株式会社ミクシィ

L1～L3 触ってる人たちに知っておいて欲しい L4～L7の技術

2017/01/20

XFLAG STUDIO
吉野 純平

XFLAG

今日の主題

コンテンツやアプリレイヤの人が
何を見ているかの共有
↓
ネットワーク等を運用している時の
トラブル解析に役に立てば嬉しい

お題目

- HTTPのリクエストヘッダ、レスポンスヘッダーのいろいろ
- アプリがTLS通信している時の通信先ホスト名の確認方法

- トンネル技術利用時のECNビットの挙動

- パケット再送に関する話とその新しい発見策

（tipsなので目新しいとかそういうのは少ないです）

HTTPのヘッダー

ヘッダーを見れば大体わかる

- **HTTP 1.1周りが RFC7230**
  - 関連するものがRFC7231-7239に連続してある
  - （目新しいものは少ないけど RFC2616他が更新された）
- **A client sends an HTTP request to a server in the form of a request message, beginning with a request-line that includes a method, URI, and protocol version (Section 3.1.1), followed by header fields containing request modifiers, client information, and representation metadata (Section 3.2), an empty line to indicate the end of the header section, and finally a message body containing the payload body (if any, Section 3.3).**
  - RFC7230より引用
- **コンテンツの中身はアプリによるけど、ヘッダーをみればいろいろわかる**

よく見るもの

Host

HTTP/1.1では必須

接続先のホスト名を送る。IPで複数ドメインをカバー

X-Forwarded-For

Proxy等する際に接続元情報等を伝える

RFC7239で”Forwarded”として定義しなおされた

併用可能なのでまだまだ生き続けそう

地味にハマったり忘れたりするもの

Range (RFC7233)

- コンテンツの一部をリクエストする
- 無効な場合があるので注意

Vary

- レスポンスヘッダー: どのリクエストヘッダーでキャッシュを別けるか
- 例 "Vary: User-Agent“とするとブラウザごとにキャッシュされる

Date

- レスポンスヘッダー: コンテンツにアクセスされた時間
- これもキャッシュされる可能性があるので注意が必要

何か変だと言われたらまずはヘッダーを見る！！

```
$ curl -v http://www.monster-strike.com/index.html
> GET /index.html HTTP/1.1
> Host: www.monster-strike.com
> User-Agent: curl/7.43.0
> Accept: */*
>
```

何か変だと言われたらまずはヘッダーを見る！！

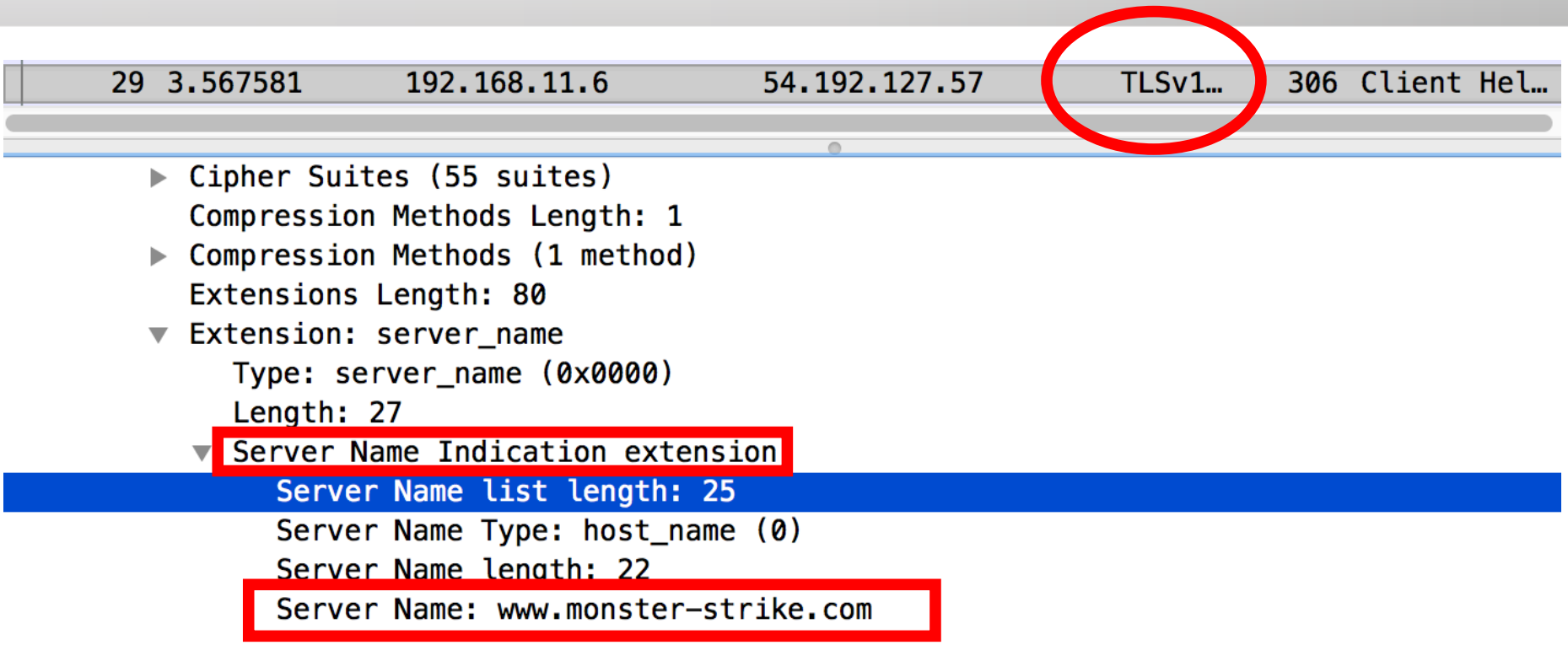
```
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Server: nginx
< Date: Thu, 19 Jan 2017 09:13:37 GMT
< Vary: Accept-Encoding
```

TLSでも接続先ホスト名がわかる

なんで知りたいんだっけ？

- **デバッグする時等にTLSセッションができる**
- **IPレンジをみればどこのAS(多くはクラウド）かわかる**
- **どんな通信先か確認したい**
  - 自社のAPIエンドポイントなのか
  - アプリになんらかのSDKを入れたことによるものなのか

Client Helloの中のSNIの情報を見よう！！



TLSでも暗号化されてない情報がある

トンネルとECN

ECNのサポート状況の変化の見込み

- JANOG LT Nightで土屋さんが発表
  - 「ECN(Explicit Congestion Notification)のあるネットワーク」
- Linuxのデフォルト実装が気になる
  - Version 4.1より前　ECNが有効な接続を受けた場合にECNを有効にして応答
  - Version 4.1以降　自分からもECNを有効で接続
- Androidも7.0から4.4.1ベースになる？
  - https://android.googlesource.com/platform/external/kernel-headers/+log/android-7.0.0_r21/original/uapi/linux/version.h
- ECN周りでハマったことを共有

特定パターンではdropする

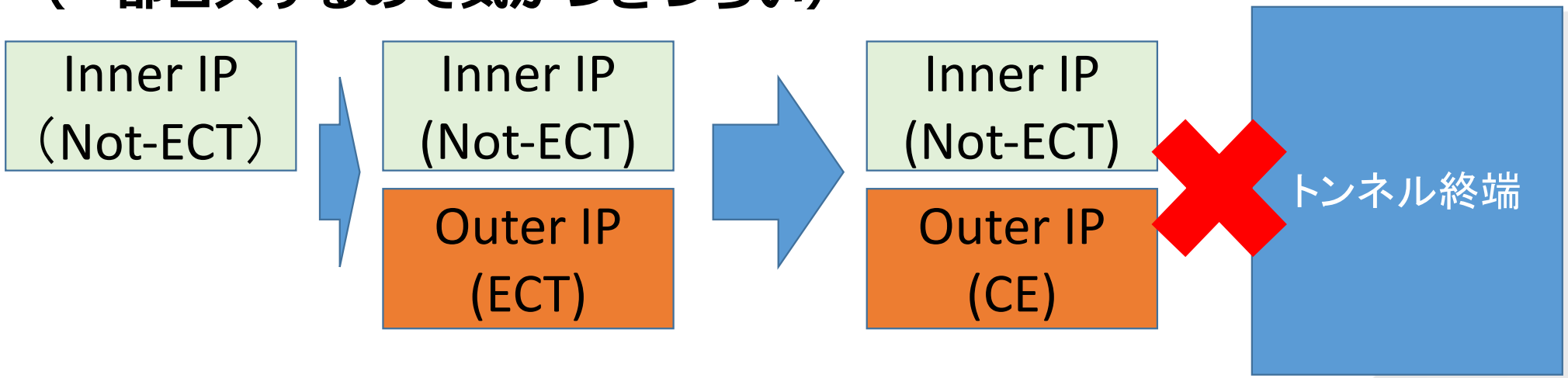
トンネル利用とECNを組み合わせた時に当たったもの

innerがNot-ECTで、outerがCE状態だとdrop

(RFC6040 Tunnelling of Explicit Congestion Notification 参照)

Linuxでもすでに IPIPトンネル等で上記のコードが動いている

（一部ロスするので気がつきづらい）



パケット再送とその発見

パケロスのインパクトと発見手段

パケロスは不幸の連鎖の始まり、検知して運用したい

- **アプリケーションサーバのプロセスが滞留**
- **バックエンドの解放が遅れる場合もある**
- **ユーザ体感の悪化**

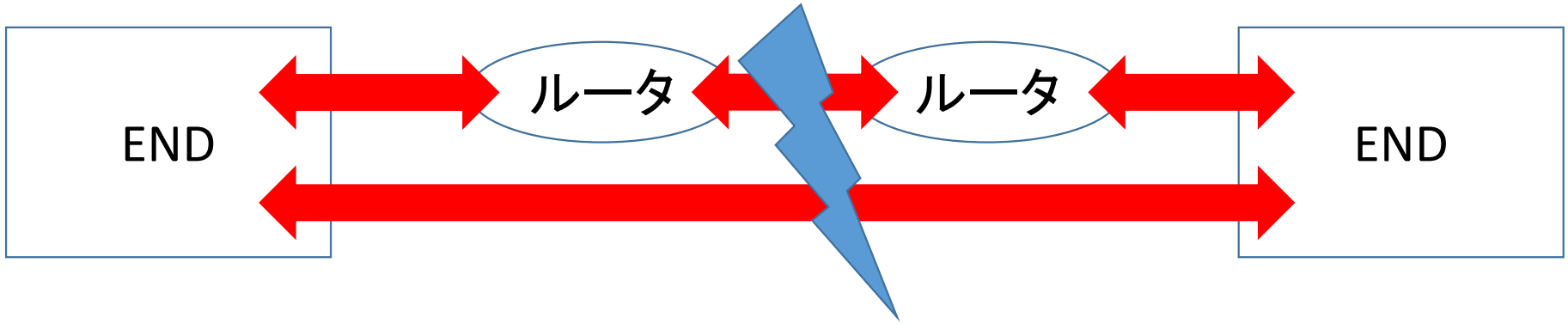
パケットロスを見つける方法として

サーバのtcpの再送処理を利用する手段の共有

（※キャプチャされたファイルを解析するのではない）

パケロス発見方法の整理：中継ノードかエンドか

中継ノードのerrorやdiscardのカウントアップを探す
　　自分の管理範囲内で閉じればできる
End to Endで再送を見つける
　　Endノードなら問題の宛先がわかる
　　ただどこで起きたかは特定できない



サーバでの旧来からある再送確認方法

SNMPで tcpRetransSegs .1.3.6.1.2.1.6.12
にてカウンターが取れる
デメリット：誰との間でretransしているかわからない
クラウド含め、全サーバで見ています

End to Endで見つける

SSコマンド

・ss -tni等するとパケロスしているセッションを発見可能

カーネル内での再送処理をフック

・発生した瞬間に記録可能

・カーネル書き換え＝＞SystemTAP => eBPFの変遷

ssコマンドによる情報取得

kernel内から情報を取り出して表示する

$ ss –nti

State Recv-Q Send-Q Local Address:Port
Peer Address:PortESTAB 0 0 X.X.X.X:XXXX
X.X.X.X:XXXXX

cubic wscale:5,7 rto:208 rtt:4.5/0.759 ato:40 mss:1368
cwnd:10 ssthresh:16 bytes_acked:5109
bytes_received:4149 segs_out:45 segs_in:60 send
24.3Mbps lastsnd:40 lastrcv:48 lastack:36 pacing_rate
29.2Mbps rcv_rtt:8 rcv_space:28160

再送も見つけられる

SYN-SENT 0 1
x.x.x.x:XXXXX
X.X.X.X:80

cubic rto:8000 backoff:3 mss:524
cwnd:1 ssthresh:7 segs_out:4
lastsnd:561476 lastrcv:561476
lastack:561476 unacked:1 retrans:1/3
lost:1

発展系

- ss –tni -Eとすると終了したセッションを流せる
  - 自分でコードを書けば以下のセッションのみを書き出すことも可能
    - 再送したセッション
    - RTTが大きなセッション

カーネルで再送する処理をフックする

パケット再送の関数が呼ばれた時に、該当の通信の情報を取る

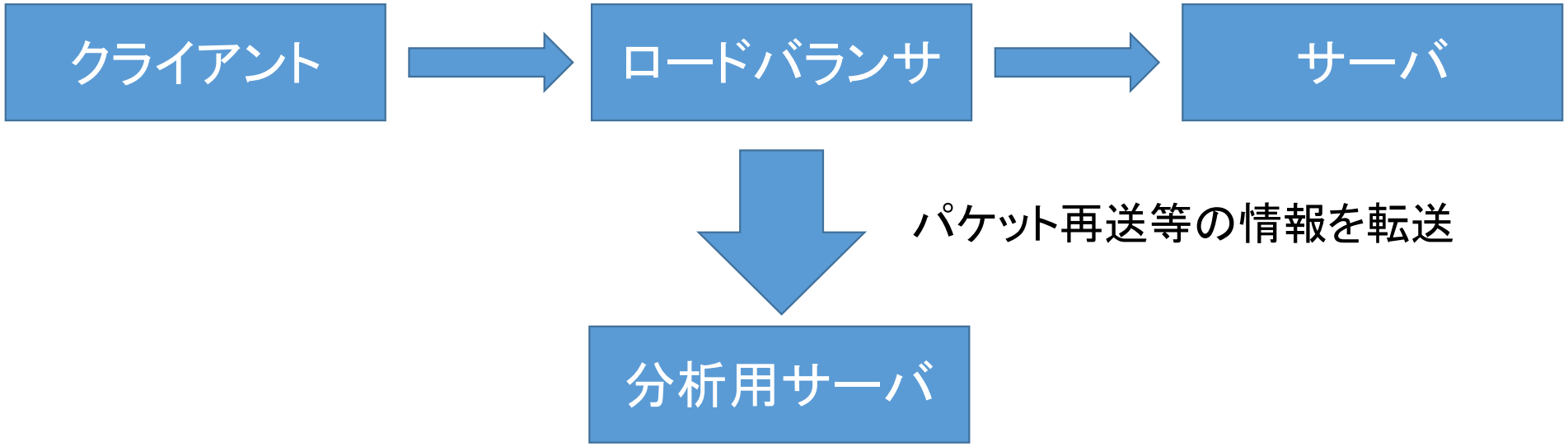
iovisor便利です！

https://github.com/iovisor/bcc/blob/master/tools/tcpretrans_example.txt

（systemtapも不要なのでdebuginfoとかのビルドがいらない！）
（ダンプパケットをファイルに書いてから解析がいらない）
（再送した度に記録されるが故にうるさいかも）

L7アプライアンス機器への要望

L7のロードバランサでコネクション終端する機器
サーバと同等の情報が取れると嬉しいです。
実装予定はありますか？

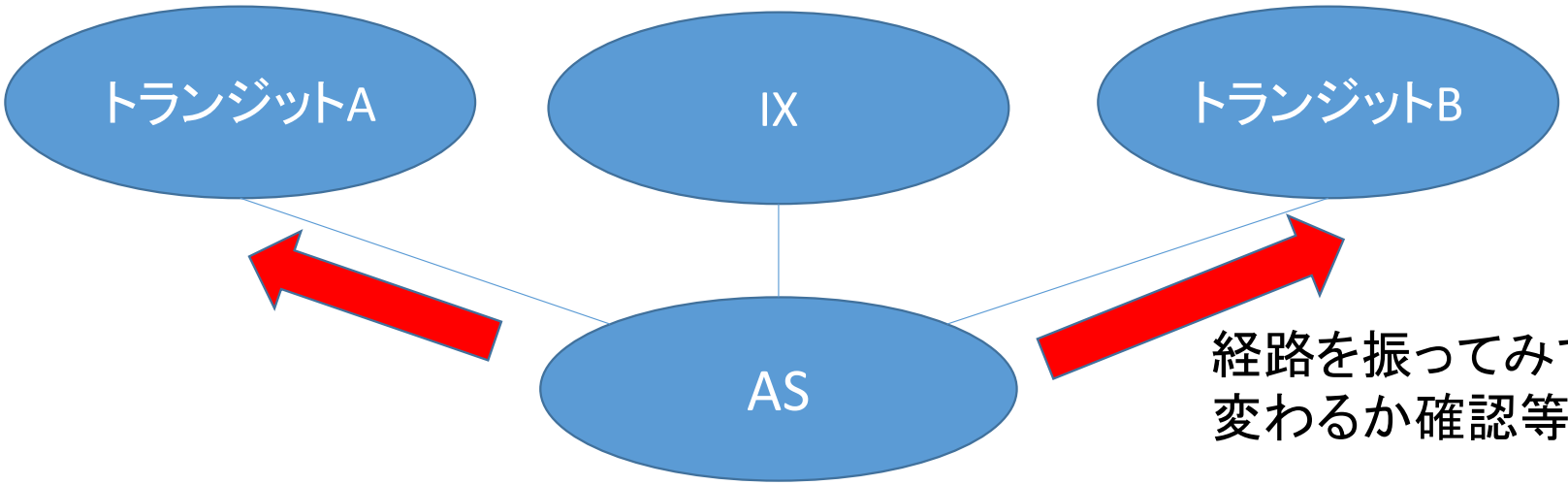


インターネットと再送の発見

いかに再送している場所を見つけるかが肝

経路情報とフロー情報と再送情報があれば戦えるかもしれない

- out方向は自分で再送状況を見えてトライしやすい
- in方向はどうしようもないので、インパクトあるなら個別交渉



議論

・理想はL1-L3とL4-L7の世界とのコラボレーション
    ・統計情報化して、お渡しするといいことありますかね
    ・例：「このAS-PATH、品質あやしくないですか？」
・コンテンツやさんで同じようなことしてます？


XFLAG
B. B. Q.