

YAMAHAルータの死活監視を AWSでやって見えたこと

株式会社ハウインターナショナル 岩男皓一郎 @JANOG 41

1枚10秒ペースなので
巻きで行きます

配布資料としての役割も兼ねているので一部スライドは飛ばします



スピーカーについて

福岡から来ました名強の中小企業

ひとり情シスでネットワークエンジニア兼

一応ネットワークが専門のIT

xrdpというOSSの

IPv6対応やなどを中心に業務とプライベートで開発

自社サービスのほぼ全てをAWSで提供しているNWについては

ほぼユーザ側です

身の上話より本題の話
あとで懇親会で！

当初の目的

弊社の対外回線の死活監視

中から外への疎通を監視してもダウン時に通知する手段がない

いまさら社内に監視・通知サーバを置いてお守りはしたくない

ダウンしたらモバイル回線でSMS通知できると冗長性・利便性の面で最高！

フルマネージドなクラウドサービスでやりたい！



皆様、覚えていますか...？





読売新聞より引用 (2016年11月8日午前7時52分、福岡市博多区で) = 山田伸彦撮影

博多駅前陥没事故

NTT西・QTnetのファイバもダメージを受ける → その節は大変お疲れ様でした

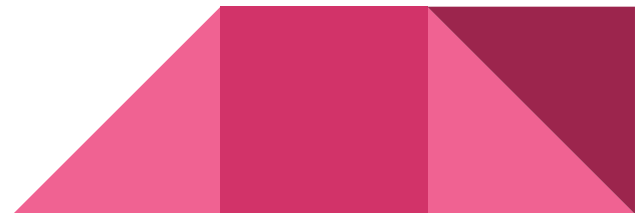
アクセス回線を冗長化していても、災害により両方が使えなくなることがある

モバイル回線を併用するという手もあるが...

そもそも中小企業ではアクセス回線の冗長化までコストをかけられない

マルチホーミングなんて論外！ 地方はアクセス回線の選択肢も少ない

フレッツと電力系でシングルホーム×2 がやっと？



大事なことから

フルマネージドなクラウドサービス
で死活監視したい！



監視の対象とするのは

中小企業御用達のYAMAHAルータ

RTX1210, RTX830



Amazon Route 53 Health Check

AWS外のエンドポイント1つにつき \$0.75/month

レイテンシ計測のオプションを追加すると +\$2.00/mo



日本の Route 53 は岡山駅付近から鳥取へ
JANOGの帰りに立ち寄ってみてください



Amazon Route 53 Health Check

Monitor an endpoint

Multiple Route 53 health checkers will try to establish a TCP connection with the following resou

[Learn more](#)

Specify endpoint by ☒ IP address ☐ Domain name

Protocol	HTTP	i
IP address *	HTTP	i
Host name	HTTPS	i
	TCP	
Port *	80	i
Path	/ images	i



NO ICMP!!!



困った



YAMAHAルータにこんな機能がある

Lua スクリプト機能

- [概要](#)
- [対応機種とファームウェアリビジョン](#)
- [Lua スクリプト機能バージョンの変更履歴](#)
- [用語の定義](#)
- [詳細](#)
- [注意事項](#)
- [コマンド](#)
- [設定例](#)
- [SYSLOG メッセージ一覧](#)
- [参考情報](#)

Luaスクリプト機能

YAMAHAルータ*1 はLuaスクリプトでプログラマブル

USBポートにバーコードリーダを接続して読み取ったりすることもできる *2

これは使える！

*1 概ね2008年11月発売のRTX1200以降の機種

*2 <https://www.slideshare.net/nvsofts/rtx-49061065>



ICMPがないなら
TCPでやればいいじゃない



Luaスクリプトで
TCPサーバを実装して
ルータ上で動かす



できた

ソースコード

<http://bit.ly/YamahaLuaTCPServer>

ライセンスはYAMAHA提供のLuaサンプル
コードのライセンスに準じます

そのライセンスがわからない＼(^o^)/

 health_check_server.lua

```
1  tcp = rt.socket.tcp()
2  tcp:setoption("reuseaddr", true)
3  res, err = tcp:bind("0.0.0.0", 3800)
4
5  if not res and err then
6      print(err)
7      os.exit(1)
8  end
9
10 res, err = tcp:listen()
11 if not res and err then
12     print(err)
13     os.exit(1)
14 end
15
16 while 1 do
17     control = assert(tcp:accept())
18     control:settimeout(10)
19     sent, err = control:send("I'm alive.\n")
20     control:close()
21     rt.sleep(1)
22 end
```

2017年7月13日 運用開始



実際に運用してみて



AWSのマネジメントコンソールから
レイテンシのログを見ることができる



AWS CloudWatch

データの保存期間は最大15ヶ月

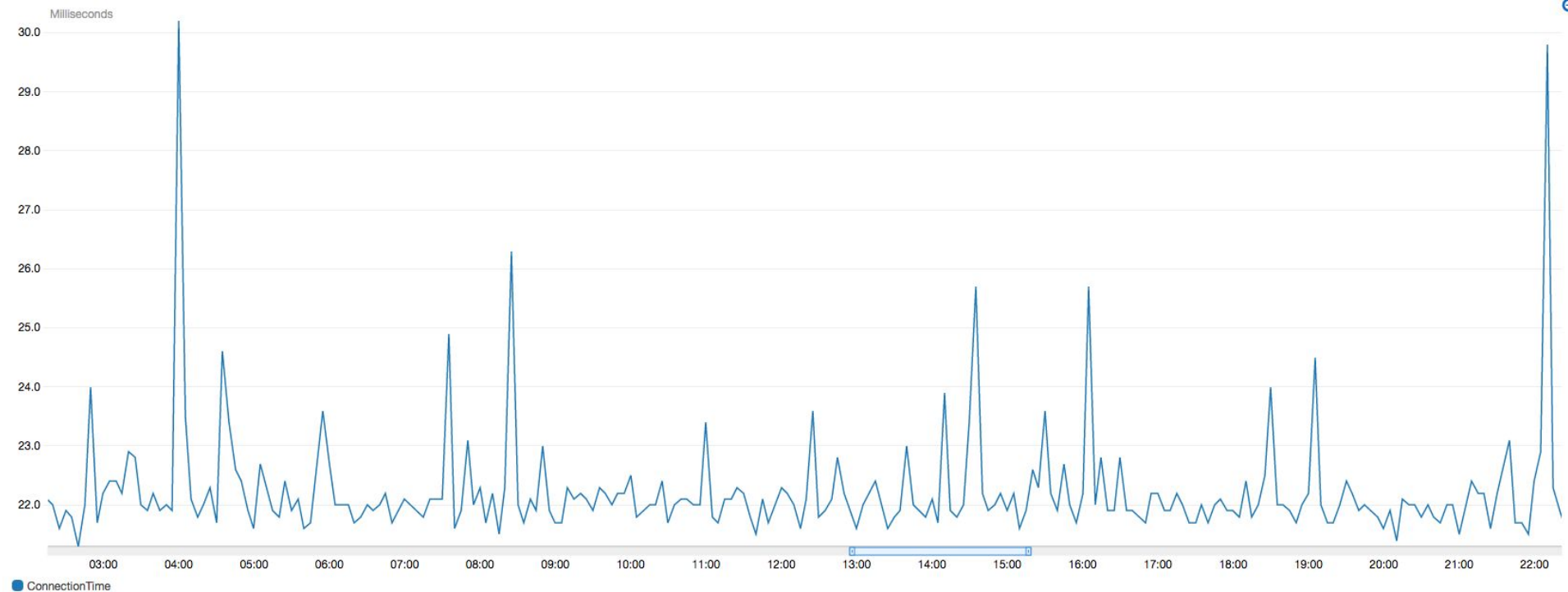
表示期間を絞ったり、対象データを絞ったり、マウスでぐりぐり

古いものほどデータが間引かれていき、1分平均、5分平均、15分平均と
だんだん粗いデータしか参照できなくなる

詳細なAWS CloudWatchの公式資料で！



ap-northeast-1 から福岡 (1min avg)

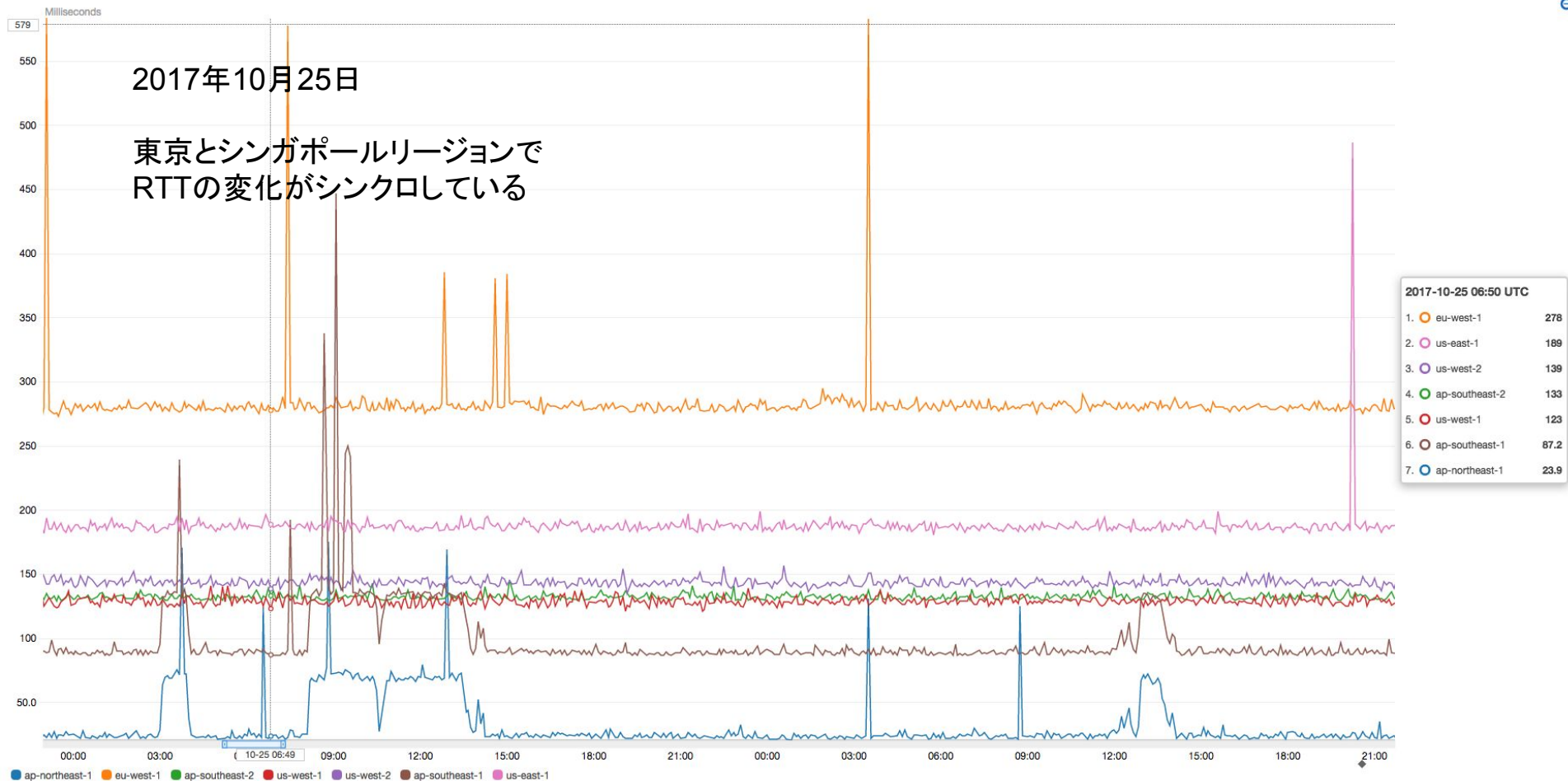


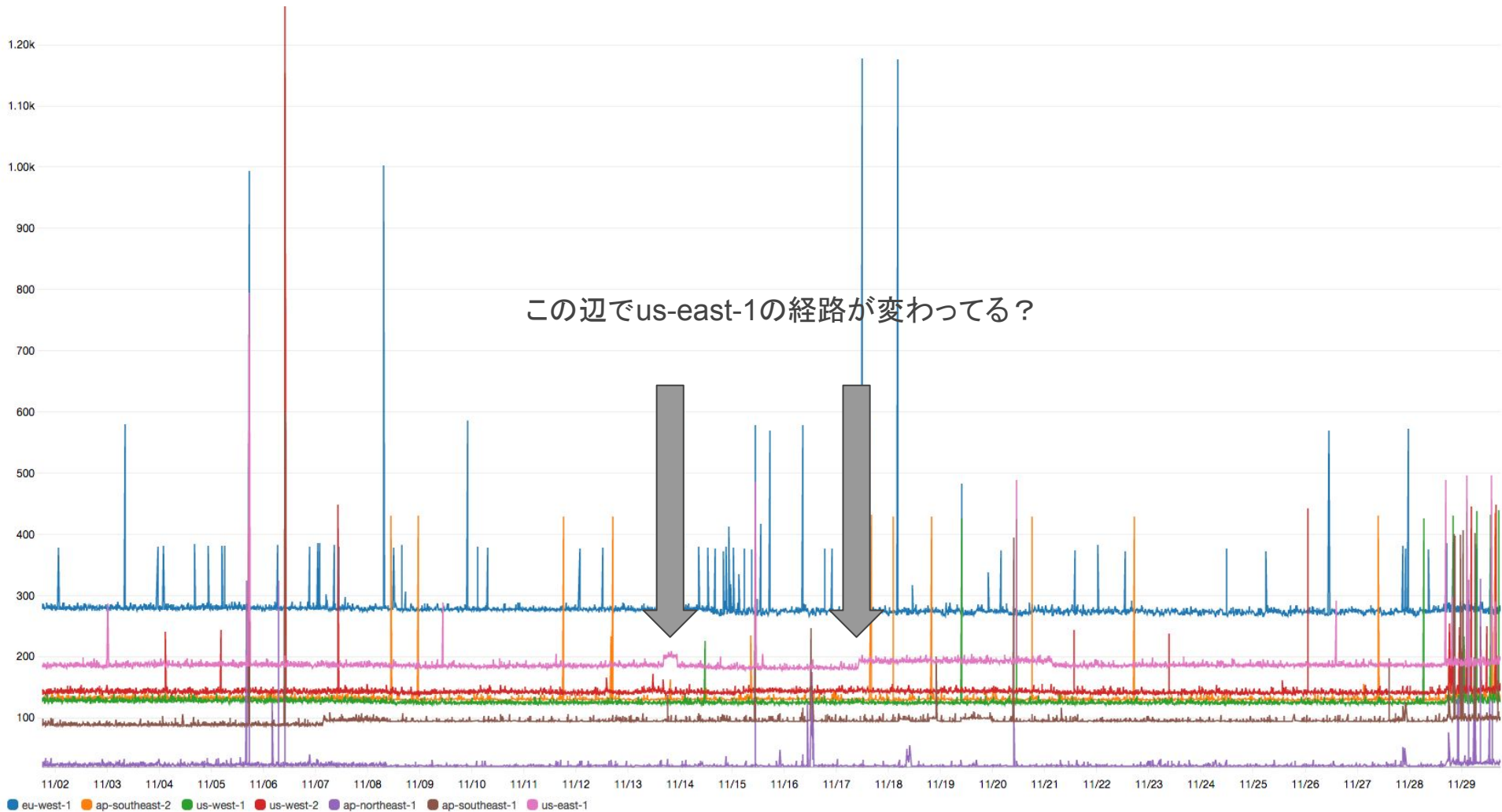
AWS東京リージョンから福岡
だいたい25ms切るくらい



2017年10月25日

東京とシンガポールリージョンで
RTTの変化がシンクロしている





あれ？



これSmokePingっぽくない？



想定していなかった副次的効果

1エンドポイントあたりわずか \$2.75/month で

回線の死活監視とSmokePing“っぽい”ことができてしまった

(RTTの幅を表現してくれないのであくまで“っぽい”)



SmokePingっぽい

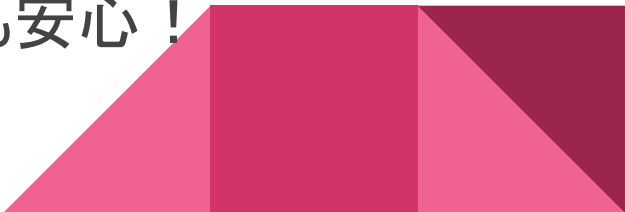
死活を監視したところで**そんなに死なない**のが現実

というより、死なせないのが仕事

運用開始後、一度もアラートは飛んできていない

こちらの副次的に得られた機能の方が重宝している

普段からRTTを記録しておくで経路が乱れても安心！



SmokePingとの違い

RTTの幅を表示してくれないので「Smoke感」がない

計測しているのはICMPではなくTCPでのRTT

行きと戻りの経路は同じとは限らない

計れるのはAWSの各リージョンからエンドポイントへのRTT



まとめ

Amazon Route 53 Health Check を使って弊社対外回線の死活監視を実現

R53HCはICMPをサポートしていないが、YAMAHAルータ上にTCPサーバを実装することでTCPによる死活監視・レイテンシの計測を実現

当初想定していなかったSmokePing的な機能が得られた

SmokePingの安価な代用としては必要十分以上 (\$2.75/month)

安く・早く・楽かつランニング人件費もわずか

