Vol.2014-IOT-26 No.11
2014/6/28

HTML5 技術を利用した授業や会議向けデスクトップ画面実時間配信システムとその管理システムの試作

山之上卓[†1]　小荒田裕理[†1]　小田謙太郎[†1]　下園幸一[†1]

HTML5 技術を利用してデスクトップ画面を、実時間で、数十台の端末に配信するシステムと、その管理システムを試作したことについて述べる。インターネットとプライベートネットワークのどちらにもサーバを配置することにより、授業や会議が遠隔地で分散して実施される場合にも対応できる。大量の端末に効率よくデータを配信するため、複数のサーバを利用するが、Web クライアントを自動的に適切なサーバに割り当てる機能も持っている。負荷分散機能も持っている。サーバを管理するため、Web 画面上でサーバを制御することができる。管理者が適切にサーバを加えたり減らしたりするため、端末数、Web クライアントで表示される単位時間あたりの表示画面枚数、Web クライアントにおけるネットワーク利用バンド幅などの変化も表示可能で、ログも採取できる。

Experimental Implementation of a Real-time PC Screen Distribution System for Classes and Meetings using HTML5 Technology

TAKASHI YAMANOUE[†1]　YUURI KOARATA[†1]　TAIKI KATAGIRI[†1]
KENTARO ODA[†1]　KOICHI SHIMOZONO[†1]

Experimental implementation of a real-time PC screen distribution system for classes and meetings is discussed. This system uses HTML5 technology. So users of this system can use this system just using their own common Web browsers. Several tens web clients can share the screen of a PC. This system is a kind of CDN which unifies servers at the Internet and hierarchical private networks. An appropriate server of the CDN is selected automatically when a Web client is connected to the CDN. This system is also equipped with administration functions for managers of this system.

1. はじめに

我々は授業やゼミや会議のとき，パソコン画面をプロジェクタでスクリーンに投影することをよく行う．しかしながら，この方法では，遠くから小さな字や絵を見ることが困難になる．このような問題に対処するため，SOLAR-CATS[1][5] や Multi VNC[2] などの PC 間画像放送システムが存在する．しかしながら，これらのシステムは，会議参加者それぞれに，これらのソフトウェアをインストールしたパソコンが必要になる．

最近，我々の多くはスマートフォンやノートパソコンを携帯している．もし，発表者のパソコンのデスクトップ画像を，このような機器で，特別なソフトウェアやアプリをインストールすることなしに，共有することができたら，授業や会議やゼミは，より効果的になる可能性がある．我々は，このような要求を満たす「Web Screen Share」[6]を開発し，我々のゼミで利用している．Web Screen Share は，HTML5 の Web Socket 技術を使ったパソコンデスクトップ画面共有システムである．これは，一般的なパソコンやスマートフォンで利用可能な，複数の，HTML5 対応 Web ブラウザに，画像中継 Web サーバを通じて，送信元パソコンのデスクトップ画面をリアルタイムで配信する．

Web Screen Share は，参加者が２０名程度までなら，問題なく動作していた．しかしながら，この人数を超えると，動画を表示するときに，動きの滑らかさが失われてしまうことが分かった．

我々は，この問題を，平衡２分木上に接続されたサーバ間データ転送ノードを使うことによって解消した．送信元パソコンのデスクトップ画像は，このサーバ間データ転送ノードを使って複数の画像中継 Web サーバに配送される．Web ブラウザは，複数の画像中継 Web サーバに，自動的に，均等に，接続される．問題を解消した Web Screen Share を，Distributed Web Screen Share (DWSS)と呼ぶ. DWSS は, Web ブラウザの接続を，トーナメントアルゴリズムを使って自動的に，均等に，複数の Web サーバに割り当てる機構も備えている. Distributed Web Screen Share によって，動画表示の動きの滑らかさを損なうことなく，より多くの Web クライアントによって同じデスクトップ画面を共有できた．

今回, Distributed Web Screen Share のサーバをインターネットと階層的なプライベートネットワークの各所に配置した，一種の CDN の仕組みを考案し，試験的に実装を行ったことについて述べる．この仕組みにより，インターネットと階層的なプライベートネットワークのどちらにもサー

†1 鹿児島大学
Kagoshima University

バを配置することが可能になり，授業や会議が遠隔地で分散して実施される場合にも対応できる．大量の端末に効率よくデータを配信するために複数のサーバを利用するが，Web クライアントを自動的に適切なサーバに割り当てる機能も持っている．このシステムの管理者は，サーバを管理するため，Web 画面上でサーバを制御することができる．管理者が適切にサーバを加えたり減らしたりするため，端末数，Web クライアントで表示される単位時間あたりの表示画面枚数，Web クライアントにおけるネットワーク利用バンド幅などの変化も表示可能で，ログも採取できる．このシステムのことを CDN for Web Screen Share, CDN4WSS と呼ぶ．

2. System 概要

図 1 に CDN4WSS の例，図 2 に CDN4WSS の利用例を示す．本システムは，Web クライアントに PC 画面を配信するサーバ-クライアントシステム(*Web Screen Share*, WSS)を 2 分木状に接続したもの(*Distributed Web Screen Share*, DWSS)が、インターネットと階層的プライベートネットワークに分散配置されたものである。各ネットワークに配置される DWSS は木状に接続されていて，その根はインターネット上に配置される．

Web Screen Share (WSS) [6]は Client-Server 型の，HTML5 の Web Socket 技術を使った，パソコンデスクトップ画像を実時間で共有システムである．図 3 に Web Screen Share の概要を示す．Web Screen Share は 1 台の Web Server と，複数の Web Client と，画像取得とその画面をサーバに送信する Screen Sender から構成されている．Screen Sender はデスクトップ画像を繰り返し取得し，通常の Socket を使って Web Server に送信する．Screen Sender から画像を受け取った Web Server は，その画像を各 Web Client に対応した Queue の最後に追加する．Web Client は HTML5 の Web Socket を使って Web Server に対して get コマンドを繰り返し発行する．Web Server が get コマンドを受け取ると，それぞれの Web Client に対応した Queue の先頭にある画像を，get コマンドを受け取った Socket に対して送信し，その画像を Queue から削除する．画像を受け取った Web Client はその画像を表示する．

Web Screen Share に接続する台数により，Web Server の CPU 使用率と通信帯域がどのように変化するかを計測した結果，Web Client 数が一定数を超えるとネットワークの利用は 100% に近くなり，これ以上 Web クライアント数が増えると Web Client に送信される単位時間あたりの情報量が減少することがわかった．CPU 利用率もクライアント数に比例して増加し，例えネットワーク容量が十分あったとしても，Web クライアント数が一定数を超えると，サーバ側の処理能力が低下し，Web Client での動画表示性能が劣化することが予測される．この一定数を超えて，大規模な授業や会議にも対応できるようにするため，数百台の Web クライアントが画面共有できるようにしたい．Web クライアント増加による性能劣化を緩和するため，我々は Distributed Web Screen Share (DWSS)を開発した．

DWSS は WSS の Web サーバにサーバ間接続サーバ(Inter Server Connection Node, ISC ノード) を組み合わせた Node System を，平衡 2 分木状に接続し，葉の位置にある Node system の Web Server に Web Client をできるだけ均等に割り当てるようにするものである(図 4). Node System は Web Server と ノード間を接続するための Inter Server Connection Node (ISC Node) を接続したものである．Screen Sender は DWSS のどれか 1 つの Web Server に接続され，Screen Sender で取得された画像はこの Web Server を経由して，すべての Node System の Web Server に転送される．画像のすべてまたは 1 部が Queue を通じて Web Client に送信され，そこで表示される．

平衡 2 分木のすべての ISC Node に画像を中継するため，ISCN はすべての Node System に配信される画像データや制御データを受け取ったとき，そのデータを，そのデータを受け取った通信経路以外の，すべての通信経路に転送する．ノード間は通信経路によって木状に接続されているため，ループは発生しない．

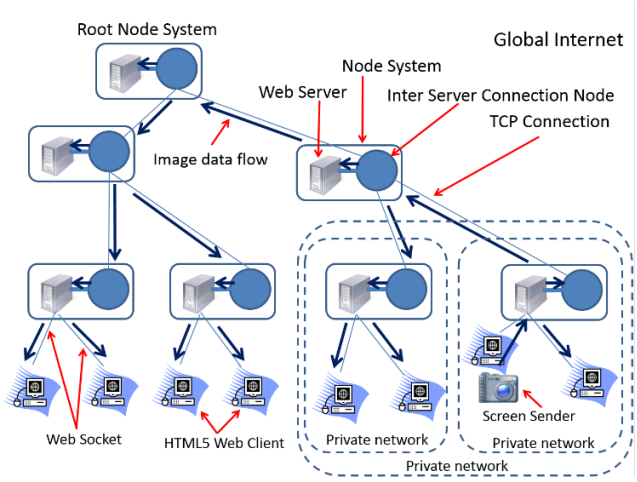

図 1. CDN4WSS の例

図 2. CDN4WSS の利用例

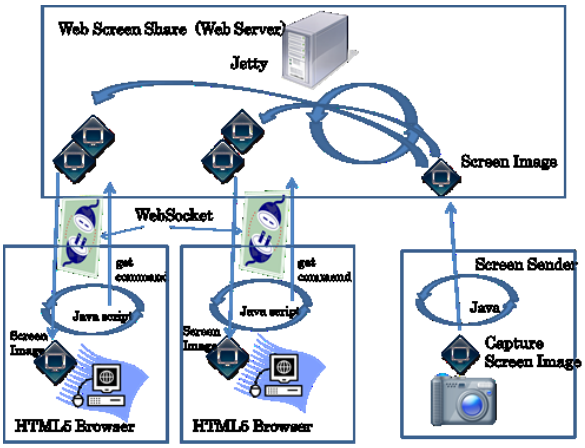

図 3. Web Screen Share

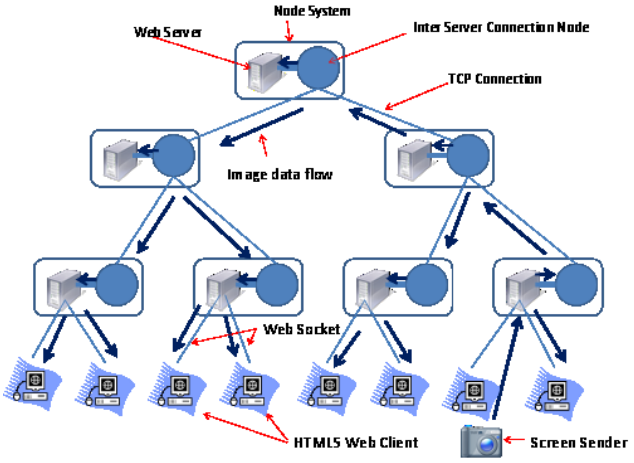

図 4. Distributed Web Screen Share

3. 負荷分散機能とプライベートネットワークをまたぐ CDN 機能

3.1 ISC ノードによる平衡２分木の構成

ノード間で配送遅延の差ができるだけ少なくなるように, DWSS では Node System を平衡２分木状に自動的に接続する機構を持っている. これは文献[7]のアルゴリズムを利用している. 2 分木を構成する ISC ノードの集合を「グループ」とする。このアルゴリズムは２分木の任意の ISC ノードにおいて, 一定期間ごとに左右の子孫のノード数の合計値を左右の子孫から受け取り, それぞれ値を left weight, right weight として記憶し, 自分が根ノードでない場合はその合計値を親ノードに送信する. このときの通信は ISC ノードの生き死に確認にも使われる. グループに新たに ISC ノードが加わるとき, 新たなノードはグループの根ノードに対して, 「参加の問い合わせ」を行う. 任意のノードに新たなノードの参加が問い合わされたとき, 左右の子孫の数を比較して, 再帰的にノード数が少ない側の子ノードに参加の問い合わせが行われ, 左右のどちらかの子ノードを持っていないノードにおいて, 新規参加ノードがそのノードの子ノードとして接続される. 左右のバランスが崩れている部分があっても, 新しいノードが次々と参加するとき, 根ノードから再帰的なノード参加処理が行われることにより, 左右のバランスの改善が行われる..

3.2 トーナメントアルゴリズムによる負荷分散

Web Client の接続先により性能の差ができるだけ生じないように, Web Client を Node System の葉ノードにある Web Server にできるだけ均等に割り当てるようにして負荷分散を行う. これを実現するため, トーナメントアルゴリズムを利用する.このアルゴリズムは, 葉ノードの Node System においては, その Web Server に接続された Web Client 数を, その Web Server の URL と共に, 親ノードに送信することを一定期間ごとに繰り返す. 任意の節ノードにおいては, 左右の子ノードから送られてきた Web クライアント数を比較し, Web Client 数が少ないほうの URL を勝者として記録し, その Web Client 数を Web Server の URL と共に親ノードに送信することを一定期間ごとに行う. 根ノードにおける勝者が全 Web Server のうち, 最も Web Client 数が少ない Web Server の URL となる. 新規に Web Client が DWSS に接続するとき, その Web Client はまず, 根の Node system に接続を行い, 次に, 根の Node system に記述されている勝者の URL に re-direct されることにより, 最も Web Client の少ない Web Server に新規の Web Client が接続する.

3.3 階層的なネットワークにまたがった CDN

CDN4WSS は, 複数の, 階層的に構成されたネットワークに跨ったサーバ群の中で, 新たに接続された Web クライアントに適切なサーバを割り当てる機能を持っている. この機能を説明するため, 階層的なネットワークのモデルと階層的な CDN のモデルを定義し, 適切なサーバ割り当てアルゴリズムとその証明を与える.

3.3.1 階層的ネットワークのモデル

準備

まず以下を定義する

H はホストの有限集合を示す.

述語 *connectable* は 1 台のホストが TCP によって別のホストに接続可能であることを示す.

$$if\ connectable(h_i, h_j),\ for\ h_i, h_j \in H\ then$$
the host h_i can connect to the host h_j.

connectable. に以下の規則を与える.

$$\text{if } connectable(h_i, h_j) \text{ and } connectable(h_j, h_k)$$
$$\text{then } connectable(h_i, h_k)$$

この規則はすべてのネットワークのホストに当てはまるものではないが, 階層的なプライベートネットワークの多くに当てはまるものである.

connectable を拡張してホストの集合に対する *Connectable* と $\neg Connectable$ を定義する

$$if\ \ Connectable(H_i, H_j)\ then$$
$$\forall h_k \in H_l, \forall h_j \in H_j, connectable(h_k, h_l),.$$

$if\ \neg Cneonctable(H_i, H_j)$, *then*

$\forall h_k \in H_i, \neg(\exists h_l \in H_j, connectable(h_k, h_l))$

階層的プライベートネットワークにおいては，ある 1 台のホストが別のホストに接続することは可能でも，その逆は不可能な場合がある．これを示す述語 *halfConnectable* と $HalfConnectable$ を与える．

if $halfConnectable(h_i, h_j)$ *then*

$connectable(h_i, h_j)$ *and* $\neg connectable(h_j, h_i)$

if $HalfConnectable(H_i, H_j)$, *then*

$Connectable(H_i, H_j)$ *and* $\neg Connectable(H_j, H_i)$

これに対し述語 *fullConnectable* と$FullConnectable$はホスト間で相互に接続可能であることを示す．

if $fullConnectable(h_i, h_j)$ *then*

$connectable(h_i, h_j)$ *and* $connectable(h_j, h_i)$

if $FullConnectable(H_i, H_j)$ *then*

$Connectable(H_i, H_j)$ *and* $Connectable(H_j, H_i)$

connectable と同様に *Connectable*, *halfConnectable*, *HalfConnectable*, *fullConnectable* and *FullConnectable*. について以下の規則を与える．

if $Connectable(H_i, H_j)$ and $Connectable(H_j, H_k)$
then $Connectable(H_i, H_k)$

if $halfConnectable(h_i, h_j)$ and $halfConnectable(h_j, h_k)$
then $halfConnectable(h_i, h_k)$

if $HalfConnectable(H_i, H_j)$ and $HalfConnectable(H_j, H_k)$
then $HalfConnectable(H_i, H_k)$

if $fullConnectable(h_i, h_j)$ and $fullConnectable(h_j, h_k)$
then $fullConnectable(h_i, h_k)$

if $FullConnectable(H_i, H_j)$ and $FullConnectable(H_j, H_k)$
then $FullConnectable(H_i, H_k)$

定義

階層的なネットワークのモデル M はホストの集合を節とする木である．

階層的ネットワーク $M = (N, C)$ は以下で構成される．

(i) 有限集合

$$N = \{H_1, H_2, \cdots, H_n\}$$

ここで要素はホストの集合であり，

$$\forall i \in \{1 \ldots n\}, H_i \subset H$$

(ii) $N \times N$ のデカルト積 C の部分集合

$C = \{(H_i, H_j) |$
$H_i \neq H_g$, and
HalfConnectable(H_i, H_j), and
$if\ H_j \neq H_k$ then $\neg\exists H_k, (H_i, H_k)$, and
$\forall H_i \in N - H_g,\ \exists H_1, H_2, \cdots, H_n$,
$(H_i, H_1), (H_1, H_2), \cdots,$
$(H_{n-1}, H_n), (H_n, H_g) \in C$ $\}$

N について以下の規則を与える．

$\forall i, j \in \{1 \ldots n\}$, if $i \neq j$ then $H_i \cap H_j = \phi$

$\forall h_l, h_m \in H_i \in N$, $fullConnectable(h_l, h_m)$

$H_g \in$ N はグローバルホストの集合であり，木の根である．(H_x, H_y) はH_x のネットワークがH_y のネットワークに，NAPT によって直接接続されていることを表す．アルゴリズムを簡単にするため，ここでは NAT の存在は無視する．$if\ H_j \neq H_k$ then $\neg\exists H_k, (H_i, H_k)$, は M が木または複数の木であることを表す．式

$$\forall H_i \in N - H_g, \quad \exists H_1, H_2, \cdots, H_n\ ,$$
$$(H_i, H_1), (H_1, H_2), \cdots,$$
$$(H_{n-1}, H_n), (H_n, H_g) \in C$$

は M が H_g を根とする 1 つの木であることと M の任意のホストがグローバルホストに接続可能であることを示す．

アドレスとホスト

集合 *A={IP addresses}* IP アドレスの集合を表す．関数 *address* は以下の写像である．

address: $H \rightarrow A$

関数 *address* に以下の規則を与える．

$$\forall h_l, h_m \in H_i \in N,$$
$$address(h_i) = address(h_j) \Leftrightarrow h_i = h_j$$

$h_{ij} \in H_j, for\ each\ (H_i, H_j) \in C$ であるようなホスト h_{ij} は H_i と H_j の間の NAPT を表す．2 つのネットワークに跨る NAPT は物理的には 1 台のホストであるが，ここでは 2 つのネットワークに設置された 2 台のホストとして扱う．

関数 *addressAt* の値を以下のように定める．

addressAt$(h_i, H_j) =$
address(h_i),
if $h_i \in H_j$
address(h_{ij}),
if $(H_i, H_j) \in C$ and $h_i \in H_i$
or $h_i \in H_i$ *and* $Connectable(H_i, H_k)$
and $(H_k, H_j) \in C$
Φ, if $\neg$connectable$(h_i, h_j), h_j \in H_j$

この関数はH_jの子孫が，あるプライベートネットワークのホストを含む場合，H_jから見たそのホストのアドレスはH_jの NAPT のアドレスになることを示す．

関数 *host* の値を以下のように定義する．

$host$(a, H_i) =
h_x,
$if\ h_x \in H_i\ and\ a = addressAt(h_x, H_i)$,
$host(a, H_j)$, $where\ (H_i, H_j) \in C$
$if\ \neg\exists h_x \in H_i, a = addressAt(h_x, H_i)$
ϕ,
if H_i=H_g and $\neg\exists h_x \in H_g, a = addressAt(h_x, H_g)$

この関数はH_iのホストから見たときにアドレス a を持つホストを見つける．

3.3.2 CDN4WSS のモデル

準備

最初に以下の用語を定義する．述語 connected はあるホストが別のホストに TCP で接続されていることを示す．

connected:$H \times H \longrightarrow \{\text{true}, \text{false}\}$

もし $\text{connected}(h_i, h_j)$ ならば h_i と h_j お互いに直接データを交換することができる. 従って if $\text{connected}(h_i, h_j)$ ならば $\text{connected}(h_j, h_i)$ である.

述語 *communicable* は 2 台のホストがお互いにデータを交換できることを示す

communicable:$H \times H \longrightarrow \{\text{true}, \text{false}\}$

If $communicable(h_i, h_j)$ then

$connected(h_i, h_j)$

or

$\exists h_1, h_2, \cdots, h_n$ such that

$\text{connected}(h_i, h_1), \text{connected}(h_1, h_2), \cdots,$

$\text{connected}(h_{n-1}, h_n), \text{connected}(h_n, h_j)$

述語 *communicable* は

if $\text{connected}(h_i, h_j)$ then $\text{connected}(h_j, h_i)$ であり,

if $\text{communicable}(h_i, h_j)$ then $\text{communicable}(h_j, h_i)$ なのでなのでは交換則が成り立つ.

定義

CDN4WSS のモデル(CDN)は階層的なネットワーク M の中にある木である. この木のノードはサーバの集合である.

CDN $CDN = (CDNS(M), D)$ は以下で構成される.

(i) サーバの集合を要素にもつ有限集合

$$CDNS(M) = \{S_1, S_2, \cdots, S_p\}$$

ここで, $\forall S_i \in CDNS(M), \exists H_j,\ S_i \subseteq H_j \in N$ of M

それぞれの $S_x \in CDNS(M)$ は root server $s_r \in S_x$を持つ. $s_{gr} \in S_g \in H_g$, $S_g \in CDNS(M)$ は global root server である。S_g を global server と呼ぶ.

(ii)デカルト積 $CDNS(M) \times CDNS$(M) の部分集合 D

$D = \{(S_i, S_j) |$

$S_i \neq S_g$, and

HalfConnectable(S_i, S_j), and

$if\ S_j \neq S_k$ then $\neg \exists S_k, (S_i, S_k)$ and

$\forall S_i \in CDNS(M) - S_g,\ \exists S_1, S_2, \cdots, S_n$

$(S_i, S_1), (S_1, S_2), \cdots, (S_{n-1}, S_n), (S_n, S_g) \in D$ $\}$

以下の性質を CDNS に与える.

- $\forall i, j \in \{1 \ldots p\}$, if $i \neq j$ then $S_i \cap S_j = \phi$
- if $S_x \in CDNS(M)\ then\ \exists H_y\ S_x \subseteq H_y \in N$ of M
- if i ≠ j then $\neg \exists H_x$ $such\ that\ S_i \in H_x\ and\ S_j \in H_x, H_x \in N\ of\ M$
- $\forall h_i, h_j \in S_x \in CDNS(M)$,

  communicable(h_i, h_j)

  $if\ (S_x, S_y) \in D$ then $S_x\ is\ directly\ connected\ to\ S_y$

  $and\ \exists s_{xy} \in S_y\ such\ that$

  $\text{connected}(s_{xr}, s_{xy})$ $(= communicable(s_{xr}, s_{xy}))$

  $where\ s_{xr}$ is the root server of S_x.

s_{xy} を S_x S_y.の間の *relay server* と呼ぶ. ある 1 台のサーバはS_yの複数の子孫の relay sever の場合がある.

ここで, CDN のサーバは multi homed ではないと仮定している. あるサーバが同じネットワークの別のサーバから接続された場合, 観測されたアドレスと元のアドレスは同じになる. あるグローバル側にあるサーバが NAPT の向こう側にあるサーバから接続された場合, グローバル側のサーバが観測したアドレスと元のアドレスは, RFC 1918 (RFC 4193 is used for the IPv6 network.) により異なるネットワークアドレスがつかわれるため, 異なるアドレスになる. 従って, グローバル側のネットワークにあるサーバは, そのサーバに接続した別のサーバが同じネットワークのものか, NAPT の向こう側のものか判別することができる. relay server s_{xy} には, 以下の値を与える.

$\{\ \text{addressAt}(s_{xr}, S_x) | (S_x, S_y) \in D, s_{xr} \in S_x\}$

性質

CDN は以下の性質を持つ.

- 性質 1

  $\forall s_i \in S_i \in CDNS - S_g,\ communicable(s_i, s_g)$

  証明:

  すべての

  $\text{communicable}(s_{ir}, s_{i1}),$

  $\text{communicable}(s_{15}, s_{12}),$

  $\cdots$

  $\text{communicable}(s_{(n-1)r}, s_{ng}),$

  where $s_{ir} \in S_i, s_{i1} \in S_1, s_{1r} \in S_1, s_{12} \in S_2, \cdots, s_{ng} \in S_g$

  は真である. なぜなら,

  D の定義により

  $(S_i, S_1), (S_1, S_2), \cdots, (S_{n-1}, S_n), (S_n, S_g) \in D$

  であり,

  $s_{(l-2)(l-1)}$, $s_{(l-1)r} \in S_{(l-1)}$ なので *CDNS(M)* の定義により,

  $\text{communicable}(s_{(l-2)(l-1)}, s_{(l-1)r})$ は真となる.

  従って, すべての

  $\text{communicable}(s_{ir}, s_{i1}),$

  $\text{communicable}(s_{i1}, s_{15}),$

  $\text{communicable}(s_{15}, s_{12}),$

  $\text{communicable}(s_{12}, s_{25}),$

  $\cdots$

  $\text{communicable}(s_{(n-2)(n-1)}, s_{(n-1)r}),$

  $\text{communicable}(s_{(n-1)r}, s_{ng}),$

  $\text{communicable}(s_{ng}, s_g),$

  は真である. これは

  $\forall s_i \in S_i \in CDNS - S_g,\ communicable(s_i, s_g)$

  が真であることを表す.

- 性質 2

  $\forall s_x \in S_x \in CDNS\ and\ \forall s_y \in S_y \in CDNS,$

  $\text{communicable}(s_x, s_y)$

  証明:

  CDNS の定義により, $s_x, s_y \in S_z \in CDNS$ なら, $\text{communicable}(s_x, s_y)$ である.

Communicable の定義により,
If $s_x \in S_u$ *and* $s_y \in S_v$ and $S_u \neq S_v$ then
communicable(s_x, s_g) and communicable(s_y, s_g)
If communicable(s_y, s_g) then
communicable(s_g, s_y)
なので,
communicable(s_x, s_g) communicable(s_g, s_y)は真であり, この結果, communicable(s_x, s_y) は真である.
性質 2 は CDN の中の任意の２台のサーバはその間で相互にデータを送受信できることを表す. 従って, CDN 内のすべてのサーバは, CDN 内のあるサーバで得られたデータをすべてのサーバで共有できることを表す.

3.3.1 適切なサーバ割り当てアルゴリズム

TCP 通信のバンド幅は遅延に大きく依存する. クライアントが CDN の中のサーバの１つに接続されるとき, クライアントとサーバ間の遅延はできる限り短い方が良い. 遅延の目安としてホスト間の距離を考える. クライアントとサーバ間の距離として, 階層的ネットワーク M を表す木の上における, クライアントを含むノードサーバを含むノード間の経路の長さを使用することにする.

クライアントに「適切」なサーバとは, 階層的ネットワークの木において, そのクライアントを含むノードから最も近いノードの中にあるサーバであるとする.

我々は一般的な Web ブラウザを CDN のクライアントとして利用したい. 幸いにして, 最近の Web ブラウザは HTML5 が利用でき, HTML5 は以下を可能にする.

- プログラムを実行できる.
- WebSocket を使ってサーバと通信できる.
- 他のサーバにリダイレクトできる..

しかしながら以下は不可能である.

- broadcast や multicast の利用. (これが利用できればクライアントは同じネットワークに存在するサーバを容易に見つけることができる)
- 接続されたサーバ以外のホストとの通信.
- 自分自身の IP アドレスの認識.

上の制限の他に, 以下を仮定する.

- クライアントはグローバルの root server 以外のサーバがどこにあるか知ることはできない.

この制限は CDN を利用者にとって使いやすいものにする. CDN を構成する木の各ノードのすべての根ノードは以下の関数を持つと仮定する.

privateRootServers(s∈ S)=
{addressAt(s_{xr}, S_x)|
$connected(s_{xr}, s_y)$ *where* $s \in S_y, s_y \in S_y$
$s_{xr} \in S_x$, *and* $(s_x, s_y) \in D$}

defaultServer(s)= one of s_y∈S_(y) such that s∈S_y

Web client では以下の手続きが実行される. この手続きは Web client に「適切」なサーバを割り当て, Web client はこのサーバに接続する.

Procedure connectToAnAppropriateServer(a∈A)
begin
connect to $host(a, H)$ temporally;
privates←*privateRootServers*($host(a, H)$);
default← $defaultServer(host(a, H))$;
if *privates*≠ ϕ then
for each $s_{ri} \in privates$
if *connectable*$(this, s_{ri})$ then
connectToAnAppropriateServer(s_{ri}) ;
connect to *default* as a client;
end;

ここで H はこのクライアントを含む階層的なネットワークを表す木のノードである. 実際のプログラムは H を明示的に使う必要はない. *connectToAnAppropriateServer*(h_w) は redirect を行うものとする. 従って, この部分を実行した後に, それに続く “connect to default as a client” は実行されない.

先に述べたとおり, 利用者はグローバルの root server s_{gr} のアドレスを知っているものと仮定している. s_{gr}はグローバルネットワーク上にあるのですべてのクライアントから接続できる. クライアントが階層的ネットワーク M を表す木のどれかのノードに含まれるホストで, s_{gr} が CDN の global root である場合, *connectToAnAppropriateServer*(s_{gr}) がそのクライアントで実行されると, このクライアントは CDN の「適切」なサーバに接続される.

証明:

クライアントに一時的に接続されているサーバが s_t であるとき, それを含む node $S_t \in CDNS$ を traversing node と呼ぶことにする. この手続きは root node から CDN の木を辿る.

クライアントh_c について, もし $h_c \in H_c$ *and* $S_t \subseteq H_c, H_c \in N$ (クライアントと traversing node は階層的ネットワークの木の同じノードにある), かつ $privateRootServers(s_t \in S_t) =$ *privates* = ϕ, を仮定すると client h_c は $default = defaultServer(s_t) \in S_t \subseteq H_c$*and* $h_c \in H_c$. に接続される. client h_c を含むノードと, *default* server を含むノードは同じであるため, ノード間の距離は 0 である. 従って, *default* server は適切な server である.

もし $h_c \in H_c, S_t \subseteq H_c, H_c \in N$ (クライアントと traversing node は階層的ネットワークの木の同じノードにある), で CDN のノードが子孫を持つ場合, $privates \neq \phi$ であり, client はそのすべてへの接続を試すが階層的ネットワークの定義である以下の理由で失敗する.

$\forall s \in privates, halfConnectable(s, h_c) \supset$,
$\neg halfConnectable(h_c, s)$

従って, client は$default \in S_t \subseteq H_c \; and \; h_c \in H_c$. に接続する.

この手続きが node S_t を traverse していて, $privateRootServers(s \in S_t) = \{s_{r1} \cdots s_{rn}\} \neq \phi$, かつ $\exists s_{ri} \in \{s_{r1} \cdots s_{rn}\}, s_{ri} \in S_x \subseteq H_x, Connectable(H_c, H_x)$, ならば、$connectToAppropriateServer(s_{ri})$ が実行される. この手続きは再帰的に, traversing node よりも client に近い root node に traverse しようとする.

なお, もし $\exists s_{ri} \in \{s_{r1} \cdots s_{rn}\}$, $s_{ri} \in S_x \subseteq H_x$, $Connectable(H_c, H_x)$なら, $\neg \exists s_x \in \{s_{r1}, \cdots, s_{rn}\}, s_x \neq s_{ri}, s_x \in S_x \subseteq H_x$, and $Connectable(H_c, H_x)$ (s_{ri} はh_c から接続できるただ一つのサーバ)である. なぜなら, CDN は木で, h_c はS_xから見て葉の側にあり, H_c と H_x の間の他のパスは, s_{ri} のノードを含むみ, $s_t \in S$ から $s_x (\neq s_{ri})$ に接続することはできない. 従って, $connectToAppropriateServer(s_{ri})$ が実行された場合, その先で必ずh_c により近いサーバが見つかり, $s_x \neq s_{ri}, s_x \in S_x$ のような s_x について $connectToAppropriateServer(s_x)$ を実行しても h_c により近いサーバが見つかることはない. また, もし$\neg \exists s_{ri} \in \{s_{r1} \cdots s_{rn}\}$, $s_{ri} \in S_x \subseteq H_x$, $Connectable(H_c, H_x)$ なら S_x の子孫はh_cから接続不可能であり, 従ってS_x がh_cから接続可能なサーバを持つ最もh_c に近いノードとなる. 従って, クライアントは, そこから接続可能な最も近い距離にある「適切」なサーバに接続される.

このアルゴリズムを試験的に実装した例を図 5, 図 6, 図 7 に示す. 図 5 の左側は global root server 起動後の GUI であり, 右側は private network で立ち上げた後, global root server に接続した server である. 図 6 のように, Web client で global root server に接続すると, 図 7 のように, 自動的に private network のサーバに redirect される.

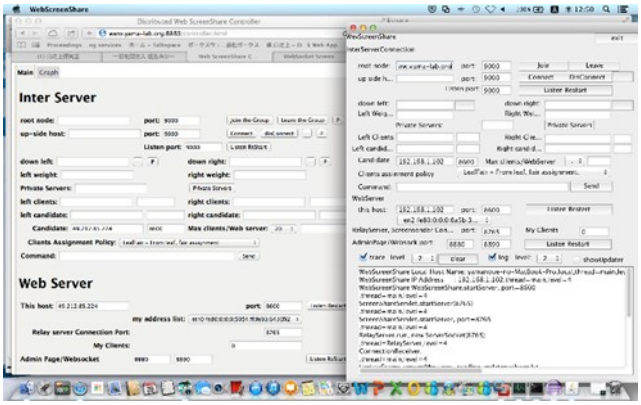

図 5. Global root server に接続した private network の server

図 6. Global server に接続しようとした Web client

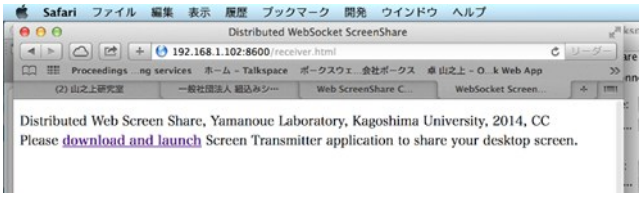

図 7. Private network の server へ redirect

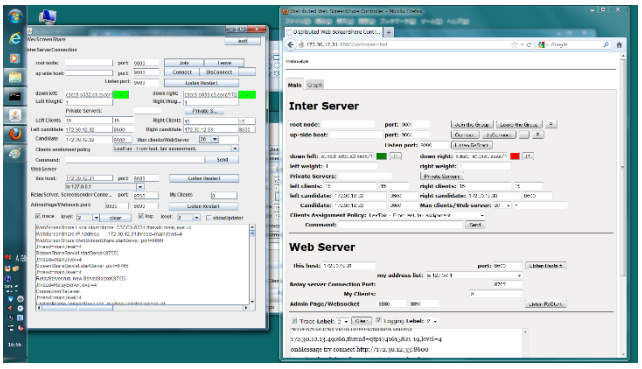

図 8. Node System の GUI(左)と WebGUI(右)

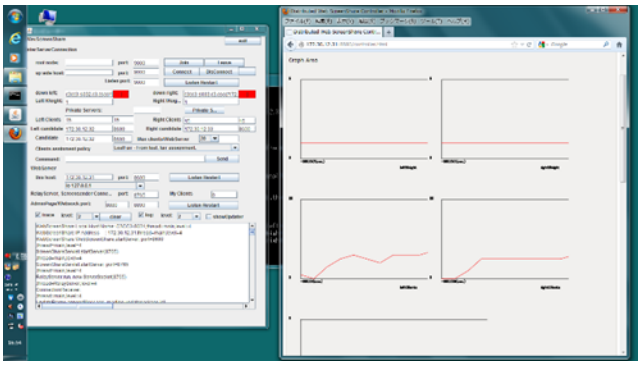

図 9. Web GUI に表示された左右の Node System 数と接続されたクライアント数の変化

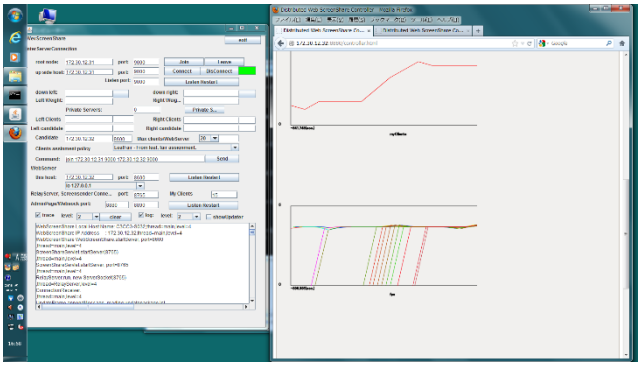

図 10. 葉ノードの GUI(左)とその Web Gui に表示されたノードに直接接続されたクライアント数の変化とそれぞれのクライアントの fps の変化

4. 管理機能

DWSS は Web Client の増加による性能低下を緩和するために, 1 台の Web Server に接続可能な Web Client 数の上限を設定し, あふれた Web Client を節ノードに配置する機能や, 動的に Node System を追加する機能を持っている. また, Web Client 数が減少している状況において, 葉の Node System に接続される Web Client 追加を停止することにより, 葉の Node System の方から, これに接続される Web Client 数を 0 にしていき, 葉の Node System を削除可能にする機能も持っている. 葉ノードの方から空にすることにより, Node System の削除のノード再配置を省くことが可能となり, DWSS に参加している Web Client への影響を小さくすることができる. 管理のための Web の GUI を備えており, Web ブラウザで管理機能を操作できる. Web の GUI の該当する部分をクリックすることにより, サーバ間を移動できる. 端末数, Web クライアントで表示される単位時間あたりの表示画面枚数, Web クライアントにおけるネットワーク利用バンド幅などの変化も表示可能である(図 8,9,10). ログも採取できる.

5. 関連研究

5.1 SOLAR-CATS

我々は SOLAR-CATS[1][5]を開発している. これは気軽に利用できる教育支援システムであり, デスクトップ画面を共有する機能も持っている.しかしながら, SOLAR – CATS は一般的な Web ブラウザで利用することができない.

5.2 拡張 Edutab

鈴木らは HTML5 技術を使った電子薄板の共有を行う遠隔教育支援システムを開発している[8]. このシステムは 1 台のサーバを利用しているため, 接続されるクライアント数が増加すると描画スピードなどの性能が劣化する恐れがある.

5.3 AKAMAI と FCAN

Akamai[10] は有名な商用 CDN である. FCAN[4] は proxy server を組織化し, 予測できない急な Client の増加(flash crows)が発生した場合, Client が proxy server を利用するようにした CDN である. Akamai も FCAN も DNS の仕組みを利用しているのに対して, CDN4DWSS は DNS を使わない. このため CDN4DWSS は DNS を直接操作するのが難しいプライベートネットワークで利用できる.

5.4 Ustream Live Producer

Ustream Live Producer [11]はデスクトップ画面の実時間配信を行うことも可能なシステムである. しかしながら, 数十秒を超える遅延が発生する場合がある. CDN4DWSS は遅延を短くする工夫を行っている.

5.5 Web RTC

Web RTC[12] は W3C で提唱されているブラウザ間の音声, ビデオチャット, P2P ファイル共有を可能にする規格であり, Chrome や Firefox で利用可能である. DWSS に Web RTC を導入することにより, DWSS の性能をより高めることが可能となる.

6. おわりに

CDN4DWSS はモデル化したネットワークに基づいて試作を行っており, NAT や IPv6 を利用したネットワークで利用できるようにするためにはモデルと実装の方法を改良する必要がある. また, CDN の 1 つのノード内で適切なサーバを選択するとき, 現在は 2 分木の平衡のみをみているが, 遅延を中心に考えた方が, 性能が高くなる. この問題については, 上田らの研究成果[3]を利用することによって解決する可能性がある. セキュリティの強化も必要である.

今後, これらの改良を行っていく予定である.

参考文献

1) 山之上　卓: P2P 技術を利用した分散システム上の実時間操作共有システム, 情報処理学会論文誌, vol.46, No.2, p.392-402 (Feb. 2005).
2) 芝崎 亮,千葉 大作,中沢 実,服部 進実: IT 教育向けデスクトップ管理ツール「MultiVNC」の実践報告, 情報処理学会コンピュータと教育研究会情報教育シンポジウム, Vol.2005, pp. 69-74, (Aug. 2005).
3) 上田達也, 安倍広多, 石橋勇人, 松浦敏雄. P2P 手法によるインターネットノードの階層的クラスタリング. 情報処理学会論文誌, Vol. 47, No. 4, pp. 1063--1076, (2006-4).
4) Norihiko Yoshida: Dynamic CDN against Flash Crowds, Springer Content Delivery Ndtworks: Principles and Paradigms (Rajkumar Buyya, Al-Mukaddim Khan Pathan, and Athena Vakari, eds), Springer, 277-298(2008).
5) Takashi Yamanoue: A Casual Teaching Tool for Large Size Computer Laboratories and Small Size Seminar Classes, Proceedings of the 37th annual ACM SIGUCCS conference on User services, pp.211-216, St.Louis, Missouri, US. (11-14, Oct. 2009).
6) 杉田 裕次郎, 小田 謙太郎, 下園 幸一, 山之上 卓: アドホックな環境で利用可能な Web ベースの画面共有システム, 電気関係学会九州支部第 64 回連合大会, (2011).
7) 山之上卓: 音声画像通信システム, 特許 No. 5186624, (Feb. 2013).
8) 鈴木 新一 , 水越 一貴 , 深澤 昌志 , 八代 一浩 , 鳥養 映子: 学校間ネットワーク上に構築した遠隔教育支援システムの接続手法の提案とその評価, 情報処理学会論文誌, vol. 54, No. 3, pp. 1050-1060 (March, 2013).
9) 山之上 卓, 杉田 裕次郎, 小荒田 裕理, 小田 謙太郎, 下園 幸一: デスクトップ画像共有システムのための、トーナメントアルゴリズムを使った負荷分散機構, 情報処理学会マルチメディア、分散協調とモバイルシンポジウム 2013 論文集, Vol. 2013, pp. 429 – 434(July 2013).
10) Akamai, http://www.akamai.com/index.html, as is March, 2013.
11) Ustream Producer, http://www.ustream.tv/producer, (As of May 17, 2013)
12) Web RTC, http://www.webrtc.org/, (As of May 17, 2013)