Vol.2014-IOT-24 No.25
2014/2/28

ローカルブローカー型テストベッド間連携フレームワークの提案

櫨山 寛章[1,a)]　三宅 喬[2]　奥田 剛[1]　河合 栄治[3]　宮地 利幸[3]　三輪 信介[3]

概要：テストベッド間連携とは，テストベッド間を相互接続し，より大規模で高度な検証を行うための試験環境を構築する行為である．グローバルブローカーを想定しない，ローカルブローカー型テストベッド間連携モデルは，各テストベッドで独自の資源管理システムや運用管理ポリシを許容しながらテストベッド間連携を行い，単一テストベッドユーザが透過的に提携テストベッドに資源を利用できる連携モデルである．本稿では，ローカルブローカーモデルによるテストベッド連携モデルにおいて，資源予約や資源割り当て，相互接続の設定における調停を補佐，または半自動化を支援するテストベッド間連携を行うためのテストベッド連携フレームワークを提案する．

キーワード：テストベッド連携，ローカルブローカーモデル，資源予約，運用支援

A Proposal of Local Broker Model Testbed Federation Framework

Abstract: Testbed federation interconnects among different testbeds to create more larger and more sophisticated experimental environment. On the local broker model of testbed federation, different testbeds are interconnected while leaving each testbed to keep each own management system or operation policy. In this paper, we propose a testbed federation framework to support or to semi-automate the operation on resource reservation, resource allocation and interconenction amoung testbeds along with the local broker model.

Keywords: Testbed federation, Local broaker model, Resource reservation, Operation support

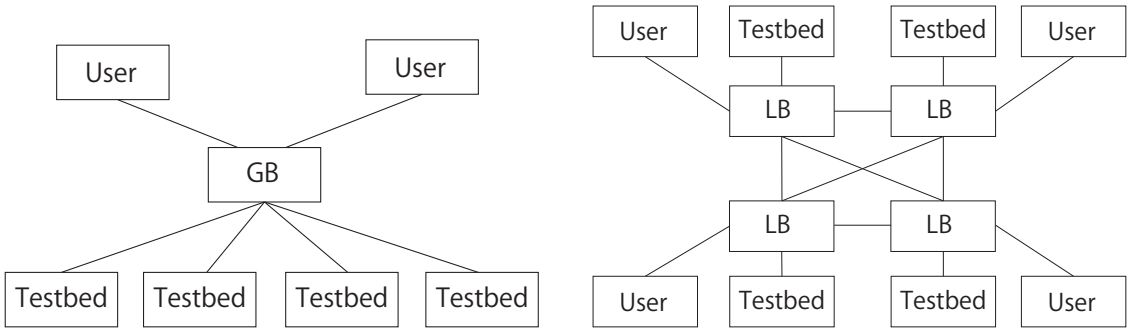
1. はじめに

テストベッド間連携とは，テストベッド間を相互接続し，より大規模で高度な検証を行うための試験環境を構築する行為である．既存のテストベッド連携では，実験者と各テストベッド運用者間でのメールなどによる手動調整によって資源予約，資源割り当ておよび相互接続設定が実施されるか，単一のテストベッド管理プラットフォームを用いたポータルサイトからの一括した管理を行うグローバルブローカーモデルによるテストベッド間連携フレームワークによって，自動化・半自動化された資源予約，資源割り当て，相互接続が試みられてきた．例を挙げると，Planetlab など [1–4] で利用されている SFA (Slice based Federation Architecture) [5]，Emulab [6] や Deterlab [7] などで利用されている DFA (Deter Federation Architecture) [8]，CONET [9] での CTF(CONET Testbed Federation) [10] がある．

グローバルブローカー型テストベッド間連携フレームワークは，連携や資源割り当ての自動化が容易である一方，テストベッドの利用方法は単一のテストベッド利用モデルに縛られる．また，各テストベッド独自の管理システムや実験支援ツールセットの利用が難しく，定められた実験方法や技術以外は利用できないという問題がある．さらに，資源割り当ての自動化により運用コストは削減されるが，反面，大規模な実験が行いにくいということも，グローバルモデルでのテストベッド利用や連携を行ってきた Emulab の過去の実験利用例の解析結果から指摘されている [11]．

JGN-X [12] のような新しいネットワーク技術を検証するテストベッドでは，こなれたネットワーク技術を定型

[1] 奈良先端科学技術大学院大学
NAIST, Takayama 8916-5, Ikoma, Nara, Japan
[2] ファットウエア
Fatware
[3] 情報通信研究機構
NICT
[a)] hiroa-ha@is.naist.jp



(a) グローバルブローカーモデル (Global Broker Model)

(b) ローカルブローカーモデル (Local Broker Model)

図 1 ブローカーモデル
Fig. 1 Broker Model

サービスとして提供する一方で，新技術のサービス化や運用者の補助による QoS 設定の調整などが実験によっては必要となる．また，StarBED [13] のような大規模なエミュレーション実験を行うテストベッドでは，運用者による資源割り当ての調停が入ることで，100 台以上の物理サーバ上に構築した 1 万を超える仮想マシンによる大規模なエミュレーション実験が実施できるようになっている [14]．このような，新技術や大規模な検証を行うテストベッドを連携する上では，個々のテストベッドとテストベッド運用の特徴を活かす必要があり，グローバルブローカーによる，単一テストベッド管理フレームワークではテストベッド連携が行いにくい．

そこで，本稿では，ローカルブローカー型テストベッド連携フレームワークを提案する．また，StarBED - JGN-X 間でのテストベッド連携を想定して開発したプロトタイプ実装の設計を説明し，現状での課題を述べる．

2. 関連研究

2.1 ブローカーモデル

図 1 はブローカーモデルの違いを図示したものである．

グローバルブローカーモデル (図 1(a)) では，単一のグローバルブローカー（GB）が存在し，ユーザは GB を通して，各ローカルテストベッドの資源予約を行う．それぞれのローカルテストベッドの資源管理は GB が基本的に行い，資源割り当てや運用者への調停依頼を行う機能も GB が担う．グローバルブローカーモデルでは，アーキテクチャや情報交換のためのプロトコルがユーザ・GB 間，GB・テストベッド間のみを取り決めればよく，実装はシンプルになり，導入や普及が行いやすいという利点がある．そのため，現在提案されているテストベッド連携アーキテクチャの多くはグローバルブローカーモデルで設計されている [5,8,10]．一方で，グローバルブローカーモデルでは同型のテストベッドや単一の利用方法のみを扱う形式になりやすく，アーキテクチャが異なるテストベッド間の連携やローカルテストベッド独自の運用方法や運用管理システム，実験支援ツール，公開 API の適用や利用が行いにくくなる．

ローカルブローカーモデル（図 1(b)）では，各テストベッドにはローカルブローカー（LB）が存在し，それぞれの管理者は LB を用いてローカルテストベッド上の資源管理を行う．また，LB は外部からのリクエストを受け取り，資源割り当てや運用者への調停依頼を行う機能も担う．そのため，ユーザは普段利用するテストベッド（ホームテストベッド）の LB に対して資源予約を行い，ホームテストベッドの LB を通して，他のテストベッドの資源を予約できる．ローカルブローカーモデルでは，各ローカルテストベッドごとに独自の運用方法や実験支援ツール，テストベッド API の利用を適用しやすい．その一方で，何らかの共通言語が存在しない限り，連携テストベッドごとに情報交換するためののプロトコルを取り決めて実装しなければならず，実装コストが高くなりやすい．

2.2 共通言語

GB を仮定しない場合，あるテストベッドの LB が実験記述を受け取ったときに，その LB は実験トポロジをローカルテストベッド内のトポロジに落とし込む必要がある．また，連携先テストベッドに残りのトポロジの作成を依頼する必要がある．この際に，連携先のテストベッドで利用可能な資源リストを取得し，連携先テストベッドに依頼するトポロジ記述を作成，連携先テストベッドにトポロジ作成依頼，作成したトポロジを取得するという手順が必要になる．この際に，テストベッドに依存しないテストベッド間共通言語が必要となる．あらかじめ策定した共通言語を使わずに，スペック交換から言語を自動生成するには，プ

ロトコル内でスペック交換を行うとともに，ドメイン知識（いわゆるオントロジ）を必要とする．

DETER [8, 15, 16] や GENI [17] では，リソース管理をモデル化し，その表現のオントロジに関する議論がされていて，現在も作業が行われている．RSpec [18] は GENI で用いられているリソース記述言語である．RSpec は基盤となる資源，ユーザ要求，資源配分にかかる制約を記述するために用いられる．基本的に Sliver とよばれるサーバ資源管理構造体中心の構造で，ネットワークに関してはそれほど豊かな情報が提供されない．定義フォーマットとして，RelaxNG Compact（RNC）[19] と XSD [20] が提供されている．

FP7 の下，Networking innovations Over Virtualized Infrastructures（NOVI）[21] というプロジェクトで，Slice-Based Federation Architecture（SFA）[5] を前提とした連携アーキテクチャを構築している．NOVI では，RSpec よりも多くの情報を提供できる NOVI-IM という情報モデルを採用している．NOVI-IM はリソースオントロジ，モニタリングオントロジ，ポリシーオントロジの 3 つからなり，それぞれの OWL が提供されている．実験実行時のリソース割当の確認や，認証連携など，我々が必要と考えている機能をある程度備えている．SFA ベースのテストベッドと連携するため，FEDERICA [22] に RSpec 拡張を施し，RSpec v2 と NOVI との協調動作を可能にするための作業を継続している．

言語の擦り合わせという意味では，テストベッド連携はサービス発見とサービス合成に似ている．例えば Moran らは，SOA の研究の中で XML ベース言語間の翻訳を Schema のマッピングで行おうとしていた [23]．

3. ローカルブローカー型テストベッド連携フレームワーク

我々は，ローカルブローカー型テストベッド連携フレームワークとして Testman テストベッド連携フレームワーク (Testbed Management Testbed Federation Framework) を設計し，プロトタイプを実装した．フレームワークの実装は，平行して実装している JGN-X テストベッド管理システムと StarBED の管理システムのモックアップシステムを JGN-X 上に構築し，それらの仮想テストベッドの連携ラッパーとして実装した．実装言語としては，XSD-python スクリプト変換ライブラリである PyXB [24] を利用して Python にて実装した．実装としては，図 2 の連携用ラッパーとテストベッド API（連携用）の部分に相当する．それ以外の部分は JGN-X テストベッド管理システムと StarBED の管理システムのモックアップシステムが該当する．後述する例で利用している `Crspeccli` はクライアント側から実行するメソッドを規定した `class Crspec_client()` のインスタンスにあたる．

3.1 ローカルブローカーモデルでの資源予約の手続き

Testman テストベッド連携フレームワークでの下記の連携ワークフローを考えている．前提として，利用者はホームテストベッドに対して利用申請を行い，アカウントやプロジェクトの登録およびそれに付随する利用契約が完了しているものとする．

(1) 利用者は連携を含んだ資源予約リクエストを自分のホームテストベッドに投入する．
(2) 利用者からの資源予約リクエストを受け取ったテストベッドは，ユーザ認証を行い，自テストベッド向けの資源予約リクエストと連携先テストベッド上の資源予約リクエストに分解する．次に，利用者に資源利用の権限があればそのテストベッドの資源を調整し，他の連携先テストベッドに対して，プロキシとして資源要求リクエストを送信する．
(3) 連携先テストベッドから資源予約リクエストを受け取ったテストベッドは資源予約リクエストに含まれている利用者または連携先テストベッドのグループの認証を行う．連携先テストベッドの利用者である場合に連携先テストベッドのはグループ属性に付与された権限を検証し，認証する．
(4) 認証に成功した場合，そのテストベッドの資源の調整を行う．

以上の手続きにより利用者は，ホームテストベッドを通して連携先テストベッドの資源を交えた実験環境の資源を入手することができ，実験環境の構築や実験そのもの開始することができる．利用者がこの LB を操作する層やインターフェースの実装において，LB が利用者ごとに資源割り当てを行い，ある利用者に割り当てた資源は他の利用者へ割り当てた資源から隔離されている状態を担保しなければらない．

3.2 テストベッド連携の手順

上述のローカルブローカーモデルでの資源予約の手続きに対して認証・権限付与を加えて考えたワークフローを詳細化し，フレームワークとして明確化すると図 2 のようになると考えられる．図 2 は StarBED と JGN-X を対象に適用した例を示しており，StarBED をホームテストベッドとするユーザから，連携先のテストベッドである JGN-X のリソースを用いた実験を投入することを考えている．図中でグレーの部分は各テストベッドに既にある管理システムを表す．連携用 API も仮定するが，連携用 API がない場合は連携用ラッパーを設置する必要がある．各ローカルテストベッドの管理システムはネットワーク運用センター（NOC）などの管理用，ローカルユーザの利用時に用いる．テストベッド連携が必要な場合には，ユーザからの要求を受け付け，一度ユーザからの要求をパースして自サイトと連携先サイト宛リクエストに要求を分解する．そして適切

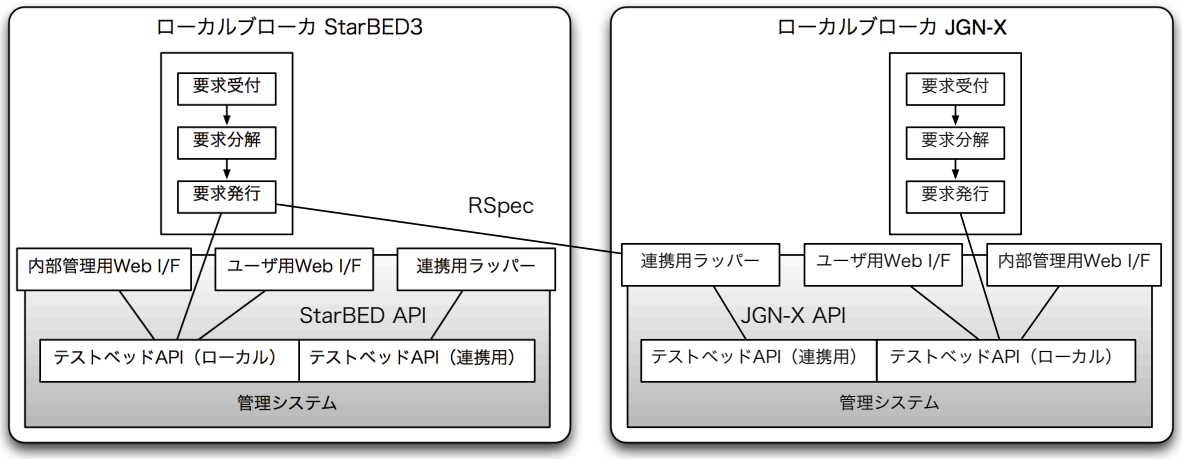


図 2 Testman テストベッド連携フレームワークを StarBED と JGN-X に適用した例
Fig. 2 Example of Testbed Federation through Testman Federtion Framework (btwn. StarBED and JGN-X)

なテストベッド API に対して要求を発行する．

実際の要求発行には複数のやり取りが含まれており，最低でも下記の要素が必要である．

- リソースリストの要求
- リソースの要求
- リソース割当結果

3.3 RSpec 拡張による中間言語の実装

Testman テストベッド連携フレームワークではテストベッド間共通言語として RSpec を拡張して実装している．RSpec は GENI で用いられているリソース記述方式であり，現在想定している StarBED と，JGN-X との連携で必要となる要素を含んでいる．StarBED と JGN-X の連携では下記の項目がリソース表記として必須であり，RSpec ではこれらが表現可能である．

CPU 資源 アーキテクチャ，動作周波数，コア数
メモリ資源 容量，データレート
ネットワーク資源 VLAN-ID，IP アドレス
ストレージ資源 容量，データレート
トポロジ スイッチ，ルータ，Ethernet の接続関係

RSpec 単体ではネットワークに関する詳細・複雑な記述が不可能であるが，XSD を拡張することで表現可能となる．また，

- リソースリストの要求
- リソースの要求
- リソース割当結果

というフローもサポートしており，本プロジェクトで用いるリソース記述方式として RSpec が適していると判断した．

我々の考える記述には施設が提供するトポロジや，実際のトポロジを記述する部分と，実験者が必要とするトポロジを記述する部分に分けられるが，両方をサポートすることにする．この擦り合わせは AnyBed [25] や ViNEYard [26] などの論理・物理資源マッピングアルゴリズムを備えた資源マッピング補助ツールが行うと仮定する．実験トポロジから，複数のテストベッドへの分割も AnyBed などの資源マッピング補助ツールが行うものとする．

3.3.1 識別子の読み替えと拡張

RSpec を利用した GENI と，JGN-X では元々提供できるリソースに大きな違いがある．そのため，RSpec の環境をそのまま利用することができない．また，プロジェクトを管理するスキームも異なっており，ユーザーやプロジェクト，物理リソース・論理リソースそれぞれにおいて相違が見受けられる．XML を利用した類似形式を利用するが，それらは違う内容のものとするほうが，設計するプロトタイプシステムにおいてはスケーラビリティや単純化を提供できる．

我々が設計段階で想定した JGN-X と StarBED とのテストベッド連携で，RSpec ではその表現が定まっていない，もしくは想定されていない項目は下記の項目である．

- **リソースの設置テストベッドの表現書式**
  JGN-X や StarBED など個々のローカルテストベッドを示す表現書式にあたる．
- **リソースの設置テストベッド内の設置拠点の表現書式**
  JGN-X 内の金沢 AP，堂島 AP，StarBED の北陸ノード，けいはんなノードなど，テストベッドのリソースが複数の拠点に分散配置されている場合に，その設置拠点を示す表現書式にあたる．
- **ユーザのアカウント，プロジェクトの表現書式**
  どのローカルテストベッドをホームテストベッドとして利用しているユーザなのかを連携するテストベッド間で一意に識別できる表現書式である．

- **テストベッド API のリストや実行を交換する書式と実行フロー**
  StarBED の SpringOS [27] など，ローカルテストベッドで公開している API を連携メッセージ越しに実行するための表現書式と実行フローである．
- **テストベッド間でのユーザ認証フロー**
  連携テストベッド間で，どのローカルテストベッドで認証されたユーザであるかを一意に識別し認証するユーザ認証フローが必要となる．グローバルブローカーモデルでは，すべての登録情報はグローバルブローカーに集約されるため認証フローの実装は簡易なものになるが，ローカルブローカーモデルではユーザやプロジェクトのの登録情報が連携するローカルテストベッドに分散するため，設計をうまく行わないと実装や実際の認証手続きが非常に複雑で処理負荷の高いものになってしまう．

RSpec の Sliver と Ticket は，JGN-X ではプロジェクト ID で代用する．RSpec の環境では，プロジェクトを管理する Ticket と，論理 Slice を管理する Sliver が存在しているが，JGN-X では貸し出しリソースを全て “プロジェクト’’ という単位において管理している．つまり，すべての貸し出したリソースは，プロジェクトに対する ID に紐付けされている．本プロトタイプシステムにおいても，これを継承しプロジェクト ID からすべての貸し出し物品を取り出せるように実装した．また，Sliver や，Ticket の URN (Uniform Resource Name) を記述する箇所はプロジェクト ID を記述するよう実装した．一方，StarBED では同一組織（ユーザ）が複数のプロジェクトを保持し，それごとにリソース申請をすることを想定して “ユーザ-プロジェクト-年” という ID の単位で貸し出しリソースを管理している．しかしながら，“ユーザ-プロジェクト-年” を JGN-X のプロジェクト ID 相当とみなすことで，JGN-X の場合と同様に扱うことができる．

URN 表記の例を示すと，リソースの表記は URN ではリソースを識別するためテストベッドドメイン名・リソース区分・リソース ID となり，

```
urn:publicid:IDN+jgn-x.jp+node+238
```

となる．ユーザ ID の表記は “アカウント名登録テストベッド名” で表記し，ローカルテストベッドで割り当てたユーザ ID を他のテストベッドでも利用し，テストベッド間での多重登録や ID マッピングのオーバヘッドを減らす設計とした．“アカウント名登録テストベッド名” とすることでどのローカルテストベッドをホームテストベッドとしているユーザなのかが一意に識別できる．

3.3.2 資源予約プロトコルの拡張

RSpec を拡張するにあたり，テストベッドでの資源予約と資源利用フローをまず洗い出した．図 3 は，利用者・研究プロジェクト・機関名の登録やテストベッドとの契約から始まり，資源予約の終了までのフローである．新規テストベッド利用者が資源予約を行うには (1) 利用契約の締結（contract），(2) アカウントの登録（account registration），(3) 操作方法や利用可能資源リストなどのリソース開示要求（manifest request），(4) 資源の予約要求（resource reservation）の 4 段階を得ることになる．

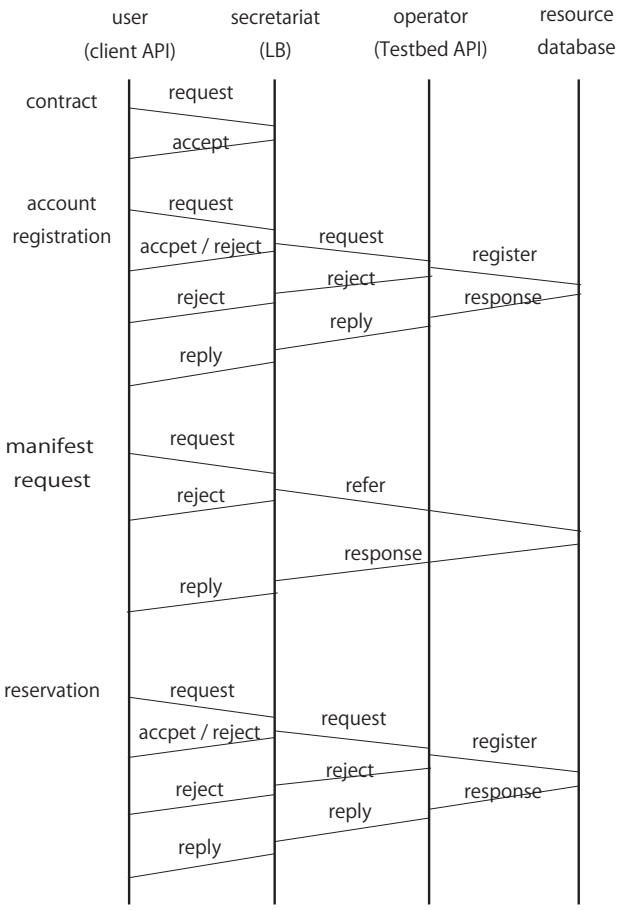


図 3 資源予約と利用の運用フロー

利用者は基本的にはテストベッド事務局（secretariat）とメールなどで事務手続きを実施し，利用者が扱うクライアント API は LB とメッセージを交換する．テストベッド事務局は利用者の要望を運用者（operator）に転送し，運用者はその要求が設定可能であれば資源管理データベース（resource database）への登録や操作を行う．システムで自動連携をサポートする場合は，LB から Testbed API 経由して各ローカルテストベッドの資源管理データベースの登録情報の取得や操作が行われる．

連携フレームワークで実施する部分は主として (3) と (4) の部分を LB が経由して連携テストベッドに伝達する．RSpec の拡張としては，リソース開示要求である `Crspccli.DiscoverResources()` とリソース割り当て要求である `Crspeccli.Request()` を実装した．`Crspccli.DiscoverResources()` は RSpec の advertize

type を拡張したもので，advertizement 要求を行い，応答は manifest type により返される．Crspeccli.Request() は Rspec の request type を拡張して実装した．応答は RSpec の manifest type の拡張によって返される．

```
Crspccli.DiscoverResources()
Crspeccli.Request()
```

3.3.3 テストベッド API 利用のためのパススルー機能の拡張

SpringOS [27] など，ユーザがローカルテストベッド固有の API を利用して割り当て資源を確認したり，割り当て資源に対して操作を加える要望が実際の実験利用の場面で想定される．そのため，標準化された API か，ローカルテストベッド API を連携テストベッド間でパススルー機能が必要となる．標準化 API は 2.2 項で述べたようなオントロジの作成や標準化団体による標準化作業を実施しないと設計および利用が難しい．また，標準化された API は最大公約数の機能になることが多く，個々のローカルテストベッド特有の機能を提供する API を含めにくい．そのため，Testman テストベッド連携フレームワークではパススルー機能を実現し，各ローカルテストベッドが備える API を連携フレームワークを通して入手し，また連携フレームワークを通して実行できるように設計した．下記がプロトタイプ実装で実装したパススルー機能を実現する API である．

```
Crspccli.DiscoverCommand()
Crspeccli.AddCommandElement(
 'urn:publicid:IDN+jgn-x.jp+command+1',
 'urn:publicid:IDN+jgn-x.jp+portinfo+1564')
Crspeccli.AddCommandArg('action',"add")
Crspeccli.AddCommandArg('isTag',"1")
Crspeccli.AddCommandArg('vid',"100")
Crspccli.ExecCommand()
```

ユーザは Crspccli.DiscoverCommand() を通してどのような公開 API が利用するテストベッドで備わっているのかを知ることができる．Crspeccli.AddCommandElement は利用する公開 API を指定するコマンドであり，Crspeccli.AddCommandArg はコマンドへの引数を登録する際に用いる．実際の実行は Crspccli.ExecCommand() によって対象となるテストベッドに実行キューが渡される．

3.4 認証スキームの実装

テストベッド API を利用する際，利用者が各種の貸し出しリソースを一つにまとめるため，申請済みのプロジェクトに対して紐付けする仕組みが必要となる．これを実現するためには，テストベッド API 利用者の特定が必要となる．これを実現するため，本調査において認証スキームのプロトタイプシステムを設計・実装した．

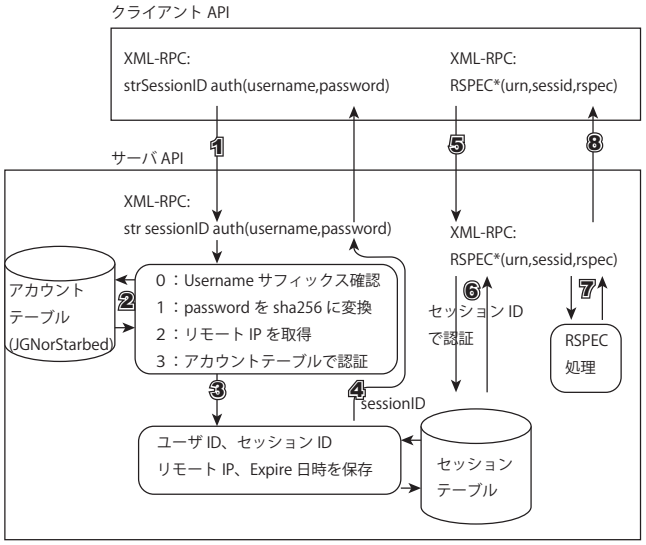


図 4 連携における認証ワークフロー
Fig. 4 Authentication work flow on the federation

図 4 は認証の手順を表わしたものである．認証方式は，現在ウェブで最も利用されるセッション ID とリモート IP アドレスを基準とした方式とした．認証は SHA265 に変換したパスワード文字列と，入力されたパスワードを SHA256 で変換したものを比較するという通常の方式を用いる．

ユーザ名とパスワードがアカウントテーブルのものと一致した場合，リモート IP とともに，セッションテーブルに記録する．以後，このセッションテーブルの情報と RSpec のリクエストに記述されたセッション ID，リモート IP で認証を行う．TCP による通信においては送信元 IP を偽造することが不可能であるため，認証済みのリモート IP とセッション ID を利用して認証することが可能である．

NAT などの環境では，送信元 IP が同一になることがある．このような環境では，セッション ID さえわかれば，認証済みのセッションを悪意のある第三者が乗っ取ることができる．本プロトタイプでは，すべてのテストベッド API 通信は SSL で暗号化されているため，セッション ID が他に漏れるということがないという前提の元の設計である．

3.5 プロトタイプ実装と課題

図 5 に，Crspeccli を通して実行したテストベッド連携の出力を示す．コマンド [1]，[2] は Python インスタンスの初期化で，コマンド [3] は認証コマンドを示す．コマンド [4] は連携テストベッドとして含まれているローカルテストベッドリストの出力である．コマンド [5] は認証されたユーザのプロジェクトと登録テストベッドの表示である．コマンド [6] は登録プロジェクトが複数存在した場合にどのプロジェクトを利用するかを明示するためのコマンドである．

コマンド [7] は addNodeElement コマンドによってローカルテストベッドに対して新たなリソースをデータベース

に追加し,，コマンド [8] は連携先テストベッドに新たなリソースを追加する要求を発行するコマンドである．

コマンド [9] は，`DiscoverRequest()` メソッドによってローカルテストベッドおよび連携テストベッドの利用可能なリソースのリストを取得している．そこでの要求 XML と応答 XML（長いので途中割愛）がコマンド [9] の下記に記された XML メッセージそれぞれに対応する．

プロトタイプの実装を通して明らかになった課題として，やはり XML 情報を利用者が直接編集するのは難しく，コマンドラインまたはウェブインターフェースなどから編集できるようにする仕組みが，テストベッド連携の実際の運用場面では必要であることが確認された．コマンドラインまたはウェブインターフェースで資源予約をサポートする実装を構築しようとすると，テストベッド連携インターフェースを通して未割り当てリソースのリストを交換する表現書式とフローを定めなければならないことも明らかとなった．また，JGN-X も StarBED も複数の拠点に実験ノードや機器を設置しているため，ユーザが明示的に利用したい拠点を表現できる拡張が考慮されていないことが，テスト利用を通して明らかとなった．

4. おわりに

本稿では，ローカルブローカー型のテストベッド連携フレームワークとして，testman テストベッド連携フレームワークを設計し，プロトタイプ実装の説明を行った．今後の課題として，別途研究開発を行っているネットワークオーケストレーションのためのテストベッド管理システムと連動させながら，StarBED - JGN-X 間での実証実験や国際連携を通して本稿で提案したローカルブローカー型テストベッド連携フレームワークの妥当性や実用性を検証していく．

謝辞 本研究は，情報通信研究機構による委託つき共同研究「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のための要素技術の研究」の一部である．

参考文献

[1] The Trustees of Princeton University: PlanetLab, The University of Southern California and Information Sciences Institute (online), available from ⟨http://www.planet-lab.org/⟩ (accessed 2013-11-30).
[2] UPMC Paris Universitas: OneLab – Future Internet Testbeds, UPMC Paris Universitas (online), available from ⟨http://www.onelab.eu/⟩ (accessed 2013-11-30).
[3] Panlab consortium: Panlab – Pan European Laboratory Infrastructure Implementation, Panlab consortium (online), available from ⟨http://www.panlab.net/⟩ (accessed 2013-11-30).
[4] FIRESTATION consortium: Future Internet Research and Experimentation – FIRE, FIRESTATION consortium (online), available from ⟨http://www.ict-fire.eu/⟩ (accessed 2013-11-30).
[5] Peterson, L., Ricci, R., Falk, A. and Chase, J.: Slice-based federation architecture version 2.0 (2010).
[6] The University of Utah: Emulab, The University of Utah (online), available from ⟨http://www.emulab.net/⟩ (accessed 2014-01-14).
[7] The University of Southern California and Information Sciences Institute: DETER Network Security Testbed, The University of Southern California and Information Sciences Institute (online), available from ⟨http://www.isi.deterlab.net/index.php3⟩ (accessed 2013-11-30).
[8] Faber, T. and Wroclawski, J.: A federated experiment environment for emulab-based testbeds, *Proc. of TridentCom 2009*, IEEE, pp. 1–10 (2009).
[9] CONET, the Cooperating Objects Network of Excellence: CONET Testbed Federation, CONET, the Cooperating Objects Network of Excellence (online), available from ⟨http://www.cooperating-objects.eu/testbed-simulation/⟩ (accessed 2013-11-30).
[10] Handziski, V., Donzelli, C. and Antonova, I.: Common Abstractions for Testbed Federation, FP7-ICT-2007-2-224053 (2009).
[11] Hermenier, F. and Ricci, R.: How to build a better Testbed: Lessons from a decade of Network Experiments on Emulab, *Proc. of TridentCom 2012*, pp. 1–10 (2012).
[12] National Institute of Information and Communications Technolgoy (NICT): Next Generation Network Testbed JGN–X.
[13] StarBED Project: StarBED Project, StarBED Project (online), available from ⟨http://www.starbed.org/⟩ (accessed 2014-01-14).
[14] Miwa, S., Suzuki, M., Hazeyama, H., Uda, S., Miyachi, T., Kadobayashi, Y. and Shinoda, Y.: Experiences in emulating 10K AS topology with massive VM multiplexing, *Proceedings of The First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures(VISA 2009)* (2009).
[15] Benzel, T., Braden, B., Faber, T., Mirkovic, J., Schwab, S., Sollins, K. and Wroclawski, J.: Current Developments in DETER Cybersecurity Testbed Technology, *Proceedings of the Cybersecurity Applications and Technology Conference For Homeland Security (CATCH 2009)* (2009).
[16] Benzel, T.: The science of cyber security experimentation: the DETER project, *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, New York, NY, USA, ACM, pp. 137–148 (online), DOI: 10.1145/2076732.2076752 (2011).
[17] Geni Project: GENI, Geni Project (online), available from ⟨http://www.geni.net/⟩ (accessed 2013-11-30).
[18] Geni Project: Resource Specification (RSpec) Documents in GENI, Geni Project (online), available from ⟨http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs⟩ (accessed 2013-11-30).
[19] RELAX NG Technical Committeee: GENI, RELAX NG home page (online), available from ⟨http://www.relaxng.org/⟩ (accessed 2013-11-30).
[20] The World Wide Web Consortium (W3C) : XML Schema, The World Wide Web Consortium (W3C) (online), available from ⟨http://www.w3.org/XML/Schema⟩ (accessed 2013-11-30).
[21] Website, N.: Networking Innovations Over Virtualized Infrastructures, Project NOVI (online), available from ⟨http://www.fp7-novi.eu/⟩ (accessed 2014-01-14).
[22] Consortium GARR: FEDERICA: Federated E-

infrastructure Dedicated to European Researchers Innovating in Computing network Architectures, FEDERICA Project (online), available from ⟨http://www.relaxng.org/⟩ (accessed 2013-11-30).

[23] Moran, M. and Mocan, A.: Towards Translating between XML and WSML based on mappings between XML Schema and an equivalent WSMO Ontology, *2nd WSMO Implementation Workshop (WIW 2005)* (2005).

[24] Bigot, P. A.: PyXB: Python XML Schema Bindings, (online), available from ⟨http://pyxb.sourceforge.net/⟩ (accessed 2014-01-14).

[25] Suzuki, M., Hazeyama, H., Miyamoto, D., Miwa, S. and Kadobayashi, Y.: Expediting Experiments across Testbeds with AnyBed: A Testbed-Independent Topology Configuration System and Its Tool Set, *IEICE Transactions on Information and System*, Vol. E92-D, No. 10, pp. 1877–1887 (2009).

[26] Chowdhury, M., Rahman, M. R. and Boutaba, R.: ViNEYard: virtual network embedding algorithms with coordinated node and link mapping, *IEEE/ACM Trans. Netw.*, Vol. 20, No. 1, pp. 206–219 (online), DOI: 10.1109/TNET.2011.2159308 (2012).

[27] Miyachi, T., Nakagawa, T., ichi Chinen, K., Miwa, S. and Shinoda, Y.: StarBED and SpringOS Architectures and Their Performance (2011).

```
In[1]:importMrspec_client
In[2]:Crspeccli=Mrspec_client.Crspec_client()

#認証
In[3]:Crspeccli.auth("admin@jgn-x.jp",
"mogemoge")
Authenticatedwithadmin@jgn-x.jpsess:
461d96d9-0cad-4817-9151-4d2f8d16fd05

#ドメインリスト表示
In[4]:Crspeccli.ShowDomain()
ID1:jgn-x.jp
ID2:starbed.org

#プロジェクトリスト表示
In[5]:Crspeccli.GetProjList()
TestbedName:jgn-x.jp
ProjectID:1
  てすとまんの研究,Research about testman

#利用するプロジェクトを明示
In[6]:Crspeccli.SetProjID(1)

#リクエストするローカルテストベッドリソースを追加
In[7]:Crspeccli.addNodeElement('note-mx80-1')
<?xmlversion="1.0"?>
<ns1:nodecomponent_uuid="urn:publicid:
IDN+jgn-x.jp+node+238"xmlns:ns1="http:
//testman.jgn-x.jp/resources/rspec/0.1"/>

#リクエストする Federation 先テストベッドリソースを
追加
In[8]:Crspeccli.addNodeElement('exsw6',2)
  #DomainID2 を第 2 引数で指定
<?xmlversion="1.0"?>
<ns1:nodecomponent_uuid="urn:publicid:
IDN+starbed.org+node+2"xmlns:ns1="http:
//testman.jgn-x.jp/resources/rspec/0.1"/>

#要求実行
In[9]:Crspeccli.DiscoverResource()
<?xmlversion="1.0"?>
<ns1:rspecgenerated="2014-1-13T15:1:17Z"type=
"manifest"valid_untel="2014-1-13T15:1:
17Z"xmlns:ns1="http://testman.jgn-x.jp/
resources/rspec/0.1">

/*****************************************
***要求 XML（ローカルリソース＋ Federation 先リソー
ス）
*****************************************/
<?xmlversion="1.0"?>
<ns1:rspecgenerated="2014-1-13T14:56:
50Z"project_id="urn:publicid:IDN+jgn-x.
jp+projid+1"valid_untel="2014-1-13T14:56:
50Z"xmlns:ns1="http://testman.jgn-x.jp/
resources/rspec/0.1"type="advertizement">
<ns1:nodelist>
<ns1:nodecomponent_uuid="urn:publicid:
IDN+jgn-x.jp+node+238"/>
<ns1:nodecomponent_uuid="urn:publicid:
IDN+starbed.org+node+2"/>
</ns1:nodelist>
</ns1:rspec>

/*****************************************
***応答 XML（ローカルリソース＋ Federation 先リソー
ス）
*****************************************/
<?xmlversion="1.0"?>
<ns1:rspecgenerated="2014-1-13T14:50:
23Z"type="manifest"valid_untel="2014-1-13T14:
50:23Z"xmlns:ns1="http://testman.jgn-x.jp/
resources/rspec/0.1">
<ns1:nodelist>
<ns1:nodecomponent_uuid="urn:publicid:
IDN+jgn-x.jp+node+238"name="note-mx80-1">

...(snip)...

</ns1:nodelist>
</ns1:rspec>
```

図 5 実際の連携における出力例

Fig. 5 Example of output of the fedeartion