

CPLで記述したサービスのサービス競合の分類

瀧 友和[†] 中村 匡秀^{††} 菊野 亨[†]

インターネットの発展により、VoIP (Voice over IP) を用いた新しいインターネット電話サービスが実用段階を迎えている。インターネット電話では、ユーザメイドのサービスの実現が一つの大きなトレンドであり、CPL(Call Processing Language) と呼ばれるサービス記述言語が IETF の RFC に提案されている。このようなサービスは、ユーザが自由に定義・運用できるため自由度は高いが、その反面、従来網の中央管理型のサービス運用では見られなかった新たなサービス競合問題が発生する可能性がある。本論文では、インターネット電話サービスにおけるサービス競合問題を明らかにするため、CPL で書かれたサービスを組み合わせて分析し、考えられる競合を分類する。具体的には、2 つのサービスの CPL スクリプトが同じサーバに配置された場合に焦点を絞り、CPL タグの構造に着目した 4 つの競合クラスを定義する。また、それぞれのクラスに対する競合解消のアプローチに関する考察を行う。

Feature Interaction Classification for Services in CPL

TOMOKAZU TAKI,[†] MASAHIDE NAKAMURA^{††} and TOHRU KIKUNO[†]

The recent advancement of the Internet technologies prompts the Internet telephony services for practical use. In the Internet telephony, implementation of user-made services is an important concern, and a service description language CPL (Call Processing Language) is proposed in RFC of IETF. Managing such services in the Internet telephony is up to users, thus users can freely deploy very flexible services. However, new types of the *feature interaction problem* may arise, which are not seen in the traditional centralized telephone network. In order to clarify the feature interaction in the Internet telephony, this paper categorizes functional conflicts of services written in the CPL. Specifically, we focus on a case where the CPL scripts of two services are deployed in the same server. Then, we define four classes of the interactions according to the structure of CPL tags. Also for each class, we discuss some idea towards resolution of the interactions.

1. はじめに

現在、情報通信システムの急速な発展に伴い、多種多様な通信サービスが提案され、開発されてきている。新規の通信サービスを既存の通信システムの上に追加したときに、新規のサービスと既存のサービスとの間で機能の競合が起こり、サービスとしてみると矛盾した状況に陥ることがある。これはサービス競合と呼ばれ、新しいサービスの迅速な開発を妨げる重大な障害となっている。そのため、サービス競合を検出し、サービス使用から取り除く効果的な方法を開発することが強く求められている。

近年、インターネットの発展により、ネットワーク

を介したインターネット電話が実用、応用段階を迎えている。インターネット電話のサービスには、大きく分けて 2 つのタイプがある。1 つは、Call Waiting(いわゆるキャッチホンサービス) や Camp On(受信側が電話中であった場合、一定時間を置いてつながるまで掛け直すサービス) など、端末システム上に実装する**端末サービス**で、端末システムの状態やデバイスに大きく依存する。このサービスの実装には、様々な端末ベンダによるカスタマイズが従来より行われてきた。もう 1 つは、Call Forward ビジー (電話中であった場合に転送電話を行うサービス) や、フリーダイヤルなどの**交換サービス**である。このサービスは、ユーザの位置情報や呼経路の制御、端末が電話中であった場合の処理など、従来交換器が行ってきたサービスである。従来の電話網では、交換サービスはサービスプロバイダのみにしか行うことができなかったが、インターネットのオープンアーキテクチャを生かして、プロバイダやユーザ個人が自由に交換サービスを構築で

[†] 大阪大学大学院基礎工学研究科

Graduate School of Engineering Science, Osaka University

^{††} 大阪大学サイバーメディアセンター

Cybermedia Center, Osaka University

きないか、という要求が出てきた。この要求を満たすため、インターネットサーバ上に交換サービスを記述するための言語 CPL(Call Processing Language) が提案されている。³⁾ 各個人が、CPL スクリプトをネットワークのシグナリングサーバ上に置くことで、自由に交換サービスを行うことができる。

このように、インターネット電話ではユーザ個人でサービスを展開することができるようになったが、サービス管理の分散化が起こることにより、従来の電話網では考えられなかったサービス競合¹⁾ が起こる可能性がある。また、従来の電話網では保証されていた通信の品質が、インターネット電話ではベストエフォート型となり保証されない。このことも含めて考えると、インターネット電話でのサービス競合の検出は、従来の電話網と比較してさらに困難なものになると思われる。

本研究では、インターネット電話上でのサービス競合がどのように起こるかを調べるために、CPL スクリプトで記述された交換サービスを組合せ、サービス競合がどのような場合発生し得るのかについて考察を行う。具体的には、文献⁴⁾ の 8 個のサービスの組合せに対して、サービス競合のクラスを定義し、CPL 上のシンタックスでその検出条件を定式化してみる。

2. インターネット電話

インターネット電話は、文字通りインターネット上で音声リアルタイムで伝える技術であり、従来の IN 網 (Intelligent Network) のような中央集権的なサービス管理ではなく、ユーザの制御で電話サービスを個々に管理することができる。また、インターネット電話では端末デバイスで、Call Waiting(いわゆる、キャッチホンサービス) など多くのサービスを実行することができる。しかし、呼の転送や分割、そして端末がビジーであった場合などの処理は、端末デバイスとは独立な処理であるため、ネットワークデバイスによって処理が行われる。このネットワークデバイスに依存するサービスは、従来ではサービスプロバイダに限られていた。しかし、インターネットの発展・普及により CGI などユーザが定義できるサービスが行われるようになってきた。

現在、インターネット電話における端末の相互間接続性を規定する標準プロトコルには、大きく H.323 と SIP の 2 つの流れがある。ここでは、この 2 つのプロトコルについて説明する。

H.323 インターネットの電話製品の異種間相互接続ができるように、1996 年に ITU-T(国際電気通信連合電気通信標準化部門) から採用されたインターネット電話標準規格である。従来の ISDN 用勧告である H.320 をベースに LAN 対応にした点で、H.320-H.323 ゲートウェイで音声・映像を変換符号化することなく相互接続が可能である。問題としては、呼接続時間が PSTN と比較して長いことと、拡張性に乏しいことがあげられる。

SIP IETF(Internet Engineering Task Force) から採用されたインターネット電話標準規格²⁾ である。このプロトコルはテキストベースであるため、ソフトがシンプルで低価格である。HTTP のような IP プロトコルへの準拠を強く意識しているためデータ・音声統合をより直接的に行える。

3. CPL を用いたインターネット電話サービスの記述

本研究では、SIP のネットワークモデルを元に考える。本研究で考慮するネットワークモデルは、端末システムとシグナリングサーバから構成される。(図 1 参照)

3.1 端末システム

端末システムは、主に呼やメディアを送受信するデバイスである。これは、電話機器のデバイスと共に、コンピュータ電話のクライアントや音声自動変換機なども含まれる。端末システムでは呼を送受信する他に、呼を拒否したり、転送したりすることもできる。

3.2 シグナリングサーバ

シグナリングサーバは、呼情報を制御・伝達するサーバである。ここでは、ユーザや管理者によって、自由にサービスを定義することができる。また、シグナリングサーバから複数の端末システムやシグナリングサーバに呼を同時に転送することもできる。

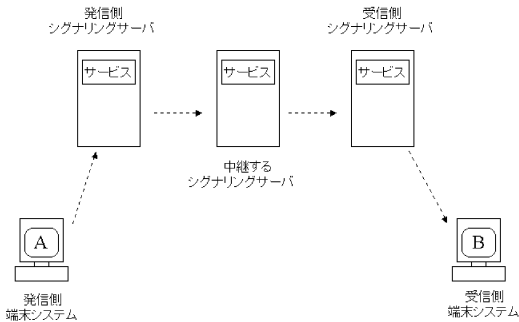


図 1 ネットワークモデル

3.3 CPL

シグナリングサーバ上のユーザ定義のサービスを規定するための言語がサービス記述言語として、XML ベースの CPL(Call Processing Language) が提案されている。この CPL 言語は、IETF の RFC2824³⁾ に提案されているものであり、特別なプロトコルやアーキテクチャは選ばない。現在のところ代表的な VoIP のためのプロトコル SIP と H.323 の両方で利用される。

本論文では、この CPL を用いて記述されたサービスのサービス競合を議論する。本節では準備として CPL のタグについて簡単に説明する。

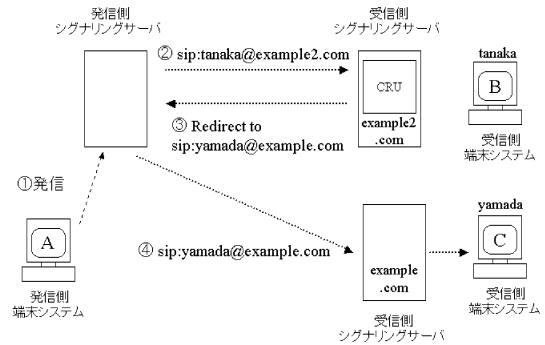
図 2 に例として、Call Redirect Unconditional(CRU)サービスの CPL スクリプトを示す。Call Redirect Unconditional とは、呼を受信した際に、転送先アドレスを発信側のシグナリングサーバに返すサービスである。これによって、発信側の呼を転送することができる。

また、図 3 に Call Forward Busy/No Answer(CF B/N) の CPL スクリプトを示す。このサービスは、呼を受信した際に、他のアドレスに呼を転送し、転送先が電話中であったり制限時間内に返答がなかった場合、留守番電話に呼を転送するというサービスである。

これらのスクリプトを用いながらタグの簡単な説明をしていく。

3.3.1 全体構造を表すタグ

CPL はシグナリングサーバ上のサービスの呼制御のシナリオを、木構造に基づいて記述する。その木の根にあたるタグが<cpl>タグであり、その子となるサブタグの代表的なものとして、<outgoing>、<incoming>、



```

1 : <?xml version="1.0" ?>
2 : <!DOCTYPE cpl PUBLIC "-//IETF//DTD
3 :         RFCxxxx CPL 1.0//EN" "cpl.dtd">
4 :
5 : <cpl>
6 :   <incoming>
7 :     <location url="sip:yamada@example.com">
8 :       <redirect />
9 :     </location>
10:   </incoming>
11: </cpl>

```

図 2 Call Redirect Unconditional

<subaction> がある。タグは<tag>で始まり、</tag>で終わる。<tag />は<tag>(空文字列)</tag>の省略形である。

outgoing 発信側のサービスを記述する際用いる。

incoming 着信側のサービスを記述する際用いる。

subaction 関数 (サブルーチン) を定義する際用いる。

3.3.2 スイッチに関わるタグ

スイッチとは、CPL スクリプトが呼情報や時間などから分岐を行うためのタグである。スイッチを扱うタグには、以下の 3 種類のタグがある。

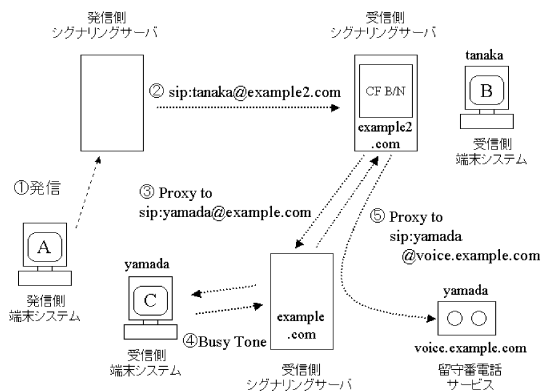
address-switch 送信側や受信側のアドレスを調べて分岐を行う。

string-switch 呼情報 (使用している言語や、アプリケーションなど) を調べて分岐を行う。

time-switch 時刻や時間から分岐を行う。

3.3.3 ロケーションに関わるタグ

ロケーションとは、呼を送信する際の送信先のアド



```

1: <?xml version="1.0" ?>
2: <!DOCTYPE cpl PUBLIC "-//IETF/DTD
3:      RFCxxxx CPL 1.0//EN" "cpl.dtd">
4:
5: <cpl>
6:   <subaction id="voicemail">
7:     <location url="sip:yamada@voice.example.com">
8:       <proxy />
9:     </location>
10:   </subaction>
11:
12:   <incoming>
13:     <location url="sip:yamada@example.com">
14:       <proxy timeout="8">
15:         <busy>
16:           <sub ref="voicemail" />
17:         </busy>
18:         <noanswer>
19:           <sub ref="voicemail" />
20:         </noanswer>
21:       </proxy>
22:     </location>
23:   </incoming>
24: </cpl>

```

図 3 Call Forward Busy/No Answer

レスのことである。CPL では、元々存在するロケーションセットに対して、増減を行う。ロケーションを扱うタグには、以下の 3 種類のタグがある。

location パラメータ"url" に記入されているロケーション先アドレスをロケーションセットに加える。

lookup パラメータ"source" に記入されているファイルから、転送先のリストを受け取り、ロケーションセットに加える。このタグは、シグナリングプロトコルに依存するため、"source" のファイルはテキスト形式や CGI ファイルなどさまざまである。

remove-location パラメータ"location" に記入されている転送先アドレスをロケーションセットから取り除く。パラメータ"location" は必須ではな

く、記入されていない場合はロケーションセットの全てのアドレスが削除される。

3.3.4 呼制御に関わるタグ

呼制御に関わるタグは以下の 3 種類である。

proxy このタグでは、呼情報を他の 1 つ以上のシグナリングサーバ、または端末システムに転送し、転送先から受け取ったレスポンスの一つを出力する。出力には、busy(電話中)、No Answer(応答がなかったことを示す)、redirection(転送先がリダイレクトを行ったことを示す) などがある。

redirect 呼情報の送信元に、別の送信先のアドレスに送るようにという情報を返す。

reject 呼情報の送信元に、呼が接続されなかったという情報を返す。

3.4 記述例

例として、図 2 の CRU と図 3 の CF B/N について説明する。

3.4.1 CRU

図 2 を基に、説明する。発信された呼が、シグナリングサーバ [example2.com] で受信された時、6 行目から 10 行目までの<incoming>タグで囲まれたシナリオが実行される。7 行目の<location>タグで、転送先のアドレスを設置する。8 行目の<redirect>タグで、発信側に転送先のアドレスを返す。これによって、発信側は呼を転送する。

3.4.2 CF B/N

図 3 を基に、転送先がビジーであるシナリオについて、説明する。発信された呼が、シグナリングサーバ [example2.com] で受信された時、12 行目から 23 行目までの<incoming>タグで囲まれたシナリオが実行される。このシナリオでは、まず受信側端末 C の [sip:yamada@example.com] に呼を転送するが、C が Busy であるので、voicemail の関数を呼び出す。すなわち、留守番電話に呼を転送する。13 行目の<location>タグで、転送先のアドレスを設置する。14 行目の<proxy>タグでタイムアウトまで 8 秒とし、呼を転送する。15 行目から 17 行目までが、転送先が busy であった時の処理、18 行目から 20 行目までが、転送先からの返呼がタイムアウトになった場合の処理である。

4. サービス競合

インターネット電話で起こり得るサービス競合は、以下のように大きく 3 つのカテゴリに分けられている。³⁾

4.1 サービス間の競合

サービス間の競合は、CPL スクリプトに記述されたサービスに対して、新しくサービスを加えようとした場合に生じる競合である。この競合は、サービスの仕様を比較した場合の競合であるため、呼の制御を行うシステムに依存しない。

図 2 に見られる Call Redirect Unconditional と図 3 に見られる Call Forward Busy/No Answer を掛け合わせた場合が例としてあげられる。この二つのサービスを同時に実行させようとした場合、呼を受け取った時に、送信元のシグナリングサーバに Redirect を行うことと、別のシグナリングサーバに呼を転送するという非決定性の処理が生じる。これが、サービス間のサービス競合の例である。

4.2 スクリプト間の競合

スクリプト間の競合は、一回の呼で複数の CPL のスクリプトを実行する場合に、起こる競合である。この競合は数多くのケースが考えられる。発信側と受信側のスクリプトが同じシグナリングサーバで実行される場合と異なるシグナリングサーバで実行される場合がまず考えられる。また、ある CPL スクリプトから転送されたアドレス先で CPL スクリプトが実行される場合、またユーザ定義の CPL スクリプト以外にサーバでサービス全体を管理する CPL スクリプトを実行する場合などが考えられる。

4.3 シグナリングサーバ間の競合

この種類のサービス競合は、スクリプト間の競合の一部であるが、実行する複数のスクリプトが別々のシグナリングサーバ上で実行される場合のサービス競合である。図 4 に見られるように Outgoing Call Screening(図 5) と Call Forward を掛け合わせた場合に起こるサービス競合が一例としてあげられる。この例では、まず発信側のシグナリングサーバでは [sip:yamada@example.com] に対して、電話を掛けることを禁止する Outgoing Call Screening サービスを行っている。そして、[sip:tanaka@example2.com] に電話を掛けた場合、受信側のシグナリングサーバでは [sip:yamada@example.com] に呼を転送するサービス

を行っている。

この場合、[sip:tanaka@example2.com] に電話を掛けることで [sip:yamada@example.com] に電話がつながってしまう。これは、発信側の Outgoing Call Screening サービスが反映されなくなるので、サービス競合となる。このシグナリングサーバ間の競合では、従来の電話網のようにサービスプロバイダがネットワーク全体のサービスの管理を行えないため、シグナリングサーバ間でサービスの状態を把握することができない。そのため、現時点ではこの種類の競合を解決することはできないとされている。

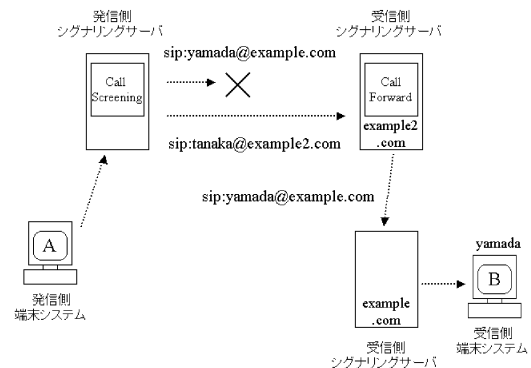


図 4 Outgoing Call Screening と Call Forward

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD
RFCxxxx CPL 1.0//EN" "cpl.dtd">

<cpl>
  <outgoing>
    <address-switch field="original-destination">
      <address is="sip:yamada@example.com">
        <reject status="reject"
          reason="Not accept allowed to call yamada"/>
      </address>
    </address-switch>
  </outgoing>
</cpl>
```

図 5 Outgoing Call Screening

5. CPL サービスの競合解析

4 章で述べたように、CPL でサービスを記述する際、色々なバリエーションの競合が考えられる。そこで、本研究では、4.1 節のサービス間の競合に焦点を絞り、2 つのサービスを同じシグナリングサーバ上で提供する際に考えられるサービス競合について議論を行う。具体的には 2 つの CPL スクリプトを同じ

サーバ上で同時に実行する際に、個々のサービスの要求機能がうまく両立するかどうかを手動で調べてみた。対象とするサービスは、文献⁴⁾にある8個のCPLサービス Call Redirect Unconditional(CRU)、Call Forward Busy/No Answer(CFB/NA)、Call Forward:Redirect and Default(CF/RD)、Call Screening(CS)、Priority and Language Routing(PL/R)、Time-of-day Routing(TR)、Location Filtering(LF)、Non-signalling Operations(NO)を用いた。

5.1 競合のクラス

ここでは、CPLのサービス競合のクラス分けを行い、例をあげた。

5.1.1 組み合わせ可能な場合 (C1)

この場合は、お互いの処理で重複する部分がないか、重複する部分の処理が同じ処理の場合であり、競合は発生せず組合せが可能である。この例として、CFB/NAとCF/RD(図6)を組み合わせた場合が挙げられる。この例では、着信後[sip:yamada@example.com]に転送されるが、CFB/NAでは、タイムアウトまたはビジーの処理を行う。一方、CF/RDでは、転送先がさらに転送されている場合、またはエラーの場合の処理を行う。従って、両者の処理における重複はないので、両サービスを同時に実行しても問題ないことがわかる。

5.1.2 呼制御タグによる競合 (C2)

この競合は、サービスを組み合わせたシナリオで、同時に呼制御タグを実行する場合に、非決定な呼制御が生じる。この例として、CRUとCFB/NAを組み合わせた場合が挙げられる。4.1節で取り上げたサービス競合が、この例である。具体的には、CRUの<redirect>とCFB/Nの<proxy>が競合する。

5.1.3 実行順序の非決定性による競合 (C3)

この競合は、複数の仕様を合わせた場合に、どちらのシナリオを先に実行するかで、複数の解釈がある場合に生じる。この例として、CSとPL/R(図7)を組み合わせた場合が挙げられる。この例でのCSは、発信者がanonymousの時、着信拒否するサービスである。一方PL/Rは、まず着信が緊急かどうかを確認し、緊急であれば着信する。緊急でなければ発信者の言語設定を確認し、スペイン語であればスペイン語のオペレータにつなぐ。それ以外の言語は、英語オペレータに接続する。この組み合わせでは、CSを先に実行すると、anonymousで発信した場合は、PL/Rが一切

行われないシナリオになる。一方、PL/Rを先に実行すると緊急電話にも関わらず、anonymousかどうか判定するため、anonymousの場合は緊急電話ができなくなるシナリオになる。このように、どちらのサービスを先に実行するかにより、シナリオが大きく変わる結果となる。

5.1.4 実行させる場所が複数ある場合の競合 (C4)

この競合は、複数の仕様を合わせた場合に、あるシナリオが複数の場所で挿入可能である場合に生じる。この例として、CRUとLF(図8)を組み合わせた場合が挙げられる。LFサービスは、条件により着信先をフィルターするものである。例えば、Inadequate Softwareというソフトにバグがあり、携帯端末に着信すると不具合が生じることがわかっていると仮定する。この例のLFサービスは、発信側の呼エージェン트가、Inadequate Softwareの該当するバージョンかどうかチェックし、該当すれば携帯端末のアドレス[me@mobile.example.com]を登録している着信デバイスの一覧(registration)から削除する。

この組合せの場合、LFのシナリオの複数の箇所でCRUを実行させることができる。例えば、発信側がInadequate Softwareを利用していない場合と、registrationのlookupが失敗した場合の2箇所でCRUサービスを実行することが可能である。その実行の場所をどう選択するかで、複数のシナリオが生じてしまう。

5.2 分析結果

同一サーバ上に複数のサービスを実行した場合のサービス競合を分類すると、表1のような結果になった。これにより、全ての組み合わせが提案する4つのクラスのどれかに分類できたことがわかる。

5.3 考察

本研究で、クラス分けされたサービス競合について、競合解消のアイデアを簡単に考察してみる。

C2 このクラスは、両サービスの呼制御が全く異なるため、同時に両方を実行することは不可能である。したがって、呼制御処理の前に優先順位をつけるため、スイッチに関するタグを直前に置くことで、両方のサービスのうちどちらかしか同時に実行できないようにするのが妥当な解決策であると思われる。

表 1 全サービスの組合わせのクラスわけ結果

	CRU	CFB/NA	CF/RD	CS	PL/R	TR	LF	NO
CRU	-	C2	C2	C1	C1	C1	C4	C1
CFB/NA	-	-	C1	C1	C4	C1	C4	C1
CF/RD	-	-	-	C1	C4	C1	C4	C1
CS	-	-	-	-	C3	C3	C3	C3
PL/R	-	-	-	-	-	C3	C3	C1
TR	-	-	-	-	-	-	C3	C1
LF	-	-	-	-	-	-	-	C4
NO	-	-	-	-	-	-	-	-

C3 このクラスは、両サービスの実行順序によってシナリオが変化する。したがって、可能な実行順序を全て考慮し、ユーザにとって一番良い組合せ順を選択するのが、解決策である。

C4 このクラスは、片方のサービスがもう片方の複数の場所で実行可能になる。従って、全ての実行可能な場所で実行させるか、ある部分のみで実行させるかにより、両サービスを組み合わせることができる。実行させる場所の選択によって、組み合わせ後のサービスのシナリオが変わる可能性があるため、ユーザの要求を考慮に入れて実行箇所の選択をするのが良いと考えられる。

6. ま と め

本研究では、CPL で書かれたインターネット電話サービスのサービス競合をクラス分けした。また、8つの例題サービスを用いてクラス分けを行い、クラスの妥当性を確認した。また、それぞれのクラスについてサービス競合を解決するための簡単なアイデアを考察した。これによって、複数の交換サービスを同時に満たす仕様を考えるための第一歩になったと考えられる。

提案した分類は、複数の CPL スクリプトが同じサーバ上に配置された場合のサービス間の競合に議論を限定したものである。今後の課題としては、サービスが複数のサーバにまたがる場合の競合クラスや、スクリプト間の競合、サーバ間の競合についても議論していく必要がある。

謝 辞

本研究の一部は、日本学術振興会科学研究費 (奨励 (A), No.13780234) の援助を受けている。

参 考 文 献

- 1) J. Lennox and H. Schulzrinne, "Feature Interaction in Internet Telephony", Proc. Feature Interaction in Telecommun. and Software Sys. VI, Glasgow, United Kingdom, May 2000.
- 2) M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol", Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.
- 3) J. Lennox and H. Schulzrinne, "Call processing language framework and requirements", Request for Comments 2824, Internet Engineering Task Force, May 2000.
- 4) J. Lennox and H. Schulzrinne, "CPL: A Language for User Control of Internet Telephony Service", Internet Engineering Task Force, May 2001.

<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <subaction id="voicemail"> <location url="sip:yamada@voice.example.com"> <proxy /> </location> </subaction> <incoming> <location url="sip:yamada@example.com"> <proxy timeout="8"> <busy> <sub ref="voicemail" /> </busy> <noanswer> <sub ref="voicemail" /> </noanswer> </proxy> </location> </incoming> </cpl> </pre>	<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <incoming> <location url="sip:yamada@example.com" > <proxy> <redirection> <redirect /> </redirection> <default> <location url="sip:yamada@voicemail.example.com"> </default> </proxy> </location> </incoming> </cpl> </pre>
---	--

図 6 Call Forward Busy/No Answer と Call Forward:Redirect and default

<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <incoming> <address-switch field="origin" subfield="user"> <address is="anonymous"> <reject status="reject" reason="Not accept anonymous calls"> </address> </address-switch> </incoming> </cpl> </pre>	<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <incoming> <priority-switch> <priority greater="urgent" /> <otherwise> <string-switch field="language"> <string contains="es"> <location url="sip:spanish@operator.example.com"> <proxy /> </location> </string> <otherwise> <location url="sip:english@operator.example.com"> </otherwise> </string-switch> </otherwise> </priority-switch> </incoming> </cpl> </pre>
--	---

図 7 Call Screening と Priority and Language Routing

<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <incoming> <location url="sip:yamada@example.com"> <redirect /> </location> </incoming> </cpl> </pre>	<pre> <?xml version="1.0" ?> <!DOCTYPE cpl PUBLIC "-//IETF/DTD RFCxxxx CPL 1.0/EN" "cpl.dtd"> <cpl> <incoming> <string-switch field="user-agent"> <string is="Inadequate Software SIP User Agent/0.9"> <lookup source="registration" ignore="feature"> <success> <remove-location location="sip:me@mobile.example.com"> </remove-location> </success> </lookup> </string> </string-switch> </incoming> </cpl> </pre>
--	---

図 8 Call Redirect Unconditional と Location Filtering