G-30

Linux 版 t-Room における実システム環境を用いた動作確認
Experiment evaluation of Linux based t-Room system in real t-Room environment

福岡　篤志 †　和田 理 †　片桐 滋 †　大崎 美穂 †
Atsushi Fukuoka　Osamu Wada　Shigeru Katagiri　Miho Osaki

1　はじめに

近年のネットワーク技術の大幅な進展に基づき，音声や映像などのメディアデータの大容量伝送が可能になってきた．こうした背景を受け，遠隔地間の共同作業を支援するためのシステムの研究が進められている [1–3]．

こうした支援システムの一つに t-Room がある [4]．t-Room は，遠隔地の共同作業者どうしがあたかも同じ部屋にいるかのような感覚，すなわち同室感を，より具体的には視聴覚メディアの知覚に関する対称性を作業者に提供することにより，共同作業の支援を目指す．その実現のため，t-Room は，等身大ディスプレイから成る壁面（モノリスと呼ばれる）によって部屋空間を構成し，また各ディスプレイに対峙するカメラと，複数台のラウドスピーカとマイクロフォンを持ち，さらにそれらのメディア機器にかかわるメディア信号の入出力制御と t-Room 間のメディア信号の伝送を制御するミドルウェアを持つ．

当初，上記の制御用ミドルウェア（以下，デバイス制御システムと呼ぶ）は Windows を OS として開発されていた [4]．しかし，メディア信号処理の高速化や開発の利便性を考慮し，著者の一部らによって Linux 版 t-Room（以降では，ディスプレイやサーバ（コンピュータ）等のハードウェア部は t-Room 装置と呼び，t-Room 装置とその制御用ソフトウェアであるデバイス制御システムとの総体を t-Room と呼ぶ）の開発が始められた [7]．Linux 版 t-Room は，カメラやマイクなどのメディア機器（デバイス）を制御するためのデバイスサーバと，そのデバイスサーバを制御するデバイス制御システム [7,8] から成り立っていた．また，デバイス制御システムにおけるサーバ制御は階層的に行われていた．そこでの制御プログラムは上位層から順番に起動され，下位層のサーバが上位層のサーバに接続する形をとりながら，その制御を次第に下位層に移行させていった．ただし，各階層におけるサーバプログラムの起動や接続の手続きは，利用者が手動で行っていた．

一部屋を 6 面のモノリス壁面で構成する現状においても，一式の t-Room 用で稼働するサーバの台数は軽く 10 台を超える．従って，起動や接続の手続きを全て手動で行うことは明らかに非効率である．こうした問題の解決を目指し，本研究ではサーバ接続手続きの自動化を目指す．以下本稿では，手続き設計の詳細を述べ，6 面のモノリスからなる t-Room 上で行った手続き実装結果を報告する．

図 1　t-Room 装置実装例の外観．

†同志社大学大学院　理工学研究科，Graduate School of Science and Engineering，Doshisha University

2　関連研究

2.1　t-Room

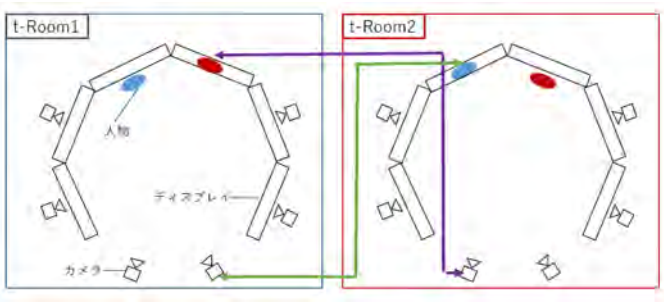

図 2　t-Room におけるメディア機器の配置とメディア機器間の信号伝送との概要．

t-Room の装置外観とメディア機器間の信号伝送概要を，それぞれ図 1 と図 2 に示す．図 2 は，6 面のモノリスから成る t-Room の装置（ハードウェア部）実装例を示している．基本的に，接続される t-Room は，いずれも同一構造を持つことを前提としている．そうした t-Room を接続する際には，図 2 に図解するように，t-Room 内の同一場所にあるモノリスどうしが対応するようにメディア信号，特に映像信号の収録と再生に（位置関係的な）工夫を施すことで，メディアの対称性の実現を目指す．図 2 から，接続される t-Room の部屋空間が仮想的に重ね合わされるように機能することがわかる．なお，音信号については，当初，映像信号の扱いと同様に収録用マイクロフォンも再生用ラウドスピーカもモノリスに設置する方式が試みられていたが，最近では（モノリス壁面の）ディスプレイ内における音像と映像との一致性を高める技術の研究の立場から，この初期の方式に見直しがなされている [5,6]．こうした状況を受け，本稿では，メディア装置の配置に関しては映像信号のそれに焦点を合わせる，すなわち図 2 のようにディスプレイとカメラを配置することを前提として議論を進める．

図 2 から，1 台のカメラで撮影された映像が 1 台のディスプレイ上に再生される関係がわかる．t-Room どうしの接続は，基本的に，このような各モノリスに対応する 1 対 1 の（1 台のカメラと 1 台のディスプレイの）メディア信号の通信の組み合わせとして実現される．

2.2　デバイス制御システム

2.2.1　階層構造に基づく制御

本研究で用いるデバイス制御システムは，図 3 に示すような階層構造による制御方式を採用する．各層のサーバは，自身の配下にある層のサーバを制御しながら最終的には t-Room 全体を制御する構造となっている．

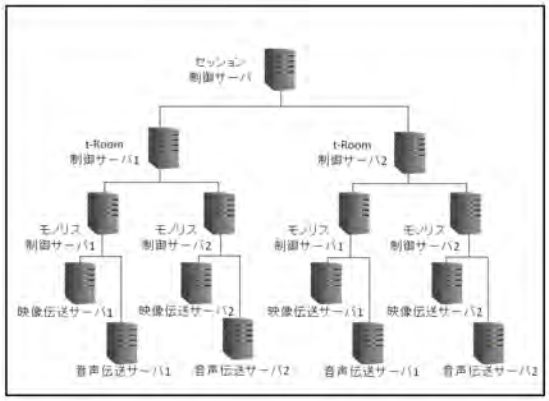

図 3　t-Room におけるサーバどうしの階層構造.

図 3 中，映像伝送サーバと音声伝送サーバは，先行研究おいて既に開発されていたサーバ [9,10] を，階層構造に従って機能できるように改良したものである．以降この 2 つのサーバをメディア伝送サーバと総称することにする．また，メディア伝送サーバの上層にあるモノリス制御サーバは，それぞれ 1 台ずつの映像伝送サーバと音声伝送サーバを制御するためのサーバである．モノリス制御サーバとそれに付随するメディア伝送サーバとの組みの数は，各 t-Room が持つモノリスの数に応じて決められる．t-Room 制御サーバは，モノリス毎に存在するモノリス制御サーバを制御し，t-Room 全体を統括する．最上位層のセッション制御サーバは，原理的にネットワーク上に 1 台存在し，電話交換機やルータなどのように t-Room どうしの接続補助を行う．

2.2.2　サーバ接続の流れ

表 1　t-Room 制御サーバ情報.

項目	詳細
IP アドレス	t-Room 制御サーバの IP アドレス
通信用ポート番号	通信用ソケット確立に用いるポート番号
制御用ポート番号	制御用ソケット確立に用いるポート番号
t-Room 名	各 t-Room 固有の名前

図 3 中の各サーバは次のような手順で動作する．

t-Room どうしの接続に先立ち，初めに各 t-Room 内においてサーバどうしの接続を行い，各サーバが持つ属性情報をツリー構造で表現するコンテンツツリーを作成する．このコンテンツツリーは，後に t-Room 間どうしの接続に用いる．t-Room 内のサーバ接続は，階層構造に沿って上位から下位に向けて次のように行う．

まず，t-Room 制御サーバをセッション制御サーバに接続する．なお，セッション制御サーバは，常に t-Room 制御サーバからの接続を受け付ける状態にある．t-Room 制御サーバからの接続を受け付けたセッション制御サーバは，接続してきた t-Room 制御サーバから表 1 にある t-Room 制御サーバ情報を受け取り，登録を行う．表 1 中，通信用ポートは t-Room 制御サーバ同士でのソケット確立に使用する．また，制御用ポートは，セッション制御サーバと t-Room 制御サーバ間でのソケット確立に使用する．t-Room 名は対応する t-Room がユニークに持つ名称である．

次に，モノリス制御サーバを，以下の手順で t-Room 制御サーバに接続する．t-Room 制御サーバからセッション制御サーバに対する接続が完了した後，t-Room 制御サーバはモノリス制御サーバからの接続を待つ状態となっている．そこでモノリス制御サーバを t-Room 制御サーバに接続する．そのとき，t-Room 制御サーバは，モノリス制御サーバから表 1 と同様の情報を受け取り登録する．

上段と同様の手順で，メディア伝送サーバをモノリス制御サーバに接続する．

こうして各 t-Room 内におけるサーバどうしの接続を完了し，続いてコンテンツツリーを作成する．まず，モノリス制御サーバ単位でコンテンツツリー情報を作成する．モノリス制御サーバは，自身のサーバ情報と，接続してきたメディア伝送サーバのサーバ情報を保持している．これらの情報をツリー構造の XML 形式のファイルに保存することで，モノリス単位のコンテンツツリー情報を作成する．続いて，作成したコンテンツツリー情報を t-Room 制御サーバに送信する．t-Room 制御サーバは，各モノリス制御サーバから受信したコンテンツツリー情報を t-Room 単位で統合し，t-Room 単位のコンテンツツリーを作成する．この t-Room 単位のコンテンツツリーは，サーバどうしの関係をツリー構造で表現する XML 形式ファイルとして保存し，また後続する t-Room どうしの接続に用いる．

t-Room 毎のコンテンツツリーの作成を終えた後，続いて各 t-Room 制御サーバは，セッション制御サーバを介して接続相手となる t-Room 制御サーバとの間でコンテンツツリーを交換し，互いに接続相手の情報を得る．接続相手となる t-Room 制御サーバのコンテンツツリー情報を受け取った t-Room 制御サーバは，コンテンツツリー情報をモノリスごとに分配して配下のモノリス制御サーバに送信する．モノリス制御サーバも同様に接続相手の情報を配下のメディア伝送サーバへ送信する．接続相手の情報を得ることによって，図 4 のように，階層毎に地点間で対応するサーバどうしは通信を確立させ，メディア伝送サーバのそれぞれに音声信号あるいは映像信号の伝送を始めさせることで t-Room 間の接続を完成する．

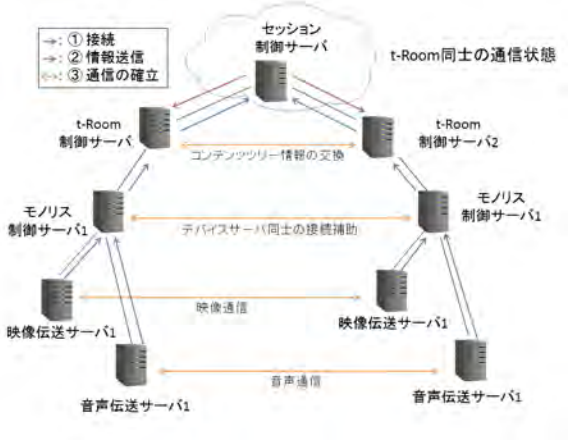

図 4　階層構造に基づく通信の確立.

3　接続手順の自動化

3.1　スクリプトによる起動

先行研究においては，デバイス制御システムにおける各サーバは，それぞれのターミナル上で手動によって起動させていた．そうした手動による起動処理は，t-Room 間の通信を行うまでに要する時間が長くかつ手間が大きくなり，研究開発の効率を低下させてきた．また，例えばデバイスサーバがモノリス制御サーバへの接続に失敗した場合，セッション制御サーバにその失敗の情報を反映した後で再度セッション制御サーバのプログラムを立ち上げる必要があった．こうした手間は，単に時間を要するだけでなく，操作を複雑にし，誤操作のリスクも高める．言うまでもなく，t-Room 技術の詳細を熟知していない一般の利用者にとって，このような手動による操作は望ましいものではない．

この問題を解決するため，本研究では，t-Room の接続を自動化する手法を提案・実装する．そこで，まず本小節では，デバイス制御システムが対象としているサーバを自動的に起動する仕組みを構築する．起動は，最上位層のセッション制御サーバから順に下位層のサーバに向かって行う．

ハードウェアの構成によっては，1 台のコンピュータ上に複数のサーバが稼働する場合がある．そのような場合は，起動予定のサーバ分の起動コマンドを Linux のシェルスクリプトに記述し，それを対象コンピュータ上のバックグラウンドプロセスとして実行させる．結果的に，記述された順でサーバは自動的に起動されていく．

一方，異なるコンピュータ上でサーバを起動させたい場合がある．例えば，図 1 の実装例においては，1 台の t-Room 内にはモノリスの台数に等しい 6 台のコンピュータがあり，そのそれぞれの上でモノリス制御サーバとメディア伝送サーバ（本稿では特に映像伝送サーバ）の 2 種のサーバが稼働し，また同時に，それらの 6 台の内の 1 台の上でセッション制御サーバと t-Room 制御サーバが稼働している．図 5 には，2 台のモノリス（同時に 2 台のコンピュータ）に着目したサーバの配置と，それらのサーバを起動させるためのシェルスクリプトに記載されるサーバ接続の様子を例示している．図 5 では，モノリス 1 用のコンピュータ上で利用者（操作者）が t-Room の起動を行おうとしていることを仮定する．そのとき，利用者は，シェルスクリプトファイル”tRoom.sh” を実行することでそのコンピュータ上でセッション制御サーバから映像伝送サーバまでの一連のサーバ群を起動することができる．一方，モノリス 2 用のコンピュータ上のサーバは，モノリス 1 用のコンピュータからリモート操作用の ssh コマンドを用いて起動する．なお，映像伝送システムでは，映像を表示するために GUI 操作が必要となり，ssh コマンドにおける環境変数 “DISPLAY” をリモート処理の受け側コンピュータ上における bash 設定ファイルにおいて DISPLAY=：0.0 と設定することで，その GUI アプリケーションの起動を可能にしている．

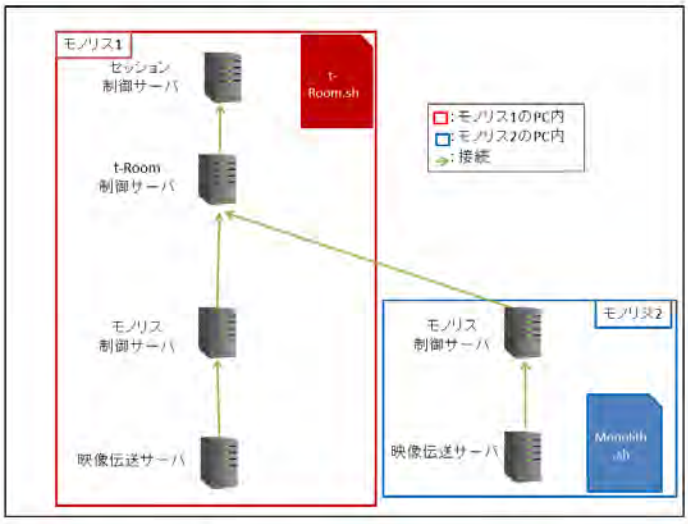

図 5　スクリプト実行によるサーバの起動．

以上のように，シェルスクリプトと ssh コマンドの利用によって，1 台のコンピュータ上で起動処理を行うことで複数台にわたるサーバ群の全てを起動することが可能となる．

3.2　メッセージによる階層間での接続確認

先行研究では，階層間のサーバ接続において接続結果の確認を行っていなかった．そのため，接続の途中で発生する問題を検出することが困難であった．また，接続作業の自動化を行うにあたり各階層間でのお互いの接続状況を把握するためにも接続の確立を確認する機能が必要である．

今回実装した接続確認機能は，接続を受け付けるサーバと接続作業を実行するサーバとが互いにメッセージの送受信を行い，相手側からメッセージを受信した際に受信に成功した旨を相手側のサーバに送信する形をとる．そこで，接続を受け付けるサーバは上位階層に置き，接続を要求するサーバを下位階層に置く．下位層のサーバは，接続を要求するにあたって “connecting” というメッセージを送信する．このとき，上位層のサーバはそのメッセージ “connecting” を受信し，さらに上位層のサーバは下位層のサーバに “acceptok” というアクノレッジメッセージを送信する．こうして下位層のサーバはメッセージ “acceptok” を受信し，接続が確認される．このメッセージの送受信の様子を図 6 に図解する．

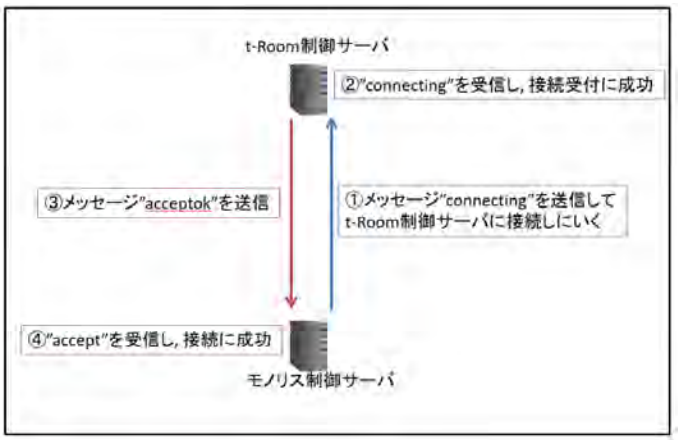

図 6　階層間の接続確認．

3.3　接続受付の自動的な終了

表 2　ユーザーコマンドの種類．

コマンド	機能	コマンドが存在するサーバ
e	接続受付終了	セッション制御・t-Room 制御・モノリス制御サーバ
a	新規 t-Room 追加	t-Room 制御サーバ
q	通信の終了	t-Room 制御サーバ

従来のデバイス制御システムでは，その制御対象であるサーバの接続処理が実行された後に，利用者がキーボードからコマンドを入力することによってその後の処理を操作できるようになっていた．そこで用いられる入力コマンドとそれぞれの機能について表 2 にまとめる．

表 2 中，コマンド a は，接続され通信が行われている t-Room の組みに対して，さらに異なる t-Room を追加的に接続するためのコマンドである．またコマンド q は，t-Room 間の通信が行われている状態において操作用ターミナルに表れるコマンドであり，通信・接続を終了させるためのものである．従って，これらの 2 つのコマンドは，t-Room を接続させる手順の自動化には関係がなく，本研究における自動化の対象外とした．

一方，表中のコマンド e は，上位層のサーバが下位層のサーバから要求された新規接続の受付を終了するために用いるものであり，接続手順の自動化にも影響する．実際，従来のデバイス制御システムでは，コマンド e が入力されることにより，接続しようとしている t-Room 中の接続対象サーバの追加を終了し，コンテンツツリーの作成や t-Room 制御サーバどうしによるコンテンツツリーの交換，さらには t-Room 間の通信へと処理が進む．これまで手入力されてきたこのコマンドに代え，本自動化の処理においては，タイムアウトの検知によって新たなサーバの追加を自動的に判断するように変更した．

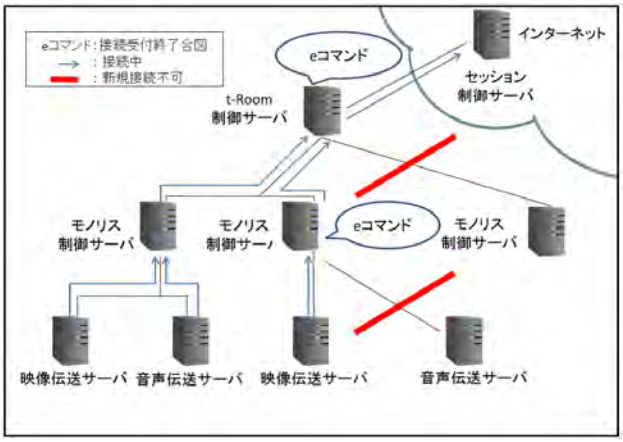

図 7　コマンド e による接続受付終了.

まず，従来のコマンド e が入力されたときに接続受付が終了する様子を図 7 に示す．図 7 のように t-Room 制御サーバを起動しているターミナル画面からコマンド e が入力されると，接続受付を終了するので，自身の配下にあるモノリス制御サーバの新たな接続を受け付けない．同様にモノリス制御サーバも起動しているターミナル画面からコマンド e が入力されると，接続受付を終了するので，自身の配下にあるメディア伝送サーバの新たな接続は受け付けない．以上のように各サーバを起動しているターミナル画面からコマンド e の入力により接続受付が終了するようになっていた．

これに対し，本研究で行ったコマンド e の処理にかかわる自動化は，スリープ機能を用いて指定した時間の経過後に接続受付を自動的に終了するように変更した．その様子を図 8 に示す.

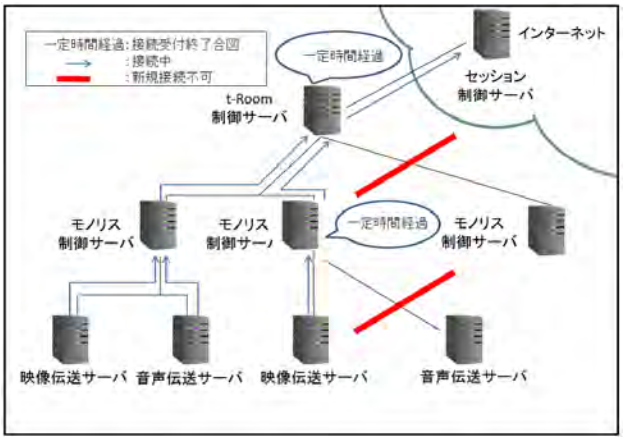

図 8　タイムアウトによる接続受付終了.

図 8 のようにスリープ処理による指定時間の間は次の処理が行われず配下のサーバからの接続を受け付けている状態となっており，スリープによる指定時間が経過したら接続受付を終了をする処理が行われることによって自動化な接続受付の終了を実現している．尚セッション制御サーバ側でのコマンド e の入力は t-Room 制御サーバからの接続受付を終了する処理が行われると共に通信開始の処理も行われるので，利用者が通信開始をしたいタイミングで始められるように従来のようにコマンド e で終了をするようにしている.

4　2 台の t-Room を用いた動作確認

4.1　動作環境

新たに開発した接続手順の自動化処理の部分を含むデバイス制御システム全体が，計画通りにサーバの起動と相互の接続を自動的に実行し得るかどうか，2 台の t-Room を用いて検証した．なお，2 台の t-Room の内の 1 台で利用できるコンピュータが 2 台のみであったため，動作確認はいずれの t-Room においても 2 台のコンピュータのみを用いて行った.

用いたコンピュータの諸元を表 3 と表 4 に示す.

表 3　動作確認実験に用いたコンピュータの諸元（t-Room 制御サーバ用）.

OS	CentOS 7.2.1511
CPU	Intel Core i7-4790 CPU 3.60GHz
コア数	8
メモリ	8GB

表 4　動作確認実験に用いたコンピュータの諸元（音声伝送サーバ用）.

OS	macOS Sierra 10.12.2
CPU	3.60GHz Intel Core i7
コア数	2
メモリ	8GB 1600MHz DDR3

用いた t-Room の構成と階層構造における各サーバの構成を図 9 と図 10 に示す.

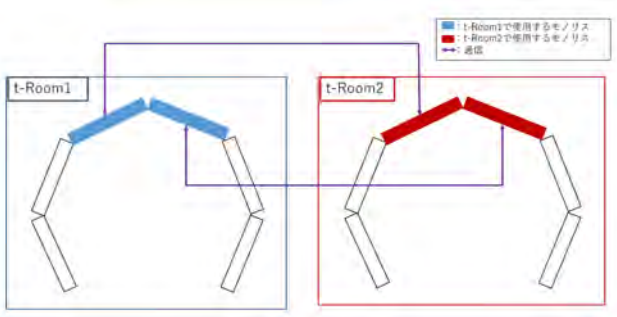

図 9　実システム環境の構成.

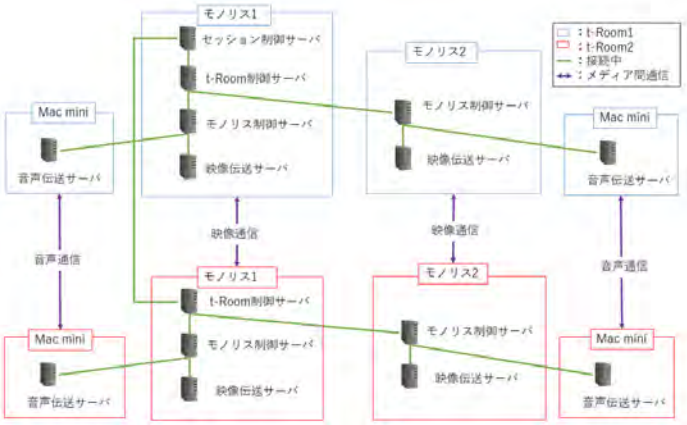

図 10　実システム環境のサーバ構成.

2 台の内の一方の t-Room を構成している 1 台のコンピュータ上でセッション制御サーバを動作させ，また 2 地点の t-Room でそれぞれ 1 台のコンピュータ上で t-Room 制御サーバを動作

させた．そして，各モノリスに対応するコンピュータ上でモノリス制御サーバと映像伝送サーバを動作させた．映像通信においては開発環境が LinuxOS をプラットフォームとしているので t-Room に使用しているコンピュータ上で動作させた．

音声通信については，従来より音声伝送を行うために開発されてきたシステム [10] は映像通信の場合と同様に開発環境が LinuxOS をプラットフォームとしているものであった．しかし LinuxOS で使用されている音声 API は複数チャンネルによる音声出力を行うには難しく，複数人または複数の楽器等による音像の再生が想定される場合にはこのシステムは適しておらず，Mac OS の CoreAudio を用いた音像再生のための制御システム [11]，及び音声伝送システムの開発 [12] が行われてきた．

このことから動作確認で行う際の音声通信における音声伝送サーバは開発済みの Mac OS による音声伝送サーバを階層構造として制御できるように改良をして組み込んだものを使用することにしている．

4.2 動作結果

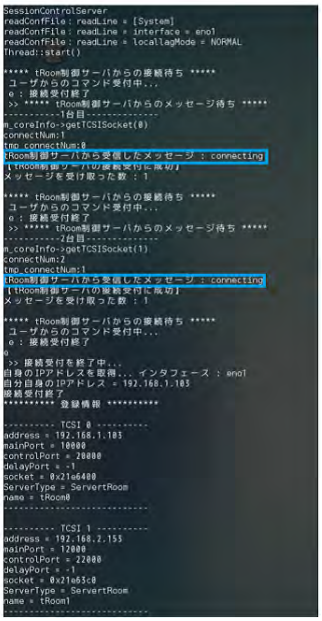

図 11　セッション制御サーバのコンソール画面．

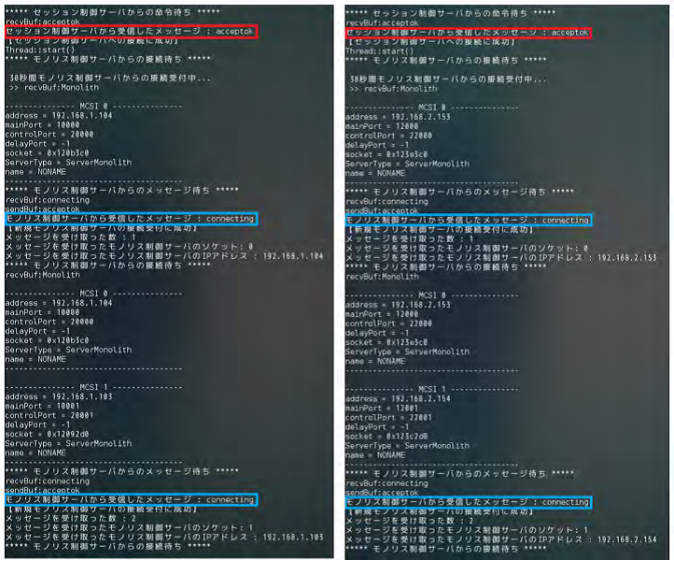

図 12　t-Room 制御サーバのコンソール画面．

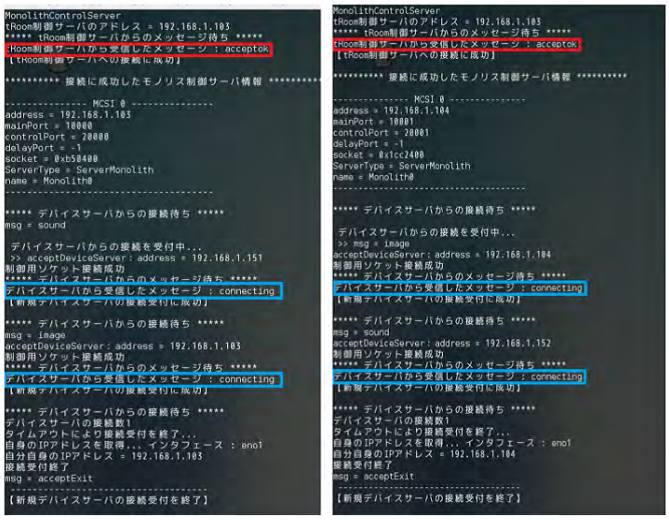

図 13　モノリス制御サーバのコンソール画面 (t-Room1 側)．

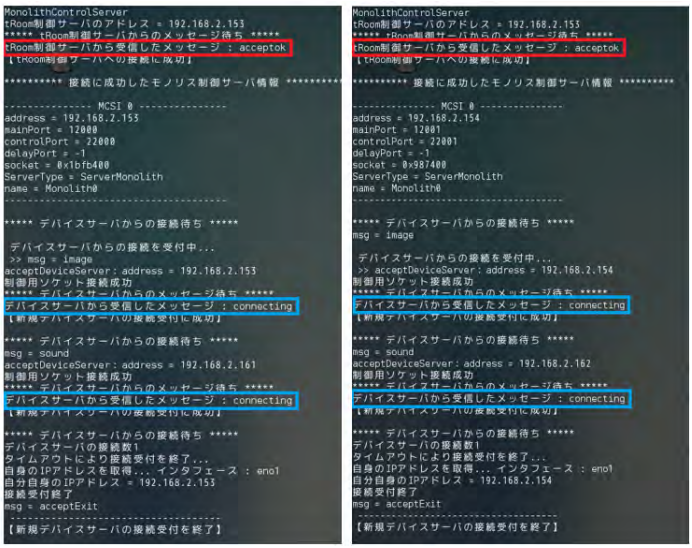

図 14　モノリス制御サーバのコンソール画面 (t-Room2 側)．

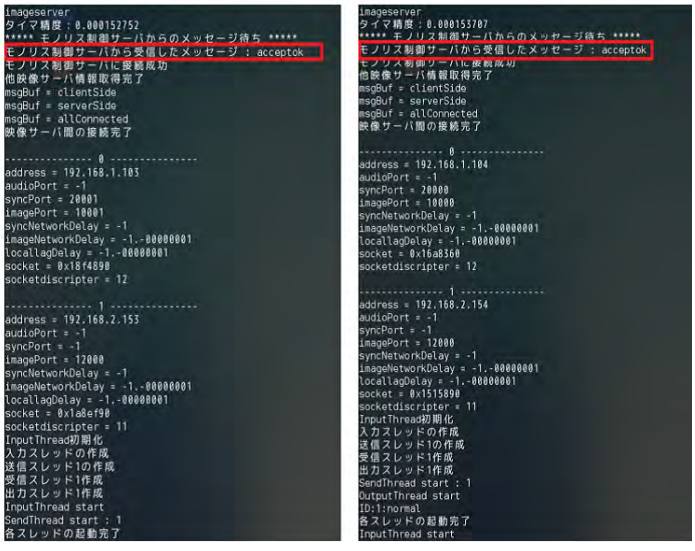

図 15　映像伝送サーバのコンソール画面 (t-Room1 側)．

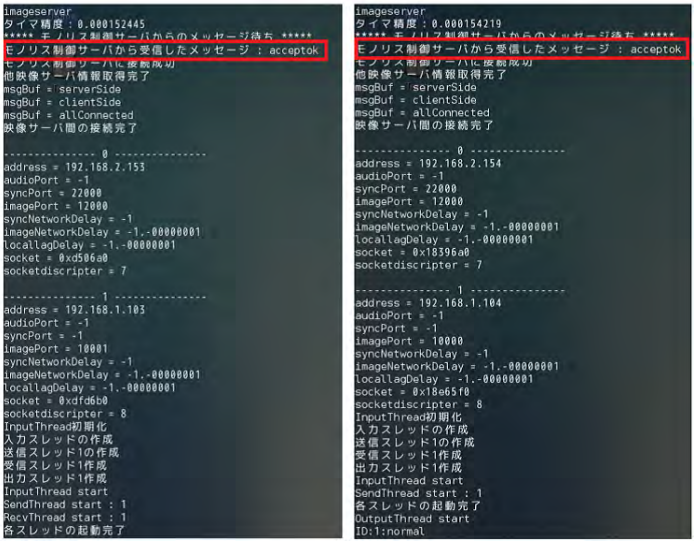

図 16　映像伝送サーバのコンソール画面 (t-Room2 側).

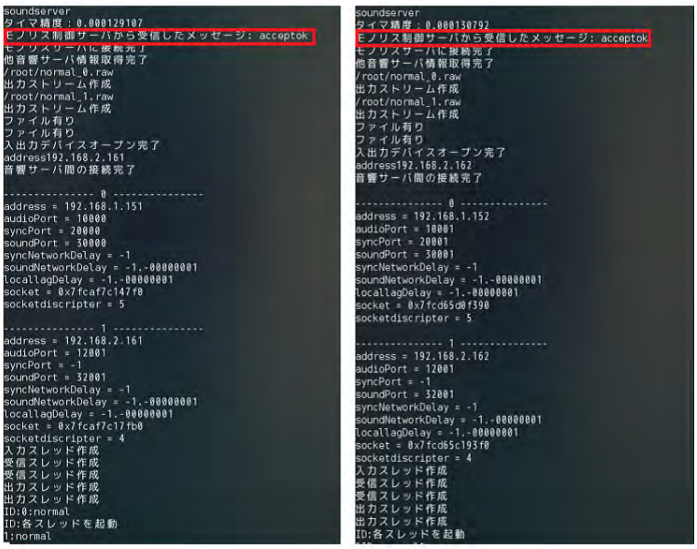

図 17　音声伝送サーバのコンソール画面 (t-Room1 側).

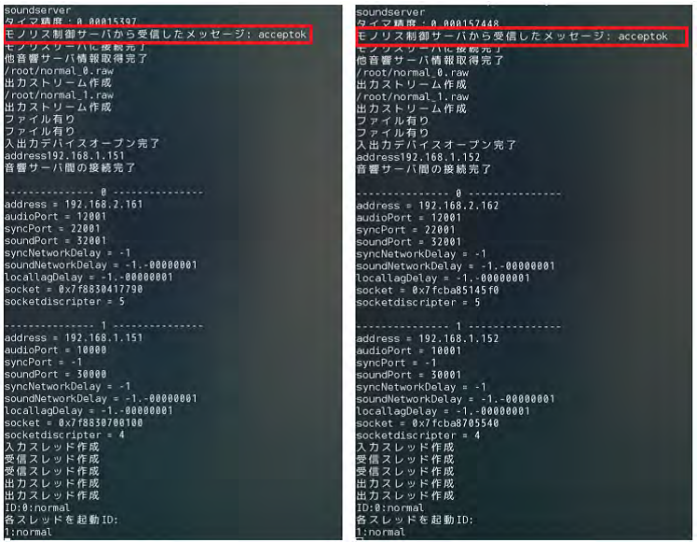

図 18　音声伝送サーバのコンソール画面 (t-Room2 側).

まず，通信が 2 地点間で行われた際の各サーバのコンソール画面を図 11〜図 18 に示す．図はそれぞれ，図 11 がセッション制御サーバの実行中の画面を，図 12 が t-Room 制御サーバの画面を，図 13 が t-Room 1 側のモノリス制御サーバの画面を，図 14 が t-Room2 側のモノリス制御サーバの画面を，図 15 が t-Room1 側の映像伝送サーバの画面を，図 16 が t-Room2 側の映像伝送サーバの画面を，図 17 が t-Room1 側の音声伝送サーバの画面を，そして図 18 が t-Room2 側の音声伝送サーバの実行中の画面を示している．なおここで，各図中の赤枠で囲んだ部分はメッセージやり取りによって下位層のサーバが上位層のサーバからメッセージ “acceptok” を受信した部分を示している．また青枠で囲んだ部分はメッセージやり取りによって上位層のサーバが下位層のサーバからメッセージ “connecting” を受信した部分を示している．

次に映像通信が行われた結果について述べる．なお，本確認実験では，確認の利便性を考慮し，t-Room 本来のカメラ位置，すなわちディスプレイに対峙する位置ではなく，ディスプレイ上部にカメラを配置した．撮影対象として，各地点の t-Room 名が書かれた椅子をそれぞれの t-Room 内に置き，それをカメラで撮影し，相手地点に伝送，そして相手地点のディスプレイに再生した．

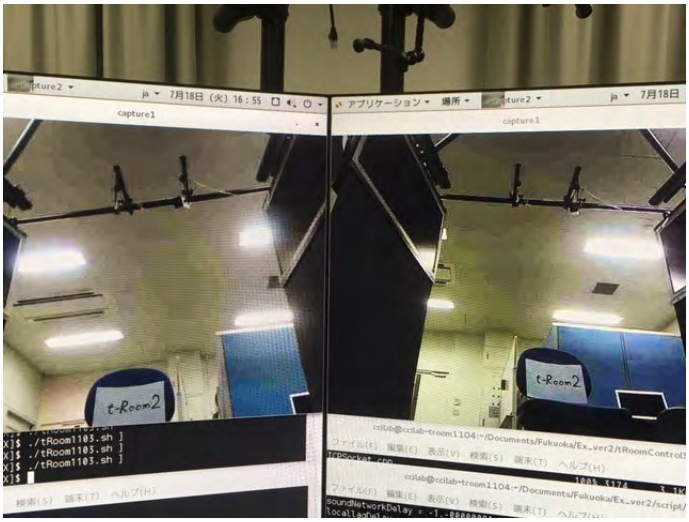

図 19　t-Room1 側の映像画面.

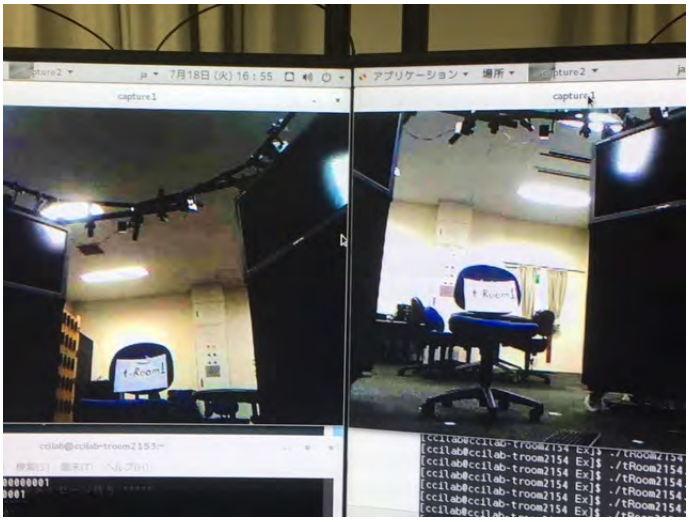

図 20　t-Room2 側の映像画面.

接続中，すなわち t-Room 間で映像通信が行われている状態における，t-Room1 側の映像画面を図 19 に，同様に t-Room2 側の映像画面を図 20 に示す．

図 19 では t-Room2 側に置かれた t-Room2 と書かれた椅子が画面に映し出されており，図 20 では t-Room1 側に置かれた t-Room1 と書かれた椅子が映し出されている．2 面ずつのモノリスを使用して 2 地点間の t-Room 同士の通信が行われていることが確認できる．

4.3　接続手続きの操作回数の変化

本節では自動的接続手続きの開発を行ったことによる従来の接続手続きとの操作回数の変化について述べる．ここでは 2 地

点間の t-Room 同士の通信を行い，その際に使用するモノリスの数は各地点共に最大数である 6 台を使用すると想定した場合で操作回数を示すことにする．

まずは従来の接続手続きにおける操作回数を以下の表 5 に示す．

表 5　操作回数 (従来版).

操作内容	操作回数
セッション制御サーバ起動	1 回
t-Room 制御サーバ起動	地点数 (2) = 2 回
モノリス制御サーバ起動	モノリス数 (6) ×地点数 (2) = 12 回
デバイスサーバ起動	デバイス数 (2) ×モノリス数 (6) ×地点数 (2) = 24 回
デバイスサーバからの接続受付け終了	モノリス数 (6) ×地点数 (2) = 12 回
モノリス制御サーバからの接続受付け終了	地点数 (2) = 2 回
t-Room 制御サーバからの接続受付け終了	1 回
合計	54 回

次に自動的に接続手続きが行われる場合における操作回数を以下の表 6 に示す．

表 6　操作回数 (自動的接続手順).

操作内容	操作回数
スクリプト起動	モノリス数 (6) ×地点数 (2) = 12 回
t-Room 制御サーバからの接続受付終了	1 回
合計	13 回

表 5，表 6 より自動的に接続手続きが行えるように開発をすることによって従来の手動で行っていた接続手続きより大幅に操作回数が減っていることが確認できる．

5　おわりに

先行研究で開発された Linux 版 t-Room にためのデバイス制御システムに関して手動による接続手続きの煩雑さと利用者の負担を軽減するために，サーバプログラムの起動と接続受付の終了処理とのシェルスクリプトによる自動化を実現した．また実際の t-Room 装置を用いて，実装した自動化手続きが設計通りに動作することの確認を行った．確認実験の結果，t-Room 間で映像と音声のメディア信号の伝送が共に正常に行えていることと，Linux 版 t-Room のためのデバイス制御システム全体が正しく動作していることが確認できた．

しかし，従来より提案されている Linux 版 t-Room のための階層構造においてはモノリスごとにメディア制御を行う構造となっているために利用者が移動をしながら通信を行う場合，即ち t-Room 全体によるメディア制御が必要な場合は現状では対応できないことが問題となっている．そのため今後は t-Room 単位によるメディア制御を考慮した新しい階層構造の実装を目指すことにしている．

参考文献

1) Harrison, Steve：Media Space 20+ Years of Mediated Life，Springer Publishing Company，Inc. (2009).
2) Morikawa，O．and Maesako，T．: HyperMirror : Toward Pleasant-to-use Video Mediated Communication System，Proc. CSCW ’98，ACM，pp. 149-158 (1998).
3) 三輪敬之，石引　力，渡辺　隆，篠原　淳：影を自己のエージェントに用いた共存在的コミュニケーションシステムの開発，電子情報通信学会技術研究報告．HCS，ヒューマンコミュニケーション基礎，Vol. 103，No. 113, pp. 35-40 (2003).
4) Hirata, K., Harada, Y., Takada, T., Aoyagi, S., Shirai, Y., Yamashita, N., and Yamato, J.: *The t-Room: Toward the Future Phone*, NTT Technical Review, Vol. 4, No. 12, pp. 26-33 （2006）.
5) 澤崎博章，山下 春香，中谷 彰皓，片桐　滋，大崎美穂： 映像の有無を伴う音響反射板方式の音像定位 , 精度の評価，電子情報通信学会，EA2013-65，pp9-16 (2013).
6) 山下 春香，中谷 彰皓，片桐　滋，大崎美穂：ディスプレイ上下に配置した音響反射板つきスピーカの特性分析，日本音響学会 2015 年春季研究発表会 (2015).
7) 荻野裕也，杉本直也，片桐　滋，大崎美穂：Linux 版 t- Room の開発：デバイス制御システムの設計と実装，情報処理学会研究報告，Vol. 2014-GN-90, No. 13 (2014).
8) 荻野裕也：Linux 版 t-Room のためのデバイス制御システムの開発，同志社大学大学院 理工学研究科 情報工学専攻 修士論文（2013）.
9) 村上　昂，飯田卓也，片桐　滋，大崎美穂：遠隔コラボレーション支援のための映像伝送システムの開発とその遅延評価，情報処理学会研究報告，Vol. 2013-GN-86, No. 19 (2013).
10) 竹森幸輝，前田佳奈，岩原正典，片桐　滋，大崎美穂：ローカル・ラグ制御機能とログ同期機能を持つ音響サーバの開発，情報処理学会研究報告，Vol. 2012-AVM-76, No. 3 (2012).
11) 中谷彰皓，片桐　滋，大崎美穂：高臨場感を与える視聴覚ディスプレイのための音像制御システム，情報処理学会研究報告，Vol. 2015-GN-94, No. 23 (2015).
12) 松尾雄真，片桐　滋，大崎美穂：Mac OS のためのローカル・ラグ制御機能を持つ音声伝送サーバの実装と性能評価，情報処理学会関西支部　支部大会　講演論文集 (2016).