SIProp

「俺流 プロトコルの実装入門」の技術的な解説

SIPropプロジェクト
代表
今村謙之（いまむらのりつな）

SIProp LLP

アジェンダ

SIProp

- 概要
  - 全体
  - ポイント
- 内容詳細
  - 仕様書
  - スタック
  - アプリケーション

概要

SIProp

■PIP（Private Implementation Protocol）という、独自のオレオレプロトコルを、定義し、実装するところまでを解説する書籍

- RFCのような仕様書の定義する方法
- RFCを、プロトコル スタック 実装に落とす方法
- プロトコル スタック を利用する方法

「俺流 プロトコル実装入門」

CPU,OS自作入門に続く、第三弾！（笑

ただし、劣化版(;^_^A

ポイント

SIProp

■RFCの理解

- RFCを策定手順に則った解説をすることにより、RFCが、どういうものかが、理解できる
- 策定したRFCを基に実装することにより、RFCを実装するという意味が、理解できる

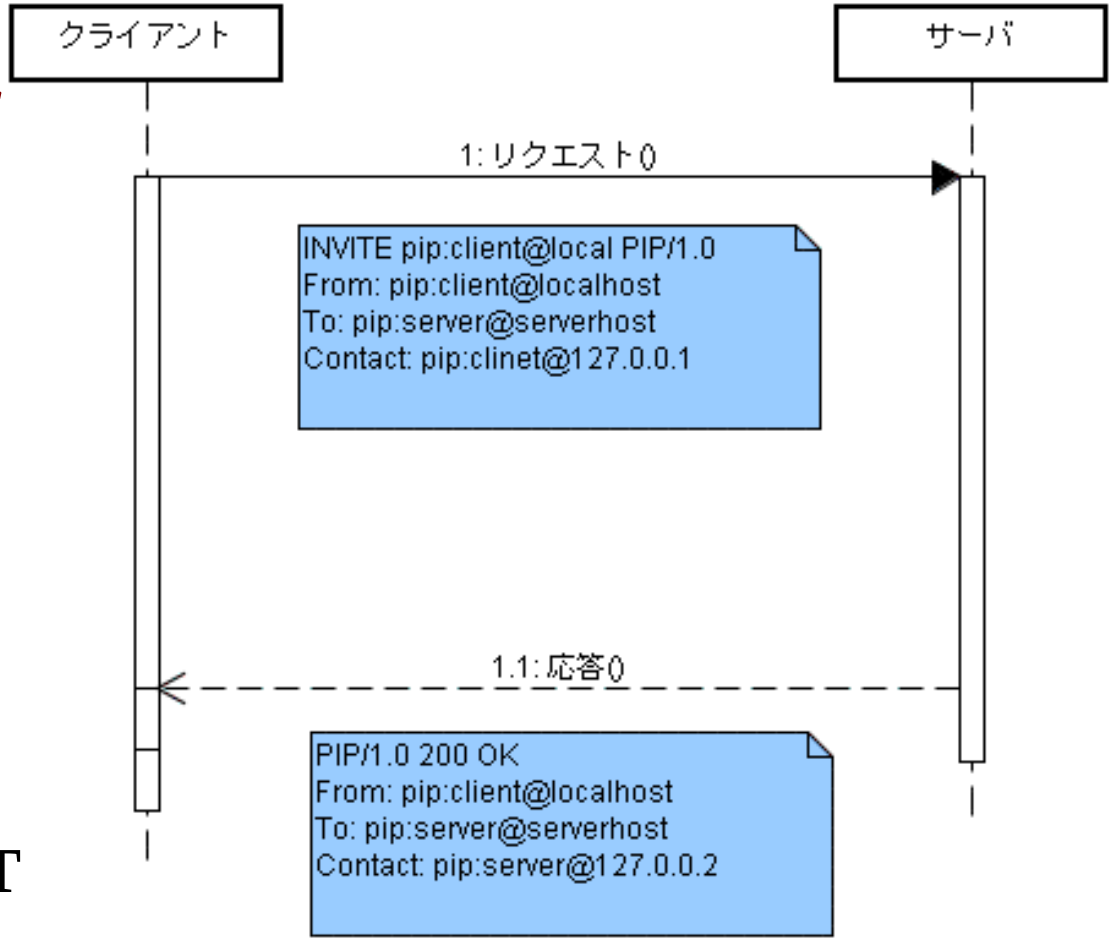
■プロトコル Stack 実装

- TCPやHTTPなどのプロトコルを肌で感じることが出来る
  - Cometのような技術的は発想の基礎を得られる

第一部・仕様書を策定する

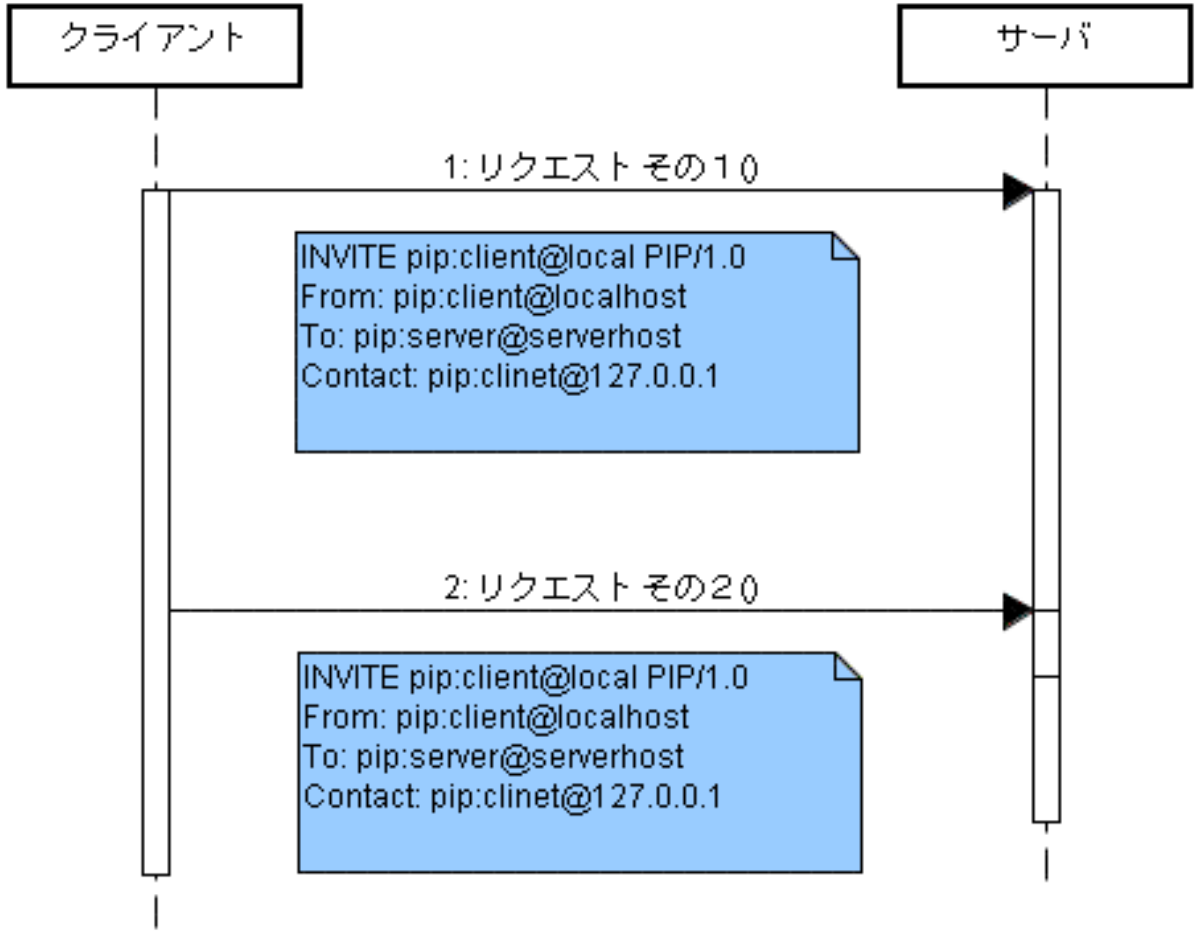
パケットの策定

- パケットの大枠の策定
  - リクエストと応答
  - メソッドと応答クラス
    - INVITE,ACK,BYE
    - 2xx,4xx
  - 個人識別
    - PIP-URI
  - ヘッダとBODY
    - TO,FROM,CONTACT



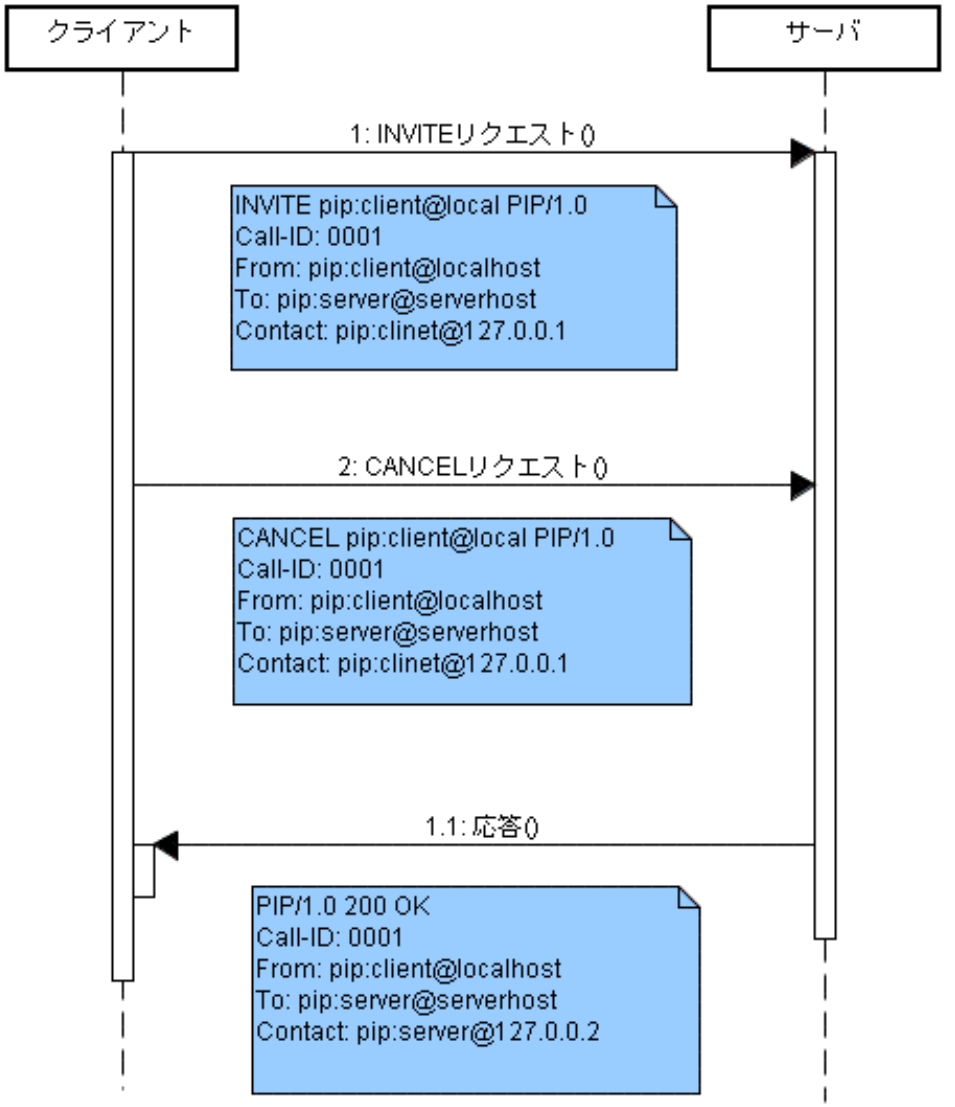
サーバ処理

- くらさば処理
  - –UACとUAS
    - サーバは、複数処理
- シーケンス
  - –Call-IDの必要性
    - Call-ID: 0001



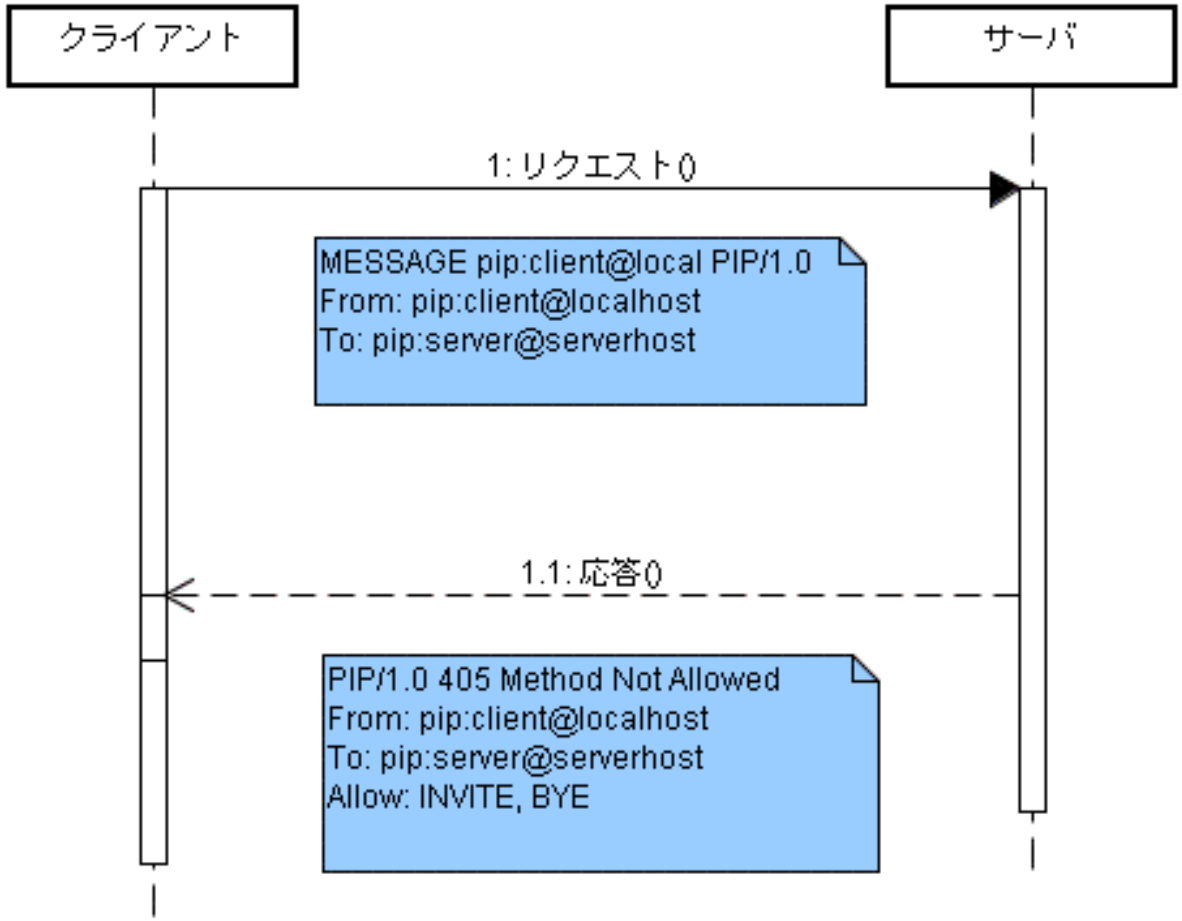
キャンセル

- 呼び出し
  - –暫定応答1xx
- CANCEL追加
  - –シーケンス番号
    - CSeq: 1 CANCEL



異常系

- 異常に対応
  - –ユーザエラー
    - 拒否
    - ビジー
  - –システムエラー
    - タイムアウト
    - トランザクションエラ
  - –能力交換
    - Allow,Accept



第二部・スタックを実装する

恐怖のパーサ編

SIProp

■構文解析とは？

- ●BNF(Backus-Naur Form)
- ●パーサの実装

```
Request-URI     =  PIP-URI / absoluteURI
      absoluteURI     =  scheme ":" ( hier-part / opaque-part )
      hier-part       =  ( net-path / abs-path ) [ "?" query ]
      net-path        =  "//" authority [ abs-path ]
      abs-path        =  "/" path-segments
      opaque-part     =  uric-no-slash *uric
      uric            =  reserved / unreserved / escaped
      uric-no-slash   =  unreserved / escaped / ";" / "?" / ":" / "@"
                         / "&" / "=" / "+" / "$" / ","
```

パーサの実装

SIProp

■パーサの実装

- ●ヘッダとボディを分ける
  - ¥r¥n¥r¥nが出てくるまで探す
- ●int phase が、3になると、break
- ●int phase は、’¥r’’¥n’が見つかるとインクリメントされる

```
int phase = 0;
while (true) {
        int n = in.read();
        if (n == -1) {
                break;
        } else if (n == '¥r') {
                if (phase == 0 || phase == 2)
                        phase++;
                else
                        phase = 1;
                bout.write('¥r');
        } else if (n == '¥n') {
                if (phase == 1)
                        phase++;
                else if (phase == 3)
                        break;
                else
                        phase = 0;
                bout.write('¥n');
        } else {
                phase = 0;
                bout.write((byte) n);
        }
}
```

説明不能のトランスポート編

SIProp

■ぶっちゃけ話

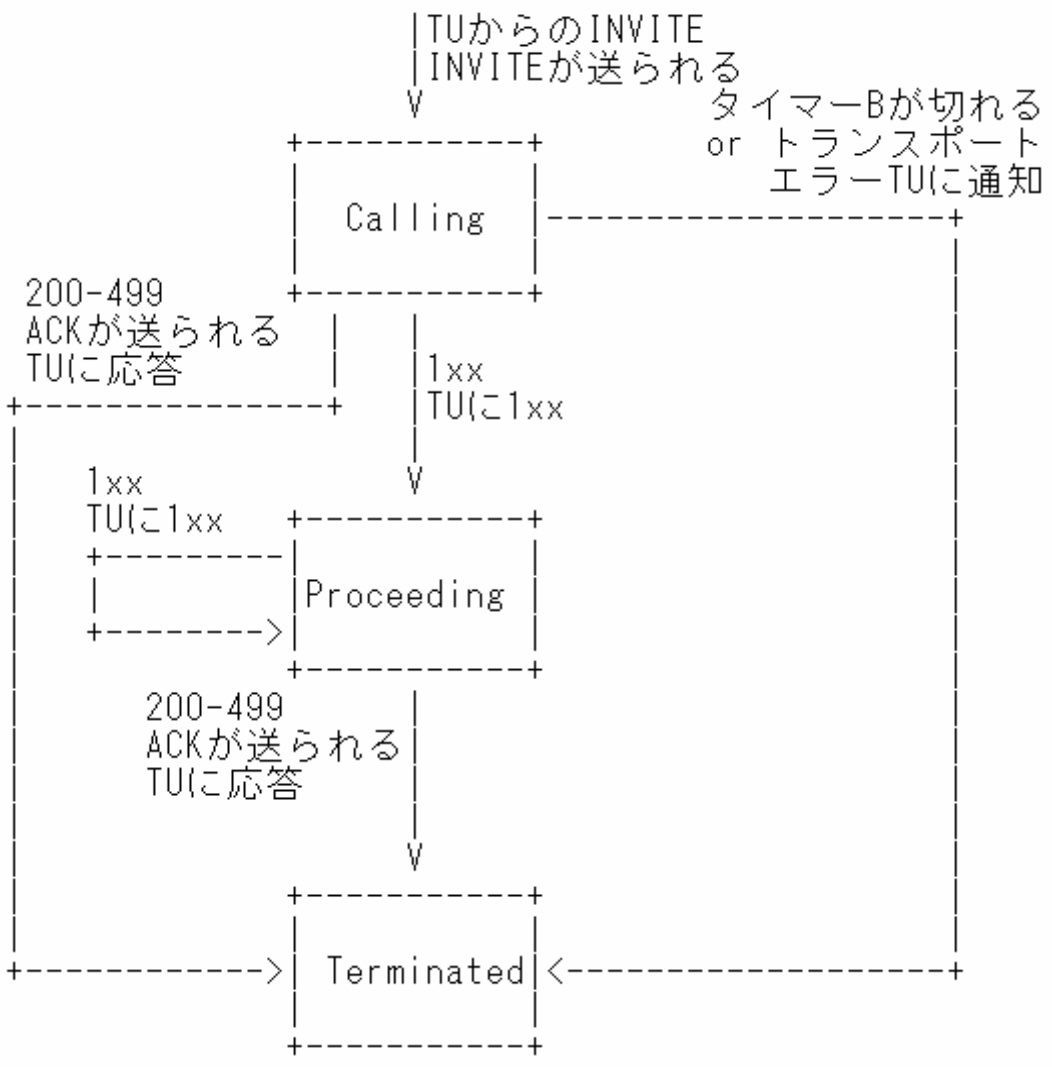
- ●OSや言語により、実装が全然違う部分
- ●PIPの適用範囲じゃない

■トランスポートトラブル

- ●スレッドとネットワーク
  - • ネットワークには、「受信」と「送信」の2スレッドが必要！
- ●パケットとストリーム
  - • パケット単位とストリーム単位がある
    - ✓図が、まだできてないのです。。。

キモのトランザクション編・ステートマシン

- ステートマシン
  - 状態遷移
    - 状態
    - トリガー
    - アクション



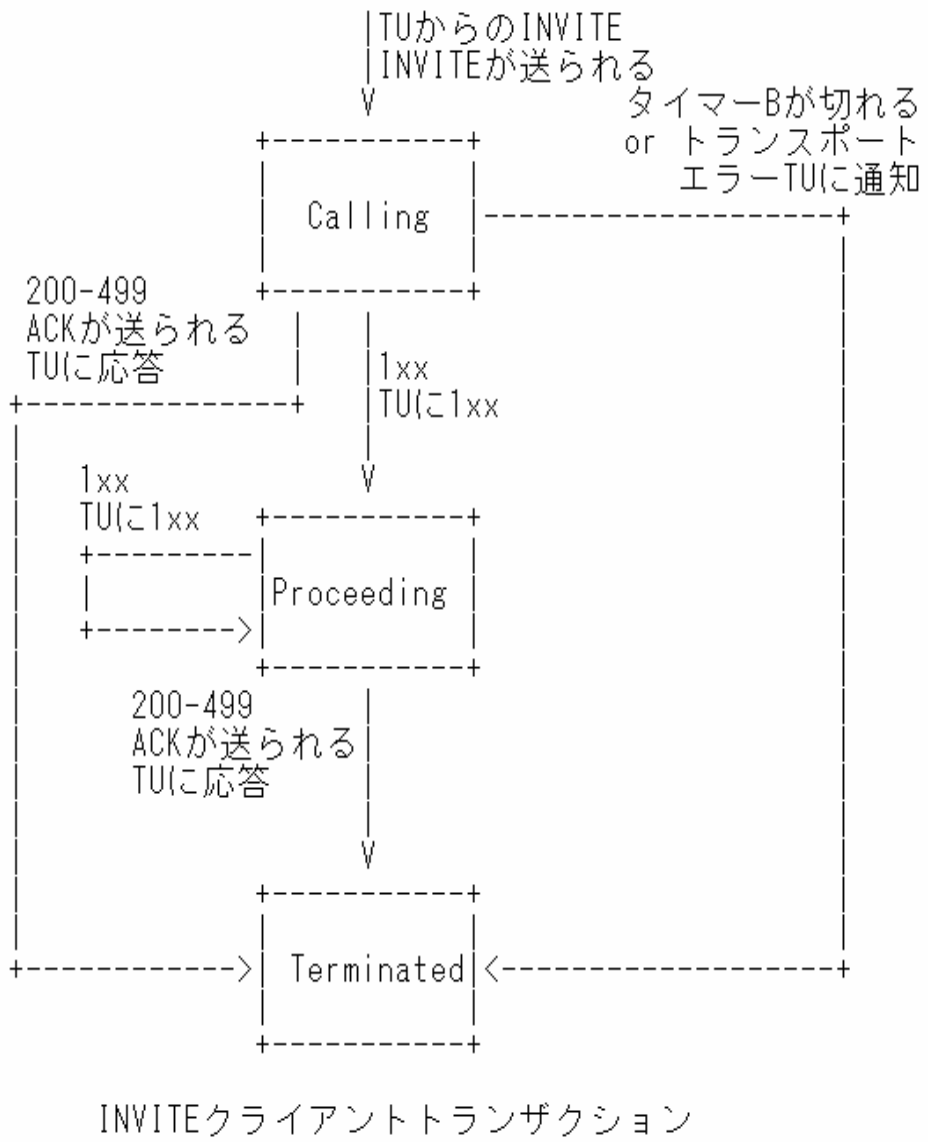
INVITEクライアントトランザクション

キモのトランザクション編・状態遷移表

- 状態遷移表
  - –状態
  - –トリガー
  - –アクション

番号	状態	トリガー(入力)	次状態	アクション(to TP)	アクション(to TU)
1	初期状態	TUからINVITE	Calling	INVITE送信	
2	Calling	タイマーBが切れる	Terminated		TUに通知
3		Transport Error	Terminated		TUに通知
4		TPから1xx	Proceeding		TUに1xx
5		TPから200-499	Terminated	ACKを返す	TUに200-499
6		*	Calling		
7	Proceeding	TPから1xx	Proceeding		TUに1xx
8		TPから200-499	Terminated	ACKを返す	TUに200-499
9	Terminated	*	*	*	*

キモのトランザクション編・ソースコード



INVITEクライアントトランザクション

```
/**
 * Calling状態を表すステートクラス
 */
private class CallingState extends TransactionState {
   /**
    * TransportからPIPメッセージを受け取った場合に呼ばれる。
    */
   public void onReceived(Transport receivedTransport,
         PIPMessage receivedPIPMessage) {
      int responseCode = receivePIPResponse.getStatusCode();
      if (100 <= responseCode && responseCode < 200) {
         // ProceedingStateに変更する
         this.transaction.changeState(new ProceedingState(
               this.transaction));

         // TUにメッセージを流す
         this.transaction.transactionListener
               .onReceivedFromTransaction(receivedPIPMessage);
      } else if (200 <= responseCode && responseCode < 500) {
         // ACKを送信する
         inviteClientTransaction.sendAck();

         // TerminatedStateに変更する
         this.transaction.changeState(new TerminatedState(
               this.transaction));

         // TUにメッセージを流す
         this.transaction.transactionListener
               .onReceivedFromTransaction(receivedPIPMessage);
      }
   }
}
```

やることないトランザクションユーザ編

■トランザクションの管理者

- ●トランザクションを管理するだけ

■他に、やることは？？？

- ●エラーを応答に変換する
  - トランスポートエラー⇒408(Request Timeout)応答
  - CANCEL処理もここにあるにはあるが。。。

意外と重要ユーザエージェント編

SIProp

■アプリケーション開発者用のAPI

- incomingINVITE()
- incoming2xxResponse()
- sendMessage()
- etc…

■実体は。。。

nRFCの[MUST]動作を実装するための存在！！！

[MUST]を満たすとは？

SIProp

- **BYEリクエスト**
  - **BYEリクエストを受け取るUASコアは、それが既存のダイアログにマッチするかどうかを確認する。BYEが既存のダイアログにマッチしない場合、UASコアは481(Call/Transaction Does Not Exist)応答を生成してそれをサーバートランザクションに渡さなくてはならない[12.1.1 UACの動作]**

```
/**
 * BYEリクエストに関する処理を行う。
 */
public void processBYE(Call call, PIPRequest byeRequest) {
   // ダイアログが無い場合は、481応答を送信する。
   if (!this.isValidDialog(call)) {
      PIPResponse errorResponse = this.createResponse(byeRequest
                  PIPResponse.CALL_OR_TRANSACTION_DOES_NOT_EXIST);
      this.sendMessage(call, errorResponse);
      return;
    }
(中略)
}
```

第三部・アプリを作成する

コンソールの実装編(第三部)

SIProp

■スタックを利用したアプリを作ると言うことは？

結局、APIの使い方でしかない！orz

ネットワークアプリケーションの秘伝

- プロトコルとは、組み合わせるものである！
  - セッション確立
    - PIP
  - セッション操作
    - SDP(Session Description Protocol)
  - セッションの中身
    - MSRP(The Message Session Relay Protocol)
    - RTP

```
INVITE pip:client@local PIP/1.0
Call-ID: 0001
CSeq: 1 INVITE
From: pip:client@localhost
To: pip:server@serverhost
Contact: pip:clinet@127.0.0.1
Content-Type: application/sdp

v=0
o=usera 2890844526 2890844527 IN IP4 alice.example.org
s= -
c=IN IP4 alice.example.org
t=0 0
m=message 7394 TCP/MSRP *
a=accept-types: text/plain
a=path:msrp://alice.example.org:7394/2s93i93idj;tcp
```

ご静聴ありがとうございました。

<(_ _)>

http://pip.siprop.org/

http://www.siprop.org/