XBurner: XENebulaを利用したトラフィックジェネレータプラットフォームの設計

宮地利幸[†1,†2] 三輪信介[†1,†2] 篠田陽一[†2,†1]

ネットワーク技術の検証を行うためには、対象技術を直接的に検証するためのトラフィックや、背景トラフィックとして存在する周辺技術のトラフィックを再現する必要がある。本論文では、一般的なOSが起動するノードを利用した大規模な環境を用いたトラフィックジェネレータのプラットフォームであるXBurnerの概念設計を行う。XBurnerではStarBEDやXENebulaといった大規模な環境に存在する多数のノードをSpringOSによって制御し、一般的な通信ソフトウェアを起動することでトラフィックを生成する。これにより、利用者によるトラフィックパターンの定義や、新しいサービスへの対応などさまざまな利点が得られる。

Design issues for XBurner: A Platform for Generating Native Network Traffic

In order to build realistic environment to evaluate network technologies, traffic for evaluating target systems and by surrounding technologies of them should be introduced to these environment. We designed XBurner; a platfome to run actual application software on many nodes to ganerate native traffic. XBurner employs network testbeds such as StarBED and XENebula to build large-scale and complex network topology and SpringOS to control nodes on these testbed. Users can introduce there own network software and traffic pattern on an environment by XBurner because actual OS run on nodes in these environment. This paper describe the design of XBurner.

1. はじめに

ネットワーク技術が成熟に向かうにつれ、その信頼性や確実性が重要視されるようになっており、このような要素を検証するために、さまざまなネットワーク実験が行われている。

このようなネットワーク実験を行うための実験用の環境には、対象技術によるトラフィックと周辺技術によるトラフィックが存在する。このようなトラフィックは対象技術を動作させることにより自然に発生するものと、より現実的な実験環境を構築するためや、対象技術のパフォーマンスや手順の試験を行うために意図的に生成されるものがある。意図的にトラフィックを生成するためには、現状では専用に設計されたトラフィックジェネレータを利用することが多い。ネットワーク実験は、その目的によりさまざまな環境で行われるが、物理的に構築された環境での検証のためのトラフィックジェネレータ[1)-4)]だけでなく、ソフトウェアシミュレーション環境でもさまざまなトラフィックを生成するモジュールが利用されている。

既存のトラフィックジェネレータは、前もって定義されたトラフィック、もしくは観測されたトラフィックを再生するものが一般的であり、ネットワーク技術の開発者が新たに実装したサービスのトラフィックを生成することは困難であるなど、利用目的によって検証に利用できないこともある。

本論文では、特殊なハードウェアやソフトウェアを利用せず、一般的な計算機とOS上で動作するツールを利用してトラフィックを生成することで、利用者が実装したサービスの検証を含め柔軟なトラフィックを生成するためのプラットフォームXBurnerの設計を述べる。

また、XBurnerを利用しトラフィックに多様性および規模性を持たせるため、大規模実証環境の一つであるStarBED[5),6)]上にXENebula[7)]を利用し、AS間トポロジを構築し、数千ノードからなるトラフィック生成専用環境の構築を目指す。

本論文では、実験用の実計算機やソフトウェアシミュレータ上に模倣された実験用の計算機やネットワーク

†1 情報通信研究機構
National Institute of Information and Communications Technology
†2 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

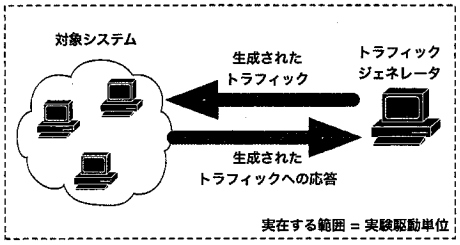

図 1 対象技術のトラフィック生成

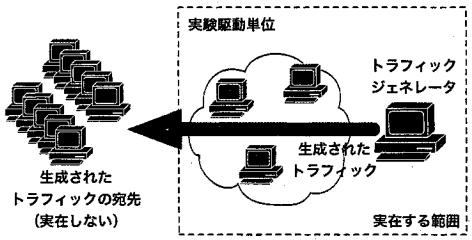

図 2 周辺技術のトラフィック生成

機器をノード、ネットワーク実験のために構築された実験用のノードや実験を管理するためのノードを含めた環境を実験駆動単位と呼ぶ。用語の定義については文献 8) が詳しい。

2. XBurner の問題意識

本章では、現状でのトラフィックジェネレータの利用ケースについて述べ、現状のトラフィックジェネレータの問題点および、我々が開発を行う XBurner での問題の解決手法について述べる。

2.1 現在のトラフィックジェネレータの利用ケース

図 1、2 は、トラフィックジェネレータの利用方法を示したものである。

図 1 は、対象システムのベンチマークやコンフォーマンス試験などに利用される、対象技術向けのトラフィックを生成するものである。このケースではトラフィックジェネレータは、対象技術のサーバーもしくは網に向けてトラフィックを生成し、負荷を与えたり、その応答として生成されるトラフィックを検証するといった用途に利用される。一方、図 2 は、周辺技術のトラフィックを生成する場合を示している。この場合は、対象技術から見たバックグラウンドトラフィックとして、既存のサービスとの相互影響を検証するために用いられるが、現状では、記録されたトラフィックを再生し、対象技術が動作しているネットワークを通過させることで実現する場合が多い。また、図 1 で示した手法により、対象技術とは関連の無い技術のトラフィックを生成させる手法も利用されている。

表 1 既存のトラフィックジェネレータの問題点

用途	問題点
対象技術 トラフィック	任意のトラフィックパターンが生成できない 新たなサービスに対応できない
周辺技術 トラフィック	任意のトラフィックパターンが生成できない 新たなサービスに対応できない インタラクティブな通信ができない

2.2 既存のトラフィックジェネレータの問題点

前節で示したトラフィックジェネレータの利用ケースでは、以下のような問題が存在する。

(1) 任意のトラフィックパターンが生成できない
(2) 新たなサービスに対応できない
(3) インタラクティブな通信ができない

新しいサービスの検証や、既存のサービスの転用などのための検証には、任意のパターンでのトラフィック生成が必要になる。一方、既存のトラフィックジェネレータでは、ある特定のプロトコルのある特定のパターンのトラフィックの生成は可能であるが、任意のパターンのトラフィックを生成することは困難であることが多い。また、同様の理由から新しいサービスの検証のために利用者が定義したプロトコルや手順にしたがったトラフィックの生成も困難である。

記録されたトラフィックを単純に再生するようなトラフィックジェネレータでは、任意の観測点で観測されたトラフィックの送出は行えるが、その応答を得ることは困難である。単にトラフィックを再生する場合には、送信されたパケットはあるポイントで破棄されればよいが、インタラクションを含めた通信を行うためには、パケットを受信したノードがそれに対して適切に対応する必要がある。このためには送信側と受信側の処理を行う要素が適切に実験駆動単位に用意されることが求められる。

表 1 に、トラフィックの発生目的とそれに対する既存のトラフィックジェネレータの問題点をまとめる。

XBurner はこれらの問題に柔軟に対応し、より現実的なトラフィックを実験駆動単位に導入し、より現実的なネットワーク実験を実現することを目的とする。

2.3 XBurner でのアプローチ

前節で説明した内容を解決するために、XBurner は以下の方針にそって開発を行う。

2.3.1 利用者独自のソフトウェア導入を許容

新たなサービスを実装した際には、その検証のための多様性をもったトラフィックを生成することが必要である[9]。XBurner では、利用者が実装したソフトウェアによるトラフィックの生成を許容するように設計を行う。これにより、トラフィックパターンについても、利用者の希望を反映したものを導入できる。

2.3.2 パケットダンプとして保存されたトラフィックの再生機能の提供

利用者が任意の観測データを再生する場合には、tcpdump や Wireshark[10] で観測したパケットダンプを利用し、トラフィックを再生することでバックグラウンドトラフィックを生成するケースも多い。このような用途に対応するため、パケットダンプとして保存されたトラフィックを容易かつ現実的に再生する機能を提供する。このような手法は現状でも利用されているが、XBurner を用いることで多数のノードから分散してトラフィックを生成することが可能となる。

2.3.3 現実的なリンク特性の導入

インターネット上を流れるトラフィックは、送信元から宛先までさまざまな特性を持ったリンクを経由することにより、さまざまな通信特性を持つことになる。XBurner で生成するトラフィックも同様の特性を持つようにリンク特性を導入する。

2.3.4 トラフィック量の調整機能

実験によって、必要とされるトラフィックの総量やプロトコル毎の割合は異なる。利用者の要望によりトラフィックの量の変更を容易に行えるインターフェースを提供する。トラフィック送出中にもその割合を変更できることを目標とする。

2.3.5 テンプレートの提供

ここまで、利用者の任意のトラフィックの導入について述べたが、典型的なプロトコルやトラフィックパターンを生成できるよう、XBurner でテンプレートを提供する。

3. XBurner の設計

2 章では XBurner が対象とする問題をまとめた。本章では、これらの要求を満たすために、XBurner で採用する技術や手法をまとめる。

3.1 XBurner への要求事項

前章で説明した XBurner のアプローチをまとめると、以下の要件を満たす必要がある。

- 一般的な OS が起動するノードの利用
- 各ノードでのソフトウェアの起動制御
- リンクエミュレータの制御
- 現実的なトポロジの導入
- これらの一括制御対応

3.1.1 一般的な OS が起動するノードの利用

XBurner では、トラフィック生成を特殊なハードウェア、もしくは専用のソフトウェアを利用するのではなく、一般的な計算機や実際に通信に利用されているソフトウェアを用いる。これにより、利用者が新たに開発したサービスのソフトウェアを容易に起動できる他、既存のサービスに対する特殊なパターンのトラフィックを生成するソフトウェアや tcpreplay[11] といった pcap 形式で保存されたトラフィック再生ソフトウェアや、harpoon[3] のような PC 上で動作する既存のトラフィックジェネレータの導入も可能である。

現実的なリンク特性の導入についても、各ノード間のリンク特性を FreeBSD や Linux といった一般的な OS 上で動作する、既存のリンクエミューレータ[12][13] を利用できる。

3.1.2 各ノードでのソフトウェアの起動制御

各ノードでは、トラフィック生成用のソフトウェアや、リンクエミュレータを外部から制御するためになんらかの手法が必要となる。

既存のソフトウェアベースのリンクエミュレータのパラメータの制御を行うことで、トラフィック量の調整機能も実現できる。トラフィック量の調節は、リンクエミュレータの設定により、実験駆動単位上のノードのネットワークインターフェースの入出力帯域などのリンク特性を制御する方法と、トラフィックを生成するソフトウェアの設定や動作頻度を変更する方法があるが、その実行は各ノードでコマンドを実行することで制御が可能である。

これらは外部から一括して行える必要があり、トラフィックを生成する前にある程度のシナリオを記述しておき、それをまとめて実行することでトラフィック生成を行う。このためのシナリオファイルを保存しておくことで、テンプレートの作成も可能である。

3.1.3 現実的なトポロジの導入

ネットワーク技術は、動作する環境のトポロジにより影響を与えられるものもある。トポロジ由来の問題を明らかにするため、規模やノード配置、そして複雑さなどを含めたトポロジを容易に変更できるような仕組みが必要となる。また、一般的な通信クライアントは多量のトラフィックを生成するために設計されていないため、必要な量のトラフィックを生成するためには、多数のクライアントを起動する必要がある。これらに対応するために、一台のノードからだけではなく、複数のノードを一括制御し、トラフィックを複数のノードから同時に生成させる。複数のノードを利用することで、送信元および送信先の IP アドレスを分散させることも可能である。

XBurner は多数のノードを確保するために、実ノードだけではなく仮想ノードを利用する。仮想ノードを利用することで、より多くの OS で動作するソフトウェアを一つの環境で利用できるほか、複数の IP アドレスを宛先、送信元とするトラフィックを生成しやすい。また、トラフィックの送信元、宛先間を直接接続するのではなく、ネットワークを挿入することで、より現実的なリンク特性の導入が可能となる。

3.2 利用技術

前節でまとめた要求事項を具体的に満足するための技術をまとめる。

3.2.1 一般的な OS が起動するノードの利用

現実的なトラフィックを生成するため、複数の計算

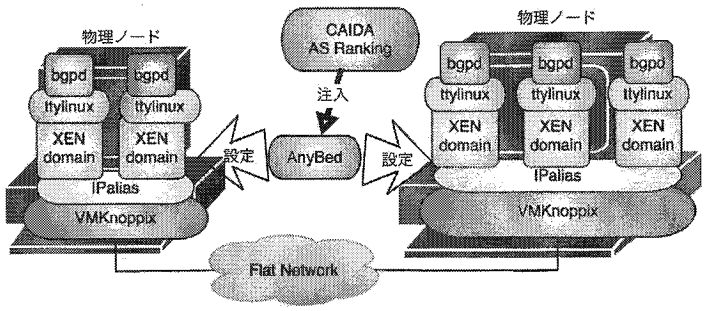

図 3 XENebula の L3 環境構築の概念図

機を利用した環境を構築する。このために 10 台程度の規模から数千台規模のトラフィック生成環境を構築する。これは、各組織で用意した小規模な PC クラスタや、StarBED のような実ノードベースの大規模な実験向けプラットフォームを想定している。また、XEN[14] や VMware[15] といった仮想ノードを利用することも可能である。

小規模な PC クラスタや StarBED を利用する場合には、SpringOS や Frisbee[16] の機能を用いて OS をインストールすることが可能である。特に SpringOS を利用した場合には、物理ノードの情報は前もってデータベース化されているため、トラフィック生成のためのソフトウェア制御に必要な情報を取得するのは比較的容易である。その一方で、仮想ノードを利用した場合は、動的に生成される仮想ノードのインスタンスの情報を何らかの方法で取得する必要がある。特に本提案では以下で述べるようにソフトウェアの起動制御のために SpringOS を利用するため、SpringOS が利用できる形のデータベースに情報をまとめる必要がある。

3.2.2 各ノードでのソフトウェアの起動制御

各ノードでのソフトウェアの起動制御には SpringOS を利用する。SpringOS は StarBED での実験を支援するための、実験支援ソフトウェアであり、実験を実行するために以下の機能を提供している。

- 実験用ノードへのソフトウェアのインストール
- L2 トポロジ設定のためのスイッチの VLAN 設定
- 実験シナリオの実行

このうち実験シナリオの実行機能を用いて、各ノードでのソフトウェアの起動を行う。具体的には SpringOS は、各実験用ノードのネットワーク設定など初期設定を行った後、シナリオファイルに記述された各実験用ノード向けのコマンドリストを実行する。ノード間の同期が必要な場合はメッセージ交換によりコマンドの実行タイミングを調整する。

XBurner では、利用者が SpringOS の言語記述に不慣れな場合でも利用できるよう、SpringOS はメッセージ交換によるノードのタイミング同期のみを担当する。すなわち、利用者は以下の役割を持ったそれぞれのプログラムを用意して、SpringOS ではそれらを起動するだけとする。

- 実験の前準備として必要なファイルのダウンロードやネットワーク設定
- 必要であれば利用するアプリケーションソフトウェアの設定ファイルの用意とアプリケーションソフトウェアの起動
- トラフィック発生
- トラフィック生成の停止およびアプリケーションソフトウェアの終了

実験ノードでは、SpringOS のクライアントソフトウェアが起動しており、サーバプログラムからのメッセージ内容により上記の機能を使い分ける。これにより利用者は、利用するアプリケーションソフトウェアにとって適切な言語でのトラフィック生成が可能となる。ただし、実験用ノードには当然それらの処理系をインストールする必要がある。

これらの段階をある程度定型化することで SpringOS の制御設定はテンプレート化が可能である。

3.2.3 現実的なトポロジの導入

単純なトラフィック生成は、実験用ノード 1 対 1 で対応が可能であるが、要求されるトラフィックが大規模化・多様化するにつれて、トラフィックを生成する環境自体も複雑化させる必要がある。このような環境構築のために、XBurner では PC ベースのクラスタと XENebula を利用した大規模な仮想化環境を利用する。

XENebula は XEN[14] を利用した大規模実証環境を構築するツールであり、XBurner は XENebula 上に現実的な AS 間トポロジ環境を構築し、さらに対象トラフィックを生成するクライアント、サーバも XENebula 上に用意する。これにより、ネットワークトラフィックを生成するソフトウェアを起動できる多数のノードと、より複雑な経路を環境中に導入できる。XENebula を利用して AS 間トポロジを作成する概念図を図 3 に示す。

以上のような手法により構築した環境に一般的な

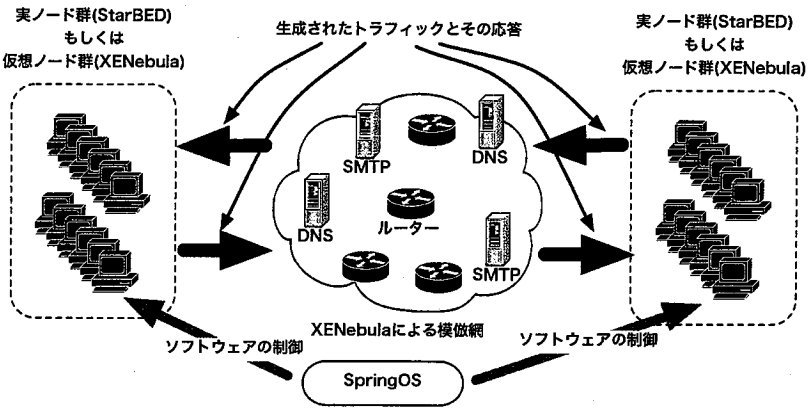

図 4 XBurner の概念図

サービスを提供するサーバを用意することで利用者が導入したアプリケーションの補助を行う。たとえば、通信を開始するノードは一般的に DNS サービスを用いて、通信相手の IP アドレスの解決を行うため、DNS などのサービスを XENebula 上に構築し、その現実性を高めることも考慮する。特に DNS については、DNS 網を模倣しその応答を生成するといった試み[17]も行われており、このようなサービスを導入する手法も考えられる。

最終的には XENebula を利用して StarBED 上に数千から数万 AS およびノードからなる環境を構築し利用する。図からわかるように XENebula は XEN を利用して一台の実ノード上で多数の仮想ノードを起動する。そして、XEN の DomU 上で動作する ttylinux で Quagga[18] の bgpd を起動し実際に経路の交換を行う。AS 間接続のデータは CAIDA の AS Relationship[19] で提供されているものを利用し、AnyBed[20] を用いて、各 bgpd の設定ファイルを生成している。

ただし、SpringOS は、StarBED と同様の構成を持ったクラスタ環境での利用が可能であり、すでにいくつかの拠点で利用されている。したがって、XBurner は StarBED での利用に限定されるわけではなく、さまざまな PC クラスタ上での利用が可能である。

3.2.4 そ の 他

現状では AnyBed および XENebula は一つの BGP 網もしくは OSPF 網を構築する。このような網の上でトラフィックを生成すると、そのトラフィックはそれぞれ最適な経路を経由して送信先ノードに適切にルーティングされる。しかし、生成したトラフィックすべてをあるシステムに流入させたい場合には、単一リンクにすべてのトラフィックを流入させる必要がある。このような要求を満たすために、AnyBed および XENebula で異なるアドレスレンジを持つ二つのネットワークを構築し、これを相互に接続することでトラフィックを単一リンクに向ける。現在の AnyBed は利用するアドレスレンジを指定できないため、これに関する拡張を行う。

さらに、ファイルをダウンロードするようなサービスを利用する場合には、ディスクの容量が問題となる。特に仮想ノードを利用した場合などについては利用できるディスク容量には限りがあるため、ダウンロードしたファイルを何らかの方法で定期的に削除する必要がある。このためには単純にファイルを削除するという方法と、そもそもディスクに書き込ませないという二通りのアプローチが存在する。できるだけ利用者の負担を軽減するため、Filesystem in Userspace(FUSE)[21] を利用し、ディスクにファイルが書き込まれないようなファイルシステムを実装する。

図 4 は XBurner の概念図である。これまでに説明したように、XBurner は StarBED もしくは同様の PC クラスタ環境もしくは XENebula による仮想ノード環境を利用しトラフィックを生成する。各 PC ノード上ではトラフィックを生成する一般的な通信用ソフトウェアや、ソフトウェアベースのトラフィックジェネレータが動作しており、その起動やパラメータの変更は SpringOS により制御される。また、同様にリンクエミュレータおよび DNS といった補助的なサービス用アプリケーションソフトウェアも XENebula 上で動作し、SpringOS によって制御される。

4. 実装までの課題

本手法では既存のツールを組み合わせることで現実的なトラフィックを生成することを目標としている。したがってトラフィックを生成すること自体はそう難しいことではない。SpringOS や XENebula、各種リンクエミュレータおよびトラフィックを生成する通信ソフトウェアやソフトウェアベースのトラフィックジェ

表 2 XBurner の特徴と改善点

特徴	改善点
一般的なソフトウェアの導入を許容	利用者のソフトウェアの利用が可能 利用者が定義したトラフィックパターンへの対応 ソフトウェアベースのリンクエミュレータの利用が可能 ソフトウェアベースの既存のトラフィックジェネレータの導入が可能
SpringOS によるソフトウェア制御	容易な環境整備 任意のタイミングでのコマンド実行による柔軟なトラフィックパターンの生成
StarBED と XENebula によるノード数の確保	大規模な実験駆動単位の構築 複数の OS の導入を許容することによる現時的なトラフィックの導入 大規模な AS 間トポロジの挿入による現実的なトラフィックの導入

ネレータはすでに提供されているものである。

本研究での課題は以下の点である。

- 既存のツール群を組み合わせた環境構築の容易化
- 現実的なトラフィックパターンの検討
- トラフィック量の調整

以下でそれぞれについてまとめる。

4.1 既存のツール群を組み合わせた環境構築の容易化

すでにトラフィックを生成する部品自体はそろっていると考えられるが、それぞれを一つずつ設定を行うコストは大きい。これらをある程度自動化するために、XBurner の制御部ともいえる SpringOS の機能拡張やこれらを包括して実行するためのインターフェースが必要となる。

4.2 現実的なトラフィックパターンの検討

トラフィックの生成自体はアプリケーションソフトウェアを実行することで行えるが、どのようなトラフィックが現実的といえるかについては議論が必要である。各プロトコルの割合やリンク特性など検討すべき点は多い。インターネットを対象とした場合には、そのトラフィック傾向は国や組織の種類などによって大幅に異なるため、単一のトラフィックパターンでは対応できない場合も多いと考えられる。本研究ではいくつかの実際のトラフィックパターンを参考とし、それに近いトラフィックパターンを生成するための議論を行っていく。

また、通信用のソフトウェアをどのように動作させた際にどの程度のトラフィックを生成するのかや、リンクエミュレータを挿入した際の実際のトラフィックへの影響などについての調査も必要となる。

4.3 パケットダンプファイルからの現実的なトラフィックの生成

パケットダンプとして保存されているトラフィックはネットワーク上のある一点で観測さたものであり、観測地点で観測されたパケットおよびフレームの送出・受信時間のみが保存されている。したがって、あるパケットもしくはフレームが実際に送信された時間は不明である。

ある網を考え、一点で観測した情報からその網全体にある程度現実的なトラフィックを導入したい場合も多いと考えられるが、その場合には、観測されているトラフィック情報から送信ノードと受信ノードのホップ数やリンク特性を推定し、生成するトラフィックパターンに反映する必要がある。

4.4 トラフィック量の調整

任意のトラフィックパターンを作り出すためには、全トラフィックに占めるプロトコル別の割合を変更したり、全トラフィック量そのものを実験中に動的に調整したいという要望も出てくる。このような用途のために、既存のリンクエミュレータのパラメータ変更への追従の精度の検証などが必要である。

4.5 トラフィック生成の精度の確保

XBurner は、インターネット上と同様に自律分散的に動作する多数のノードをある程度の粒度で制御し、最終的な出力としてトラフィックを生成することを目的としている。一方で、最終的なトラフィックが実際にどの程度「現実的」であるかや、その制御粒度についても検討が必要である。

5. おわりに

本論文では、柔軟かつ現実的なトラフィックを生成するために XBurner の設計を行った。XBurner は多数の一般的な OS が起動するノードからなる環境において、それぞれのノードが利用者が新たに開発したモノを含め、一般的な通信ソフトウェアによりトラフィックを生成するため、より現実的なトラフィック生成が可能であるとともに、利用者による拡張も容易である。2.2 節で示した既存のトラフィックジェネレータの問題は、表 2 にまとめた XBurner の特徴により改善が可能である。ただし、現段階は設計段階であり、今後実装が必要である。

XBurner は一般的なアプリケーションソフトウェアを一括して実行し、その結果としてトラフィックを生成するためのプラットフォームである。仮想ノードを利用し一台の物理ノードのリソースを共有するため、一台で大量のトラフィックを生成するといった用途には不向きであるといえる。この点も踏まえ、XBurner を道具として利用して、現実的なトラフィックを生成するための議論は今後の課題となる。まずはトラフィックパターンの検討や現実的なリンク特性の導入についての議論を、実際に生成したトラフィックを観察し

た上で検討し、典型的な通信ソフトウェアを利用したトラフィック生成のためのテンプレートを用意する。同時に大規模なネットワークを経由することでトラフィックにどのような影響があるのかについても検討し、XBurner のトラフィック生成アルゴリズムに組み込んでいく予定である。

参考文献

1) Spirent Communications (online): http://www.spirentcom.com/ (2009).
2) Agilent N2X (online): http://advanced.comms.agilent.com/n2x/ (2009).
3) Sommers, J. and Barford, P.: Self-Configuring Network Traffic Generation, *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement* (2004).
4) Alessio Botta, Alberto Dainotti, A. P.: Multi-protocol and multi-platform traffic generation and measurement, *INFOCOM 2007 DEMO Session* (2007).
5) Miyachi, T., Chinen, K. and Shinoda, Y.: StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software, *International Conference on Performance Evaluation Methodlogies and Tools (Valuetools) 2006* (2006).
6) 宮地利幸，中田潤也，知念賢一，ラズバン・ベウラン，三輪信介，岡田崇，三角真，宇多仁，芳炭将，丹康雄，中川晋一，篠田陽一：StarBED: 大規模ネットワーク実証環境，情報処理，Vol.49, No.1, 情報処理学会 (2008).
7) Miwa, S., Suzuki, M., Hazeyama, H., Uda, S., Miyachi, T., Kadobayashi, Y. and Shinoda, Y.: Experiences in Emulating 10K AS Topology with Massive VM Multiplexing, *The First ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures (VISA'09)* (2009).
8) 宮地利幸，三輪信介，篠田陽一：ネットワーク実験の支援技術，マルチメディア，分散，協調とモバイル（DICOMO2009）シンポジウム論文集，情報処理学会シンポジウムシリーズ，pp.797–807 (2009).
9) 野中雄太，知念賢一，宇多仁，宮地利幸，篠田陽一：StarBED を用いたサーバ負荷試験の実現，マルチメディア，分散，協調とモバイル（DICOMO2007）シンポジウム論文集，情報処理学会シンポジウムシリーズ，pp.199–204 (2007).
10) Wireshark (online): http://www.wireshark.org/ (2009).
11) Tcpreplay (online): http://tcpreplay.synfin.net/trac/ (2009).
12) Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols, *ACM Computer Communication Review*, Vol. 27, No.1, pp.31–41 (1997).
13) Hemminger, S.: Network Emulation with NetEm, *Australia's National Linux Conference* (2005).
14) Pratt, I.: Xen and the Art of Open Source Virtualization, *the Proceedings of the Linux Symposium Volume Two* (2004).
15) VMware (online): http://www.vmware.com/ (2009).
16) Hibler, M., Stoller, L., Lepreau, J., Ricci, R. and Barb, C.: Fast, Scalable Disk Imaging with Frisbee, *In Proceeeding of the 2003 USENIX Annual Technical Conference* (2003).
17) 三輪信介，宮本大輔，櫨山寛章，井上大輔，門林雄基：模倣 DNS によるマルウェア隔離解析環境の解析能向上，pp.19–24 (2008).
18) Quagga Software Routing Suite (online): http://www.quagga.net/ (2009).
19) CAIDA AS Relationships (online): http://www.caida.org/data/active/as-relationships/ (2009).
20) Suzuki, M., Hazeyama, H. and Kadobayashi, Y.: Expediting experiments across testbeds with AnyBed: a testbed-independent topology configuration tool, *Proceedings of 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)* (2006).
21) FUSE: Filesystem in userspace: http://fuse.sourceforge.net/ (2009).