peercast技術合同誌

Broadcast yourself

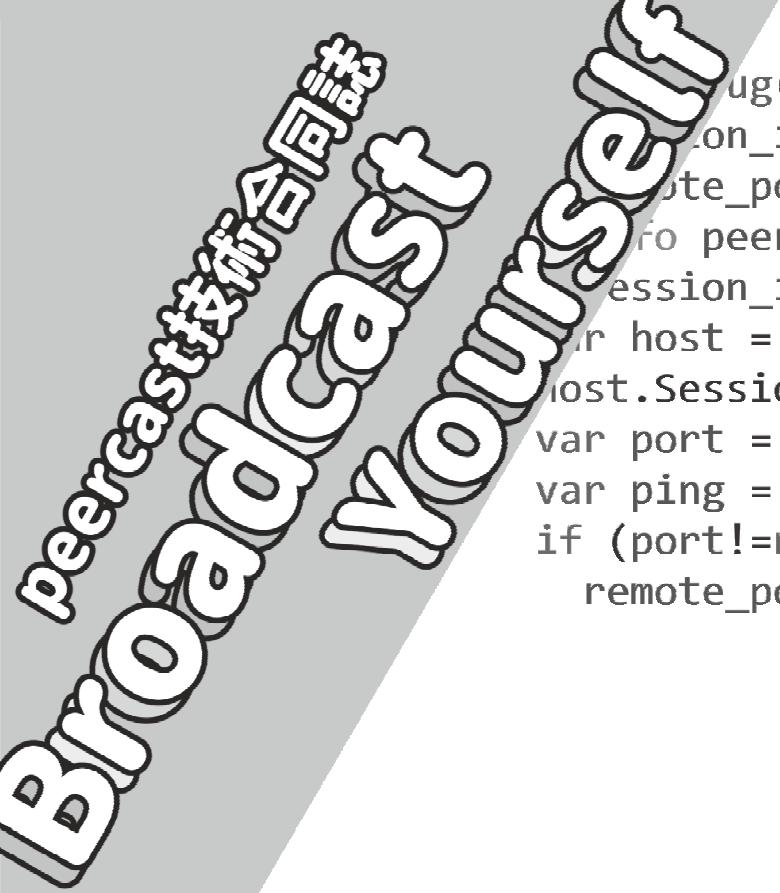
IPv6とつながらないP2P

CS:GO サーバー改造事始め

VRC民に効く(かもしれない)
Unity操作Tipsまとめ

Misreading Chat
リスニングガイド

センテンス指向
プログラミングの提案



IPv6とつながらないP2P

CS:GO サーバー改造事始め

VRC民に効く(かもしれない)
Unity操作Tipsまとめ

Misreading Chat
リスニングガイド

センテンス指向
プログラミングの提案

2019 あれくま

目次

1. IPv6 とつながらない P2P 6
    1. IPv6 とは 6
    2. よくわからない「IPv6」 7
    3. P2P とは 9
    4. IPv4 でもつながらない P2P 11
    5. NAT 越え 13
    6. IPv6 でつながらない P2P 16
    7. ファイアウォール越え 18
    8. 参考文献 19
2. CS:GO サーバー改造事始め 20
    1. サーバー構築 20
    2. addon の導入 22
    3. 自分だけの mod を作ってみよう 23
3. VRC 民に効く (かもしれない)Unity 操作 Tips まとめ 26
    1. カメラ 26
    2. エディタ 27
    3. プロジェクト 28
4. Misreading Chat リスニングガイド 30
    1. 序文 30
    2. 機械学習 30
    3. カメラ、画像処理、GPU 33
    4. あとがき 35
    5. 出典 35
5. センテンス指向プログラミングの提案 36
    1. はじめに 36
    2. マーチン・ファウラーさんについて 36
    3. 流れるようなインタフェース 36
    4. センテンス指向 38
    5. デメリット 41
    6. さいごに 42

はじめに

この本は PeerCast コミュニティの技術合同誌です。

PeerCast っていうのはオープンソースの P2P 動画配信ソフトなんですが、そのユーザーが集まって技術関係の記事を載せました。

ただ PeerCast のユーザーが書いたってだけで、この本の内容に PeerCast 自体はほとんど関係ないので、PeerCast 全然知らんという人でも安心してお読みいただけます。逆に PeerCast についてはなーんにも書いてませんので、もし PeerCast について知りたければ別途調べてみましょう。

IPv6 とつながらない P2P

著 :kumaryu

おうちで IPv6 を使うことと、IPv6 で (IPv4 でも)P2P がなかなかつながらないことを解説します。

おうちで使うことを想定しているので、IPv6 についてよくわからない人向けです。

1. IPv6 とは

IPv6 ていうのはインターネットで使われる新しい通信形式です。

みんながインターネットでよく使ってるのは IPv4 です。

IPv6 は何が違うかっていうといろいろあるんですが、IPv4 との大きな違いはみんなにグローバルアドレスが割り当てられることです。グローバルアドレスっていうのは直接インターネットで通信できるアドレスですね。LAN 内のみのアドレスなんかはプライベートアドレスです。

グローバルアドレスくらい今でも貰ってるよ〜と思うかもしれませんが、普通のご家庭であればグローバルアドレスはせいぜい 1 個しか貰えない一方で、どこのご家庭でも PC3 〜 3 台はあるしスマホはあるしゲーム機もあるしで 5 台くらいはインターネットにつなぎたい機器があるでしょう。普通のご家庭でも 5 台くらいはね。それらにはローカルネットワークのアドレス (プライベートアドレス) が割り当てられ、インターネットとのやりとりはグローバルアドレスが割り当てられたルーターが仲介しています。

それに対して IPv6 の場合は全部の機器にグローバルアドレスを割り当てることができます。IPv6 だとおうちにグローバルアドレスが約 2 の 64 乗個分貰えます。さすがにそんなにいらないし全部は使えないんですけど、これなら PC が 20 〜 30 台あっても余裕で全部にグローバルアドレスが割り当てることができます。グローバルアドレスがみんなに割り当てられていると、家のルーターがインターネットとのやりとりを仲介する必要がなくなるのでちょっとすっきりします。まあいろんな都合で家のルーターは相変わらず必要なんですけど！

他に IPv4 との違いと言えば中身が全然違うことです。中身がどう違うかは気にしなくていいんですけど、全然違うものなので相互につながりません。IPv4 と IPv6 は (直接には) つながりません。つなげたい場合には途中で変換を入れる必要があります。

今まで IPv4 で足りてるのにそれとつながらない IPv6 を入れる必要ないじゃん……という気がするのはもっともですが、IPv4 もここまでインターネットが使われるとは思ってなかった時代にできたものなんで、一般ユーザーにはわかりづらい世界でいろいろと不都合が出ているようです。なんとか頑張って IPv4 を維持はしていますが、無理を重ねるより IPv6 に移行した方が素直ですよねということで、徐々に IPv6 への移行が行われています。ほんと徐々に。

ところで IPv4 と IPv6 は相互につながらないので、従来 IPv4 で見ていた Web サイトは IPv6 で

は見れなくなるのでしょうか？実のところ見れなくなるんですが、Web サーバーに IPv4 と IPv6 の両方でつながるようにしておくのは比較的簡単であり、現にそうしているサイトもありますので、IPv6 は従来のインターネットと別物になる！とかそういうわけではありません。とはいえ全部のサイトが IPv6 から見れるかというとそういうこともなく、むしろ 2019 年時点でもいまだに IPv6 で接続できないサイトも非常に多くあります。そのため IPv6 を使うときは IPv4 も同時に使って補完するのが普通です。徐々に IPv6 に移行はしていくはずですが、IPv6 のみで十分にインターネットを使えるようになるにはまだまだかかりそうです。

2. よくわからない「IPv6」

家で光回線を使ってると「IPv6」もしくは「IPoE」にするとインターネットが速くなる！という話を聞いたことがある人も多いでしょう。多いかどうかは知らんけど。

IPv6 は速いのでしょうか？いや実際はそんな速くないっす。IPv4 に比べて仕様が整理された IPv6 の方がネットワーク機器の処理が速くなる余地はありますが[1]、その差は家の回線での通信速度には大きく表れないでしょう。

じゃあ速くなるってのは嘘なのかというとそうでもないんですけど、IPv6 だから速いわけじゃないです。簡単に言うとフレッツ光の IPv4 は混んでるけど、IPv6 はなんとかなりそうなので速い (ことがある) てわけです。

日本の光回線だとフレッツ光[2]のシェアが高いのですが、フレッツ光の複雑な事情により IPv4 が遅くなってたり IPv6 が速いと言われることになっています。つまり NTT の光回線使ってない場合には関係ないです。他の会社でも IPv6 の方が速いことはあるかもしれないけど、まあそれはまたその回線業者の事情によるもので、一般的に言われてるのとは違う可能性があります。

フレッツ光は光回線ですが、直接インターネットにはつながっていない独自ネットワークを引いています。なにやら法律によって NTT 東西が直接インターネット接続を提供することができなくなっています。そのためみんながインターネットに接続する場合には、NTT 独自ネットワークからインターネットに出ていく装置を経由して接続しています。PPPoE というプロトコルでインターネットに出ていく装置に接続しているのですが、近年ではこのインターネットに出ていく装置が混んでいるので回線速度が遅くなっています。じゃあ装置を増強すればいいんですけど、この装置は NTT 東西が増強することになっており、どうもあんまり増やしたくないようです (お金もかかるし、IPv6 に移行してほしい？)。

ところでフレッツ光の独自ネットワークは IPv6 で構築されています。せっかくなのでこのままインターネットにつながれば便利なんですけど、NTT 東西は直接インターネット接続サービスを

1　実際速くなるかどうかは機器の実装によりますし、IPv4 には長年の最適化があるので絶対 IPv6 の方が速いとは言えないようです。

2　ここで言うフレッツ光は正確にはフレッツ光ネクストで、B フレッツはまた話が別なんですが、めんどうなのでフレッツ光と略します。B フレッツはもうほとんど無いので。

提供できません。そこでいくつかのISP[3]がNTT独自ネットワークのIPv6からそのままインターネットに出ていく装置を接続できるようにしました。こっちの装置はプロトコル変換が必要ないのでそんなに詰まらなくて速くできるようです。この方式で接続するのがIPoEという方式です。

これがIPv6(またはIPoE)にすると速くなると言われている仕組みです。IPv6自体の話っていうよりフレッツ光の話なんですよね……。

ところで、ここにIPv6は従来のIPv4と違うので相互につながらないという問題が出てきます。Webサービスや通信するアプリケーションなんかはまだまだIPv4を使ってるのが大半で、IPv6ではつながりません。IPoEだとプロトコル変換の必要がなくIPv6のままインターネットに出ていくことができますが、いくら速くてもいままで使ってるWebサイトやアプリが使えないと意味ないですね。そこでIPv6のネットワークにIPv4のデータを流してどっかで変換してやりましょうという手法が使われています。

IPv6とIPv4の変換にはDS-LiteやMAP-Eといった手法が使われているんですが、どの方式にしても、まずIPv4のリクエストを投げると途中の変換器(ご家庭にあるルーターとか)でそのデータをIPv6のパケットに載せてIPv6のネットワークに流します。その後IPv6ネットワーク上にある変換器(ISPが管理してるルーターとか)が受け取った中のIPv4のデータをIPv4のネットワークに流す、ということをやっています。逆にIPv4のネットワークからデータを受け取る時も、ISP側の変換器で受け取ったIPv4のレスポンスをIPv6パケットに載せてIPv6ネットワークに流し、ご家庭の変換器で中のIPv4データを戻してIPv4として通信します。

この方法を使うと、各ご家庭からIPv4とIPv6両方のネットワークに接続することなく、IPv4とIPv6の両方のネットワークと通信することができます。一方で問題点もあり、一つは変換が挟まることです。IPoEを使ってPPPoEの変換が不要になったので速い！とか言ったくせに、でもIPv4を使うには変換が必要でーす、では意味あるのかそれ……？となってしまいますね。まあ意味あるかどうかで言うと今のところ意味はあるようです。この変換を使っているユーザーはまだ比較的多くないため、混みあってるPPPoEより空いているので速度が出ます。また、変換器はISPの所有なので、ISPが足りないと判断すればNTTの都合に関係なくISPの責任で増やすことができます。ただ今後いつまで速度を維持できるかは経過を見ないとわかりません。

もう一つの問題はP2P通信がとてもしづらいことです！やっと本題が出てきたぞ！インターネット通信するにはグローバルアドレスが必要になりますが、IPv4のアドレスはIPv6のアドレスよりずっと少ないので、IPv6のグローバルアドレス1つに対してIPv4のグローバルアドレスを割り当てるわけにはいきません。そこまでいかずともIPv4はもういっぱいいっぱいで、これから各ご家庭に1つずつ割り当てられるよう確保するのも難しくなっています。そのため各種IPv4 over IPv6技術では、IPv4アドレスは複数のご家庭で共用することになります。PPPoEのIPv4時代では各家庭に1つのグローバルアドレスをご家庭内の機器で共用していましたが、IPv4 over IPv6時代には複数のご家庭の機器で1つのグローバルアドレスを共用するため、P2P的には状況は悪化しています。なんてこった。しかしながら近年は回線速度の問題も大企業がお金でゴリ押す時代、P2Pな

3　インターネットサービスプロバイダ、略してISP。プロバイダってよく言うやつ。OCNとかYahoo!BBとかBiglobeとかの回線事業者と別に契約しないとインターネットにつなげないやつ。

んざきょうび流行んねーんだよ！と言わんばかりに肩身の狭い思いをさせられているので、大して問題視はされてないのかもしれません。

3. P2P とは

P2P とはなんだろなということですが、これといった技術的な定義があるものではありません。Peer to Peer の略で P2P なんですが、直訳すると仲間から仲間へ、とか、同等の地位にあるやつ同士でということになります。これだけだといまいち意味がわからないですね。

まず P2P じゃないものとしてクライアント / サーバー方式を説明しましょう。これはサービスを提供するサーバー (アプリケーション) に対して、サービスの提供を受けるクライアント (アプリケーション) が接続に行く方式で、代表的なのは Web サーバーと Web ブラウザ (クライアント) です。サービスを提供するっていうのがちょっと曖昧なところもあるのですが、だいたいは接続を待ち受けてる方がサーバー､接続しに行く側がクライアントと呼ばれます｡またクライアントとサーバーではそれぞれ機能が異なります。たとえば Web サーバーはファイルやデータを Web ブラウザ (クライアント) に送りますが、クライアントから HTML 等を受信して画面に表示する機能は持っていません。Web ブラウザは Web サーバーからの要求を受けてデータを提供することはありません[4]。

逆に P2P というのはサービスを提供する側、受ける側という区別をしません。接続する各アプリケーションが相互に同等の機能を持って通信しあうのが P2P です。

P2P の例としては Bittorrent という P2P のファイル共有に使われるアプリケーション (やプロトコル) があります。ファイルを転送するという意味では Web サーバーや Web ブラウザと似ていますが、Bittorrent のアプリケーションはファイルをダウンロードすると、同時に他の Bittorrent アプリケーションから接続を待ち受けて他のコンピューター (ノード) にもファイルを転送します。Bittorrent アプリケーションとしてはファイルの提供もダウンロードも両方行いますので、接続される側・接続しに行く側も同じ機能を持っています。これが P2P です。

P2P はどういう時に使われているかというと、上に挙げたファイル共有の他にゲームのオンライン対戦やボイスチャット (VoIP) があります。オンライン対戦は参加者間で内容の同期が取れていればよくサーバーを経由するのは無駄に遅くなることもあるので、対戦するプレイヤーそれぞれが相互に通信したりします[5]。ボイスチャット (やビデオチャット) も同様に、参加者間で相互に音声や動画のやりとりができれば良いので P2P が合っています。録音やリアルタイム文字起こし・翻訳サービスなんかが入るとクライアント / サーバー方式になりそうですが。

P2P の利点とはしては主に 2 つあります。

4　基本動作としては、ね。最近の Web ブラウザはいろんな機能があるのでサーバーからの要求に応えてデータを送ることもありますが

5　もちろん全部のオンライン対戦が P2P になっているというわけではありませんが、特に 1 対 1 のような少人数であれば P2P が使わることが多いでしょう。人数が多くなってくると相互に接続するのが難しく無駄も出てくるので、クライアント / サーバー方式の方が便利です。

一つはネットワークの帯域消費を分散できるということです。Bittorrent なんかのファイル共有で活用されている利点ですね。普通に Web サーバーから大きなファイルを配信しようとしますと、サーバーから外に出ていくネットワーク帯域がファイルサイズ×ダウンロード数分必要になります。沢山のネットワーク帯域消費は沢山のお金がかかるため、できれば少なく抑えたいものです。そこで既にダウンロード済みのノードから再配信してもらうことで、ネットワーク帯域の消費を分散させます。ユーザー側から外に出ていくネットワーク帯域が必要になってしまいますが、たとえば 1 回ダウンロードにつき 1 回再配信したとしすると、落としたファイルサイズ分のネットワーク帯域が余分にかかるだけですので、その程度は許容できることが多いでしょう。また、ダウンロードする場合も、サーバーがネットワーク的に遠くにあるときには近くの他ノードから落とした方が速くなることが (いくらか) 期待できます。ファイル共有以外に VoIP の場合も特にビデオチャットなんかでは帯域消費が大きくなりがちですし、サーバーを余計に通過させないことでネットワーク帯域の消費が抑えることができます。

もう一つはレイテンシが抑えられることです。レイテンシってのは相手にデータが届くまでの時間です。こちらはゲームのオンライン対戦や VoIP なんかで活用される利点です。ゲームのオンライン対戦でこちらの動きを対戦相手に伝えることを考えましょう。サーバーおよび各プレイヤー間の通信がそれぞれ 10ms で行えるとします。クライアント / サーバー方式では、サーバーに対して自分の動きデータを送り、対戦相手の動きデータはサーバーから受け取ります。サーバーに対して自分の動きデータを送るのに 10ms、サーバーから対戦相手に自分の動きデータが届くまでさらに 10ms で、対戦相手に自分の動きが届くまで合計 20ms かかります。一方で P2P にすると、対戦相手に直接自分の動きを送るため、合計 10ms で対戦相手に自分の動きが届きます。一般的には、入力してから動きが反映されるまでの時間は短い方が操作しやすいので、10ms で届く P2P の方が操作感覚は良くなります。VoIP なんかでもこのレイテンシは短い方が会話がしやすくなるので P2P が使われがちです。テレビの海外中継なんかでお互いの声や映像が届くのが遅れていると会話しづらそうなのを見ますよね。

P2P の欠点もいくつもありますが、同じく 2 つ挙げるとしましょう。

欠点の一つは作るのが難しいところです。

クライアント / サーバー方式の場合はクライアントはサーバーとの接続だけで良く、サーバーだけが多数のクライアントとの接続を管理します。また、クライアントがどのサーバーに接続するかというのは事前に決まってたりユーザーが指定するのが普通です。P2P の場合は各ノードが他のノードとの接続を管理しないといけません。そしてどこに接続しに行けばいいかというのはそのときどきによって変わることがほとんどです。

もう一つの欠点はつながらないことです。ネットワークで通信をつなげる時、特に接続待ち受けをする時には、グローバルアドレスを持っているかやファイアウォールの設定だとかいろいろ気をつけるところがあります。クライアント / サーバー方式の場合には接続待ち受けをするのはサーバーだけですし、サーバーだけが気をつけて設定してやれば接続できるようになります。P2P の場合は各ユーザーが接続待ち受けをするため、つながるようにする設定を各ノードで行わないといけません。しかしながら各ユーザーというのはただアプリケーションを使いたいだけであって、サー

バー管理者に比べてネットワークの設定について疎く適切に設定することが難しい場合が多いのです。そのためアプリケーション側でなんとか簡単につながるようにサポートする必要があります。

4. IPv4 でもつながらない P2P

IPv4 で P2P がそう簡単につながらない話をしましょう。

まず IPv4 アドレスだと各マシンがグローバルアドレスを持っていないのが普通です。IPv4 のグローバルアドレスはご家庭に 1 個ずつくらいしか貰えないので普通はルーターにだけ割り当てられます。各マシンにはプライベートアドレスだけが割り当てられ、インターネットとの通信はルーターで中継して行われます。

このプライベートアドレスを割り振られたマシンとインターネット間をルーターで中継して通信する仕組みを NAT(Network Address Translation) と呼びます。NAT が使われている場合には P2P での通信が難しくなります。

例を挙げて説明していきます。

まず一般的なご家庭で 5 台のマシンがルーターに接続されており (有線無線を問わず)、`192.168.1.10` ～ `192.168.1.14` のプライベートアドレスが割り振られているとします。一般的なご家庭なら 5 台くらいは余裕であることでしょう。ルーターにはグローバル IP アドレスの `198.51.100.2` を持っています。

Web サイトを提供する Web サーバーはインターネット上にあってグローバルアドレス `203.0.113.1` を持っているとします。

`192.168.1.10:54321`[6] から `203.0.113.1:80` にアクセスしようとすると、まずルーターが `203.0.113.1:80` にアクセスしたいというパケットを受け取ります。リクエスト送信先のサーバーはグローバルアドレスを持っているのでそのまま接続できるのですが、返事を受け取るマシンはプライベートアドレスしか持っていません。サーバーが返事を送ろうにもプライベートアドレスだとつながらないのです。そのためルーターはじゃあ自分が持っているグローバルアドレスをかわりに使ってくれよな、`198.51.100.2:55555` から `203.0.113.1:80` に接続しまーす、とパケットを書き換えて転送します。さらにルーターは `198.51.100.2:54321` に返事が来たら `192.168.1.10:54321` に転送してやるよう記憶しておきます。で、Web サーバーからの返事はその通り `198.51.100.2:54321` に来ますので、ルーターは覚えておいた通り `192.168.1.10:54321` に転送してあげることで正しく通信が完了します。

6　アドレス `192.168.1.10` のポート番号 `54321` の意味です。ポート番号は適当なので特に意味はないです。

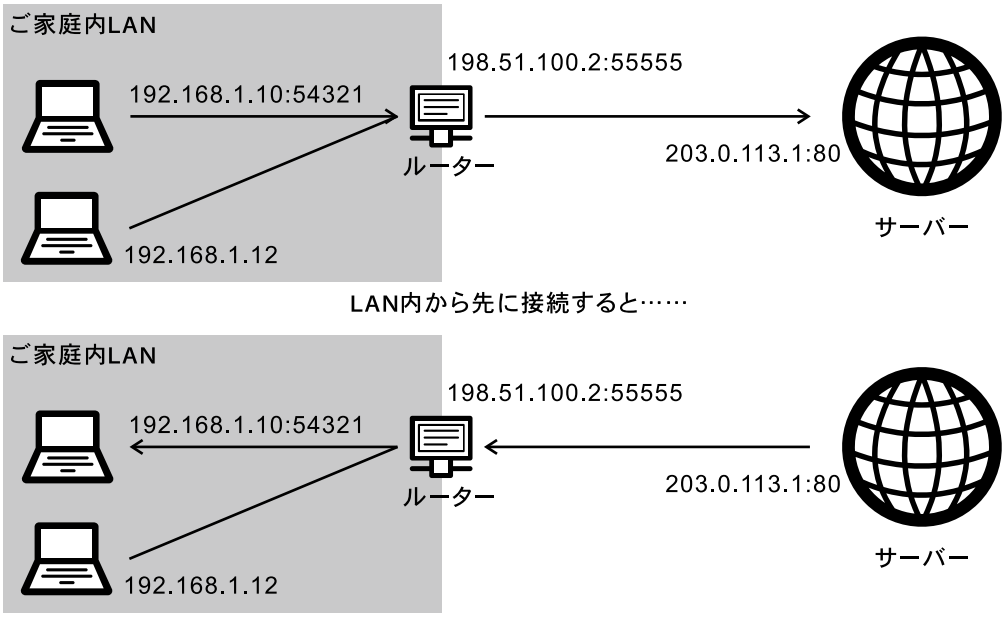


図 1. NAT で LAN 内からサーバーに接続

複数のマシンが同時にアクセスすることも当然あるでしょう。上の例と同時に `192.168.1.12:54321` から `203.0.113.1:80` にアクセスしようとすると、今度は `198.51.100.2:55555` から `203.0.113.1:80` に接続しまーすとルーターがパケットを書き換えて転送します。さらに `198.51.100.2:55556` に返事が来たら `192.168.1.12:54321` に転送してやるよう記憶しておきます[7]。Web サーバーからの返事が同時に来ても `198.51.100.2:55555` に来た返事は `192.168.1.10:54321` へ、`198.51.100.2:55556` に来た返事は `192.168.1.12:54321` へと覚えていたプライベートアドレスに転送することで上手いこと同時の通信もできるのです。

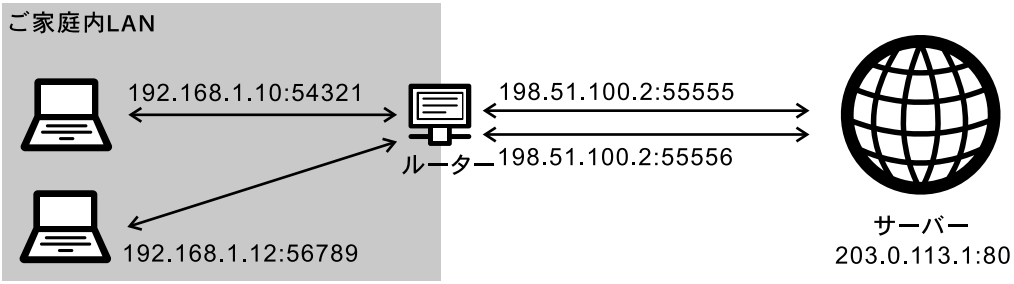


図 2. NAT で LAN 内からサーバーに複数接続

これが NAT の動作です。

LAN 内からインターネット上の Web サーバーに接続しにいく場合は上手くいきましたが、P2P

7　この返事を受け取るアドレスは無限には覚えていられないのでルーターの実装にもよりますが 1024 とか 4096 個の数が上限になっています。また、ある程度の時間が経過すると忘れます。

だとインターネット上のマシンから接続しに来るのが発生しましょう。

インターネット上の 203.0.113.42:56789 から 198.51.100.2:7144 に接続されてきたとしましょう。 198.51.100.2 はルーターに割り当てられているグローバルアドレスですが、ルーターで P2P アプリケーションが動いてるわけではないので、本当に接続したいのは LAN 内にある 5 台のうちどれかです。さあどれでしょうか。わかりませんね。おしめえだ。

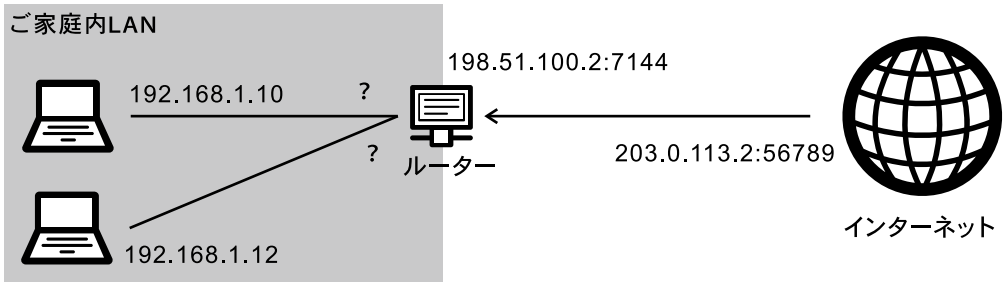


図 3. インターネットから先に LAN 内に接続

というわけで NAT があると P2P ができません。え、これでおわり？と思われるかもしれませんが終わりです。接続できません。

LAN 内からインターネットに接続しに行く時には返事を貰うためのプライベートアドレスが事前に分かったのでルーターで記憶しておくことができましたが、インターネットから接続しに来る時にはどこに接続するべきかが事前にわからないので自動的に接続することができないのです。

5. NAT 越え

でもなんとかして接続したい！ということでいろんな方法が編み出されます。これらの方法を NAT 越えと呼びます。

事前に LAN 内の接続先が分からないのが問題なので、逆に言えば事前に接続先をなんとかしてわからせれば接続できそうですね。実際その方法で接続できるのですが、事前に分からせる方法 (ポート開放と言うことが多い) がいくつかあります。

1 つは手動で設定です。

ルーターにはインターネット側からの接続をどのプライベートアドレスに転送するかを手動で設定できるインターフェースが付いています。物によってはないかもしれませんけど、まあたぶんだいたいあると思います。以下はうちのルーターの設定画面です。

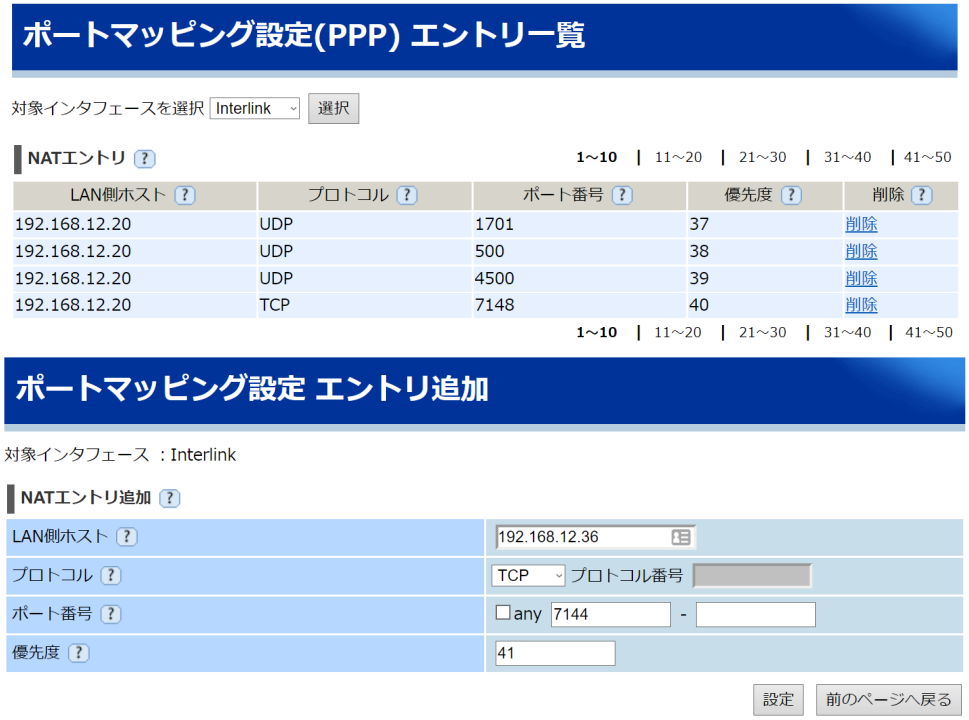


図 4. ルーターのポートマッピング設定

手動で設定は確実なんですけど、みんなこれやっとけよ！というのはちょっとハードルが高いのが問題です。ルーターの設定画面への入り方を知らない人とか忘れている人も多いですし、なんとか設定画面に辿り着いたとしてもルーター毎に設定画面の作りが違うので、この手順通りにやれば OK ！と説明できないのです。みんながどんなルーター使ってるかわかりませんし。

できることなら自動でやってくれるとうれしいですね。ということでいくつか自動でやる方法があります。その 1 つが UPnP(Universal Plug and Play) です。

UPnP はネットワーク経由でいろんな機器を制御するためのプロトコル標準です。いろんな機器ってのはルーターだったり HDD レコーダーだったりテレビだったりプリンターとかスキャナーとかいろいろあるんですが、ルーターを制御するプロトコルとして耳にすることが多いでしょう。[8]

ルーターが UPnP に対応していると、手動で設定するのと同様に UPnP でこのポート番号に接続に来たらこのプライベートアドレスのこのポート番号に転送してくれと設定できます。UPnP は設定方法が標準で決まっていますので、手動で設定するのと違ってルーターによって手順が違うとかいうのはありません。ルーターの設定画面への入り方がわからなくても使えます。ただ UPnP でルーターの設定は人が手動でやるのは一般的ではなく、そういうアプリケーションはなかなか見つかりません。P2P を使うアプリケーションが必要な時に自動的に設定するように実装するのが一

8　HDD レコーダーやテレビをネットワーク経由で操作したり動画再生できる DLNA とかいうのも UPnP を基にした機能なんですが、あんまり UPnP とは言わないですね。UPnP で操作できるものも UPnP と違った名前で呼ばれることが多いので、UPnP と言えばルーターを操作するものと広く知れ渡ってしまっています。いや本当はルーター以外にもいろいろあるんだよ。

般的です。勝手にやってくれるのは便利と言えば便利なんですが、上手くいかなかった時にちゃんと設定されているのかどうか分かりづらいのが難点ですね。

もう一つは NAT-PMP です。これも UPnP と同じく自動的にポート開放をするためのプロトコルです。

UPnP との違いは、ポート開放専用のプロトコルであることです。UPnP はいろんな機器を制御するプロトコルであり、そのうちの一つとしてルーターがあるのですが、ポート開放するだけにはいらない機能も実装しないといけないので面倒です。NAT-PMP はポート開放専用なので、UPnP に比べて簡単に実装できます。

が、NAT-PMP を実装しているルーターは UPnP に比べて少なく、特に日本ではほとんど見かけません。日本では UPnP が事実上標準でしょう。

ということで、UPnP を使えば P2P もつながるようになるんだね！と言いたいところですが上手くいかないことがあるのです。

まず UPnP の実装が標準と違っていたり、ひどい時にはバグってて使えないルーターがあることです。UPnP でポート開放はもう何年も昔から使われているものなので、いまさらバグってて使えないということはないだろう、と思っていたのですが、2019 年に発売された Buffalo のルーターで UPnP がバグってて事実上使えないという報告がありました。安いルーターほんと信用ならねえな……。これはもう仕方ないので手動で設定してもらうしかないですね。それかそんなルーター窓から投げ捨ててまともなのを買い直すか。ただ一番の問題はルーターがバグってるって気付くのが難しいことなんですが。

他に UPnP に対応していないルーターがあったり、UPnP が無効になっていることがあったりします。2019 年現在では対応していないルーターはあまり無いとは思うんですが、ユーザーが UPnP を無効にすることは大抵のルーターでできますのでその場合は動きません。まあご家庭で使っているルーターでは既定で有効なものがほとんどでしょうし、わざわざ無効にできる人は設定画面に入って設定をいじれる人なので自分でなんとかできるでしょう。ただ会社や (たとえばホテルとかの) 共有の回線では無効になっていることがほとんどでしょう。そんなところで P2P とかやるなよと言う話もありますが、まあなんやかんやでやりたいこともあるでしょう。この場合はセキュリティなどの問題で無効になっているでしょうから難しいです。会社でどうしても必要なら管理者に言って設定してもらえばいいですが、共有回線だとそれも相当難しいでしょう。

さらに難しいのが大規模 NAT が使われている場合です。これも UPnP が無効になっているルーターと状況は近いのですが、まずそもそもご家庭のルーターにグローバルアドレスが割り当てられない場合があります。ご家庭のルーターに割り当てられるのがまずプライベートアドレスで、グローバルアドレスはご家庭じゃなくて ISP が管理するルーターにしか割り当てられないのです。なんでこんなことするかというと、IPv4 のグローバルアドレスは数が少ないので普通はご家庭に 1 個しか割り当てられないという話でしたが、最近は各ご家庭に 1 個すら難しくなったので、いくつかのご家庭に対して 1 個のグローバルアドレスを割り当てるようにしました。で、ISP 側のルーターで、いくつかのご家庭を LAN 内に入れたような形で扱います (まあちゃんと分離はしますが)。この状

況は CATV だとかマンションに標準装備の回線だとかでたまにありますが、最近だと IPv4 over IPv6 でよく見られるようになりました。さらにモバイル回線もこれになります。これらの場合はルーターが ISP の管理下にありますし、UPnP なんか無効ですし (というか ISP のルーターはたぶん対応していない)、手動で設定してと言っても大抵お断りされるでしょう。

UPnP も手動での設定もできない場合はもうお手上げですね……と言いたいところですが、まだ頑張る方法があります。それが UDP ホールパンチングです。

プライベートアドレスとグローバルアドレスの関連付けができればインターネット側からの受信ができるのですが、確実に自動的に関連付けができるのは LAN 内からインターネットに通信を始めたときです。さらに一度関連付けがされれば一定時間は記憶されています。もしかして、一度 LAN 内からインターネットに通信すればそのあとしばらくはインターネット側からの接続も転送してくれるのでは……？ということをやってみるのが UDP ホールパンチングです。

これが実際動いたりします。ただこれも確実ではなく、いろいろと動かない場合もあります。

TCP では動かない可能性が高いです。TCP だと接続が確立したかどうかというのをちゃんと管理できるので、まず正しく接続が確立できなければ関連付けを諦めるということをルーターがやったりします。UDP だと接続の確立とかないので成功しやすいのです。

最初に LAN 内から通信を試みないといけないというのも難しいです。インターネット側からの接続を受け付けたいのですが、こちらから通信を開始しないと接続が受け付けられません。いつインターネット側から接続してくるのか、そのタイミングがわからなければこちらから通信もできないですね。というわけで今から誰かが接続しにくるよ、というのを別途通信で受け取らないといけないのです。

6. IPv6 でつながらない P2P

IPv4 ではプライベートアドレスしかもらえないことが多いので P2P がなかなかつながりませんでした。一方で IPv6 は全マシンにグローバルアドレスを割り当てることができますし、実際に割り当てられるので簡単に P2P が実現できます！建前ではな！！

実際にはご家庭で IPv6 を使うとほぼ確実にインターネットからの接続着信を受け付けません。こちらから IPv6 でサーバーなどに接続しに行くのはできるのですが、IPv6 で接続待ち受けをしても外からはつながりません。なぜかというとご家庭のルーターは IPv6 での接続着信を全部拒否することになっているためです。P2P 全否定です。なんてことを！

なんでこんなむごい仕打ちをするのかというとセキュリティ確保ということになっています。IPv4 ではグローバル IP を直接割り振ることは (あんまり) なかったので NAT を使っていることは前に書きましたが、結果的に余計な接続待ち受けをインターネット側に晒さずに済み、LAN 内からしかアクセスすべきでないサービスについうっかりインターネットからアクセスできるようになっていた！というのが起きづらい状況でした。それを意図して NAT を使ってたわけじゃないんですが、結果的にね。

しかし IPv6 だと接続されている全マシンの接続待ち受けがインターネットに晒されてしまいま

す。もちろん個々のマシンのファイアウォール的な機能 (Windows Defender とか) によって接続可否を判定することは可能ですが、一般ご家庭の全マシンでそれが正しく設定されていると期待するのはちょっと楽観的すぎます。なので安全側に倒しといた方がいいだろうということで、ご家庭の IPv6 対応ルーターはインターネット側からの接続着信を既定で全部拒否しろや！というガイドラインが作られました。たぶん 2019 年現在売ってるご家庭用のルーターのほとんどはこのガイドラインに従っていると思われます。

ガイドラインではユーザーが明示的に接続着信をできるようにはすべき、とはなっているので、手動で設定はできるようになっていると思います。ただ、その設定方法がものすごくわかりづらくそしてマニュアルにもろくに何も書いていないルーターも多く、手動設定は非常にハードルが高いのが現状です。噂によると設定が出来ないルーターもあるとかないとか……？

以下に例として家で使っているルーターの IPv6 パケットフィルターの設定画面を挙げます。

図 5. IPv6 パケットフィルター一覧

いくつか記述がありますが、自分で追加したものですので初期状態では空でした。空でしたが、何の説明もなく外部からの接続着信を一切受けつけませんでした。ドキュメントにもそのことは書いてありませんでした。まじかよ。

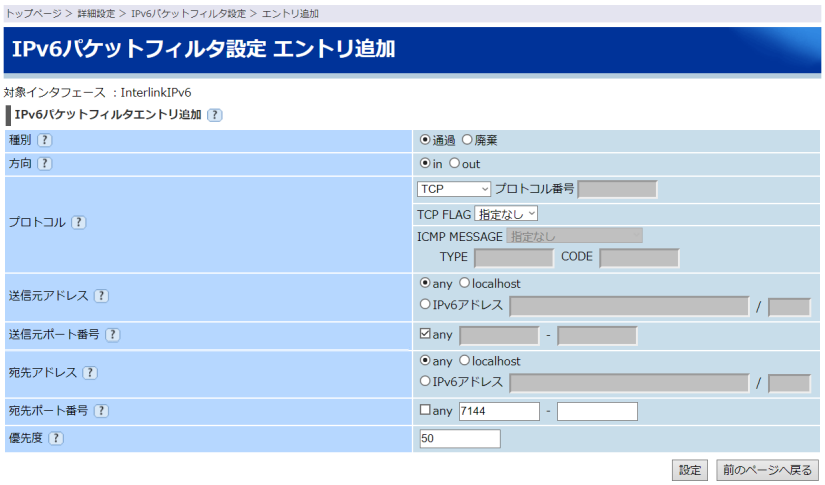


図 6. IPv6 パケットフィルター追加画面

追加しようとするとこんな画面になります。最低限設定できるようにはしたよっていうだけです。

これをみんなに適切に設定させるのは至難でしょう。

そんな感じで IPv6 で簡単になるはずの P2P 通信は、ご家庭ではなぜか IPv4 よりハードルが高いものとなっています。

7. ファイアウォール越え

手動でやるのが大変なら UPnP などでプログラム的に着信を許可できないのでしょうか？その方法はいくつかあります。

まず UPnP。IPv4 だと UPnP に対応したルーターに対してグローバルアドレスのポートと指定したプライベートアドレスのポートを関連付けて着信許可するように指定できました。IPv6 では UPnP でルーターの IPv6 ファイアウォールを操作できるようになっています。この IPv6 アドレスのこのポートへの着信を許可してください、と指定することができます。が、しかし……この WANIPv6FirewallControl という機能を実装したルーターはありません！いや、ほんとはあるのかもしれないけど、少なくともご家庭のルーターでこれを実装したものを見かけないですし、使えたという情報もないです[9]。つまり仕様としてはありますが、現状では事実上使えません。

Port Control Protocol (PCP) というのもあります。これは IPv4 でやっぱりポートマッピングに使われていた NAT-PMP というプロトコルの IPv6 拡張版みたいなものです。こちらは着信許可を制御するという点に絞ったプロトコルで UPnP よりシンプルになっています。ご家庭用の IPv6 対応ルーターのガイドラインの一つである RFC 7084 では、ルーターは PCP を実装して着信許可を出せるようにすべき、と書いてあります。が、しかし……！やっぱりこの PCP を実装したルーターは見たことがありません。もしかしたら主に海外製のルーターだと実装されているのかもしれませんが[10]、日本のご家庭で広く使われているルーターでは実装されていないようです。使えないじゃーん。

さて、日本で売ってるご家庭のルーターだと手動で着信許可するのも難しいし自動で許可もできないし、いったいどうしたらいいんでしょうか？答えは簡単！

どうしようもないです……。いつか簡単につなげる日が来ることを夢見て待ちましょう。

おしまい。

と、言いたいところですが、UDP ホールパンチングなんかは動くようです。

ガイドラインとしては UDP ホールパンチングが動くような実装にすべき、となっていますし実際に多くの環境で動くらしいです。まあ実装によるので必ずしも動くとは限らないんですが、NAT がない分 IPv4 よりは成功する可能性は高そうです。UDP ホールパンチングはこちらから通信開始しないといけないので、意図せずうっかり外から接続されることがないので大丈夫ということなん

9　pfSense あたりでオプションを有効化すると使えそうという情報は貰ったことがありますが、それでもオプションのようです。さらに pfSense はご家庭のルーターとして一般的とは全く言えないでしょう。

10　PCP の元となった IPv4 版の NAT-PMP は、日本では AirMac 等、主に海外で売ってるルーターの日本向け製品くらいでしか実装されてませんでした。

でしょうね。

TCP ？ TCP は諦めよう。

8. 参考文献

適当に書き散らしただけなのであんまり無いんですが……。

8.1. IPv6 普及・高度化推進協議会 IPv6 家庭用ルータ SWG

http://v6pc.jp/jp/wg/coexistenceWG/v6hgw-swg.phtml

IPv6 の家庭用ルーターのガイドラインが置いてあります。日本のご家庭で使われている IPv6 対応ルーターの多くはたぶんこのガイドラインに従った動作をするんじゃないかと思われます。

8.2. Basic Requirements for IPv6 Customer Edge Routers (RFC 7084)

https://tools.ietf.org/html/rfc7084

これもご家庭の IPv6 対応ルーターのガイドラインです。これは日本に限らないガイドラインなので、海外メーカー製のルーターがこれに沿っている可能性があります。

日本のガイドラインとは似ているもののいくらか違う点があるので、日本のご家庭用ルーターがこれに沿っていることはあまり期待できません。PCP を実装すべきとか書いてあるし。

CS:GO サーバー改造事始め

著 : ぷろぐれ

Counter-Strike シリーズは 2019 年でリリースから 20 年を迎えたチーム対戦型 FPS です。Counter-Strike シリーズは mod (改造) がとても盛んです。単にゲーム中の演出を強化するものから、全く別のゲームにするものまで多種多様です。例えば「キルを重ねたときに Killing Spree の効果音と表示をする」、「感染によって増えるゾンビから逃げ続ける (ゾンビエスケープ)」といったものは mod をサーバーに導入する事で実現されます (但しゾンビエスケープなどはマップに組み込まれた複雑なスクリプトとの組み合わせで実現されます。マップに組み込むスクリプトはここでは扱いません)。 ここでは Counter-Strike シリーズの最新作である Counter-Strike: Global Offensive (以下 CS:GO) での mod サーバーの作り方を紹介します。ちなみに CS:GO は 2018 年 12 月に無料化しているので安心して読み進めてください。

CS:GO は Source Engine というゲームエンジンで作られています。 Source Engine は、1998 年に開発された GoldSrc エンジンの後継として 2004 年に Half-Life 2 に初めて採用され、以降バージョンアップを重ねながら様々なゲームに採用されています。 2018 年にリリースされた Apex Legends も Source Engine が採用されています (ただし 2019 年 9 月現在はカスタムサーバーを建てることができないので当記事の技術を使うことはできません。残念)。 CS:GO は 2012 年に最初のバージョンがリリースされました。CS:GO では当時の Source Engine のバージョンをベースとして独自のカスタマイズを含めていますが、ベースはどのゲームも同じです。 GoldSrc エンジンの頃から mod の開発は活発で、Source Engine でもその流れを引き継いでいます。 Source Engine には addon という mod のための仕組みが最初から用意されています。 addon はローレベルの仕組みなので、ほとんどの mod は addon を直接実装せず Metamod:Source と SourceMod を用いて、その plugin として実装されています。 Metamod:Source は addon を直接実装するときのいくつかの問題を解決する addon です。 SourceMod は SourcePawn という独自の言語で実装した plugin を読み込むことで、比較的簡単にサーバーを改造することができる addon です。 SourceMod は Metamod:Source に依存しています。 addon を直接実装しないための別の addon として Source.Python があります。 Source.Python も SourceMod と同様に追加で plugin を読み込んでサーバーを改造することができる addon です。 Source.Python は python(3.6) で mod を実装できるので、大規模な mod を実装する場合は表現力の乏しい SourcePawn を用いる SourceMod よりも遥かに楽に実装ができます。

1. サーバー構築

CS:GO は過去のシリーズと違い公式のサーバーが用意されていますが、過去のシリーズと同様に自前でサーバーを建てることもできます。 mod サーバーを作るには自前のサーバーを用意する必要があります。以降、環境は Amazon Linux2 または Ubuntu (18.04 LTS) を想定し、基本的な

Linux 知識がある前提で書かれています。

サーバーのインストール作業は以下の通りです。 /opt/steam/csgo-ds/ 以下に CS:GO をインストールします。

```
$ WORK_DIR=/opt/steam

$ sudo useradd -m steam                                  # サーバー用ユーザー作成
$ sudo mkdir --parents $WORK_DIR                         # インストール場所作成
$ sudo chown steam $WORK_DIR

### 依存ライブラリのインストール
### ↓ Amazon Linux2 の場合
$ sudo yum install -y glibc.i686 libstdc++.i686
$ sudo yum install -y zlib.i686 libffi.i686        # source.python
$ sudo yum install -y python3
### ↓ Ubuntu の場合
$ sudo apt install -y lib32gcc1 lib32stdc++6
$ sudo apt install -y unzip
$ sudo apt install -y python3-distutils
$ sudo curl -kL https://bootstrap.pypa.io/get-pip.py | sudo python3
$ sudo dpkg --add-architecture i386
$ sudo apt-get install -y zlib1g:i386 libffi6:i386 # source.python

### CUI 版 Steam のインストール
$ cd $WORK_DIR
$ curl -sqL "https://steamcdn-a.akamaihd.net/client/installer/steamcmd_linux.
tar.gz" | tar zxvf -

### Steam で CS:GO をインストール
$ ./steamcmd.sh +login anonymous +force_install_dir ./csgo-ds +app_update 740
+quit

### ↓ Ubuntu で steamclient.so のエラーが起きる場合に実行してください
$ mkdir -p /home/steam/.steam/sdk32/
$ ln -s $WORK_DIR/linux32/steamclient.so /home/steam/.steam/sdk32/
```

ここまででひとまず標準サーバーのインストールは完了です。

続いて以下の場所にファイルを作成します。

/opt/steam/csgo-ds/csgo/cfg/server.cfg

```
hostname サーバー名
sv_lan 0 // LAN内に限定する場合は1にする
sv_region 4 // Asia
```

そしてサーバーの起動のために、Steam のサイトから以下の 2 つのトークンを発行してください。

- **AUTH_API_KEY**

Steam コミュニティ :: Steam Web API キー : https://steamcommunity.com/dev/apikey

- **LOGIN_TOKEN**
  Steam コミュニティ :: Steam ゲームサーバーのアカウント管理 : https://steamcommunity.com/dev/managegameservers

トークンを発行したら、以下のコマンドでサーバーが起動します。それぞれのトークンは発行されたものに置き換えてください。尚、ファイヤーウォールが有効な場合は UDP の 27015 番からのアクセスを許可しておいてください。

```
$ WORK_DIR=/opt/steam

$ LD_LIBRARY_PATH=$WORK_DIR/csgo-ds:$WORK_DIR/csgo-ds/bin \
  $WORK_DIR/csgo-ds/srcds_linux -game csgo -usercon \
  -authkey AUTH_API_KEY \
  +sv_setsteamaccount LOGIN_TOKEN \
  +map de_dust
```

サーバーを起動したら、ゲームを起動しモードを「OFFICIAL MATCHMAKING」から「Community Server Browser」に変更して右下の「サーバーの追加」から IP を入力すると、あなたのサーバーが表示されるはずです。

Source Engine にはゲームやサーバーの振る舞いを変える convar (設定値) が大量に公開されています。これでもってサーバー名などの基本的な設定や、あるいはこれをいじるだけでも変なサーバーを作ることができます。 例えばサーバーのコンソールで以下の通りに入力すると、ゲーム内の重力を変更できます。

```
sv_gravity 200
```

ゲームに戻ると重力がものすごく軽くなったでしょう。こういった convar は server.cfg に記述しておくことでも効果を発揮します。 convar の一覧は Valve の開発者サイトで公開されています。ただし一部情報が古いので有志がまとめたものを参照するのも良いでしょう。「`csgo-list-of-cvars`」で検索すると出てきます。

2. addon の導入

引き続き、addon をインストールしていきます。以下では記述時の最新版の URL を記載していますが、各 addon のサイトで最新版を確認してください。

- Metamod:Source : https://www.sourcemm.net/
- SourceMod : https://www.sourcemod.net/
- Source-Python : https://forums.sourcepython.com/

```
$ WORK_DIR=/opt/steam

$ cd $WORK_DIR/csgo-ds/csgo/
### Metamod:Source
$ curl https://mms.alliedmods.net/mmsdrop/1.10/mmsource-1.10.7-git970-linux.tar.
gz | tar zx
### SourceMod
$ curl https://sm.alliedmods.net/smdrop/1.9/sourcemod-1.9.0-git6281-linux.tar.gz
| tar zx
### Source-Python
$ curl -o tmp.zip -O http://downloads.sourcepython.com/release/691/source-
python-csgo-July-07-2019.zip
$ unzip tmp.zip
$ rm tmp.zip

### SourceMod にはサンプル plugin が入っているのですべて無効化する
cd $WORK_DIR/csgo-ds/csgo/addons/sourcemod/plugins/
mv *.smx disabled/
```

先に記した通り公開されている殆どの mod は SourceMod の plugin 形式で実装されています。SourceMod のサイト内では公開されている plugin の検索ができます。ここでは以下のダメージ量を表示する plugin を導入します。

[CSGO] Fortnite like damage showing [1.2.0] - AlliedModders : https://forums.alliedmods.net/showthread.php?p=2604384

zip を解凍すると csgo/ 以下と同じ構成になっているはずですので、その構成のままコピーしてください。コピーしたら導入は完了ですのでサーバーを起動してみてください。弾を当てた時にエフェクトが発生するはずです。

3. 自分だけの mod を作ってみよう

ではいよいよ mod を作ります。ここでは Source-Python の plugin を作ります。まず以下のディレクトリを確認してください。

```
/opt/steam/csgo-ds/csgo/addons/source-python/plugins/
```

ここに、適当な名前を決めてディレクトリとファイルを作成します。ディレクトリとファイル名は同じ名前にしてください。

```
/opt/steam/csgo-ds/csgo/addons/source-python/plugins/myplugin/myplugin.py
```

ファイルを作ったら、2 つのメソッドを定義します。これらは plugin の開始時、終了時に呼ばれます。

```
def load():
    print('Plugin has been loaded successfully!')


def unload():
    print('Plugin has been unloaded successfully!')
```

そして server.cfg に以下を追記してください。

```
sp plugin reload myplugin
```

これでサーバー起動時に `Plugin has been loaded successfully!` と表示されれば成功です。

次はチャットの入力で音が鳴るというものを作ります。 Source Engine では mp3 形式か wav 形式の音を扱うことができます。適当な mp3 ファイルを用意して、`/opt/steam/csgo-ds/csgo/sound/myplugin/chat.mp3` に保存してください。用意したら `myplugin.py` に以下を追記してください。

```
from typing import Dict
from engines.sound import Sound
from events import Event

chat_sound = Sound('myplugin/chat.mp3', download=True)

@Event('player_say')
def player_say(game_event: Dict):
    sound.play()
```

`download=True` はプレイヤーがサーバーに入ろうとするときに当該のファイルがあるかを確認し、無い場合はサーバーからダウンロードするように指定しています。

これでゲーム内でチャットを使うと音が鳴るようになります。

`@Event('player_say')` は CS:GO における `player_say` イベントが発生したときに装飾したメソッドを動作させるという意味になります。また、`game_event` にはこのイベントの引数が入っており、この `player_say` イベントには `text` `userid` が入っています。 例えばメソッドの中身を以下のように書き換えると、特定のチャットを行った場合にのみ音を鳴らすことができます。

```
@Event('player_say')
def player_say(game_event: Dict):
    if game_event['text'] == 'gappoi':
        sound.play()
```

あるいは以下のように書き換えると、特定のユーザーにのみ音を鳴らすことができます。

```
from players.entity import Player

@Event('player_say')
def player_say(game_event: Dict):
    player = Player.from_userid(userid)
    if player.name == '大佐':
        sound.play()
```

この他にも様々なイベントやオブジェクトがあるので、それらを組み合わせて mod を作ることになります。以下で完全なリストを確認できます。

- Source.Python documentation — Source.Python v689 documentation : http://wiki.sourcepython.com/
- Counter-Strike: Global Offensive — Source.Python v689 documentation : http://wiki.sourcepython.com/developing/events/csgo.html

後はドキュメントを眺めながらアイディアを形にしていくことになります。どのようなサーバーを作るかはあなた次第です。 既存のサーバーを参考にするのも良いかもせれません。 CS:GO を起動してゲームモードを「OFFICIAL GAME MATCHING」から一番下の「COMMUNITY SERVER BROWSER」にし、地域を「アジア」(重要) に変更するとアジア圏のサーバーが表示されますのでその中から mod サーバーっぽいサーバーを探してください。2019 年現在は「はかた鯖」や「planetia」が、どちらも常設ではありませんが一癖ある mod サーバーとして稼働しています。ぜひ一度サーバーに接続し遊んでみてください。そしてあなたも mod サーバーを作ってみましょう！

VRC 民に効く (かもしれない) Unity 操作 Tips まとめ

著 : ぬか

VRChat を始めて Unity を触り始めた方が多いと思う。そこで基本的操作から、VRchat のアバターやワールド制作等にも関連しそうなもう一歩くらい進んだ Unity の操作をまとめる。

1. カメラ

- **右クリック +WASD**
  カメラ移動　QE で上下 さらに Shift で高速移動
- **右クリック＋ Alt ＋ドラッグ**
  カメラをゆっくりとズーム
- **Alt ＋ドラッグ**
  対象を中心にカメラを回せる
- **Shift ＋ Ctrl ＋ F**
  対象を現在の Scene のカメラの向きにセット（※画像 1）
  Camera に使うとシーンビューで見ている向きの通りに配置される。

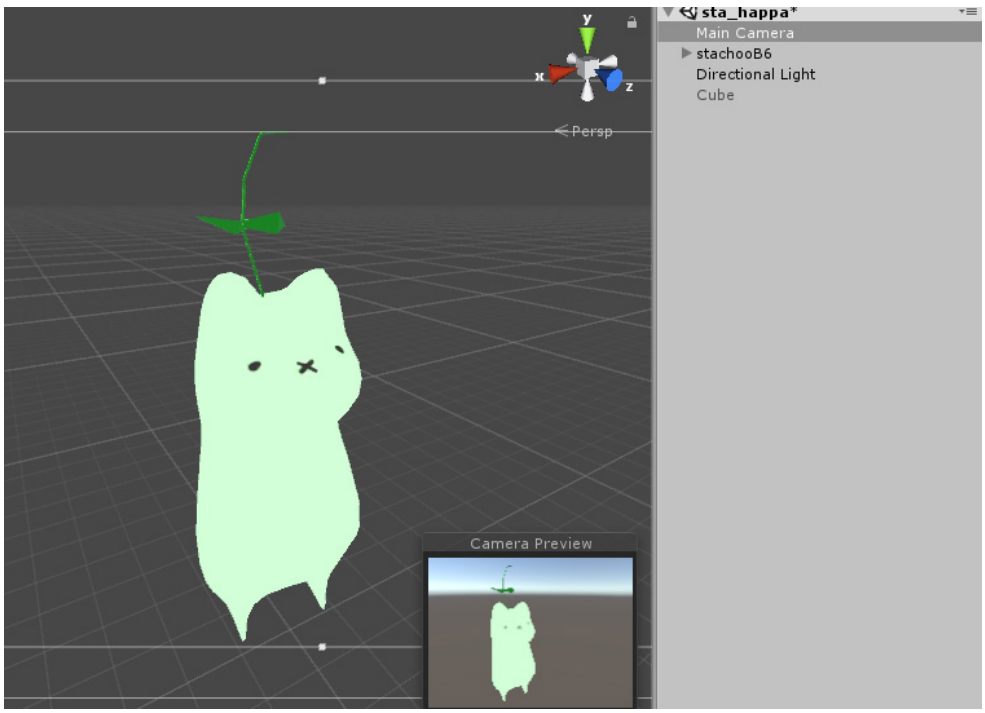


図 1. 画像 1

2. エディタ

- **Hierarchy やインスペクターのタブを増やす**

  メニューから Add Tab でタブをつくり、エディタ上の好きな場所にセット。鍵のアイコンでロックも可。(※画像 2) ロックしないと互いのタブに干渉してしまうので活用していきたい。

  タブを切り離すとマルチディスプレイ等にも効率的。

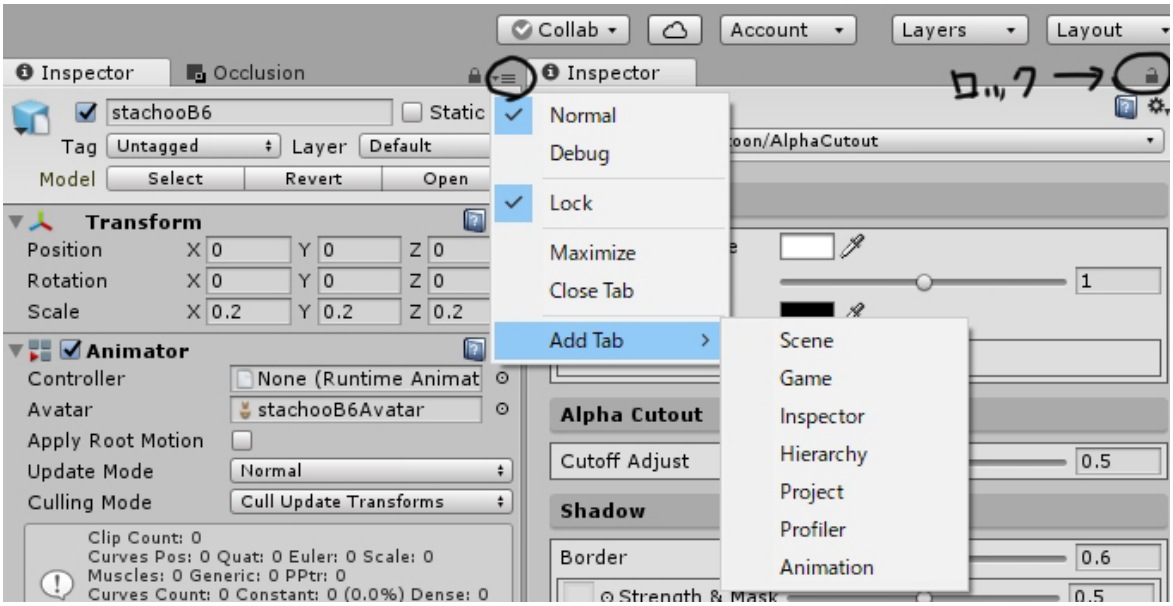


図 2. 画像 2

- **Hierarchy の検索欄**

  ファイルの種類やコンポーネント名を入力でアタッチされたオブジェクトを検索してくれる。さらに☆印をクリックで Favorites 入りし、検索状態を保存できる。ここの検索欄の消し残しで Hierarchy にあるはずのものが消えて一瞬パニクったりするのあるある。

- **オブジェクトの階層を全表示**

  Alt ＋クリック、階層構造を全展開してくれる。アーマチュアとかにどうぞ。(※画像 3)

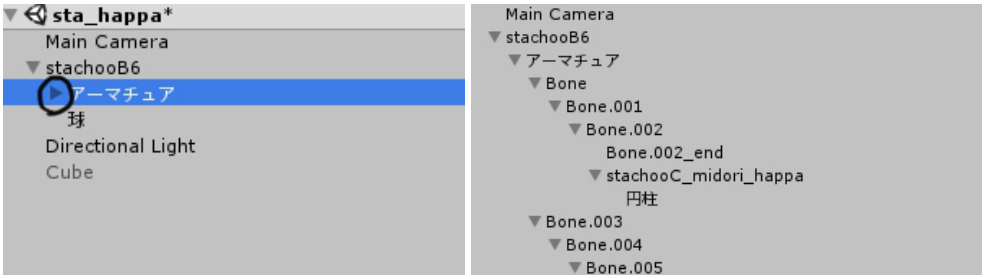


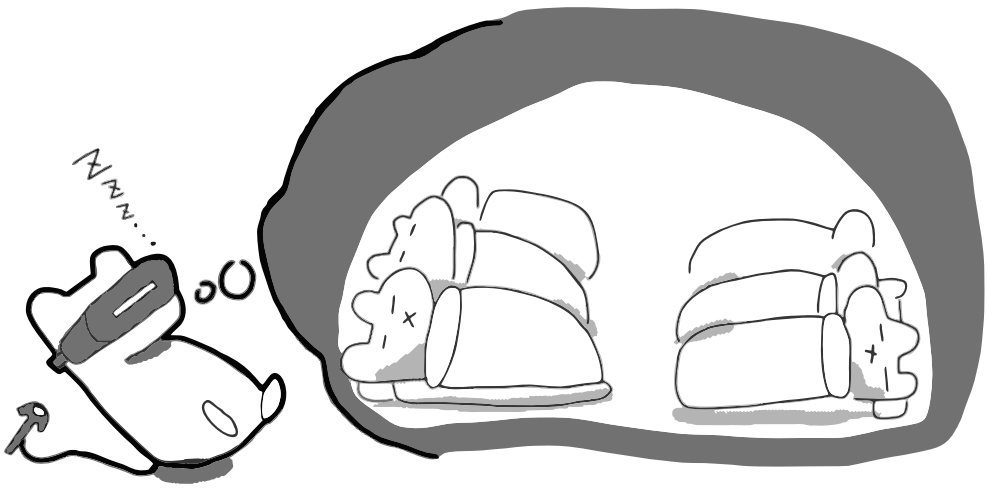
図 3. 画像 3

3. プロジェクト

- **Project ビューから全ファイル閲覧、開ける**
  ファイルやフォルダ、ショートカットなど表示できるのでここに置いたファイルやフォルダを開くことができる。ここから PSD ファイル等の編集でアバターに反映しつつ作業。
- **0_My フォルダ**
  これはただのお節介かもだけども、いろんな Assets やアバターのインポートやら scene やらで project の構成がめちゃくちゃになりがちな人なんかは、目立つように自分のフォルダを作りましょう。

構成の仕方は人それぞれですが、Assets 直下に自身のフォルダを作ってそこを作業フォルダに展開するのが結局個人規模ではいいんじゃないかなあ。外部アセットの導入でエラーや誤作動を防ぐためにそれらの階層にはタッチしない思想。

かといって Scenes の場合はフォルダを作ってそこに全部の Scene ファイルを入れずに各アバターのフォルダの中に入れとくのが好き。ちなみに Quest 対応用に Android に都度切り替えをしていくくらいなら project 自体を別にするのがセオリーっぽい。



Misreading Chat リスニングガイド

著 :vivivi

1. 序文

「Misreading Chat」という CS（コンピュータサイエンス）の論文を読んで解説するポッドキャストが面白いので紹介します。

https://misreading.chat/ Misreading Chat - CS の論文読んで話をしよう

2018 年 3 月から途中何度か休暇も挟み、現在も週 1 ペースで更新されています。

技術系ポッドキャストは時事ネタや雑談系が多いですが、このポッドキャストでは論文を噛み砕くことに主眼を置いています。もちろん全部聴くのがおすすめなのですが、回数も多くなってきたので気になったものを分類しメモ書きをしてみようと思います。

2. 機械学習

#1a Tensor Comprehensions

Facebook が出した TensorComprehensions というフレームワークに依存しない機械学習の高性能な抽象についての説明です。お前は何を言ってるんだ、って感じですがいきなり初回でついていけるか不安になる題目です。簡単に言うと機械学習は単純な計算を膨大にやる事が多いのでそれを簡単に書けるようにしたり速くしたりってことらしいです。こういうのは実際自分で書いてみないとよくわからないですね。

#8 Generative Adversarial Nets

いわゆる GANs と呼ばれてるやつの説明です。ジェネレータとディスクリミネータを使いながら徐々に賢く画像を生成していくアレです。ジェネレータがディスクリミネータを騙せるような画像を生成できれば勝ちです。しかし機械学習はなぜそうなるのかがわからんがうまくいくから OK!、というものが多くてむず痒いですね。

#9 Automatic Differentiation in Machine Learning: a Survey

自動微分の論文です。ってなんやねんてのは説明してくれます。ようするに適当な数値から始めて計算してその傾きから少しずつそれらしい値に近づけていくものです。この回はサーベイ（調

査）論文で、機械学習の説明などの話になるのでかなりわかりやすい回になってます。

#15 Neural Machine Translation by Jointly Learning to Align and Translate

Attention を使って機械翻訳する論文回です。#15,51,53 で自然言語処理機械学習 3 部作になってます。Misreading 屈指の難解回で RNN や Attention を知らなければ全くわからないと思います。RNN を事前に予習してから聞くことをお勧めします。しかし機械学習はこういった難しいことをわからなくても使えてしまうのが面白いとも言えますね。

#16 A Deep Learning Approach for Generalized Speech Animation

ディズニー映画の口パクの部分を機械学習で作るという論文です。昔からこういうことやってるのかと思ったら意外に新しい論文なのでまだ新しい分野なのでしょう。バリバリ機械学習だぜ、っという論文ではないので聞きやすくオススメです。

#46 An Introduction to Neural Information Retrieval

機械学習で検索をやる入門の論文です。検索文字列から検索対象の文字列をさがして持ってくるというフェーズと、持ってきた検索対象をランク付けして出すというフェーズに分かれてるので、ランク付けのフェーズを機械学習でいい感じにしてあげると良い検索になるということです。ランク付けは昔からいろいろな手法があるので、機械学習しないランク付けから DL するランク付けまで説明してくれます。

#51 Attention Is All You Need

#15 で最後の方に出てきた Transformer の話です。Transformer とは RNN を過去のものにした Attention ベースの自然言語処理モデルです。中で森田さんが「全くわからなかった」というくらい難解でしたね・・・。当然難解回なので予習する、もしくは解説記事 (http://deeplearning.hatenablog.com/entry/transformer) を読む（これも難解だが）もしくは聞き流すのがよいでしょう。

#53 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Google が作った双方向 Transformer の説明です。超難解回。自然言語処理機械学習 3 部作の最

後です。しかし向井さんが易しく説明してくれるのでわかった気にしてくれます。今の所ここが自然言語処理機械学習の最先端っぽいのでこれを聞いてればたぶん OK です。

#58 Gradient Boosting Decision Trees

kaggle でよく使われてるという GBDT についての話です。 kaggle とは一言でいうと機械学習でやる atCoder みたいな感じ。様々な機械学習アルゴリズムを用いて出される問題を解きます。例えば kaggle のチュートリアルはタイタニックの乗船情報から生存者を当てていく問題です。この回は GBDT を GB(Gradient Boosting) と DT(Decision Tree) に分けて説明してくれます。random forest や regularization の解説も後半にしてくれています。

#60 XGBoost: A Scalable Tree Boosting System

#58 で紹介された GBDT の実装である XGBoost の説明です。後半に LightGBM の話もしてます。ここでも話されてますが今は LightBGM や CatBoost なども出てきてるのでこの手のやつはすぐ時代遅れになりますね・・・。kaggle ネタはこれからもまた話に出てくるかもですね。

#65 Communication-Efficient Learning of Deep Networks from Decentralized Data

携帯端末でやる分散機械学習の話です。FederatedLearning について説明してくれます。FederatedLearning とはエッジデバイスで行った機械学習結果をサーバーで集約して新しく学習する技術です。各端末で一度学習させることによって個人データなどを使うけど個人データ自体は送信しなくてすむので個人情報保護の観点からも良いですね。各端末上のデータ量少ない状態で学習させてそんなのでうまくいくのか？と思われますがうまくいくのが不思議です。これがうまくいくのなら今後わりと面白いことができそうですね。この回から何回かで FederatedLearning シリーズです。

#67 Towards Federated Learning at Scale: System Design

#65 の続きです。FederatedLearning やってみた論文です。android に載ってるキーボードアプリ Gboard で分散機械学習してみた実例の話です。

#69 Bandana: Using Non-Volatile Memory for Storing Deep Learning Models

Facebook が出した論文で、NVM を機械学習で使う話です。NVM とは Non-VolatileMemory（不揮発性メモリ）のことで SSD などに使われてるメモリのことです。NVM は安いけど DRAM よりは遅いので細かい性能改善が必要なようです。大きいデータを扱う Facebook ならではの論文ですね。

#71 Machine Learning at Facebook: Understanding Inference at the Edge

Facebook が出した論文で、android 端末上で機械学習をやるけど辛いという話です。低スペック、低回線でもなんとかがんばるには？という話です。

#74 FaceNet: A Unified Embedding for Face Recognition and Clustering

顔認識の話です。顔認識は機械学習の得意分野なので話題が豊富ですね。その歴史から最新の技術や手法、改善などの話に及びます。量こそパワー！

3. カメラ、画像処理、GPU

森田さんはカメラ部門で働かれてるようで面白い話が多いです。

#2b GPU Raytracer(21:25 〜)

NVIDIA の OptiX というレイトレーサーの話です。ここではレイトレーシングの説明からしてくれます。GPU でレイトレーシングやパストレーシングする時代になりました。

#8 Generative Adversarial Nets

機械学習の項で。画像処理も機械学習と密接になりました。

#19 Light Field Rendering

ライトフィールドカメラという複数のレンズ、センサーを使って後から焦点を変えたりできるカメラについて話してます。現在は光学的ではなく機械学習でやってしまうのでこの方法は使われなくなってるようです・・・。 この回から ComputationalPhotography（計算写真学）のシリーズが始まります。

#21 The Frankencamera

フランケンカメラというスタンフォード大とノキアが作ったオープンソースの研究用デジタルカメラの話です。ここでの研究が Android のカメラの元になったようです。今の android との比較の話が出てきて面白いです。

#23 Halide: Decoupling Algorithms from Schedules for High-Performance Image Processing

Halide という高速画像処理用 DSL についての話です。カメラで使う画像処理は速さが必要なのでこういうものが必要なようです。

#25 Burst photography for high dynamic range and low-light imaging on mobile cameras

Pixelについてる HDR+ モードのアルゴリズムの話です。HDR についても詳しく説明してくれます。

#29 Image Analogies and Image Style Transfer

NN を使わない Style Transfer の話をしています。Style Transfer とは絵のスタイルだけを変えて再描写する技術です。昔はそれをどのように実現しているのかを説明してくれます。

#33 Distinctive Image Features from Scale-Invariant Keypoints

2004 年に書かれた 48,000 引用の画像から特徴量をとりだすアルゴリズムの論文の話です。画像の特徴とはなにか、それらはなににどういうふうに使われるのか、など画像の機械学習につながる重要な話をしてくれます。ここではパノラマに使うための（特徴点をもとに合成して合わせるような）特徴量の取り出しの話を前半にしています。後半は、その後どうなったかというところから Neural Best-Buddies という論文の話に移ります。ここで紹介されている Two Minute Papers という Youtube のチャンネルも面白いのでぜひ。なかなか興味深い論文なので直接あたるのがいいのではないでしょうか。

#35 Darkroom: Compiling High-Level Image Processing Code into Hardware Pipelines

画像処理用の DSL(Halide とか) を FPGA にかけるようにする Darkroom というものの説明です。もうひとつ、PixelVisualCore という Pixel に入っている画像処理用チップの話です。

#37 Synthetic Depth-of-Field with a Single-Camera Mobile Phone

Pixel に入ってる PortraitMode のアルゴリズムについての論文です。

#52 Convolutional Color Constancy

オートホワイトバランスのアルゴリズムについての論文の話です。Pixel に入ってるナイトモードで使われてるらしいです。

4. あとがき

他にも面白いエピソードはたくさんあります。#59 や #72 などはプログラマじゃない人が聴いても面白いと思います。1 話 30 分くらいのなのでぜひ視聴してみてください。またサイトではその論文へのリンクが張ってあるので読んでみるのもいいですね。雑談ではないテック podcast は貴重なので続いて欲しいですね。

5. 出典

https://misreading.chat/ Misreading Chat CS の論文読んで話をしよう・・・この PODCAST

https://note.mu/morrita/m/mf1be602603eb Misread Footnotes・・・MisreadingChat の 補 足ブログ

http://deeplearning.hatenablog.com/entry/transformer ディープラーニングブログ 論文解説 Attention Is All You Need (Transformer)

https://www.youtube.com/channel/UCbfYPyITQ-7l4upoX8nvctg TwoMinutePapers・・・ 2 分で論文を紹介する Youtube チャンネル

センテンス指向プログラミングの提案

著 :We

1. はじめに

マーチン・ファウラーさんの「流れるようなインタフェース」にインスピレーションを得て自分なりに考えた結果、やりたいことを文章で書いたら動く、と幸せなんじゃないだろうかというお話です。

サンプルコードはすべて C# で書かれています。

2. マーチン・ファウラーさんについて

あなたはマーチン・ファウラー (Martin Fowler) さんをご存知でしょうか。ThoughtWorks 社で働く技術者で、アジャイル開発やエクストリーム・プログラミングに精通したソフトウェア工学の第一人者です。著書も多数あり、『リファクタリング : プログラムの体質改善テクニック』などが有名です。昨年第２版が発売されましたが、まだ日本語版はありません。

マーチン・ファウラーさんはインターネット上でも精力的に活動しており、自らのサイトでさまざまなソフトウェア論を展開されています。

- Martin Fowler's Bliki
  https://martinfowler.com/bliki/

こちらはありがたいことに、有志による日本語訳サイトがあります。数百にもおよぶ記事が公開されています。

- Martin Fowler's Bliki (ja)
  https://bliki-ja.github.io/

3. 流れるようなインタフェース

前項の日本語訳サイトに、流れるようなインタフェースという記事があります。

- 流れるようなインタフェース
  http://bliki-ja.github.io/FluentInterface/

同記事では、流れるようなインターフェースがなんなのかについて、サンプルコードにて解説しています。時間の間隔を得るためのコードは、通常だと以下のようにしますが…

```
TimePoint fiveOClock, sixOClock;
...
TimeInterval meetingTime = new TimeInterval(fiveOClock, sixOClock);
```

流れるようなインターフェースではこうなります。

```
TimeInterval meetingTime = fiveOClock.until(sixOClock);
```

可読性が上がりましたね。より直感的になったとも言えます。ほかにもサンプルコードを用いて解説していますが、ここでは割愛します。流れるようなインターフェースについてさらに知りたい方は記事を参照してください。

もう一つ、例を出しましょう。文字列が null または空かを判断する場合、何も考えずに書くとこうなります。

```
string name = GetName();
if (name == null || name == "") return;
```

条件式に and や or があると一気に可読性が下がりますね。幸いにも、null または空の判定は専用のメソッドが用意されています。

```
if (string.IsNullOrEmpty(name)) return;
```

いくぶんマシになりました。さらに改良してみます。

```
if (name.IsNullOrEmpty()) return;
```

どうでしょうか。さらにすっきりしました。これは string に拡張メソッドを生やすことで実現できます。

```
public static class StringExtensions
{
    public static bool IsNullOrEmpty(this string value)
    {
        return string.IsNullOrEmpty(value);
    }
}
```

3.1. 可読性

なぜ可読性が上がったのでしょうか。文字数が減ったからでしょうか。それもあるかもしれませんが、私は手続き的な記載から文章になったからだと考えています。手続き的な記載の場合は、それが意味することを考える（変換する）必要がありますが、文章であれば読んでそのまま意味を理解できるからではないでしょうか。

最初のサンプルコードは、以下のような思考の経過をたどるかと思います。

- 「meetingTime は…TimeInterval クラスをインスタンス化して…引数には 5 時と 6 時を渡す…ああ、5 時から 6 時の範囲か。」

流れるようなインターフェースのコードは、読んでそのまま理解できます。

- 「meetingTime は…5 時から 6 時まで。」

なんとすばらしいことでしょう。ダイレクトに意味が伝わる、それだけではありません。コードを書くときもやりたいことをそのまま文章で書けばいいのです。やりたいこと（ミーティング時間は 5 時〜 6 時）をコードに変換する必要がなく、コードを読むときも変換が不要です。このスタイルを「センテンス指向プログラミング」と名づけることにしました。ちょっと仰々しいですが。

4. センテンス指向

どのように書けばセンテンス指向になるのか。実はとってもかんたんです。コードを Google 翻訳につっこんで、意味の通った日本語訳が出てきたらそれはもう立派なセンテンス指向プログラムです。この Google テストに通るかどうかで判断します。

実際にサンプルコードを Google テストにかけてみましょう。コードそのままだとさすがにうまくいかないので、複合語はバラバラにして、＝は is に、記号は空白に置き換えます。

```
TimeInterval meetingTime = fiveOClock.until(sixOClock);
↓
Time Interval meeting Time is five O'Clock until six O'Clock.
↓
時間間隔のミーティング時間は、5時から6時までです。
```

やったー大成功。型名が混じったので主語がちょっとおかしいですが、ちゃんとした文章になって出てきました。ついでにもう一つのサンプルコードも、テストにかけます。if 文の場合は、条件節と帰結節の間に , を入れます。

```
if (name.IsNullOrEmpty()) return;
↓
if name Is Null Or Empty, return.
↓
名前がnullまたは空の場合、戻ります。
```

これであなたもセンテンス指向をマスターしました！それだけでは投げやりすぎるので、より詳細なテクニックについてお話します。

4.1. メリット、目的

その前に目的の確認を。センテンス指向ではなんといっても生産性アップが第一の目的です。キーボードを叩いてコードを書いてる時間より、読む時間のほうが長いですよね。可読性アップは生産性アップももたらすのです。また、センテンス指向で作られたオブジェクトは直感的に扱えるため、

コードを書く時間も短くなります。そして文章になるようにオブジェクトをつくったりメソッドを生やしたりするのは中々苦労しますが、それがいい感じなオブジェクトを設計できるのです。

- 可読性がアップする - コーディング速度もアップする - 優れたオブジェクト設計ができる

4.2. 文法について

UTF8 の時代になりましたがソースコードを日本語で書くわけには行かないので、英語で書く必要があります。5 つの基本文型、覚えていますか？ SV, SVO, SVC, SVOO, SVOC…などありましたがセンテンス指向ではただ一つだけ覚えれば十分です。

- 主語 + 動詞 (+ 必要があれば何か追加する)

どうでしょう、これなら簡単ですよね。括弧内が何となくいやらしいですが、大事なのは主語→動詞なのです。ここがしっかり読み取れるよう作ればあとに続く形が雑でも Google テストは通りますし、可読性も保たれます。 1 つずつみていきましょう。

4.3. 主語

主語になるのは以下のものです。

- クラス、構造体 - 変数 - プロパティ

ようするにメソッド以外ですね。なのでこれらはすべて名詞にしてください。

4.4. 動詞

メソッドが該当します。じゃあ ToString() は使っちゃだめなんですか？という声が聞こえてきました。文法を守るのはあくまでも可読性を上げるための手段なので、そう書いたほうが可読性がいい場合は文法を無視してかまいません。ただしタイポはだめです。また、動詞には自動詞とか他動詞があった気がしますが、特に意識しなくていいです。

4.5. あとに続くもの

目的語とか補語とかなんか勉強しましたね。変数やプロパティ、メソッド名の末尾につける場合もあります。引数にする場合は引数名を前置詞にするといい感じになるかも。

```
class Schedule
{
    public string Title { get; set; }
    public TimeInterval Time { get; set; }
    public Schedule(string title, TimeInterval at)
    {
        Title = title;
        Time = at;
    }
}
...

new Schedule("ミーティング", at: fiveOClock.until(sixOClock));
```

前置詞はメソッド名の末尾に追加してもいいですね。（前置詞なのに！）

```
meeting.ChangeTimeTo(sixOClock.until(sevenOClock)));
```

4.6. 文章を長くしない

長い文章は可読性が下がります。検索するメソッドがいて、検索条件をメソッド名に含めると大変なことになります。

```
void findById(int id)
void findByName(string name)
void findByCreateDateAndIsDeteted(Date date)
```

メソッド数が爆発して死んでしまいます。引数を潤沢にしても、オーバーロードの爆発で死にます。こういうのはクエリオブジェクトにひとまとめにして、引数で渡したほうがいいです。

```
class Query
{
    public int Id { get; set; }
    public string Name { get; set; }
    public Date CreateDate { get; set; }
    public bool IsDeleted { get; set; }
}

void findBy(Query query)
```

4.7. そのほか

自クラス内のメソッドにアクセスする時は主語はなくていいです。朝の準備を行うメソッドは…

```
/// シャワー浴びて、朝ごはん食べて、歯を磨く
void prepare() {
    take(shower);
    eat(bread);
    brush(teeth);
}
```

命令文なので主語がいらないですし、対象は自分だと分かってますからね。

5. デメリット

光あるところには闇もあります。センテンス指向を実践する上での注意点や、向かないものについてお話します。

5.1. あいまいさ

既存のコードが手続き的で分かりにくいのには理由があります。こういったコードはあいまいさが発生しにくいからです。

人間があやつる自然言語は、あいまいさが非常に入りやすい作りになっています。センテンス指向では自然言語に近い表現になりますが、あいまいさが入らないよう注意してください。

たとえば save という単語は

- 救う
- 保存する
- 節約する
- 割り引く

など複数の意味を持ちます。自然言語ではこういった多義語は重宝されますが (get give take とか)、センテンス指向ではできる限り単一の意味しか持たない単語をチョイスしてください。救うなら rescure、割り引くなら discount など。でないとオーバーロードであふれかえって、しかもオーバーロードごとに異なる動作をするので使いにくくなってしまいます。文章的な正しさよりも、ユニークさ間違わなさを優先してください。

5.2. やりすぎ注意

何事もほどほどが一番、いきすぎたセンテンス指向はむしろ可読性が下がる場合があります。

TODO アプリを作っているとして、新しい TODO 項目を作る時は以下のようにしますよね。

```
class TodoItem {
    public string Title { get; set; }
    public bool Checked { get; set; }
}

...

new TodoItem
{
    Title = "歯磨き粉を買ってくる",
    Checked = false
};
```

よくある初期化です。言語機能を使って十分シンプルで可読性もあります。これを無理やりセンテンス化するとどうなるでしょうか。

```
TodoItem.Create()
        .Title("歯磨き粉を買ってくる")
        .Unchecked();
```

new やプロパティと言った既存の言語機能をあまりにも無視したデザインは、利用者の学習コストがいたずらに上がるだけで可読性にはあまり寄与しません。インスタンス化は new、値の設定はプロパティ、問い合わせや命令はメソッドで行えばいいのです。なんでもかんでもセンテンス化するよりも、既存の仕組みのほうがわかりやすい場合もあります。

6. さいごに

すべてをセンテンス指向にする必要はありません。センテンス指向は 1 メソッドから、1 オブジェクトから適用できます。使いにくいなあ、分かりにくいなあと感じたメソッドと出会ったとき、この手法を使ってリファクタリングしてください。そして、どうなったかをぜひ私に教えて下さい。

編集後記

ページが余ったのでなんか書いときます……。

PeerCast コミュニティの技術合同誌としてお届けしましたが、いかがだったでしょうか。

PeerCast はゲーム実況動画が主に配信されていることもあってかゲーム寄りの記事もいくつかありましたが、意外と技術的なものが揃いました。全然技術関係ないのでも良いということで募集してたんですが、さすがにそこまでチャレンジングな記事を書く人は居ませんでしたね。全般的にあっさり目な内容ですが、ざっくりとした合同誌なんでこんなもんでしょう。気軽に読んでもらってなにか一つでも気になるものがあれば幸いです。

こんな本を企画した経緯についてですが、技術書典に本は出したいが年に 2 冊も新刊書くのしんどかったんですよ。じゃあ合同誌にすれば自分で書く内容少なくてもいいんじゃね？みんなで書けばそれなりのページ数になるんじゃね？というよこしまな考えからこの合同誌が誕生しました。目論み通り自分では 14p しか書いてないけど全体ではそれなりのページになりました。やったね！

いや本当はそんな理由だけじゃないんですけど。

PeerCast コミュニティってのはほんと緩いつながりしかないうえに、やはりゲーム配信がメインですから技術的なことをやってる人のイベントというのは少なくなりがちです。たまに (オンライン) 勉強会やハッカソンを企画する人もいますが、最近ではあまりそれも行われていないし、自分でやってもいいけど普通にやってもなーと思ってました。で、技術書典に行くといくらか合同誌みたいなのが出てて、ちょっといいなーと思うところがあったので、じゃあ PeerCast コミュニティでやればいいんじゃねえのということでやってみました。

今後続くかどうかは不明ですが、何かしら PeerCast コミュニティでの技術イベントとしてはぼちぼちやっていきたいと思っています。もし今後があれば全然技術関係ない記事も欲しいですね。もし書きたいという方がおりましたらご意見ください。

著者紹介

kumaryu - IPv6 とつながらない P2P

ネットワーク全然わからんのに PeerCastStation という P2P 動画配信ソフトを作ってたりする人。

あとこの本を企画した。

ぷろぐれ - CS:GO サーバー改造事始め

普段は Web やスマホアプリのシステム作ったり面倒見たりする人。 FPS は Co-op しかやらん模様。札幌在住。

https://twitter.com/progremaster

ぬか - VRC 民に効く (かもしれない)Unity 操作 Tips まとめ

ゲームをつくったり絵をかいたりしています。

twitter　@nuka000

vivivi - Misreading Chat リスニングガイド

php や ruby などの LL でみなさんのめんどくさいことを自動化する仕事をしてます。

We - センテンス指向プログラミングの提案

Web とか .NET とか。今は iOS アプリ。勉強会もきまぐれにやってます。

https://nomipro.doorkeeper.jp/

twitter @RunLucky

めいまぁ - 表紙

pixiv129039

こんにちはこんばんは

peercast で絵を描く気力を出しています

配信出来なかったらやる気も出ません

奥付

タイトル：　PeerCast 技術合同誌 Broadcast Yourself
発行日：　2019 年 9 月 22 日 初版発行
サークル名：あれくま
発行者：　kumaryu
連絡先
　Web:　https://www.kumaryu.net/
　メール：　kumaryu@kumaryu.net

2019
あけまして