Vol.2013-IOT-20 No.26
2013/3/15

異種テストベッド間のテストベッド連携手法の提案

榎本 真俊[1,a)]　櫨山 寛章[1,b)]　三輪 信介[2,c)]　奥田 剛[1,d)]　山口 英[1,e)]

概要：本論文では，異なる運用方針や運用形態をもつテストベッドである StarBED と Deterlab でテストベッド連携を行うための方法について紹介を行う．我々は CTF (Collaborative Testbed Federation) という名前の異種テストベッド間の連携を行うためのアーキテクチャを設計し，実際行った StarBED と Deterlab でテストベッド連携について説明する．StarBED，Deterlab ともにネットワーク実験を行うことを目的としたテストベッドであるが，資源の提供方法，予約方法や割り当て方針や実験者の認証方針はそれぞれのテストベッドで異なる．また，StarBED では SpringOS や Deterlab では emulab tools といったテストベッドを運用するためのツールを独自に作成している．本論文では，各テストベッドの運用方針や運用形態を変更することなくテストベッド連携を可能にする CTF の設計を行い，実際に運用した結果について報告を行う．

Enomoto Masatoshi[1,a)]　Hazeyama Hiroaki[1,b)]　Shinsuke Miwa[2,c)]　Takeshi Okuda[1,d)]
Suguru Yamaguchi[1,e)]

1. はじめに

ネットワークに関する実験を行うテストベッドとしてネットワークテストベッドが構築されている．ネットワークテストベッドでは，ネットワークプロトコル，ネットワークのコア技術やアプリケーションを検証が行われている．ネットワークテストベッドは検証する種類によって 2 種類に分類することができる．一つ目は，JGN-X [1] や Internet2 [2] のようなネットワークのコア技術を検証することを主な目的としているテストベッドである．二つ目は，StarBED [3] や Emulab [4] や Deterlab [5] のようなネットワークプロトコルやアプリケーションの検証を目的としたテストベッドである．本論文では二つ目に挙げたテストベッドをネットワークエミュレーションテストベッド (NET) と呼ぶ．

NET には以下の 3 点の特徴がある．一点目は保有している資源がテストベッドごとに異なることである．二点目は実験環境の構築や実験に用いるツールは各テストベッドで異なることである．三点目は各テストベッドで行うことができる実験規模は，各テストベッドが保有する資源量に依存することである．このような特徴があるため，各テストベッドで実施可能な実験が異なる．よって，近年テストベッド同士を接続することで実施可能な実験の制限を取り払う，さらに単一テストベッドでは実施困難だった実験を行うことを目的としたテストベッド連携に関する研究が活発に行われている．

各テストベッドは，異なる運用形態や運用ツールで管理されている．そのため，テストベッド連携を行う方法として，既存の運用形態を維持しつつテストベッド連携を行う方法と，テストベッド連携用の運用形態や運用ツールに移行する二つの方法が考えられる．前者は既存の運用形態を維持するためテストベッドのオペレータにとって容易にテストベッド連携の運用を行うことができる反面，テストベッド連携を行うテストベッドの運用形態によってはテストベッド連携自体が困難になってしまう場合が考えられる．後者はテストベッド連携を行うのに適した運用形態にテストベッドの運用方法を変更することになるため，テストベッド連携を行うことは容易である．しかし，テストベッドの独自性が失われてしまう可能性がある．本論文では，テストベッド連携を行う方法として，既存の運用形態を維持しつつテストベッド連携を行うことを目的とする．

[1] 奈良先端科学技術大学院大学
Nara Takayama Ikoma 8916–5, Japan
[2] 情報通信研究機構
東京都, Nukuikita-machi Koganei 4–2–1 , Japan
[a)] masatoshi-e@is.nasit.jp
[b)] hiroa-ha@is.nasit.jp
[c)] danna@nict.go.jp
[d)] okuda@is.nasit.jp
[e)] suguru@is.nasit.jp

2. 関連研究

関連研究として, Slice Federation Architecture (SFA) [6], Deterlab Federation Architecture (DFA) [7], CONET Federation Architecture (CFA) [8] について説明する．基本的に，現在提案されているテストベッド連携アーキテクチャでは実験シナリオを入力や資源の管理を行うために中央サーバが存在する．以下ではそれぞれのテストベッド連携アーキテクチャについて説明を行う．

SFA は実験ノードをスライスと定義し，スライスの貸し出しとスライスを接続するネットワークを貸し出すことを目的としたテストベッド連携アーキテクチャである．SFA は Planetlab [9]，Onelab [10]，Panlab [11] や GENI で採用されている．SFA では，ユーザは中央サーバ (Centralized Federator) にノード情報と実験トポロジを入力すると，実験にトポロジに合わせて実験環境が構築される．SFA は利点として，MyPLC [13] と呼ばれるコントローラを用いて連携する方法が確立されている．MyPLC は Planetlab や GENI で使われており，これらのテストベッドと連携する場合は MyPLC を利用すれば良い．SFA を導入する場合，テストベッドのシステム構築形態が Slice-base の管理形態に固定されるため，現在他のシステム構築形態でテストベッドを運用している場合は変更が必要になる．

DFA は Emulab や Deterlab といった Emulab-base のテストベッドを連携することを想定したアーキテクチャである．DFA はプラグインを用いてそれぞれのテストベッドの運用方針を変更せずにテストベッド連携を行うことを目的としている．ユーザは中央サーバである Experiment Controller (EC) に tcl で記述された実験トポロジを入力する．EC はユーザが入力した実験トポロジを topodl という中間言語に変換し，各テストベッドにある Access Controller (AC) に変換した情報を渡す．AC では topodl に変換された情報もとに，テストベッド内に実験環境を構築する．DFA は利点として，AC をテストベッド側が実装するため，各テストベッドの運用形態に合わせたテストベッド連携を可能にしている．一方，Emulab-tools を用いたテストベッドを想定している．そのため，テストベッドのシステム形態は Emulab-tools に限定されてしまう．

CFA は API を基本としたテストベッド連携アーキテクチャであり，CFA は各テストベッドが独自の API を提供することでテストベッド連携を行なっている．CFA はユーザに対してテストベッド間で共通する API に関しては RESTful の共通 API(TA API) を定め，中央サーバを介して各テストベッドへのインターフェースを提供している．ユーザは提供されたインターフェースを利用して各テストベッドのリソースの操作を行う．CFA の利点は CFA は API を提供するのみのアーキテクチャのため，各テストベッドの自治が可能であることである．一方，ユーザが共通 API に定義されていないテストベッド固有の資源や機能を使う場合，ユーザは個々のテストベッドが提供する API を直接使用しなければならない．そのため，CFA を採用する場合，共通 API と固有の API の開発を行う必要がある．

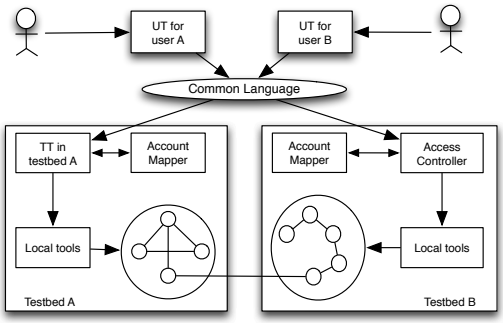

図 1 CTF architecture

3. CTF の設計

本論文ではテストベッドのシステム構築形態や管理形態を改変せずにテストベッド連携を実現するアーキテクチャとして，Collaborative Testbed Federation (CTF) を提案する．図 1 に CTF の概要を示す．

CTF では，テストベッド側のシステム構築形態や管理形態を改変せずにテストベッド連携を実現するために二種類の翻訳機を用いる．翻訳機は，利用者側の翻訳機 (UT) とテストベッド側の翻訳機 (TT) になる．UT はユーザから入力された実験記述を TT が理解可能な言語への翻訳を行う．TT では UT が翻訳した言語を入力として，実験ノードおよびネットワークの設定を行う．TT は UT からの入力に対して，各テストベッドが環境構築に用いるソフトウェアを用いて実験環境構築を行う．TT では TT が管理するテストベッドでは対応していない機能の場合は記述を無視して設定を行う．

テストベッドを使用するときにはじめに行うことは実験を行う上で必要な資源を確保することである．連携した環境では複数のテストベッドの資源を確保しなければならない．よって，テストベッド連携アーキテクチャでは利用者の資源確保を助ける機能をもつ必要がある．

テストベッドではそれぞれ独自のユーザ認証システムをとっている．テストベッドのユーザ認証システムは資源の貸出や資源のアクセス制御にも用いられているため，これを変更することはテストベッドの管理の大きな部分を変更する事になってしまう可能性がある．そのため，ユーザアカウントの差異を吸収する機能が必要になる．CTF ではこの機能をアカウントマッパーとする．

最後に，テストベッド連携の環境では実験に用いるトラフィックが流れるネットワークを接続しなければならない．テストベッドによっては，資源はインターネットから隔離されているためインターネットを通した接続には特別

な方法が必要になる．よって，テストベッド連携アーキテクチャではテストベッド同士の実験トラフィックが流れるネットワークの接続方法の提供が必要になる．

4 節では StarBED と Deterlab 間でテストベッドを行うための実装である TT4S の設計及び実装について述べる．

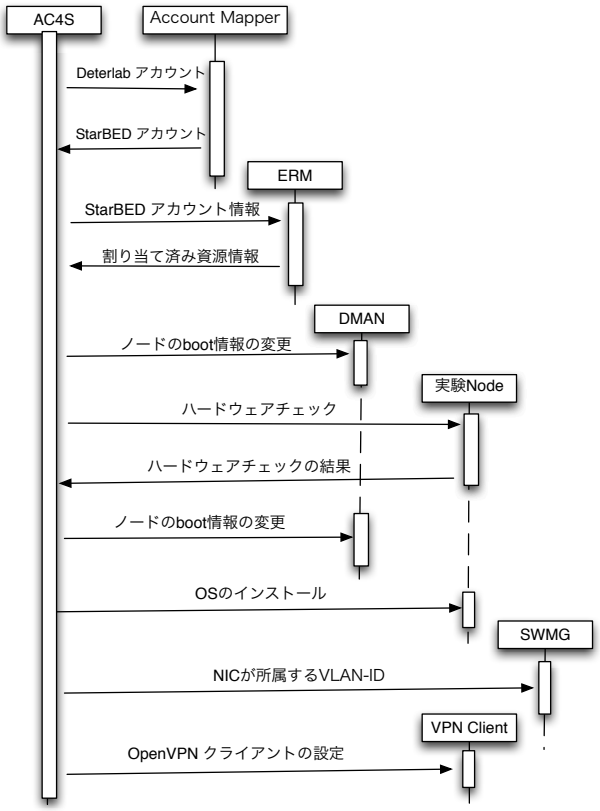

図 2 TT4S の処理の流れ

4. TT4S の設計

Deterlab では Emulab-base のテストベッドで連携を行うためのテストベッド連携のアーキテクチャとして DFA が実装されている．本論文では，CTF を用いて Deterlab と StarBED のテストベッド連携を行うために，DFA を拡張した TT4S を提案する．

4.1 StarBED と Deterlab の運用方針，方法の違い

テストベッドには運用形態により，システムによるテストベッド連携が困難であるため，オペレーションにより解決しなければならない部分が存在する．そのため，はじめにあらかじめオペレーションにより解決しなければならない部分を知るために，テストベッドの違いをまとめる．

運用方針や運用方法について，StarBED と Deterlab では以下の違いがある．一点目は資源の予約方法である．資源の予約はそれぞれ，StarBED のように事前に使用する資源と期間を決める方法と Deterlab のように実験を行う時に使用可能な資源を借りる方法である．二点目は，ユーザアカウントである．アカウントは例として，Deterlab ではユーザ個人がそれぞれユーザ ID をもち，ユーザ ID の認証にはユーザパスワードを用いている．資源の借用や資源へのアクセスはユーザ ID とプロジェクト ID を関連付けることでアクセス制御を行なっている．一方 StarBED ではユーザ ID，パスワードとプロジェクト ID で管理されているが，ユーザ ID は各プロジェクトでひとつしかなく，すべての実験者が共通のユーザ ID，パスワードとプロジェクト ID を共有している．三点目は資源の利用制限である．資源の利用制限の例としては，Deterlab では OS はあらかじめ用意されたイメージを利用するのに対し，StarBED ではユーザが好きな OS を使用することが可能である．また，ネットワークに関しては Deterlab では VLAN ID はユーザには隠されており，ユーザが入力した実験トポロジをもとにシステムが自動的にネットワークの分離を行う．一方 StarBED では VLAN の ID 空間を切り分けてユーザに提供しており，ユーザは割り当てられた ID 空間を利用してネットワークの分離を行う．これらは，テストベッド間の本質的な違いであり，これらの違いを吸収する機能を設計する必要がある．

この中で，資源の予約については StarBED では自動的な資源確保を運用方針として許可しておらず，必ず研究員との調整が必要になるためシステム的な解決は困難である．そのため，今回の CTF ではこの部分についてはオペレーションによる解決が必要になる．資源の予約以外の二つについてはシステム的な解決を行う．

4.2 TT4S の設計上の制限

TT4S を設計するに当たり，いくつかの設計上の制限を述べる．一つ目は，実験者は各テストベッドが保持している資源やテストベッド固有機能を熟知していることである．これは，CTF では実験者がユーザインターフェースの選択や改変することを前提としているためである．二つ目は，実験者があらかじめテストベッドの資源を予約しておくことである．これは，前節で述べたように，資源予約についてはシステムでの解決が困難であるからである．よって，本論文ではあらかじめ資源が使用出来る状態に対して，資源の設定を行うアーキテクチャの提案までを行う．三つ目は，各テストベッドが提供している固有機能は，その固有機能を保持するテストベッドの資源でのみ利用可能とすることである．他のテストベッドで別のテストベッド固有機能を有効にする場合，固有機能によってはテストベッドの資源の機能制限や運用制限に触れる可能性が考えられるためである．これらの制限を設定した上で，TT4S の設計を行う．以下でそれぞれの機能について説明を行う．

5. TT4S の実装

本節では，TT4S の実装の説明を行う．今回実装した機能は，アカウントマップ機能，資源確保機能，StarBED の実験環境構築ツールである SpringOS の制御機能と StarBED と Deterlab の間のネットワーク接続機能 4 つである．

5.1 アカウントマップ機能

はじめにアカウントの差を吸収する機能であるアカウントマッパーについて述べる．4 で述べたように，StarBED と Deterlab ではユーザの認証方法が異なっている．よって，TT4S では StarBED と Deterlab のアカウントの対応を保持させる．これにより，UT で Deterlab のアカウントを用いて StarBED の資源の確保や実験環境の構築が可能になる．

5.2 資源の確保と動作確認機能

次に実験に用いる資源の確保の方法について述べる．StarBED ではあらかじめ実験に用いる資源を確保し，使用期間内になることで資源が使用可能になる．また, StarBED では資源を貸し出すときに資源がハードウェアに問題がないか確認しないため，実験者には実験に使う資源量より多めの資源を予約することを推奨している．よって，あらかじめ実験に使う資源よりも多めの資源が確保されていることを前提にする．

TT4S では実験者側の翻訳機である UT(今回は DFA の EC である) から実験に用いるノード数が送られてくる．送られてきたノード数のノードをあらかじめ確保されているノードから選び出すが，使用するノードを選び出す前にハードウェアが問題を抱えていないかどうかの確認を行う．StarBED では OS の起動をネットワークブートである pxeboot で行う．そこで，ノードの起動確認としてノードを実験に割り当てる前に pxeboot で knoppix を起動し，knoppix が起動するかおよびすべてのネットワークインターフェースが knoppix から確認できるかの確認を行う．

5.3 StarBED 内の実験環境構築

使用するノードが確定したあと，TT4S は実験シナリオに基づいてノードおよびスイッチの設定を行う．OS のインストールについては，現在 TT4S でインストール可能な OS は Debian，Knoppix または tftpboot が可能な OS となっている．Debian はハードディスクにインストールしハードディスクから起動するが，knoppix と tftpboot に関してはネットワークブートになっている．スイッチの設定については StarBED の管理ツールである SpringOS の機能の一部である SWMG を使用している．SWMG にユーザ情報，VLAN-ID とノード名を渡すことで VLAN をポート VLAN で設定することができる．TT4S はアカウントマップ機能で変換された StarBED のアカウント，TT4S が実験に割り当てたノード名と TT4S が解釈した実験シナリオに書かれたネットワークのトポロジをもとに SWMG を用いてネットワークの分離を行う．

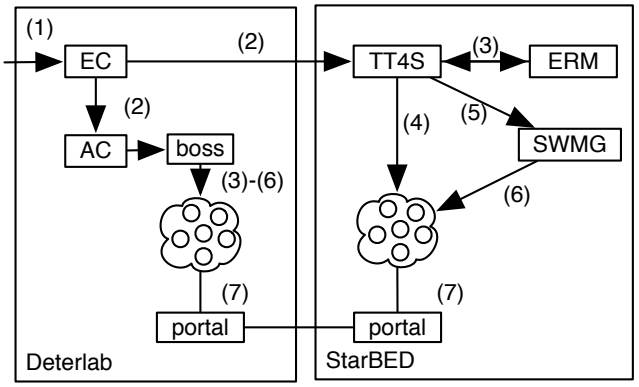

図 3 StarBED および Deterlab での環境構築の流れ

5.4 実験ネットワークの接続

最後に，実験トラフィックが流れるネットワークの接続の方法の説明を行う．実験では broadcast パケットが流れるようにレイヤ 2 でのトンネリングが要求される．そのため，CTF では OpenVPN を用いたレイヤ 2 の仮想プライベートネットワークを構築する．OpenVPN で接続するためには，実験に使用するノードの一部がグローバル IP アドレスを持ち，インターネットに直接に接続していなければならない．Deterlab は数台のグローバル IP アドレスを持った portal と呼ばれるノードが利用可能である．StarBED でも実験ノードにグローバル IP アドレスを設定することも可能であるが，設定にはオペレータによる作業が発生してしまう．ただし，HTTP プロキシを経由した接続に関しては StarBED の運用方針で許可されている．よって，TT4S では実験に用いるノードとは別に実験トラフィックが流れるネットワークを Deterlab と接続するために，OpenVPN のクライアントとして設定する．以上の処理の流れを図 2 に表す．

6. StarBED と Deterlab の連携例

今回提案した CTF のユースケースとして，StarBED と Deterlab 間のテストベッド連携を行った結果について述べる．今回のプロトタイプ実装では，CTF の中でユーザ側の翻訳機である UT として DFA の EC を利用している．また，UT と TT 間で翻訳する言語として DFA で開発された topodl を用いた．今回は，DFA の EC を利用するため，実験シナリオを入力する際に用いられるユーザアカウントは Deterlab で使用しているアカウントとなる．よって，Deterlab 側の AC ではアカウントマッパーの必要がないため，Deterlab の AC は DFA で設計された AC をその

まま利用することが出来る．最後に TT4S の実装を行い，DFA の EC と通信可能な状態にすることでテストベッド連携環境を構築することが可能な環境が整う．

6.1 EC で指定可能な設定情報

UT に入力する情報は大きく分けてノードに関する情報とネットワークに関する情報になっている．ノードに関する情報は，ノードにインストールする OS の種類，ノードを保持しているテストベッド名とノードの種類になっている．ノードの種類とは各テストベッドで定められたノードのグループの名前となっている．ネットワークに関する情報は，ネットワークの名前，ネットワークに接続しているノード，帯域と遅延の情報となっている．この中で，StarBED では帯域の制限や意図的な遅延を発生させることは通常行うことができない．そのため，TT4S では帯域と遅延の情報については，UT から数値の指定があったとしてもそれを無視している．

6.2 StarBED および Deterlab での実験環境構築手順

まずはじめ，ユーザは自分が使用する実験シナリオを EC に入力する．UT は入力された実験シナリオを topodl の記述に翻訳を行う．topodl に翻訳された実験シナリオはユーザが使用するテストベッドの Deterlab の TT および TT4S に転送される．

TT4S では 2 の手順で実験環境構築が行われる．Deterlab 側の AC では，実験シナリオで指定された環境の情報を Boss サーバに渡す．Boss サーバは実験シナリオに記述された環境を構築するために十分な資源が確保できるかどうかを判断し，確保できる場合は実験環境の構築を行う．Boss は指定された資源を確保できない場合，もしくは実験シナリオに指定されたパラメータが設定不可能な場合は TT は UT にエラーを返し，実験環境の設定を終了する．

6.3 テストベッド間ネットワークの構築

今回の実装では，StarBED と Deterlab 上に構築された実験環境の接続の自動化は行なっていない．今回は Deterlab が提供している portal を OpenVPN のサーバ，StarBED のノードの一台を OpenVPN クライアントとし，StarBED のノードから Deterlab の portal に接続することで実験トラフィックが流れるネットワークをレイヤ 2 VPN で接続している．Deterlab の portal は openvpn サーバとして自動的に起動し StarBED の OpenVPN クライアントからの接続を待ち受けている．しかし，TT4S が Deterlab の portal の情報を取得する事ができないため，StarBED 側の OpenVPN クライアントの設定を行うことができない．portal のグローバル IP アドレスを取得できない理由は，TT4S は Deterlab の EC とは接続しているが，Deterlab の AC とは接続することができないためである．そのため，

表 1 StarBED と Deterlab のノード間のスループット

	D to S		S to D	
Average[Mbits/sec]	2.21	2.39	3.68	3.86
Variance	0.06	0.03	0.04	0.05
Standard Deviation	0.25	0.18	0.21	0.22
Maximum[Mbits/sec]	2.44	2.61	3.86	4.10
Minimum[Mbits/sec]	1.59	2.09	3.28	3.46

表 2 StarBED と Deterlab 上のノード間のジッター

	D to S		S to D	
Average[ms]	4.69	4.69	6.95	6.95
Variance	0.28	0.28	14.30	14.31
Standard Deviation	0.53	0.53	3.78	3.78

今回は自動的に設定可能な部分がすべて出来上がったあとで，MyDETERlab で portal のグローバル IP アドレスを確認し，手動で StarBED の OpenVPN のクライアントの設定を行い portal に接続を行なっている．

7. 議論

CTF はテストベッドが各自の管理形態や運用方針を変えずにテストベッド連携を実現するために制限やオペレーションでの解決を試みている．

7.1 資源確保の方法について

一つ目の制限は FTA は資源の確保の方法についてである．先に述べたように各テストベッドで資源の確保の方法や貸し出し形態は異なっている．StarBED のように，あらかじめ資源量と期間が決まるような方法は，実験計画を立てる際に詳細な実験計画を立てることや大規模な実験の計画を立てることが容易である．一方，ノードの予約を予め行わなければならないため，急遽必要となった実験を行う場合は数日間またなければならない可能性がある．Deterlab のように，資源に空きがある場合に実験シナリオに応じた資源をユーザに割り当てる方法では，急遽必要となった実験に関して資源に空きがある場合に限り実験が可能である．ただし，実験の期間が明確に定められないため，資源の回収のポリシを明確に定義しない場合，資源が開放されないという事態が想定される．また，この方法の場合大規模な実験を行うのに難があるといわれている．[14] 今回は StarBED であらかじめノードの確保をしておき，ノードの利用期間内で Deterlab の資源を確保することでテストベッド連携を行った．この方法によりそれぞれのノードの確保の方法を変えることなくテストベッド連携が可能になった．Deterlab と StarBED 以外では，Emulab-base のテストベッド，Planetlab や Onelab では Deterlab のような資源の確保方法をとっているため，この方法であれば Deterlab と StarBED 以外のテストベッドともテストベッド連携が可能であると考えられる．

7.2 資源のコントロールについて

CTF は，資源の操作を行うインターフェースは持たずに，各テストベッドで提供する操作方法をユーザに利用することを想定している．StarBED では ssh 等によるリモートログインを行うための踏み台サーバが提供されている．さらに，StarBED では実験者がノードへ OS をインストールすることを許可しているため，Remote KVM を実験者に提供している．一方，Deterlab ではユーザがサーバに対して OS をインストールすることを許可しておらず，予め用意されている OS イメージをインストールした状態でユーザにサーバを貸し出す．そのため，Deterlab では remote KVM の機能はユーザに提供されておらず，ssh を用いた遠隔ログインでサーバの管理を行う．StarBED，Deterlab の両テストベッドともにノードを管理するためのネットワークと実験トラフィックを流すためのネットワークは隔離されている．ssh によるリモートログインのみで良い場合は，実験者が入力した実験環境のシナリオに 1 つ管理用のネットワークを加えて環境構築することで，1 つの踏み台からすべてのノードにアクセスすることは可能である．しかし，この方法では実験トラフィックを流すネットワークに対して管理用トラフィックを流してしまうことになるため，一部実験用トラフィックと管理トラフィックが混在するネットワークが存在する可能性が発生してしまう．

7.3 テストベッド間の実験ネットワークの接続について

テストベッド間の接続に関しては現在手作業で行なっている．これは，FTA では TT からの情報を UT で受けとることを想定しておらず，テストベッドで行った設定を TT や他のテストベッドの TT で把握できないため，VPN で接続する場合の接続先の情報が得られないためである．

CTF を用いて構築した StarBED と Deterlab の VPN サーバ、クライアントの間で帯域計測を行った．計測は 10 回行い，帯域計測の結果を表 1，ジッターの計測結果を表 2 に示す．その結果，日本とアメリカでインターネットを利用した VPN では 2Mbps から最大でも 4Mbps しか帯域がない．また，ジッターも大きいため実験内容によっては実験の信頼性が損なわれる可能性がある．ただし，狭帯域でもよくジッターが大きくても問題ない実験や，実験環境内部にインターネットで起きるような外乱を発生させる実験であれば，実験規模の拡大やさまざまな資源を利用したテストベッド連携環境が構築できる．また，テストベッド間に安定した広帯域なネットワークを構築するためには，Deterlab のように専用線をテストベッド間に構築することや JGN-X [1] や Internet2 [2] のようなインフラストラクチャを提供するテストベッドを利用することが考えられる．

8. おわりに

本論文では，StarBED と Deterlab 間の異種テストベッド連携の実例について述べた．今回は，実際に Deterlab で運用されており，Emulab-base のテストベッドを対象としたテストベッド連携アーキテクチャである DFA を StarBED に適用することで StarBED と Deterlab のテストベッド連携を行うことを目指した．しかし，StarBED と Deterlab はそれぞれ異なる運用方針を持つテストベッドであり，このようなテストベッドで連携を行う際には資源に対するアクセス制御，テストベッドで許可されている資源への操作やテストベッドの資源の外部への接続についての制限の問題があり，これらを解決する必要がある．これらを解決する方法として本論文では StarBED の運用方針に合わせたテストベッド連携を行うものとして TT4S を設計した．我々は，TT4S を用いて StarBED と Deterlab の両方の資源を用いて実験環境を構築し，構築した環境で複数の実験を行うことで TT4S の有効性を確認した．

謝辞
本研究は，独立行政法人情報通信研究機構 (NICT) による委託研究「テストベッドネットワークにおけるリソース管理および運用連携を実現するアーキテクチャの研究」の一部である．

参考文献

[1] National Institute of Information and Communications Technolgoy (NICT). Next Generation Network Testbed JGN–X. http://www.jgn.nict.go.jp/english/index.html.
[2] Internet2. http://www.internet2.edu/.
[3] StarBED Project. StarBED – A Large Scale Network Experiment Environment. http://www.starbed.org/.
[4] The University of Utah. Emulab - Network Emulation Testbed. http://www.emulab.net/.
[5] The University of Southern California and Information Sciences Institute. DETER Network Security Testbed. http://www.isi.deterlab.net/index.php3.
[6] L. Peterson, S. Sevinc, S. Baker, T. Mack, R. Moran, and F. Ahmed. PlanetLab Implementation of the Slice-Based Facility Architecture, Draft Version 0.05, June 2009.
[7] T. Faber and J. Wroclawski. A federated experiment environment for emulab-based testbeds. In *Proc. of TridentCom 2009*, pages 1–10. IEEE, 2009.
[8] CONET, the Cooperating Objects Network of Excellence. CONET Testbed Federation. http://www.cooperating-objects.eu/testbed-simulation/.
[9] The Trustees of Princeton University. PlanetLab – An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/.
[10] UPMC Paris Universitas. OneLab – Future Internet Testbeds. http://www.onelab.eu/.
[11] Panlab consortium. Panlab – Pan European Laboratory Infrastructure Implementation. http://www.panlab.net/.
[12] L. Peterson, A. Bavier, M.E. Fiuczynski, and S. Muir. Experiences building planetlab. In *Proc. of OSDI 2006*, pages 351–366. USENIX Association, 2006.
[13] Fabien Hermenier and Robert Ricci. How to build a better testbed: Lessons from a decade of network experiments on emulab. In *Proc. of Tridentcom 2012*, 2012.