NAIST-IS-MT0851123

修士論文

大規模多人数参加型オンラインロールプレイングゲームにおける トラフィック成分分析を用いた **Bot** 判別手法の提案

森部　皓裕

2011 年 2 月 3 日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

森部　皓裕

審査委員：

山口 英 教授　　(主指導教員)
関 浩之 教授　　(副指導教員)
門林 雄基 准教授　(副指導教員)

大規模多人数参加型オンラインロールプレイングゲームにおけるトラフィック成分分析を用いた **Bot** 判別手法の提案*

森部　皓裕

内容梗概

現在，MMORPG (Massively Multiplayer Online Role-Playing Game) において Bot プレイヤーがゲーム内のコンテンツを急速に消費し，ゲームバランスを崩壊させるという問題がある．Bot はルーチンによって動作するため，人間プレイヤーと Bot プレイヤーのトラフィックには差異が現れるはずである．本稿ではトラフィック成分分析手法を用いて人間プレイヤーと Bot プレイヤーとの差異を抽出できるか否かを検証する．また，検証の結果，トラフィック成分分析により人間プレイヤーと Bot プレイヤーとの差異を抽出できることが明らかとなった場合，具体的な Bot プレイヤー判別手法を提案する．本稿では，トラフィック成分分析手法としてゆらぎ分析手法，エントロピー分析手法，ウェーブレット解析手法を用いた．まず，台湾国際大学が公開しているデータセットを用いて各解析手法によりトラフィックの差異を抽出できるか否かを検証した．検証結果から人間プレイヤーと Bot プレイヤーのトラフィックの差異が現れたため，人間プレイヤーと Bot プレイヤーのトラフィックには差異があるという対立仮説を設定した．そして，独自に取得した別の MMORPG のトラフィックデータセットを用いて解析を行い，解析結果を母集団とした t 検定による対立仮説の検定を実施した．検定の結果，対立仮説は統計的に有意であることが示された．これらの検証結果をもとに Bot プレイヤー判別手法に関する議論を行い，議論の結果ゆらぎ分析を用いた Bot プレイヤー判別手法を有効なトラフィック分析手法として提案した．

キーワード

Bot, MMORPG, トラフィック分析, 波形解析, アルゴリズム

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT0851123, 2011 年 2 月 3 日.

Proporsal of bot detection methods based on traffic freacance analysis for masivily maltiplayer online games *

Kousuke Moribe

Abstract

In Massively Multiplayer Online Role-Playing Game (MMORPG), administrators need to detect Bot players because of the threat they impose on the game balance by exhausting game contents at a quick pace. Indeed, Bot operation follows a scripted routine that automates the Bot activities, requiring no human intervention. This can result in differences of traffic pattern between human and Bot players. In this research, we examine whether the difference between Bots and human players traffic can be characterized. We performed traffic componential analysis by leveraging fluctuation analysis, entropy analysis and wavelet analysis. At first, we employed a dataset published by the Academia Sinica of Taiwan to validate, for each analysis method, whether we were able to characterize differences between human and Bot player traffic. As a result, differences between human and Bot player traffic arose and we were able to form an alternative hypothesis that differences exist between human and Bot player traffic. We conducted a Student's t-test against the output of the analysis, by each method, using another independently collected MMORPG traffic dataset. The results of the t-test were statistically significant. According to these results, we discussed Bot player detection methods and proposed a noise analysis based detection method as a countermeasure to Bot players.

Keywords:

Online game,Traffic analysis, wave analysis, algorithm

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0851123, February 3, 2011.

目 次

図目次

表目次

1. はじめに

今日，インターネットを介して専用のサーバや他のユーザのクライアントマシンと接続し，オンラインで同時に同じゲーム進行を共有することができる MMORPG (Massively Multiplayer Online Role-Playing Game) というサービスが広まっている [1]．中には，ユーザ数が数千万人というものもあり，インターネット上で大きな市場を形成している．

現在，MMORPG において Bot による被害が問題となっている．Bot とは,「決まった行動」，つまりルーチンにしたがって自動的に行動するプレイヤーのことである．この Bot が複数で 24 時間動き続けると，ゲーム内コンテンツ (アイテムなど) は急速に消化され，MMORPG 内のゲームバランスは崩壊してしまう．MMORPG のゲームバランスを保つためには，短時間で効果的に Bot を判別できる Bot 検知手法が必須である．本研究では以降，MMORPG における Bot 検知手法に関して議論を進める．本章では 1.1 節で MMORPG についての説明を行い．1.2 節，1.3 節，1.4 節にて現状の Bot 検知手法を整理し，1.4 節にて本研究の目的を述べる．

1.1 MMORPG とは

MMORPG とは，インターネットを通して専用のサーバや他のユーザのクライアントマシンに接続し，オンラインで同時に同じゲーム進行を共有することができるロールプレイングゲームサービスを指す．基本的には，定額課金性であり，ゲームサービスを長い期間ユーザに使ってもらうことで収益を上げるサービスである．その多くは，ユーザの使用するゲーム内のキャラクターの成長やゲーム内アイテムのコレクションなどのゲーム内パラメータの上昇を目的とおいたサービスである．現在，このサービスは成長しており，欧米では 1.6 億ドルの市場に達しているとされている [2,3]．この MMORPG においては，問題点も様々にある．その中の一つが Bot という問題である．

1.2 Bot とは

Bot とはルーチンにしたがって自動的に行動するプレイヤーのことである．Bot は，ゲーム内で自動で動きゲーム内のモンスターを倒したり，アイテムを集めたりすることができる．また，自動で動くために人間のように休憩することも無いため，人間よりもゲーム内コンテンツの消費を早めてしまう．そのコンテンツが，ユーザ同士のアイテム等の交換や Real Mony Trade (RMT) と呼ばれるゲーム内の通貨やアイテムを実際の金銭と交換する行為によりゲーム内に広く分散していく．MMORPG の運営者は，ユーザが飽きないようにゲーム内コンテンツを追加していく．し

かし，Bot が急速にコンテンツを消費していき，それが分散していくため，このコンテンツの追加が追いつかず，ユーザがゲームから離れていく．終いには，ゲームとして成り立たなくなる状態までいたり，ゲームサービスとして運営できなくなる．

実際に，Bot の蔓延により深刻な状態に陥ったゲームサービスは「ラグナロクオンライン [4]」「リネージュ[5]」など韓国製の MMORPG を中心に多々ある．Bot への対策は MMORPG サービスを運営する上で重要な要素である．

1.3 行動分析手法

現状の MMORPG において，ゲーム運用者は行動分析手法を用いて Bot プレイヤーもしくは Bot プレイヤーを利用しているユーザを検知し，Bot プレイヤーを用いているユーザへの注意やアカウント停止などの対応を実施している．本節 1.2 では現状の行動分析手法を紹介する．

1.3.1 ゲーム管理者による声かけ

この方法は，ゲームの管理者がゲーム内を巡回し，Bot らしきプレイヤーを見つけるという方法である．具体的には，ゲームマスター (通称 GM) と呼ばれるゲームの管理者が，各ゲーム内 Map を回り，Bot らしきプレイヤーにチャットで声をかけ，その反応から Bot を判別するというものである．この方法は，ほぼ確実に Bot を判別しなんらかの処置を行うことができる方法である．すべての Bot を判別し，処置を行おうとすると人件費によるコストが非常に大きくになってしまう．コストが大きくなると，その分，ユーザかゲームの運営者に負担がかかってしまう．ゲーム会社が負担を負うと，ゲーム運営者にとっては経済活動として魅力的なものでは無くなってしまう．また，ユーザが負担を負う料金設定にする場合，ユーザの集客力に多少の他のゲームと比べて不利になり，これもサービスとして運営することができない状況になってしまう場合もある．そこで，コストが低い Bot 対策手法が必要となる．

1.3.2 対策コード

自動的に Bot に対する対応を行う手段として，Bot 自体を使えなくするという方法が提案された．具体的には，ゲームのクライアントに Bot への対策コードを追加し Bot を使用不可能にする方法である．しかし，クライアントがユーザの端末側にある以上，Bot 製作者による解析ができてしまうため，最終的にはこの方法に対応した Bot が広まってしまう．このため，効果的な対策とは言えない．

図 1 ゲーム管理者による声かけ画面

1.3.3 不正行動無効化ソフトウェア

外部の不正行動無効化ソフトウェアにより Bot を使用不可能にする方法がある．これは，不正行動無効化ソフトウェアを強制的にゲームをする端末にゲームクライアントをインストールさせる際，強制的に不正行動無効化ソフトウェアをインストールさせることで，クライアントの解析や Bot を使用不可能にするというものである．

このコンセプトのもと，作られたソフトウェアが nProtectGameGuard [6] である．これは，韓国のインカインターネットが開発したソフトウェアであり，悪性コードの検知及びブロック，オートマウス及びマクロツールのブロック，ゲームクライアントの不正なアクセス操作遮断，パケットクライアントの暗号化，CPU 占有率の最適化，クライアント環境のリアルタイム監視を行うソフトウェアである．このソフトウェアは多数の MMORPG で採用されていたが，オペレーティングシステムの動作障害やユーザがゲームにログイン出来なくなる等の障害が発生したことで，現状では nProtectGameGuard は導入しない MMORPG が一般的である．

一見，これらのバグを解消すれば問題は解決され Bot 問題は解消されるように考えられるが，nProtectGameGuard には 2 つの大きな問題がある．ひとつは，クライアントのリアルタイム監視による CPU の負荷増加による計算機の必要計算能力の相対的な上昇である．一般に MMORPG は，高い計算能力が要求される．高い計算能力が要求される MMORPG ほど，サービス対象となるユーザは少なくなってしまう．多数のユーザに遊戯をしてもらうことでサービスを運営する MMORPG にとってサービス対象となるユーザ数を少なくしてしまうのはサービスとして致命的である．現在サービスが行われているゲームにおいても，nProtectGameGuard の導入により，必要となる計算機の計算能力が上昇し，通常市販されている PC のスペックでは遊戯に支障をきたしてしまいユーザが離れてしまうというケースが多々ある．その上，さらに問題なのが，こういった監視ソフトウェアは，クライアント側にあるので，Bot 使用者によるソフトウェアの解析が容易である為，図 2 に示すとおり，ソフトウェアの回避方法がすぐに出てきてしまい広まってしまう傾向がある．現に nProtectGameGuard は，解析により回避方法が広まっており Bot の抑止方法としては全く意味を持たないものになっている．その為，クライアント以外での対策が必要となる．

1.3.4 ゲーム内判別法

Bot はスクリプトにより一定の行動（ルーチン）に従った行動を繰り返す．ルーチン自動検知ではプレイヤーの行動を観測しルーチンをを自動的に検知することで Bot を検知するというものである．ルーチンの具体例には，特定のゲーム Map でしか移動しない，チャットを使わない等が挙げられる [7–9]．また，ゲーム内のプレイヤーの軌跡から移動ルーチンを検知し Bot を判別するという学術研究もある [10,11].

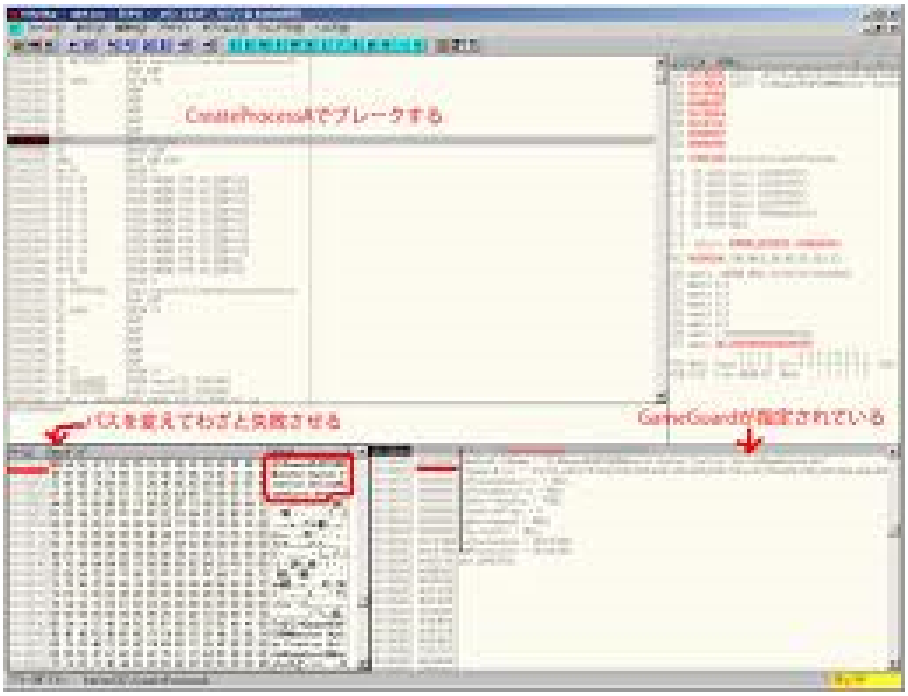
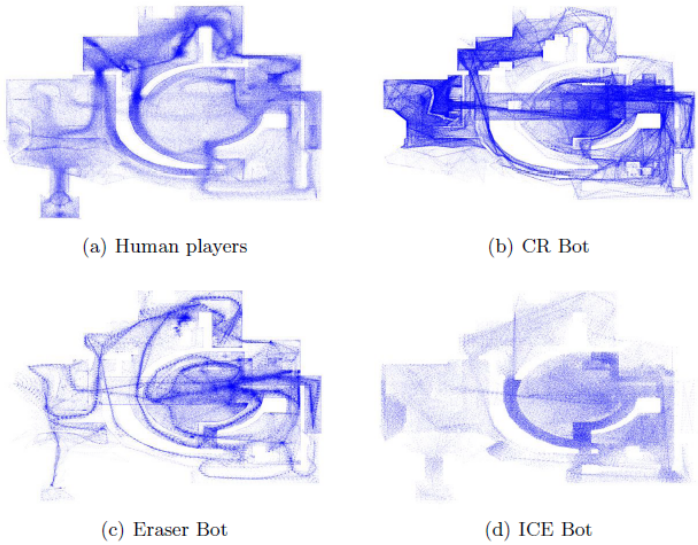


図 2 nProtectGamegurd の無効化方法



(a) Human players (b) CR Bot

(c) Eraser Bot (d) ICE Bot

Fig. 1. Presence locations of all players

図 3 軌跡によるルーチン検知を用いた Bot 判別方法 (参考文献 [10] から引用)

ルーチン自動検知には一定期間の効果はあるのだが，ゲーム運営者側のルーチン検知アルゴリズムが Bot 作成者から推測できるため，Bot 作成者による回避が容易である．実際に，文献 [10,11] のようなゲーム内でのプレイヤーの軌跡をもとにしたルーチン検知アルゴリズムに対する Bot 作成者側の回避策としてゲーム内ルート作成ツールが存在する．現状の MMORPG では，ゲーム運営会社により新しいルーチン判別アルゴリズムが定期的に導入されるが，結果的に Bot の製作者による対応が行われ，いたちごっこになっている．

これは，ゲーム上のマップにおけるユーザの軌跡において，Bot は同じような軌跡を通るので，その軌跡から Bot とユーザとを判別するというものである．しかしながら，先でも述べた通り，このような方法は Bot の製作者による対策が容易である (先の例では，ゲーム内ルート作成ツールが存在)，ゲーム内の行動とは関係のない Bot の検知方法が求められる．

1.3.5 CAPTCHA

CAPTCHA とは公開チューリングテストである．この用語はカーネギーメロン大学の Luis von Ahn，Manuel Blum，Nicholas J. Hopper，IBM の John Langford によって 2000 年に作られた．CAPTCHA という語は「Completely Automated Public Turing test to tell Computers and Humans Apart」（コンピュータと人間を区別する完全に自動化された公開チューリングテスト）の人為的頭

図 4 CAPTCHA による質疑応答を用いた認証画面

文字である [12]．この公開チューリングテストを用いて Bot の検知を行うという方法がある [13]．図 4 は，英雄オンラインにおける CAPTCHA 画面である．これはプレイ中にいきなり，足し算をせよという画面が出てくるというものである．この方法は,「ゲームのプレイ自体に支障が出る」「世界観が壊れてしまう」等のユーザからの不評な意見が相次ぎ，導入は進まなかった．プレイ自体に支障がないような仕様のものを作れば有用な方法であると考えられるが，現時点においてそのようなアイディアはない．

1.3.6 Real Money Trade (RMT) の禁止と取り締まり

RMT は MMORPG 上のお金や所有アイテムなどを現実のお金と交換するというもので，RMT サイト [14] などを使ってユーザ間で取引がされている．

RMT は，レアアイテム（稀少なゲーム内アイテム）が高額で取引される．レアアイテムの収集に Bot が使われることが多い．RMT を禁止もしくは取り締まることで Bot 使用に対するインセンティブが低下し，Bot の利用者が減るのではないかとの考えのもと，ほとんどの MMORPG において RMT は禁止されており，運営会社による RMT の取り締まりも行なわれている [15]．RMT

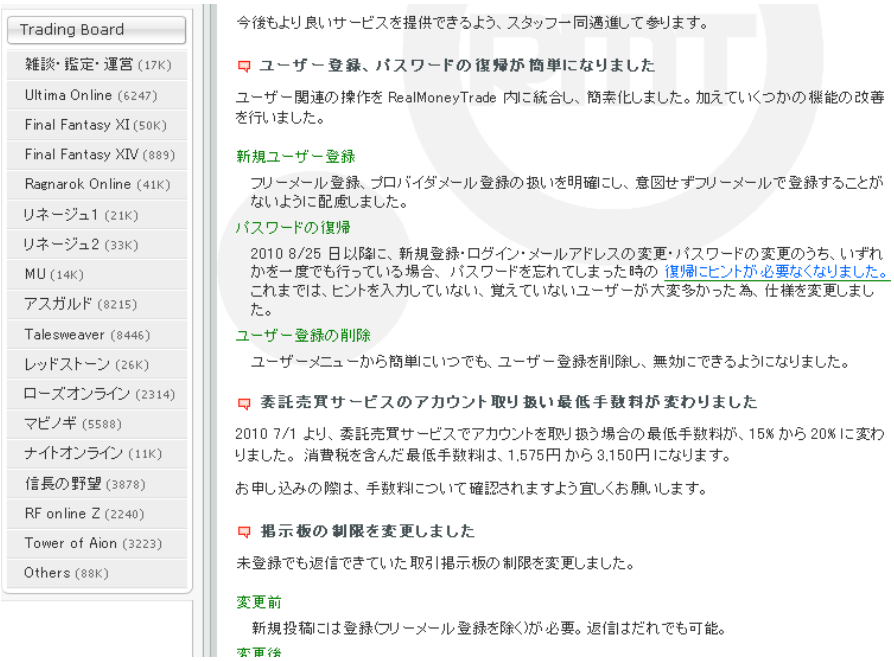


図 5 RMT サイト

利用者の取り締まりの際，MMORPG 運営会社は RMT を使ってゲーム内の通貨やアイテムを交換したであろう形跡から RMT 利用者を見つけ出し，アカウント削除やアカウント停止等の処置を行う．

しかしながら，運営側における RMT の使用検知手法が非常に曖昧であった．例を挙げると，それまで交流のなかったキャラクターとの大口での取引を検知するというものである．これの他にも，検知手法は存在するのだが，どれも RMT を明確におこなったことが確かと言える手法ではなく，かつ通常のゲーム内での行動でも十分に行われる可能性のある条件を手法としたものであった．それゆえ，false positive の割合を非常に多く含むものであったため，RMT を行っていないアカウントを削除してしまうという事例が多くでてしまった為，この方法は今現在，極めて普遍的でない状況以外では適用されていない．この方法に関しては，実際の金銭の動きをゲーム運営者側が認知できれば，RMT を取り締まることができ，Bot を減少させることができる可能性もあるが，実際の金銭の動きをゲーム運営者側が把握することは非現実的であり，それゆえ効果的な Bot 問題の問題解決方法とはいえない．しかし，今現在においても多くの MMORPG において RMT は禁止されており，実際に取り締まりをおこなっているゲーム運営会社も存在する．

1.4 トラフィック判別手法

現在，Bot に対する有効な検知方法としてトラフィックによる Bot 判別手法が研究されている．トラフィック解析手法は，ゲーム内の局所的な行動とは関係がないため Bot 検知手法を Bot 制作者が推定しにくい．また，仮に Bot 制作者により対策が立てられたとしても，回避のためのルーチンが複雑になるため Bot の制作者による対策がされにくい検知方法とされている．文献 [16] では，ラグナロクオンラインを対象とし，ラグナロクオンラインにおける Bot の静的解析から導きだされた複数の検知手法を組み合わせることにより，高精度な Bot の検知手法を提案している．文献 [17] の実験結果では 95 %以上の検知精度を示している．

しかし，文献 [17] のトラフィック解析手法は，Bot を入手し静的に解析しゲームトラフィックにおける Bot の特徴を捉えることで複数の検知手法を生成するため，MMORPG のサービス開始時点から利用できない．静的解析による Bot の検知手法が確立する前に Bot が蔓延すると，検知手法が確立し Bot が検知可能になるまでの期間は，その MMORPG サービスにおけるゲームバランスが崩れ，MMORPG 運営に支障をきたすことになる．こような状況を回避するため，静的解析による Bot の検知手法が確立されるまでのつなぎとして，ゲームの種類を問わずサービス開始時点から利用できるトラフィック解析手法が必要である．

文献 [17] の Bot 検知手法の中にも，パワースペクトル変換におけるスパイクから判別する手法や，サーバに通信が集中している場合のトラフィック量の違いから判別する手法など，どのようなゲームにおいてもサービス開始時点から Bot 検知が可能であると考えられるトラフィック解析手法がある．これらを組み合わせ，かつ新たな検知方法を提案し，ゲームのサービス開始時点から使用できる検知方法を確立していくことが重要である．

1.5 本研究の取り組み

本章で述べた通り，現状では Bot に対する有効な対策となるものは存在しない．研究段階においてトラフィックによる Bot 検知方法が検討されているが，この方法は Bot を入手し静的に解析しゲームトラフィックにおける Bot の特徴を捉えることで複数の検知手法を生成しその手法を組み合わせ検知するため，MMORPG のサービス開始時点から利用できない．従来の方法より，効果的に Bot を検知を行うためには，ゲームのサービス開始時点から Bot 検知方法として使用できうる方法が必要となっている．そこで，本論文では，ゲームのサービス開始時点から Bot 検知できうるトラフィックによる Bot 解析手法のための手法を検討し提案を行う．本論文では，2 章にて，トラフィック分析手法の検討を行う．3 章では，その特徴を踏まえたうえで，過去の研究からゲーム開始時点から使用できうる手法を抜粋し評価を行う．その後，4 章では他ゲームにおいて再

度手法を検討し，差が見られるようであれば，統計的に有意であるか検定する．5 章では，その結果を用いて実際に Bot 検知を行なった．6 章では，本論文のまとめと今後の課題を述べる．また，付録に解析に使用したプログラム，R スクリプトを記載する．

2. 成分分析手法と仮説の設定

ここまでで述べたとおり，ゲームサービス開始時点から使用可能なトラフィックによる Bot 判別手法が有効である．しかしながら，現在のトラフィックによる Bot 判別手法は，Bot を静的に解析した上で手法を作り，その上で高い精度で Bot を判別するというものであった．しかし，解析を行い，トラフィックによる Bot 判別方法を確立する間に，Bot が広まってしまい，ゲームのコンテンツが消費され，ユーザが離れてしまっている状態になっている例が多々ある．そのため，ゲームの開始時点から低精度であっても Bot の判別ができる手法が必要になる．本章では，ゲーム開始時点から適用可能な分析方法である成分分析手法と仮説について述べる．

2.1 Bot の構造

Bot の構造は大きく分けて 2 つの構造からできている．一つめの構造は，ゲームサーバからの信号に対し，ルーチンに従い特定のパケットを返す構造である．もう一つの構造は，ルーチンの特性からタイムアウトをするごとに一定の行動を繰り返す為，定期的に同じパケットをサーバに送信するという構造である．プレイヤーの行動がルーチン化されているため，Bot プレイヤーの行動は直前の行動からの経過時間をもとにして実行されることが多い．そのため，Bot プレイヤーのゲームクライアントからは一定の時間間隔で定期的にゲームサーバに対してパケットを送信することになる．

この Bot プレイヤーからのパケット送信の定期性から，Bot プレイヤーのトラフィックの発信タイミングには定期的な成分が含まれていると考えられる．トラフィックに定期的な成分が検知できたならば，そのトラフィクを送信しているプレイヤーは Bot プレイヤーであると判別できるのではないかと考えた．本章では以降，トラフィック内の定期的な成分を検知するための成分分析手法の説明する．また，各成分分析手法において人間プレイヤーと Bot プレイヤーとの差異を示す仮説を説明する．

2.2 ゆらぎ

ゆらぎとは時間の経過とともに規則的な変化をする現象である．また，不規則的な変化をしているように見える現象でも，その中で，規則的な変化をする現象もある．それらの現象をゆらぎという．ゆらぎによる分析は音の解析などに主に使われている [18]．特にゆらぎの一種である「$1/f$ ゆらぎ」は様々な分野で応用されている [19] [20].

ゆらぎは，パワーと周波数との関係で観測できる．ゆらぎの観測手順は，まず縦軸に観測する関数をパワースペクトル変換した値をおき，横軸にパワースペクトルに応じた周波数をおき，両

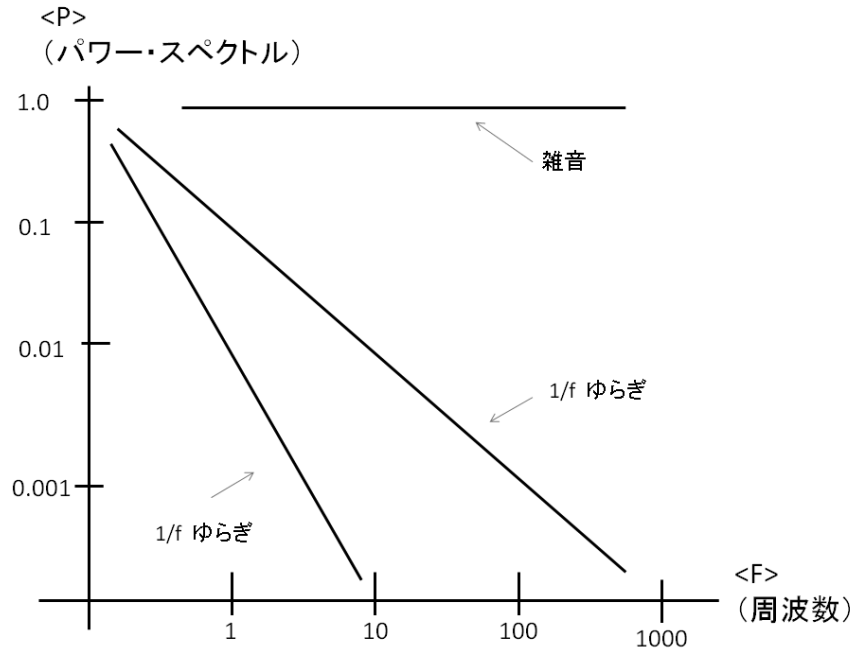


図 6 ゆらぎ

軸とも対数（両対数）にする．そして，導出したグラフにおいて，最小二乗法による近似直線からゆらぎを観測する．一般にこの直線の傾き水平ならばその関数はランダムであり，傾きが大きければ大きいほどその関数は定期的な成分が多く含まれ単調であるとされる．「$1/f$ ゆらぎ」は，この近似直線の傾きが-1 から-2 付近の値を示すゆらぎを指す．インターネットのトラフィックにおいても，ゆらぎは観測できる．例としては，VoIP トラフィックにおけるゆらぎ観測が報告されている [21].

Bot のトラフィックの定期的な成分により,「Bot プレイヤーのトラフィックのゆらぎは人間プレイヤーのトラフィックのゆらぎに比べ，ゆらぎの性質を表す近似直線の傾きが人間プレイヤーと負の方向に大きくなるのではないか」という仮説を提案する．

2.3 ウェーブレット解析

波形の解析の手法として様々な手法が編み出されている．そのなかでもウェーブレット解析 [22] は，近年有力な解析手法として注目されているものである．文献 [23] によると，ウェーブレット解析は，短時間フーリエを応用した一定の形をした短い波（wavelets of constant shape）を使った

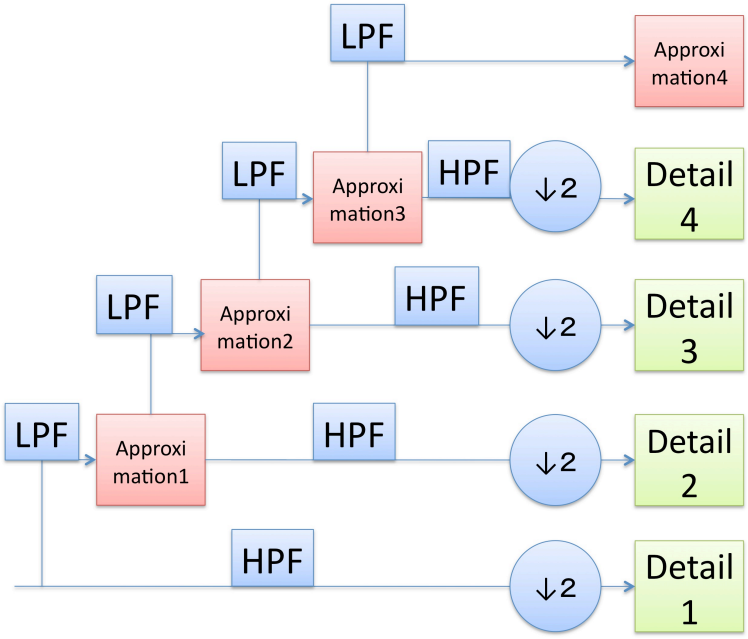


図 7 wavelet 変換におけるサブバンド符号化

新しい時間周波数解析に始まるとされている．ウェーブレット解析はフーリエ解析のように三角関数の波の重ね合せで関数（信号）を表現するのではなく，短い波（wavelets）の重ね合せで関数（信号）を表現するというものである．時間周波数解析には不確定性原理という大きな制限があって，時間と周波数の両方の情報を同時に詳しく知ることはできない．ウェーブレット解析は時間周波数の窓により，時間領域の情報を残すことが可能である．このことから，時間分解機能込みでの Bot 独自の波形，もしくは人間独自の波形を見つけ出し，これを検知手法とすることを仮説として提案する．

また，ウェーブレット解析は，一種のサブバンド符号化であり，周波数の低域側について分割／合成の処理をツリー状に繰り返すオクターブ分割により表すことができる．これらは，ローパスフィルタ（LPF），1 サンプルおきに信号を間引くサブサンプラ，各サンプルの間に値 0 を挿入するアップサンプラ，加算器により構成される．4 段構成（レベル 4）の場合，図 7 のように分割できる．この過程で，定期的な成分が存在する Bot のトラフィックと，定期的に動くとことは難しい人間のプレイヤーのトラフィックは，高周波成分 (Detail 成分) において違いがでてくるのではないかと推測される．具体的には，Bot のパケット送信タイミングは定期的であり，不定期なものはランダムなノイズである高周波成分として出力されるため，ウェーブレット解析における Detail1 の成分の割合が人間のプレイヤーのトラフィックに比べ大きくなるのではないかというものである．そこで，本節では，Bot のトラフィックにおける Detail1 成分は，人間のプレイヤーにおける Detail 成分に比べ大きい値がでるのではないかという仮説も提案する．

2.4 エントロピー分析 (平均情報量)

エントロピーは，情報理論の概念であり，あるできごとが起きた際，それがどれほど起こりにくいかをあらわす尺度である．頻繁に起こるできごとは，それが起こったことを知ってもたいした情報にはならないが，めったに起こらないできごとが起これば，それはより多くの情報を含んでいると考えられる [18]．つまり，エントロピーとは情報の発生を予想する難しさの定量的な手法であり，エントロピーが大きいということは，情報の発生に関して予想することが難しいということであり，エントロピーが小さいということは情報の発生が予想しやすいということである．エントロピーは次の式で表す．ここで，$H(n)$ は平均情報量，P_i を確率とすると

$$H(n) = -\sum_{i=1}^{n} P_i log_2 P_i \tag{1}$$

で表すことができる．この予想のしやすさというのは分散で表すことができ，一般にデータの分散度合いが大きければトロピーの値が大きく，データが規則的であればエントロピーの値は小

さい．Bot は，パケット送信タイミングは定期的であるために，そのデータの分散は人間に比べ大きくなり，平均情報量は人間のトラフィックに比べ少なくなるのではないかという仮説を提案する．

2.5 検証する仮説

本章で提案した仮説を以下に挙げる．

- ゆらぎ分析
  - Bot プレイヤーのトラフィックのゆらぎと人間プレイヤーのトラフィックのゆらぎに違いが現れる
  - Bot プレイヤーのトラフィックのゆらぎは人間プレイヤーのトラフィックのゆらぎに比べ，ゆらぎの性質を表す近似直線の傾きが人間プレイヤーに比べ負の方向に大きくなる
- ウェーブレット解析
  - Bot プレイヤーのトラフィック独自の波形の特徴，もしくは人間プレイヤーのトラフィック独自の特徴がある
  - また，Bot プレイヤーのトラフィックのウェーブレット解析における Detail1 の成分の割合が人間のトラフィックに比べ小さい
- エントロピー分析
  - Bot プレイヤーのトラフィックの平均情報量は人間プレイヤーのトラフィックに比べ小さくなる

これらの仮説に関して 3 章にて分析及び解析を行う．

3. 仮説に対する予備検証と手法の設定

本章では，2 章で提案した分析手法を用いて，同時に提案した仮定の証明のために実際に Bot プレイヤーのトラフィックと人間プレイヤーのトラフィックの公開データにおいて分析し，その結果について示し考察する．この公開データは，Kuan-Ta Chen 氏が公開しているラグナロクオンラインにおける人間プレイヤーのトラフィックと Bot プレイヤーのトラフィックに関する pcap ファイルのデータセットを使用した [24]．このデータセットは，文献 [17] において Bot 検知に使用されたデータセットである．このデータセットにおいて人間のプレイヤーのトラフィックをデータセット A とし，Bot プレイヤーのトラフィックをデータセット B とする．このデータセット A，B において 2 章で述べた仮説に対し検証を行う．

3.1 ゆらぎによる Bot 検知手法

本節では，2 章で提案した次の 2 点の仮説について実験にて検証する．1 点目は，ゆらぎ分析を用いて，Bot プレイヤーと人間プレイヤーの違いが表れるのかを検証する．2 点目は，仮にゆらぎ分析により，Bot プレイヤーと人間プレイヤーとの違いが現れた場合，2 章で立てた仮説の通り，Bot プレイヤーのゆらぎの性質を表す近似直線の傾きが人間プレイヤーに比べて負の方向に大きくなるかを検証する．

3.2 サンプリング

検証にあたるデータにおいて単位時間あたりのパケットの出力回数をサンプリングし，そのサンプリングしたデータを基に検証する．サンプリングとは数学的に連続関数の値からある点の値だけを標本として取り出して離散関数に変換する操作である．サンプリング関数は，ある離散値（連続でない，ある周期での飛び飛びの値）x に対してのみ $g(x) = 1$ となり、その他の x に対しては $g(x) = 0$ となるような関数である．また，サンプリング関数において $g(x) = 1$ となる間隔をサンプリング周期という．サンプリングを行なう際，サンプリング周期の２倍以下でおこる事象に関してはその情報は失われる．

3.2.1 実験

本実験は，オフライン解析で解析を行った．実験では，Kuan-Ta Chen 氏が公開している MMORPG (Massively Multiplayer Online Role-Playing Game) ラグナロクオンラインにおける人間プレイヤーのトラフィックと Bot プレイヤーのトラフィックに関する pcap ファイルのデータセットを使用し

た [23]．このデータセットは，文献 [17] において Bot 検知に使用されたデータセットである．文献 [17] で公開されている人間プレイヤー 3 人分のトラフィック 6 セットと 2 種類の Bot による Bot プレイヤーのトラフィックが 6 セットが含まれている．これらをそれぞれ，Human1〜Human6, Bot のトラフィックを Bot1〜Bot8 と表記し，データセット A を作成した．データセット A から文献 [17] にならい，TCPdump [25] を用いてクライアントの送信パケットのみを取り出した．その後，純粋な TCP ack パケット（応答確認用パケット）をノイズ成分としてデータセット A から除去したデータセットを生成する．このデータセットにはゲームの内容に関するトラフィックが含まれることになる．

以上のようにして生成したデータセットを，10 ヘルツのサンプリング周期で 0.1 秒あたりのパケット出力回数をサンプリングし，2048 個のデータセット $f(n)$ を得る．このデータセット $f(n)$ を式 (2) のようにフーリエ変換し $F(n)$ を得る．

$$F(n) = \mathcal{F}[f(n)] \tag{2}$$

(3) 式のように $F(n)$ をパワースペクトル変換したものを $P(n)$ とした．

$$P(n) = F(n)^2 \tag{3}$$

また，周波数 f とパワースペクトル $P(n)$ の軸を両対数で示したグラフ上に最小二乗法で近似して導出し，ゆらぎを観測する．

3.2.2 実験結果

表 1 Bot プレイヤートラフィックにおける解析の実行時間と両対数グラフにおける近似直線の傾き

	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7	Bot8
傾き	0.023	0.055	0.021	0.065	0.051	0.246	-0.089	0.325

表 2 人間プレイヤートラフィックにおける解析の実行時間と両対数グラフにおける近似直線の傾き

	Human1	Human2	Human3	Human4	Human5	Human6
傾き	-0.254	-0.225	-0.097	-0.068	0.0633	-0.111

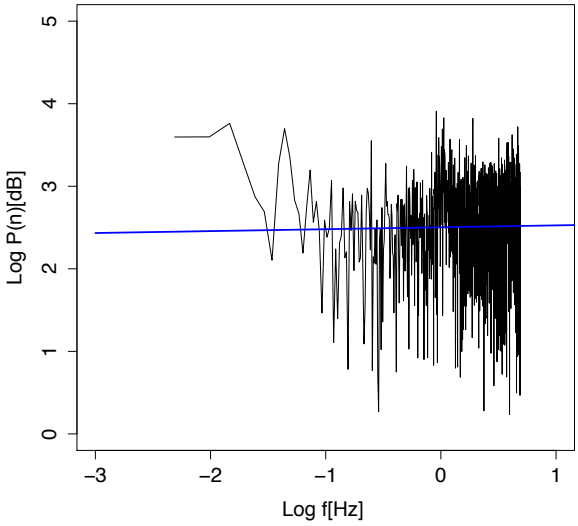


図 8 Human1:最小二乗法による近似値直線

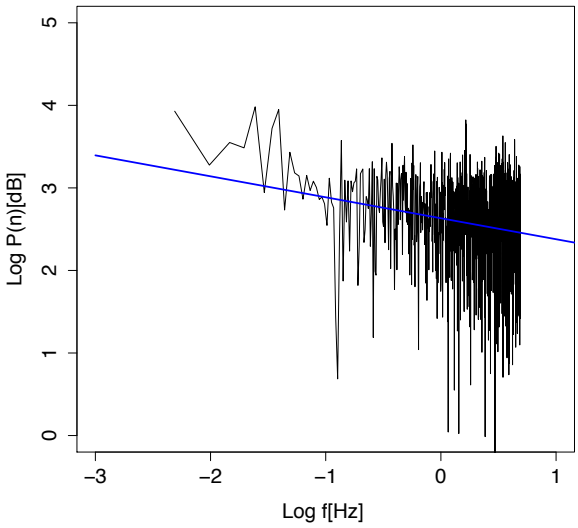


図 9 Bot1:最小二乗法による近似値直線

データセット A，B のフィルタリング作業及び解析作業は，以下の計算機で行った．OS は Mac OS X バージョン 10.5.8 [26]，プロセッサは 2 GHz Intel Core Duo, メモリは 2GB DDR2 SDRAM である．また，データセットのサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには R version 2.11.0 [27] を使用し解析を行った．表 1，表 2 に実験結果を示す．

図 9 に示すように，Bot のトラフィックは周波数成分が高いところも，低いところもパワー分布が変わらない水平な近似直線を持つトラフィックが多くある．

それに比べ，図 8 に示すように，人間プレイヤーのトラフィックでは，パワースペクトルと周波数との関係は低周波成分におけるパワーが高く，高周波成分におけるパワーが低い状態に推移している．そのため，最小二乗法での近似直線は，反比例 (負の傾き) の関係になっている．

表 1，表 2 は，各データ構造を作る際の実行時間とその合計，最小二乗法で近似した直線の傾きを示したものである．表 1 より，人間のトラフィックの傾きは周波数に対し負の傾きを示しているものが多くある．Bot のトラフィックでは，傾きが人間のトラフィックの傾きに比べ 0 に近い値を示し，比例する傾き (正の傾き) を示しているものが多い．

3.2.3 考察

2 章において,「Bot のトラフィックのゆらぎは人間プレイヤーのトラフィックのゆらぎに違いが現れるのではないか」ということと「Bot のトラフィックのゆらぎは人間プレイヤーのトラフィックのゆらぎに比べ，ゆらぎの性質を表す近似直線の傾きが人間プレイヤーと比べ負の方向に大きくなるのではないか」という仮説のもとに実験を行なった．実験結果から，Bot のトラフィックは人間に比べゆらぎ性質を表す近似直線の傾きが水平となり，白色雑音と呼ばれるほとんどランダムな成分で構成されているゆらぎに近いものが多いことがわかった．この結果により，トラフィックにおいて違いが現れるが，Bot のトラフィックのゆらぎは人間プレイヤーのトラフィックのゆらぎに比べ，ゆらぎの性質を表す近似直線の傾きが人間プレイヤーよりランダムな成分を多く含むことを表す水平な傾きになっていることがわかった．また，表 1 の Bot7 に見られるように，人間プレイヤーのトラフィックのゆらぎに近い負の傾きを示すものもあれば，反対に表 1 の Human5 のように人間プレイヤーのトラフィックでも正の傾きを示す場合がある．ゆらぎによる Bot 判別手法として用いる場合に誤検知率が高くなる可能性がある．また，この結果を用い，人間のプレイヤーのトラフィックより傾きの値が小さいものを Bot とする手法を使用したとしても，Bot に対しランダムな成分を多く入力するようにルーチンを組めば，容易に回避方法を確立できる可能性がある．また，トラフィックの送信周期としては無いはずの周期の短い波が多く観測された．これはサンプリング周期を短くしすぎた為，その分のノイズが発生していると考えられる．図 8，図

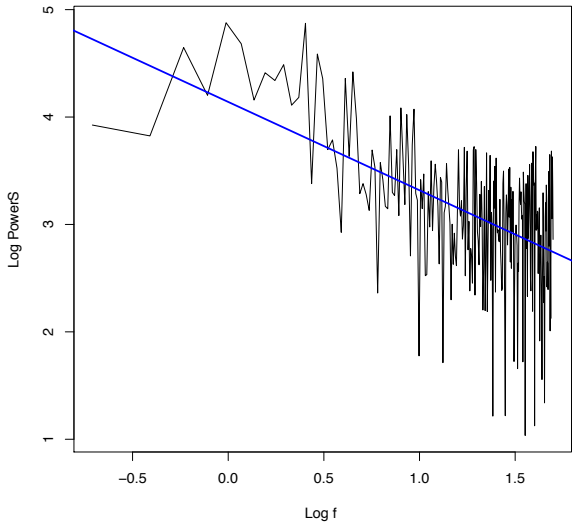


図 10 Bot:2Hz, サンプル数 512 におけるパワースペクトル分布

9 を見ると，本実験においてのゆらぎ性質を表す近似直線の傾きにおいてその部分の成分が大きく影響していることがわかる．この高周波成分を，観測しなかった場合にゆらぎがどうなるか検証を行った．

3.2.4 サンプリング周期を大きくにした場合

前節の実験を考察した結果，前回の方法では，サンプリング周期が短すぎる為，解析結果においてノイズである高周波成分が多く含まれているのではないかと考えられた．今実験では，この成分による影響を少なくするために，サンプリング周期を大きくして実験を行った．実験環境は前の実験と同じくデータセット A，B のフィルタリング作業及び解析作業は，以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセットのサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには R version 2.11.0 を使用し解析を行った．その結果を表 3 に示す．

サンプリング周期 1 秒から 10 秒のある程度大きいサンプリング間隔でサンプリングした結果，Bot におけるゆらぎの傾きは人間の場合より大きくなるというデータが出た．この Bot の傾きが，人間の傾きより大きくなるという結果は，前節での仮説の通りとなる．次にさらにサンプリング

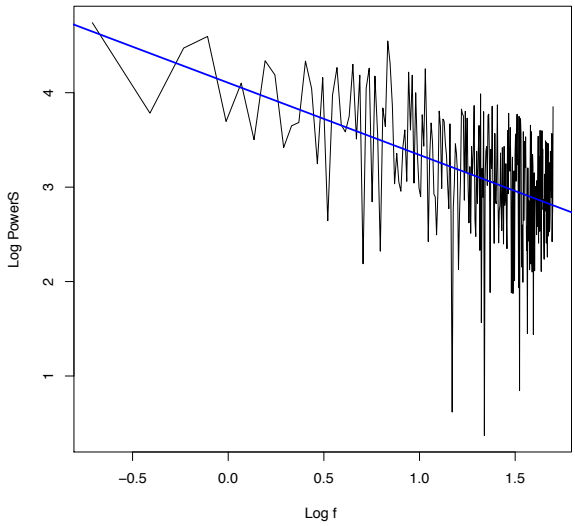


図 11 Human:2Hz, サンプル数 512 におけるパワースペクトル分布

周期を大きくしたときの傾きを観測した結果を表 3 に示す．表 3 においては，だいたいどのデータも 1〜3 秒では差が確認できる．しかし，3 秒の以上サンプリング間隔でサンプリングを行なった際には，差が確認できなかった．その中でもサンプリング周期２秒のときに一番差が現れそうであることがわかった．そこで，サンプリング周波数 2 秒において，サンプル数を変化させていったところ，サンプル数 512 から差が見られ始めた．サンプル数を増やしていくとサンプリング数 2,048 から差が小さくなり，4,096 では差がなくなっている．サンプリング数が大きければ大きいほど精度が高くなると考えられるのだが，結果はそうではない．この理由は 1 つ考えられる．1 時間単位の周期では人間と Bot の変化がなく，サンプリング数が多ければ多いほどこの部分が含まれるため両者に差がない状態になっていると考えられる．

これらの実験結果と考察から，サンプリング周期 2 秒，サンプリング数 512 のデータを閾値とし，このデータにおける「ゆらぎ」において，人間のトラフィックと Bot のトラフィックとを判別できる手法として本節では提案する．

3.3 ウェーブレット解析による Bot 検知手法

今回ウェーブレット解析において，前章で述べた仮説である「時間成分込みでの Bot 独自の波形，もしくは人間独自の波形があるのではないか」ということと，「Bot のパケット送信タイミン

表 3 サンプリング周波数とゆらぎの傾き (サンプリング周波数 1Hz)

サンプリング周期	1 秒	2 秒	3 秒	4 秒	5 秒	6 秒	7 秒	8 秒	9 秒	10 秒
Bot1	-0.69	-0.82	-0.89	-0.92	-0.98	-0.87	-0.84	-0.72	-0.84	-0.70
human1	-0.43	-0.76	-0.79	-0.85	-0.90	-0.84	-0.88	-0.85	-0.83	-0.80
サンプリング周期	1 秒	2 秒	3 秒	4 秒	5 秒	6 秒	7 秒	8 秒	9 秒	10 秒
Bot3	-0.62	-0.94	-0.92	-0.90	-0.92	-0.91	-0.86	-0.82	-0.87	-0.84
human3	-0.49	-0.56	-0.55	-1.27	-1.34	-1.55	-1.55	-1.55	-1.85	-1.57

グは定期的であり，不定期なものはランダムなノイズである高周波成分として出力されるため，ウェーブレット解析における Detail1 の成分の割合が人間のトラフィックに比べ小さいのではないか」という二つの仮説を検証するため，トラフィックデータに対しウェーブレット解析を行う．今回の検証では，基本的な構造のウェーブレット解析であるハールウェーブレット解析を用いて解析を行った．

3.3.1 ハールウェーブレット解析

ハールウェーブレットは，基本的な離散ウェーブレットの一つで，ハンガリーの数学者 Alfred Haar によって示された．これは，$2n$ の長さを持つ数列が入力されると，隣接した値の差分と和を求めるものである．この処理は再帰的に行われ，和の数列は次の処理の入力となる．最終的には，$2n-1$ の差分値と一つの和の値を得る．この単純な離散ウェーブレットは，ウェーブレットの一般的な特性を示している．離散ウェーブレットの計算量は $O(n)$ である．また，この変換は，時間及び周波数の両方の特性をつかむことができる．これら 2 つの特徴は，FFT と比較した場合の離散ウェーブレットの大きな特徴である．

もっとも一般的な離散ウェーブレットは，ベルギーの数学者 Ingrid Daubechies（ドベシーもしくはドブシー）によって 1988 年に証明された．この証明では，解像度が以前のスケールの 2 倍となっていく漸化式によって，もっとも密にサンプリングされたマザーウェーブレットを生成している．彼女の講義資料には，Daubechies wavelet と呼ばれるウェーブレットファミリーが提供されており，その中の最初のウェーブレットがハールウェーブレットである．

このハールウェーブレットは，先に述べたようにもっとも基本的なウェーブレット解析であり，その構造は最も単純である．

図 12 wavelet tool box

$$\Phi(x) = \begin{cases} 1(0 \leq x < 1) \\ 0(otherwise) \end{cases}$$

$$\Psi(x) = \begin{cases} 1(0 \leq x < \frac{1}{2}) \\ -1(\frac{1}{2} \leq x < 1) \\ 0(otherwise) \end{cases}$$

としこれを用いることで小さな波 (wavelet) に分割し，解析を行うというものである．

3.3.2 実験

本実験は，オフライン解析を行った．実験では，Kuan-Ta Chen 氏が公開しているラグナロクオンラインにおける人間プレイヤーのトラフィックと Bot プレイヤーのトラフィックに関する pcap [25] ファイルのデータセットを使用した [24]．このデータセットは，文献 [16] において Bot 検知に使用されたデータセットである．文献 [24] で公開されている人間プレイヤー 3 人分のトラフィックが 6 セットと 2 種類の Bot による Bot プレイヤーのトラフィックが 8 セットが含まれている．これらをそれぞれ，Human1 から Human6，Bot のトラフィックを Bot1 から Bot8 とし，データセット A，B を作成した．今回，データセット A，B においてウェーブレット解析を行う為，単位時間あたりのクライアントのパケット送信回数をサンプリングしたものを使用した．実験環境は同

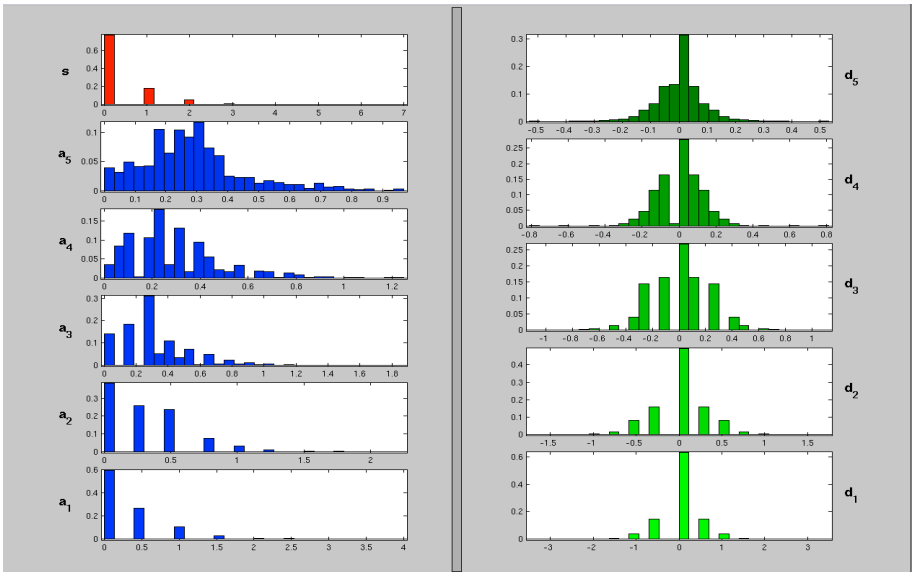


図 13 B2 におけるハールウェーブレット解析時の各成分グラフ

じくデータセット A，B のフィルタリング作業及び解析作業は，以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセット A，B のサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには MATLAB version7.9.0.0.529 [28]，wavelet tool box12 を使用し解析を行った．

3.3.3 実験結果

データセット A，及び B に関し単位時間あたりのクライアントのパケット送信回数をサンプリングしたものに対しハールウェーブレット変換を行った．解析レベルは 5 であり，この場合における Detail 成分をそれぞれ d1, d2, d3, d4, d5, Approximation 成分を a1, a2, a3, a4, a5 とし，これらに関して観測を行った．サンプリング周波数 0.1 サンプル数 16,384 を行なった結果を図 13 に示す．

3.3.4 考察

実験の結果，Bot のトラフィックにおける Approximation 成分には，仮説である時間分解機能込みでの Bot 独自の波形，もしくは人間独自の波形特異な特徴を見出すことはできなかった．次

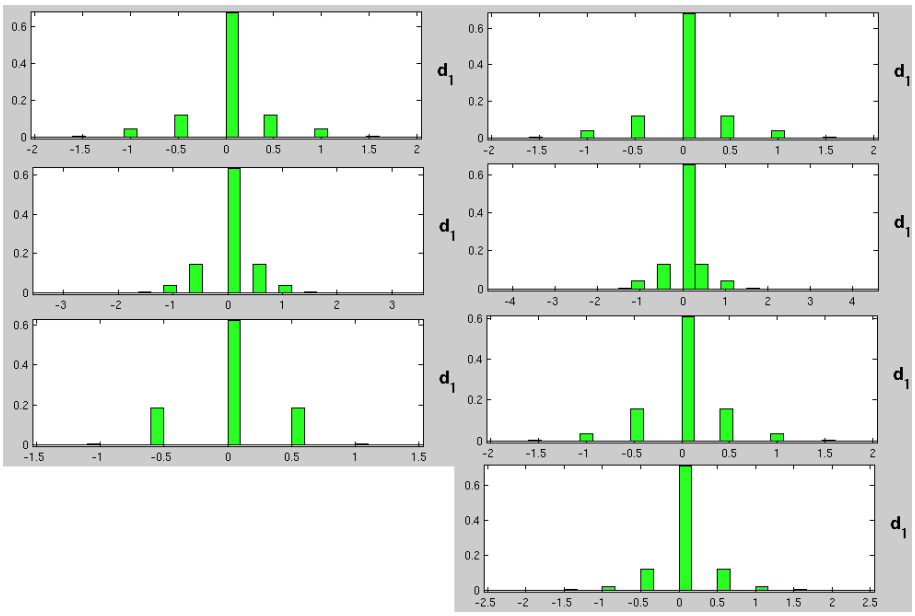


図 14 Bot のトラフィックにおける D1 成分

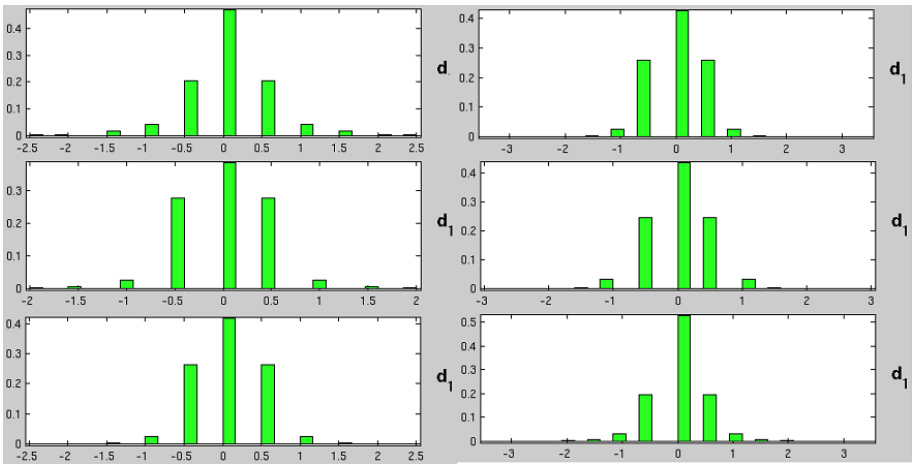


図 15 人間のトラフィックにおける D1 成分

表 4 人間プレイヤートラフィックにおける Detail1 成分における平均偏差

データセット	Human1	Human2	Human3	Human4	Human5	Human6
Detail1 の平均偏差	0.345	0.3204	0.3209	0.3181	0.3243	0.2876

表 5 bot プレイヤートラフィックにおける Detail1 成分における平均偏差

データセット	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7
Detail1 の平均偏差	0.212	0.2312	0.2055	0.2209	0.2415	0.1725	0.3533

にサンプリング間隔を 0.1 秒としてウェーブレット解析を行ったところ，人間のプレイヤーのトラフィックにおいては，結果の Detail1 成分に大きな割合の信号が多く含まれていることがわかり，Bot のトラフィックにおいては a2 以降の成分に大きな割合の信号が多く含まれていることがわかった．これらの結果から Detail1 成分に着目し，Detail1 成分が大きければ人間のプレイヤーのトラフィックであり，Detail1 成分が小さければ Bot プレイヤーとする．このとき，Detail1 成分の大きさは平均偏差にて判別する．平均偏差は

$$\mathcal{M}eandev = -\sum_{i=1}^{n} \left| X_i - \bar{X} \right| /n$$

であり，平均値からどれだけ分散しているかを測るものである．表 4 及び表 5 から読み取れるように，平均偏差は Bot のトラフィックにおいては 2.3 前後の値になっており，人間のトラフィックにおいては，3.2 前後となっており「Bot のパケット送信タイミングは定期的であり，不定期なものはランダムなノイズである高周波成分として出力されるため，ウェーブレット解析における Detail1 の成分の割合がが人間のトラフィックに比べ小さいのではないか」との仮説は確からしいことがわかった．この結果から，これらの実験結果と考察から，サンプリング周期 2 秒，サンプリング数 512 のデータを閾値とし，このデータにおけるウェーブレット解析にける「Detail1 成分」において，人間のトラフィックと Bot のトラフィックとを判別できる手法として本節では提案する．

3.4 エントロピーによる Bot 検知手法

今節では，2 章での仮説の検証として，Bot プレイヤーのトラフィックと人間プレイヤーのトラフィックにおいてエントロピー分析を行なった．

3.4.1 実験

本実験は，オフライン解析で解析を行った．今までの実験のとおり，Kuan-Ta Chen 氏が公開しているラグナロクオンラインにおける人間プレイヤーのトラフィックと Bot プレイヤーのトラフィックに関する pcap ファイルのデータセットを使用した [24]．このデータセットは，文献 [16]

において Bot 検知に使用されたデータセットである．文献 [24] で公開されている人間プレイヤー 3 人分のトラフィックが 6 セットと 2 種類の Bot による Bot プレイヤーのトラフィックが 8 セットが含まれている．これらをそれぞれ，Human1 から Human6，Bot のトラフィックを Bot1 から Bot8 と表記し，データセット A,B を作成した．今回，データセット A，B においてウェーブレット解析を行う為，単位時間あたりのクライアントのパケット送信回数をサンプリングしたものを使用した．実験環境は同じくデータセット A，B のフィルタリング作業及び解析作業は，以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセット A，B のサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには MATLAB version7.9.0.0.529 を使用し解析を行った．

3.4.2 実験結果

解析には MATLAB の entropy 関数を用いて解析を行った結果を表 6，表 7 に示す．

表 6 人間プレイヤートラフィックにおけるサンプリング間隔 0.1 秒，サンプル数 16,384 でのエントロピー

データセット	Human1	Human2	Human3	Human4	Human5	Human6
エントロピー (bit)	0.943	0.968	0.954	0.947	0.960	0.865

表 7 Bot プレイヤートラフィックにおけるサンプリング間隔 0.1 秒，サンプル数 16,384 でのエントロピー

データセット	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7
エントロピー (bit)	0.761	0.779	0.736	0.778	0.832	0.855	0.947

3.4.3 考察

実験結果を行った結果，サンプリング周期を短く，かつサンプリング数を大きくしていった場合において，両者には差異があるように見られるという結果がでた．また，実験結果から，サンプリング周期が 0.1 秒以下のときには，ある程度判別でき，それ以上のときには差異がみられるとはいい難い結果となっているのでサンプリング間隔を 0.1 秒に注目し，計算機での扱いを考慮

し 2 の倍数でサンプル数を変化させていったところ，サンプル数が 16,384 から 32,768 の場合において，人間のプレイヤーのトラフィックと Bot プレイヤーのトラフィックとの間に差異がはっきりと見られるような実験結果となった．この実験の結果からサンプリング間隔 0.1 秒かつサンプル数 16,384 をエントロピー手法の閾値とし，この場合におけるエントロピーの違いを今回 Bot トラフィックと人間プレイヤートラフィックを分ける手法として提案する．

4. 手法の検証と検定

Kuan-Ta Chen 氏が公開しているデータセットにおいて検証を行なっただけでは汎用的な Bot 指標になるとは言えない．また，データセットが少なすぎて統計的に人間プレイヤーのトラフィックと Bot プレイヤーのトラフィックとが有意的な差があるとは言えない．本章では，われわれで独自に取得したトラフィックデータを用いて，3 章で人間プレイヤーと Bot プレイヤーの違いが出た点について，再度検証を行い．結果に違いが現れた場合，そのデータセットに統計的に有意な差があるか検定する．

4.1 実験環境

本実験において，サーバのプログラムは JAthena [28] を使用した．この JAthena はラグナロクオンラインに似せて作られたオープンソースのプログラムであり，ラグナロクオンラインとほぼ同じ仕様のゲームであり，Bot もラグナロクオンラインの Bot が使用できる．クライアントプログラムには，ラグナロクオンラインのクライアントを使用し，JAthena に接続した．トラフィックは，学校内の有線ネットワークにおいて，サーバ側で取得し，ほとんど遅延が起こらないネットワーク環境において実験を行った．実験では，7 人の方に協力していただき，トラフィックを取得した．Bot プレイヤーのトラフィック取得に関しては，Open-kore [29] という Bot を使用した．トラフィックの取得は TCPdump で行った．この取得したプログラムを前章で用いた実験環境で解析を行った．また，これらのデータセットを人間のプレイヤーのトラフィックをデータセット C，Bot のトラフィックのデータセットを D とする．

4.2 t 検定

検定には t 検定を用いた．t 検定は，t 検定は 2 つのグループの平均の差が偶然誤差の範囲内にあるかどうかを調べるものである．1 つ目はデータに対応があるときの t 検定「それぞれの被験者が 2 つのテスト A，B を受けたときの平均点の比較」のようにグル-プ A のデータとグループ B のデータに同一被験者のデータという対応があるとき，これら 2 つのテストの平均点の差に有意差があるかどうかは「対応があるときの t 検定」を用いる．

データに対応があるときは，単にデータの個数が等しいだけでなく，対応するデータ間の差を求めることができるので，それらの差の平均と分散から有意差を判断できる．次に，データに対応がないときは，データの個数が等しいときもある．これらの場合は，各グループの平均と分散だけから t 検定を行うことになり，さらに分散がほぼ等しいと見なせる場合と分散が等しいとは

見なせない場合に応じて，各々「分散が等しいときの t 検定」「分散が等しくないときの t 検定」を適用する．分散が等しいかどうかの判断は F 検定によって行う．t 分布において「外側 5 パーセントの範囲にあれば同一母集団からの標本ではなく，有意差があると考える」．95 パーセント信頼区間の外側に来る確率を p (probability) とする．この場合，p>0.1 の場合データ間には有意差はない．$0.05 < p \leq 0.10$ において有意傾向がある．p<0.05 の場合有意差があるといえる．

このように t 検定は仮定とデータセットとの間に有意な差があるかどうかを検定するものである．Bot のトラフィックの傾きの平均値と人間のトラフィックの傾きのデータセットとを比較し，この傾きに有意な差が見られるかどうか検定を行った．

4.3 ゆらぎ手法の検証と検定

今回，前章で示した最も人間のトラフィックと Bot のトラフィックとの違いがわかるサンプリング間隔及びサンプリング数，つまりサンプリング間隔を 2 秒，サンプリング数 512 個において，データセット C（人間のプレイヤーのトラフィック）とデータセット D (Bot プイヤーのトラフィック) のデータセットにおいて傾きに違いが現れるか検証を行った．解析環境は以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセットのサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには R version 2.11.0 を使用し解析を行った．

4.3.1 ゆらぎ手法の解析結果

データセット C，D において解析した結果を表 8 に示す．解析を行った結果，前章の結果とは異なる値となっている．前章では，Bot プレイヤーのトラフィックの傾きの場合は傾きが-0.85 以下であったのに対し，今回は 0.6 付近が平均の値となっている．人間プレイヤーのトラフィックの傾きも前章の実験においては，0.9 付近であったのに対し，今回は 1 以上の値が多くなっている．これらの結果から，ほぼ同じようなゲームであってもゲームデザインのアップデートによるゲームデザインの変化によってゆらぎの傾きが違ってしまうということが推測される．しかし，全体の結果として Bot のプレイヤーのトラフィックにおける傾きの値は，Bot のプレイヤーのトラフィックの傾きの値に比べて大きくなる．これらの値の違いに有意な差が現れているならば，ゲームコンテンツのアップデートがあったとしても，サンプルのトラフィックを集めるだけで Bot の検知手法として適用可能であると考えられる．今回，この結果について検定を行なった結果，検定統計量 P は分散が等しいと仮定した場合 $P = 6.064e^{-10}$ となり，分散が等しくないと仮定した場合 $P = 1.650e^{-9}$ となった．どちらの場合も P が 0.01 以下であり，有意水準 1 パーセントで統計的

表 8 データセット C，D における近似直線の傾き

データセット C	human1	human2	human3	human4	human5	human6	human7	human8
傾き	-0.495	-0.579	-0.844	-0.551	-0.618	-0.763	-0.713	-0.614
データセット C	human9	human10	human11	human12	human13	human14	human15	human16
傾き	-0.570	-0.638	-0.863	-0.849	-0.711	-0.9477	-0.767	-0.428
データセット D	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7	bot8
傾き	-0.558	-1.414	-1.199	-1.408	-1.295	-1.160	-1.368	-1.332
データセット D	Bot9	Bot10	Bot11	Bot12	Bot13	Bot14	Bot15	Bot16
傾き	-1.308	-1.429	-1.220	-1.373	-1.210	-1.165	-1.223	-1.453

に有意差があるという結果となった．3 章の結果も踏まえると，Bot プレイヤーのトラフィックの周波数とパワースペクトル平面上における最小二乗法での近似直線の傾きは，人間プレイヤーのトラフィックに比べて大きくなるということは確かであり，予め人間プレイヤーのトラフィックのサンプルを取ることで Bot 検知の手法として使用できうると言える．

4.4 エントロピー手法の検証と検定

今回，前章で示した一番人間のトラフィックと Bot のトラフィックとの違いがわかるサンプリング間隔及びサンプル数，つまりサンプリング間隔 0.1 秒，サンプル数 16,384 において，データセット C（人間のプレイヤーのトラフィック）とデータセット D (Bot プレイヤーのトラフィック) のデータセットにおいて傾きに違いが現れるか検証を行った．解析環境は以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセットのサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには MATLAB version7.9.0.0.529 を使用し解析した．

4.4.1 エントロピー手法の解析結果

データセット C，D において解析した結果を表 9 に示す．解析の結果，2 章の仮説や 3 章の実験結果と違い，人間プレイヤーのトラフィックのエントロピーより，Bot プレイヤーのエントロピーの方が高くなるという結果になった．しかし，結果をよく見ると人間と Bot の間には差異があるように見えた．t 検定を行なった結果，検定統計量 P は分散が等しいと仮定した場合 $P = 1.909e^{-15}$ となり，検定統計量 P は分散が等しくないと仮定した場合 $P = 8.244e^{-13}$ となった．どちらの場合も P が 0.01 以下であり，有意水準 1 パーセントで統計的に有意差があるという結果となった．この結果により人間のプレイヤーのトラフィックと Bot プレイヤーのトラフィックはエント

表 9 データセット C，D におけるエントロピー

データセット C	human1	human2	human3	human4	human5	human6	human7	human8
エントロピー (bit)	0.204	0.158	0.218	0.221	0.443	0.392	0.367	0.469
データセット C	human9	human10	human11	human12	human13	human14	human15	human16
エントロピー (bit)	0.312	0.277	0.219	0.330	0.204	0.421	0.342	0.565
データセット D	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7	Bot8
エントロピー (bit)	0.983	0.729	0.890	0.841	0.800	0.795	0.891	0.830
データセット D	Bot9	Bot10	Bot11	Bot12	Bot13	Bot14	Bot15	Bot16
エントロピー (bit)	0.748	0.806	0.770	0.827	0.763	0.816	0.808	0.724

ロピーにおいて差があることが証明された．3 章の解析結果も踏まえると，エントロピーは人間，Bot 各々に特有のエントロピーがあるのではないかと考えられる．また，3 章の Bot のエントロピーは 0.82 前後であることから Bot は 0.82 から 0.98 の値に，母集団がある可能性がある．この可能性については，今回の実験のみではわからないので，今後の検討が必要である．

4.5 ウェーブレット手法の検証と検定

前章において，人間のプレイヤーのトラフィックは，Bot プレイヤーのトラフィックに比べ，サンプリング間隔 0.1 サンプリング数 16,384 にてハールウェーブレット解析におけるマザーウェーブレット Detail1 成分が大きくなるという結果が見られた．データセット C (人間のプレイヤーのトラフィック) とデータセット D (Bot プレイヤーのトラフィック) のデータセットにおいて傾きに違いが現れるか検証を行った．解析環境は以下の計算機で行った．OS は Mac OS X バージョン 10.5.8，プロセッサは 2 GHz Intel Core Duo，メモリは 2GB DDR2 SDRAM である．また，データセットのサンプリングには libpcap を用いて C 言語でプログラムを作成し使用した．また解析ソフトには MATLAB version7.9.0.0.529，wavelet tool box を使用し解析を行った．

4.5.1 ウェーブレット手法の解析結果

データセット C，D においてサンプリング周波数 0.1，サンプル数 16,384 にて解析を行なった結果を表 10 に示す．解析の結果，2 章の仮説や 3 章の実験結果と違い，Bot プレイヤーのトラフィックの Detail1 成分は人間のものに比べ大きくなるという結果になった．しかし，結果をよく見るとエントロピー手法の場合と同じく，人間と Bot の間には差異があるように見える．今回の結果に対し t 検定を行なった結果，検定統計量 P は分散が等しいと仮定した場合 $P = 8.056e^{-10}$ となり，分散が等しくないと仮定した場合 $P = 1.499e^{-8}$ である．どちらの場合も P が 0.01 以下であ

表 10 データセット C，D における Detail1 成分

データセット C	human1	human2	human3	human4	human5	human6	human7	human8
Detail1 成分における平均偏差	0.031	0.023	0.034	0.034	0.090	0.076	0.070	0.97
データセット C	human9	human10	human11	human12	human13	human14	human15	human16
Detail1 成分における平均偏差	0.055	0.045	0.035	0.058	0.031	0.056	0.088	0.13
データセット D	Bot1	Bot2	Bot3	Bot4	Bot5	Bot6	Bot7	bot8
Detail1 成分における平均偏差	0.441	0.168	0.248	0.217	0.197	0.198	0.247	0.220
データセット D	Bot9	Bot10	Bot11	Bot12	Bot13	Bot14	Bot15	bot16
Detail1 成分における平均偏差	0.199	0.184	0.215	0.182	0.202	0.197	0.173	0.173

り，有意水準 1 パーセントで統計的に有意差があるという結果となった．この結果により人間のプレイヤーのトラフィックと Bot プレイヤーのトラフィックはエントロピーにおいて差があることが証明された．このことから，人間も Bot もトラフィックのウェーブレット解析における Detail1 成分において特徴的な数値を示すが，3 章の結果をふまえると，その値はゲームにより変動すると考えられる．この方法を Bot 検知法とするには，人間プレイヤーのトラフィックサンプルもしくは Bot のトラフィックサンプルを予め取得し，そこから Bot の検証すべきである．

4.6 まとめ

本章では，3 章での解析結果から得られた Bot 判別に最適と思われるパラメータにおいて，各手法の検証を行なった．その結果，どの手法においても仮説とは異なるものの Bot と人間を区別することが可能であるということがわかった．中でもゆらぎ手法は，仮説の理論にもとづいた解析結果があらわれており，人間のトラフィックのサンプルさえあれば Bot 検知ができるので，MMORPG のサービス開始時期から使用する Bot 判別手法として適していると考えられる．

5. ゆらぎ手法での Bot 検知

前章にて Bot 検知手法に関する検証を行なった結果「ゆらぎ手法」にて Bot 検知することが有効であることが確からしいということがわかった．本章では,「ゆらぎ手法」を使用し，MMORPG のサービス開始時点からの Bot 検知を想定した実験を行なった．MMORPG は前章と同じ JAthena を使用し，Bot も Openkore を使用した．トラフィックは，学校内の有線ネットワークにおいてサーバ側で取得し，ほとんど遅延が起こらないネットワーク環境において実験を行った．実験では，6 人の方に協力していただき，トラフィックを取得した．Bot プレイヤーのトラフィック取得に関しては，Open-kore を使用した．トラフィックの取得は TCPdump で行った．この取得したプログラムを前章で用いた実験環境で解析を行った.「ゆらぎ手法」の使い方としては，まず人間のプレイヤーのサンプルを集めそれをもとに指標を設定する．本実験では，その指標を前実験におけるデータセット C の最大値である-0.863 に着目し，これよりも値が大きければ Bot，-0.863 以下ならば人間とした．これらのデータセットを人間のプレイヤーのトラフィックをデータセット E，Bot のトラフィックのデータセットを F とする．このデータセットにおいて，Bot を Bot として，人間を人間として検知できうるか検証を行なった．

5.1 Bot 検知実験

Bot のトラフィック 12 個のデータを持つデータセット E と，人間のトラフィック 12 個のデータを持つデータセット F を混ぜたデータセット G をつくり，データセット G において Bot 検知を行なった．その結果を表 11 に示す．表 11 の結果から，誤検知率が約 20 パーセントとなり．検知精度は約 80 パーセントとなった．しかし，その内訳を見ると誤検出はあるが，検出漏れはないという結果となった．特に誤検出が 40 パーセントとなっており，閾値の設定に問題があったと考えるのが妥当であると言える．仮に傾きの閾値の値を-0.9 とした場合は，検知精度約 92 パーセント，誤検出約 16 パーセント，検出漏れはなしとなる．また，-1.1 を閾値とした場合，検知精度約 92 パーセント，誤検出約 8 パーセント，検出漏れ約 8 パーセントとなる．人間のサンプルを取得し，閾値とする場合には，サンプルの値よりも大きい値を閾値としたほうがよいと推測される．また，本実験での人間のデータセットの中で一番傾きが大きい値である-1.117 という値は 24 個中 11 番目に傾きが大きい値であり，その次が 13 番目の-0.906 という値でそれより傾きが小さい値に Bot のトラフィックがないことから，トラフィックを取得し傾きの値が大きいものから順に管理者による声かけをおこなうという方法もよいと推測される．この考察を踏まえ，実運用にて「ゆらぎ手法」を適用するには，人間のサンプルを取得し，傾きの閾値をある程度大きめに設定することで，誤検出の割合を抑え Bot のみを検出することが可能となる．また，リアルタイム

表 11 データセット G における Bot 検知の結果

データセット G	data1	data2	data3	data4	data5	data6
傾き	-0.644	-0.788	-0.700	-0.617	-1.117	-0.883
判定	human	human	human	human	Bot	Bot
正解	human	human	human	human	human	human
正誤判定	○	○	○	○	×	×
データセット G	data7	data8	data9	data10	data11	data12
傾き	-0.906	-0.885	-0.624	-0.887	-0.637	-0.250
判定	Bot	Bot	human	Bot	human	human
正解	human	human	human	human	human	human
正誤判定	×	×	○	×	○	○
データセット G	data13	data14	data15	data16	data17	data18
傾き	-1.176	-1.178	-1.101	-1.311	-1.413	-1.176
判定	Bot	Bot	Bot	Bot	Bot	Bot
正解	Bot	Bot	Bot	Bot	Bot	Bot
正誤判定	○	○	○	○	○	○
データセット G	data19	data20	data21	data22	data23	data24
傾き	-1.178	-1.101	-1.312	-1.129	-1.236	-1.084
判定	Bot	Bot	Bot	Bot	Bot	Bot
正解	Bot	Bot	Bot	Bot	Bot	Bot
正誤判定	○	○	○	○	○	○

で 2 秒毎のクライアントの ack パケット以外の出力回数を記録することができるのであれば，1 ユーザーあたり 5 k〜10 k バイト程度のデータサイズで検知法が適用できる．そのため，数万人の規模の MMORPG でも数百メガバイトの記憶媒体で検知することが可能である．また，実行時間も R で 1 秒以下であるので，実行時間の上でも実用可能であると考えられる．

6. おわりに

本稿では，ゲーム開始時点からある程度判別可能な Bot 判別手法の提案を行なった．今回はトラフィック分析，特に MMORPG においてサービス開始時点から使用できうる Bot 検知の手法として，パケットの発生タイミングに注目したトラフィック成分分析により人間のトラフィックと Bot のトラフィックとを分け，Bot を検知する手法をいくつか提案しその実証実験を行った．その結果，本稿で提案した手法に関しては人間のプレイヤーのサンプルを取得し，それをもとに基準を設ければ Bot の検知手法として実用可能であると考えられる．その中でもゆらぎ手法は 2 章で立てた仮説の通りの結果となっており，ゆらぎ分析の分析結果は統計的有意差がみめられ，5 章において閾値の調整により Bot 検知が可能であることがわかったことから，本論文ではとくにゆらぎ分析法を MMORPG のサービス開始時期から使用できうる Bot 判別手法として適しているとし,「ゆらぎ手法」がゲーム開始時点から Bot 検知手法として使用しうる手法であると結論する．

6.1 他の種類のオンラインゲームへの適用

今回提案した方法を他の種類のネットゲームに適用した場合を考察する．今回提案したトラフィックによる Bot 検知の方法は，MMORPG における Bot の定期的なパケット送信タイミングに着目したものであった．他のオンラインゲームにおける Bot の検知も可能であるか検証する．MMORPG と同じく多数のユーザが存在する FPS(First Person shooter) の場合を考察する．そのゲーム性から MMORPG に比べてユーザの割合が少ないものの Bot が存在する．これらの Bot もゲーム性の崩壊に繋がるので，FPS においても MMORPG と同じく，Bot を検知しなければいけないという課題がある．FPS において今回の Bot 検知が使用できうるか考察する．FPS における Bot も MMORPG における Bot と同様に，ゲームサーバから来たパケットに対して，返信パケットを返すことと定期的にパケットを送信することを繰り返すというものである．しかし，MMORPG に比べ FPS はアイテムを使う等の動作が少ないため，なにかを繰り返えす行動は MMORPG の場合と比べ少なくなる．そのため，本研究の Bot 検知方法がそのまま適用できるとは考えられない．だが，少いとはいえ定期的な行動を繰り返すので，本研究の方式を適用することでなんらかの違いが見えてくる可能性もある．その他のゲームでは，将棋やカードゲームなどあまり Bot を使う意味のないゲームデザインとなっているので，Bot に関する被害も少ないのでここでは考察を割愛する．

6.2 実運用への適用

次にこれらの手法がMMORPGサービスの実運用に実際に使われた場合について考える．解析に使うトラフィックはどの手法もだいたい約20分程度のトラフィックが必要である．MMORPGサービスは大体，βテストで1万人，正式運営で数万人のユーザが遊戯する．そのような状態で1ユーザあたり20分のトラフィックを取り解析を行うには膨大な計算機の記憶容量が必要になってしまう．しかし，今回のようにトラフィックのタイミングに着目した場合，リアルタイムにフィルタリング及びサンプリングできるシステムがあれば，一人当たり5 k〜10 kバイト程度の記憶容量でBot検知可能である．各計算の実行時間に関しては，実験環境で述べた計算機では1秒前後である．この程度のデータ量と実行時間であれば数万人の規模のMMORPGでも数百メガバイトの記憶媒体で検知することが可能である．検知できたならば，管理者による声かけによりBotかどうか判別することでMMORPGのゲームサービス開始時点におけるBot検知および対策が可能となる．

6.3 今後の課題

今後の研究においては，ネットワークの遅延によるトラフィックの変化を考慮に入れていないため，ネットワークの遅延を含めた際にトラフィックがどのように変化していくか検証することが必要である．また，本稿で提案した分析方法は，ゲーム内の局所的な行動とは関係ないことからBot製作者による対策は容易ではなく，半永久的にBot検知に使用できうる可能性がある手法である．しかし，完全にBot製作者による対策が回避できるとは限らない．ゆらぎ手法を例に挙げると，ランダムな成分をBotに加えれば，理論的にはゆらぎ手法を回避できうる．そのため，今後の課題として各手法に対する更なる検討すべきである．例えば，ゆらぎ手法ならばランダムな成分を加えすぎると白色雑音に近いゆらぎになるので，人間のトラフィックより大きくなるトラフィックと合わせて，人間のトラフィックの値より小さすぎるものを検知する．このように各手法について更に検討を行い洗練させていくべきであると考える．また，本研究では考え付くことができなかった他のトラフィックでのBot検知方法を確立し，多角的にBotを検知できるように，Bot検知法を増やしていくことも必要である．トラフィックでのBot検知以外にも

- 自動的なBot検知可能
- ゲーム内の局所的な行動とは関係ない
- ゲームの進行の支障にならない
- ゲーム開始時点から使用できうる

以上の 4 点を擁する Bot 検知手法を見つけ出し，より多角的に Bot 検知を行なっていかなければならない．そして，ゲームサービスが開始されしばらく時間が経過し，Bot を手に入る状況になったら，Bot を一つ一つ解析し，[17] の文献のような高精度な Bot 検知手法を確立していかねばならない．

謝辞

本研究における全過程を通じて，懇切なるご指導，ご鞭撻を賜りました本学情報科学研究科の山口英教授，関浩之教授並びに，門林雄基准教授に厚く感謝の意を表します．本研究を進めるにあたり櫨山寛章助教，奥田剛助手には様々な意見を頂き誠にありがとうございました．また，宮本大輔，岡田和也さんには実験，実装に大いにお力添えを頂き，お礼を申し上げます．研究の際，インターネット工学講座の皆様には実験の際大変お世話になりました．

参考文献

[1] Leigh Achterbosch, Robyn Pierce, and Gregory Simmons. Massively multiplayer online role-playing games: the past, present, and future. *Comput. Entertain.*, Vol. 5, pp. 9:1–9:33, March 2008.

[2] Newzoo BV. Graphs mmo revenues. `http://www.gamesindustry.com/about-newzoo/todaysgamers_graphs_MMO`, 2009.

[3] Piers Harding-Rolls. Subscription mmogs - mixed fortunes in high risk game. `http://www.screendigest.com/reports/2010822a/10_09_subscription_mmogs_mixed_fortunes_in_high_risk_game/view.html`, September 2010.

[4] ガンホーオンラインエンターテイメント株式会社. ラグナロクオンライン公式サイト. `http://www.ragnarokonline.jp/`.

[5] エヌ・シー・ジャパン株式会社. リネージュ公式サイト. `http://lineage.plaync.jp/`.

[6] 株式会社インカインターネット. nProtect:GameGuard. `http://www.gameguard.jp/`.

[7] Steven Gianvecchio, Zhenyu Wu, Mengjun Xie, and Haining Wang. Battle of botcraft: fighting bots in online games with human observational proofs. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pp. 256–268, New York, NY, USA, 2009. ACM.

[8] Hyungil Kim, Sungwoo Hong, and Juntae Kim. Detection of auto programs for mmorpgs. *AI 2005 Advances in Artificial Intelligence*, Vol. 3809, pp. 1281–1284, 2005.

[9] Ruck Thawonmas, Yoshitaka Kashifuji, and Kuan-Ta Chen. Detection of mmorpg bots based on behavior analysis. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pp. 91–94, New York, NY, USA, 2008. ACM.

[10] Kuan-Ta Chen, Andrew Liao, Hsing-Kuo Kenneth Pao, and Hao-Hua Chu. Game bot detection based on avatar trajectory. In *Proceedings of the 7th International Conference on Entertainment Computing*, ICEC '08, pp. 94–105, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] Kuan-Ta Chen, Hsing-Kuo Kenneth Pao, and Hong-Chung Chang. Game bot identification based on manifold learning. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 21–26. ACM, 2008.

[12] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. In *EUROCRYPT*, pp. 294–311, 2003.

[13] Roman V. Yampolskiy and Venu Govindaraju. Embedded noninteractive continuous bot detection. *Comput. Entertain.*, Vol. 5, pp. 7:1–7:11, March 2008.

[14] RMT . RMT Real Money Trade. `http://trade.netgame-rmt.jp/`.

[15] Kuan-Ta Chen, Hsing-Kuo Kenneth Pao, and Hong-Chung Chang. Exploiting MMORPG log data toward efficient RMT player detection. In *Proceedings of the 7th International Conference on Advances in Computer Entertainment Technology (ACE)*. ACM, November 2010.

[16] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Game traffic analysis: An MMORPG perspective. *Computer Networks*, Vol. 50, No. 16, pp. 3002–3023, 2006.

[17] Kuan-Ta Chen, Jhih-Wei Jiang, Polly Huang, Hao-Hua Chu, Chin-Laung Lei, and Wen-Chin Chen. Identifying mmorpg bots: a traffic analysis approach. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, ACE '06, New York, NY, USA, 2006. ACM.

[18] Tomoaki Nizuma, Atsuko Toba, Yukiko Yajima, Keiichi hisada, Shigeki Okino, and Fujio Suda. Application of $1/f^{\beta}$ fluctuation analysis of the zero cross and the energy envelope of music to its style analysis. *Journal of Advanced Science*, Vol. 19, No. 3/4, pp. 41–46, 2007.

[19] 河原剛一. 生体リズムゆらぎの機能的意義と 1/f ゆらぎの個体発生. *BME*, Vol. 8, No. 10, pp. 22–28, 1994 年 10 月.

[20] 橋場参生, 新井浩成, 大崎恵一. ゆらぎ信号を用いた電子機器制御技術. Technical report, 北海道立工業試験場, 2005 年 8 月.

[21] 小林淳史, 近藤毅, 廣川裕, 石橋圭介, 山本公洋. 選択的 sFlow による VoIP トラフィックのゆらぎ測定方法の提案. 電子情報通信学会技術研究報告 vol.106 no.495, CQ2006-87, pp. 47–52, 2007 年 1 月.

[22] Ingrid Daubechies. The wavelet transform, time-frequency localization and signal analysis. *Information Theory, IEEE Transactions on*, Vol. 36, No. 5, pp. 961 –1005, September 1990.

[23] 芦野隆一, 守本晃. ウェーブレット解析とその応用. Technical report, 大阪教育大学, 2003 年 2 月.

[24] Kuan-Ta Chen. Ragnarok Online Traffic Traces for Game Bot Identification. http://www.iis.sinica.edu.tw/~swc/traces/ro_bots/.

[25] Tcpdump/Libpcap. Tcpdump & Libpcap. http://www.tcpdump.org/.

[26] Apple Inc. Mac OS X Snow Leopard. http://www.apple.com/macosx/.

[27] The R Project. The R Project for Statistical Computing. http://www.tcpdump.org/.

[28] The MathWorks, Inc. MATLAB - The Language Of Technical Computing. http://www.mathworks.com/products/matlab/.

[29] Open Kore Developers. OpenKore. http://sourceforge.net/projects/openkore/.

付録

ソースコード 1 Rのゆらぎ解析スクリプト

```
#ファイル読み込み
player<-read.csv("/Users/iplab/Downloads/sample", header=FALSE)
#フーリエ変換, パワースペクトル変換
ftp<-fft(player[,2])
aftp<-abs(ftp)^2
aftp2<-aftp[2:513]
retu1<-player[,1]
retu2<-retu1[2:513]
xlab<-log10(retu2/(1024))
aftp3<-log10(aftp2)
x<-xlab[1:256]
y<-aftp3[1:256]
(out<-lsm(x,y))
 #傾きと切片
plot(x,y,xlim=c(-3,1),ylim=c(0,10),type="l",xlab="Log f[Hz] ",
 ylab="Log P(n)[dB]",cex.axis=1.4,cex.lab=1.4) #データセットをプロット

lines(-3:10,out[1]*(-3:10)+out[2],col="blue",lwd=2) #近似曲線を引く
```

ソースコード 2 nw.c(トラフィックサンプリングプログラム)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <net/if.h>
#include <net/ethernet.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>

#include <pcap.h>

#include "nw.h"

char version[] = "Ver:0.3, 2009/01/19";
int fourier = 0;
int base = 0;

double tick = 1;
long min;
long minsec = 0;
long minusec = 0;
long max = 0;
long packet_count[PACKET_MAX];
```

```
long packet_byte[PACKET_MAX];

int main(int argc, char *argv[])
{
        pcap_t *pd;
        char *filename = NULL;
        char *filter;
        char errbuf[PCAP_ERRBUF_SIZE];
        struct bpf_program fcode;
        pcap_handler callback;
        int datalink;
        int op;

        while((op = getopt(argc, argv, "m:r:t:bfh?")) != -1){
                switch(op){
                        case 'f': /* fourier mode */
                                fourier = 1;
                                break;

                        case 'm': /*  max */
                                max = atol(optarg);
                                break;

                        case 'r': /*    filename */
                                filename = optarg;
                                break;

                        case 't': /* tick time */
                                tick = atof(optarg);
                                break;

                        case 'b': /* tick base */
```

```
                        base = 1;
                        break;

                case 'h':
                case '?':
                default:
                        usage();
        }
}

if (filename == NULL || max == 0){
        usage();
}

if ((pd = pcap_open_offline(filename, errbuf)) == NULL){
        fprintf(stderr, "pcap_open_offline: %s\n", errbuf);
        exit(EXIT_FAILURE);
}

if (argv[optind] != NULL){
        filter = argv[optind];
        if (pcap_compile(pd, &fcode, filter, 0, 0) < 0){
                fprintf(stderr, "pcap_compile: %s\n", pcap_geterr(pd));
                exit(EXIT_FAILURE);
        }
        if (pcap_setfilter(pd, &fcode) < 0 ){
                fprintf(stderr, "pcap_setfilter: %s\n", pcap_geterr(pd));
                exit(EXIT_FAILURE);
        }
}

if ((datalink = pcap_datalink(pd)) < 0){
```

```
                fprintf(stderr, "pcap_datalink: %s\n", pcap_geterr(pd));
                exit(EXIT_FAILURE);
        }

        switch(datalink){
                case DLT_EN10MB:
                        callback = (void *)nw_ether;
                        break;

                default:
                        fprintf(stderr, "linktype = change other link\n");
                        exit(EXIT_FAILURE);
        }

        if (pcap_loop(pd, -1, callback, NULL) < 0){
                fprintf(stderr, "pcap_loop: %s\n", pcap_geterr(pd));
                exit(EXIT_FAILURE);
        }

        nw_result();

        exit(EXIT_SUCCESS);
}

void usage(){
        printf("\t nw\n");
        printf("\t\t -r [filename]\n");
        printf("\t\t -m [max count]\n");
        printf("\t\t -f (fourier mode: optional)\n");
        printf("\t\t -t (specify tick counter: optional: default is 1)\n");
        printf("\t\t -b (display (1 * tick, 2 * tick ...)
     instead of (1, 2..) as counter (optional)\n");
```

```
        printf("\n");
        printf("Version: %s\n", version);
        printf("\n");

        exit(EXIT_SUCCESS);
}

void nw_init ()
{
        int i;
        for (i = 0; i <= PACKET_MAX; i++){
                packet_count[i] = 0;
                packet_byte[i] = 0;
        }
        return;
}

void nw_ether
(u_char *userdata, const struct pcap_pkthdr *h, const u_char *p)
{
        struct ether_header *ep;
        if(h->caplen < ETHER_HDRLEN){
                return;
        }
        else {
                ep = (struct ether_header *)p;
        }

        if (ntohs(ep->ether_type) != ETHERTYPE_IP){
                return;
        }
        else {
```

```
                nw_ip(h, (u_char*)(p + ETHER_HDRLEN),
                 (u_int)(h->len - ETHER_HDRLEN));
        }
        return;
}

void nw_ip(const struct pcap_pkthdr *h, u_char *p, u_int len)
{
        struct ip *ip;
        struct timeval tv;
        int target;
        double diff;

        if (len < sizeof(struct ip)){
                return;
        }
        else {
                ip = (struct ip *)p;
        }

        tv.tv_sec = h->ts.tv_sec;
        tv.tv_usec = h->ts.tv_usec;

        if (min == 0){
                minsec = tv.tv_sec;
                minusec = tv.tv_usec;
                min += 1;
        }

        if (tick < 0.001){
diff = (tv.tv_sec - minsec) * 1000000 + (tv.tv_usec - minusec);
target = (int)(diff/(tick * 1000000));
```

```
        }
        else {
diff = (tv.tv_sec - minsec) * 1000 + (tv.tv_usec - minusec) / 1000;
target = (int)(diff/(tick * 1000));
        }

        if (target < 0){
                return;
        }

        packet_count[target] += 1;
        packet_byte[target] += h->caplen;

        return;
}

void nw_result()
{
        int i;
        for (i = 0; i <= max; i++){
                if (fourier == 1){
                        if (base == 1){
printf("%.6lf,%.6lf+%ldi,%.6lf+%ldi\r\n",
(double)(tick * i),
(double)(tick * i),
packet_count[i],
(double)(tick * i),
packet_byte[i]);
                        }
                        else {
printf("%d,%d+%ldi,%d+%ldi\r\n", i, i, packet_count[i], i, packet_byte[i]);
                        }
```

```
                }
                else {
                        if (base == 1){
printf("%.6lf,%ld,%ld\r\n",
(double)(tick * i),
packet_count[i],
packet_byte[i]);
                        }
                        else {
printf("%d,%ld,%ld\r\n", i, packet_count[i], packet_byte[i]);
                        }
                }
        }
}
```

ソースコード 3 nw.h(トラフィックサンプリングプログラム 2)

```
#ifndef __NW_H__
#define __NW_H__

#define PACKET_MAX 86400

#define NULL_HDRLEN 4 /* 4byte */
#define ETHER_HDRLEN 14 /* dst macaddr[6] + src macaddr[6] + type[2] */

void usage();
void nw_init();
void nw_result();
void nw_ether (u_char *, const struct pcap_pkthdr *, const u_char *);
void nw_ip (const struct pcap_pkthdr *, u_char *, u_int);

#endif
```