエストを送信してよい
- レスポンスの 順 序 がリクエストの 順 序 と異なってよい

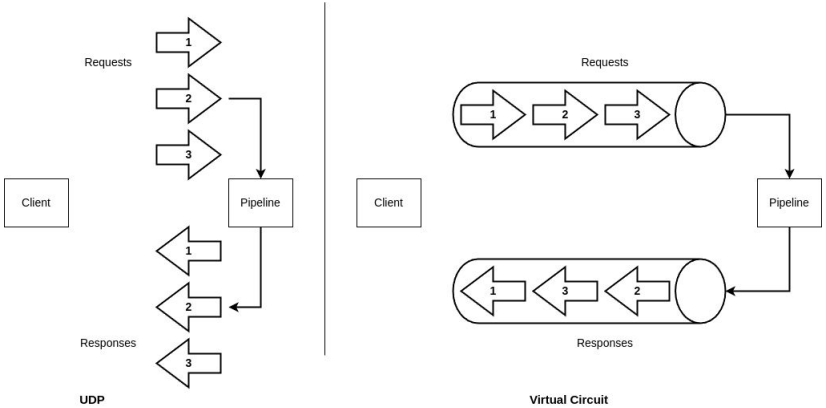

図: UDP と Virtual Circuit

DNSのVirtual Circuit

bowlineのパイプラインでの Virtual Circuit

軽量スレッドとキューを利用して
パイプライン処理を構成する

- **1つのセッション内で複数の DNSリクエストを送信してよい**
- **レスポンスの順序がリクエストの順序と異なってよい**

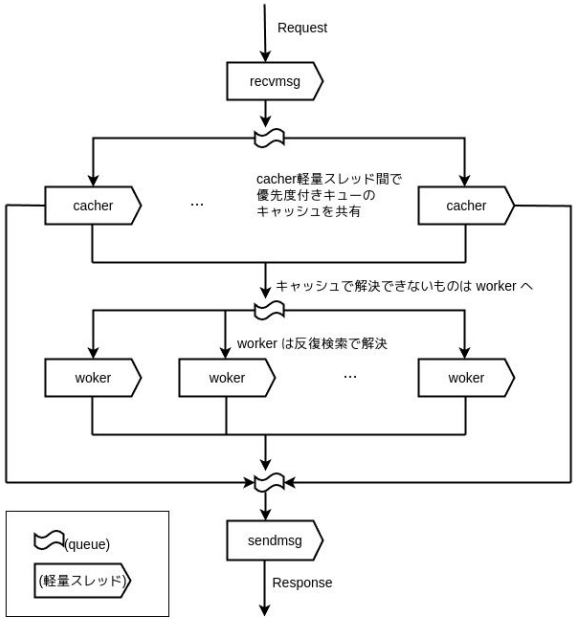

図: 軽量スレッドによるフルリゾルバの構成

DNSのVirtual Circuit

bowlineのパイプラインでの Virtual Circuit(VC)

軽量スレッドとキューを利用してパイプライン処理を構成する

- 1つのセッション内で複数の DNSリクエストを送信してよい
- レスポンスの順序がリクエストの順序と異なってよい
- セッションごとに、リクエスト入力ループとレスポンス出力ループの軽量スレッドを起動する

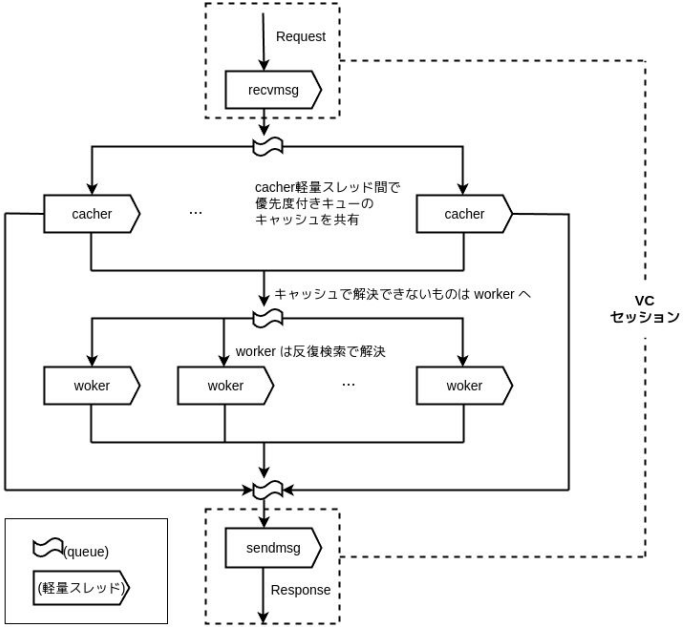

図: フルリゾルバパイプライン内のVCセッション

bowlineのセッション状態管理

Virtual Circuitのセッションと状態管理

- 1つのセッション内で複数の DNS リクエストを送信してよい
- レスポンスの 順 序 がリクエストの 順 序 と異なってよい
- セッションごとに、
  リクエスト 入 力 ループと
  レスポンス 出 力 ループの
  軽量スレッドを起動する

→ セッションを状態管理して、終了条件を判定したい

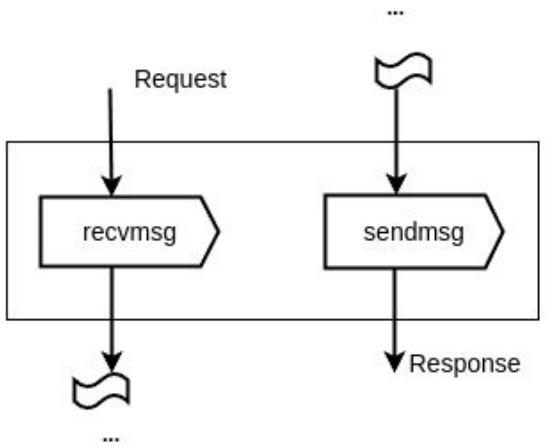

図: セッションごとに起動される軽量スレッド

bowlineのセッション状態管理

セッションに対するイベント列の例

- 1つのセッション内で複数の　DNS リクエストを送信してよい
- レスポンスの順序がリクエストの順序と異なってよい

- イベント
  - Rn: n 番のリクエストが入力可能
  - Sn: n 番のレスポンスが出力可能
  - Eof: End of File
  - Timeout: タイムアウト

イベント列の例 1:

R1, R2, S2, R3, Eof, S1, S3

イベント列の例 2:

R1, R2, S2, Timeout, R3, S1

bowlineのセッション状態管理

STM による入力判定

STM(Software Transactional Memory) による入力可能判定

- **リクエストが 入 力 可 能 あるいはタイムアウトによる終了を待つ**
- **タイムアウトかどうかを返す**

```
waitVcInput =
  atomically
    (do  -- atomically 内が不可分に成立するまで待つ
      timeout <- readTVar vcTimeout  -- タイムアウトフラグ読み出し
      unless timeout waitInput
          -- タイムアウトしていないときは、入力が来るまでやりなおす(待つ)
      return timeout                 -- タイムアウトフラグを結果とする
    )
```

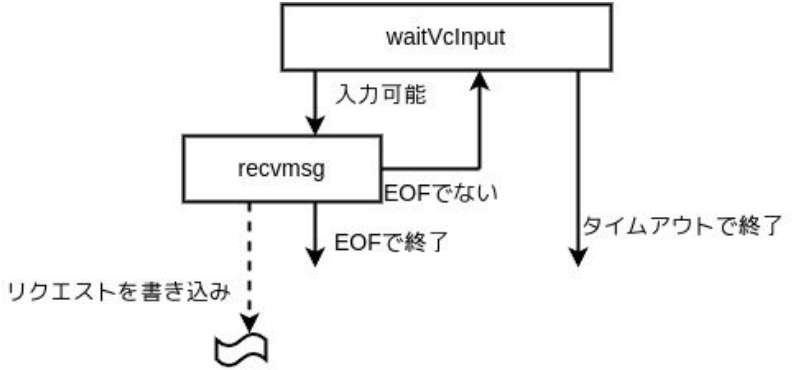

図: 軽量スレッド: リクエスト入力ループ

bowlineのセッション状態管理

STM による出力判定

STM による出力可能判定

- **レスポンスが出力可能あるいは終了を待つ**
- **終了かどうかを返す**

```
waitVcOutput =
  atomically
    (do  -- atomically 内が不可分に成立するまで待つ
      eof <- readTVar vcEof           -- EOF フラグ読み出し
      timeout <- readTVar vcTimeout   -- タイムアウトフラグ読み出し
      avail <- vcRespAvail            -- レスポンスが生成済みかどうか
      if eof || timeout
        then
          (if avail
            then return False         -- レスポンスが生成済みなら終了しない(次の処理は送信)
            else
              (do pendings <- readTVar vcPendings    -- 処理中の集合
                  unless (Set.null pendings) retry   -- 処理中が無くなるまでやりなおす(待つ)
                  return True                        -- 処理中が無いので終了する
              )
          )
        else
          (do unless avail retry      -- レスポンスが生成されていなければやりなおす
              return False            -- レスポンスが生成済みなら終了しない(次の処理は送信)
    )
```

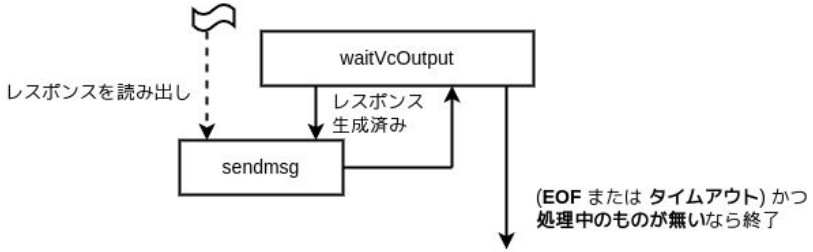

図: 軽量スレッド: レスポンス出力ループ

bowlineのセッション状態管理の検証

ランダムテストによる状態管理の検証

イベントを起こす mockとパイプラインの mockを使ってテスト

- イベント
  - Rn: n 番のリクエストが入力可能
  - Sn: n 番のレスポンスが出力可能
    - n
      (イベント列を生成しやすいように条件を加えている)
  - EOF: End-of-File
  - Timeout: タイムアウト

- イベント列の制約
  - Rn は Sn よりも前
  - EOF は全ての Rn よりも後

番のリクエストが入力済みのときのみ

イベント列の制約のもとでのランダムイベント列を生成

bowlineのセッション状態管理の検証

デモ

ランダムイベント列に対する終了状態と処理結果

まとめ / 参照

- フルリゾルバで Virtual Circuit のセッションを扱うには状態管理が必要
- bowlineでは軽量スレッドと STMを利用してセッション状態管理を行なう
- ランダムテストでセッション状態管理の検証を行なった

- リポジトリ : https://github.com/kazu-yamamoto/dnsext/
- ランダムテストの実装 : https://github.com/kazu-yamamoto/dnsext/blob/main/dnsext-iterative/test/SessionPropSpec.hs

ありがとうございました