TTC標準
Standard

JF-IETF-RFC5407

SIP における準正常状態のコールフロー例

Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP)

第 1.1 版

2009 年 11 月 17 日制定

社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目 次

＜参考＞

1. 国際勧告等との関係

本標準は、IETF において制定された RFC5407 に準拠している。

IETF RFC5407 の日本語翻訳および補足説明を記述している。ただし、RFC5407 を拡張するものではない。

2. 上記国際勧告等に対する追加項目等

2.1. オプション選択項目

特になし

2.2. ナショナルマター項目

特になし

2.3. 原標準に対する変更項目

特になし

2.4. RFC5407 との章立ての構成比較表

RFC5407 との章立ての構成の相違を下表に示す。

RFC5407	本標準
－	<参考> 1.概要 2.本標準で規定する内容 3.本標準の内容 4.本標準の記述形式について
State of this Memo Copyright Notice Abstract Table of Contents	5.RFC5407 の前文
1.Overview	5.1.概観
2.The Dialog State Machine for INVITE Dialog Usage	5.2.INVITE Dialog Usage の状態遷移機構
3. Race Conditions	5.3.準正常
4. Security Considerations	5.4.セキュリティ上の考慮点
5. Acknowledgements	5.5.謝辞
6. References	5.6.参照文献
Appendix A. BYE in the Early Dialog	付録 A.アーリーダイアログ内の BYE
Appendix B. BYE Request Overlapping with re-INVITE	付録 B. BYE リクエストと re-INVITE のオーバーラップ
Appendix C. UA's Behavior for CANCEL	付録 C. CANCEL に対する UA の動作
Appendix D. Notes on the Request in the Mortal State	付録 D. Mortal 状態におけるリクエストに関する注釈
Appendix E. Forking and Receiving New To Tags	付録 E. フォーキングと新規 To タグの受信

3. 改版の履歴

版数	制定日	改版内容
第 1.0 版	2009 年 5 月 27 日	制定
第 1.1 版	2009 年 11 月 17 日	日本語翻訳および補足説明を追記

4. 工業所有権

TTC の「工業所有権等の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページで公開されている。

5. その他

(1) 参照する主な勧告、標準

[1] " Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP) ", RFC5407, December 2008

その他RFC5407[1] 内で参照している勧告、標準については本文 5.6 参照文献にて記述している。

(2) 本出版の規定はすべて準拠元である IETF RFC によっている。

6. 標準作成部門

信号制御専門委員会

1　標準の概要

このドキュメントは SIP における準正常状態のコールフロー例を提供する。準正常は本質的に難解で防ぐことが難しい。このドキュメントではそれら準正常状態の最適な扱い方を示す。コールフローの要素として、SIP User Agent と SIP Proxy Server を含む。コールフローダイアグラムとメッセージ詳細を示す。

2　本標準で規定する内容

本標準で規定する内容は下記の IETF RFC による。

IETF RFC5407 : 「Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP)」

3　本標準の内容

本標準の記述する内容は以下の通りである。

本文：RFC5407[1]の日本語翻訳

注釈：RFC5407[1]の補足説明

4　本標準の記述形式について

本標準では、RFC の日本語翻訳を本文に記載し、本文への注釈を欄外に挿入する記述形式とする。

5　RFC5407 の前文

Network Working Group
Request for Comments: 5407
BCP: 147
Category: Best Current Practice

M. Hasebe
J. Koshiko
NTT-east Corporation
Y. Suzuki
NTT Corporation
T. Yoshikawa
NTT-east Corporation
P. Kyzivat
Cisco Systems, Inc.

December 2008

Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP)

Status of This Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements.　Distribution of this memo is unlimited.

Abstract

This document gives example call flows of race conditions in the Session Initiation Protocol (SIP).　Race conditions are inherently confusing and difficult to thwart; this document shows the best practices to handle them.　The elements in these call flows include SIP User Agents and SIP Proxy Servers.　Call flow diagrams and message details are given.

Network Working Group
Request for Comments: 5407
BCP: 147
Category: Best Current Practice

M. Hasebe
J. Koshiko
NTT-east Corporation
Y. Suzuki
NTT Corporation
T. Yoshikawa
NTT-east Corporation
P. Kyzivat
Cisco Systems, Inc.

December 2008

SIP における準正常状態のコールフロー例
Example Call Flows of Race Conditions in the Session Initiation Protocol (SIP)

本文章の位置づけ

本文書は、インターネットコミュニティのためのインターネット標準トラックプロトコルを規定するものであり、改善のための議論や提案を依頼するものである。本文書の配布は無制限である。

著作権表記

要約

本文書は Session Initiation Protocol (SIP)における準正常のコールフロー例を示す。準正常は理解が難しく解決が困難である。本文書は準正常を取り扱う上でのベストプラクティスを示す。SIP ユーザエージェント及び SIP プロキシサーバがコールフローに含まれるエレメントである。本文書にはコールフロー図、及びメッセージ詳細が示されている。

5.1　概観

本文書で示されているコールフローは、SIP IP 通信ネットワークの構築の過程で検討された。示されている例は準正常コールフローであり、主として INVITE 送信後のセッション確立段階におけるダイアログの状態遷移に起因する。

SIP の実装の際には、様々な難しい状況が生じると想定される。従って、端末やサーバの推奨動作例を提示することが、SIP 実装者にとって有益であろう。

本文書では、メッセージが交差する準正常における SIP User Agent (UA)の動作を明確化する。準正常における処理を明確化することで、実装間における解釈の不一致を避け、相互接続性を高める事が期待される。

本文書が、SIP実装者、検討者、及びプロトコル研究者にとって有益、かつRFC 3261[1]の標準実装という目標の達成に役立てば本意である。

コールフローはRFC 3261[1]で規定されるSIPバージョン 2.0 に、SDPの使用はRFC 3264[2]に従う。

本文書における以下の用語、「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「MAY」、及び「OPTIONAL」の解釈は、BCP 14 であるRFC 2119[3]に従うこととする。

5.1.1 全般的想定事項

本文書のコールフローでは、アーキテクチャ、ネットワーク及びプロトコル上多くの想定事項が存在する。これらの想定事項は要求条件ではない。本節では、コールフロー例の理解のための考慮点としてそれら想定事項を概説する。

フローは、TCP、TLS、またはUDPといった特定のトランスポートプロトコルは想定しない。SIPにおけるトランスポートについてはRFC 3261[1]を参照のこと。

5.1.2 メッセージフロー凡例

点線(---)、及び斜線(/, \ )は、コールシナリオに必須の信号メッセージを表す。(X)は信号メッセージの交差を示す。(->x, x<-)はパケットロスを表す。矢印はメッセージフローの方向を示す。2 重点線はネットワークエレメント間のメディアパスを表す。

メッセージは F1、F2 といった記号で表す。これらの番号は、フロー図と、続くメッセージ詳細で参照するために用いられる。メッセージ詳細中のコメントは以下の形式にて表す。

/* Comments.　*/

5.1.3 SIPプロトコルに関する想定事項

本文書は、フローに関する詳細な指示はせず、ベストプラクティスの原則を表す。これは SIP メソッド、ヘッダ、及びパラメータのベストプラクティス上の使用法(順序、構文、ある目的のためのフィーチャの選択、またはエラー処理)である。

注) 解説上の記述が追加されているため、実装者は本文書のフローをそのままコピーしてはならない。

本文書に記述されている手順は、IETF における合意に基づく一般的なベストプラクティスを表す十分に検討され

た SIP 使用例である。

解釈や編集の容易さの観点から、メッセージ例と実際の SIP メッセージには種々の差異がある。例えば、同じ Call-ID 値が繰り返し利用される、CSeq が 1 から開始する、ヘッダフィールドの順序が同じである、最小限のヘッダフィールドしか示されていない、Accept、Allow など通常含まれる他のヘッダが省略されているなどである。

アクター

エレメント	ディスプレイネーム	URI	IP アドレス
-------	----------	---	----------
User Agent	Alice	sip:alice@atlanta.example.com	192.0.2.101
User Agent	Bob	sip:bob@biloxi.example.com	192.0.2.201
User Agent	Carol	sip:carol@chicago.example.com	192.0.2.202
Proxy Server		ss.atlanta.example.com	192.0.2.111

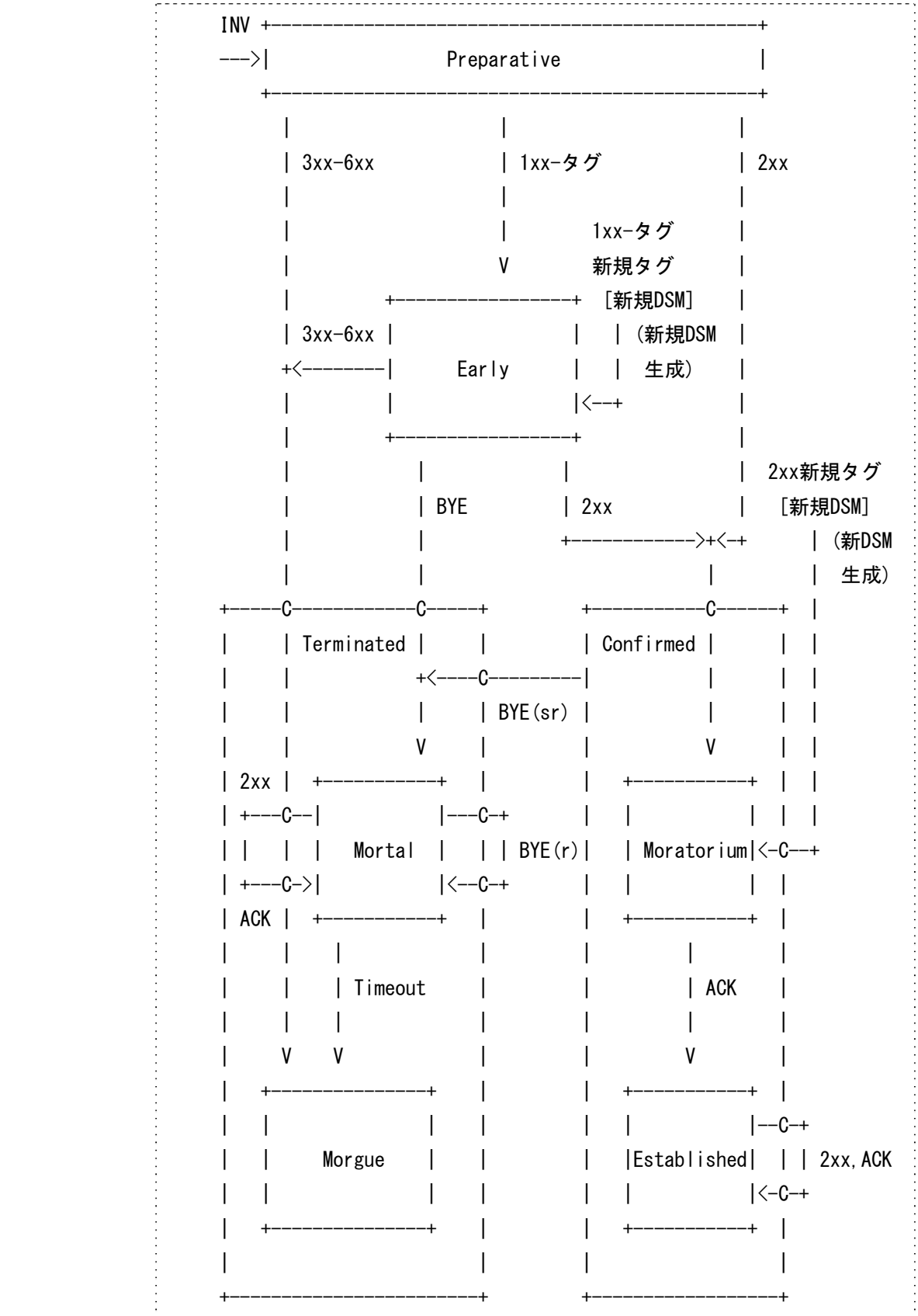
本文書の「セッション」は、RFC 3261[1]の 13 章から 15 章と同様な用法で用いられる(RFC 3261 での定義とは多少異なる)。RFC 5057[6]では、さらに明確に定義された「Invite dialog usage」という用語が導入されている。本文書における「セッション」は「Invite dialog usage」とほぼ同義であるが、全く同一ではない。これらは、状態が終了する時期の定義が異なる。セッションの方が早く終了し、その契機はBYEの送信または受信である。

5.2 INVITE Dialog Usageの状態遷移機構

準正常は、送信側と受信側でダイアログ状態に差異が出ることから発生する。

例えば、準正常は、UAS (User Agent Server)がイニシャル INVITE (以降「ini-INVITE」と表記する)に対する 200 OK を送信して Early 状態から Confirmed 状態に遷移する間に、UAC (User Agent Client) が Early 状態にて CANCEL を送信する場合に発生する。準正常における UA 動作の理解のために、INVITE dialog usage の DSM (Dialog state machine)を以下に示す。

ここで示されるDSMでは、RFC 3261[1]によるダイアログ状態を種々の内部状態に細分化することでUAの動作を明確化している。ダイアログ確立前の状態をPreparative状態と呼ぶ。Confirmed状態はMoratorium及びEstablished状態の 2 つの内部状態に分けられる。Terminated状態はMotal及びMorgue状態に分けられる。これら状態の遷移の契機となるメッセージを矢印で示す。図内では、状態遷移に無関係なメッセージは省略している。始めに発信側、次いで着信側のDSMを以下に示す。



(r)：受信のみ可能であることを示す。(r)が示されていない場合、
「応答」は受信を、「リクエスト」は送信を示す。
(sr)：送受信可能であることを示す。

図 1：INVITE dialog usage の DSM (発信側)

図 1 はINVITE dialog usageにおける発信側DSMを表す。推奨されていないが、発信者はEarly状態においてBYEを送信しても良い[1]。Early状態で送信されたBYEは、特定のToタグを利用するアーリーダイアログを終了する。つまり、プロキシがフォーキングを行なっている場合、BYEは特定のUAとのアーリーダイアログのみ終了可能である。発信者が、ある特定のUAとのアーリーダイアログだけではなく全てのアーリーダイアログを終了したい場合、BYEではなくCANCELを送信する必要がある。しかし、特定のアーリーダイアログを終了することが発信者の目的である場合、Early状態にてBYEを送信することは違反ではない。さらに、発信者が最終応答を受信しINVITEトランザクションを終了するまでは、例え既にCANCELまたはBYEを送信しダイアログを終了していたとしても、発信者はINVITEに対する新規応答の受信によるダイアログ確立に備えなくてはならない(MUST)(付録 A参照)。

[1]発側は、フォーキングを期待した発呼を行ない、かつ、ある特定の UA との Early ダイアログを終了する可能性がある場合に限り Early ダイアログ上での BYE 送信を考慮し、それ以外の場合は CANCEL を送信する。

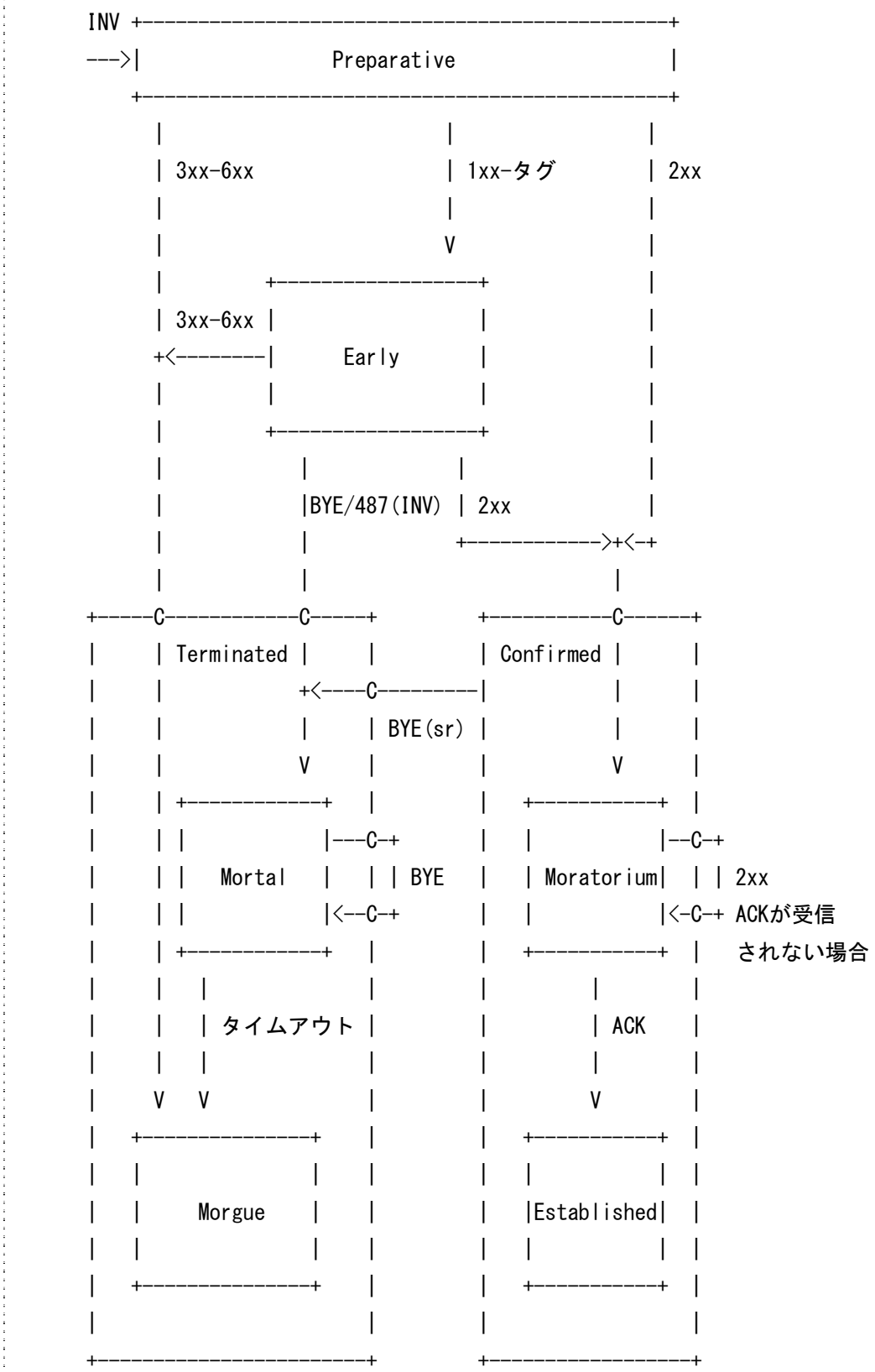


図2：INVITE dialog usageのDSM (受信側)

図2はINVITE dialog usageにおける着信側DSMを表す。図ではCANCELに関連した状態遷移を示していない。CANCELリクエストはダイアログ状態遷移の契機とはならない。しかし、着信側はCANCELの受信直後に487を送信することにより、ダイアログ状態遷移を始動しダイアログを終了する。CANCELリクエスト受信時の動作についての詳細説明は付録 Cを参照。

それぞれの状態におけるUAの動作は以下の通り。

Preparative (Pre) : Preparative状態は、ini-INVITEの送信または受信後にToタグが付与された暫定応答を受信または送信することによりアーリーダイアログが確立されるまでの状態である。Preparative状態においてダイアログはまだ存在しない。UAが2xx応答を送信または受信するとダイアログ状態はPreparative状態からConfirmed状態の内部状態であるMoratorium状態へ遷移する。UAが3xx-6xx応答を送信または受信すると、ダイアログ状態はTerminated状態の内部状態であるMorgue状態へ遷移する。3xx-6xx応答に対するACK及び3xx-6xxの再送は、INVITEトランザクション上で送信されるため、DSMには示されていない。

Early (Ear)：アーリーダイアログは、100 Trying応答を除く暫定応答の送信または受信により確立される。この状態においてダイアログは存在しないが、アーリーダイアログは存在する。ダイアログ状態は、2xx応答の送信または受信により、Early状態からConfirmed状態の内部状態であるMoratorium状態に遷移する。また、3xx-6xx応答の送信または受信により、ダイアログ状態はTerminated状態の内部状態であるMorgue状態へ遷移する。3xx-6xx応答に対するACK及び3xx-6xxの再送は、トランザクションレイヤで自動的に処理されること、およびダイアログ状態には影響を与えないことから、DSMには示されていない。UACはEarly状態においてCANCELを送信可能である。また、推奨動作ではないが、UACはBYEも送信可能である。UASは1xx-6xx応答を送信可能である。CANCELリクエストの送受信はダイアログ状態に直接的な影響を及ぼさない。
CANCELリクエスト受信時のUA動作についての詳細説明は付録 Cを参照。

Confirmed (Con)：2xx最終応答の送信または受信の結果ダイアログが確立する。ダイアログはこの状態で開始する。BYEリクエストの送信または受信により、Confirmed状態はTerminated状態の内部状態であるMortal状態に遷移する。Confirmed状態は、Moratorium及びEstablishedの2つの内部状態を持つ。これら二つの内部状態では、UAが送信可能なメッセージが異なっている。

Moratorium (Mora) : Moratorium状態はConfirmed状態の内部状態であり、Confirmed状態同様の動作を行なう。ACKリクエストの送信または受信により、Moratorium状態はEstablished状態に遷移する。UACはACKの送信、UASは2xx最終応答の送信が可能である。

Established (Est)：Established状態はConfirmed状態の内部状態であり、Confirmed状態同様の動作を行なう。発信側、受信側共に、ダイアログに影響を及ぼす種々のメッセージを送信可能である。発信者は、ini-INVITEに対する2xx応答の再送に対応するACKの送信をサポートする。

Terminated (Ter)：Terminated状態はダイアログ終了動作に対応するために、Mortal及びMorgueの2つの内部状態を持つ。この状態においてUAは、終了対象のダイアログに関する情報を保持している。

Mortal (Mort) : 発信側及び受信側は、BYEの送信または受信によりMortal状態に遷移する。ダイアログが存在しないため、UAはこのダイアログ内での新規リクエストを送信してはならない(MUST NOT) (ここでの新規リクエストは、2xxに対するACK、及び401、407に対するBYEを除く。詳細は付録 Dを参照)。Mortal状態に

おいてBYEは受け入れ可能だが、INVITE dialog usageにおけるその他のメッセージに対してはエラー応答処理となる。Mortal上でBYEを受け入れ可能であることは、発信側と着信側でセッション終了に関する情報を交換するケースに対処している。従って、UAは内部処理のためにダイアログ情報を保持するが、外部からダイアログが存在するように見えるべきではない。UAはBYEトランザクションの終了を契機としてダイアログ状態の管理を終了し、Morgue状態に遷移する。

Morgue (Morg) : この状態ではもはやダイアログは存在しない。ダイアログに影響を及ぼす信号の送受信は行われない。(ダイアログは事実上終了している。) 発信側及び受信側は BYE または INVITE トランザクションの終了により Morgue 状態に遷移する。

5.3 準正常

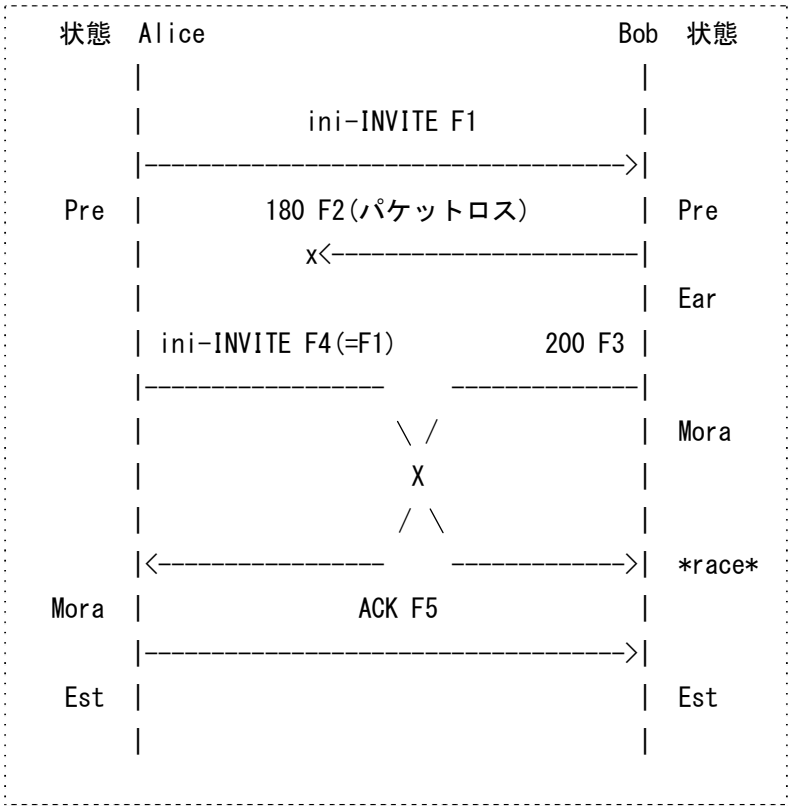
本章では Alice と Bob の 2 つの SIP UA 間での準正常を詳解する。Alice (sip:alice@atlanta.example.com)及び Bob (sip:bob@biloxi.example.com)は SIP 電話または SIP 対応機器であることを想定とする。重要な信号のみ記述する。SIP メッセージの送信または受信によるダイアログ状態遷移も記し、準正常は「*race*」で示す。(ダイアログ状態遷移の略称については 5.2 章を参照。) 「*race*」は準正常発生時点を表す。

準正常の例を以下に示す。

5.3.1 Moratorium状態でのメッセージ受信

本節では、Moratorium 状態において他の状態から送信されたメッセージを受信した際に発生する準正常のコールフロー例を示す。

5.3.1.1　着信側がMoratorium状態でInitial INVITEの再送(Preparative状態)を受信する場合



本シナリオはUASがMoratorium状態でPreparative状態のメッセージを受信した際に生じる準正常を示す。Initial INVITE (ini-INVITE F1)に対する暫定応答は全て失われ、UACはini-INVITE (F4)を再送する。この再送と同じタイミングで、UASはRFC 3261[1]の 13.3.1.4 節に従いini-INVITEに対する 200 OK (F3)を生成しINVITEサーバトランザクションを終了する。

しかし、200 OKの送信によるINVITEサーバトランザクションの終了については、SIPにおける重要修正事項として指摘されている [7]。このため、INVITEサーバトランザクションはF3 で終了せず、F4 は再送として適切に処理されなくてはならない(MUST)。

RFC 3261[1]では、ini-INVITEの再送に対しUASが 200 を再送することについて規定していない。RFC 3261[1]の 13.3.1.4 節によるタイマーを契機とした 200 の再送(トランザクションユーザ(TU)は、ACKを受信するまでT1 及びT2 に従い 200 の再送を継続する)を考慮し、再送されたini-INVITEを受信する度にUASが 200 を再送することは不要と考えられる。(再送の 200 は何も問題を引き起こさないため、実装上UASが 200 再送を送信しても問題はない。)

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

/*180 応答は失われ、Alice まで届かない*/

F3 200 OK Bob -> Alice

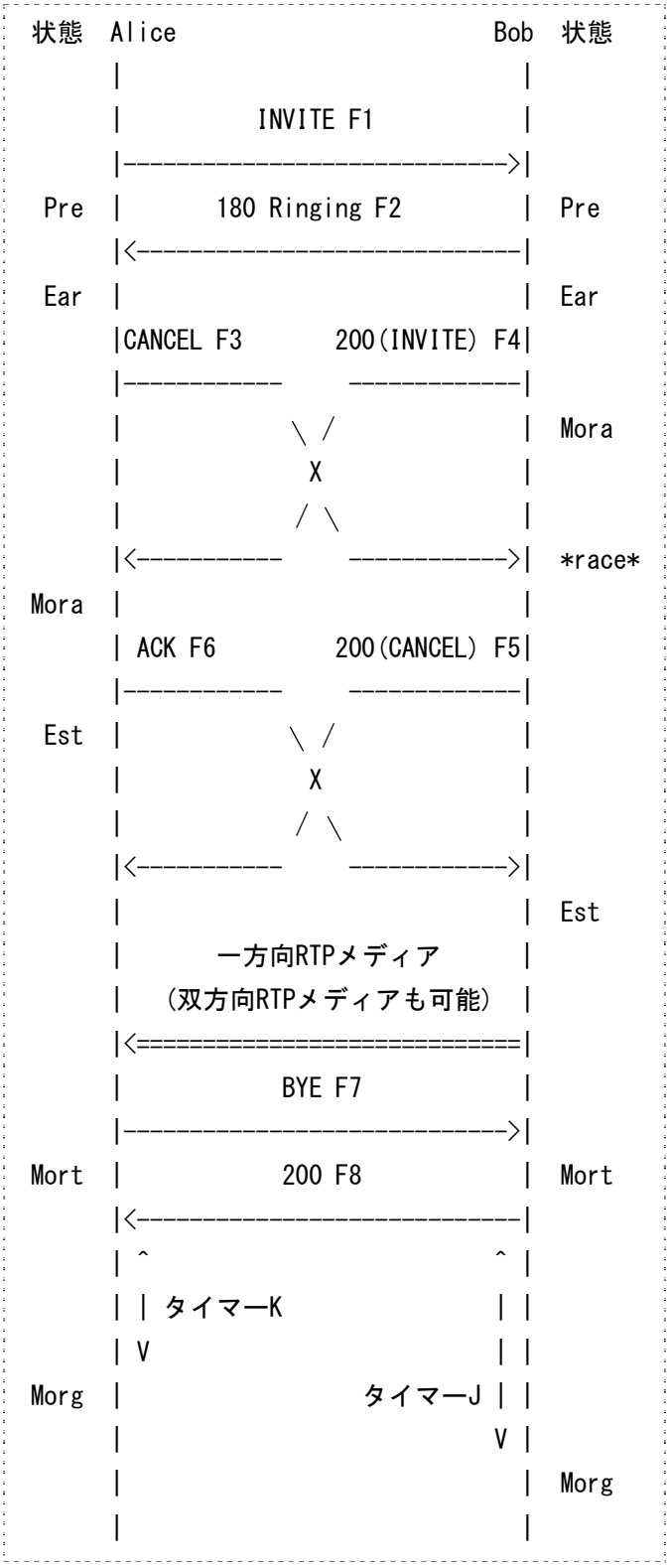
/* RFC 3261[1]の 13.3.1.4 節により、この時点でINVITEサーバトランザクションは終了する。しかし、本動作はSIPにおける重要修正事項として指摘されており、UASはini-INVITE (F4)を再送として適切に認識しなくてはならない(MUST)*/

F4 INVITE (再送) Alice -> Bob

/*F4 はF1 の再送であり、全く同一のINVTEリクエストである。[7]の修正に対応していないUA(INVITEサーバトランザクションがINVITEへの 200 送信により終了するUA)にとって、このリクエストはTo tagを持たないためトランザクションにもダイアログにもマッチしない。しかし、Bobはこのリクエストを新規INVITEではなく、INVITEの再送であると正しく認識しなくてはならない。*/

F5 ACK Alice -> Bob

5.3.1.2　着信側がMoratorium状態でCANCEL (Early状態)を受信する場合



本シナリオは、UASがMoratorium状態においてEarly状態のメッセージであるCANCELを受信した際に生じる準正常を示す。AliceがCANCELを送信するのと同時にBobがInitial INVITEメッセージに対する 200 OK応答を送信する。前節でも説明したように、RFC 3261[1]によるINVITEサーバトランザクションが 200 応答で終了する動作は [7]で修正されている。

本節ではINVITEサーバトランザクションがINVITEに対する 200 応答で終了しない場合について説明する。この場合、CANCELリクエストの対象であるINVITEトランザクションが存在するため、CANCELリクエストに対し 200 応答を送信する。この 200 応答は、単に次ホップがCANCELリクエストを受信したことを意味している(CANCELの成功(200)はINVITEがキャンセルされたことを示すものではない)。UASが [7]による修正に対応していない場合、CANCELリクエストがマッチするトランザクションが存在しないため、UACはCANCELに対する 481 応答を受信するかもしれない(MAY)。この 481 はCANCELがマッチするトランザクションが存在せず、CANCELは次ホップに送信されないことを単に示している。CANCELの成功/不成功に関係なくAliceはINVITEに対する最終応答を確認し、INVITEリクエストに対して 200 を受信した場合には直ちにBYEを送信しダイアログを終了する。(RFC 3261[1]の 15 章を参照。)[2]

Bob が F1 を受信してから F8 を送信するまでの間、Bob から Alice へ片方向メディアが流れる可能性がある。また、Alice が Bob からのアンサーを受信した時点から、Alice から Bob へメディアが流れる可能性もある。しかし、Alice が一旦呼のキャンセルを決めるとメディアを送信しないと想定されるため、実際には通話のメディアストリームは片方向のみとなる。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 CANCEL Alice -> Bob

/*Alice が Early 状態で CANCEL を送信する*/

F4 200 OK (INVITE) Bob -> Alice

/*Alice が Moratorium 状態で INVITE (F1)に対する 200 を受信する。Alice はメディアを送受信する能力を潜在的に持っているが、呼を終了する意思があるため実際は送信しないであろう。*/

F5 200 OK (CANCEL) Bob -> Alice

/*CANCELに対する 200 は単にCANCELが受信されたことを意味する。本シナリオでは [7]による修正に対応していることを想定しているため、この 200 応答が送信される。RFC 3261[1]の手順に従ってINVITEサーバトランザクションが終了された場合、UACは 200 ではなく 481 応答を受信するかもしれない。*/

F6 ACK Alice -> Bob

---

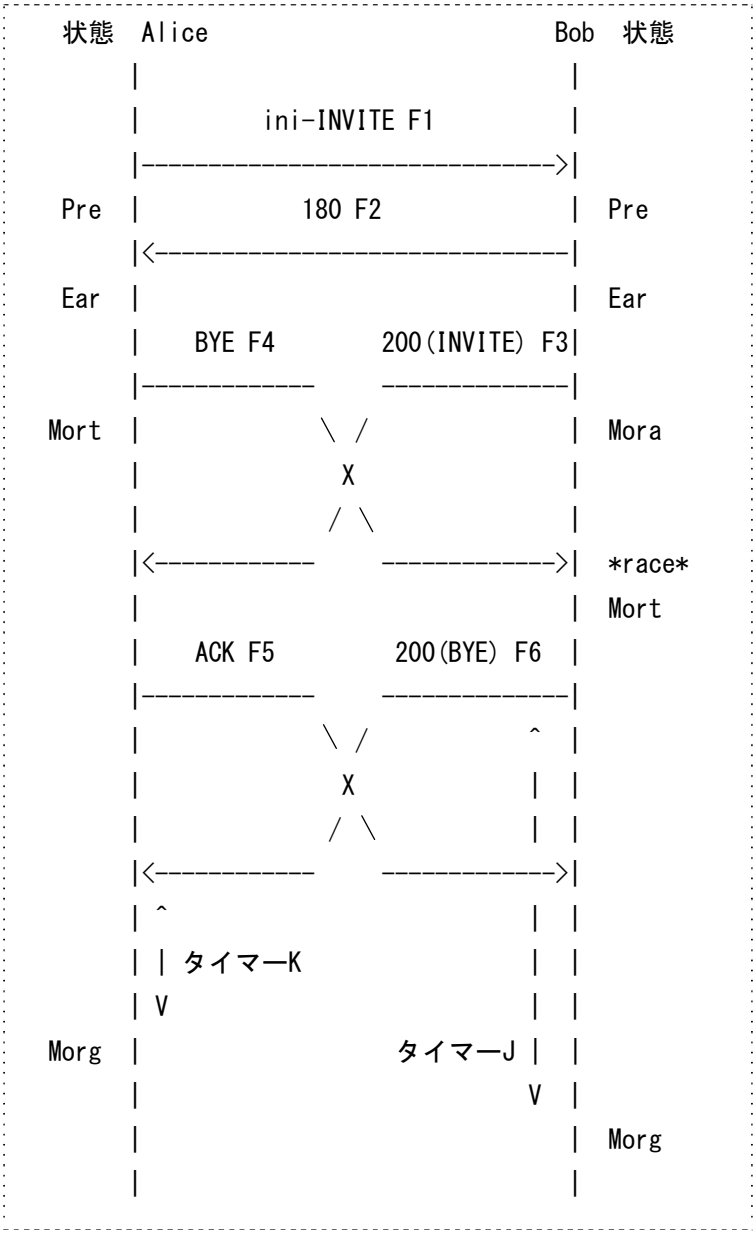
[2] RFC3261 の 15 章では、UAC は CANCEL 後に 2xx 応答で確立されたダイアログを継続することも終了することもできると記載されており、また、CANCEL が「電話を切る(haging up)」に相当するかどうかは特定のユーザーインターフェースに固有のものであると記載されている。本例では、主な用途として電話のようなユーザーインターフェースを想定しているため、BYE によりダイアログを終了している。

/*INVITE は成功し、CANCEL は無効となる。Bob は RTP ストリームを確立する。しかし、後の BYE が直ちにダイアログ及びセッションを終了する。*/

F7 BYE Alice -> Bob

F8 200 OK Bob -> Alice

5.3.1.3 着信側がMoratorium状態でBYE (Early状態)を受信する場合



シナリオはUASがMoratorium状態でEarly状態のメッセージであるBYEを受信した際に生じる準正常を示す。AliceはEarly状態でBYEを送信すると同時に、BobはInitial INVITEリクエストに対する 200 OKを送信する。AliceはEarly状態でBYEを送信するが(5.2 章及び付録 Aで説明する通り、これは推奨動作ではない)、BobはConfirmedダイアログ状態中でこのBYEを受信する。プロキシがフォーキングを行なっている場合、BYEは特定のUAとのアーリーダイアログのみを終了可能である。発信側が特定のUAのみではなく全てのアーリ

ーダイアログを終了する場合、BYEではなくCANCELを送信する必要がある。しかし、特定のアーリーダイアログを終了することが発信側の目的である場合、Early状態でのBYEの送信は違反ではない。[3]

BYE は CANCEL とは異なり、リクエストではなくダイアログを対象とするため、INVITE トランザクションが終了した後に BYE を受信しても正常に動作する。Alice は BYE の送信で Mortal 状態に遷移し、タイマーK が切れるまで Mortal 状態を継続する。Mortal 状態では、UAC は INVITE に対する 200 応答を受信してもセッションを確立しないが、INVITE トランザクションを終了するためにこの 200 に ACK を送信する。ACK は、INVITE トランザクションの 3Way ハンドシェイクを完了するために必ず送信される(詳細は付録 D を参照)。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK (ini-INVITE) Bob -> Alice

F4 BYE Alice -> Bob

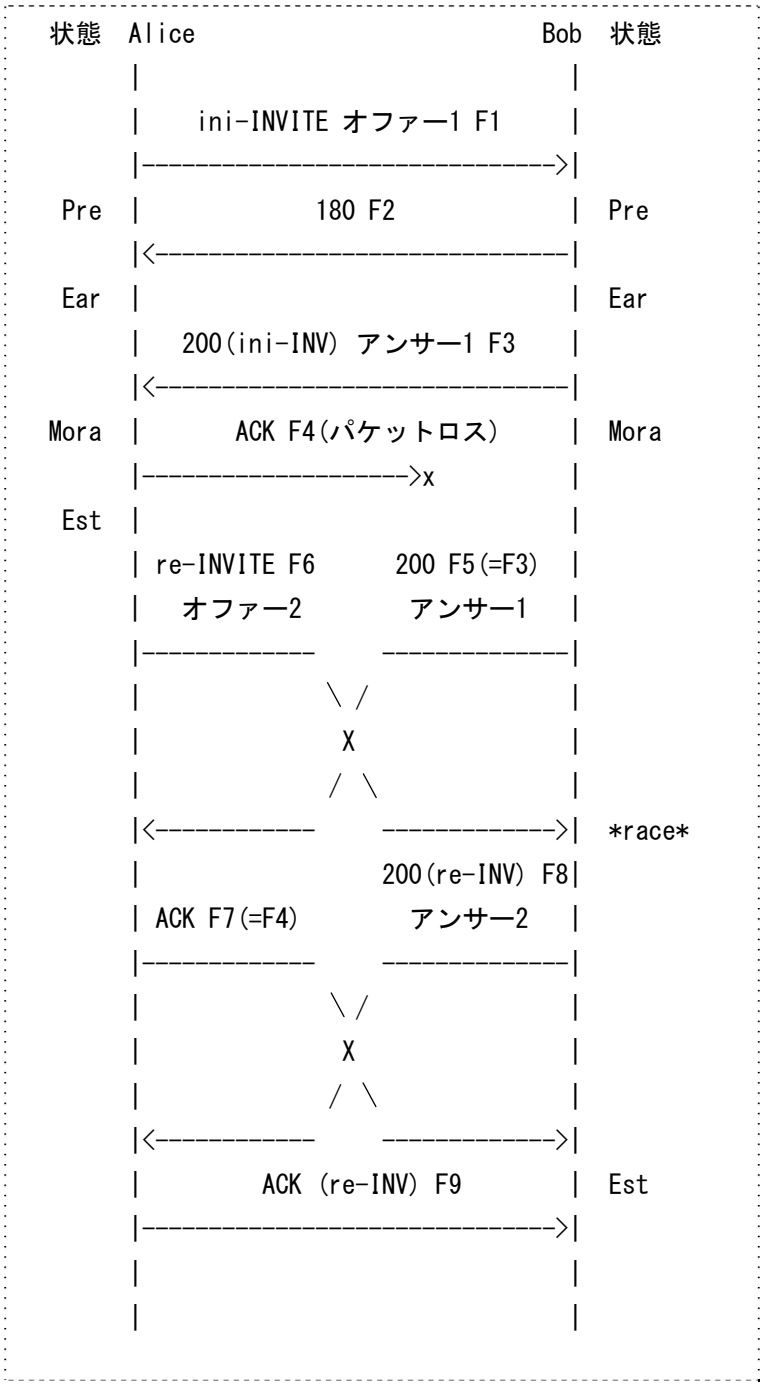
/* Alice は BYE の送信で Mortal 状態に遷移する。このため、Alice はアンサーを含む 200 応答を受信してもセッションを開始しない。*/

F5 ACK Alice -> Bob

F6 200 OK (BYE) Bob -> Alice

[3]発側は、フォーキングを期待した発呼を行ない、かつ、ある特定の UA との Early ダイアログを終了する可能性がある場合に限り Early ダイアログ上での BYE 送信を考慮し、それ以外の場合は CANCEL を送信する。

5.3.1.4　着信側がMoratorium状態でre-INVITE (Established状態)を受信する場合(ケース 1)



本シナリオは、UAS が Moratorium 状態で Established 状態の UAC から re-INVITE を受信した場合に生じる準正常を表す。

UAS は、ini-INVITE (オファー1)に対する ACK を受信する前に re-INVITE (オファー2)を受信する。UAS は ini-INVITE(F3 及び F5)に対して 200 OK (アンサー1)を送信し既にダイアログを確立しているため、UAS は re-INVITE (F8)に対する 200 OK (アンサー2)を送信する。(F5 は F3 の再送であり、SDP ネゴシエーションは行なわれない。)

5.3.3.2 節で説明するように、491 応答は INVITE 交差以外の場合でもセッション確立に密接に関係してい

ると考えられる。セッションに影響がないことからここでは 491 ではなく 200 を利用することを推奨する。しかし、491 応答でも辿り着く結果は同じであるため、どちらの応答も利用可能である。

さらに、UASは長時間ACKを受信しなかった場合にBYEを送信しダイアログを終了すべきである。ACK F7はini-INVITE F1 と同じCSeq番号を持つ(RFC 3261[1]の 13.2.2.4 節を参照)。UAは、CSeq番号を理由にACKを拒否または破棄するべきではない。

注：実装上の課題は本文書の範囲外であるが、本シナリオのような準正常を避けるためのヒントを以下に挙げる。発信側は、ACK が受信されたと推定できる程度の時間(例えば 2 秒ほど)、re-INVITE F6 の送信を待つことが可能である。UA がこのような動作をするために、実装者は、ユーザ動作(例：保留ボタン押下)とプロトコル動作(re-INVITE F6 の送信)を切り離して実装することが可能である。この場合、待ち時間については実装者依存である。大抵の場合において、このような実装は本節で説明するタイプの準正常を避けるために有効かもしれない。本文書では、ACK の送達を待つか否かについて特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

メッセージ詳細

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 137

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

/*オファー及びアンサーの例を示すためメッセージ詳細を記す。*/

F2 180 Ringing Bob -> Alice

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Length: 0

F3 200 OK Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

F4 ACK Alice -> Bob

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashd8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0

/*ACK リクエストが失われる。*/

F5(=F3) 200 OK Bob -> Alice (再送)

/*UAS は、ACK を受信したため、ini-INVITE に対する 200 OK を再送する。*/

F6 re-INVITE Alice -> Bob

INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf91
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

F7(=F4) ACK Alice -> Bob (再送)

/*ACK F7 の「(=F4)」は、F3 に対する ACK であるという観点から F4 と同様であることを示している。F4 と F7 の Via-branch 値が同じである必要はない。ACK F7 の Via-branch 値が F4 のものと異なるか否かについては RFC 3261 では不明確だが、この点は UAS の動作に影響しない。*/

F8 200 OK (re-INVITE) Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf91
Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 143

v=0

o=bob 2890844527 2890844528 IN IP4 client.biloxi.example.com

s=-

c=IN IP4 192.0.2.201

t=0 0

m=audio 3456 RTP/AVP 0

a=rtpmap:0 PCMU/8000

a=recvonly

F9 ACK (re-INVITE) Alice -> Bob

ACK sip:sip:bob@client.biloxi.example.com SIP/2.0

Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK230f21

Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

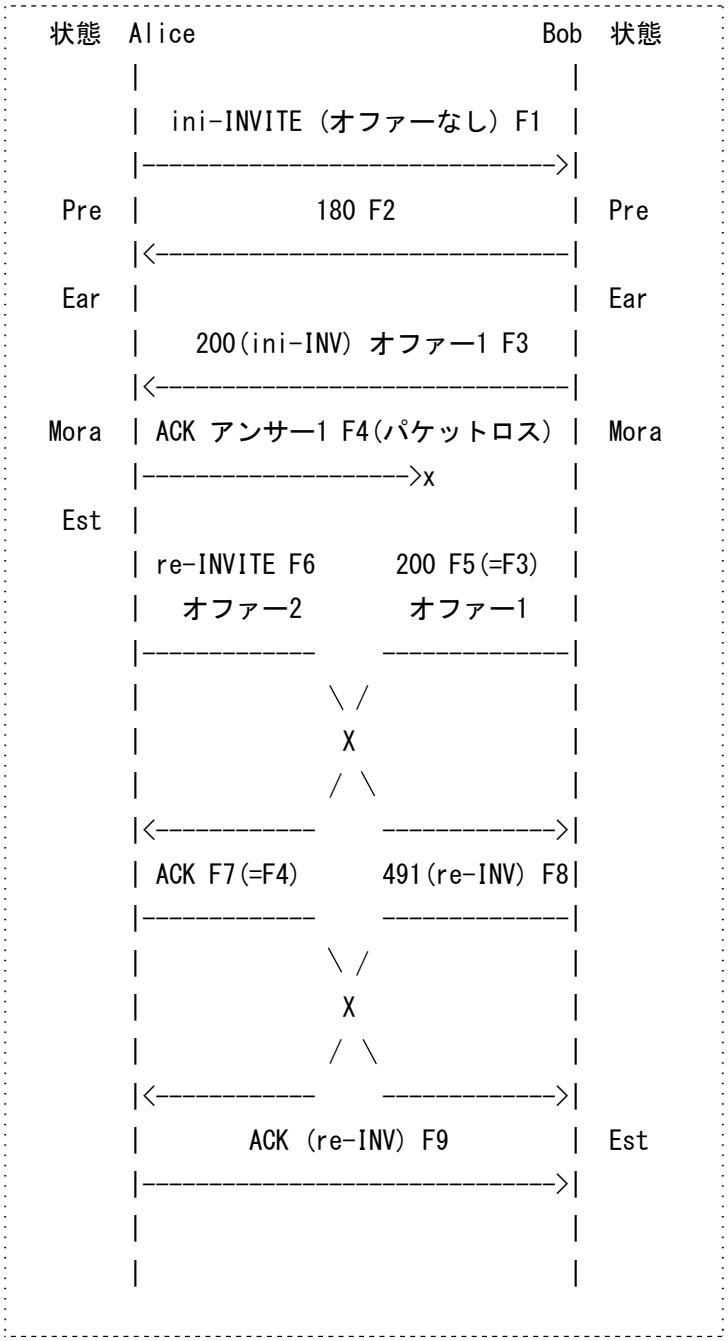
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 2 ACK

Content-Length: 0

5.3.1.5　着信側がMoratorium状態でre-INVITE (Established状態)を受信する場合(ケース 2)



本シナリオは基本的に 5.3.1.4 節のシナリオと同様であるが、200 でオファーを送信し ACK でアンサーを送信する点が異なる。本ケースでは、200 (F3)のオファーと re-INVITE (F6)のオファーが衝突する。

アンサーを受信するまで新規リクエストを適切に取り扱えないため、Bobはre-INVITE (F6)に対して 491 を送信する [4]。(注：491 応答がリクエストの衝突を表すと解釈される場合、代わりにRetry-Afterヘッダを含む 500 が送信されるかもしれない。しかし、500 は多くのケースに適用されエラーの真の原因を特定すること

[4] 491 エラー応答により、送信中のオファーが存在するために新たなオファーを受け付けられない状態であることをオファー側に通知する。

が難しいため、ここでは 491 を推奨する [5]。) F6 がオファーを含むUPDATEの場合も同様の結果となる。

注 : 5.3.1.4 節で説明の通り、本シナリオのような準正常を避けるため、発信側は、ACK が受信されたと推定できる程度の時間(例えば 2 秒ほど)、re-INVITE F6 の送信を待つことが可能である。本文書では、ACK の送達を待つか否かについては特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

メッセージ詳細

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Length: 0
```

/*このリクエストはオファーを含まない。オファー及びアンサーの例を示すためメッセージ詳細を記す。*/

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 133
```

[5]本例はリクエストとアンサーの衝突であるという理由により、491 エラー応答を選択しない実装が存在する可能性がある。しかし、エラー原因の特定やエラー応答受信後の動作という点で、ここで挙げた 500 のような他の応答よりも 491 エラー応答の方が優れている。

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*200 でオファーが行なわれる。*/

F4 ACK Alice -> Bob

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashd8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Type: application/sdp
Content-Length: 137

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*アンサーを含むが、このリクエストは失われる。*/

F5(=F3) 200 OK Bob -> Alice (再送)

/*UAS は ACK を受信していないため、ini-INVITE に対する 200 OK を再送する。*/

F6 re-INVITE Alice -> Bob

INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf91
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

/*このリクエストはオファーを含む。*/

F7(=F4) ACK Alice -> Bob (再送)

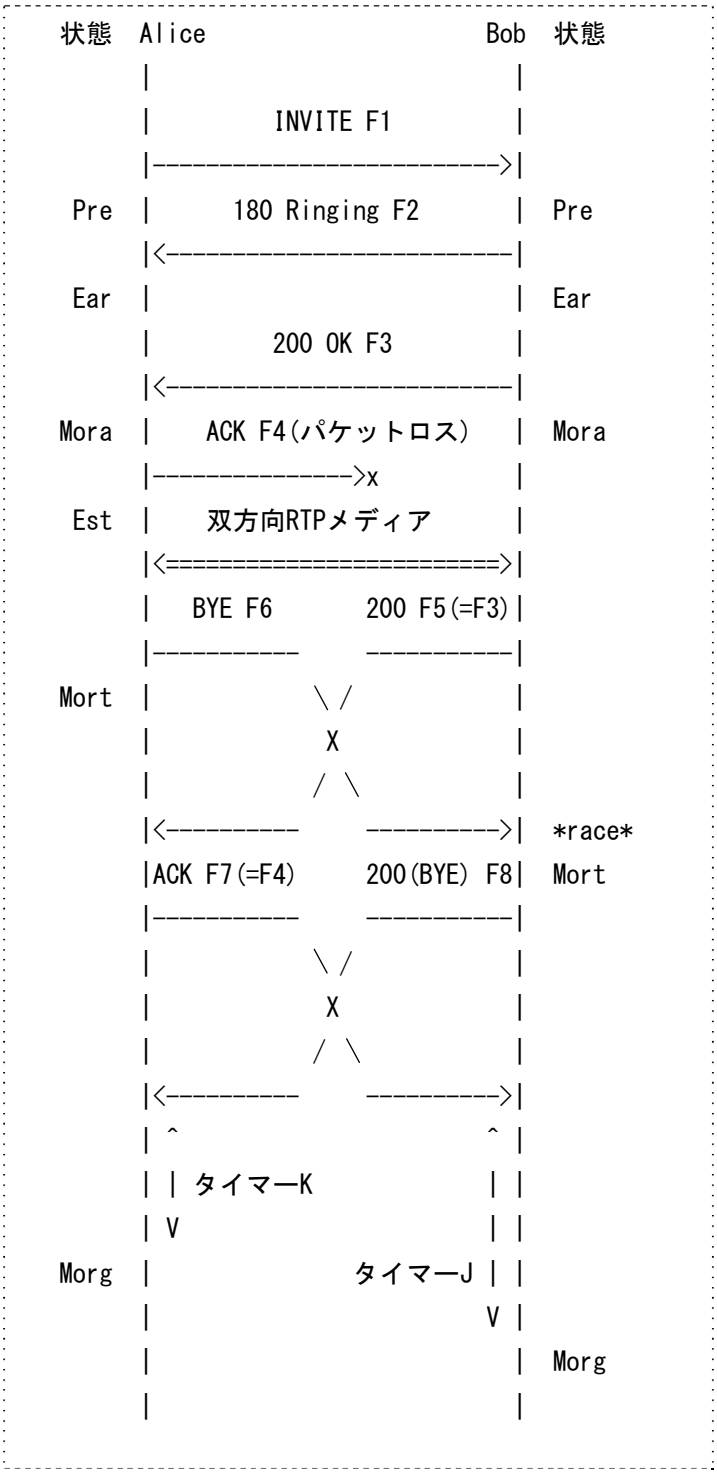
/*再送された 200 の受信を契機とした再送。ACK F7 の「(F4)」は、F3 に対する ACK であるという観点から F4 と同様であることを示している。F4 と F7 の Via-branch 値が同じである必要はない。ACK F7 の Via-branch 値が F4 のものと異なるか否かについては RFC 3261 では不明確だが、この点は UAS の動作に影響しない。*/

F8 491 (re-INVITE) Bob -> Alice

/*保留中のオファーがあるため、Bob は 491 (Request Pending)を送信する。*/

F9 ACK (re-INVITE) Alice -> Bob

5.3.1.6　着信側がMoratorium状態でBYE (Established状態)を受信する場合



本シナリオは、UAS が Moratorium 状態で Established 状態のメッセージである BYE を受信した場合に生じる準正常を示す。200 OK 応答に対する ACK リクエストが失われる（または遅延する）。Alice が BYE リクエストを送信しセッションを終了すると同時に、Bob は ini-INVITE に対して 200 OK を再送する。Alice の UA は、200 OK の再送の受信を契機にセッションを再確立しようとするかもしないが、この動作は誤りである。ダイアログが Mortal 状態の場合、セッションを再確立するべきではない。さらに同様の理由から、UAS が 200 OK でオファーを送信し BYE の受信後に ACK の再送を受信した場合にも、UAS はセッションを再開するべきではない。

注：5.3.1.4 節で説明の通り、実装上の課題は本文書の範囲外であるが、本シナリオのような準正常を避けるためのヒントを以下に挙げる。発信側は、ACK が受信されたと推定できる程度の時間(例えば 2 秒ほど)BYE F6 の送信を待つことが可能である。UA がこのような動作をするために、実装者は、ユーザ動作(例：受話器を置く)とプロトコル動作(BYE F6 の送信)を切り離して実装することが可能である。この場合、待ち時間については実装者依存である。大抵の場合において、このような実装は本節で説明するタイプの準正常を避けるために有効かもしれない。本文書では、ACK の送達を待つか否かについては特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

/*ACK リクエストが失われる。*/

F5(=F3) 200 OK Bob -> Alice

/*UAS は ACK を受信していないため、ini-INVITE に対して 200 OK を再送する。*/

F6 BYE Alice -> Bob

/*Bob が 200 OK を再送すると同時に Alice が BYE を送信する。Alice は Mortal 状態に遷移するため、これ以降 re-INVITE に対する 200 応答を受信したとしてもセッションを開始しない。*/

F7(=F4) ACK Alice -> Bob

/*ACK F7 の「(F4)」は、F3 に対する ACK であるという観点から F4 と同様であることを示している。F4 と F7 の Via-branch 値が同じである必要はない。ACK F7 の Via-branch 値が F4 のものと異なるか否かについては RFC 3261 では不明確だが、この点は UAS の動作に影響しない。*/

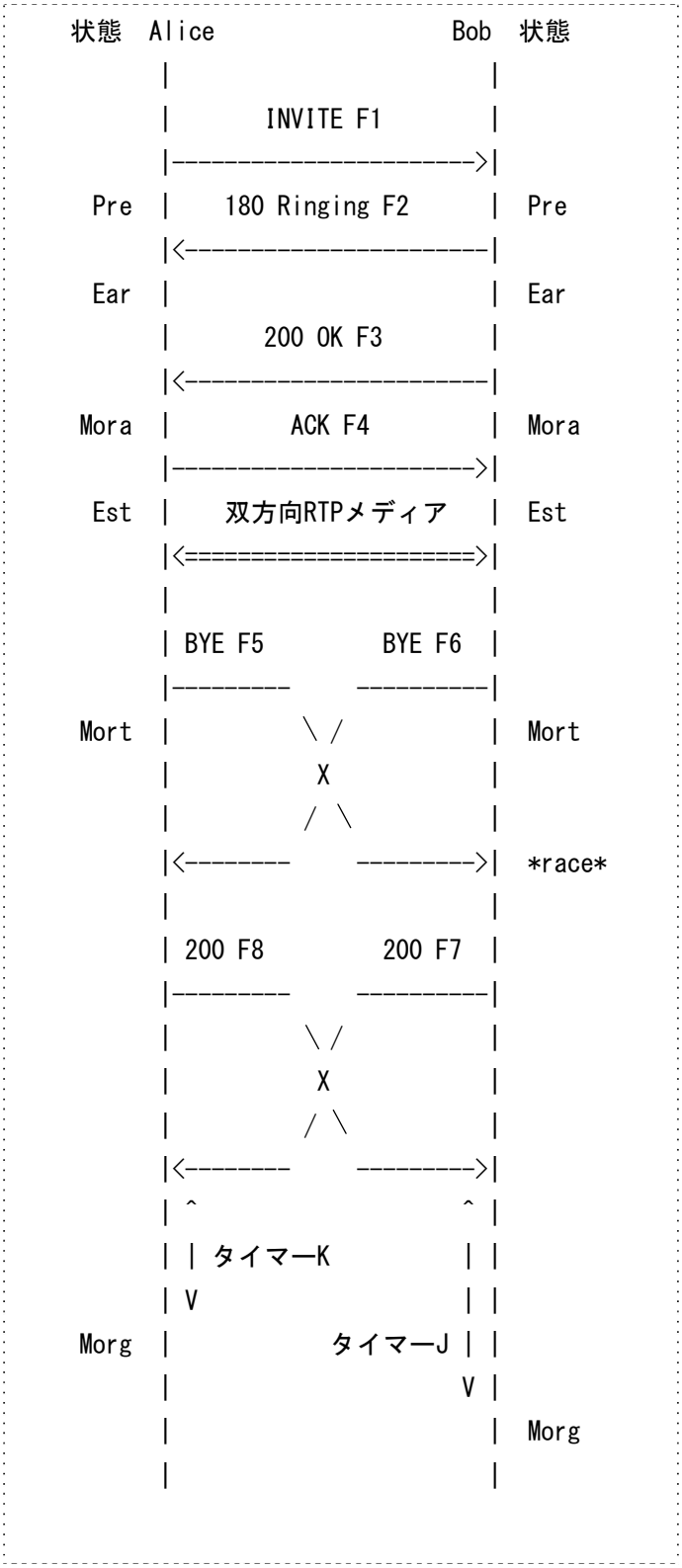
F8 200 OK (BYE) Bob -> Alice

/*Bob は BYE に対する 200 OK を送信する*/

5.3.2 Mortal状態でのメッセージ受信

本節では、Mortal 状態において他の状態から送信されたメッセージを受信した際に発生する準正常のコールフロー例を示す。

5.3.2.1 UAがMortal状態でBYE (Established状態)を受信する場合



本シナリオは、UAS が Mortal 状態において Established 状態のメッセージである BYE を受信した場合に生じる準正常を表す。Alice と Bob は同時に BYE を送信する。BYE リクエストがクライアントトランザクションに送られた少し後にダイアログ及びセッションは終了する。5.2 章で示したように、UA は Mortal 状態に留まる。

Mortal 状態の UA は、re-INVITE、UPDATE、または REFER のようにダイアログまたはセッション内で機能するリクエストに対してエラー応答を送信する。しかし、発信側と着信側がセッション終了の際にセッション関連情報を交換するユースケースを考慮し、UA は BYE へ 200 OK を送信する。(ダイアログとセッションは共に BYE の送信を契機に終了するため、Mortal 状態における BYE 受信時に 200 またはエラー応答のどちらを送信したとしても、終了するという結果は変わらない。従って本例では 200 応答を利用しているが、他の応答も利用可能である。)

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob

/*Alice が BYE を送信した時点でセッションは終了する。ダイアログはまだ存在するが、この後すぐに終了する。ダイアログは、BYE リクエストのタイムアウトが起こった時点で完全に終了する。*/

F6 BYE Bob -> Alice

/*Bob は Alice と同時に BYE を送信する。Bob はセッションとダイアログを終了する。*/

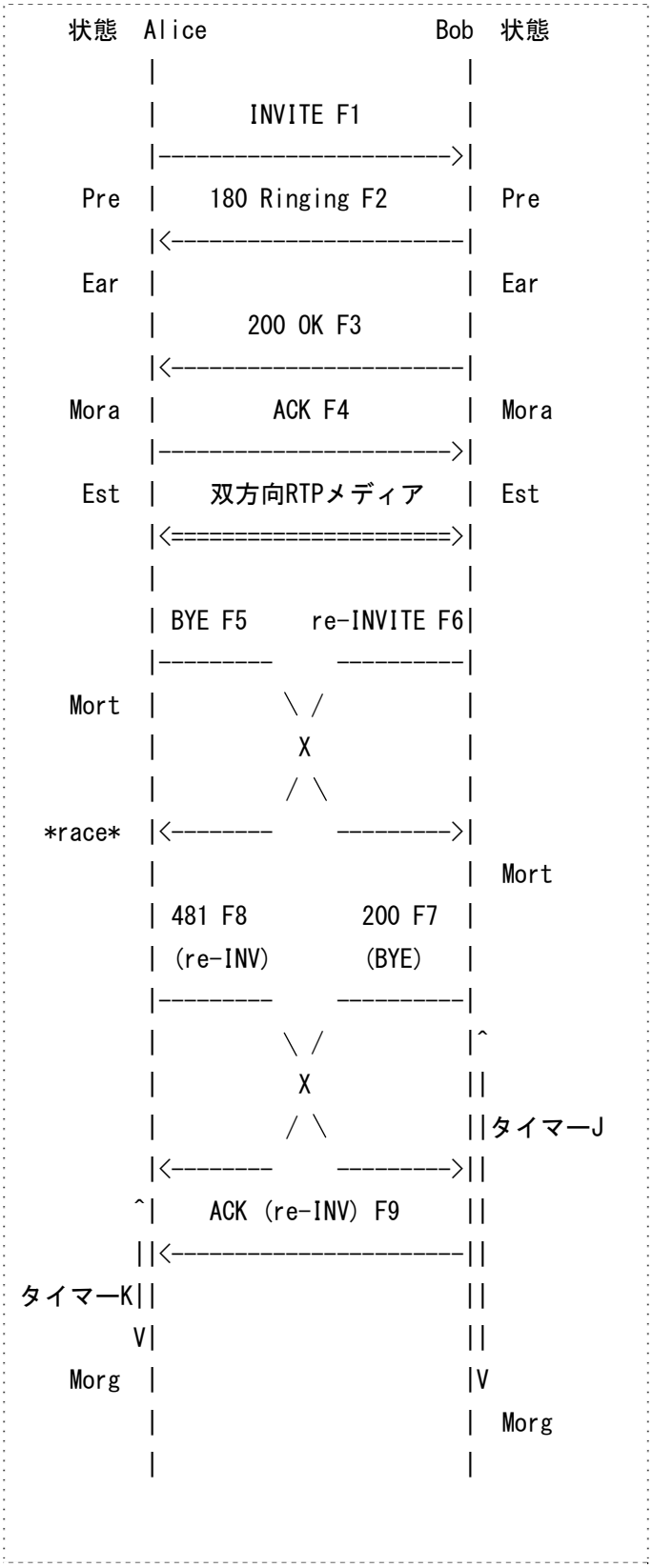
F7 200 OK Bob -> Alice

/*ダイアログは Moratorium 状態にあるので、Bob は BYE リクエストに対して 200 応答を送信する。*/

F8 200 OK Alice -> Bob

/*Alice は BYE の送信により Established 状態から Mortal 状態に遷移したので、Alice は BYE リクエストに対し 200 応答を送信する。*/

5.3.2.2　UAがMortal状態でre-INVITE (Established状態)を受信する場合



本シナリオは、UAS が Mortal 状態で Established 状態のメッセージである re-INVITE を受信する場合に生

じる準正常を表す。ボブは re-INVITE を送信すると同時に Alice が BYE を送信する。エラー応答に対する ACK はトランザクション層で処理されること、および 481 を受信した時点では INVITE クライアントトランザクションはまだ存在する(当然ダイアログは終了している)ことから、Bob は 481 応答に対し ACK を送信する。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob

/*Alice は BYE を送信しセッションを終了し、Established 状態から Mortal 状態に遷移する。*/

F6 re-INVITE Bob -> Alice

/*Alice が BYE を送信すると同時に Bob は re-INVITE を送信する。Alice は BYE を送信した時点で Mortal 状態に遷移するが、Bob は BYE を受信するまでそのことを認識しない。従って、Alice の観点ではダイアログは Terminated 状態であるが、Bob の観点では Confirmed 状態のままである。準正常が生じている。*/

F7 200 OK (BYE) Bob -> Alice

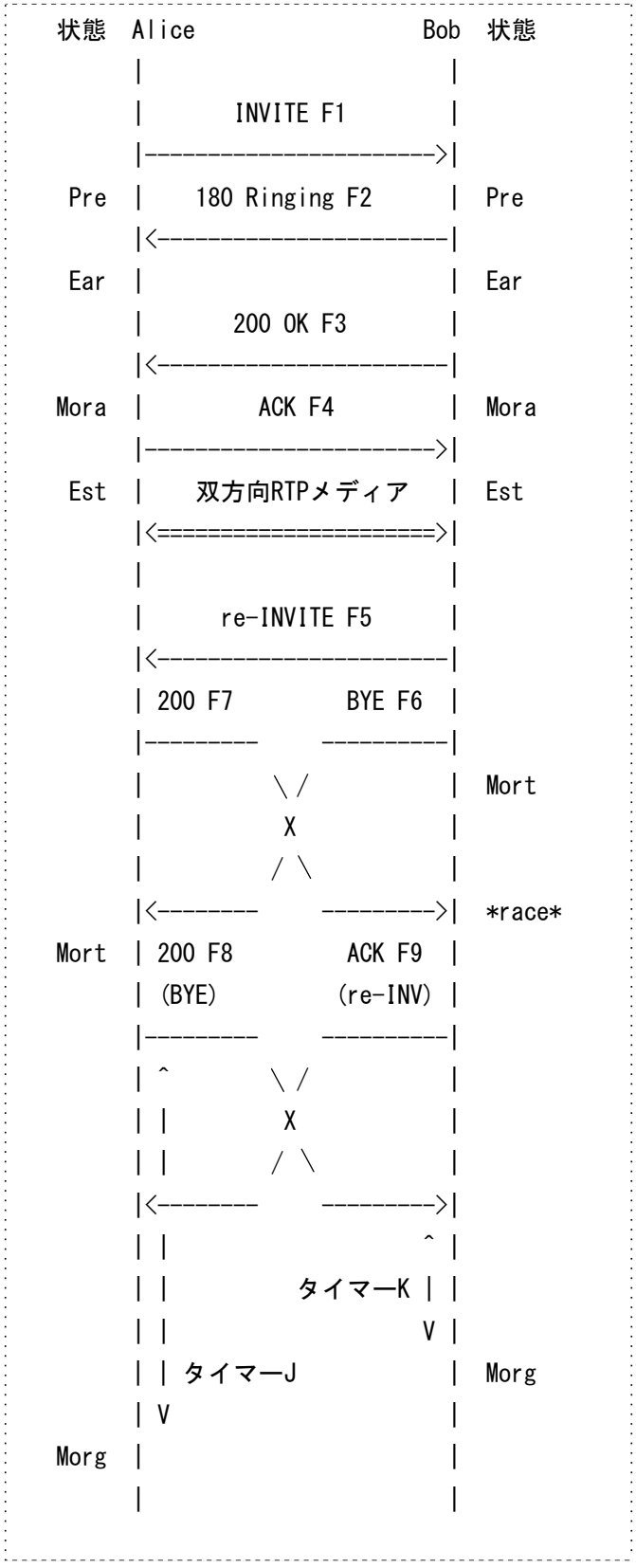
F8 481 Call/Transaction Does Not Exist (re-INVITE) Alice -> Bob

/*Alice は Mortal 状態にいるので、re-INVITE に対し 481 応答を送信する。*/

F9 ACK (re-INVITE) Bob -> Alice

/*エラー応答に対する ACK は Bob の INVITE クライアントトランザクションで処理される。*/

5.3.2.3　UAがMortal状態でre-INVITEに対する 200 OK (Established状態)を受信する場合



本シナリオは UAS が Mortal 状態で Established 状態のメッセージである re-INVITE に対する 200 を受信した際に生じる準正常を示す。Bob は re-INVITE 送信直後に BYE を送信する。(例：電話アプリケーションの場合、セッションリフレッシュの直後にユーザが電話を切る可能性がある。) Bob は Mortal 状態で INVITE

に対する 200 応答に対して ACK を送信し、INVITE トランザクションを完了する。

注：5.3.1.4 節で説明の通り、実装上の課題は本文書の範囲外であるが、本シナリオのような準正常を避けるためのヒントを以下に挙げる。UAC は、re-INVITE トランザクション(F5)が完了するまで BYE F6 の送信を待つことが可能である。UA がこのような動作をするために、実装者は、ユーザ動作(例：受話器を置く)とプロトコル動作(BYE F6 の送信)を切り離して実装することが可能である。この場合、待ち時間については実装者依存である。大抵の場合において、このような実装は本節で説明するタイプの準正常を避けるために有効かもしれない。本文書では、ACK の送達を待つか否かについては特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 re-INVITE Bob -> Alice

```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

/*Mortal 状態においても re-INVITE が通常通り処理されることを表すため、メッセージ詳細を乗せている部分がある。*/

F6 BYE Bob -> Alice

/*Bob が re-INVITE を送信した直後に BYE を送信する。Bob はセッションを終了し、Established 状態から Mortal 状態に遷移する。*/

F7 200 OK (re-INVITE) Alice -> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashd7
;received=192.0.2.201
Require: timer
Session-Expires: 300;refresher=uac
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0

/*Bob が BYE を送信すると同時に、Alice が re-INVITE に対し 200 OK を送信する。準正常が生じている。*/

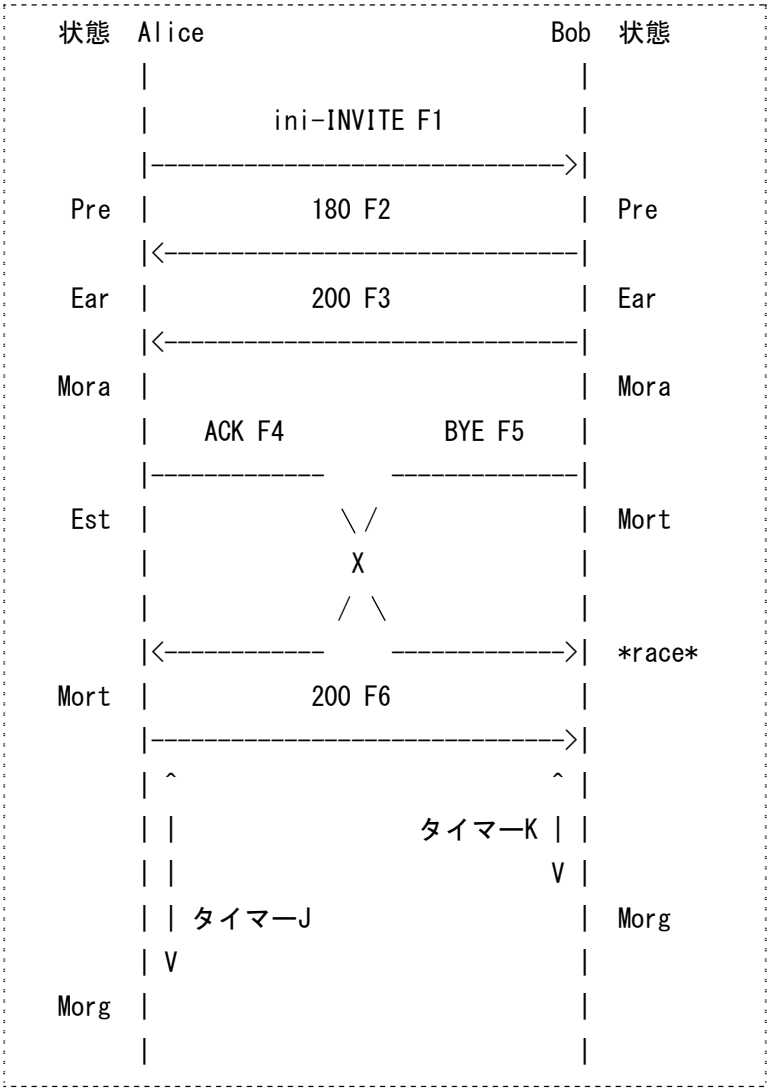
F8 200 OK (BYE) Alice -> Bob

F9 ACK (re-INVITE) Bob -> Alice

ACK sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bK74b44
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0

/*INVITE トランザクションのスリーウェイハンドシェイクを完了するために、Bob は Mortal 状態において ACK を送信する。*/

5.3.2.4　着信側がMortal状態でACK (Moratorium状態)を受信する場合



本シナリオは、UAS が Mortal 状態において Established メッセージである 200 に対する ACK を受信する場合において生じる準正常を示す。Alice が ACK を送信するのと同時に Bob が BYE を送信する。2xx がオファーを含み ACK がアンサーを含む場合準正常が生じる。Bob は既に BYE を送信しセッションを終了しているため、Bob が ACK を受信してもセッションは開始されない。ACK リクエストに含まれるアンサーは無視される。

注：5.3.1.4 節で説明の通り、実装上の課題は本文書の範囲外であるが、本シナリオのような準正常を避けるためのヒントを以下に挙げる。実装者は、ユーザ動作(例：受話器を置く)とプロトコル動作(BYE F5 の送信)を切り離して実装することが可能である。この場合、待ち時間については実装者依存である。大抵の場合において、このような実装は本節で説明するタイプの準正常を避けるために有効かもしれない。本文書では、ACK の送達を待つか否かについては特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

/*Alice と Bob の間で RTP ストリームが確立される。*/

F5 BYE Alice -> Bob

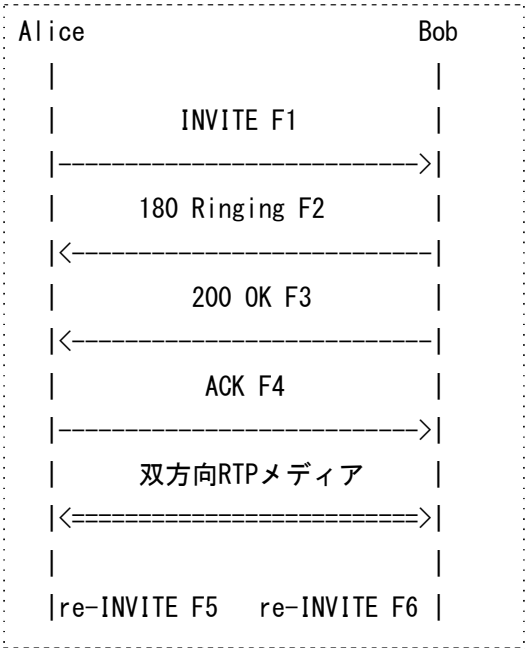
F6 200 OK Bob -> Alice

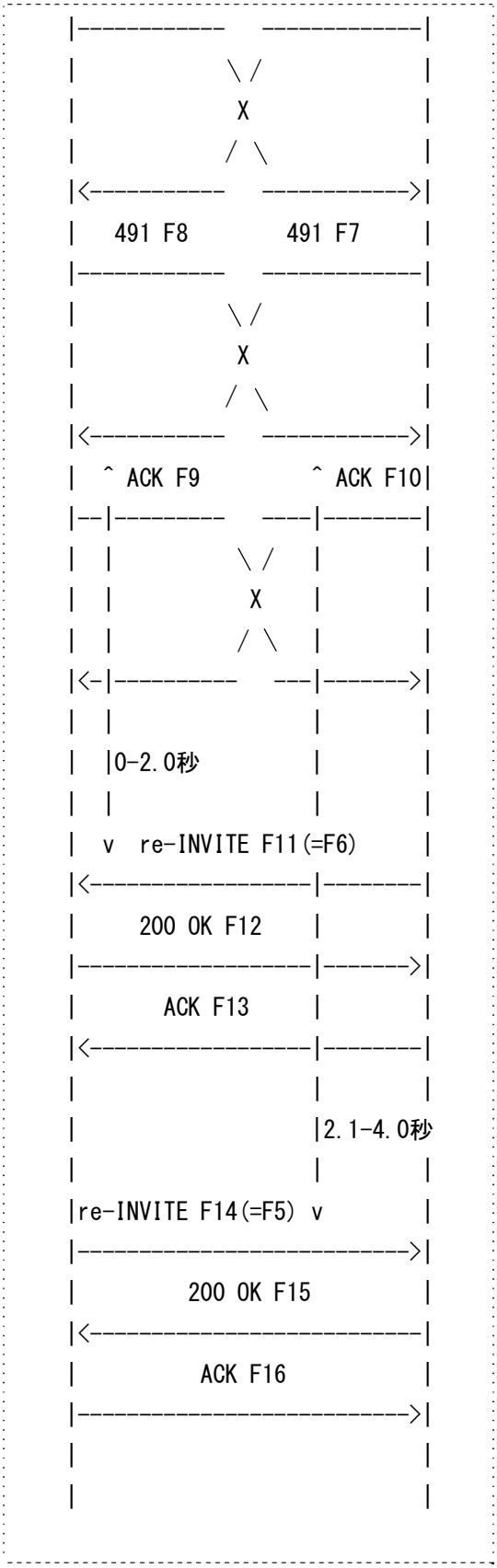
/*Alice が BYE を送信しセッション及びダイアログを終了する。*/

5.3.3 その他の準正常

本章ではダイアログ状態遷移とは直接に関係しない準正常例を示す。SIPにおける処理機能は、ダイアログ、セッション及びトランザクションの 3 層に分かれている。お互いに関連性はあるものの、それぞれは個別に扱われなくてはならない。トランザクションでの処理の詳細はRFC 3261[1]の 17 章で説明されている。本文書ではこれまでダイアログでの処理について明確化してきた。本章では、SIPで確立されるセッションに関連する準正常について説明する。

5.3.3.1 Re-INVITEの交差





本シナリオでは、AliceとBobが同時にre-INVITEを送信する。同じダイアログ内で 2 つのre-INVITEが交差した場合、RFC 3261[1]の 14.1 節に従い、異なるインターバル後にre-INVITEの送信を試みることになる。Alice

がCall-IDを所有する(Call-IDを生成した)ため、AliceとBobのre-INVITEが交差した場合、Aliceは 2.1 から 4.0 秒後に再びre-INVITEの送信を試みる。BobはCall-IDの所有者ではないため、0.0 から 2.0 秒後に再びINVITEの送信を試みる。

従って、INVITEの交差が発生することを考慮し、それぞれのユーザエージェントはCall-IDを生成したかどうかを記憶しておく必要がある [6]。

本例で、Alice の re-INVITE はセッション変更、Bob の re-INVITE はセッションリフレッシュである。この場合、491 応答後、Bob は Alice より先にセッションリフレッシュのための re-INVITE を再試行する。もし Alice が先に re-INVITE を再試行した場合(Alice が Call-ID の所有者でない場合)、このリクエストによりセッション変更とリフレッシュが同時に行なわれる。この場合、Bob はセッションリフレッシュのために自身の re-INVITE を再試行する必要はないと判断することになる。

その他、セッション変更のための 2 つの re-INVITE が交差した場合、Call-ID 所有者(他の UA の後に re-INVITE を再試行する UA)が 491 の後に同じ re-INVITE を再試行することは意図しない動作となる可能性があるため、UA は re-INVITE 再試行の必要性を判断しなくてはならない。(例：保留と映像メディア追加が交差し、単純にタイマーにより re-INVITE を再試行した場合、音声の保留直後に映像が送信されることが想定される。これはユーザの意図した動作ではないと考えられる。)

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 re-INVITE Alice -> Bob

```
INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 147

v=0
```

[6] Call-ID を所有する（Call-ID を生成する）とは、当初に発信(ini-INVITE を送出)した側を示すものである。

o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com

s=-

c=IN IP4 192.0.2.101

t=0 0

m=audio 49172 RTP/AVP 0

a=rtpmap:0 PCMU/8000

a=sendonly

/*交差する INVITE リクエストの内容を示すため、一部メッセージ詳細を示す。*/

F6 re-INVITE Bob -> Alice

INVITE sip:alice@client.atlanta.example.com SIP/2.0

Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd7

Session-Expires: 300;refresher=uac

Supported: timer

Max-Forwards: 70

From: Bob <sip:bob@biloxi.example.com>;tag=8321234356

To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Content-Length: 0

/*セッションリフレッシュの re-INVITE リクエストと保留のための re-INVITE リクエストが同時に送信される。*/

F7 491 Request Pending Bob -> Alice

/*re-INVITE を処理中のため、491 応答を返す。*/

F8 491 Request Pending Alice -> Bob

F9 ACK (INVITE) Alice -> Bob

F10 ACK (INVITE) Bob -> Alice

F11 re-INVITE Bob -> Alice

INVITE sip:alice@client.atlanta.example.com SIP/2.0

Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd71

Session-Expires: 300;refresher=uac

Supported: timer

Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*BobはCall-IDの所有者ではない[7]ため、0.0 から 2.0 秒後に再度re-INVITEを送信する。*/

F12 200 OK Alice -> Bob

F13 ACK Bob -> Alice

F14 re-INVITE Alice -> Bob

INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf91
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 3 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

[7] Bob は当初の発信者（ini-INVITE の送信者）ではない。着信側である。

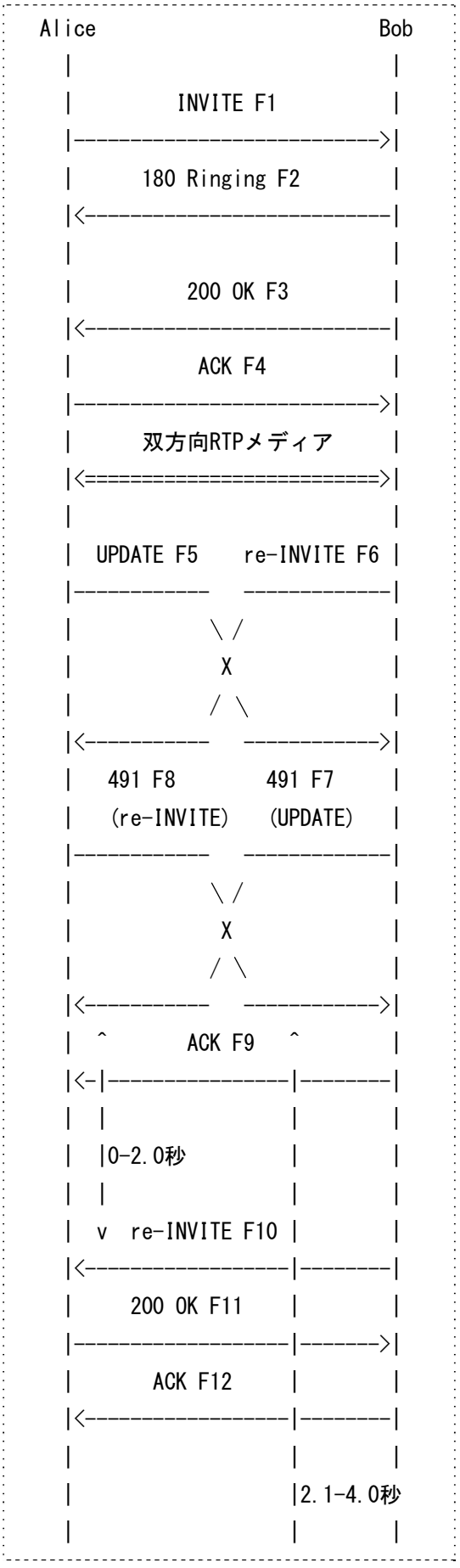
/*AliceはCall-IDの所有者である[8]ため、2.1 から 4.0 秒後に再度re-INVITEを送信する。*/

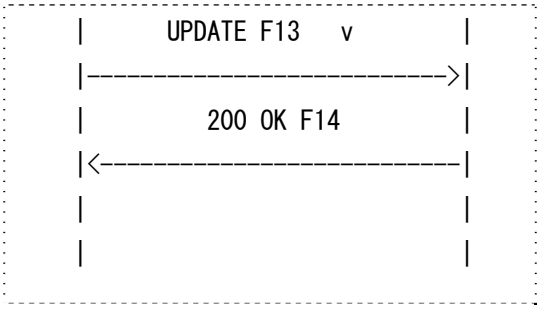
F15 200 OK Bob -> Alice

F16 ACK Alice -> Bob

[8] Alice は当初の発信者（ini-INVITE の送信者）である。発信側である。

5.3.3.2　UPDATEとre-INVITEの交差





本シナリオでは、UPDATE が SDP オファーを含むため、「re-INVITE の交差」と同様に UPDATE と re-INVITE は共に 491 で応答される。セッション記述を含まないセッションリフレッシュのための UPDATE と re-INVITE が交差する場合には、両リクエストは 200 で成功する(491 は UA が処理中のリクエストが存在することを意味する)。このことは UPDATE の交差に対しても同様に当てはまる。前者同様に両方の UPDATE がセッション記述を含む場合、両リクエストは 491 で失敗する。後者同様の場合、200 で成功する。

注：SDP オファーが処理中のため 491 が送信される。491 は SIP により確立されるセッションに影響を与える事項に関連したエラーである。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 UPDATE Alice -> Bob

```
UPDATE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 UPDATE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
```

t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

/*交差するメッセージを表すため、メッセージ詳細を示す。*/

F6 re-INVITE Bob -> Alice

INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*本例は、セッションリフレッシュのための re-INVITE と保留のための UPDATE が同時に送信されるケースである。*/

F7 491 Request Pending (UPDATE) Bob -> Alice

/*re-INVITE が処理中のため、491 応答が返される。*/

F8 491 Request Pending (re-INVITE) Alice -> Bob

F9 ACK (re-INVITE) Alice -> Bob

F10 re-INVITE Bob -> Alice

INVITE sip:alice@client.atlanta.example.com SIP/2.0

Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd71
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70

From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Bob は Call-ID の所有者ではないため、0.0 から 2.0 秒後に再度 INVITE を送信する。*/

F11 200 OK Alice -> Bob

F12 ACK Bob -> Alice

F13 UPDATE Alice -> Bob

UPDATE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf91
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 3 UPDATE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0

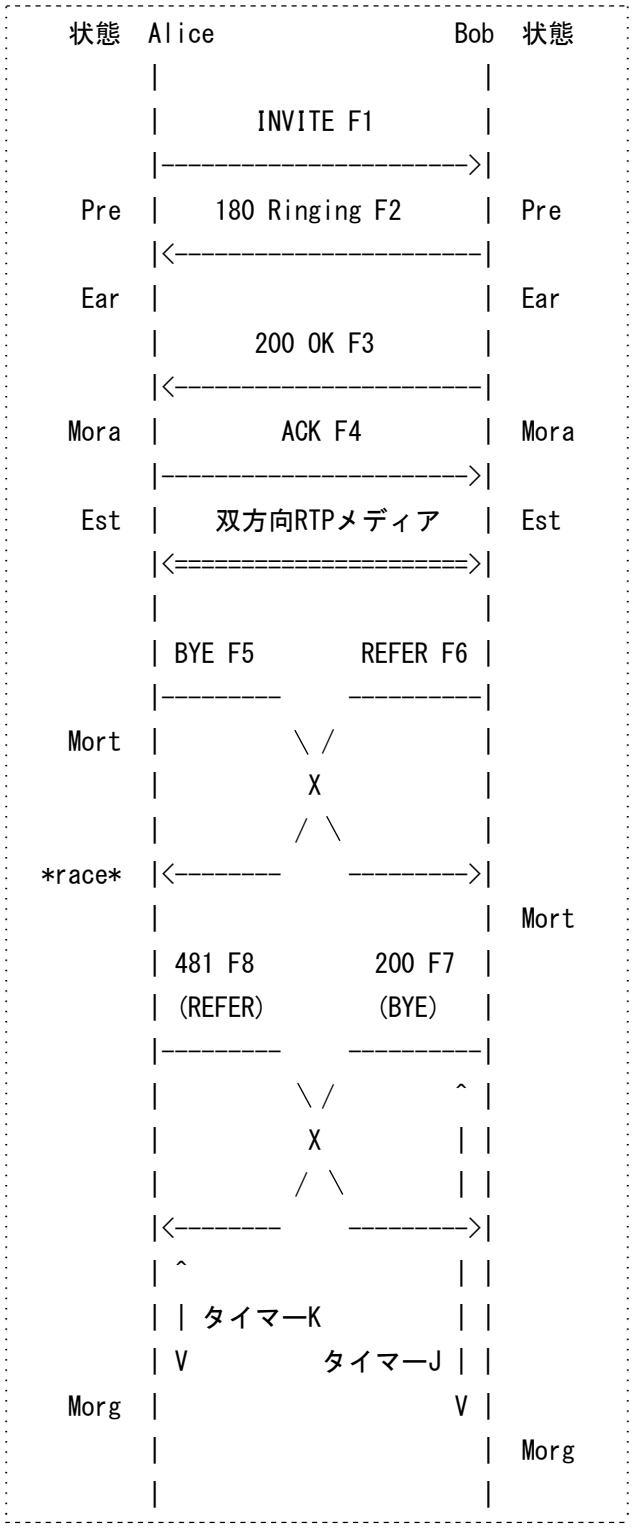
a=rtpmap:0 PCMU/8000

a=sendonly

/* Alice は Call-ID の所有者であるため、2.1 から 4.0 秒後に再度 UPDATE を送信する。*/

F14 200 OK Bob -> Alice

5.3.3.3　Mortal状態でREFER (Established状態)を受信する場合



本シナリオは、UASがMortal状態においてEstablished状態のメッセージであるREFERを受信した際に生じる準正常を示す。BobがREFERを送信すると同時にAliceがBYEを送信する。BobはREFERを同じダイアログの中で送信する。BYEの送信時点でAliceのダイアログ状態はMortal状態に遷移する。Mortal状態において、UAは内部処理のためにダイアログ情報を保持するが、外部に対してダイアログが存在すべきではない。従って、このUAは、ダイアログ内リクエストとして送信されたREFERに対してエラー応答を送信する。つま

り、Mortal状態にあるAliceはREFERに対してエラー応答を送信する。しかし、Bobは既にREFERのSUBSCRIBE usageを開始しており、BYEの受信でINVITE dialog usageが終了しても、SUBSCRIBE usageが終了するまでダイアログは継続する。この場合のBobの動作はRFC 5057[6]に従う必要がある。

メッセージ詳細

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob

/*Alice は BYE リクエストを送信すると共にセッションを終了し、Confirmed 状態から Terminated 状態に遷移する。*/

F6 REFER Bob -> Alice

/*Alice が BYE を送信すると同時に Bob が REFER を送信する。Bob は INVITE ダイアログ上で REFER を送信する。Alice が BYE を送信することでダイアログ状態は Mortal 状態に遷移するが、Bob は BYE 受信までこれを認識しない。準正常が生じている。*/

F7 200 OK (BYE) Bob -> Alice

F8 481 Call/Transaction Does Not Exist (REFER) Alice -> Bob

/*Mortal 状態にある Alice が REFER に対し 481 を送信する。*/

5.4 セキュリティ上の考慮点

本文書はRFC 3261[1]、RFC 3264[2]、及びRFC 3515[4]で規定する動作の明確化を行なう。本文書における明確化にもこれらの文書におけるセキュリティ上の考慮点が適用される。

5.5 謝辞

本文書にご協力いただいた以下の方々に謝意を表したい。

Robert Sparks 氏、Dean Willis 氏、Cullen Jennings 氏、James M. Polk 氏、Gonzalo Camarillo 氏、小上賢一氏、清水昭弘氏、宗像麻由美氏、稲垣泰典氏、城所忠篤氏、平木健一氏、Dale Worley 氏、Vijay K. Gurbani 氏、Anders Kirstensen 氏。

5.6 参照文献

5.6.1 引用文献

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[5] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

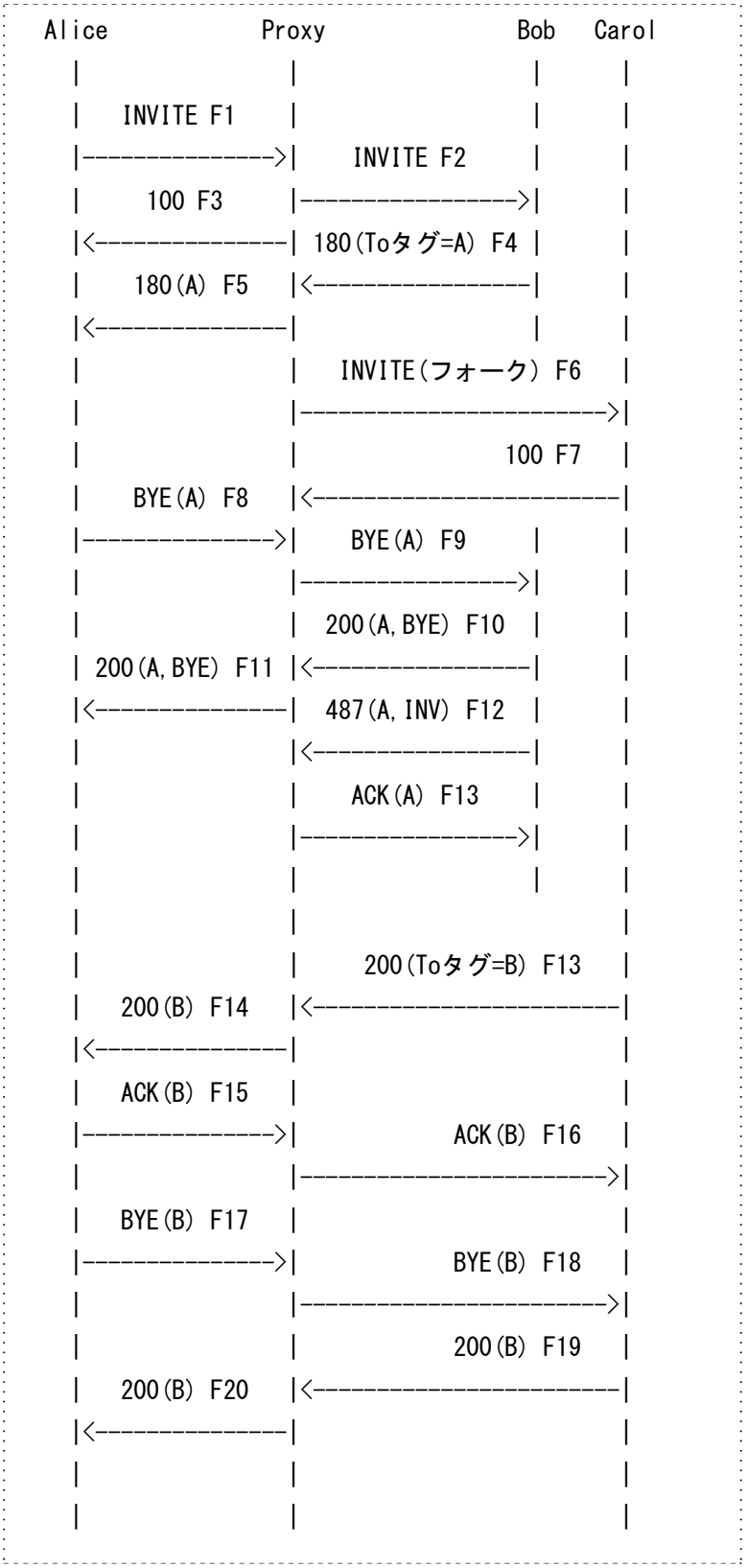
5.6.2 参考文献

[6] Sparks, R., "Multiple Dialog Usages in the Session Initiation Protocol", RFC 5057, November 2007.

[7] Sparks, R., "Correct transaction handling for 200 responses to Session Initiation Protocol INVITE requests", Work in Progress, July 2008.

付録A.　アーリーダイアログ内の BYE

本シナリオは、5.3.1.3 節に関連し、アーリーダイアログにおける BYE が混乱を招くケースを示し、Early 状態における BYE が推奨されない理由について説明する。



プロキシによるフォーキングが想定される場合、Early状態におけるBYEの送信には注意が必要である。本例ではBYEリクエストは正常に処理され、Bobとのダイアログを正しく終了することに成功する。BobがBYEの受信によりダイアログを終了した後、ini-INVITEに対して 487 を送信する。RFC 3261[1]の 15.1.2 節により、UASはBYEの受信後、全ての処理中のリクエストに対し 487 を生成することが推奨されている(RECOMMENDED)。本例では、Bobはini-INVITEの処理中にBYEを受信するため、ini-INVITEに対して 487 を送信する。

しかし、Bob とのダイアログを終了しても、Alice は INVITE に対し最終応答(Carol からの 200)を受信する。これは、Early 状態における BYE の成功/失敗に関わらず、Alice は INVITE に対する最終応答の受信と共に INVITE トランザクションを終了し、新規ダイアログを確立する準備がなくてはならない(MUST)ことを示している。

特定のアーリーメディアを終了する目的で Early 状態において BYE を送信することは違反ではない。それは発信者の意図を満たすかも知れない。しかし、Early 状態における BYE または CANCEL の選択は慎重になされなくてはならない。CANCEL は呼接続全てを取り消す場合に適切である。BYE は特定のアーリーダイアログを取り消すが、他の着信先との呼接続を継続する場合に適切である。BYE または CANCEL のどちらを利用する場合においても、UAC は一つまたは複数の着信先との呼接続を確立する可能性に備える必要がある。

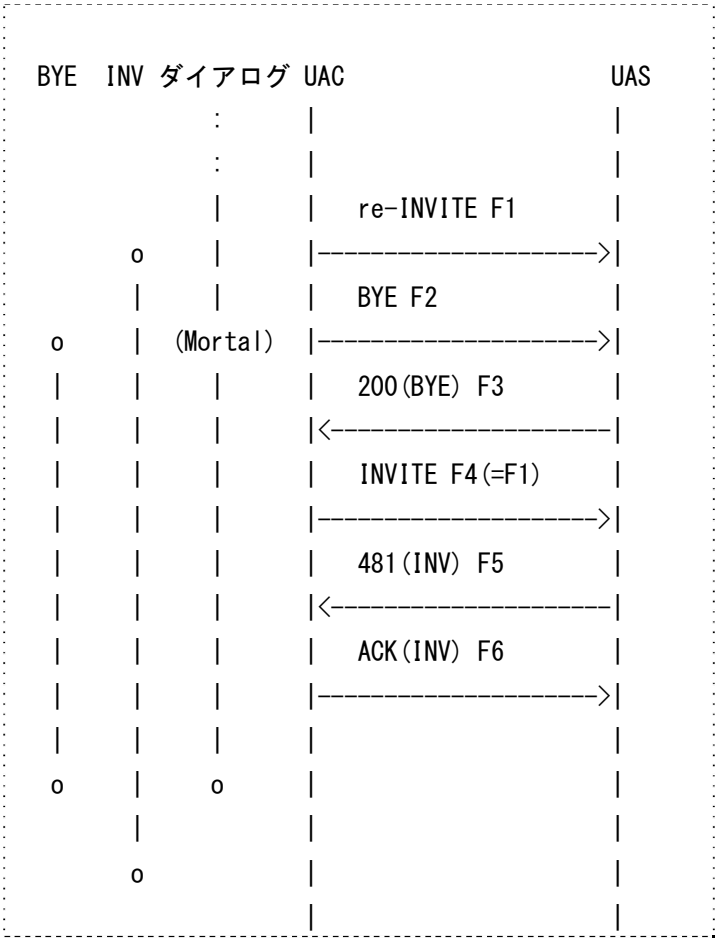
付録B.　BYE リクエストと re-INVITE のオーバーラップ

本ケースは 5.3.2.3 節と同様に見えるが、準正常ではない。このケースは何らかの理由により INVITE に対する応答がない場合の動作について述べる。このケースは混乱を招き易いため、本付録でこのケースにおける動作とその理由について説明する。

まず第一に、トランザクション層とダイアログ層の動作を混同しないことが重要である。RFC 3261 [1]はトランザクション層の動作詳細を規定している。ダイアログ層の動作については本文書で説明している。これら二つの層では同じ SIP メッセージの送受信により状態遷移を行なう可能性があるが、これら二つの動作は独立であることに注意しなくてはならない。(トランザクションを残したままダイアログを終了することは可能である。逆も同様である。)

上記のシーケンスでは、F1 に対し応答がなく、F1 の直後に F2 (BYE)が送信されている。(F1 はダイアログ内リクエストである。F1 が ini-INVITE だった場合、UAC が INVITE に対し To タグのついた暫定応答を受信するまで、BYE の送信は不可能である。)

UAC のダイアログ及びトランザクション状態を以下に示す。



UAC の INVITE クライアントトランザクションは F1 が送信された時点で開始する。UAC は F1 の直後に BYE (F2)を送信する。これは可能な動作である。(通常、BYE 及びその他 SIP メソッドの利用はそれぞれ独立している。しかし、前の SDP オファーの処理中に別の SDP オファーを含むリクエストを送信することは禁じられていることに注意。)

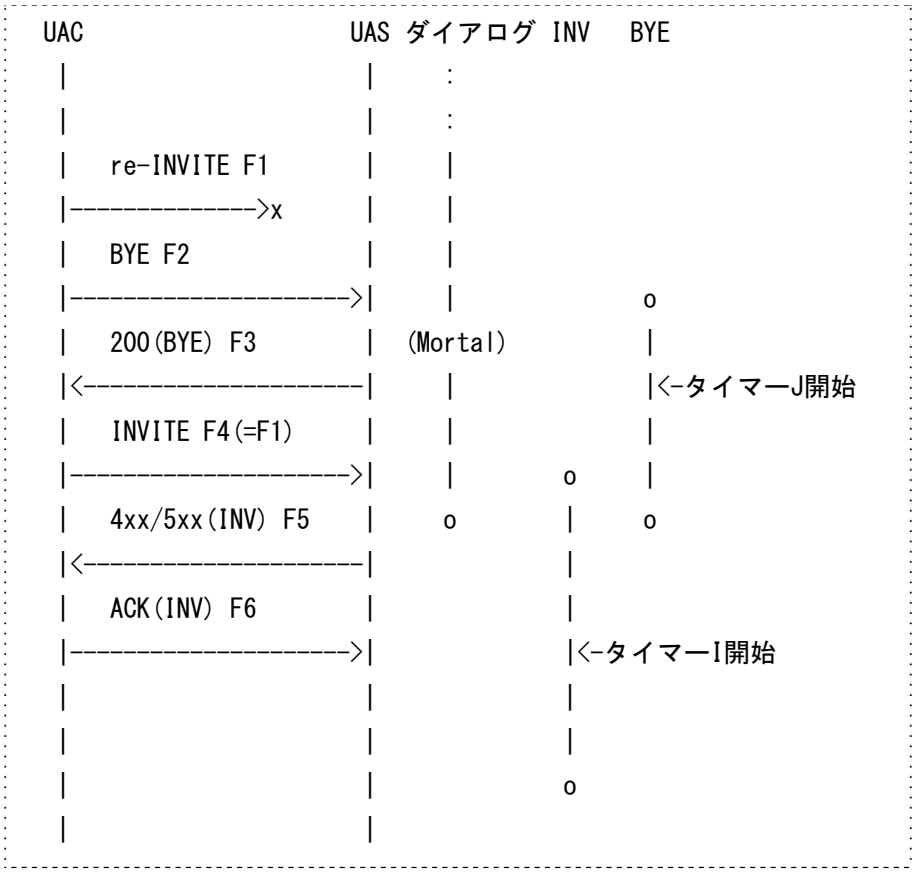
その後、F2 が BYE クライアントトランザクションの契機となる。同時にダイアログ状態は Mortal 状態に遷移し、BYE または BYE に対する応答のみ処理可能となる。

F1 の再送はトランザクション層で処理され、さらに INVITE トランザクションはまだ Terminated 状態に遷移していないため、Mortal 状態において F4 (INVITE の再送)を送信することは許容されている。上記のように、ダイアログとトランザクションはお互い独立して動作する。従って、ダイアログが Terminated 状態に遷移してもトランザクション処理は継続されなければならない。

注：5.3.1.4 節で説明の通り、実装上の課題は本文書の範囲外であるが、本シナリオのような準正常を避けるためのヒントを以下に挙げる。UAC は、re-INVITE トランザクション(F1)が完了するまで BYE F2 の送信を待つことが可能である。UA がこのような動作をするために、実装者は、ユーザ動作(例：受話器を置く)とプロトコル動作(BYE F2 の送信)を切り離して実装することが可能である。この場合、待ち時間については実装者依存である。大抵の場合において、このような実装は本節で説明するタイプの準正常を避けるために有効

かもしれない。本文書では、ACK の送達を待つか否かについては特に推奨事項はない。ユーザの使用感は実装に依存しプロトコル動作とは直接の関係はないため、実装者はユーザの使用感に与える影響を考慮した上でこの待ち動作を行なうか決定するべきである。

続いて、UAS の状態を以下に示す。



UAS からは F1 パケットが失われる、または遅延するように見える(ここでは UAS が F1 INVITE の前に F2 BYE を受信する際の動作について説明する)。従って、UAS は F2 により BYE トランザクションを開始し、同時にダイアログは Mortal 状態に遷移する。その後、F4 の受信により INVITE サーバトランザクションが開始する(Mortal 状態で INVITE サーバトランザクションを開始することは許容されている。INIVTE サーバトランザクションは、ダイアログ状態に関わらず受信した SIP リクエストを処理するために開始される)。UAS の TU は、481 (INVITE に対応するダイアログは Terminated 状態であることを TU が認識しているため)、または 500 (再送である F4 の CSeq が不適切な順番であるため)のエラー応答を F4 INVITE に対し送信する。(既に述べた通り、INVITE メッセージ F4(及び F1)はダイアログ内リクエストである。ダイアログ内リクエストは To タグを持つ。UAS の TU は、To タグを持つ INVITE の受信では新規ダイアログを開始しないことに注意すべきである。)

付録C.　CANCEL に対する UA の動作

本節では、Early 状態におけるダイアログ状態遷移に間接的に影響を与える CANCEL 動作について説明する。CANCEL は UAC のダイアログ状態に全く影響を与えない。しかし、CANCEL リクエストは ini-INVITE に与える大きな影響によりダイアログ状態遷移に間接的に影響を与える。UAS においては CANCEL リクエストが 487 応答送信の契機となるため、UAC による CANCEL リクエスト送信の場合よりも直接的な影響をダイアログに及ぼす。図 3 は Early 状態における UAS の動作を示す。このフロー図は説明上のものであり、実際のダイアログ状態遷移については図 1 及び 2 を参照されたい。

フローにおいて、実線はダイアログ状態遷移、点線は CANCEL に関連し、(r)は信号の受信、(s)は送信を示す。CANCEL のためのダイアログ状態は存在しないが、CANCEL 送受信の際の UA 動作の理解を促すためだけに、ここでは仮想的に Cancelled 状態を扱うこととする。

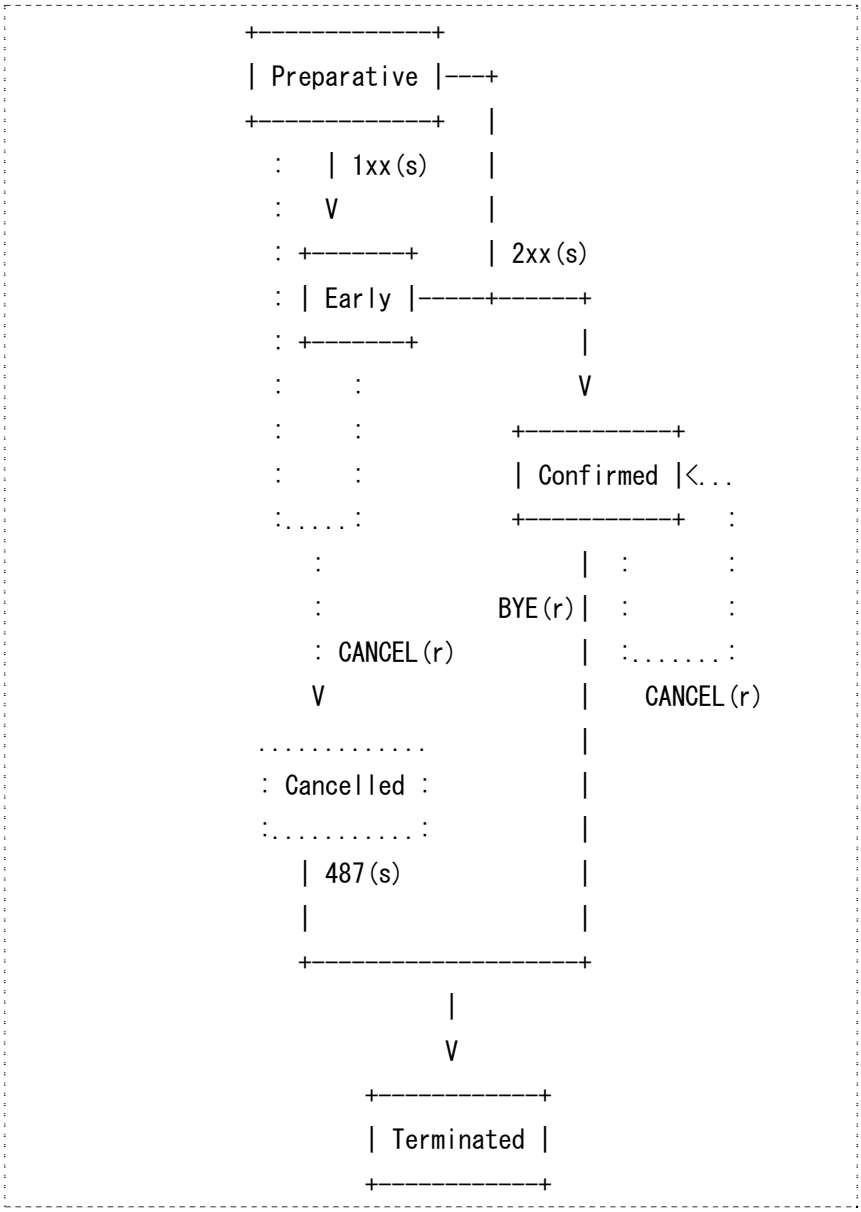


図 3：UAS の CANCEL フロー図

CANCELを受信する際のUASの状態により、二つの動作が存在する[9]。

一つはUASがEarly状態においてCANCELを受信する場合である。この場合、UASは直後にINVITEに対する487を送信し、ダイアログはTerminated状態に遷移する。

もう一つはUASがConfirmed状態においてCANCELを受信する場合である。この場合、UASはCANCELが対象とするINVITEに対し既に最終応答を送信済みであるため、ダイアログ状態遷移は生じない。(UACの動作により、Confirmed状態でCANCELを受信したUASはその直後にBYEを受信しTerminated状態に遷移すると想定されることに注意すること。しかし、UASの状態は実際にBYEを受信するまで遷移しない。)

[9] CANCELが有効に機能したかは、UAS側での判断に委ねられる。通常の実装において、UAS側がEarly状態で受け取ったCANCELは有効に機能するだろうが、既にUAS側がConfirmed状態で受け取ったCANCELは有効に機能しないだろう。

付録D.　Mortal 状態におけるリクエストに関する注釈

本節は、細心の配慮が必要なMortal状態におけるUAの動作について記述する。RFC 3261 [1]による原則に従い、全てのトランザクションは独立して完了することに注意する。

Mortal状態においてはBYEのみが許容され、INVITE dialog usageにおけるその他のメッセージはエラーで応答される。ただし、この状態におけるACKの送信とBYEのための認証手順は実行可能である。(複数dialog usageに関連したメッセージの処理については本文書の範囲外である。詳細はRFC 5057 [6]を参照。)

エラー応答に対する ACK はトランザクション層で処理されるため、ダイアログ状態には関係しない。エラー応答に対する ACK と異なり、2xx 応答に対する ACK は TU により新規に生成されたリクエストである。しかし、2xx に対する ACK とエラー応答に対する ACK は、処理は異なるが両方とも INVITE トランザクションの一部である(RFC 3261 [1]の 17.1.1.1 節)。そのため、Mortal 状態においても、INVITE トランザクションは ACK を含めて 3Way ハンドシェイクによって完了する。

実際の実装を考慮すると、Mortal 状態における 2xx 応答に対しての ACK の送信を可能にするため、UA は Mortal 状態が終了するまで INVITE dialog usage を保持する必要がある。Mortal 状態において INVITE に対する 2xx を受信した場合、2xx の再送の可能性に対処するために INVITE dialog usage の継続期間は 2xx の受信後 64*T1 まで延長される。(2xx の再送期間は 64*T1 であり、UA はこの期間再送を処理することに備えなくてはならない。) しかし、Mortal 状態における INVITE dialog usage は ACK 及び 2xx の送信のためにのみ保持されているため、UA はその他のリクエストに対してはエラー応答を送信する。

BYE の認証手順は Mortal 状態で処理される。認証は 401 または 407 応答で要求され、UAC は適切な証明書を含んだ BYE を再度送信する。また、UAS は要求した認証に対して再度送信された BYE を処理する。

付録E.　フォーキングと新規 To タグの受信

本節は、ini-INVITE に対して異なる To タグを持つ複数の応答を受信した際の TU の動作を詳解する。

SIPネットワーク内でINVITEがフォーキングされた場合、TUはini-INVITEに対し異なるToタグを持つ複数の応答を受信する可能性がある。(RFC 3261[1]の 12.1 節、13.1 節、13.2.2.4 節、16.7 節、19.3 節などを参照)。TUが異なるToタグを持つ複数の 1xx応答を受信した場合、元のDSMはフォークし新しいDSMインスタンスが生成される。結果として、複数のアーリーダイアログが生成される。

複数のアーリーダイアログのうちの一つが 2xx応答を受信した場合、通常通りConfirmed状態へと遷移する。その他のアーリーダイアログについては状態遷移が生じず、またそれらのセッション(アーリーメディア)は終了する。UACのTUは、最初の 2xx応答を受信後 64*T1 秒後にINVITEトランザクションを終了する。さらに、Established状態に遷移しない全てのmortalなアーリーダイアログは終了する(RFC 3261[1]の 13.2.2.4 節を参照)。「mortalなアーリーダイアログ」とは、あるアーリーダイアログが確立した際に、UAが終了するその他のアーリーダイアログのことを指す。

以下は、異なる To タグを持つ二つの 180 応答を受信し、一つのアーリーダイアログ(ダイアログ A)に対し 200 応答を受信する場合のシーケンス例を示す。シーケンス中の点線(..)はダイアログ B に対する影響を示す。

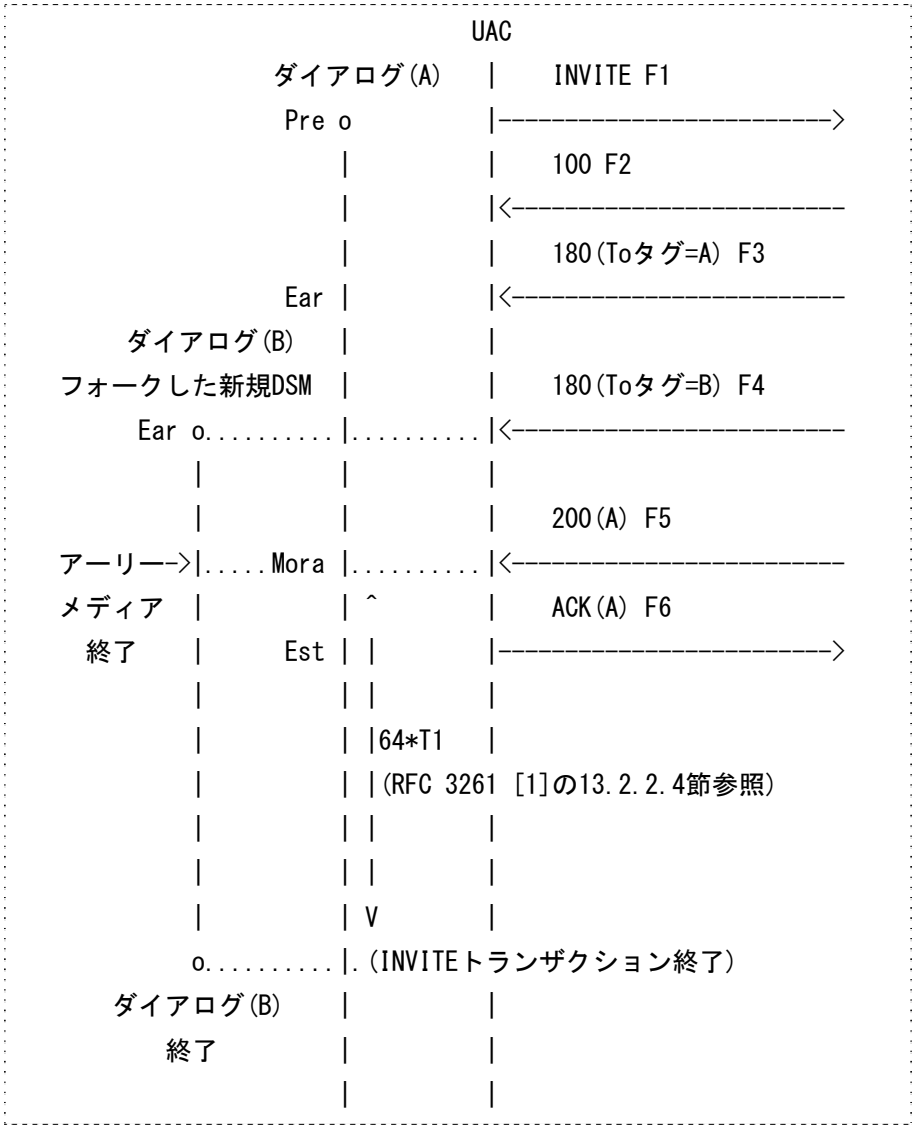


図 4：異なる To タグを持つ 1xx 応答の受信

上記の図は SIP TU 内の DSM を示す。異なる To タグを持った暫定応答(F4 180 (To tag=B))の受信により、DSM はフォークしアーリーダイアログ B が生成される。64*T1 秒後、ダイアログ A は 200 OK 応答を受信する。ダイアログ B は Established 状態に遷移せず終了する。

次に、異なる To タグを持つ複数の 2xx 応答を受信した場合の TU の動作を説明する。TU が最初に受信した 2xx 応答にマッチしなかった Mortal なアーリーダイアログが、64*T1 秒の INVITE トランザクション期間が終了する前にその To タグにマッチする 2xx 応答を受信した場合、その DSM は Confirmed 状態に遷移する。しかし、mortal なアーリーダイアログ上のセッションは TU が最初の 2xx を受信しダイアログを確立した時点で終了するため、mortal なアーリーダイアログのためのセッションは確立しない。従って、mortal なアーリーダイアログが 2xx 応答を受信した場合、TU は ACK を送信し、その直後に、DSM を終了するために通常は BYE を送信する。(UA が複数のダイアログの確立を望むような特殊なケースでは、TU はこの BYE 送信を行なわない可能性がある。)

最初のダイアログに対する200の受信後に二つ目のアーリーダイアログに対処することは電話のような典型的な機器にとって全く適切である。ここで示したことは典型的な有効動作であるが、唯一の動作ではないことに留意すべきである。機器の種類により異なった処理を行なうことが想定される。カンファレンスフォーカスが送信するINVITEがフォークした場合、全てのダイアログを受付け、ミックスする可能性がある。この場合、どのアーリーダイアログもMortalとして扱われない。

以下は、異なるToタグを持つ2つの180応答を受信し、それぞれのアーリーダイアログに対して200応答を受信する場合のシーケンス例を示す。

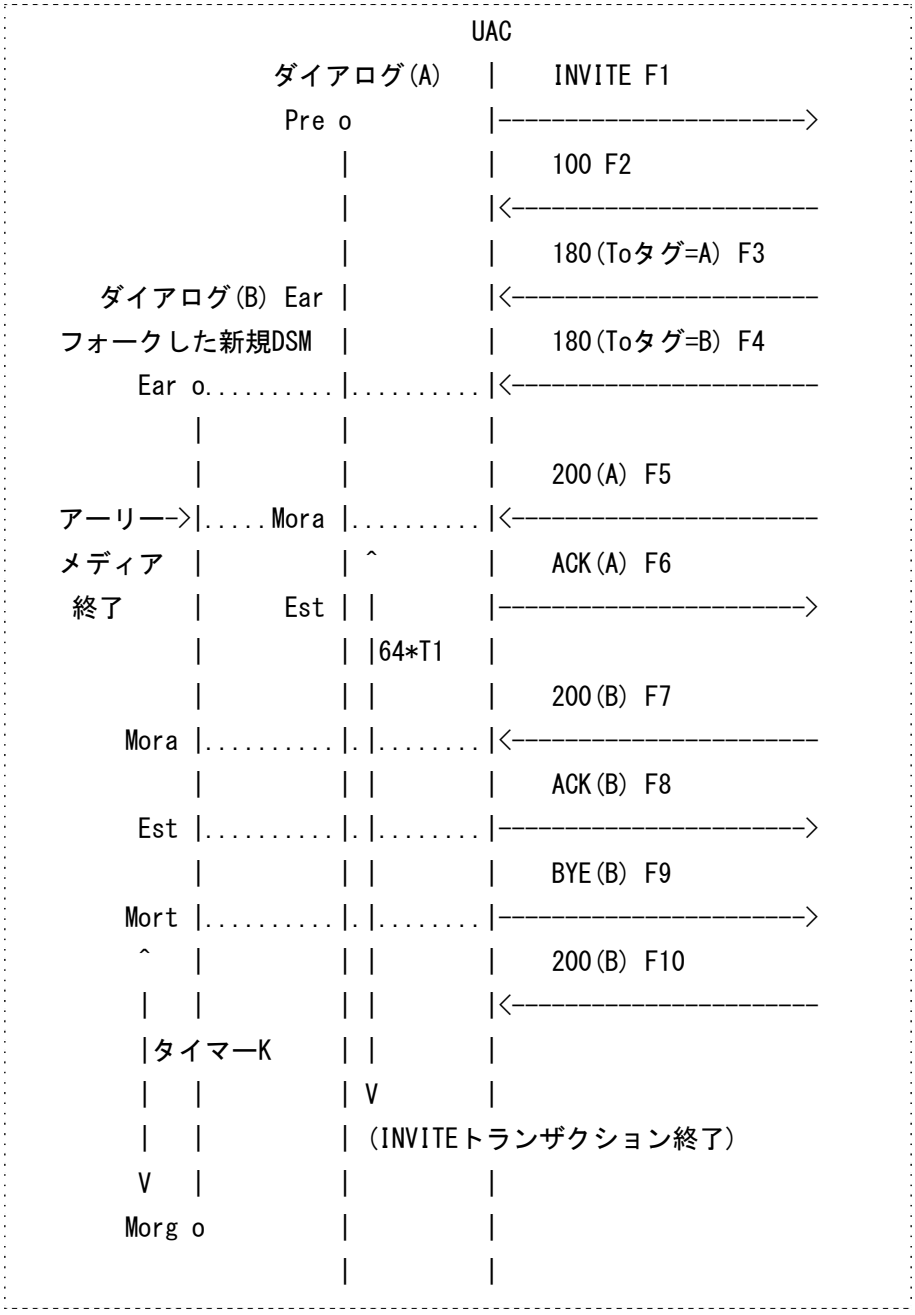


図5：異なるToタグを持つ1xx及び2xx応答の受信

以下は、64*T1秒のINVITEトランザクション期間が終了する前に、暫定応答未受信のまま異なるToタグを持つ複数の200応答をTUが受信するシーケンス例を示す。TUは、暫定応答を受信しない場合においても、2xx

応答を処理する必要がある(RFC 3261[1]の 13.2.2.4 節を参照)。この場合、DSMはConfirmed状態でフォークし、TUは 2xx応答に対してACKを送信した直後、通常はBYEを送信する。(UAが複数ダイアログの確立を望むような特殊なケースでは、TUはBYEを送信しないかもしれない。)

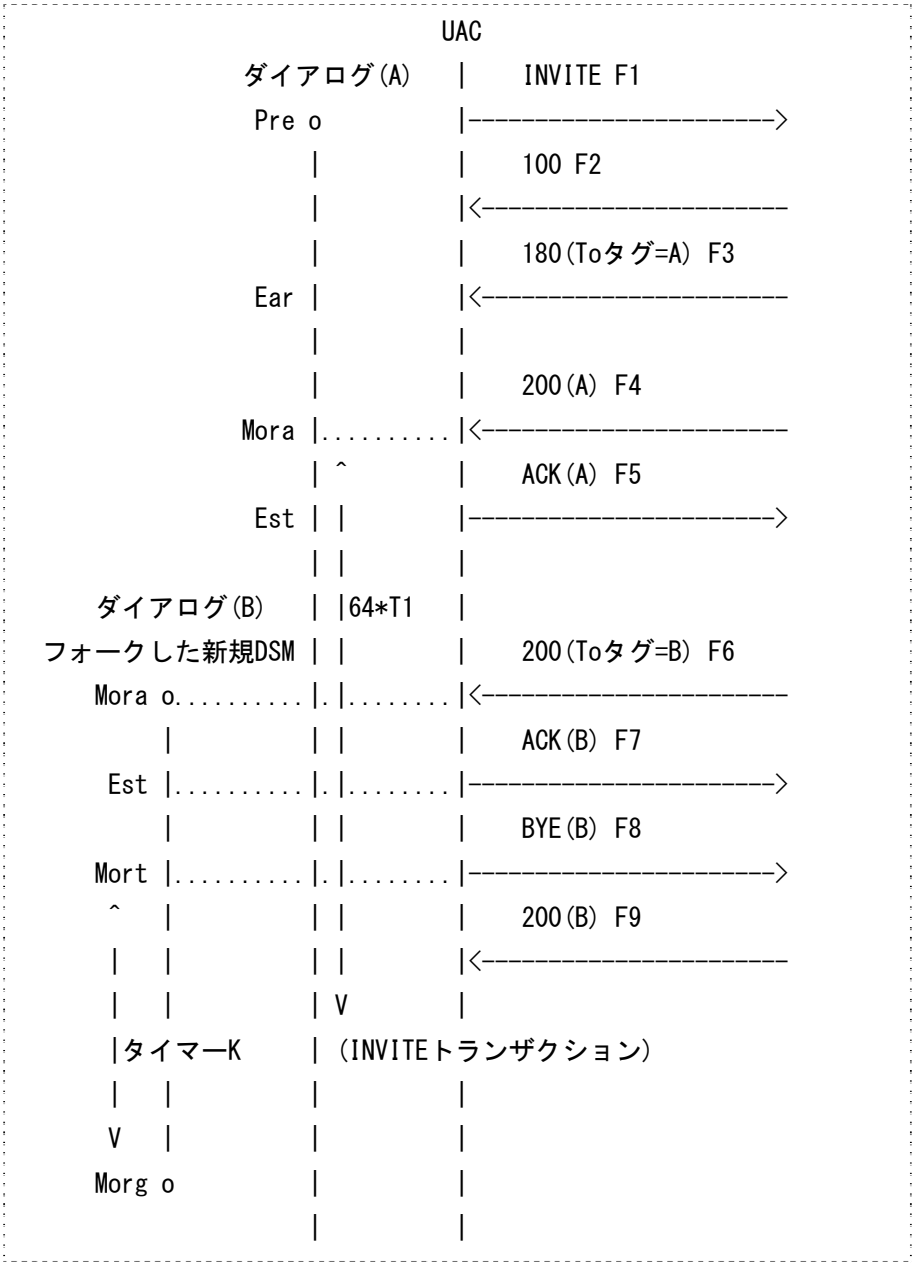


図 6：異なる To タグを持つ 2xx の受信

以下は、180 により 100relオプションタグ(RFC 3262[5])が要求される場合のシーケンス例を示す。

フォーキングプロキシが 100rel をサポートする場合、100rel が設定された Require ヘッダを含む暫定応答を透過的に UAC に送達する。100rel を含む暫定応答の受信により、UAC はアーリーダイアログ(B)を確立し、PRACK (Provisional Response Acknowledgement)を送信する。(ここでもまた、全てのトランザクションはそれぞれ独立して完了する。)

図 4 同様、アーリーダイアログ(B)は INVITE トランザクションの終了と同時に終了する。プロキシが 100rel

をサポートしない場合、暫定応答は通常の動作にて処理される(100rel を含む暫定応答はプロキシにより廃棄され UAC には送達されない)。

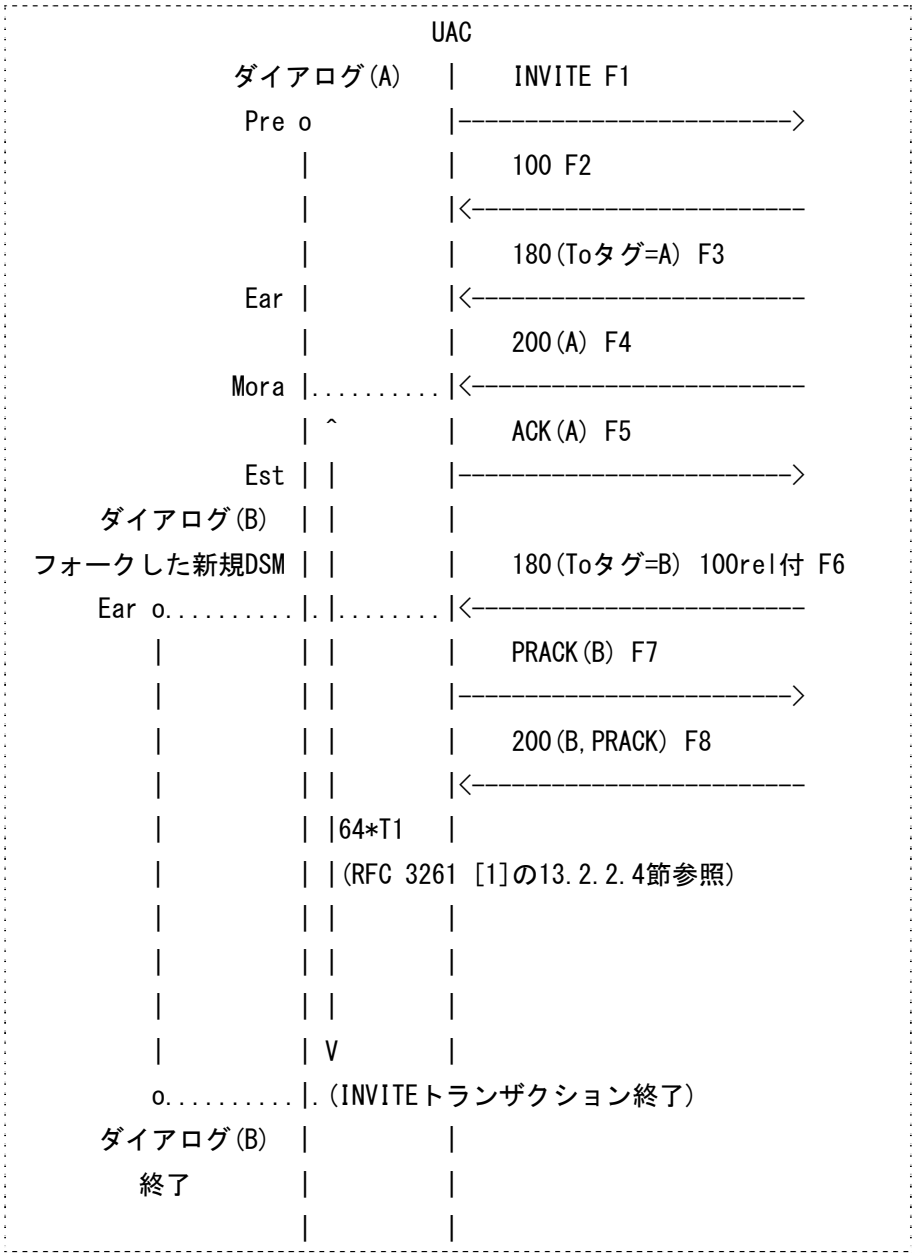


図 7：信頼性のある暫定応答メカニズム利用時の異なる To タグを持つ 1xx の受信

著者連絡先

長谷部未来
東日本電信電話株式会社
〒163-8019
東京都新宿区西新宿三丁目 19-2
EMail: hasebe.miki@east.ntt.co.jp

越湖淳
東日本電信電話株式会社
〒163-8019
東京都新宿区西新宿三丁目 19-2
EMail: j.koshiko@east.ntt.co.jp

鈴木康士
日本電信電話株式会社
〒180-8585
東京都武蔵野市緑町三丁目 9-11
EMail: suzuki.yasushi@lab.ntt.co.jp

吉川智之
東日本電信電話株式会社
〒163-8019
東京都新宿区西新宿三丁目 19-2
EMail: tomoyuki.yoshikawa@east.ntt.co.jp

Paul H. Kyzivat
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA　01719
US
EMail: pkyzivat@cisco.com