Vol.2010-CSEC-50 No.39
2010/7/2

GPGPUを用いたWinnyp通信の検知および評価

仲小路博史[†]　鬼頭哲郎[†]　重本倫宏[†]
寺田真敏[†]　石山智祥[††]

本稿では国内で最も高いシェアを持つP2P型ファイル共有ソフトウェアWinnyの発展系であり，未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．具体的には，GPGPU（CUDA）を活用することによって，現在，コンシューマ向けインターネット接続サービスの中でも最も高速な1Gbps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，1020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 using GPGPU

HIROFUMI NAKAKOJI[†]　TETSURO KITO[†]
TOMOHIRO SHIGEMOTO[†]
MASATO TERADA[†]　TOSHIAKI ISHIYAMA[††]

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 1020 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ソフトと記す.）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，未だに沈静化の兆しが見えない．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ものが存在する．このため，従来のIDS（Intrusion Detection System）を使った手法では，ある通信がP2P通信であるかどうかを特定することすら困難な状況にある．

2. P2P通信の検知手法

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方式と記す.）とがある．DPI方式による検知を実現するにあたっては，バイナリ解析などの手法によってP2Pソフトのプロトコルの詳細な解析が必要となり，多大なコストを要してしまう場合がある．このため，一部のP2Pソフトについては未だに検知手法が確立しておらず，検知結果に基づくアクセス制限などの強制的な管理が実施できていないのが現状である[2].

本稿では，未だ検知手法が確立していないWinnypを対象として，DPI方式によるP2P通信のリアルタイム検知機能を実装し，評価を行った結果について報告する．

2.1 Winnyp通信検知機能の概要

Winnypは，Winnyの匿名化処理を強化したP2Pソフトである．また，Winny系の利用者のうち約8%（約1.6万）がWinnypの利用者であるとの報告がある[3]．現在普及しているWinnyp v2.0b7.28のネイティブな通信を検知するための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Winnyプロトコル互換設定を無効にすることによって，従来のWinny用の検知機能では全く検知ができなくなる．本稿で実装するWinnyp通信検知機能は，このようなWinnyp専用モードで動作するWinnypをリアルタイムに検知することを目的としている．

2.2 Winnypプロトコルの特徴

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Winnyと似た構造を持つ．まずは比較のためにWinnypの原形である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[a]は11バイトの固定長のTCPペイロー

*[†] (株)日立製作所
Hitachi Ltd.
[††](株)フォティーンフォティ技術研究所
Fourteenforty Research Institute, Inc

a) TCP接続完了後に初めて送受信するパケット．

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[4]．暗号データを RC4 暗号鍵で復号すると，Winny 検知用パターンバイト列 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することができる．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 は比較的容易・高速に検知することが可能であった．

次に，Winnyp について述べる．著者らの解析[5]により，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ことがわかっている．これらの改変によって，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た．また，Winny では RC4 関数を 1 度呼び出すだけであった匿名化処理が，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且つ多処理になった（図 2）．

2.3 Winnyp 通信検知の問題

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ために，処理の飛躍的な高速化が重要な課題となる．以降の節で，それぞれの問題について具体的に述べる．

2.3.1 匿名化処理の複雑化

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ことが考えられる．

ここで表 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 の処理量の 21,437 ステップと，約 186 倍となった．これらの値には，実行時における分岐や繰り返し，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処理量の目安になる．

2.3.2 ペイロード長の多様化

Winny の通信を検知するには，最初にペイロード長が 11 バイトのパケットに絞り込んでから検査すればよかった．一方，Winnyp はランダムサイズのダミーデータがペイロードに含ま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った．

表 1 RC4 暗号鍵生成に関わるステップ数

P2P 種別	ステップ数
Winny	115
Winnyp	21,437

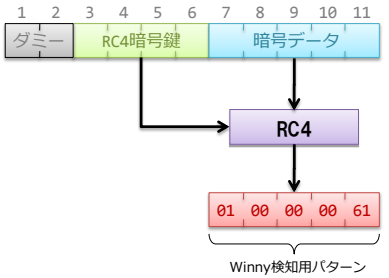

図 1 Winny 初期パケット復号フロー

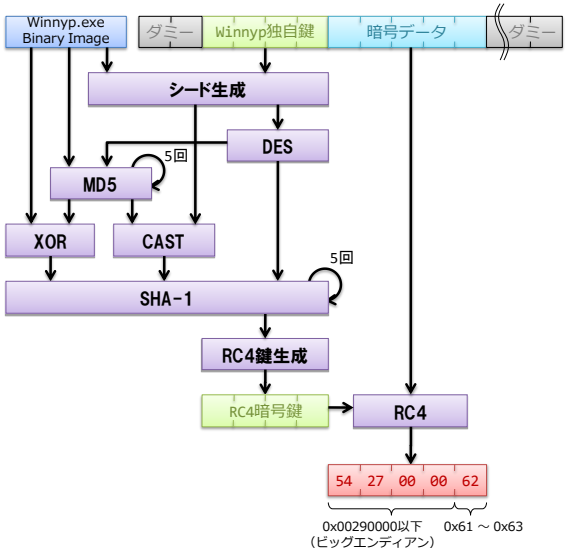

図 2 Winnyp 初期パケット復号フロー

ここでは図 3 および図 4 に，Winny と Winnyp そして参考として Share の通信をそれぞれ観測して得た初期パケットのペイロード長の分布を示す．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，

Vol.2010-CSEC-50 No.39
2010/7/2

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無く，1,460 バイトの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．

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．

これに対して 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．改めて表 2 に，Winny，Share，Winnyp のペイロード長の傾向を示す．このように，他の P2P ソフトと比較して，Winnyp だけが突出してペイロードが長く，標準偏差が大きい（ペイロード長が一定しない）特徴がみられる．

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 バイトのものに絞り込んだ上で図 1 に示した検知フローを処理すればよかったが，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 バイト以上の全てのパケットに対して 2.3.1 節に示した複雑な処理を繰り返さなければならない．単純に考えても，初期パケットを検査する頻度は Winny の場合と比較して 1,400 倍以上になる．

表 2 ペイロード長の傾向

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

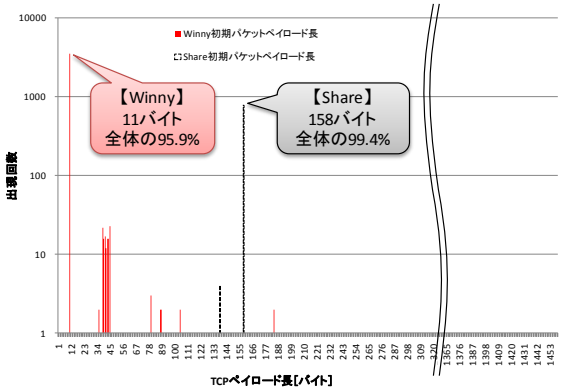

図 3 Winny と Share の初期パケットペイロード長分布

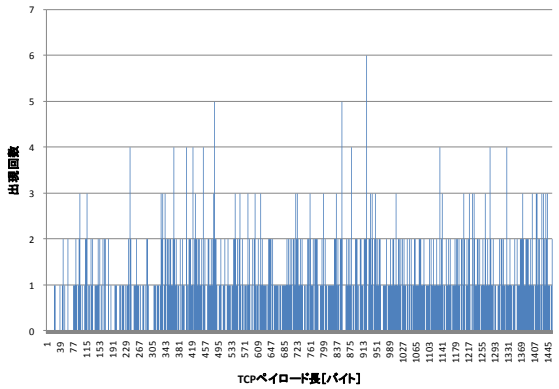

図 4 Winnyp の初期パケットペイロード長分布

3. Winnyp 通信検知機能の GPGPU 実装

本章では 2.2 節で掲げた課題を解決するために，近年 HPC（High Performance Computing）分野で注目されている GPGPU（General Purpose computing on Graphics Processing Unit）を活用した Winnyp 通信検知機能の実装を試みる．

GPGPU は，単純なデータを超並列的に大量・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．パケットデータの一部（ここではペイロードの先頭 11 バイト）を検知用データに用い，大量の検知データ（パケット）を同一アルゴリズムによって同時に処理できる Winny や Winnyp の通信の検知処理は，GPGPU の得意とするところである．そこで，GPGPU を用いた Winnyp 通信検知機能では，コンシューマ向けインターネット接続サービスの中でも最も高速な 1Gbps 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の達成を目指す．

類似の研究としては，UNIX 向けウイルス対策ツールキット ClamAV[6]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照合に GPGPU を利用して高速化した例[7]がある．本稿で実装する Winnyp 通信検知機能はパターン照合に加え，DES，MD5，SHA-1，CAST などの各種暗号処理を GPGPU によって高速に処理する．なお，GPGPU に機能を実装するにあたっては，NVIDIA 社が提供する CUDAb（Compute Unified Device Architecture）を利用する．

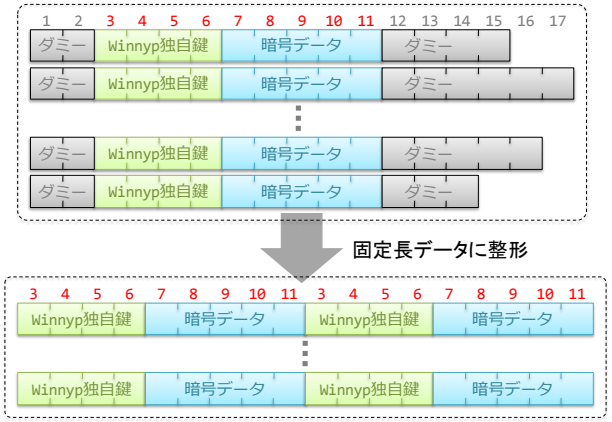

図 5 固定長データ（CUDA 演算用）

3.1 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．CUDA 上で実行される多数のスレッドは，同一のデータ（ここでは多数のパケットが結合したパケット群）を参照する．並列に動作してい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．本稿では，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，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．このため，個々のパケットのデータ長は固定長であることが望ましい．2.2 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．このため，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 5）．

3.2 カーネル実装

CUDA は，図 6 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．

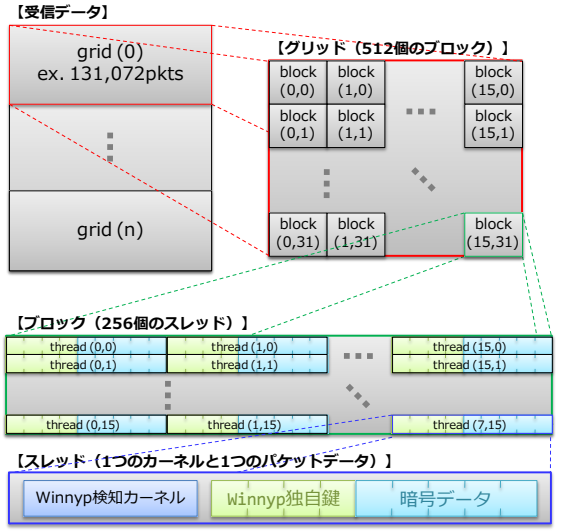

図 6 スレッドへのデータ割り当ての仕組み

CPU の処理によって取得した通信データを，前節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，GPU の種類によって異なる．著者らが利用した NVIDIA 製 GeForce 285 GTX には 240 個のプロセッサコアが搭載されており，240 スレッドの同時処理が可能である．

4. Winnyp 通信検知機能の性能評価

本章では，2.2 節で挙げた課題（処理の飛躍的な高速化）の達成を確認することを目的として，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 つの観点で評価を行う．最初に実験環境について述べ，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する．最後に，評価用データを用いて実験を行った結果について報告する．

4.1 事前準備

4.1.1 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[8] を活用した．StarBED では，全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うことが可能である．このため，インターネットを利用した実験とは異なり，想定外の通信の混入による不確定要素の排除や，Winnyp ノードの存在の把握など，事前に真値の用意が可能である．

本実験では，図 7 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[c]によって構成されるネットワーク A, B を相互に接続させた計 1,020 台のノードを構築した．Winnyp 通信検知装を中央のルータ（基幹部）のミラーポートに接続し，ネットワーク A, B 間で発生する Winnyp の通信を検知する．

各 Winny ノードは，17 個[d]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際に流通している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．ノードの HDD の容量による制約から，ファイルサイズは実際のサイズの 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・アップロードを行うようになる．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．

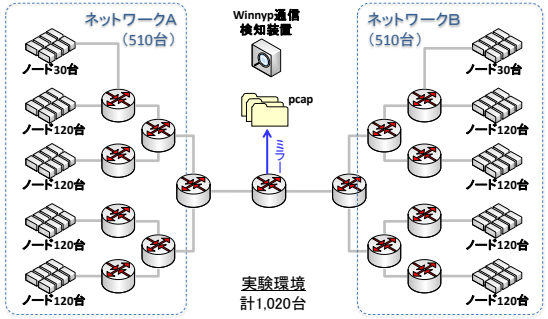

図 7 Winnyp 通信検知装置の実験環境

c) グループ H170 台に VMWare ESXi をインストールし，1020 台の Windows 環境を構築（6 仮想ノード／物理ノード）.

d) 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 3 Winnyp 通信検知装置のスペック

項目	仕様
CPU	Intel Core i7 920 (2.66GHz)
Memory	DDR3-1333 Triple Channel (12GB)
OS	Windows Vista Ultimate 64bit
GPU	NVIDIA 製 GeForce 285 GTX

表 4 検知速度性能評価用 pcap ファイルの概要

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

4.1.2 評価用データ作成

Winnyp 通信検知装置の繰り返しテストを行うために, 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速度, および検知精度に関する性能計測を同一条件下で実施できる．

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る．評価に用いる 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 通信検知装置が本来検知すべき 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留まっていたりするなどの理由により，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も評価する必要がある．このため，検知速度性能評価用 pcap ファイル（表 4）に加えて，Winnyp 以外の様々な通信（非 P2P ソフト系 6 種，P2P ソフト系 7 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.

表 5 検知精度性能評価用 pcap ファイルの概要

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
プロトコル（非 P2P ソフト系）	http, https, ftp, rtsp, pop3, dns
プトロコル（P2P ソフト系）	Winny, Share, Cabos, LimeWire, WinMX, BitTorrent, PerfectDark

表 6 評価結果

項目	CPU	GPGPU
総検知数	3,078	3,078
ユニーク IP 数 [nodes]	717	717
検知率	100%	100%
誤検知率	0%	0%
CPU 使用時間 [ms]	14,141.10	998.97
実効スループット（処理能力）T_a [pps]	70,715,85	1,001,032.91
実効スループット（処理容量）T_c [Mbps]	632.56	8954.38
CPU 性能比	100%	1,416%

4.2 評価と結果

評価用データを Winnyp 通信検知装置へ読み込ませることによって，Winnyp 通信検知機能の性能評価を行った（表 6）．また，比較対象として，GPGPU を利用しないで（CPU で実行して）得られた結果も併せて示す．

表中の実効スループット（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 $Avg(P_{size})$ と平均ヘッダ長$Avg(H_{size})$ 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 70,715.85pps であったことと，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，632.56Mbps の実効スループット（処理容量）T_cを達成したと算定した．

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

検知速度性能評価用 pcap ファイルを用いた検知速度性能評価では，CPU の処理では約 633Mbps だった実効スループット（処理容量）が, GPGPU の処理では約 8,954Mbps へと飛躍的に向上した．これは CPU と比較して 14 倍以上の高速化となる．また，検知精度性能評価用 pcap ファイルを用いた検知精度性能評価結果は検知率 100%，誤検知率 0%となり CPU の結果と全く同じであった．つまり，検知精度を低下させることなく，GPGPU の活用により飛躍的な検知速度性能の向上を達成することができた．

4.3 考察

4.2 節で算出した実効スループット（処理容量）約 8,954Mbps は，Winnyp ノードだけで構成される実験環境で観測したパケットの平均ペイロード長を元に算出している．Winnyp パケットのペイロード長は他の P2P ソフトと比較しても長くなる傾向（表 2）がみられたことから，この値がインターネット環境における実効スループット（処理容量）と乖離している可能性は否定できない．そこで，文献[9]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計算して得た結果（441.57 バイト／パケット）を元に，改めてインターネット環境における実効スループット（処理容量）を算出したところ 3,536.21Mbps（3.5Gbps）という結果を得た．これでも 1Gbps フルワイヤスピード対応の要件（2Gbps）を十分に満たす性能であることから，インターネット環境においてもコンシューマ向けインターネット接続サービスの中でも最も高速な 1Gbps のフルワイヤスピードに，本稿で開発した Winnyp 通信検知装置を適用することが可能であると判断する．

5. おわりに

本稿では，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った.その結果，検知速度性能評価においては実効スループットを 14 倍以上に高速化することに成功した．また，検知精度性能評価においては検知率 100%，誤検知率 0%と高い検知精度性能の達成を確認した．これによって，約 3.5Gbps の実効スループット（処理容量）を達成し，1Gbps フルワイヤスピード（Full Duplex）に対応できる見込みを得た．

今回は GPGPU に実装したプログラムは，CPU に実装した Winnyp 通信検知機能のソースコードの大半を流用して作成したため，GPGPU の性能を最大限に活用できてはない．今後は，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，CUDA への最適化を進める予定である．そして，10Gbps 時代を見据えて Winnyp 通信検知装置の 10Gbps 対応を目指す．さらに，トラフィックをリアルタイムに観測して，パケットキャプチャ部や CPU-GPU 間データ転送部のオーバーヘッドを考慮した評価を行い，実フィールドへの適用を目指す．

Vol.2010-CSEC-50 No.39
2010/7/2

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頂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致します．また，StarBED 上の実験環境構築にあたり，有益な助言と協力を頂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致します．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技術開発」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頂いた関係者各位に深く感謝致します．

参考文献

[1] AzureusWiki，Message Stream Encryption (aka PHE) format specification, http://www.azureuswiki.com/index.php/Message_Stream_Encryption
[2] 園田道夫（2007），「Winny はなぜ破られたのか」，株式会社 九天社，pp50-51
[3] 宮川雄一，安心・安全インターネット推進協議会 P2P 研究会「Winnyp 状況調査報告」，http://www.scat.or.jp/stnf/contents/p2p/p2p080910_4.pdf
[4] 鵜飼裕司，「検出ツールの開発者が語る，「Winny を検出する方法」」，http://itpro.nikkeibp.co.jp/article/Watcher/20060411/235051/
[5] 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」，http://www.fourteenforty.jp/research/research_papers/PacSec2008_ishiyama_jp.pdf
[6] ClamAV, “ClamAV”, http://www.clamav.net/
[7] Hubert Nghuyen（2008），「GPU Gems3」，株式会社ボーンデジタル，pp689-699
[8] Hokuriku Research Center, StarBED Project, http://www.starbed.org/
[9] MAWI (Measurement and Analysis on the WIDE Internet) Working Group, “Packet traces from WIDE backbone 2009/12/1”, http://mawi.wide.ad.jp/mawi/samplepoint-F/2009/200912011400.html