情報処理学会論文誌　Vol. 52　No. 1　2–13 (Jan. 2011)

# Winnyp通信検知機能の実装および評価

仲小路　博史[†1]　鬼頭　哲郎[†1]　重本　倫宏[†1]
寺田　真敏[†1]　石山　智祥[†2]

近年のP2P型ファイル共有ソフトウェアには通信自体の秘匿性を高めるための中継機能や，通信内容を隠すためのDiffie-Hellman鍵交換や，サービスポートを通信ごとに変更する機能が備わり，従来のIDSなどを使った手法では，調査対象とする通信がP2P通信であるかどうかを特定することが困難となってきた．本論文では国内で高いシェアを持つP2P型ファイル共有ソフトウェアWinnyの発展系であり，いまだ有効な対策が確立されていないWinnypを対象に，ディープ・パケット・インスペクション方式による通信検知機能の実装と性能の評価を行う．さらに，GPGPU（CUDA）に本機能を実装することによって，現在，コンシューマ向け接続サービスの中でも最も高速な1 Gbpsのフルワイヤスピードにも適用可能なWinnyp通信検知装置を実装し，1,020台のノードによって構成される実験環境を用いて有効性を示す．

# Implementation and Evaluation of Winnyp Detection

Hirofumi Nakakoji,[†1] Tetsuro Kito,[†1]
Tomohiro Shigemoto,[†1] Masato Terada[†1]
and Toshiaki Ishiyama[†2]

As recent P2P file-sharing applications employ various functions for hiding their communications such as connection replays, Diffie-Hellman key exchanges and dynamic port changes per connection, it becomes further difficult for existing IDS to detect P2P communications. This paper describes a method for detecting Winnyp communications using deep packet inspection. Winnyp is a successor of Winny, which is the most popular P2P application in Japan. To the best of our knowledge, our method is the first approach that detects Winnyp effectively. We implement this method on GPGPU (CUDA). Evaluation experiments using 1,020 Winnyp nodes demonstrate that this method can handle with traffic at 1 Gbps, which is equal to the fastest available Internet access speed for consumers.

## 1. はじめに

2003年以降，Winny，ShareなどのP2P型ファイル共有ソフトウェア（以下，P2Pソフト）を悪用したウイルスに起因する情報漏洩事故が多数発生し，いまだに沈静化の兆しが見えない．特に，個人情報取扱業者の業務に関わる範疇で問題が発生した場合，企業・組織，そして経営トップの監督責任が問われることとなる．WinnyやShareのような管理者不在のP2P型ファイル共有ネットワーク（以下，P2Pネットワーク）に流出した情報は，不特定多数の利用者間で増殖しながら広がっていくため，事後にそれらの情報を回収することが事実上不可能な状況にある．

また，世界的にも，誤操作による情報漏洩や著作物ダウンロードによる著作権侵害，大容量トラフィックによるISP回線の逼迫など，P2Pソフトによる通信が社会的な問題となっており，P2P通信の制御技術に対する期待が高まっている．

P2Pソフトの中には，通信自体の秘匿性を高めるために中継機能を実装していたり，通信内容や通信行為自体を秘匿するためにDiffie-Hellman鍵交換を利用した本格的な暗号処理MSE（Message Stream Encryption[1)]）を実装していたり，サービスポートを固定せずに通信ごとに変更する機能を持っていたりする．このため，従来のIDS（Intrusion Detection System）を使った手法では，対象とする通信がP2P通信であるかどうかを特定することが困難になってきている．

これに対し，対処する側はこれらのP2Pソフトを1つ1つ解析して実用的な検知技術を確立していくことで，上記社会的な問題の解決を図る必要がある．そこで，本論文では，情報漏洩などの社会的な問題にもなっていながら実用化されていなかったWinnyp検知技術の確立を目的として，GPGPUを活用した高速処理（2 Gbps）手法を提案する．さらに，プロトタイプ実装の評価を通して提案方式の有効性を示す．

提案する方式は，解析から得られたWinnyp（およびWinny）のプロトコルの特徴に着目して個々のパケットのペイロードから9バイトの固定長のデータを抽出・連結することにより，GPGPU上での検知処理実行に向けた最適化を行い，2 Gbpsの高速処理を実現する検知手法である．

†1 株式会社日立製作所
Hitachi Ltd.
†2 株式会社フォティーンフォティ技術研究所
Fourteenforty Research Institute, Inc.

本論文の構成について述べる．2 章で GPGPU を利用した関連研究について示す．3 章で Winnyp プロトコルの概要と検知における課題について，4 章で GPGPU を利用した提案方式の実現方法と有効性を述べ，5 章でまとめる．

## 2. GPGPU の概要と関連研究

本章では，P2P 通信検知技術の現状と GPGPU を利用した研究について述べる．

### 2.1 P2P 通信検知技術の現状

P2P 通信を検知する手法としては，コネクション特性に着目して検知する「フロー・ステート・コントロール」方式（以下，FSC 方式）と，ペイロードから P2P ソフト固有の通信手順（プロトコル）を読み解いて検知する「ディープ・パケット・インスペクション」方式（以下，DPI 方式）とがある．

文献 2) では，国内で利用されている代表的な P2P ソフトである Winny や Share，Gnutella などの通信の検知・制御技術として，FSC 方式および DPI 方式を組み合わせた P2P 通信検知システムについて報告している．また，いくつかのセキュリティベンダからも，同様の機能を実装したサービスや製品などが提供されている．しかしながら，DPI 方式においては一部の P2P ソフトについていまだ検知技術が確立されていない．このため，検知・制御などの技術が展開できていないのが現状である[3]．

Winnyp は，匿名性を向上させるために，Winny の既存の匿名化処理に加えて，より複雑な暗号処理を加えた P2P ソフトで，Winny 系の利用者のうち約 8%（約 1.6 万）が Winnyp 利用者であるとの報告がある[4]．

現在普及している Winnyp v2.0b7.28 の通信を検知するための研究として，パケットの送受信パケットのサイズと送出パターンから Winnyp の通信を検知する方法が提案されているが[5]，誤検知を減らすには同じ検査を何度か繰り返す必要があるため，検知結果を得るまでに複数のパケットを検査しなければならなかった．また，文献 6) では，Winnyp を含む P2P ソフトによる通信を汎用的に検知する方法が提案されているが，Winnyp や Share といった P2P ソフトの種類を弁別することについては言及されていない．

### 2.2 GPGPU を利用した研究

汎用的な算術処理を得意とする CPU（Central Processing Unit）に対して，GPU（Graphics Processing Unit）は画像処理に特化した処理を行うプロセッサとして，PC やワークステーションに搭載されていた．近年，GPU を汎用的なコンピューティングデバイスとして進化させた GPGPU（General Purpose computing on GPU）が登場し，従来のゲーム分野などにおける画像処理にとどまらず，物理シミュレーションやデータ解析などの様々な研究分野に GPGPU が利用されるようになってきた．GPGPU は，大量のデータを超並列的に高速に処理することに特化したアーキテクチャとなっている．

通信分野においては，GPGPU の超並列性とルータのパケット処理の並列処理適応性に着目して，GPGPU を利用した IP ルーティングに関わる処理を高速化する研究がある[7]．文献 8) では，GPGPU の並列処理によって LOF（Local Outlier Factor）に関わる処理を高速化し，IDS の性能向上に活用する方法が提案されている．また，IDS のパターンマッチング処理に GPU を用いて高速化する研究として文献 9)–11) がある．さらに，IDS のパターンマッチング処理の負荷を軽減するために，線形計画法を用いたプレフィルタを GPGPU に実装する研究[12] がある．文献 13) では，UNIX 系システム用ウイルス対策ツール ClamAV[14] におけるネットワークストリームとウイルスシグネチャとのパターンマッチング処理に GPGPU を利用して高速化した例が報告されている．これらの研究は，パターンマッチング処理の高速化や負荷軽減に関わる研究である．

近年，ますます大容量化するネットワークに対し，従来は ASIC（Application Specific Integrated Circuit）などの専用集積回路でデータの解析を行ってきたが，ASIC の開発には莫大な費用を要するという課題があった．また，大量のデータを分散的に処理するグリッドコンピューティングの活用も考えられるが，リアルタイム性や，大規模な設備投資が課題となる．

## 3. Winnyp プロトコルの概要と検知における課題

本章では，Winnyp プロトコルの概要と，検知における課題について述べる．

### 3.1 Winnyp プロトコルの概要

Winnyp プロトコルは，Winnyp ペイロードを暗号・復号する処理部を除いて Winnyp の原形である Winny と似た構造を持つ．まずは比較のために Winny のプロトコルについて簡単に述べる．Winny の初期パケット[⋆1]は 11 バイトの固定長の TCP ペイロードを持つことが特徴で，**図 1** に示すような 2 バイトのダミーデータ，4 バイトの RC4 暗号鍵，5 バイトの暗号データから構成されている[15]．暗号データを RC4 暗号鍵で復号すると，Winny 検知用パターンのバイト列 01 00 00 00 61 が得られ，これによって Winny の通信であると判定できる．このように，復号に関わる演算部は RC4 関数部のみであるため，Winny は比

⋆1 TCP3WayHandshake 完了後に初めて送受信するパケット．

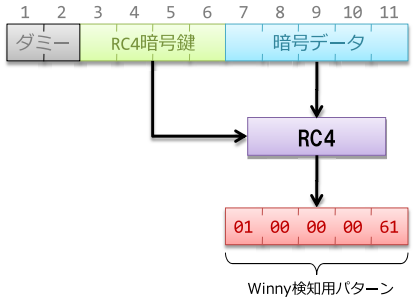


**図 1**　Winny 初期パケット復号フロー
Fig. 1　Steps of decoding a Winny first packet.

較的容易・高速に検知することが可能であった．

次に，著者らが実施した Winnyp の解析結果について概要を述べる．Winnyp は Winny に Winnyp.dll を読み込ませる形態で機能を拡張しており，Winny.exe に対して 200 カ所ほどの改変が行われている[16]．これらの改変により Winnyp の初期パケットの長さはランダムに変化する可変長となっている．また，Winny では RC4 関数を 1 度呼び出すだけで得られていた Winny 検知用パターンが，Winnyp では RC4 暗号鍵を生成するだけでも DES，MD5，SHA-1，CAST，独自処理など，様々な匿名化処理を重ねて行うようになり，Winnyp 検知用パターンの復号が複雑かつ多処理になっている（**図 2**）．

### 3.2　Winnyp 通信検知の課題

2010 年 4 月現在で，Winnyp のネイティブな通信を DPI 方式によって高精度に検知が可能な製品やサービスは調査の範囲では存在していない．ただし，Winnyp の初期設定では Winny プロトコル互換設定が有効となっているため，Winny 用の DPI 方式の検知機能によって Winnyp の通信を検知できる場合がある．しかし，Winnyp の設定を Winnyp 専用モードに変更することによって，従来の Winny 用の検知機能ではいっさい検知ができなくなる．このような Winnyp の通信を検知するにあたっては，前述した匿名化処理の複雑化，およびペイロード長の多様化の 2 つの問題を解決する必要がある．

また，Winnyp 通信検知機能を実用化するためには，ISP のコンシューマ向け接続サービスの中でも最も高速な 1 Gbps 回線に適用可能な処理性能が求められる．このため，Full Duplex に換算して 2 Gbps の実効スループット（処理容量）を達成することで本論文の提案手法の有効性を示す．

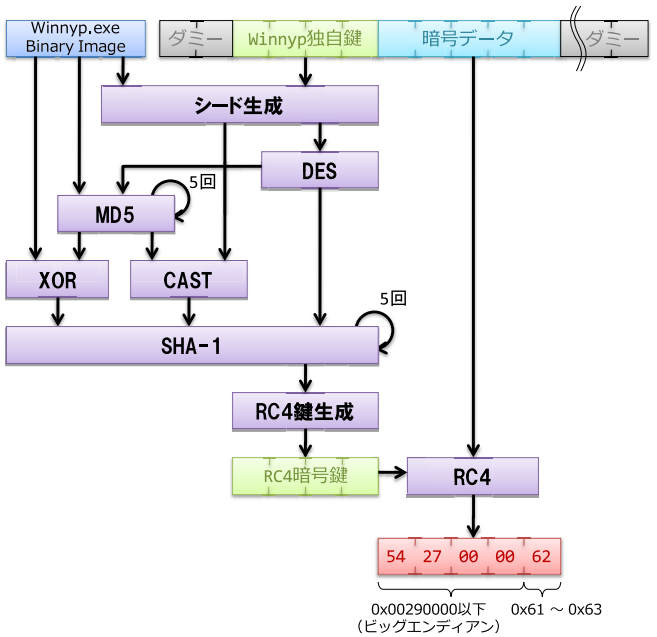


**図 2**　Winnyp 初期パケット復号フロー
Fig. 2　Steps of decoding of a Winnyp first packet.

**表 1**　RC4 暗号鍵生成に関わるステップ数
Table 1　The number of steps for generating RC4 keys.

| P2P 種別 | ステップ数 |
|---|---|
| Winny | 115 |
| Winnyp | 21,437 |

#### 3.2.1　匿名化処理の複雑化

3.1 節で述べたように，Winnyp の初期パケットの生成処理には Winny と比較して匿名化に関わる暗号関連処理が多く含まれる．つまり，Winnyp の 1 回の初期パケット復号フローにかかる処理量は Winny の初期パケット復号フローの処理量よりも多くなる．

**表 1** に，Winny と Winnyp のそれぞれの実行コードに含まれる RC4 暗号鍵生成に関わる処理部をニーモニックに変換して得たステップ数を示す．Winny の初期パケット復号

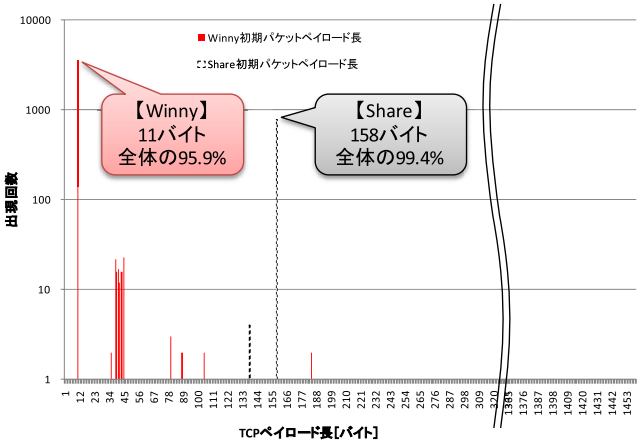


**図 3**　Winny 初期パケットペイロード長分布
Fig. 3　Payload size distribution of Winny first packets.

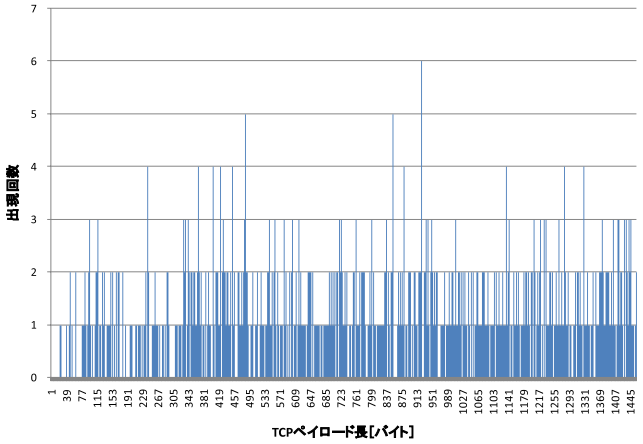


**図 4**　Winnyp 初期パケットペイロード長分布
Fig. 4　Payload size distribution of Winnyp first packets.

フローに含まれる RC4 暗号鍵生成処理に関わる処理量は 115 ステップであったのに対し，Winnyp の処理量は 21,437 ステップと，約 186 倍であった．これらの値には，実行時における分岐や繰返し，構成する命令のクロックサイクル数などが加味されていない．このため，実行時の処理量との厳密な比較はできないが，処理量とステップ数には相関性が現れることから目安になるといえる．

### 3.2.2　ペイロード長の多様化

Winny の通信を検知するには，ペイロード長が 11 バイトのパケットに絞り込んで検査することで検知の効率化を図れた．しかし，Winnyp はペイロードにランダムサイズのダミーデータが付与される仕様であるため，ペイロード長による絞り込みができなくなり，すべてのパケットを対象とした検査処理が必要となる．

ここでインターネットに接続した Winny と Winnyp，そして参考として Share の各ノードの通信を，それぞれ 10 分程度観測して得た初期パケットのペイロード長の分布を**図 3**，および**図 4** に示す．なお Winnyp の初期パケットの 90%以上が 1,460 バイトに分布しているため，1,460 バイトのパケットを母集団から除いてグラフに示す．この 1,460 バイトは，測定環境上における TCP パケットの上限値（MSS：Max Segment Size）と一致する．これは，ダミーデータの付加によって 1,460 バイトを超えてしまった通信データが複数のパケットに断片化され，断片化した最初のパケット（初期パケット）のペイロード長が 1,460 バイトとして計数されることに起因する．一方で，Winny および Share の通信には断片化するほどの大きなペイロードを持ったパケットが含まれておらず，1,460 バイトのペイロード長を持つ初期パケットが含まれていなかったため，前記母集団から除外する処理は実施していないが，ペイロード長の分布に著しい偏りが現れたために両者のグラフは Y 軸対数グラフで表した．

**表 2**　ペイロード長の傾向
Table 2　Trend in payload length.

| P2P 種別 | 平均 [byte] | 標準偏差 |
|---|---|---|
| Winny | 12.77 | 10.74 |
| Share | 157.70 | 5.70 |
| Winnyp | 810.36 | 390.97 |

このように，初期パケットのペイロード長は，Winny の場合は 95%以上が 11 バイトに，Share の場合は 99%以上が 158 バイトに，それぞれ偏って分布している．

一方，Winnyp の初期パケットは，20 バイトから 1,460 バイトの広範にわたりほぼ均等に分布している．Winny，Share，そして Winnyp のペイロード長の分布傾向を示した**表 2** にみられるように，Winnyp は，他の P2P ソフトと比較して，突出してペイロードが長く，

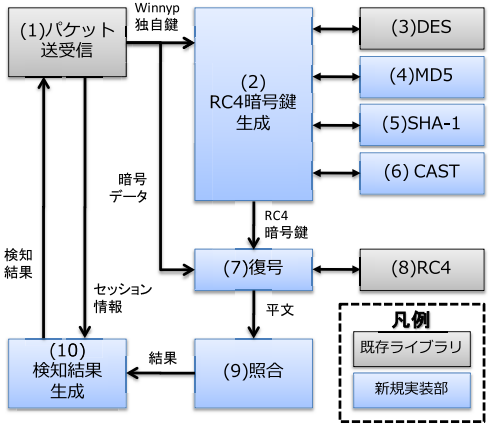


**図 5**　Winnyp 通信検知機能ブロック
Fig. 5　The function block for detecting Winnyp.

標準偏差が大きい特徴（ペイロード長が一定しない特徴）がある．

これは，Winny の初期パケットの検知にあたっては，ペイロード長が 11 バイトのパケットに絞り込んだうえで図 1 に示した復号フローを処理することで効率化を図れるが，Winnyp の初期パケットの検知にあたっては，ペイロード長が 11 バイト以上のすべてのパケットに対して 3.2.1 項に示した複雑な処理を繰り返さなければならないことを意味する．単純に考えても，初期パケットを検査する頻度が Winny の場合と比較して 1,400 倍以上となる．

### 3.3　DPI 方式による Winnyp 通信検知機能

3.1 節で解析した Winnyp のプロトコル仕様に基づいて，Winnyp の通信に含まれる初期パケットをリアルタイムに検知する機能について述べる．

Winnyp 通信検知機能は **図 5** に示す 10 の機能ブロックによって構成されており，通信の受信から SNMP Trap（Simple Network Management Protocol Trap）を用いた検知結果の送信までの機能を含んでいる．次に，各機能ブロックにおける処理内容ついて述べる．

（1）　パケット送受信
ネットワークインタフェースを監視してパケットを受信し，Winnyp 独自鍵（TCP ペイロードの 3〜6 バイト）を RC4 暗号鍵生成機能に，暗号データ（TCP ペイロードの 7〜11 バイト）を復号機能に，セッション情報（5-tuple）を検知結果生成機能に，それぞれ渡す．また，検知結果生成機能が生成した SNMP Trap データを送信する機能も有する．

（2）　RC4 暗号鍵生成
パケット送受信機能より受け取った Winnyp 独自鍵をもとに，DES 機能，MD5 機能，SHA-1 機能，CAST 機能と連携し，RC4 暗号鍵を生成して復号機能に渡す．

（3）　DES
標準の DES の実装と同様である．

（4）　MD5
標準の MD5 と比較して，メッセージ処理関数の呼び出し順序および内部定数が異なる独自の MD5 ハッシュ変換処理を行う．

（5）　SHA-1
標準の SHA-1 と比較して，新たな撹拌処理および内部定数，ハッシュ生成処理が行われるなど，独自の SHA-1 ハッシュ変換処理を行う．

（6）　CAST
標準の CAST-128 と比較して，S-box の一部が入れ替わっている独自の CAST 暗号処理を行う．

（7）　復号
RC4 暗号鍵生成機能より受け取った RC4 暗号鍵をもとに，RC4 機能と連携して復号処理を行う．本機能は，Winny による暗号データの復号機能と同一である．

（8）　RC4
標準の RC4 の実装と同様である．

（9）　照合
復号機能より受け取った平文の先頭 4 バイト（ビックエンディアン表記）が 0x00002900（リトルエンディアン表記）以下で，さらに続く 1 バイトが 0x61，0x62，0x63 のいずれかであった場合に結果（陽性）を検知結果生成機能に渡す．

（10）検知結果生成
照合機能より受け取った検知結果およびパケット送受信機能より受け取ったセッション情報に基づき，Winnyp 通信被疑ノードの IP アドレス，ポート番号などを MIB（Management Information Base）に格納して，SNMP Trap データを生成する．さらに，生成した SNMP Trap データをパケット送受信機能に渡す．

### 3.4　CPU 向けの Winnyp 通信検知機能の実装と有効性評価

3.2 節であげた課題を解決するため，まず，CPU 向けに実装した Winnyp 通信検知機能の評価を行う．性能評価にあたり，検知速度性能および検知精度性能の 2 つの観点で実験を

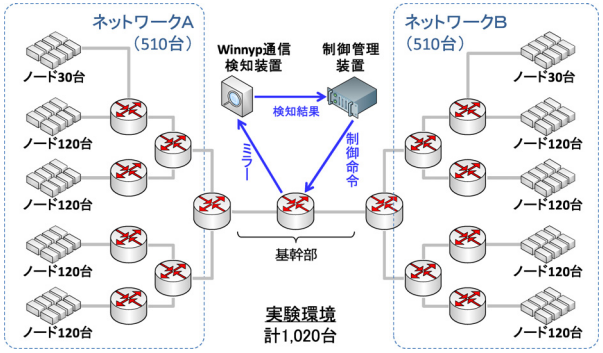


**図 6**　Winnyp 通信検知システム実験環境
Fig. 6　An experiment environment for evaluating the Winnyp detection system.

行う．本節では，最初に実験環境について，次に実験環境を用いて取得した評価用データについて説明した後，実験結果について述べる．

### 3.4.1　事 前 準 備

（1）環境構築

評価にあたっては，情報通信研究機構北陸リサーチセンターが開発した大規模テストベッド設備である通称 StarBED [17] を活用した．StarBED では，すべてのノードを管理下に置いて実験の実施および観測を行える．このため，インターネットを利用した実験とは異なり，より厳密な条件での評価が可能である点や，Winnyp ノードの存在の把握が可能な点などの理由により，事前に正確な真値の用意が可能となる．

本実験では，**図 6** に示すようなインターネットを模擬した階層的ネットワークと，510 台の Winnyp ノード[⋆1]が存在するネットワーク A，B を相互に接続させた計 1,020 台のノードによって構成される実験環境を構築した．Winnyp 通信検知装置を中央のルータ（基幹部）のミラーポートに接続して，ネットワーク A，B 間で発生する Winnyp の通信を検知する．

検知結果は SNMP Trap で制御管理装置へ渡され，制御管理装置はルータに対して当該 IP アドレスの制御命令（遮断）を即時に行う．

各 Winnyp ノードは，17 個[⋆2]のファイルをアップロードする設定とし，各ファイルの名称は，インターネットをクローリングして得られた実ファイルの名称を利用した．また，ファイルの内容は 0x00 で埋め尽くされたダミーデータとした．ファイルサイズは，ノードの HDD の容量による制約から，実際のサイズの 1/100 にスケールダウンして生成した．また，Winnyp のクラスタワードは，クローリングして得られたノード情報から無作為に抽出して設定した．初期ノードは，実験環境のいくつかのノードの IP アドレスとポート番号を無作為に設定した．これにより Winnyp ノードはネットワーク A，B 間でファイル検索やファイルのダウンロード・アップロードを行うようになる．

**表 3** Winnyp 通信検知装置のスペック
Table 3　The specification of Winnyp detection system.

| 項目 | 仕様 |
|---|---|
| CPU | Intel Core i7 920 (2.66Ghz) |
| Memory | DDR3-1333 Triple Channel (12GB) |
| NIC | Intel 82567V-2 (on Board 1GbE) |
| OS | Windows Vista Ultimate 64bit |

**表 3** に今回評価に用いた Winnyp 通信検知装置のスペックを示す．

（2）評価用データ作成

Winnyp 通信検知装置の評価を繰り返して行えるようにするために，実験環境の基幹部を通過するパケットを，Winnyp 通信検知装置でキャプチャして pcap ファイルとして保存する．pcap ファイルを Winnyp 通信検知機能部に直接読み込ませることによって Winnyp 通信検知機能部の検知精度，および検知速度に関する性能計測を同一条件下で実施できるとともに，Winnyp 通信検知装置のハードウェア上の制約である 1 Gbps を超えるトラフィックに対する性能評価を実施できるようになる．検知速度性能評価に用いる pcap ファイルは，図 6 に示した実験環境で 1,020 台の Winnyp ノードを動作させて得た．pcap ファイルの概要を**表 4** に示す．

接続数とはネットワーク A，B 間で TCP 接続が確立した数を指す．ユニーク IP 数は，キャプチャ時間内に TCP 接続が確立した IP アドレス（送信元，宛先）の数（ユニーク）で，Winnyp 通信検知機能が本来検知すべき Winnyp ノードの数（真値）である．これは，Winnyp が活性化していなかったり，同じネットワーク内での通信にとどまっていたりする

⋆1 グループ H170 台に VMWare ESXi をインストールし，1,020 台の Windows 環境を構築（6 仮想ノード/物理ノード）．

⋆2 クローリングデータによって得られた流通ファイルの総数（ハッシュがユニーク）を，全ノード数（IP とポート番号がユニーク）で割ることにより算定．

**表 4**　検知速度性能評価用 pcap の概要
Table 4　Summary of a pcap for throughput evaluation.

| 項目 | 値 |
|---|---|
| ファイルサイズ [bytes] | 1,109,664,593 |
| パケット数 [pkts] | 1,000,000 |
| 接続数 | 32,557 |
| ユニーク IP 数 [nodes] | 717 |
| 平均パケット数 [pkts/node] | 2,237.13 |
| 平均ペイロード長 [bytes/pkt] | 1,078.14 |

などの理由により，監視対象となる中央のルータで検知することが原理的に不可能なノードを，評価対象から除外することを目的としている．つまり，全 1,020 台中 717 台の Winnyp ノードが今回の検知すべき対象となる．平均パケット数は，1 台あたりの送受信パケット数の平均を求めた値である．

検知精度性能評価にあたっては，Winnyp 通信検知機能の誤検知率（False Negative/False Positive）を評価する必要がある．このため，検知速度性能評価用 pcap ファイ（表 4）に加えて，Winnyp 以外の様々な通信（非 P2P ソフト系 8 種，P2P ソフト系 8 種）を含む pcap ファイルを用意した．誤検知率の評価に用いる pcap ファイルの概要を**表 5** に示す．この中の各非 P2P ソフト系通信の帯域占有率は，ISP の協力を得てカスタマエッジ部で調査した結果をもとにトラフィックジェネレータによって再現した．また，P2P ソフト系通信の帯域占有率は，P2P ソフトの国内シェア[18] および図 6 に示した実験環境で各 P2P ソフトを実際に動作させて得られた平均スループットをもとに算出した結果を用いて再現した．

### 3.4.2　検知性能評価

（1）検知精度性能評価

3.4.1 項で作成した 2 つの pcap ファイルを Winnyp 通信検知機能に読み込ませて得られた結果を**表 6** に示す．なお，評価結果は同一条件下（装置構成・pcap ファイル）で 5 回実施して得られた値を平均して求めた．

結果，評価対象としていた 717 台の Winnyp ノードのすべてを漏れなく検知（False Negative が 0%）し，Winnyp 以外の通信に関しても，誤検知はまったくない（False Positive が 0%）という結果を得た．

本実験では pcap ファイルを逐次読み込みしているため，パケットの取りこぼしは発生しない．このため，パケット送受信部に起因する検知精度の低下は考慮していない．

**表 5**　検知精度性能評価用 pcap の概要
Table 5　Summary of a pcap for evaluation of precision.

| 項目 | 値 |
|---|---|
| ファイルサイズ [bytes] | 8,925,618,647 |
| パケット数 [pkts] | 7,712,439 |
| 接続数 | 30,258 |
| ユニーク IP 数 [nodes] | 1,841 |
| 平均パケット数 [pkts/node] | 1,157.30 |
| 平均ペイロード長 [bytes/pkt] | 1,124.39 |
| 非 P2P ソフト系通信（帯域占有率） | http(21.81%), https(1.17%), smtp(0.05%), ftp(0.20%),rtsp(2.18%), pop3(0.17%), dns(0.06%), ssh(0.06%) |
| P2P ソフト系通信（帯域占有率） | Winny(22.39%), Share(3.72%), Cabos(4.40%), LimeWire(14.00%), WinMX(15.39%), BitTorrent(11.37%), PerfectDark(0.58%), Winnyp(2.54%) |

**表 6**　検知精度性能評価結果（CPU）
Table 6　Evaluation results of detection accuracy (CPU).

| 項目 | 結果 |
|---|---|
| 総検知数 | 3,078 |
| ユニーク IP 数 [nodes] | 717 |
| 検知率 | 100% |
| 誤検知率 | 0% |

（2）検知速度性能評価

検知速度性能を測定するにあたっては，Winnyp 通信検知機能の主要部である機能ブロック（**図 7** の点線部）を対象に CPU 使用時間を計測して評価を行った（**表 7**）．本来，前記実験環境を利用したリアルタイム評価を実施することが望ましいが，ネットワークインタフェースのスペックによる制約や実ノードによるトラフィック発生量の限界により 1 Gbps を超えるトラフィックの正確な測定が困難であるため，Winnyp 通信検知機能の主要部に対象を絞って限界性能の評価を実施する．

検知速度性能の一指標である実効スループット（処理容量）$T_c$ は，実効スループット（処理能力）$T_a$ と平均ペイロード長 $\mathrm{Avg}(P_{\mathrm{size}})$ と平均ヘッダ長 $\mathrm{Avg}(H_{\mathrm{size}})$ をもとに，式 (1) によって求めることができる．ここでは実効スループット（処理能力）$T_a$ が 70,715.85 pps

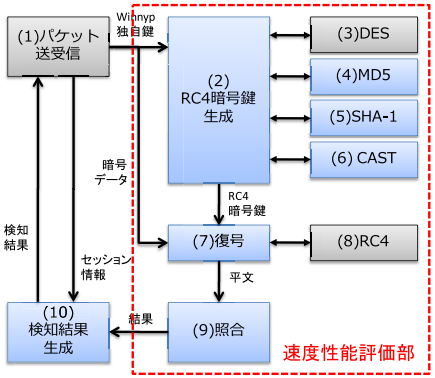


**図 7**　速度性能評価の対象
Fig. 7　Targets for estimating the detection speeds.

**表 7**　評価結果（CPU）
Table 7　Evaluation results (CPU).

| 項目 | 結果 |
|---|---|
| 総検知数 | 3,078 |
| ユニーク IP 数 [nodes] | 717 |
| 検知率 | 100% |
| 誤検知率（False Negative） | 0% |
| 誤検知率（False Positive） | 0% |
| CPU 使用時間 [ms] | 14,141.10 |
| 実効スループット（処理能力）[pps] | 70,715.85 |
| 実効スループット（処理容量）[Mbps] | 632.56 |

であることと，一般に TCP と IP ヘッダの合計値が 40 バイトであることから，実効スループット（処理容量）$T_c$ は 632.56 Mbps となる．

$$T_c = T_a \times (\mathrm{Avg}(P_{\mathrm{size}}) + \mathrm{Avg}(H_{\mathrm{size}})) \times 8 = 70715.85 \times (1078.14 + 40) \times 8 = 632562911 \quad (1)$$

3.2 節に示したように，広く普及している 1 Gbps フルワイヤスピードのトラフィックを監視する場合，Full Duplex 換算で最大 2 Gbps のスループットを達成することが要件となる．つまり，実用的に利用するには，前述した Winnyp 通信検知機能の実装では検知速度性能が大幅に足りない．

評価に用いた装置に搭載された CPU はネイティブクアッドコアであり（表 3），Winnyp 通信検知機能をシングルスレッドプログラムで実装した場合，前述したように 632.56 Mbps の処理性能となる．これを，4 並列（マルチプロセス）で動作させた場合には，1 コアあたり 506.28 Mbps の実効スループット（処理容量）となる．この結果から，装置全体としては 2,025.14 Mbps の実効スループット（処理容量）を達成できる．しかしながら，文献 19) に公開されている最近のインターネットのトラフィック統計情報をもとに平均パケット長を求めると 441.57 バイトを得る．これは表 4 にまとめた Winnyp のパケットの平均パケット長（平均ペイロード長 ＋ 40 バイト）よりも大幅に小さい．インターネットトラフィックでの実効スループット（処理容量）を算定するために平均パケット長として 441.57 バイト/パケットをパラメータとして用いて再計算すると，1 コアあたり 199.94 Mbps となり，装置全体としても 799.75 Mbps 程度にとどまることになる．つまり，インターネットの利用状況によっては，1 つの CPU で 1 Gbps フルワイヤスピードに対応する検知速度性能を達成することは難しいといえる．

## 4. Winnyp 通信検知機能の GPGPU 実装

本章では 1 Gbps フルワイヤスピードのトラフィックに含まれる Winnyp 通信を漏れなく検知可能な検知速度性能を達成するため，GPGPU 向けの Winnyp 通信検知機能について述べる．

### 4.1 GPGPU 実装の特徴

GPGPU 向けの実装にあたっては，NVIDIA 社が提供する CUDA（Compute Unified Device Architecture）を利用し，パターンマッチング処理に加え，Winnyp の RC4 暗号鍵生成，復号処理を処理する．

CUDA には，被検査データ（ネットワークインタフェースで受信したパケット群）を GPGPU に接続されたメモリ（ビデオメモリ）に転送する処理と，被検査データを並列に処理する GPGPU 用プログラム（カーネル）とをそれぞれ実装する．GPGPU 上でプログラムを実行する複数のスレッドは，上記ビデオメモリに格納された被検査データ（パケット

群データ）から個々のスレッドの処理すべきパケットを重複なく読み出して処理できなければならない．このため，個々のスレッドが効率的に処理可能なように被検査データを再構成する必要がある．

#### 4.1.1　データ整形

Winnyp 通信検知機能を CUDA で実装するにあたって必要となる処理がデータの整形である．前述したように，CUDA で実行される複数のスレッドは，同一のデータ（ここでは多数のパケットを結合したパケット群）を参照する．並列に動作する個々のスレッドには連続的で一意な識別子が割り当てられている．本提案手法では，各スレッドが自身に割り当てられた識別子をもとに処理すべきデータの位置（ここでは特定の 1 パケットの位置を示すアドレス）を導出し，その位置に基づき，パケット単位で並列的に解析するインターパケットアプローチをとる．この場合，スレッドに割り当てられた一意な識別子から容易にアドレスを算出可能とすることが望ましい．そこで，Winnyp のプロトコルの特徴に着目して，個々のパケットのデータ長を固定長に整形する．3.1 節で述べたように，Winnyp 通信の検知には，3 バイト目以降の Winnyp 独自鍵（4 バイト長）と，7 バイト目以降の暗号データ（5 バイト長）があれば検知可能であることから，先頭のダミーデータを除いたデータ（3 バイト目以降の 9 バイト分）を解析対象として抽出することで，固定長データへと整形する（**図 8**）．さらに，抽出した固定長データを結合して，スレッド処理用にパケット群データを生成する．

#### 4.1.2　カーネル実装

CUDA は，**図 9** に示すように，処理単位が階層構造を持っている．最も大きな単位がグリッド（grid）である．グリッドは複数のブロック（block）から構成され，個々のブロックはさらに複数のスレッド（thread）から構成される．カーネルとは，ホスト（CPU）から呼び出され，GPU デバイス上の各プロセッサコア（に割り当てられたスレッド）で実行される共通のプログラムのことで，ここでは 3.3 節で述べたような，パケットを解析して Winnyp であるか否かを判定するプログラムを指す．

CPU の処理によって取得した通信データを，前項で述べた方法に従って 9 バイト単位の固定長データに整形する．さらに，受信した 131,072 パケット分の固定長データを結合したパケット群データを作成し，それを CUDA 上の 1 つのグリッドで処理する．グリッドは 512 個のブロックから構成されており，ブロックは 256 個のスレッドから構成されるように実装する．これによって，各スレッドはスレッド識別子（番号）に基づきパケット群データの中から解析すべき単一のパケットを取得し，判定を行う．同時に処理可能なスレッド数は

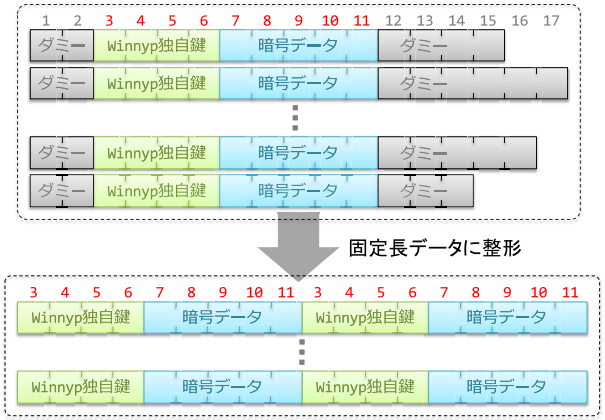


**図 8**　固定長データ（CUDA 演算用）
Fig. 8　Fixed-length data (for CUDA computation).

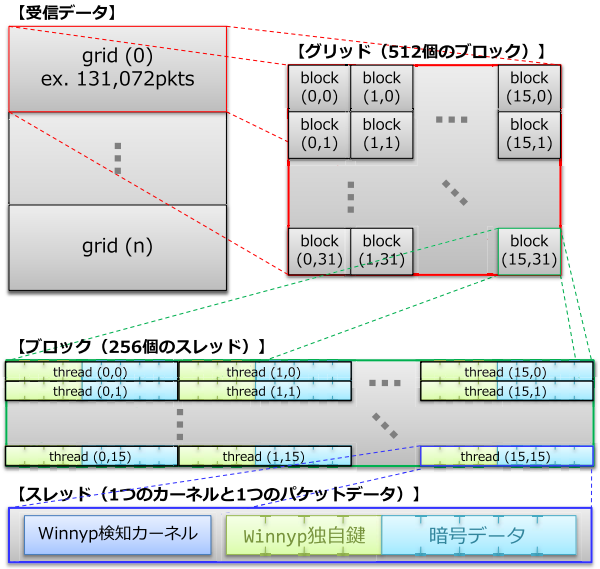


**図 9**　スレッドへのデータ割当ての仕組み
Fig. 9　A mechanism for allocating data to threads.

**表 8**　評価結果（GPGPU）
Table 8　Evaluation results (GPGPU).

| 項目 | 結果 |
|---|---|
| 総検知数 | 3,078 |
| ユニーク IP 数 [nodes] | 717 |
| 検知率 | 100% |
| CPU 使用時間 [ms] | 998.97 |
| 実効スループット（処理能力）[pps] | 1,001,032.91 |
| 実効スループット（処理容量）[Mbps] | 8954.38 |
| CPU 性能比 | 1,416% |

GPU の種類によって異なるが，著者らが利用した NVIDIA 製 GeForce 285 GTX は 240 スレッドの並列処理が可能である．

### 4.2　GPGPU 版 Winnyp 通信検知機能の性能評価

3.4.2 項で利用した Winnyp 検知装置ハードウェアに前記 NVIDIA 製 GeForce 285 GTX を搭載し，同じ手順で検知機能の単体性能評価を行った（**表 8**）．

検知速度性能について，CPU（1 コア）の処理では 633 Mbps だった実効スループット（処理容量）が，GPGPU の処理では 8,954 Mbps へと飛躍的に向上した．これは 14 倍以上の高速化となり，インターネットトラフィックでの実効スループット（処理容量）に換算して 3,536.21 Mbps（3.5 Gbps）となる．これは，3.2 節で目標に掲げた 1 Gbps フルワイヤスピード対応の要件（2 Gbps）を十分に満たす性能である．なお，検知精度性能に関しては CPU の結果とまったく同じで性能の低下はなかった．

### 4.3　考　　察

考察では，限定的な環境ではあるが，実用性について検証した結果について述べる．pcap を用いた検知速度および検知精度評価から，1 Gbps フルワイヤスピードに対応できる見通しを示した．しかし，3.4.2 項で述べたように，対象を Winnyp 通信検知機能の主要部である暗号解読およびパターンマッチング（照合）に関わる箇所に限定した評価であった．実際の運用にあたっては，パケットの受信やデータの整形，SNMP Trap の生成，送信などの処理が必要になる．そこで，Winnyp 通信検知機能および実装した装置と，実際の Winnyp 通信の発生する StarBED に構築した実験環境（図 6）を用いて，検知から遮断までの一連の処理とトラフィック量の推移について検証した．Winnyp 通信検知装置が検知した Winnyp ノード数と，検知結果に基づいて制御管理装置が制御（遮断）したことによるトラフィック

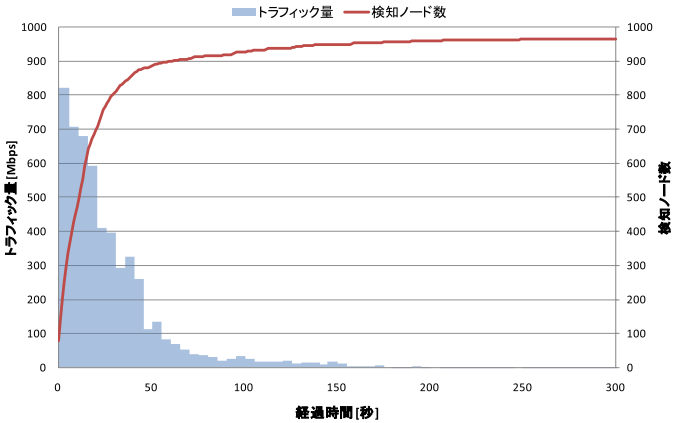


**図 10**　Winnyp 検知数とトラフィック量の推移
Fig. 10　Changes in the number of Winnyp detection and the amount of traffic.

量の推移をみると，Winnyp 通信検知装置は実験開始直後から大量の Winnyp ノードを検知し始め，約 60 秒後には，900 近いノードを検知した（**図 10**）．また，開始直後に 800 Mbps を超える値を示していた基幹部のトラフィック量は，Winnyp 通信検知装置が検知し始めて 20 秒程度で半減し，約 180 秒後には 1 Mbps 以下となった．このように，制御管理装置と連携した検証から Winnyp 通信検知装置の実装が実用上でも十分に有効であると考える．

## 5. ま　と　め

本論文では，Winny の発展系である Winnyp の通信プロトコルを明らかにし，DPI 方式を採用した Winnyp 通信検知機能を Winnyp 通信検知装置として実装した．大規模テストベッド「StarBED」に 1,020 台のノードで構成された実験環境を構築して本装置の評価を行い，検知率 100%，誤検知率 0%の高精度な評価結果を得た．また，Intel Core i7 920 プロセッサを搭載した PC で，速度性能を評価し，799 Mbps の実効スループット（処理容量）を達成した．

さらに，1 Gbps フルワイヤスピード（Full Duplex）への対応を目標に，Winnyp 通信検知機能を GPGPU（CUDA）上に実装し，検知精度を落とさずに実効スループットを 14 倍以上に高速化できることを評価実験を通して示した．これにより，約 3.5 Gbps の実効スループット（処理容量）を達成し，1 Gbps フルワイヤスピード（Full Duplex）に対応可能

であること，Winnyp 通信検知において GPGPU による実装が有効であることを示した．

GPGPU に実装することによって大幅な性能向上を実現できたが CUDA が苦手とする分岐（if 文など）の多い処理を要する解析では，逆に性能が低下する場合もある．今回は CPU 向けのソースコードの変更を最小限にして GPGPU 向け実装に適用した．そのため，分岐処理の削減，レジスタ・シェアードメモリの活用，バンクコンフリクトの回避，アライメントの最適化など，多くの点で改善の余地が残されている．今後は，近い将来訪れるであろう 10 GbE 時代を見据えて CUDA に最適化した実装を行い，Winnyp 通信検知装置の 10 Gbps 対応を目指したいと考えている．

**謝辞**　大規模ネットワーク実験環境 StarBED を本実験環境として利用するにあたりご協力をいただいた独立行政法人情報通信研究機構北陸リサーチセンター，ICT 研究開発機能連携推進会議（HIRP）の関係者各位に深く感謝いたします．また，StarBED 上の実験環境構築にあたり，有益な助言と協力をいただいた北陸先端科学技術大学院大学ならびに，独立行政法人情報通信研究機構北陸リサーチセンターの篠田陽一教授，三輪信介氏，宮地利幸氏，中井浩氏，安田真悟氏に深く感謝いたします．本研究は総務省から委託を受けた「ネットワークを通じた情報流出の検知および漏出情報の自動流通停止のための研究開発」の支援を受け実施しています．本研究を進めるにあたって有益な助言と協力をいただいた関係者各位に深く感謝いたします．

## 参 考 文 献

1) VuzeWiki: Message Stream Encryption (aka PHE) format specification, VuzeWiki (Online), available from 〈http://www.azureuswiki.com/index.php/Message_Stream_Encryption〉(accessed 2010-07-01).
2) 重本倫宏，大河内一弥，寺田真敏：コネクション解析による P2P 通信端末検知手法，情報処理学会コンピュータセキュリティ研究報告，Vol.2008, No.71, pp.329–334 (2008).
3) 園田道夫：Winny はなぜ破られたのか，株式会社九天社 (2007).
4) 宮川雄一：Winnyp 状況調査報告，安心・安全インターネット推進協議会 P2P 研究会（オンライン），入手先 〈http://www.scat.or.jp/stnf/contents/p2p/p2p080910_4.pdf〉(参照 2010-07-01).
5) 水谷正慶，白畑　真，南　政樹，村井　純：複数セッションの相関関係を利用したセキュリティイベント検知手法の提案，情報処理学会コンピュータセキュリティシンポジウム，Vol.2007, No.10, pp.595–600 (2007).
6) 元木伸宏，泉　裕，塚田晃司：トラフィックパターン解析に基づく P2P ファイル共有ソフトウェアの利用検出，情報処理学会第 71 回全国大会，Vol.71, No.3, pp.3.241–3.242 (2009).
7) Shuai, M., Xinya, Z., Nairen, Z., Jiaxin, L., Deng, Y.S. and Shu, Z.: IP Routing Processing with Graphic Processors, *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), pp.93–98 (2010).
8) Malak, A., Byunghyun, J. and David, K.: Accelerating the local outlier factor algorithm on a GPU for intrusion detection systems, *ACM International Conference*, Vol.425, pp.104–110 (2010).
9) Giorgos, V., Spiros, A., Michalis, P., Evangelos, P.M. and Sotiris, I.: High Performance Network Intrusion Detection Using Graphics Processors, *11th International Symposium On Recent Advances In Intrusion Detection* (*RAID*), Vol.5230/2008, pp.116–134 (2008).
10) Antonino, T., Oreste, V. and Donatella, S.: Efficient pattern matching on GPUs for intrusion detection systems, *Proc. 7th ACM International Conference on Computing Frontiers*, pp.87–88 (2010).
11) Smith, R., Goyal, N., Ormont, J., Sankaralingam, K. and Estan, C.: Evaluating GPUs for Network Packet Signature Matching, *The International Symposium on Performance Analysis of Systems and Software 2009*, pp.175–184 (2009).
12) Chengkun, W., Jianping, Y., Zhiping, C., En, Z. and Jieren, C.: An Efficient Prefiltering Mechanism for Parallel Intrusion Detection Based on Many-Core GPU, *Communications in Computer and Information Science*, Vol.58, pp.298–305 (2010).
13) Hubert Nghuyen: *GPU Gems3*, Addison-Wesley Professional (2007). 中本　浩（訳）：GPU Gems3, pp.689–699, 株式会社ボーンデジタル (2008).
14) ClamAV: ClamAntiVirus, ClamAV (Online), available from 〈http://www.clamav.net/〉(accessed 2010-07-01).
15) 鵜飼裕司：検出ツールの開発者が語る，「Winny を検出する方法」，ITpro（オンライン），入手先 〈http://itpro.nikkeibp.co.jp/article/Watcher/20060411/235051/〉(参照 2010-07-01).
16) 石山智祥：Inside “Winnyp” — Winnyp の内部動作とネットワーククローリングシステムの全貌，株式会社フォティーンフォティ技術研究所（オンライン），入手先 〈http://www.fourteenforty.jp/research/research_papers/PacSec2008_ishiyama_jp.pdf〉(参照 2010-07-01).
17) Hokuriku Research Center: StarBED Project, Hokuriku Research Center (Online), available from 〈http://www.starbed.org/〉(accessed 2010-07-01).
18) 社団法人コンピュータソフトウェア著作権協会：ファイル共有ソフト利用実態調査，社団法人コンピュータソフトウェア著作権協会（オンライン），入手先 〈http://www2.accsjp.or.jp/activities/2009/news98.php〉(参照 2010-07-01).
19) MAWI (Measurement and Analysis on the WIDE Internet) Working Group: Packet traces from WIDE backbone 2009/12/1, MAWI (Online), available from 〈http://mawi.wide.ad.jp/mawi/samplepoint-F/2009/200912011400.html〉(accessed

2010-07-01).

(平成 22 年 4 月 19 日受付)
(平成 22 年 10 月 4 日採録)

**仲小路博史**（正会員）

2001 年東京理科大学大学院理工学研究科情報科学科修士課程修了．同年（株）日立製作所システム開発研究所入所．PKI ならびに X.509 属性証明書の研究開発に従事．現在はネットワークセキュリティ技術に関する研究開発に従事．

**鬼頭　哲郎**（正会員）

2005 年東京大学大学院情報理工学系研究科電子情報学専攻修士課程修了．同年（株）日立製作所システム開発研究所に入所．以来，ネットワークセキュリティ技術に関する研究開発に従事．

**重本　倫宏**

2006 年大阪大学大学院基礎工学研究科システム創成専攻修士課程修了．同年（株）日立製作所システム開発研究所入所．現在はネットワークセキュリティ技術に関する研究開発に従事．

**寺田　真敏**（正会員）

1986 年千葉大学大学院工学研究科写真工学専攻修士課程修了．同年（株）日立製作所入社．博士（工学）．現在，システム開発研究所にてネットワークセキュリティの研究に従事．2004 年から Hitachi Incident Response Team チーフコーディネーションデザイナー．2004 年 4 月から JPCERT コーディネーションセンター専門委員．2004 年 4 月から 2007 年まで中央大学研究開発機構客員研究員．2004 年 8 月から情報処理推進機構セキュリティセンター研究員．2008 年から中央大学大学院客員講師を兼務．

**石山　智祥**

2002 年東京電機大学工学部情報通信工学科卒業．2007 年 12 月（株）フォティーンフォティ技術研究所入社．脆弱性分析や P2P システムセキュリティに関する調査・研究に従事．現在は Web 感染型マルウェア検出システムの研究開発に従事．