=== ch05.docx

[chap]5章　Wiresharkの高度な機能

Wiresharkの基本的な使い方をマスターしたら、次のステップでは解析とグラフ機能を学びます。この章では、エンドポイントと[Conversation]ウィンドウ、名前解決の詳細、プロトコルの分析、ストリームフォロー、IOグラフなどを含む強力な機能の一部を見ていきます。

ネットワークエンドポイントと対話

ネットワーク通信が行われるには、最低でも2つの機器の間でデータが流れている必要があります。「エンドポイント」とは、ネットワーク上でデータを送受信する機器を指します。たとえばTCP/IP通信では、2つのエンドポイントがあります。192.168.1.25と192.168.1.30のような、データを送受信するシステムのIPアドレスです。

仮にレイヤ2で、物理的NICとそのMACアドレスの間でコミュニケーションが行われているとしましょう。データを送受信するNICのアドレスが00:ff:ac:ce:0b:deと00:ff:ac:e0:dc:0fの場合、図5-1に示したように、これらのアドレスがエンドポイントとなります。

図5-1　ネットワークのエンドポイント

通信A

エンドポイントA　エンドポイントB

通信B

エンドポイントA　エンドポイントB

ネットワーク上の「対話」は、人の会話のように2つのホスト（エンドポイント）の間で行われます。たとえば「やぁ、元気？」「元気よ。あなたは？」「この上なく元気だよ！」というジムとサリーの会話を、192.168.1.5のコンピュータと192.168.0.8のコンピュータの対話に置き換えると、「SNY」「SYN/ACK」「ACK」となります（TCP/IPの通信についての詳細は6章で学びます）。

エンドポイントを見る

あるエンドポイントの問題をピンポイントでパケット解析することもできます。[Endpoints]ウィンドウで[Statistics]から[Endpoints]を選択すると、そこから各エンドポイントのアドレスや、送受信したパケット数、バイト数などを見ることができます（図5-2）。

ウィンドウの上部にあるタブで、エンドポイントをプロトコルごとに表示することができます。[Name resolution]チェックボックスをオンにすると、名前解決を有効にすることができます。

[Endpoints]ウィンドウは、パケット一覧のペインから特定のパケットを消去するフィルタとして使用することもできます。エンドポイントを右クリックすると、いくつかのオプションが表示されます。そこから、選択したエンドポイントを含む、または除くトラフィックを表示するフィルタを作成することができます。また、選択したエンドポイントを色分けすることも可能です（色分けルールについては3章で説明しています）。

図5-2　[Endpoints]ウィンドウから各エンドポイントを表示する

ネットワーク対話を見る

[Statics]から[Conversations]を選択すると図5-3のような[Conversations]ウィンドウが開き、「Address A」および「Address B」として一覧化された対話に関連するエンドポイントのアドレスと、各コンピュータが送受信したバイトが表示されます。

ウィンドウ上部にあるタブで、このウィンドウにリストアップされた対話をプロトコル別に分けることができます。特定の対話を右クリックすると、エンドポイントAからのトラフィックのみを表示、エンドポイントBから受信したトラフィックのみを表示、エンドポイントAとB間のトラフィックのみを表示といった、フィルタを作成することができます。

図5-3　[Conversations]ウィンドウでそれぞれの対話を表示する

エンドポイントおよび対話ウィンドウのトラブルシューティング

トラブルシューティングにおいて、[Endpoints]と[Conversations]ウィンドウは非常に重要です。特にネットワーク上のかなりの量のトラフィックのソースを確かめたり、どのサーバでのコミュニケーションがもっとも多いかなどを判断したりする際に不可欠となります。

たとえば、lotsofweb.pcapというファイルを開くと、複数のクライアントがインターネットをブラウジングしていることを示す、大量のHTTPトラフィックが目に入ります。[Endpoints]ウィンドウを開けば、現在目にしているトラフィックが何かがすぐに分かるはずです。

Ipv4タブを見ると（図5-4）、バイトで分類した場合に一番上に表示されるアドレスはローカルの172.16.16.128であり、これがネットワーク上で最もコミュニケーションを多く行っていることが分かります。2番目のアドレスである74.125.103.163はローカルアドレスではないので、この段階では、ひとつのクライアントがこのIPアドレスにかなりのデータを送信しているのか、あるいは複数クライアントからデータが送信されていると推測できます。WHOIS検索（<http://whois.arin.net/ui/>）をかければこのIPアドレスがGoogleのものであると分かり、パケットを調べればこれがYouTubeのトラフィックであると分かるでしょう。

注　IPアドレス割り当ては、地理的場所によって異なるエンティティが管理します。この例ではAmerican Registry for Internet Numbers（ARIN）を使っており、ARINは米国および周辺国におけるIPアドレス割り当てを担当しています。通常はそのIPアドレスを管理している組織のWebサイトで、IPのWHOIS検索を実行します。所在地が分からず、間違った登録サイトで検索すると、実際に登録されているサイトが表示されます。アドレス登録を行っている組織には、他にAfriNIC（アフリカ）、RIPE（欧州）、APNIC（アジア太平洋）があります。

図5-4　[Endpoints]ウィンドウでどのホストの通信量が一番多いかが分かる

この情報があれば、一番上に表示されたエンドポイントの通信量が最も多いと仮定しても大丈夫でしょうか？今度は[Conversations]ウィンドウでIPv4タブを開き、バイト別分類で検証します。するとトラフィックが動画のダウンロードと一致していることが分かります。アドレスA（74.125.103.163）から送信されているバイト数が、アドレスB（172.16.16.128）から送信されているバイト数よりもかなり多いからです（図5-5）。

図5-5　[Conversations]ウィンドウで互いの通信量がもっとも多いエンドポイントが確認できる

[Endpoints]および[Conversations]ウィンドウを実際の場面でどのように使うかは、のちほど説明します。

プロトコル階層統計

巨大なキャプチャファイルを解析するとき、たとえばキャプチャしたパケットのうち何パーセントがTCPで、何パーセントがIP、何パーセントがDHCPかなど、各プロトコルがどのような配分になっているかを知る必要がある場合があります。その際、パケットを1つ1つ数える必要はありません。[Protocol Hierarchy Statistics]ウィンドウを見ればよいのです。このウィンドウはネットワークのベンチマークを知るのに役立ちます。たとえば普段はARPのトラフィックが全体の10％なのに、それが50％になっていたら、何か問題が起きていると予測できます。

lotsofweb.pcapファイルがまだ開いていることを確認し、メインメニューの[Statistics]から[Protocol Hierarchy]を選択して、[Protocol Hierarchy Statistics]ウィンドウを開きます（図5-6）。合計が100％にならない場合があります。キャプチャ量が大きくなると、最少単位以下の割合になってしまうプロトコルが出てきてしまってカウントされなくなるので、パケットの総数とプロトコルの割合が一致しなくなってしまうからです。それでも、プロトコルの配分はかなり正確といえます。

図5-6　[Protocol Hierarchy Statistics]ウィンドウでは各プロトコルの配分が表示される

トラフィックを調べる際に最初に見るもののひとつが[Protocol Hierarchy Statistics]ウィンドウです。このウィンドウを見れば、現在ネットワークで何が起きているかがよく分かるからです。トラフィックに精通するようになると、プロトコルの配分を見るだけで、ネットワーク上のユーザーや機器の概要が分かるようになります。筆者自身、ネットワークセグメントのトラフィックを見れば、そのネットワークセグメントがどこの管理下にあるかがほぼ即時に判断できます。ICMPやSNMPなどの管理プロトコルが存在すればIT部門、SMTPトラフィック量が多ければ受注処理部門、あるいは厄介なインターンがゲームに夢中になっているといった具合に。

名前解決

ネットワークのデータは00:16:CE:6E:8B:24というような、覚えにくい英数字のアドレスを使って転送されています。名前解決とは、各プロトコルが使用するアドレスを別のものに変換するプロセスのことです。たとえば00:16:CE:6E:8B:24というMACアドレスは、DNSとARPによってMarketing-2.domain.comという名前に変換されます。暗号のようなアドレスを読みやすいアドレスに変換することによって、コンピュータを識別しやすいようにしているのです。

名前解決を有効にする

メインメニューの[Capture]から[Options]を選択して[Capture Options]ダイアログを表示してください。図5-7のように、Wiresharkには3種類の名前解決ルールがあります。

MACアドレスの名前解決

ARPを使って、00:09:5B:01:02:03というようなMACアドレスを10.100.12.1というようなIPアドレスに変換します。IPアドレスに変換できない場合は、Wiresharkはプログラムディレクトリ内のethersファイルを使って変換を試みます。それにも失敗すると、MACアドレスの先頭3バイトをNetgear\_01:02:03というようにIEEEが定めたメーカー名に変換します。

IPアドレスの名前解決

192.168.1.50というようなIPアドレスを、DNSを利用してMarketingPCI.domain.comというような読みやすい名前に変換します。

ポート番号の名前解決

ポート番号を名前に変換します。たとえば80番ポートをhttpに変換します。

図5-7　[Capture Options]ダイアログから名前解決を有効にする

名前解決を使って、キャプチャファイルをより読みやすくすれば、解析の時間を節約することができます。たとえばDNSの名前解決を使えば、あるパケットの送信元のコンピュータを容易に識別することができます。

名前解決の欠点

名前解決はいいことづくしのように見えますが、以下のような欠点があります。

・ネームサーバがリクエストされた名前を解決できなければ、名前解決はできません。

・名前の情報はキャプチャファイルに保存されないため、キャプチャファイルを開くたびに名前解決を行う必要があります。そのためキャプチャファイルを開いたときにネームサーバに接続できなければ名前解決できません。

・DNSのパケットがキャプチャファイルに追加されるため、キャプチャファイルが見にくくなるかもしれません。他の問題を解析しているときには自分のトラフィックは見たくないものですが。

・名前解決のための処理時間が必要になります。巨大なキャプチャファイルを扱っていてメモリが不足しているときには、名前解決は行わないほうがよいかもしれません。

プロトコルの分析

Wiresharkはプロトコルごとの分析機能を持っており、その機能を使ってプロトコルをさまざまな要素に分解することで、解析がしやすいようにしています。たとえば生のパケットをICMPの分析機能を使って、ICMP特有の各ヘッダやデータに分けて表示するという具合です。

分析機能は、生のパケットをプロトコルとしてWiresharkに表示させる翻訳機のようなものです。Wiresharkがあるプロトコルの分析機能を持っていれば、そのプロトコルをサポートしているといえます（またはC言語やPythonを使って自分で記述することも可能です）。

Wiresharkは各パケットを解決するのに複数のプロトコルの分析機能を使用します。どの分析機能を使用するかは、プログラム化された論理から推測します。

分析機能の変更

残念ながら、Wiresharkがいつでも正しい分析機能を選択するとは限りません。デフォルトのポートを使用しないプロトコルは特に間違えやすいです（ネットワーク管理者によってセキュリティ警戒として設定されたり、または従業員によってアクセス権を管理されたりすることがあります）。そういうときには、使用する分析機能を変更することができます。

たとえば、wrongdissector.pcapを開いてみてください。このファイルには2台のコンピュータ間でのSSLを使った通信が記録されています。SSLはSecure Socket Layerプロトコルの略で、ホスト間での暗号化された安全な通信のために使用されます。普通の状況であれば、SSLは暗号化されているため、Wiresharkでチェックしてもあまり有益な情報は得られません。しかしながら何かがおかしいのは明らかです。パケットをクリックしてバイナリのペインを見てみれば、プレインテキストのトラフィックがあることにすぐに気付くでしょう。実際、4番目のパケットでは、FileZilla FTPサーバアプリケーションについて言及されているのが分かります。また次のいくつかのパケットは、ユーザー名とパスワードに対するリクエストとレスポンスを明らかに示しています。

これが本当にSSLトラフィックなら、パケットに含まれているデータは一切読めないはずで、送信されているユーザー名やパスワードは見えないはずです（図5-8）。ここに示されている情報を見る限り、これはおそらくSSLトラフィックではなく、FTPトラフィックであると仮定して問題ないでしょう。このFTPトラフィックがHTTPS（HTTP over SSL）の標準である443番ポートを使っているために、SSLと分析してしまったのです。

図5-8　プレインテキストのユーザー名とパスワード。これはSSLではなくFTPのトラフィック

この問題を解決するためには、FTPの分析機能を使うようWiresharkに指示しなければなりません。以下の手順に従って設定してください。

1．SSLパケットを右クリックして[Decode As]を選択します。どのプロトコルの分析機能を使うかを選択するダイアログが表示されます。

2．[Transport]タブでドロップダウンメニューから[destination (443)]を選択し、[FTP]をクリックします。これで、443番ポートのトラフィックでFTPの分析機能が利用されます（図5-9）。

図5-9　[Decode As]ダイアログで分析機能を指定する

3．[OK]ボタンをクリックすれば、キャプチャファイルに即座に変更が適用されます。

パケット一覧のペインから、正しく分析されているかどうかを確認してください。

注　分析機能の変更はキャプチャファイルには保存されず、Wiresharkを終了すると失われます。キャプチャファイルを開くたびに、分析機能を変更する必要があります。

1つのキャプチャファイル上でなら、同じ変更を繰り返し行うことができます。複数のキャプチャファイルを扱っているときに分析機能の変更をいちいち覚えていられないでしょうから、Wiresharkが代わりに覚えておいてくれます。[Decode As]ダイアログの[Show Current]ボタンをクリックすれば、今までに行った変更の一覧を見ることができます（図5-10）。[Clear]ボタンを押すことで、それらの変更をクリアすることも可能です。

図5-10　[Show Current]ボタンを押すと今までに行った変更の一覧が表示される

分析機能のソースコードを見る

オープンソースアプリケーションを使う醍醐味は、何が起きているのかがわからなくなったとき、ソースコードを見ればその原因が分かることです。特定のプロトコルが間違って分析されている理由を判断するときには特に便利です。

WiresharkのWebサイトで、[Develop]のリンクの上にマウスを持っていき、[Browse the Code]をクリックすれば、プロトコル分析機能のソースコードを調べることができます。ここからWiresharkのサブバージョンのリポジトリへと移動し、現在のバージョンのコードと、以前のバージョンのコードが見られます。[releases]フォルダをクリックすると、すべての公式なWireshark（およびEthereal）のリストが、上から古い順に表示されます。調べたいバージョンを選択すると、epan/dissectorsフォルダ内にプロトコル分析機能が見つかるはずです。各分析機能にはwithpackets-protocolname.cというラベルが貼られています。

ファイルはちょっと複雑ですが、いずれも標準的なテンプレートを採用しており、豊富なコメントがついていることに気づくでしょう。C言語に詳しくなくても、分析機能の基本的な機能は理解できます。Wiresharkの内容をさらに深く理解したいならば、単純な機能のプロトコルの分析機能を見ておくことをお勧めします。

TCPストリームの表示

Wiresharkのもっとも役に立つ機能の1つが、TCPストリームを読みやすいフォーマットへと変換するものです。クライアントからサーバに送信される細かいデータの破片を見るよりも、Follow TCP Stream機能はデータをまとめて見やすくしてくれます。HTTPやFTPなどのプレインテキストのアプリケーションレイヤプロトコルを見る際に特に有益です（これらの一般的なプロトコルの機能については次章で詳しく見ていきます）。

簡単なHTTPトランザクションを考えてみましょう。http\_google.pcapファイルを開き、ファイル内のTCPまたはHTTPパケットのどれかをクリックし、ファイルを右クリックして、[Follow TCP Stream]を選択します。すると別のウィンドウでTCPストリームが表示されます（図5-11）。

図5-11　読みやすいフォーマットで通信を表示する[Follow TCP Stream]ウィンドウ

このウィンドウ内のテキストは2色に色分けされています。赤は送信元から宛先へのトラフィックを、青は反対方向、つまり宛先から送信元へのトラフィックを示しています。どちらから発信されたかで色分けされているのです。たとえばこの例では、クライアントがWebサーバへ向けて接続を開始しているので、赤で表示されています。

このTCPストリームでは、2つのホスト間での通信のほとんどを見ることができます。通信はWebルートディレクタ（/）へのGETリクエストで始まり、サーバはリクエストを受け取ったことをHTTP/1.1 200 OKのフォームで返しています。クライアントが個々のファイルをリクエストし、サーバがそれに対して応答するといった、同様のパターンがストリームで繰り返されています。これはユーザーが実際にGoogleホームページで見ているものを、内側から見ているのです。

TCPストリームは、ウィンドウ内で生のデータとして見るだけでなく、テキスト内を検索したり、テキストファイルとして保存または印刷したりすることができます。また文字列をASCII、EBCDIC、16進数、C言語の配列に変換することも可能です。これらのオプションは[Follow TCP Stream]ウィンドウの下にあります。

TCPストリームの表示は、特定のプロトコルを扱うときに非常に役立つでしょう。

パケット長

ひとつのパケットまたはひとまとめのパケットのサイズによって、状況が分かります。一般的な状況であれば、イーサネットの最大フレームサイズは1518バイトです。この数値からイーサネット、IP、TCPヘッダを差し引くと1460バイトが残り、これをレイヤ7プロトコルヘッダまたはデータ送信に使うことができます。これを頭に入れたうえで、パケット長の分布を使って、キャプチャしたトラフィックについて考えてみましょう。download-slow.pcapファイルを開き、[Statistics]から[Packet Lengths]を選択、[Create Stat]をクリックします。すると図5-12のようなウィンドウが表示されます。

図5-12　キャプチャしたファイルのトラフィック理解に役立つ[Packet Lengths]ウィンドウ

サイズが1280バイトから2559バイトのパケットの統計を示している部分がハイライトされています。サイズの大きなパケットは通常データ転送を、小さなパケットはプロトコルのシーケンス制御を示しています。ここでは、大きなパケットの比率がかなり大きくなっています（66.43パーセント）。そのためファイルの中のパケットを見なくても、キャプチャファイルには１つまたは複数のデータ転送が含まれていることがわかります。つまりこれはHTTPダウンロードかFTPアップロード、あるいはホスト間でデータ転送が行われているその他のネット通信だということになります。

残りのパケット（33.44パーセント）のほとんどのサイズは、40バイトから79バイトです。このサイズのパケットは通常、データを含まないTCP制御パケットです。一般的なプロトコルヘッダのサイズを考えてみましょう。イーサネットヘッダは14バイト（プラス4バイトのCRC）、IPヘッダは最少で20バイト、そしてデータやオプションのないTCPパケットも20バイトです。つまり標準的なTCP制御パケット（SYN、ACK、RST、FINなど）は54バイト程度であり、この範囲に収まることになります。もちろんIPやTCPオプションがあればサイズは増えます。

パケット長を調べると、キャプチャファイルの全景がよく分かります。サイズの大きなパケットがたくさんあれば、データが転送されていると考えてよいでしょう。またパケットの大半が小さいサイズなら、データ転送があまり行われていない、プロトコル制御命令で構成されていると考えられます。これは確実なルールではありませんが、さらに細かい分析を行う前に、こうした仮説を立てておくのが有益な場合があります。

グラフ

グラフは分析の基本であり、データセットの概要を得る最適の方法のひとつです。Wiresharkではキャプチャしたデータの理解を助けるいくつかのグラフ機能がありますが、そのひとつがIOグラフ機能です。

IOグラフを見る

Wiresharkでは、[IO Graphs]ウィンドウで送受信されているデータのグラフを見ることができます。このグラフで各プロトコルのスループットを見てネットワークの動静を確認したり、パフォーマンスのラグを見つけたり、データストリームを比較したりすることができます。

インターネットからファイルをダウンロードしたときのIOグラフの例を見るには、download-fast.pcapを開いてください。TCPパケットをクリックしてハイライトし、[Statistics]から[IO Graphs]を選択します。

[IO Graphs]ウィンドウで、キャプチャファイルのデータの流れを、グラフとして見ることができます。図5-13の例では、1秒平均500パケットの転送が最初から最後までほぼ同じように続き、最後に下がっていることがグラフから分かります。

図5-13　安定した高速ダウンロードを示すIOグラフ

これを速度の遅いダウンロードと比較してみましょう。現在のファイルは開いたままにし、Wiresharkの別のインスタンスを開いて、download-slow.pcapを開きます。遅いダウンロードをIOグラフにしてみると、図5-14のように違いがはっきりとわかります。

図5-14　安定していない遅いダウンロードを示すIOグラフ

このダウンロードのデータ転送レートは1秒当たり0から100パケットで、安定からは程遠く、1秒当たりのパケット数がほぼ0になっている時さえあります。2つのダウンロードのグラフを並べてみると、その不安定さは一目瞭然です（図5-15）。

図5-15　複数のIOグラフを並べると違いの識別に役立つ

このウィンドウの下に比較のオプションがあるのが分かります。最高5つまでフィルタが作成でき（6章と7章で説明しますが、表示やキャプチャにも同じ構文が使えます）、フィルタを色分けできます。たとえば、ARPを赤、DHCPを青で表示するフィルタを作成、折れ線グラフで表示すれば、2つのプロトコルにおけるスループットの傾向を簡単に見分けることが可能です。

ラウンドトリップタイムグラフ

Wiresharkには、キャプチャファイルのラウンドトリップタイムを表示できるグラフ機能もあります。ラウンドトリップタイム（RTT）とは、パケットの受信が確認されるまでにかかる時間を指します。つまりパケットが別のコンピュータに届き、その返答が返ってくるのにかかった時間です。RTTの分析により、通信が遅れたポイントやボトルネックが見つかり、レイテンシがわかります。

この機能を使ってみましょう。download-fastファイルを開いてください。TCPパケットを選び、[Statistics]から[TCP Stream Graph]、さらに[Round Trip Time Graph]を選択すると、図5-16のようなRTTグラフが表示されます。

図5-16　このダウンロードのRTTグラフはほぼ安定しており、値の分散はわずかしか見られない

グラフのそれぞれの点がパケットのRTTを表しています。デフォルトではシーケンス番号によって分類された値が表示されます。グラフ内の点をクリックすると、パケット一覧のペインの該当パケットに直接移動できます。

RTTグラフを見ると、高速ダウンロードのRTT値はほぼ0.05秒以下で、やや遅い場合でも0.10秒から0.25秒程度であることが分かります。許容範囲を超えている値もわずかにあるものの、RTT値のほとんどは問題がないので、ファイルのダウンロードとしては問題ないとみなせるでしょう。

フローグラフ

接続を視覚化し、時間経過におけるデータの流れを表すのに、フローグラフ機能は非常に便利です。フローグラフは基本的に、ホスト間の接続を表すカラムビューとなっており、見てすぐにわかるよう、トラフィックが分けられています。

フローグラフを作成するには、http\_google.pcapファイルを開き、[Statistics]から[Flow Graph]を選択します。すると処理を行うパケットに関するいくつかのオプションと、フローのタイプを示す小さなダイアログが表示されます。この例ではデフォルトのままで問題ないので、[OK]　をクリックし、フローグラフを作成します（図5-17）。

図5-17　TCPフローグラフで接続を視覚化できる

エキスパート情報

Wiresharkのプロトコル分析では、パケット内である状態が発生した場合に警告してくれるよう、「エキスパート情報」を設定することができます。これらの状態は4分野に分けることができます。

チャット　通信の基本情報

ノート　通常通信の一部であると思われる異常なパケット

警告　通常通信の一部とは思われない異常なパケット

エラー　パケットのエラー、または分析でエラーと判断された場合

では例を見てみましょう。download-slow.pcapファイルを開き、[Analyze]をクリック、[Expert Info Composite]を選択して、[Expert Infos]ウィンドウを表示します（図5-18）。

図5-18　[Expert Infos]ウィンドウはエキスパートシステムからの情報を表示する

ウィンドウには分類された情報ごとにタブがあり、エラーはゼロ、警告が3、ノートが18、チャットが3あることがわかります。タブに示されたカッコの外の数字はメッセージの件数、カッコ内の数字はメッセージの総数を示しています。

このキャプチャファイルのメッセージがすべてTCP関連なのは、この文章を書いている時点では、他のプロトコルにエキスパート情報システムが導入されていなかったからです。ここにあるTCP関連の14個のエキスパート情報メッセージは、トラブルシューティングに非常に役立ちます。特定の条件を満たすと、個々のパケットにフラグを立ててくれるからです。

チャットメッセージ

Window Update　TCP受信ウィンドウのサイズが変わったことを送信者に知らせる通知が受信者によって送られる。

ノートメッセージ

TCP Retransmission　パケット消失の結果。ACKが重複して受信された、またはパケットの再送タイマーがタイムアウトとなった場合。

Duplicate ACK　ホストが次のシーケンス番号を受け取らなかった場合、受信した最後のデータの重複ACKを生成する。

Zero Window Probe　ゼロウィンドウパケットが送信された後、TCP受信ウィンドウの状態をモニターするのに使われる（9章で説明）。

Keep Alive ACK　キープアライブパケットに応答を送る。

Zero Window Probe ACK　ゼロウィンドウプローブパケットに応答を送る。

Window is Full　受信者のTCP受信ウィンドウがいっぱいの状態であることを、送信側のホストに知らせる際に使う。

警告メッセージ

Previous Segment Lost　パケット消失を意味する。データストリームにおいてシーケンス番号がとんだ場合に起きる。

ACKed Lost Packet　ACKパケットが見えるものの、認識されない場合に起こる。

Keep Alive　接続のキープアライブパケットが見える場合にトリガされる。

Zero Window　TCP受信ウィンドウのサイズが一定値に達し、ゼロウィンドウ通知が送られ、送信者にデータ送信を停止するよう要求した場合に見られる。

Out-of-Order　シーケンス番号を利用して、番号から外れたパケットがいつ受信されたかを調べる。

Fast Retransmission　重複ACKの20ミリ秒以内に起こる再送信。

エラーメッセージ

No Error Messages

6章でTCPを学習し、9章で遅いネットワークのトラブルシューティングについて学べば、これらのメッセージの意味が明確になるでしょう。

本章で説明したいくつかの機能は、めったに起こらない状況でしか使わないように見えるかもしれませんが、思った以上に利用することになるでしょう。これらのウィンドウとオプションに慣れておいてください。この後の章で何度も触れることになります。