Second Life用プロキシサーバの作成

井 関 文 一*

近年、いわゆるメタバースと呼ばれる３次元仮想空間が注目を集めている。これらの技術は次世代のインターネットの主要なサービスに発展する可能性もあり、3Dインターネットなどとも呼ばれている。

このような３次元仮想空間をリアルな教育に利用しようとする動きも活発であるが、一般的な大学などの環境ではファイアウォールが存在するなどの理由から、これらの３次元仮想空間サービスへの接続は困難である場合が多く、実質的にサービスを利用できないこともある。

今回この論文ではメタバースの典型例としてSecond Lifeを取り上げ、大学内部などから接続する場合の問題点と、その解決法の提案と実装を示す。

キーワード：3D仮想空間、メタバース、セカンドライフ、プロキシサーバ、ファイアウォール

Construction of Proxy Server System for the Second Life

Fumikazu ISEKI

Many people try to using the metaverse to education. And the Second Life is enumerated as a typical example of the metaverse. But, it is very difficult to use the Second Life in the environment where firewall exists such as universities and the enterprises.

In this paper, I clarify the problems and propose the solution and the implementation of method that connecting to the Second Life from inside of the firewall.

Keyword：metaverse, second life, proxy server, fire wall

*東京情報大学総合情報学部情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems

2008年７月９日受理

1．はじめに

近年、いわゆるメタバースと呼ばれる3次元仮想空間が注目を集めている。これらの技術は次世代のインターネットの主要なサービスに発展する可能性もあり、3Dインターネットなどとも呼ばれている。本論文執筆時点で存在する様々な3次元仮想空間サービスの中で、最も本来の意味でのメタバースに近いものとしては、米Lindenlab社のSecondLife［1］が挙げられる。

このような3次元仮想空間を現実での教育に利用しようとする動きも活発であり、Second Lifeに於いては、すでに欧米の多数の大学がこの3次元の仮想空間に進出しており、日本でもSecond Life内に仮想的な大学を作る動きが出始めている。またSecond Life内から代表的なLMSであるMoodle［2］にアクセスするためのシステム、Sloodle［3］などのツール面の開発も盛んに行われつつある。

Second Lifeなどのメタバースが教育に本当に活用できるかどうかは賛否両論があり、今だはっきりとはしていないが、それだけにこれらの環境を教育者自身の手で検証する必要がある。ところが、一般的な大学などの環境では、ファイアウォールが存在するなどの理由からSecond Lifeへの接続は困難である場合が多く、実質的にこれらのサービスを利用できないこともある。

今回この論文ではメタバースの典型例としてSecond Lifeを取り上げ、大学内部などから接続する場合の問題点と、その解決法の提案と実装を示す。

2．Second Lifeのサーバ構成と通信プロトコル

2-1．サーバ構成

Second Lifeにおけるサーバ構成［4］の詳細は明らかにされていないが、Second LifeサーバとOpenSIM［5］のサーバ群は図1のような構成になっている。両者の構成は細部での違いはあると思われるが、我々の議論においては無視できる程度の違いであると考えている。

OpenSIMやSecond Lifeでは256x256の区画（SIM）を1台のコンピュータ（SIMサーバ。Second LifeではSimulatorとも呼ぶ）が管理している。このSIM内の全ての処理（オブジェクトの管理やスクリプトの実行、衝突判定なども含む）をSIMサーバが行い、ユーザ側の表示ソフト（Viewer）はSIMサーバからの情報を表示しているに過ぎない。各SIMの位置関係はGrid Server（Second LifeではSpace Server）が管理し、ユーザのアバターがSIM間を移動する場合には、すばやくSIMサーバの切り替えが行われる。

2-2．通信プロトコル

ViewerとSIMサーバ間の通信にはHTTPSとUDPが用いられ、3次元オブジェクト（プリム）のデータやマップデータなどはUDPパケットで転送される。一方、SIMサーバの切り替えやユーザIDなどの機密性を必要とする情報はHTTPSでやり取りされる。ユーザのアバターが現在のSIMから別のSIMに移動する場合は、現在のSIMのサーバから、移動先のSIMサーバのURL（またはIPアドレス）と通信用のポート番号が通知される。

アバターが隣接するSIM間を移動する場合、移動先のSIMサーバからの応答が遅れると、Viewerはアバターの位置を正確に知ることができず、SIMサーバからの応答があるまで一時的にアバターが制御不能になることもある（場合によってはそのままログアウトになる）。

また、SIM間の移動が無い場合でも、絶えず隣接するSIMの情報がUDP通信で送られてきており、SIMサーバ群との間で10〜15個のセッションが同時に張られることも珍しくは無い。

リスト1にViewerとSIMサーバ間の通信のサンプルを示す。

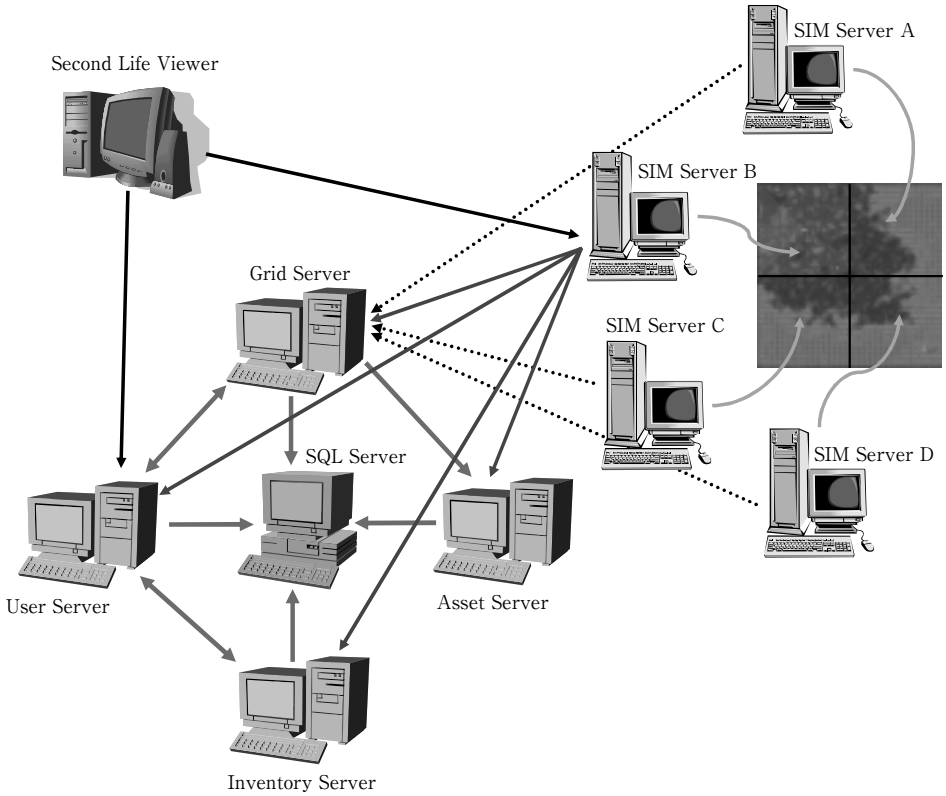

図1．Second Life互換のOpenSIMのサーバ構成。矢印は問い合わせの方向。

リスト1.UDPとHTTPS通信例

UDPのEnableSimulatorメッセージ

40 00 00 00 60 00 ff ff 00 97 00 88 03 00 00 ec
03 00 **40 81 2b 13 32 cc**

40 81 2b 13はSIMのIPアドレス（64.129.43.19）
32 ccはポート番号（13004）

HTTPSでのXMLメッセージ例

```
<llsd><map>
<key>events</key><array><map>
<key>body</key><map>
<key>agent-id</key><uuid>3158ca38-cd95-4a23-9e22-b55f7cf8a666</uuid>
<key>seed-capability</key>
<string>https://sim1216.agni.lindenlab.com:12043/cap/9f923691-886f-c993-1593-531c2e8eb8cc
</string>
<key>sim-ip-and-port</key><string>72.5.13.112:13006</string>
</map>
<key>message</key><string>EstablishAgentCommunication</string>
</map></array>
<key>id</key><integer>1</integer>
</map></llsd>
```

3．ファイアウォール内部からのSecond Lifeへの接続の問題点

前章でも説明した通り、Second Lifeでの通信方法はWWW（HTTP）での通信方法と全く異なる。

即ち、以下の通りである。

1．通信にHTTPSの他にUDPを用いる。
2．通信内容に別のSIMの情報（IPアドレスとポート番号）が埋め込まれる。

このため、大学や企業などのように、セキュリティ維持のために内部のネットワークとインターネットの境界にファイアウォールが存在するような環境では、Second Lifeを利用することは非常に困難となる。通常このような環境下でWebページを参照する場合にはWWW用のプロキシサーバを使用するが、WWW（HTTP）やSocksなどのプロトコルを対象としたプロキシサーバではSecond Lifeの通信を中継することは不可能だからである。

このような環境下に於いて、どうしてもSecond Lifeを利用したい場合には以下のような手法をとる必要がある。

1）既存のネットワークとは別に、Second Life専用の回線を引く。
2）Second Lifeを利用するPC一台一台に対して、グローバルアドレスを割り振り、使用ポートの通信がファイアウォールを通過するように設定する。
3）NAPT機能のあるブロードバンドルータなどを使用して組織内部にSecond Life用のプライベートネットワークを形成し、ファイアウォールではブロードバンドルータの該当する通信のみを通過させる。
4）外部のSecond Lifeプロキシ業者までVPNを張って接続する。
5）通信内容を書き換え可能なSecond Life専用のプロキシサーバを用意する。

1）の手法では、別の回線を引くコストが問題となる。また組織内のネットワークと完全に切り離されるため、利便性は低下する。

2）の手法ではファイアウォールの管理に手間がかかる上、PCにグローバルアドレスを割り振り、直接インターネットに接続することになるのでセキュリティの著しい低下に繋がる。組織のネットワーク管理者の立場から言えば、最も避けたいパターンであるといえる。

3）の手法は手軽に使えて効果もあるが、Second Lifeのためだけにプライベートネットワークを形成するため、プライベートネットワーク内のPCを他の用途にも使用する場合には設定が複雑になる可能性がある。

4）の手法では海外に専門の業者もあるようだが、VPNの設定が難しく、プロキシ業者に対するコストも発生するので一般的であるとは言えない。

5）の手法では管理者、ユーザ共に手間が掛からず理想的であり、PC側ではSecond Lifeの起動スクリプトを若干修正する（-loginuriオプション）だけで良い。サーバプログラムの作成にコストが掛かるが、一度作成してしまえば、WWWのプロキシのように、ユーザは何の違和感なくプライベートアドレスを割り振られたPCからでもSecond Lifeに接続すとことが可能である。

今回は5）の手法による接続を目指してSecond Life用のプロキシサーバの設計と実装を行った。

4．Second Life用プロキシサーバ

4-1．動作原理

前章での議論を踏まえて、今回開発したHTTPS及び、UDPの通信内容を書き換え可能なSecond Life用プロキシサーバの動作について説明を行う。

まず、HTTPS・UDPの中継プロセスの管理（中継プロセスの起動や削除をなど）を行うためのコントロールプロセス（コントローラ）を、

１ユーザ（１セッション）に付き、HTTPS・UDP用にそれぞれに一つずつ用意する。中継プロセス自体はViewer－SIMサーバ間の一対の通信に対して、HTTPS・UDP用をそれぞれ一つずつ用意する（図２）。

中継プロセスは絶えず通信内容を監視し、他のSIMの情報があれば、それに対応する中継プロセスの確認（起動）要求をコントロールプロセスに対して行い、通信内容の書き換えを行う。

コントロールプロセスは、中継プロセスからの新規中継プロセスの確認（起動）要求に対して、該当する中継プロセスが既に起動されているかどうかをチェックし、起動されていなければその起動を行う。

今回、Viewer－SIMサーバ間の一対の通信に対して中継プロセスを一つずつ用意しているが、これは一見するとリソースを無駄使いしているように思われるかもしれない。しかしながら、これによりそれぞれの通信が混在しなくなるので、通信の管理が容易となり、通信内容をセマンティックに解析することも可能になる。これはプロキシプログラムにさらに別の機能（キャッシュやフィルタリング機能）を持たせる場合などに有益となる（５章参照）。

中継プロセスがコントロールプロセスへ別の中継プロセスの確認（起動）リクエストを送る場合の具体的な手順は次の通りである（図３）。

図３において、現在使用しているSIMサーバから別のSIMサーバの情報が送られてきた場合（①）、中継プロセスはそのSIMに対応する中継プロセスの存在をコントロールプロセスに問い合わせる（②）。

問い合わせを受けたコントロールプロセスは、起動中の中継プロセスのプロセス一覧リストを参照する。もしプロセス一覧リストに問い合わせを受けた中継プロセスが存在しない場合には、その中継プロセスを起動し（④）、中継プロセスの待ち受けポート番号を問い合わせの

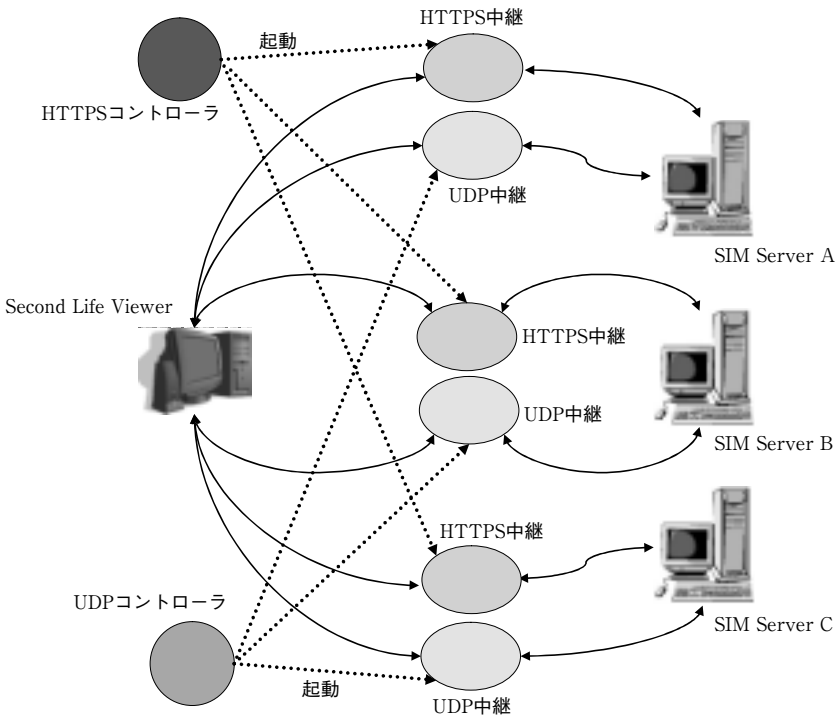

図2．中継プロセスの動作概要

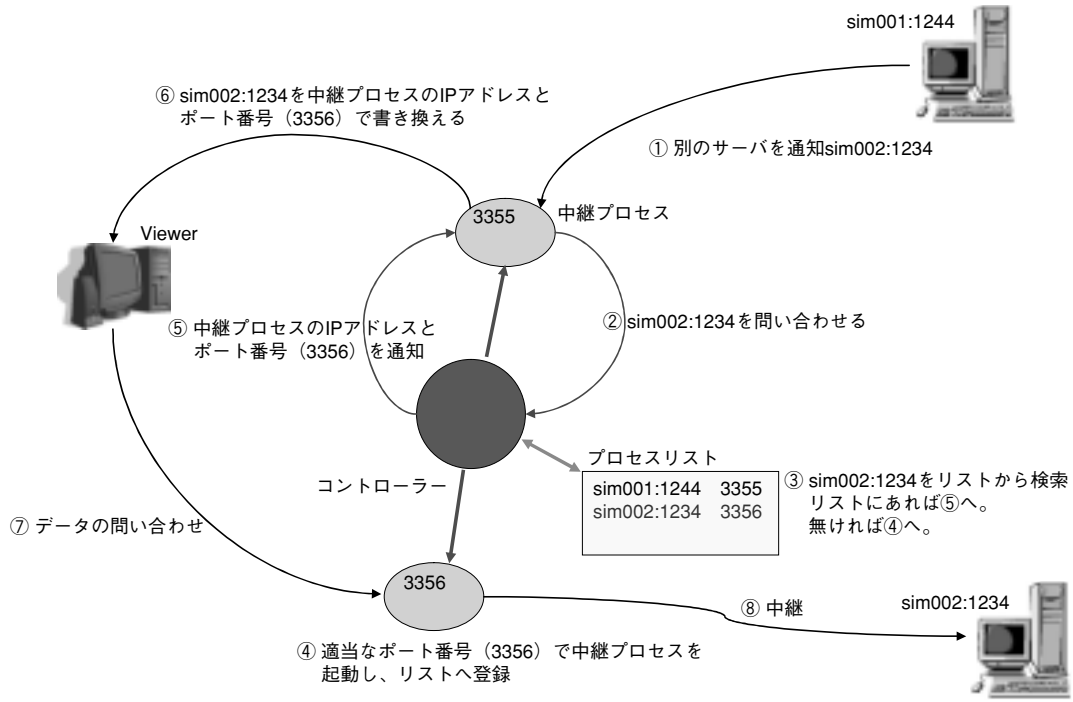

図3．具体的な中継プロセスの起動の流れ

あった中継プロセスに返す。また同時にプロセス一覧リストに、その中継プロセスの情報を待ち受けポート番号と共に格納する（⑤）。

もし既に中継プロセスが起動していれば、プロセス一覧リストに載っている中継プロセスの待ち受けポート番号をそのまま返す。問い合わせを行った中継プロセスは、コントロールプロセスより返されたポート番号と、自身のIPアドレスを使用してSIMサーバからの通信内容を書き換えてViewerに送信する（⑥）。

先に述べたようにコントロールプロセスにはHTTPSとUDP用の2つが存在し、一方のコントロールプロセスは他方のコントロールプロセスが起動した中継プロセスからもリクエストを受ける可能性がある。

4-2．実装

本プログラムの実装はC言語を使用して行った。プログラムの基礎部分は、筆者が以前より開発を行っている汎用C言語ライブラリTUIS_Lib［6］を使用した。なお、Second LifeのHTTPS通信は主にXMLを用いて行われている。XMLを処理するC言語ライブラリは幾つか存在するが、今回は限られた用途で、かつ書き換えスピードが重要であるため既存のライブラリは使用せず、XMLの主要なサブセットを高速に処理するコンパクトなライブラリを独自に開発しTUIS_Libに組み込んだ。

今回作成したプログラムsl_relayのプロセス構成を図4に示す。図中で**fork**と記載されている箇所は、C言語のシステムコールの**fork**を利用して子プロセスを起動している箇所である。今回はプロセスの並列実行に全てマルチプロセスを利用しているが、マルチスレッドの方が効率の良い場面があるかもしれない。これは実装面での今後の課題である。

本プログラムにより書き換えられるHTTPS通信のXML部分と、UDPプロトコルをリスト2、3に示す。ただしここに挙げたUDPプロ

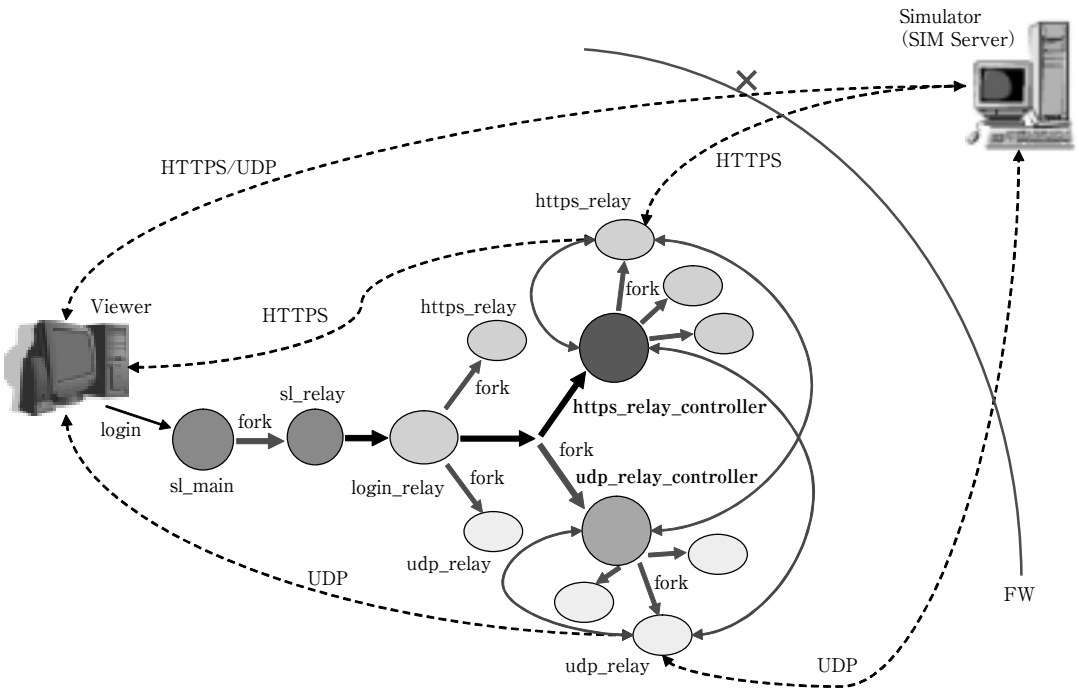

図4．sl_relayのプロセス構成

トコルの内、KickUser, TeleportFinish, CrossedRegionは実際のSecond Lifeでの通信では殆ど使用されていない。これらの変換処理は、Viewerの将来的なバージョンのための実験的な目的、またはOpenSIM用に実装されている。なお、UDPのプロトコル種別についてはlibsecondlifeの公開資料［7］［8］に基づいている。

また、中継プロセスからコントロールプロセス（コントローラ）への、中継プロセスの起動リクエストやレスポンスなどのプロセス間コマンドにはUDPを使用している。UDPを使用しているのは、通信のデータサイズが小さいこと、セッションを維持する必要性がないことなどが挙げられる。リスト4に中継プロセスとコントロールプロセス間のコマンドの一覧を示す。

4-3．ViewerのWebプロキシ機能

Second LifeのViewerはv1.19.0以降のバージョンからWebプロキシの設定機能が搭載されている。この機能を使用した場合、今回開発したプロキシプログラムの内、HTTPSの中継機能は不必要になると思われるかもしれない。しかしながら、現在のところViewerに搭載されたWebプロキシ機能は非常に不完全な機能で、Viewerのログイン処理またはViewerがSIMサーバ以外のWebサイトと通信する場合に用いられる機能であり、Viewer－SIMサーバ間のXML通信にはWebプロキシを使用することはできない。

また、Second Life内でWebサイトから画像データをダウンロードするメディア転送機能でも、このWebブラウザ設定は無視され、Viewerは特定のWebブラウザのプロキシ設定を転用する（v1.19.1.4まで確認済み）という統一性のないものになっている。

もし、これらの問題が統一され、Viewerの

リスト２．書き換えを行うXMLタグ

```
<member><name>sim_port</name><value><i4>
<member><name>sim_ip</name><value><string>
<member><name>seed_capability</name><value><string>
<map><key>MapLayer</key><string>
<map><key>uploader</key><string>
<map><key>seed-capability</key><string>
<map><key>SeedCapability</key><string>
<map><key>sim-ip-and-port</key><string>
<map><key>SimPort</key><integer>
<map><key>SimIP</key><binary>
```

リスト３．書き換えを行うUDPプロトコル

EnableSimulator（Low 151）
KickUser（Low 163）
TeleportFinish（Low 69）
CrossedRegion（Medium 7）　OpenSIM用
ParcelProperties（High 2）

リスト4．中継プロセス、コントローラ間のUDPコマンド

COM_ERROR_REPLY	C→R:中継サーバからのコマンドがエラーを起こしたことを通知する。
COM_ALIVE_REQUEST	R→C:中継サーバが起動中であることを、コントローラに通知する。
COM_IP_PORT_REQUEST	R→C:指定したIPとポート番号に対応する中継プロセスが存在するか確認する
COM_IP_PORT_REPLY	C→R:指定されたIPとポート番号に対応する中継プロセスの情報を返す。
COM_FQDN_PORT_REQUEST	R→C:指定したFQDNとポート番号に対応する中継プロセスが存在するか確認する
COM_FQDN_PORT_REPLY	C→R:指定されたFQDNとポート番号に対応する中継プロセスの情報を返す。
COM_IP_PORT_DEL_REQUEST	R→C:指定されたIPとポート番号に対応する中継プロセスが終了することを通知。
COM_IP_PORT_DEL_REPLY	C→R:指定されたIPとポート番号に対応する中継プロセス中継プロセスをプロセスリストから削除したことを通知。
COM_FQDN_PORT_DEL_REQUEST	R→C:指定されたFQDNとポート番号に対応する中継プロセスが終了することを通知。
COM_FQDN_PORT_DEL_REPLAY	C→R:指定されたFQDNとポート番号に対応する中継プロセス中継プロセスをプロセスリストから削除したことを通知。
COM_TERM_PROCESS_REQUEST	R→C:コントローラに停止を要求する。
COM_RESET_PROCESS_REQUEST	R→C:コントローラに情報をリセットすることを要求。

R→C:は中継プロセスからコントローラへのリクエスト。C→R:はコントローラからのレスポンスを示す。

全てのHTTPS通信でWebプロキシが使用できるようになった場合でも、UDPの通信を中継するためにHTTPS通信の内容を解析しなければならないので、このWebプロキシ機能がsl_relayのHTTPS中継プロセスの代用となるのは難しい。

このような状況下で、このWebプロキシ機能とsl_relayを併用した場合、さらに問題が生じる。

ViewerでWebプロキシを設定した場合、Webページの参照はsl_relay経由ではなくなるので、不要な通信内容のチェックが行われなくなり、通信速度の向上が見込まれる。しかしながら、Webプロキシが設定されている状態では、Viewerはログイン処理はWebプロキシ経由で行うがその後のSIMサーバとの通信は直接行おうとするため、sl_relayから見るとログイン処理とSIMサーバへの通信が違うクライアント（ViewerとWebプロキシサーバ）から送信されて来るように見える（図５）。

初期の段階では、sl_relayの中継プロセスはセキュリティ維持のため、最初に接続してログイン処理を行ったクライアント以外のIPアドレスからの接続は不正接続であると判断して遮断していた。そのため、Webブラウザ機能を使用したViewerは、ログイン処理後sl_relayと通信できなくなってしまっていた。この理由から、sl_relayは途中からWebプロキシを別扱いで登録するオプション（-xpオプション）を装備し、ログイン処理を行ったクライアントが、登録されたWebプロキシである場合には、その後一度だけ異なったIPアドレスからの接続を許可し、以後はそのIPアドレスをクライアントと認識する設定になっている。しかし、この場合には若干セキュリティが低下することは避けられない。

これらの問題を解決するために、最新版のsl_relayではさらに内蔵Webプロキシ（Internal Web Proxy）機能（-ipオプション）を搭載しており、自身で外部のWebページへのHTTP（S）通信のWebプロキシ機能を実現している。これにより、Webページの参照では通信速度

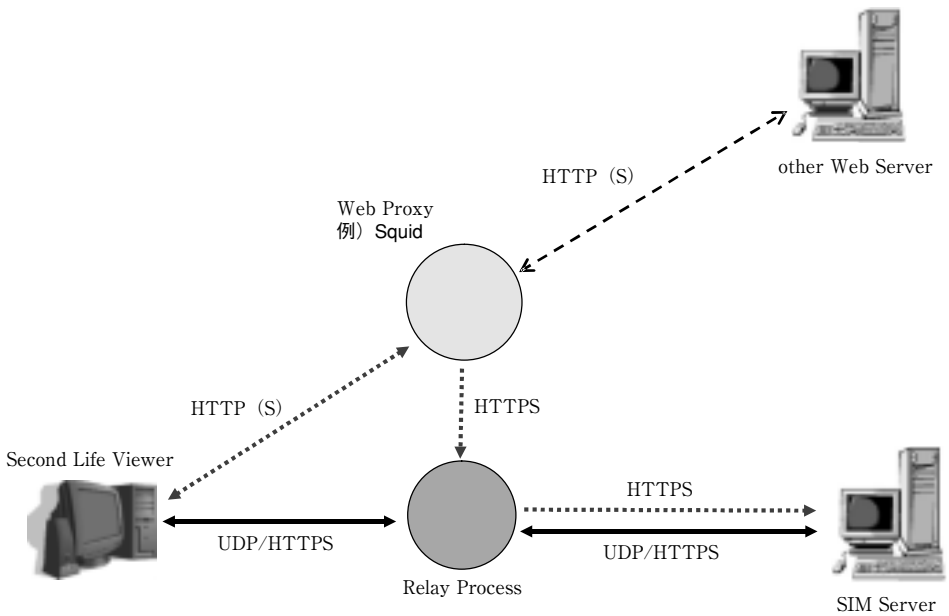

図5．外部Webプロキシを使用した場合

を落とすことなく、かつ、内蔵Webプロキシに接続してきたIPアドレスを特定できるため、ログイン処理後の接続に於いてもセキュリティレベルを維持することが可能となっている。

4-4．セキュリティ

通常Second LifeではViewerとSIMサーバ間のTCP通信には必ずHTTPSが用いられ、Viewer－SIMサーバ間のXML通信は暗号化されている。しかしながら、sl_relayではViewerとsl_relay間はデフォルトではHTTP通信となっている。オプションによりHTTPS通信を行うことも可能であるが、設定には若干の専門知識を必要としている。HTTP通信であってもパスワードなどはMD５でハッシュ化され転送されるので、一応のセキュリティレベルは確保できるが、完全とは言えない。もし、組織内のユーザを全て完全に信用できるものでないならば、HTTPS通信を行うべきであろう。

Viewer－sl_relay間でHTTPS通信を行う場合は以下の手順を踏む。

1）sl_relayが動作するマシンで、秘密鍵ファイルと証明書請求フォーマット（CSR）を作成して、CSRを認証局（CA）に渡す。この場合CSRのマシン名には必ずIPアドレスを指定する。また認証局は自前のものでも良い。
2）認証局（CA）からサーバ証明書を発行してもらう。
3）秘密鍵ファイルとサーバ証明書をsl_relayの設定ディレクトリに設置する。
4）Viewerの認証局証明書ファイルに、上記の認証局の証明書を追加する。
5）sl_relay側でSIMサーバの認証を行う場合は、Viewerの認証局証明書ファイルをsl_relayの設定ディレクトリに設置する。
6）HTTPS通信を指定するオプション（-as,-aca）付きでsl_relayを起動する。
7）Viewerの起動スクリプトで、ログインサーバのURI（-loginuri）にhttpsを指定する。

sl_relayは最初の接続時にアクセス許可リストによる接続制限を行う事が可能であり、外部のWebプロキシを使用する場合を除き、中継プロセスはログイン処理を行ったIPアドレスからの接続のみを許可する。ただし、UDP通信の中継においては暗号化なしの上にセッションを維持しないので、UDP通信のパケットを盗聴・解析された場合は、送信元のIPアドレス偽装により意図的に改変されたUDPパケットを流し込まれる可能性もある。ただし、これは元々Second Lifeのシステムに内在する問題であり、sl_relayに由来する問題ではない。

5．問題点とまとめ

平成19年９月より、当プログラム（sl_relay）をオープンソースとして公開している［６］。我々の研究室でも使用しているが、現在の所、重大な問題は発生していない。ただし、プロキシ経由のためどうしても通信スピードが若干低下してしまう。そのため学内のネットワークが非常に混雑しているような状況やsl_relayの動作しているサーバのCPU使用率が高い状況では、通常の使用時よりは、アバターが制御不可能に陥る確率が高くなる場合がある。

現在のその他の問題点、または拡張点として以下のものが挙げられる。

1）プロトコル処理がSecond Lifeの実装に依存。
2）slvoiceには未対応。
3）キャッシュの可能性。
4）フィルタリングの可能性。

１）に関しては、Lindenlab社はViewerのソースコードは公開しているが、通信プロトコルに関しての詳細なドキュメント等はまだ完全には公開していない（特にHTTPSでのXMLのフォーマットについて）。従って、現在は使用されていないが、将来使用される可能性のあるプロトコルの処理については、一部（KickUser, TeleportFinish）を除いて具体的な使用例が不

明であるためこれらの実装を行っていない。

例えば、隣接するSIM間の移動時の切り替えについて、Second LifeとOpenSIMでは同じ処理を違うプロトコルで実現している（つまり実装が異なるということ）。当初はSecond Lifeにあわせて処理を行っていたため、OpenSIMではSIMの切り替えがうまく行かなかった。今後も同じような問題が発生する可能性がある。

2）では、まだ調査不足ではあるが、slvoiceはプログラムの中に最初の接続ポイントが予め固定的に設定されているようである。Viewerが起動オプションによりログインサーバを自由に変更できるのとは対照的に、slvoiceの場合は接続先が固定でなので、プロキシで中継させることは非常に困難になると思われる。

3）、4）は拡張点であるが、3）ではこのプログラムを利用てSecond Lifeの通信データをキャッシュし、速度を向上できる可能性がある。ただし、HTTPS通信は先に述べた通り、機密性の高いデータの転送を行うだけなので、HTTPSをキャッシュしても速度は上がらない可能性が大きい。またそもそもSecond LifeでのHTTPS通信は殆どの場合、CGIで処理を行うので、キャッシュ自体が無意味になる場合が多い。

従って、キャッシュにより通信速度の向上を狙うとすれば、UDPのデータをキャッシュしなければならないが、Second Lifeのように動的にデータが変化する環境では、キャッシュを行うデータの種類とキャッシュ時間を注意深く選択しないと、受信データに矛盾を生じることになる。

現在のところ、マップレイアウトやテクスチャデータなどのUDPデータをキャッシュするのが最も有効的ではないかと考えている。

また、4）に関して、Seconf LifeではSIMによっては暴力や性的表現が許されている場合もある。WWWでも有害なサイトへのアクセスを規制する動きが出始めており、Second Lifeでも教育関係で使用する場合には、それらの場所への移動を禁止したいと考える教師も多いと思われる。

この場合、プロキシの中継プロセス部分にホワイトまたはブラックリストを持たせ、リストの参照によりデータの中継を取りやめれば、フィルタとして動作させることも可能である。ただし現実的には、移動させたくないSIM全てを列挙するブラックリスト方式は完全なリストの作成が非常に困難なので、ホワイトリスト方式の方が実現し易いと思われる。

これらの拡張点については今後の課題であるが、sl_relayのそれぞれの中継プロセスはSIMサーバと1対1に対応しているので、これらの拡張機能を実装することは、技術的にはそれ程難しくはないと思われる。

このプログラムにより、大学や企業などのファイアウォールが設置してある環境でも容易にSecond Lifeに接続することができるようになった。我々教育者としては、Second Lifeのようなメタバースが本当に教育に使えるのか、使えないのか、または使うべきなのか、使うべきではないのかなどを注意深く見極める必要がある。

この研究は東京情報大学平成19年度共同研究の援助を受けて行われました。

参考文献・参照

［1］http://secondlife.com/
［2］http://moodle.org/
［3］http://www.sloodle.org/
［4］https://wiki.secondlife.com/wiki/Server_architecture
［5］http://opensimulator.org/wiki/Main_Page
［6］http://www.nsl.tuis.ac.jp/xoops/modules/wmpdownloads/viewcat.php?cid=2
［7］http://www.libsecondlife.org/wiki/Protocol_%28network%29
［8］http://www.libsecondlife.org/template/