2020年8月20日 投稿者: SATOXITS

GShell 0.1.6 – リモートシェル化

開発：さて、今日はGShellのリモートシェル化に取り組みたいと思います。

社長：いよいよ本丸ですね。

開発：最大の目玉は、ローカルシェルとリモートシェルとの協調動作だと思います。ローカルはただリモートの入出力を端末にストリーム的に中継するだけでは無い。少くともこの2つはやりたい。

- ヒストリ共有
- ファイル共有

社長：コマンドの実行プログラムとファイル環境とかはリモートにあるんだけれど、コマンドの解釈まではローカルでやるという方法があり得ますね。

開発：特にリモートとの時間距離が遠い場合、手元のttyでインタラクションしながら実行する場合、その方法しか無いと思います。

基盤：フランクフルト支部での苦い体験が考えを変えましたね。

開発：あれで地球の大きさを実感しました。HDDのレイテンシ10msを許容基準に考えると、ホスト間250msのレイテンシは問題外です。そういう世界がまだあるというか、今後も本質的にあり続けるんだろうと実感したのです。

社長：ライトセールのお陰様でした。

基準：レイテンシが10msでもうんともすんとも動かないFUSEって何者なんでしょうね？

開発：4月にITSTPプロトコルを試作した際には、ファイル関係のシステムコールを全部リモートに飛ばすというナイーブな方法をやったわけですが、これはレイテンシーの高い

環境では使い物にならない可能性が高いです。

社長：やはりアプリケーションレベルでの意味的な大きなまとまりでやらないとダメですね。そして処理とデータの局所性。

開発：ただファイルを共有できれば良いという意味でなら、リモートのファイルをローカルのファイルシステムの一部として見せるという通常のネットワークファイル共有があります。ですが、リモートにあるファイルに対して何らかの処理をする場合、その処理はそのファイルのあるリモートのプログラムで行わないと、まともな性能にならない。

社長：たとえば find コマンドのファイルがリモートにあれば find もリモートで実行するってことですね。

開発：一番シンプルな例が、同じホスト内でのファイルのコピーで、これをいちいちリモート経由でやっていると話になんらない。

基盤：ネットワークファイルシステムでは実際そういう事にんっちゃうわけですけど。

開発：ファイル関係のシステムコールに、コピーが無いのが問題と思いますね。それに、コピーする際の簡単な加工方法、圧縮とか指定できたら、世の中の無駄なトラフィックは大幅に減るかもしれません。

社長：システムコールとかNFSとか、ファイルの中身に微細にアクセスするレイヤでそれをやるのはやはり難しそうなので、FTPとかWebDAV的なファイルをひとつの塊としてしか見ないプロトコルにそういうコマンドを入れるのが適切でしょうね。

社長：ユーザにローカルとリモートをどう見せるかが重要と思います。両方を同時に見せたいわけですよね。

開発：シェルから見える仮想的なディレクトリにするのが自然と思います。で、カレントディレクトリと言うかワーキングディレクトリの移動によって、接続先のシェルを代える。chdir @host/path みたいな。リモートのファイルに対するアクセスを行うコマンドはリモートで実行する。

開発：両者にまたがるコマンドについては、どっちにしろ局所性が無いので、例えばファイルの単純コピーとかはどっちでやってもよいでしょね。実現のしやすさで言えばローカルで実行。処理を伴うもの、例えばリモートで圧縮した結果をローカルに書くとかだと、もちろんリモートで圧縮を実行するのが吉。

社長：そのへんの自動判定が難しくて面白いでしょうね。状況を見て途中で処理を交代するとかできたらブラボーですが。ランダムアクセスとかリアルタイムなアクセスが多いよ

うなファイルの近くでコマンドを実行する。

開発：場合によっては、リモートに必要なコマンドが無い場合もあり得るので、その場合にはローカルまで持ってきては処理する。RPC的にリモートでopenしてローカルなファイルディスクリプタからアクセスする。ここはITSTPでなくても、Goのパッケージを使えば良いかなと最近は思います。

社長：あるいは、Goプログラム型式の処理コマンドを生成してリモートに送り込んで、リモートでコンパイルして実行する。

基盤：夢は広がりんぐですが、どこから手を付けるんでしょう？

開発：やっぱり chdir と which と find ですかね。ベースは、Remote Procedure Call 的な Remote GShell Commandじゃないかと。

社長：そういえば昔、rexec とか使ってたような記憶があります。

基盤：r系はなんで絶滅しちゃったんですかね？

開発：別のレイヤで実現されるべきセキュリティの話と混同されて魔女狩りにあったように思いますね。NFSとかtelnetなんかも。そもそも rlogin が telnet の上位互換に実現されなかった時点でアレ？って感じでしたが。

社長：OSIの7層モデルみたいのの上の層がインターネットには無かったですからね。セッション層みたいなのが。全部がアプリケーションプロトコルに要求されちゃってなんともまあ。

開発：その点SSLというかTLSはセキュアなレイヤをアプリと独立に挿しはさんで良かったと思うんです。なぜSSHが必要だったのかとか、普及したのかは良くわかりません。

社長：垂直統合的にワンセットにパッケージされててとっつきやすかったんだと思います。

＊　＊　＊

開発：さて、今日の仕事を始める前に、現状の0.1.5をアーカイブして置きます。

GShell-0.1.5-by-SatoxITS ダウンロード

開発：この版では、移動して来たディレクトリの履歴を @N で参照できるようにしました。

```
iMac% gsh
!1! version -l
gsh/0.1.5 (2020-0819a)
!2! pwd
/Users/ysato/gsh
!3! cd
!4  @1 [Aug 20 12:41:42] /Users/ysato
!4! cd /
!5  @2 [Aug 20 12:41:45] /
!5! cd -a
!0  @0 [Aug 20 12:41:30] /Users/ysato/gsh
!4  @1 [Aug 20 12:41:42] /Users/ysato
!5  @2 [Aug 20 12:41:45] /
!6! hi -at
!1  @0 /Users/ysato/gsh version -l
!2  @0 /Users/ysato/gsh pwd
!3  @0 /Users/ysato/gsh cd
!4  @1 /Users/ysato     cd /
!5  @2 /         cd -a
!7! cd @0
!8  @3 [Aug 20 12:42:34] /Users/ysato/gsh
```

開発：N番目のコマンドを実行した時のディレクトリという参照もしたいので、これを !Nd というにすることにしました。一方、N番目のディレクトリに居た時に何をやったかを見たいことも多いので、これは history @N で表示できるようにしました。

社長：history コマンドに限らず、@Nh とかで、@N でのヒストリに展開するのが良いかもですね。

開発：そうですね。表記方法はまだ仮り置きですが、機能はそういう事にしましょう。

基盤：要は、各種の履歴を一旦普通の文字列の配列にしたら、あとは汎用のフィルタリングとかソートとか集計ができる、という感じですかね。

開発：汎用的な処理をした結果をまた、情報の出どころの属性に従って適切に表示できるというところが肝かなと思います。

社長：ところでGShellのコードの行数が2500行を超えましたが。

開発：色々試行錯誤中だったり、Goの事をよく知らないで非効率な書き方をしてたり、ロゴのイメージデータを含んでますからね。正味ではまだ2000行に達してないと思います。

基盤：リモートシェル機能を入れたらどっとデカくなるのでは。

開発：骨組の部分は本体に残しますが、部品的な部分は流石に外に出すことになると思います。

社長：わたし的には昔のPascalのコンパイラが3000行くらいだったとか、Unix v6 のカーネルが10000行足らずだったというのがすごく基準なんですよね。Unixのコードなんてコメントも多かったしドライバとかも含んでいたから、いわゆるカーネル部分はやっぱり数千行だと思うんです。たったそれだけでそれまで世に無かった新しい世界が創造できちゃう。感動的です。

開発：DeleGateは9.9.13で30万行に達しましたが、ネジ一本まで自作でしたし、今日手に入る部品を最大限に活用したら、10分の1くらいでできるかも知れません。

社長：昔はそのネジが世の中になかったんで、仕方がないですけどね。

＊　＊　＊

広報：そういえばロゴですが、現行のは、Noto Sans の S がやせてて元気がないという指摘がありました。

広報：Courier New の Sが好ましいとのことでしたので、GShまではCourier という版を作ってみました。

社長：うーむ、Gがへなちょこ過ぎる感じですね…

開発：Courier New って大きくするとへなちょこなんですね。小さくても読みやすいというところにデザインの力点があるのかな。

基盤：というかこのメタフォント、やたら微妙で情報量が多そうな気がしますが…

社長：そもそもGは、GoのロゴのGにあやかりたいわけでして…

基盤：Sを画像にして横に引き伸ばしたらどうですかね？

広報：こんな感じでしょうか？

社長：これいい。今度からこれにしましょう。

基盤：画像にしちゃうと使いまわしが面倒にはなりますけどね。

広報：いずれ時間がとれましたら、フォントを自作したいと思います。ただ、よさげなフォント作成ソフトが安いのでも数千円はしまして…

社長：承認。

社長：そういえば、カッチカチだったメロンのお尻がようやく柔らかくなったので、冷蔵庫に入れて冷やします。

基盤：今日の打ち上げはメロン (^-^)/

＊　＊　＊

社長：さて、どこから始めますかね。

開発：まずは、隣で動いてる GShell にちょっかいを掛けてコマンドを送って結果を得るということろだと思います。rexecですね。これはこないだウェブブラウザからHTTPでちょっかい掛けたのと大して違いません。ただし、結果は標準入出力というストリームだけでなく、実行結果のメタ情報的な標準結果出力的なものも得る。

社長：ちょっかいを受けるほうがgshdということですね。2種類の結果をどう受け取るかですが。単一コネクション上でシーケンシャルに受け取るか、もうひとつコネクションを張るか。シーケンシャルに受け取る場合にはコネクションを元からマルチプレックスして複数のチャネルを作るか、一つのストリームの中にtelnetプロトコルのように特殊なパケット的なものを返すか。ストリーム入出力はリアルタイムですが、実行結果のメタ情報は最後の付け足しというか、並列性はない場合が多いように思います。

社長：マルチプレックスするならDeleGateで作ったYYMUXも使えます。yysh は YYMUX を使って X Window とかファイル転送をバックグラウンドでやっていますね。GoはHTTP2対応なので、HTTP2のマルチプレクサみたいのも使えると思います。

開発：基本、屋上屋的なものはやめたいと思います。並列チャネルはそれぞれ実際に複数のTCPをつないでしまう。FTPのデータコネクション方式ですね。あれは野蛮だと思ってましたが、最近はあれが正しいような気がしています。

開発：経験上、アプリケーション層でのマルチプレックスは、本来意図されている通信のリアルタイム性を、マルチプレクサでのスケジューリングが邪魔をする危険性があります。特にスループットが必要な大きなデータの転送が、レスポンスが重要な小さいデータの転送の邪魔をする。

社長：まあそれはyysh/YYMUXでもありました。ファイル転送中はキーのエコーがとろくなるとか。

開発：あるいは、マルチプレックスのための処理が、ギガビットのデータ転送能力を殺してしまうかも知れない。

社長：out of band で追い越せるんじゃなかったでしたっけ？telnet ではやってますよね。

開発：あれは必ずしもうまくいかなかった記憶が。到達性とか。システム依存だったようにも思います。一方コネクションを用途別に分ければ、一つのコネクション上にオリジナルのひとつながりのデータが流れるので、中間での加工が必要な場合のような用途にも適しています。

社長：ただあれは、別チャネルにはサーバ側に別ポートが必要になりますね。まさかPORTみたいにサーバ側から接続するというのは、外部サーバでの使用を考えるとありえない気がしますが。PASVにしたって、ルータを動的に設定できないと、攻撃者の妨害を受ける危険性があります。

開発：ですのでそこは、同一のポートで受けたいと思います。通常の、ログイン的な主制御のコネクションを受けるし、副コネクションも同じポートで受ける。

社長：複数並列セッションを考えないなら良いですが、複数セッションのサーバが並列に動いていた時に、副コネクションをしかるべきサーバが受け取るのが難しいと思います。

開発：確かプロセス間でファイルディスクリプタを渡せた気がするんですよ。おそらくAF_UNIXのソケット経由で。だから、ポートの入り口の管理プロセス、デーモンが、主コネクションを受けたらGShellサーバプロセスを生成する、副コネクションを受けたらそれを適切なGShellサーバプロセスに渡す。これで良いのではないかと。最悪、入り口デーモンで中継しても良いです。

開発：そもそもこれはサーバ側の問題なので、サーバ側でのファイアウォールフレンドリ性はあまり考慮しなくて良いと思います。

社長：リモートログインの場合には、接続先がおうちのがっちりファイアウォール内ってこともありますよね。

開発：その場合はFTPのPORTコマンド式に、サーバからクライアントに接続すれば良いと思います。

基盤：サーバもクライアントもがっちりファイアウォール内ってこともありますね。

開発：その場合は、リフレクタ方式で対処しても良いですね。リフレクタで中継すれば良いと思います。

開発：なんにしろこれは、副チャネルではUDPを目一杯使う可能性もあるわけです。FTP的に言えば、制御コネクションはTCPで、データコネクションはUDPでっていう事もあり得る。一本のTCP接続の上に主チャネルとマルチプレックスするというのは、その目的にも適合しません。

社長：確かにUDPという話になると、そもそも単一ポートで複数のチャネルを最高速で受け取るのは難しいですね。

開発：サーバに複数のコマンドを先出しで送っておいて、あるいは並列に実行させて、結果が順不同で帰ってくることもありえます。

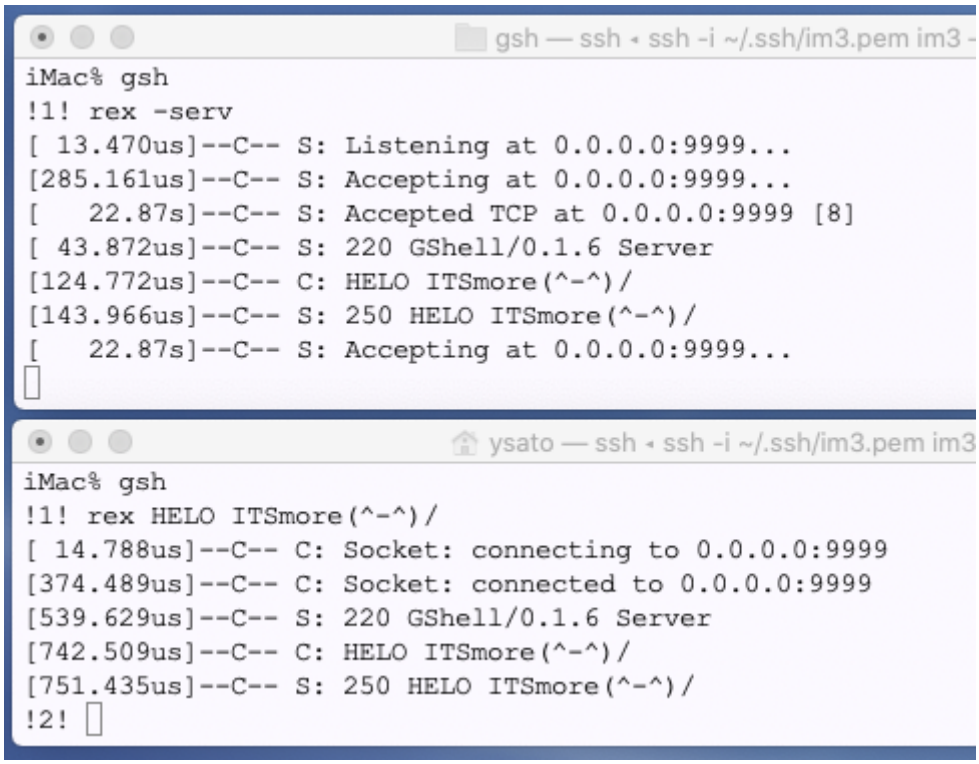
基盤：えーと、この件はとりあえず先延ばしで良さそうにも思います。

開発：基本、クライアントとサーバ間は自由にTCPコネクションが張れUDPが送れることを想定します。一番性能がでて一番簡単に作れる方式。運用上それが出来ない場合には、TCP上にマルチプレクスするなりで対処する。これは実装を変えずにトランスペアレントにラッピングなりできるはずです。VPN使うという手もあります。

社長：そうですね。ではそういう方向性で。

＊　＊　＊

開発：ということで、第1版ができました。じゃん。



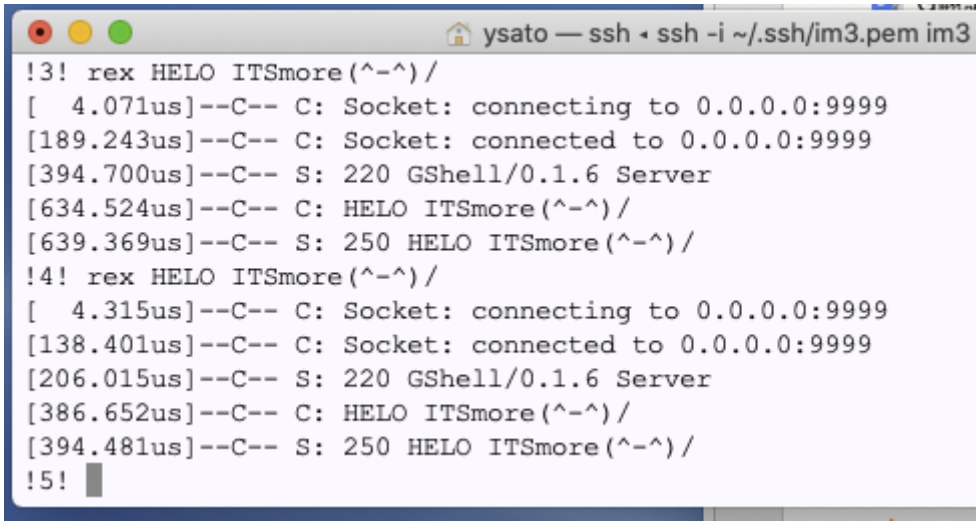
開発：遠隔コマンドを rex という名前にしました。rexec への懐旧です。rex -serv でサーバになります。-serv オプションがなければクライアントとしてサーバにコマンドを送ります。

社長：これは、SMTPプロトコル互換なんですかね…

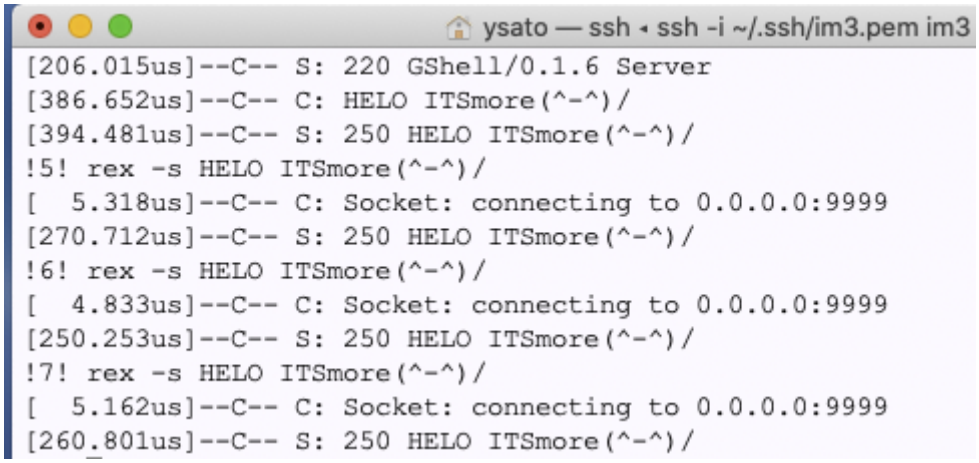
開発：先人の知恵に従うということで。と言いますか、SMTPでもFTPでもNNTPでもPOPでも、古典的なインターネットのアプリケーション・プロトコルはみんなこれ流ですよね。HTTPはコネクションレスなのでちょっと違いますが踏襲しているわけです。系統が違うと言えばTelnetくらいですね。

基盤：マルチコアの同一ホスト上で1リクエスト750マイクロ秒もかかるんですか？

開発：最近はホストのiMacもそれなりに仕事てますしね。状況によりけりです。最短だと400マイクロ秒は切ります。



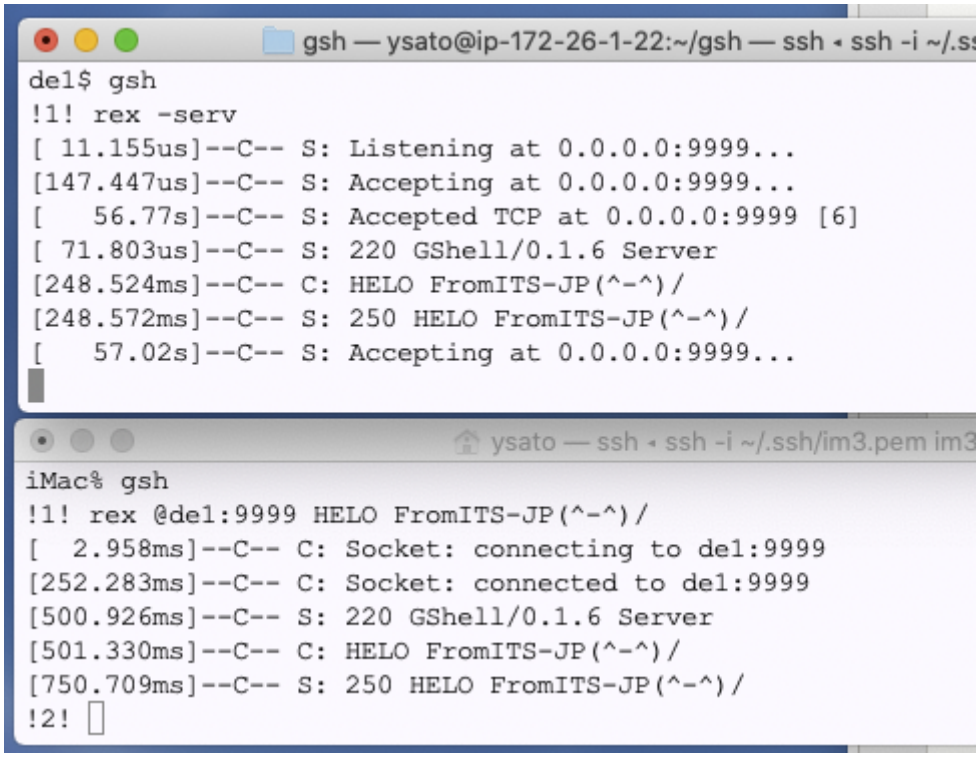
開発：そもそもログをコンソールにだしているので時間がかかりますから、これを時間に影響の無いのだけにしぼれば、250マイクロ秒くらいにはなります。



社長：コマンドはどのようなものがあるのですか？

開発：今の所HELOコマンドだけです。

社長：さて、それでは焦眉のフランクフルトを呼んでみましょう。



開発：750ミリ秒と出ました。

基盤：ローカルホストより2000倍遅いですね。

社長：きっちり250msの3回分ですか。

開発：サーバのオープニングご挨拶を待たずにコマンドを送れば500msにはなると思います。でもそれが限界ですかね。

基盤：SYNパケットにリクエストデータが載せられたらいいのにｗ

社長：でもTCPが開通した後は250msで帰ってくるのでは。少し待って応答をACKに乗せてくれませんかね。コマンドの応答として250msなら、許容できると思います。

開発：双方の現状をバックグランドで定期的に知らせ合っておけば、いざ人間がコマンドを送るという時に送る情報も少なくて済みますね。

社長：pwd とかは毎回応答を待つ必要も無いですね。

基盤：SMTPじゃなくてFTPにしませんか？

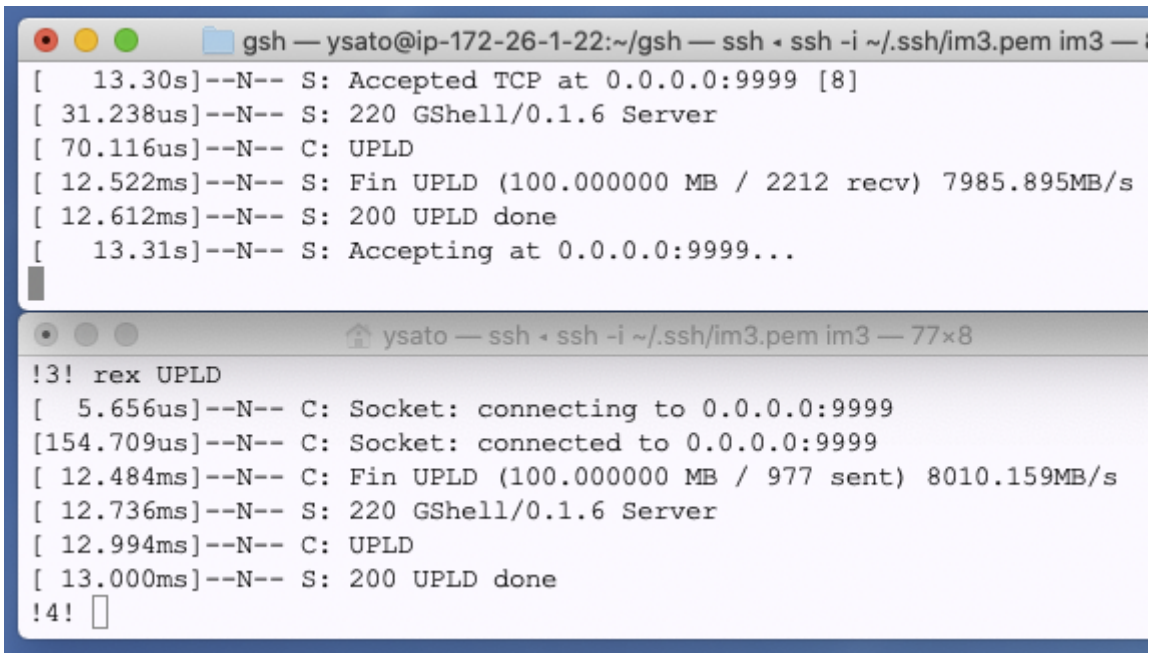
開発：構いませんよ。というか、太古にはFTPコネクションにSMTPコマンドを同居させてたりした時代もあるようです。

社長：putとgetコマンドを所望。NNTP的にPOSTでも。

開発：別チャネルコネクションの問題は置いておきたいので、ダミーでUPLOADとDOWNLOADみたいなコマンドでも作りましょうか。

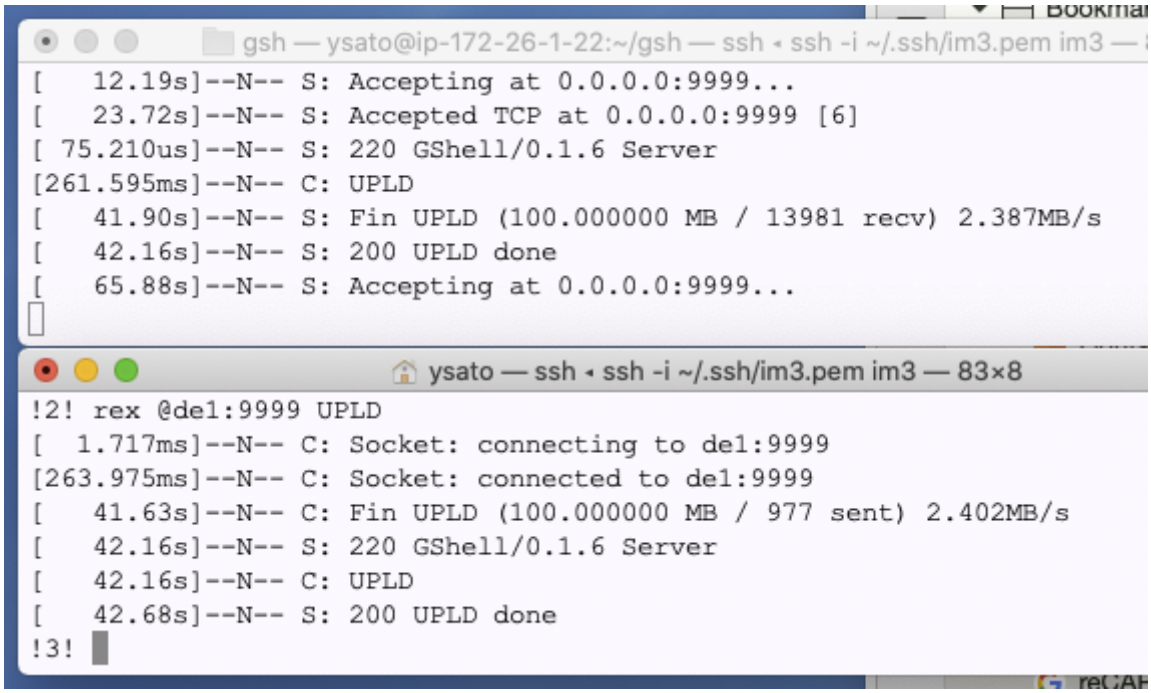
社長：あと、やってみるまでも無いですがリモートでのファイルの超速コピーを実現するCOPYコマンドとか(^-^)

開発：では刹那的に、100MBのデータをサーバに向けて送信するコマンド、UPLDを作りました。まずはホスト内での転送。



基盤：8GB/s出ますね。

開発：そして問題のフランクフルト…



開発：2.4MB/sでした。

基盤：予想通りですね。

開発：だいぶこま切れになって届いています。受信一回平均7152バイト。

社長：ファイルのsyncコマンドを作りましょうよ。

開発：でもちょっと休みましょう。というか、そろそろロイヤルなリンクスの時間・・・

＊　＊　＊

基盤：ところで、ポート番号ってなんなんですかね。アプリケーションプロトコルごとにポート番号を代えるより、一つのポートに入ってからどのプロトコルで話すかネゴっても良いと思うのですが。

開発：まあHTTPのUpgradeとかはそれですよね。というかそもそも初期のプロトコルにそういうプロトコルを切り替える変態コマンドを持ってるのがありましたよ。なんでしたっけ？

社長：TCPなら性能上問題ないでしょうけど、UDPだとネゴった結果の状態が保持が難しいし、一つのポートに到着したパケットを誰が使うか、応用層のプログラムで振り分けたら重いんじないですかね。

基盤：でも、iptables みたいなルータも、ただのソフトですよね。普通OSの中にあるでしょうけど。そもそもさっきの例みたいに、ソフトでやったって8GB/s程度の中継はできるみたいですから、UDPの振り分けだってユーザのアプリでやっても問題ないんじゃないでしょうか？とりあえずDeleGateのudprelayをSOCKSで使うとか。

開発：そうですねえ。実際に使えるネットのバンド幅が100MB/sでしかないとすると、ユーザ空間のプログラムで中継をやっても楽勝っぽいですね。まあ優先度が低くされて受信をこぼしちゃうことは増えるかもですが。

社長：そういえば、カーネルの中にあると思われるUDPパケットのバッファっの量って、ユーザが設定できないんでしょうかね。どうもちっちゃい臭い気がするんですが。受信脳力が他のアプリの動作に影響を受けるというか、緩衝能力が低い感じ。

開発：まあ1ギガビット、約100MB/sのネットなら、バッファがわずか100MBあれば、1秒間アプリでの処理が途絶えても持ちこたえる計算ですね。

基盤：それでポート番号ですが、ネットワークが作られ始めたころのもろもろの処理能力とかで、処理を軽くしたかったとか、機能ごとにきっちり縦割りしたい文化とか、硬直的な機能割当てとか、ユーザにはネットワークに手を触れさせない前提とか、とにかくなんか時代錯誤な感じもするんです。

開発：まあ、ポート番号ていらなくね？てすると、IPアドレスだってほとんどはいらなくね？て感じにはなると思いますけどね。アプリケーションレベルで識別する位置な名前だけあれば。末端で相手のアドレスをレゾルブする必要があるのか、とても疑問です。アプリケーションレベルで、プロキシサーバみたいのにやらせておけば良いと思うんです。

社長：番号での識別子には、アクセス制御の面の利点があると思います。IPアドレスやポート番号ならハードウェアでフィルタリングできる。このポートはこのサービスに使うって確定していれば、サービスへのアクセス制御が高速にできる。あるいは、ネットワークのモニタリングにも都合がよいですよね。

開発：やはりシンプルな処理は、専用ハードウェアでやれば、汎用CPUで実行するソフトウェアより2桁以上は高速にできます。私自身もソフトで300マイクロ秒かかる処理が、FPGAで3マイクロ秒でできちゃうとか、感動しました。

基盤：ただ、入り口は一つにして、そこで共通の認証なりアクセス制限をして、その先に進めたほうが、管理も実装も簡単になると思うんですよね。プロトコルごとにてんでにセッションの認証サブプロトコルを作るとか意味わからないです。

開発：そういえば、UnixのプロセスIDとかもう、まるでユニークIDじゃなくなってますよね。例えば1ミリ秒に1プロセス生成したら、1分で6万ですから、あっというまに一回りしてしまう。find -exec grep とか、実際に普通にやるわけですし。

開発：おや、いきなりボギーで発進ですか・・・

＊　＊　＊

— 2020-0820 SatoxITS