Web技術を利用したモダンなパケットアナライザの開発

—Dripcapによる次世代のパケット解析—

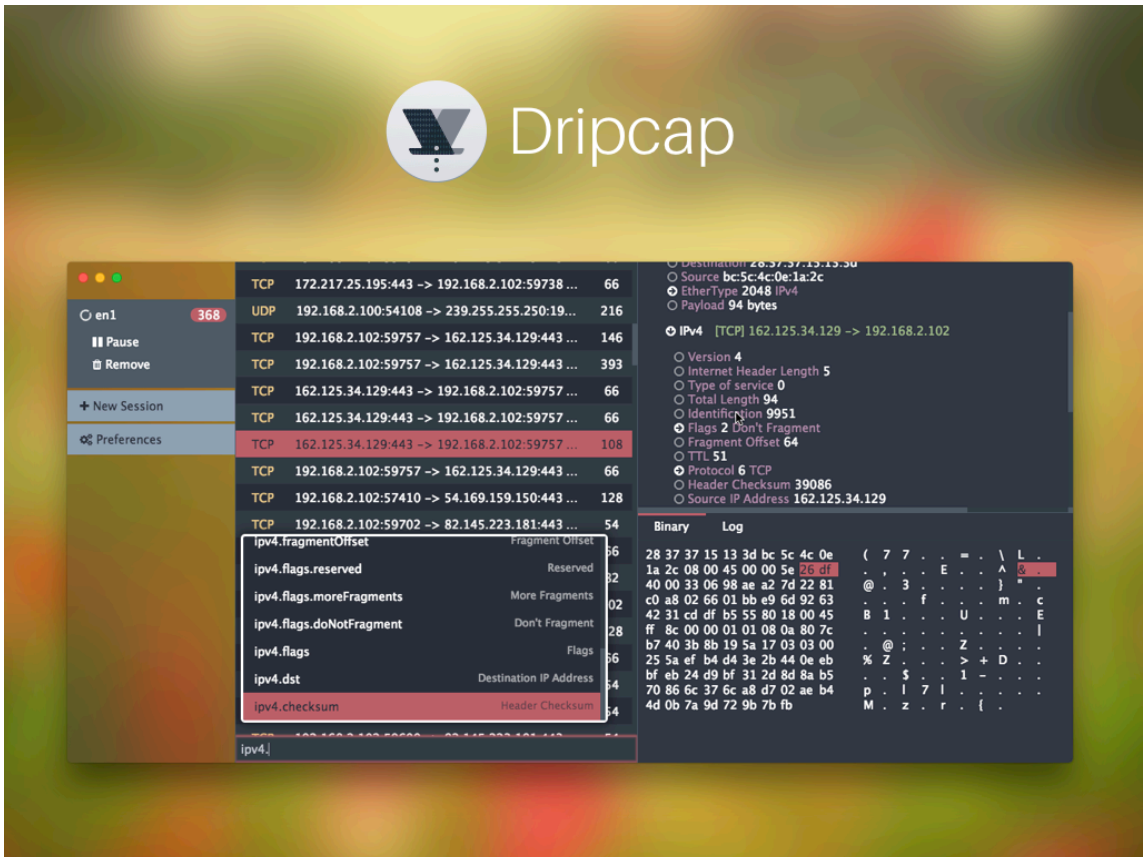

図1 Dripcap

1. 背景

近年のネットワークアプリケーションプログラミングの環境においては、Web分野ではデータサイズの増加に伴うレイテンシーが課題とされる一方、モバイルデバイスやIoTの分野では通信の軽量化や安定性が必要とされている。いずれの場合もパフォーマンスを追求するためには、汎用的なプロトコルよりも、利用目的に最適化されたプロトコルの設計が重要となる。

HTTP/2などの例に見られるように、アプリケーション層のプロトコルであってもバイナリプロトコルが採用されるようになっているが、テキストベースのプロトコルとは違ってコンピュータにとっては容易に解釈できる反面、人間にとってはデバッグが難しくなる。このような環境でプロトコルを設計・評価したり、対応するソフトウェアを効率的に開発するためには、高性能かつ拡張性に優れたデバッグツール（=パケットアナライザ）の助けを借りる必要がある。

現在、パケットアナライザに類するソフトウェアは有償・無償のものも含めて様々な種類のものがリリースされているものの、ユーザーが手軽に利用・拡張できるも

のは選択肢が限られており、オープンソースソフトウェアでは実質的にWireshark一択となっている。Wiresharkは利用実績があり数多くのプロトコルに対応しているが、ソフトウェアの設計が古く、現在のOSやハードウェア環境を十分に活かすことができていない。

2. 目的

本プロジェクトでは、ネットワークデバッグツールの選択肢を増やし、ネットワークプログラミングの開発環境を改善するために、より現代的なコンセプトのもとで新しいパケットアナライザ“Dripcap”を設計・開発した（図1）。

3. 開発の内容

Dripcapは、Wiresharkのようなパケットアナライザの基本的な機能を踏襲しており、リアルタイム・キャプチャ、プロトコル・スタック構造の表示、パケットのフィルタリングなどをGUIから実行することができる。

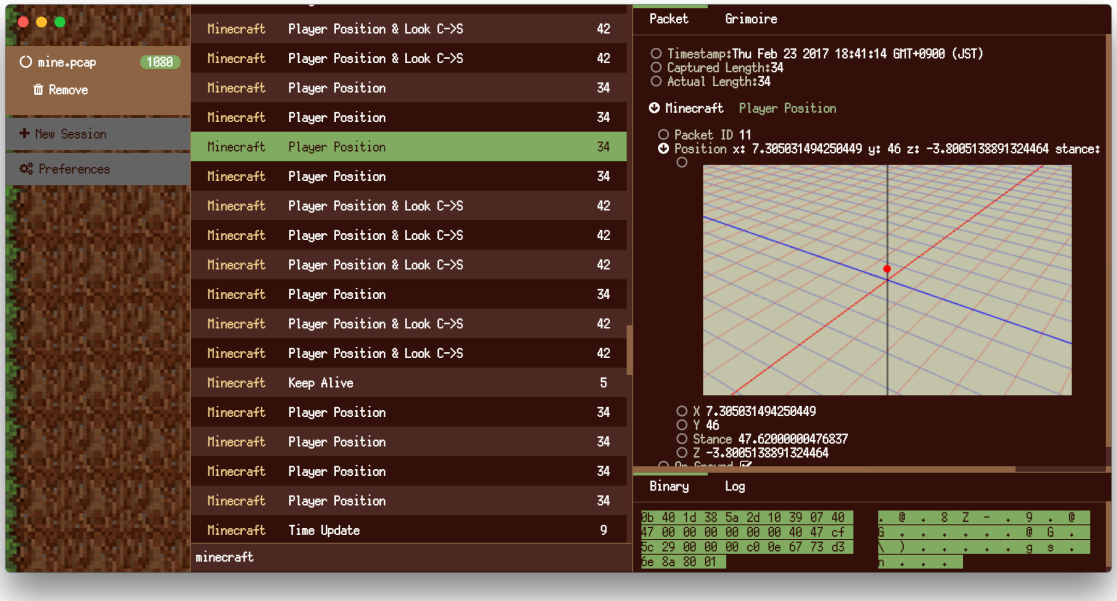

図2 座標を3D空間にプロットする

・Webベースの柔軟なGUI

アプリケーションフレームワークにはElectronを採用し、GUIのほとんどをWeb技術をつかってHTML/CSS/JavaScriptで実装している。GUI上の各コンポーネントはパッケージ単位で分割されており、パッケージを個別に有効化・無効化することでGUIのカスタマイズが可能になる。Webブラウザのレンダリングエンジンをベースとしたため、HTML/CSSにとどまらず、Canvas APIやWebGLなどの高性能なグラフックスAPIに対応している。複雑なデータ構造やアプリケーション固有の

データなどを、より視覚的にわかりやすく表現することで活用の幅を広げることができる（図2）。

・JavaScriptによるパケット解析
　JavaScriptをつかって解析用のプラグインを記述することで、対応するプロトコルを追加できる。解析用のエンジンではパフォーマンスのチューニングを行っており、複数のJavaScriptスレッドの起動によりパケット解析が並列に実行される。また、ディスプレイフィルタでもJavaScriptの文法・標準APIが利用できる。

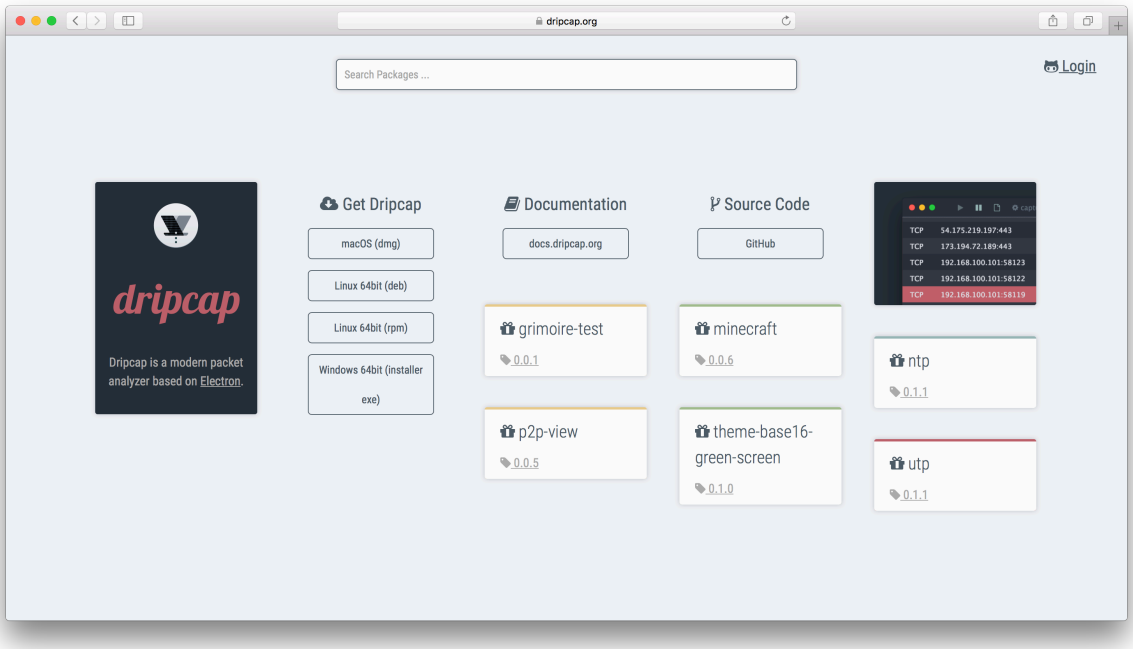

図3 パッケージレジストリ

・パッケージ管理システム
　作成したパッケージを誰でもWeb上のレジストリに登録することができ、レジストリに登録された拡張パッケージを、アプリケーションの起動中にGUIから簡単にインストール・アンインストールできる（図3）。依存しているライブラリなども自動的にnpmからダウンロードしてインストールするため、ユーザーは細かなインストール手順を気にする必要がない。データのエクスポート機能やフィルタの入力補完、UIのテーマなどの付属的な機能も、JavaScriptを使ってパッケージ単位で実装されている。

4.　従来の技術（または機能）との相違

　Wiresharkでは、それぞれの機能が密接に関連するモノリシックな設計になっており、特定の機能だけを無効化したり、コアの機能に変更を加えずに新しい機能を

追加することが難しい。またGUIの描画処理やパケットの解析などをすべて単一のスレッドで実行しているため、近年のマルチコア環境においてCPUコア数を増やしても処理がスケールせず、パフォーマンスの改善があまり期待できない。

DripcapではGUIだけでなく解析エンジンについてもモジュール化を徹底して行っているため、各機能が独立して動作するので拡張機能の開発が容易になっている。さらにGUIのスレッドとは別に解析用のスレッドを複数作って、マルチコア環境で効率的に動作するように設計されており、CPUの性能を最大限に引き出すことができる。

5. 期待される効果

高度なスクリプト機能を活用することでプロトコルの設計やアプリケーションの開発のサイクルを効率化できる。さらに、他の開発者が作った拡張機能を簡単にインストールできるようになるので、新しいプロトコルを試すたびに解析機能を自作するといった手間が軽減されるだけでなく、ネットワークプロトコルに関する情報交換を円滑にすることができる。

また、従来のネットワーク関係のプログラミングではC/C++などの低レベルの言語を中心に使うことが多かったが、DripcapではWeb技術との親和性が高いという特徴がある。そのため、低レイヤーを扱っている開発者だけではなく、Web系の開発者などにも興味を持ってもらうことができ、ネットワークプログラミングのコミュニティの活発化が期待できる。

6. 普及（または活用）の見通し

DripcapはすでにGitHub上で1300以上のスターを獲得しており、ユーザーからのフィードバックを参考にして、対応しているプロトコルの追加やパフォーマンスの改善などを続けていく。また、プラグインの開発などに参加してもらえるユーザーを増やすために、コミュニティ活動などを並行して行う予定である。

7. クリエータ名（所属）

楾本 論（株式会社PTP）

（参考）関連URL

https://github.com/dripcap/dripcap