Vol.2013-HPC-140 No.22
2013/8/1

ユーザインタラクションと連動したフロー制御を行う大規模可視化

木戸 善之[1,a)]　古都 知哉[2]　渡場 康弘[1,2]　伊達 進[1]　阿部 洋丈[3]　市川 昊平[4]　山中 広明[5]
河合 栄治[5]　下條 真司[1,5]　竹村 治雄[1]

概要：科学全般において大規模な可視化が必用とされ，可視化装置とミドルウェアに注目を集めている．複数のディスプレイで構築するタイルドディスプレイは大規模な可視化を実現する技術であり，ネットワークを通じてアプリケーションやコンテンツを表示することができる．動画配信などの大容量ストリーミングとは異なり，タイルドディスプレイでのアプリケーション表示ではユーザのインタラクションが発生し，その結果を画面に反映することとなるが，ネットワークの遅延，障害がユーザインタラクションに多大な影響をおよぼす．そのためタイルドディスプレイでのユーザインタラクションを考慮したネットワークの制御が必要不可欠となる．先行研究では，SAGE によるタイルドディスプレイと OpenFlow を用い障害検知による冗長経路の変更について取り組んだ．本研究では，ユーザ操作によって生じるデータ転送量の変更に対応するため OpenFlow を用いたフロー制御に取り組み，インタラクティブネスの向上を目指す．

1. 背景

近年のプロセッサ性能，ネットワーク性能の向上により，クラウドやグリッドに代表される大規模分散計算技術はますます実用的かつ有用な技術へと成熟しつつあり，学術，産業のあらゆる分野における利用が急速に進んでいる．その結果，今日，われわれは大容量データを短期間の間に計算処理，解析処理し，さらに大容量の計算結果データ，解析結果データを取得することができる．実際，創薬研究の分野では，タンパク質と化合物の振る舞いを確認・検証するために分子動力学シミュレーションを利用するが，そのようなシミュレーションにおいては，1 試行に対して 2〜5GB の中間結果，最終結果を生成することがありうる [1][2]．一般的に，このような計算結果や解析結果データは数値データとして出力される．そのため，扱うべきデータの大容量化に伴い，そのデータの意味を直感的に提示することのできる可視化がますます必要かつ重要になりつつある．

大容量データの直感的な理解・解釈のためには，そのデータがもつ時間的かつ空間的な解像度情報を失わせることなく可視化することが重要となる．しかし，今日では，大規模分散計算によって取得可能な計算結果，解析結果データを情報欠損なく表示可能とする可視化装置はほとんどないのが現状である．例えば，今日では一般的な 1 千万画素級のデジタルカメラを例にとってみても，そのデジタルカメラで記録可能な画像を画素情報を欠損させることなく表示させることができるディスプレイは非常に高価なものが存在するのみである．このため，近年では複数台のコモディティ LCD モニタを方形上に組み合わせ，それら LCD モニタを複数台の PC で連携駆動させることで，数千万〜数億画素の高精細画像を表示可能とするタイルドディスプレイ技術 (図 1) に関する研究開発が盛んとなっている．

とりわけ，今日利用できるタイルドディスプレイ技術の中でも，イリノイ大学シカゴ校の EVL が研究開発を推進中のミドルウェア SAGE[3] への注目が集まっている．SAGE は，インターネット上の複数の拠点上のデータを，ネットワークストリーミング技術によって配送することで，それらタイル上に構成された複数台のモニタ上に同時に可視化することを可能にする．そのため，複数拠点の研究期間や大学と共同研究を日常的に推進する e-science 分野の研究者らからの SAGE への期待と関心は高まる傾向にあり，多くの研究グループが SAGE によるタイルドディスプレイ

[1] 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[3] 筑波大学大学院コンピュータサイエンス専攻
Department of Computer Science, University of Tsukuba
[4] 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, NAIST
[5] 情報通信研究機構テストベッド研究開発推進センター
Network Testbed Research and Development Promotion Center, NICT
[a)] kido@cmc.osaka-u.ac.jp

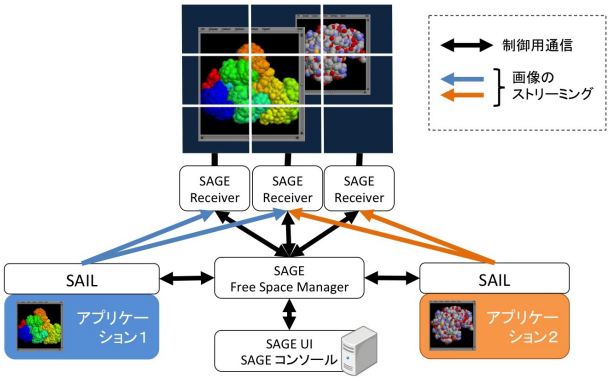

図 1 SAGE の構成図

を複数拠点に実際に構築・配備し，研究，教育分野への実証実験を活発に行っている．しかし，この SAGE によるタイルドディスプレイは，その可視化性能をネットワーク性能に大きく依存するにも関わらず，ネットワーク制御に関する機能が具備されていないことが問題である．SAGE ではタイルドディスプレイ上のどの位置にどのようなサイズでアプリケーションからのストリーミング映像を表示するかに依存して，タイルドディスプレイを構成する各計算機と映像を生成するアプリケーション間のネットワーク通信量は動的に変動する．つまり，ユーザによるアプリケーションウィンドウの配置やサイズの変更などのインタラクションに応じて，ネットワーク流量は急激に変動し，時にはネットワークの輻輳を生じさせる可能性がある．しかしながら，現状の SAGE はこのようなユーザインタラクションに応じたネットワーク流量の変動に関して特別な制御を実装していない．

本研究では，そのような観点から，SAGE によるタイルドディスプレイ上で，ユーザインタラクションによって変動するネットワークパラメータと連動させ，ネットワークを効率的に利用した大規模可視化を実現することを目的とする．より具体的には，ユーザインタラクションによって変動するネットワークパラメータを基に，ネットワークの動的なプログラミング制御を可能とする Software Defined Network (SDN) を応用したフロー制御を行う大規模化可視化技術を構築する．本稿では，タイルドディスプレイ上でのユーザ操作に合わせたネットワークフローの制御について取り組み，その内容を記述する．

本論文の構成は，2 節で SAGE について概説し，3 節で技術要素と提案手法について述べる．4 節でまとめと今後の課題ついて述べる．

2. SAGE

2.1 SAGE のアーキテクチャ

SAGE は分散レンダリングに対応し，かつ複数のアプリケーションをタイルドディスプレイに同時表示を可能としたミドルウェアである．SAGE は，アプリケーションが生成するコンテンツを，タイルドディスプレイ上の表示位置と各ディスプレイの描画を担当する計算機に応じて，コンテンツを分割してストリーミングする特徴を備えている．各アプリケーションからのコンテンツを一箇所に集めてから描画担当の計算機群に配信するような方式ではなく，アプリケーションと描画担当計算機が直接的に分散通信を行うため，規模に対してスケーラブルなアーキテクチャとなっている．また SAGE では複数のタイルドディスプレイに対し，画面を同期して表示する事が可能である．複数のアプリケーションが同時に表示される様を，遠隔地での複数のタイルドディスプレイにて見ることができ，研究者らは巨大な可視化コンテンツを遠隔地においても共有することができる．図 1 は一般的な SAGE の構成図である．SAGE は主に 1) SAGE Receiver，2) SAGE Free Space Manager，3) SAIL (SAGE Application Interface Library)，4) SAGE UI および SAGE コンソールの 4 つのコンポーネントから成る．

SAGE Receiver がディスプレイ描画担当サーバであり，タイルドディスプレイを構成する各ディスプレイに接続された計算機上で動作する．1 つの SAGE Receiver が複数のディスプレイの描画を担当することもでき，図 1 の例では，各 SAGE Receiver がそれぞれ縦に 3 面ずつのディスプレイの描画を担当している構成を図示している．SAGE Free Space Manager はタイルドディスプレイ上のコンテンツの表示位置を管理している管理サーバである．この SAGE Free Space Manager が各アプリケーションからのコンテンツをタイルドディスプレイ上のどの座標位置に表示するかを管理しており，その情報を SAGE Receiver とアプリケーションに組み込まれる SAIL に提供している．図 1 の例では，SAGE Free Space Manager はアプリケーション 1 のコンテンツは左側 6 面の中央辺りに表示し，アプリケーション 2 のコンテンツは右上 4 面の中央辺りに表示するという座標情報を有している．

SAIL は SAGE におけるアプリケーション用の描画 API を提供する．アプリケーションはこの SAIL を通じてタイルドディスプレイ上に表示するコンテンツを送信する．SAIL は SAGE Free Space Manager より得た情報に基づき，アプリケーションから提供された画像データを表示先のディスプレイに応じて分割し，そのディスプレイの描画を担当する適切な SAGE Receiver に送信する．この分割と送信先 SAGE Receiver の選択は自動的に行われるため，アプリケーション側からは意識すること無く，描画処理に専念できる．

図 1 の例ではアプリケーション 1 のコンテンツは 2 つに分割され，タイルドディスプレイの左側 6 面の描画を 2 台の SAGE Receiver に送信される．その一方でアプリケーション 2 のコンテンツは右側 2 台の SAGE Receiver に送信される．このようにして，SAIL を通じてアプリケーショ

ンからコンテンツデータを受け取った SAGE Reciever は，同じく SAGE Free Space Manager から得た情報に基づき，コンテンツを適切な座標位置に表示する．

SAGE UI および SAGE コンソールは SAGE におけるユーザインタフェースを提供する．これらユーザインタフェースからは SAGE Free Space Manager が管理するコンテンツの表示位置とサイズをインタラクティブに変更することが可能である．コンテンツ表示領域の変更は直ちに SAGE Receiver と SAIL に伝達され，SAIL からのコンテンツ表示宛先が変更されたり，SAGE Receiver でのコンテンツ描画位置の変更が反映される．

2.2 SAGE の問題

前述のように，SAGE では複数ディスプレイへのマルチストリーミングの実現，かつ分散レンダリングや複数アプリケーションの実行に対応するなど，ネットワークに依存する機能を多数実装しながら，ストリーミングに利用するネットワークを制御する実装がなされていない．そのため，コンテンツを生成するアプリケーションと SAGE Receiver の間は既存のネットワークルーティング手法がそのまま利用され，ネットワーク輻輳などにより，ネットワークの性能を適切に維持することができず，大規模可視化を行う上で大きな障害となる可能性がある．

SAGE では大規模な画面をネットワークでストリーミングするという特性上，ネットワークの性能がユーザの操作性に与える影響は大きい．ユーザのマウスやキーボード操作がイベントとしてアプリケーションに伝播し，その反応を画面に反映する，といったスタンドアローンのアプリケーションではスムーズにこなせる処理も，ネットワークを介することで画面表示遅延やユーザ操作における反応速度など，ネットワークのスループットや応答速度などがユーザ操作性に多大な影響をおよぼすこととなる．

したがって，SAGE におけるネットワーク利用状況に応じて，ネットワークルーティングを制御し，ネットワークを効率的に使用する技術の開発が強く望まれる．より具体的には，各アプリケーションのタイルドディスプレイ上での配置やサイズなどの情報から，アプリケーションと SAGE Receiver 間で実行されるネットワークストリーミングによるネットワーク流量を把握し，それに応じたネットワーク制御を実現することによって，ネットワーク性能を維持することが必要である．このようなアプリケーションレベルのネットワーク利用形態に応じて細粒度にネットワーク制御を実施するためには，既存のネットワークルーティング技術では実現不可能であり，近年提唱されつつあるソフトウェアによってネットワークを完全に制御する SDN (Software Defined Networking) の活用が不可欠である．

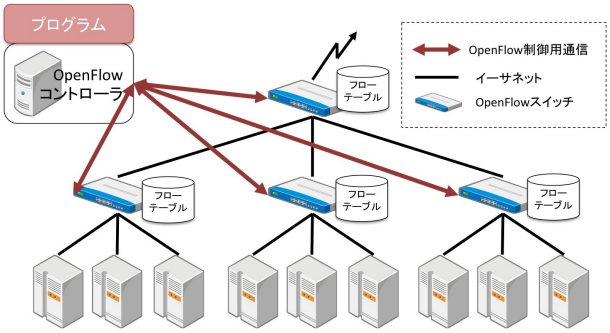

図 2 OpenFlow スイッチとネットワークの構成図

3. 技術要素と提案手法

3.1 SDN

SDN (Software Defined Networking) とはソフトウェアによってネットワーク全体を制御・管理可能とする技術の総体を示す．OpenFlow[4] は SDN を実現する技術の 1 つであり，ネットワークスイッチにおける制御やプロトコルを定義している．OpenFlow を用いることによって，ネットワークの挙動を動的に変更することが可能となり，またプログラマブルな機器も現在，市場に出始めている．OpenFlow プロトコルで制御御可能なネットワークを OpenFlow ネットワークといい，OpenFlow プロトコル対応の複数の OpenFlow スイッチと，これら OpenFlow スイッチを中央集約的に制御・管理する OpenFlow コントローラから成る．プログラマブルな箇所はこの中央コントローラにある．

図 2 に OpenFlow ネットワークの構成図を示す．OpenFlow では，各 OpenFlow スイッチ内にフローテーブルと言われるフローのマッチングルールとそのフローに該当するパケットの転送先や処理方法が書かれたフローエントリを格納するデータベースが実装されている．OpenFlow スイッチに入ってきたパケットはまずこのフローテーブルに処理方法が記述されていないか検索され，マッチするフローエントリがある場合は，そこに記述されている処理を実行する．フローテーブルにエントリがないパケットについては，OpenFlow コントローラに処理方法が問い合わせられ，コントローラによって新しいフローエントリが動的に書き込まれることによって処理される．この時，コントローラは，障害検知や帯域測定などにより集計した結果に基づいて，パケットの行き先を動的に決定することが可能であり，動的で細粒度なネットワーク制御を実施することが可能となる．

3.2 先行研究

OpenFlow スイッチによって構成されたネットワークと SAGE を用いたタイルドディスプレイにおいて，ネットワーク障害検知，回避を行った先行研究がある [5]．この研究では OpenFlow スイッチで接続されたネットワークにお

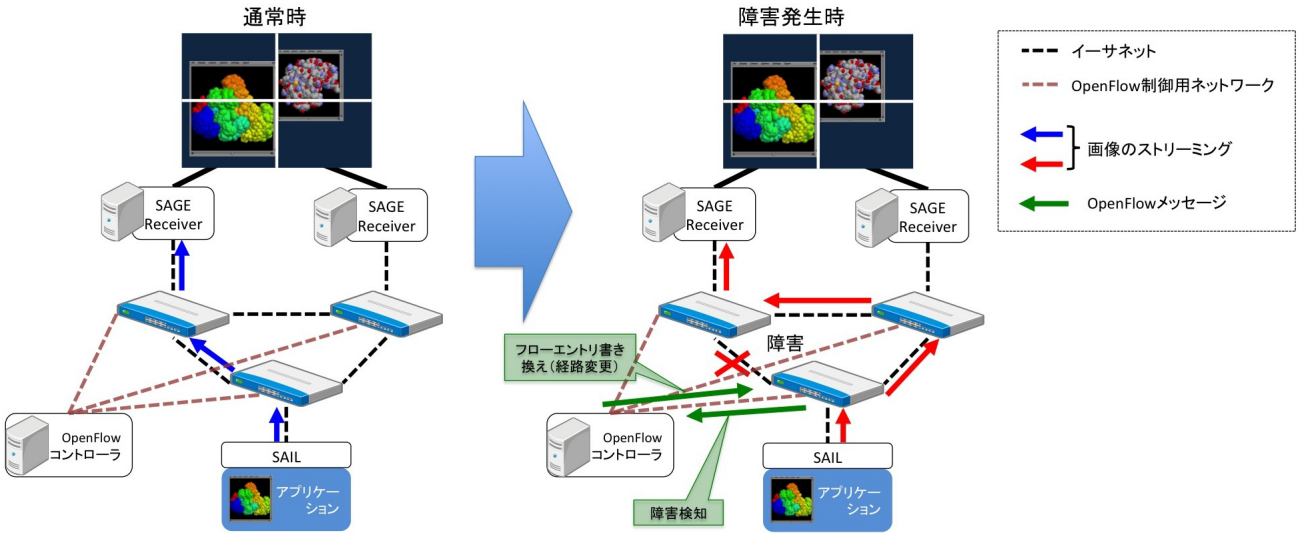

図 3 ネットワーク障害を検知，回避する研究の概要図

いてネットワーク障害を検知し，障害のある経路を動的に避けることを行った．図 3 は，SAGE と OpenFlow スイッチで構成されたシステムにて，ネットワーク障害が発生した場合の経路変更を動的に行う様を表した概要図である．OpenFlow コントローラでは，OpenFlow スイッチを通じてネットワークの状態を取得することができる．先行研究では OpenFlow スイッチ間のレイテンシを取得し，応答がない場合を障害と判断する．障害区間を避ける経路を探索し，現在送信中のパケットの経路を動的に変更する．先行研究では，アプリケーションの稼働中にネットワーク障害を意図的に発生させるのと同時に SAGE ディスプレイ上でのフレームレートを計測を行い，約 3 秒で回復すると報告している．先行研究では，SAGE と OpenFlow に対し，3 つのモジュールを追加している (図 4 参照) .

a) ネットワーク障害検知機能

ネットワーク障害検知機能は，OpenFlow スイッチ間のレイテンシを測定し応答なしであれば障害と判断する．その後，当該経路をブラックリストに情報を追記する．

b) トポロジ管理機能

ネットワークの論理構造を事前に知っておく必要があるためトポロジ管理機能として実装した．トポロジ管理機能ではネットワークのトポロジ情報をグラフとしてファイル出力する．

c) 動的パケットフォワーディング機能

トポロジ情報とブラックリスト情報を元に，ブラックリストにある経路を避けるようにフォワードルート情報を書き換える．これにより OpenFlow スイッチは障害を回避するようにパケットの転送を行う．

3.3 提案手法

前節で述べた先行研究を踏まえた上で，ユーザ操作に

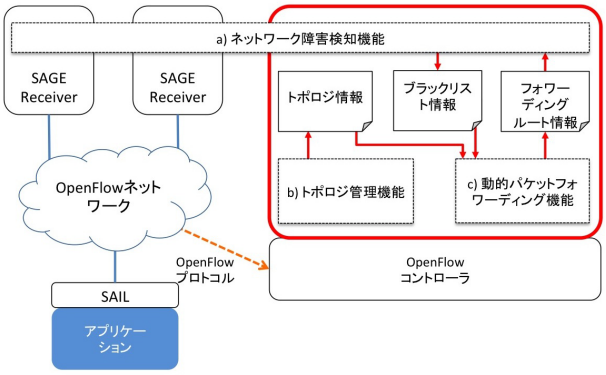

図 4 先行研究でのアーキテクチャ

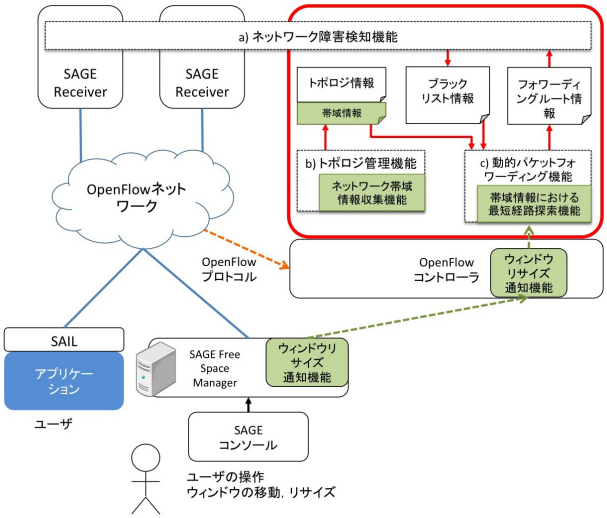

図 5 アーキテクチャ概要図

よって生じるデータ転送量の変化に対応するフロー制御について取り組む．OpenFlow スイッチで構成されたネットワークにタイルドディスプレイが接続された環境にて，タイルドディスプレイ上でのユーザの操作，具体的にはウィンドウのサイズ変更や移動をディスプレイ間で行うと想定

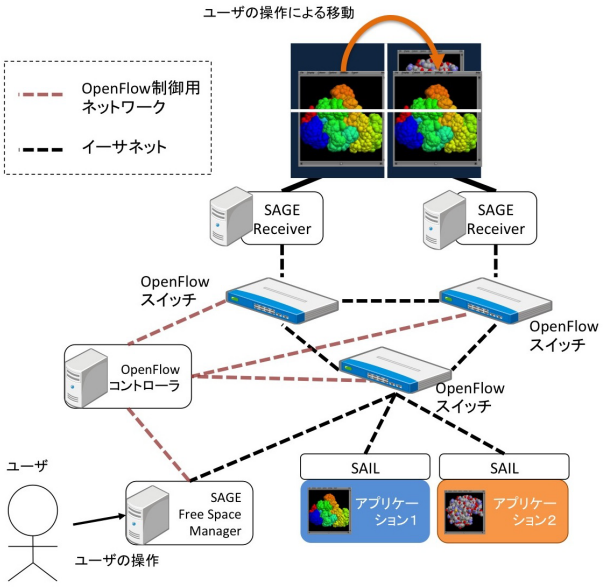

図 6 タイルドディスプレイと OpenFlow スイッチのネットワークの構成

する（図 6 参照）．タイルドディスプレイ上にてウィンドウを操作する場合，ディスプレイをまたいでウィンドウをサイズ変更，移動することが頻繁に起こりうる．ディスプレイをまたいでウィンドウを移動することは，高解像度のコンテンツつまり大容量のデータの送り先が変わることになる．複数のアプリケーションが動作している状態において，ウィンドウを移動することはネットワークの輻輳が考えられる．ネットワーク上での輻輳は，高解像度コンテンツをストリーミングで送信するに十分でない帯域のネットワークを通ることとなり，タイルドディスプレイにおいてフレーム落ちやフレームレートの低下など，高解像度コンテンツの劣化につながる．

そこで，ウィンドウ操作に伴うネットワーク上での輻輳を回避するため，ウィンドウ操作をトリガーとしてネットワーク経路のパケット流量の少ない経路を選択し，経路変更を行うシステムを提案する．図 5 は，システムのアーキテクチャの概要図である．先行研究を踏襲しているため図 3 からの拡張となる．追加モジュールは緑の箇所に該当する．まずユーザは，SAGE コンソールによってウィンドウのサイズ変更，移動を行う．サイズ変更と移動は，Free Space Manager，正確には共有のコアライブラリでリサイズ機能として実装されており，ウィンドウリサイズ通知機能を Free Space Manager に組み込んだ．Free Space Manager からウィンドウリサイズの通知を受け取った OpenFlow コントローラでは，帯域情報を元に，現在パケット量などのトラフィック量の少ない経路を選択し，フォワーディングルート情報に追記する．その情報を元に OpenFlow スイッチは経路変更を行う．経路選択に必要な情報としては，経路におけるトラフィック量が挙げられるが，トラフィック量の集計機能はトポロジ管理機能にネットワーク帯域情報収集機能として追加した．帯域情報としてトラフィック量を追記することで，トポロジ情報，論理的なネットワークのグラフデータにトラフィック量を重みとして負荷させた．このトポロジ情報は 5 秒毎のインターバルで取得している．ネットワーク帯域とトラフィック量により，利用可能な帯域が算出でき，利用可能な帯域を重みづけしたネットワークグラフの最短経路問題として経路選択を行った．各機能について以下，箇条書きにて述べる．

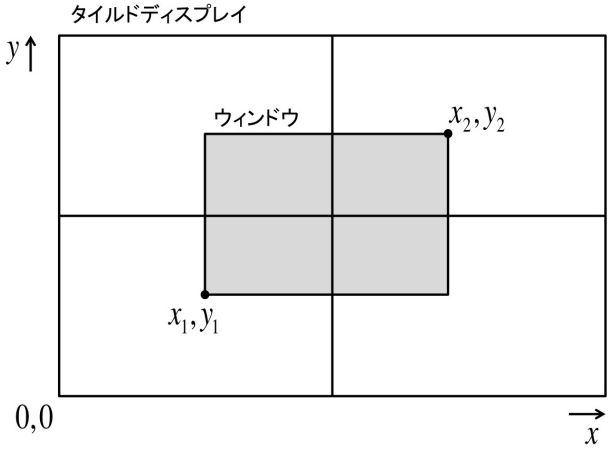

図 7 タイルドディスプレイのおけるウィンドウ座標の指定

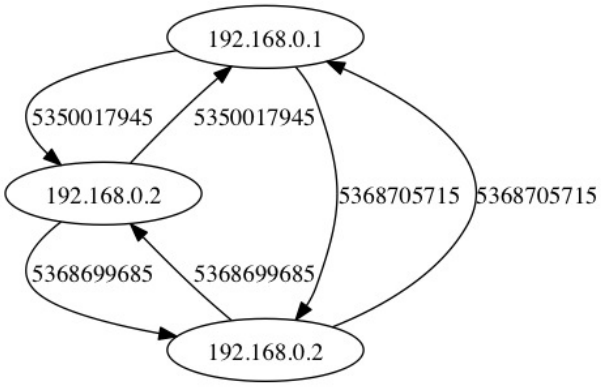

図 8 トポロジ情報と空いている帯域のグラフ

a) ウィンドウリサイズ通知機能

SAGE コンソールから受け取ったウィンドウのサイズ変更，移動のイベントを起点として，OpenFlow コントローラに対し通知を行う．通知する内容はウィンドウごとに発番されるウィンドウ ID とサイズ変更後の座標になる．この座標は，画面左下を x,y 座標を (0,0) として，絶対座標として表される．また座標はウィンドウ左下の座標と，ウィンドウ右上の座標が含まれる．図 7 は，タイルドディスプレイとウィンドウの模式図である．移動先の座標 (x_1, y_1, x_2, y_2) を指定することでサイズ変更，移動が可能となる．座標の単位はピクセルで表され，絶対座標であるため，この座標によりどのディスプレイにどれだけの画面割合になるかがわかる．

b) ネットワーク帯域情報集収集機能

先行研究にて開発したトポロジ管理機能は，５秒毎にネットワークトポロジの情報を収集している．収集する情報は OpenFlow スイッチで構成された IP アドレス，レイテンシなどの情報に加え，トラフィック量を収集するよう変更を加えた．

c) 帯域情報

b) によって収集した情報は，グラフとして保存する．そこに重みづけとして，ネットワーク帯域からトラフィック量を引いた空いている帯域を算出している．３点の IP アドレス間の取得した重みつきグラフの例を図 8 に示す．グラフは有向グラフで示しており，空いている帯域を Byte 単位で表している．トラフィック量については両方向のトラフィックの総量としているため，どちらの向きも同じ値が設定されている．

d) 帯域情報における最短経路探索機能

a) のウィンドウリサイズ通知機能から通知をもらうと，c) の帯域情報を元に最短経路を算出し，最も空いている経路への経路変更を行う．

OpenFlow スイッチを制御するための OpenFlow コントローラソフトウェアはいくつかある．その多くはフレームワークとして提供され，開発者は OpenFlow コントローラの制御部分のみのソースコードを書くだけで，OpenFlow コントローラとして動作することができる．その 1 つが Trema[6] である．Trema は主に Ruby 言語で開発され，開発者は Ruby 言語，もしくは C 言語，C++で制御部分を記述できる．Trema は開発用のフレームワークであるためテスト用に仮想 OpenFlow スイッチ環境も用意している．今回の実装ではこれらを利用し開発を行った．

4. まとめと今後の課題

本研究では，SAGE によるタイルドディスプレイ上で，ユーザインタラクションによって変動するネットワークパラメータと連動させ，SDN を応用したネットワーク制御技術により，ネットワークを効率的に利用した大規模可視化を実現することを目的として研究開発を進めてきた．本稿ではユーザのウィンドウ操作に伴うストリーミング通信のネットワーク流量の急激な変化に追従し，ネットワーク輻輳を避けるために最も空き帯域が確保できるネットワーク経路を通るよう経路の探索と切り替えを実現する仕組みについて提案した．

今回の提案はプロトタイプ実装であり，OpenFlow スイッチは Trema 附属の仮想 OpenFlow スイッチを用いているため，今後は実機での計測が必要となる．また，ネットワーク経路探索のポリシとして，今回は最も空きのある帯域を優先するという単純な手法を取っている．しかし，ネットワークの利用効率向上の観点からは，各ストリーミング通信の通信量を正確に見積もり，利用可能帯域に無駄なく詰めていくような高度な経路決定方式の方がより適切な場合もあると考えられる．またネットワーク切り替え時においても，ネットワーク通信の継続性を考慮せず，コントローラから各スイッチに一斉に切り替えの制御を行なっているが，切り替えのタイミングで多少のパケットロスなどを生じさせている可能性がある．今後はこのような状況も計測・考察し，更なる高度なネットワーク制御の実現を目指す．

謝辞 本研究は，独立行政法人情報通信研究機構 (NICT) の委託研究「仮想分散コンピューティング・データ流通技術の研究」の支援に基づき推進している．

参考文献

[1] Fujitani, H., Tanida, Y. and Matsuura, A.: Massively Parallel Computation of Absolute Binding Free Energy with Well-Equilibrated States, *Phys. Rev. E*, 79, 021914, (2009).

[2] Watanabe H., Tanaka, S., Okimoto, N., Hasegawa, A., Taiji, M., Tanida, Y., Mitsui, T., Katsuyama, M. and Fujitani, H.: Comparison of binding affinity evaluations for FKBP ligands with state-of-the-art computational methods: FMO, QM/MM, MM-PB/SA and MP-CAFEE approaches, *Chem-Bio Informatics Journal*, Vol. 10, pp.32–45, (2010).

[3] Leigh, J., Johnson, A., Renambot, L., Peterka, T., Jeong, B., Sandin, D., Talandis, J., Jagodic, R., Nam, S., Hur, H. and Sun, Y.: Scalable Resolution Display Walls, *Proc. of the IEEE*, Vol. 101, Issue 1, pp.115–129, (2013).

[4] The Open Flow Switch Specification, Version 1.1.0, http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, (2011).

[5] Furuichi, T., Date, S., Yamanaka, H., Ichikawa, K., Abe, H., Takemura, H. and Kawai, E.: A Prototype of Network Failure Avoidance Functionality for SAGE using OpenFlow, *Proc. of IEEE 36th International Conference on Computer Software and Applications Workshops*, pp.88–93, (2012).

[6] Trema, http://trema.github.io/trema/