2004 年度 修士論文

状態正常化シグナリングに基づく SIP-ALG の設計と実装

提出日： 2005 年 2 月 2 日

指導：後藤滋樹教授

早稲田大学 大学院理工学研究科 情報・ネットワーク専攻
学籍番号： 3603U067-3

匂坂岳志

目次

図一覧

表一覧

第 1 章

序論

1.1 研究の目的

近年のインターネットアクセス回線のブロードバンド化，常時接続化に伴い， ISP や通信事業者が IP 電話サービスを提供し始めている．また 2002 年 11 月から，総務省が 050 から始まる 11 桁の IP 電話専用の番号を配布している．このように，一般の家庭や小規模なオフィスでも， IP 電話を利用できる環境が次第に整ってきている．従来の公衆電話と比較してコスト面で格安な IP 電話は，今後ますます普及していくものと期待されている．

IP 網内で従来の電話と同様なサービスを実現するためには，自端末と相手端末との間で適切なネゴシエーションを行うことで，通話セッションを確立する必要がある．このセッション確立の手続き，つまり呼制御 (シグナリング) を実現するためのプロトコルとして既に標準化され，製品にも応用されているものがある．それらのシグナリングプロトコルの中でも特に SIP は，HTTP をベースとしているためインターネットとの親和性が高く，軽量かつシンプルな仕様となっている．そのため SIP は IP 電話への応用をはじめ， IP 網内での様々な通信セッションの確立を実現するために，広く利用されるようになっている．

しかし SIP にはいくつかの問題点が存在する．その 1 つとして， SIP を利用した通信では，シグナリングがエンティティの主要な動作を決定する．そのため SIP による適切なシグナリングがないままに，通信途中で何らかのエラーが発生すると，ユーザ端末を含めシグナリングパス上に存在するサーバは，適切に初期状態へ移行することができない．これによって，エンティティは復旧までの間，様々なリソースを浪費することになってしまうということがある．従来まで，この問題に対する解決策は，エンティティに対して独自にタイマを持たせることでしか実現されていなかった．通信セッションに異常が発生しても，エンティティはそれを検知しなかったり，たとえ検知したとしても，リソースの解放をタイムアウトまで個別に待つ必要があった．

本論文では， SIP におけるこの問題を解決をするために，状態正常化シグナリングに基づく SIP-ALG を提案する．また提案方式に基づいて実装を行い，その有用性を検証する．

1.2 論文の構成

本論文は以下の章から構成される．

第 1 章 序論

研究の目的と本論文の構成について述べる．

第 2 章 SIP プロトコル

SIP の概要について述べる． SIP の特長と主な仕様について取り上げた後， SDP，および RTP/RTCP といった， SIP と併用するプロトコルの概要についても述べる．

第 3 章 SIP を利用した通信での問題

SIP を利用した通信での問題点について述べる．具体的には，セッション異常時からの復旧問題，および異種ネットワーク間でのセッション確立問題について述べる．後者については，既に提案されている解決手法についても述べる．

第 4 章 状態正常化シグナリングに基づく SIP-ALG

SIP を利用した通信での問題点を解決するための提案について述べる．本論文では，セッション異常時からの復旧問題に焦点を当てる．問題を解決するために状態正常化シグナリングに基づく SIP-ALG を提案し，その検討内容について述べる． SIP-ALG に対して，自律的シグナリング機能，およびメディアモニタリング機能を付加することによって，セッション異常時からの適切な復旧を図る．

第 5 章 検証実験

検討内容に基づいて実装を行った SIP-ALG の検証実験について述べ，その結果について考察する．実験では SIP ソフトフォンを利用して，セッションの途中で様々な異常を発生させる．その後トランザクションの正常形が構築され，セッションが適切に終了していることを確認する．考察では， UA に対して定義されているオートマトンを利用して，状態正常化シグナリングの有効性，および最も効率的な状態正常化の手法について検討する．

第 6 章 結論

本論文のまとめと今後の課題について述べる．

第 2 章

SIP プロトコル

2.1 SIP の特長

SIP (Session Initiation Protocol)[1] は，双方向型セッションの開始，変更，終了を行うための呼制御 (シグナリング) を実現するプロトコルである．セッションとは，関係する相手同士でのデータ交換をいう． SIP はインターネット技術の国際的な標準化組織である IETF (Internet Engineering Task Force) によって， 1999 年 3 月に RFC2543 として策定された． RFC2543 はその後， 2002 年 6 月に RFC3261 として改版されている．そして現在 2004 年でも， SIP は拡張仕様が検討，策定され，より実用性の高いプロトコルを目指して発展を続けている．

表 2.1に示すように， SIP の他にも IP 網上でのシグナリングプロトコルは複数のものが策定されている．その中でも ITU-T (International Telecommunication Union - Telecommunication sector) によって策定された H.323[2] は，策定時期が早かったことなどの理由から従来まで数多くの適用実績がある．今日 H.323 と肩を並べるまでに台頭を果たし，さらに今後主流となりつつある SIP には， H.323 にはない次のような特長がある．

インターネットとの親和性が高い

SIP は Web コンテンツの転送に使われる HTTP (HyperText Transfer Protocol) [3] や，メールの送信に使われる SMTP (Simple Mail Transfer Protocol)[4] といったプロトコルが備える特長を活かし， IP 網上での動作を基本として考え出されたものである．そのため SIP は H.323 と比較して，インターネットで利用される様々なプロトコルとの親和性が高いものとなっている．

軽量で拡張性が高い

H.323 が複数のプロトコルを組み合わせているため複雑になっているのに対して， SIP はシグナリングの手順をはじめシンプルな仕様であるため軽量である．また SIP は他のプロトコルと組み合わせることによって機能を付加することができ，拡張性も高い．

テキストベースで扱いやすい

H.323 がバイナリで記述されるのに対して，SIP はテキストで記述されるため，特に開発や運用保守に際してデバッグが比較的容易となっている．またテスト用の特別な機器を用意する必要もなく扱いやすい．

このような特長をもつ SIP は，IP 網上での音声通話 (固定電話，携帯電話) をはじめ，ビデオ通信やチャットなどへ応用がなされ，普及を始めている．

表 2.1: シグナリングプロトコルの比較

プロトコル	概要	
H.323	標準化団体	ITU-T
	標準化時期	初版: 1996 年／最新版: 2000 年
	目的	IP 網上での音声や動画像などマルチメディアによる通信を目的に策定
	データ形式	バイナリ
	通信方式	ピア・ツー・ピア
	長所	企業ネットワークへの適用実績が多い．機能が豊富
	短所	シグナリングの手順が複雑
MGCP	標準化団体	IETF
	標準化時期	初版: 1999 年／最新版: 2003 年
	目的	通信事業者が IP 網上での電話サービスの提供に利用することを目的に策定
	データ形式	テキスト
	通信方式	マスター・スレーブ
	長所	異なるメーカ製機器でも相互接続性が高い
	短所	拡張性に乏しい
H.248/ Megaco	標準化団体	ITU-T ／ IETF
	標準化時期	初版: 2000 年／最新版: 2002 年
	目的	H.323 システムをより大規模に対応させることを目的に策定
	データ形式	バイナリ／テキスト
	通信方式	マスター・スレーブ
	長所	網の管理や制御が容易．拡張性が高い
	短所	シグナリングの手順が複雑．実装製品が少ない
SIP	標準化団体	IETF
	標準化時期	初版: 1999 年／最新版: 2002 年
	目的	インターネットとの親和性の高い通信を目的に策定
	データ形式	テキスト
	通信方式	ピア・ツー・ピア
	長所	製品への実装が容易．実装製品や対応サービスが増加し，将来性が高い
	短所	拡張仕様に関する標準化が遅れている

2.2 SIP による通信の構成要素

SIP は HTTP と同様，クライアント／サーバ型のトランザクションモデルを採用している．つまり SIP による通信ではクライアントからリクエストが送出され，それに対するレスポンスがサーバから返されるという形式をとる．また SIP ではピアツーピア型の通信を行うため，単一のセッションの間でもエンドポイントの役割は変化するのが普通である．エンドポイントは，通信の内容に応じてクライアント (リクエストを送る側) にもなれば，サーバ (レスポンスを返す側) のいずれにもなる．

SIP による通信の構成要素は，図 2.1に示すようにユーザエージェント， SIP サーバ，ロケーションサーバの大きく 3 つに分けられる．

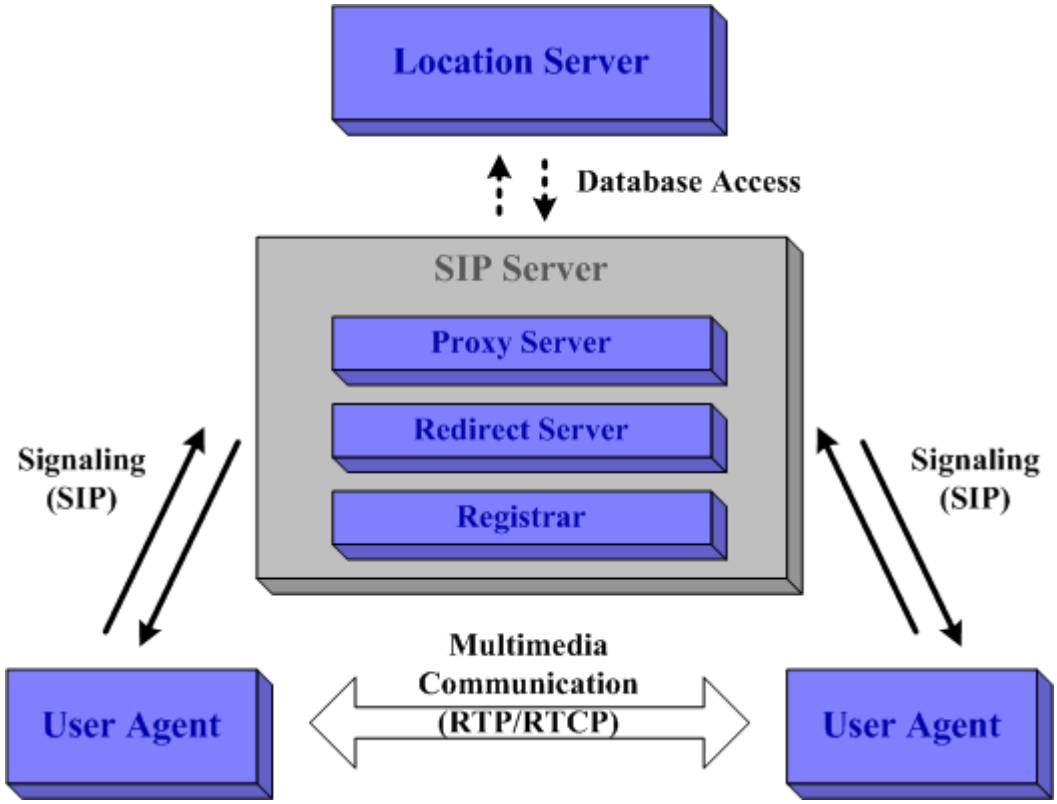

図 2.1: SIP による通信の構成要素

エンドポイント間では，サーバを経由して間接に，または相手同士で直接に， SIP によるシグナリングを行う．シグナリングによりセッションが確立した後，実際に音声や映像などメディアの交換が直接に行われる．

2.2.1 ユーザエージェント

SIP による通信のエンドポイントに相当するのが，ユーザエージェント (UA：User Agent) である．ユーザのフロントエンドとして，セッションを確立するための SIP メッセージの送受信やメディアの送受信などの処理を行う．

UA はユーザエージェントクライアント (UAC：User Agent Client) とユーザエージェントサーバ (UAS：User Agent Server) という 2 つの機能モジュールに分けられる． UAC はリクエストを開始する機能モジュールである．それに対して， UAS は受け取ったリクエストに対するレスポンスを生成する機能モジュールである． UA には必ず UAC と UAS の両方の機能が含まれており，単一のセッションであっても，どちらの機能も利用される．

UA の実装例としては， SIP 電話端末，パソコンや PDA 上で動作するソフトウェア (ソフトフォン) が挙げられる．図 2.2にソフトフォンの例を示す．

図 2.2: SIP ソフトフォン

2.2.2 SIP サーバ

SIP サーバは，UA 間で行われるシグナリングの経路上に位置し，セッション確立を仲介する処理を行う．SIP サーバはさらにプロキシサーバ，リダイレクトサーバ，レジストラの 3 つに分けられる．

プロキシサーバ

プロキシサーバ (Proxy Server) は，UA の代理として SIP メッセージを中継する．プロキシサーバは，SIP メッセージを受け取るとロケーションサーバに問い合わせを行うことで宛先を判断し，リクエストを UAS に，レスポンスを UAC に転送する[1]．ネットワークの構成に応じて，複数のプロキシサーバが経由されることもある．

リダイレクトサーバ

リダイレクトサーバ (Redirect Server) は，UA やプロキシサーバからリクエストを受け取ると，リダイレクトレスポンス (ステータスコードが 3xx のレスポンス) を返す．このレスポンスには UA やプロキシサーバがリクエストを送り直すべき宛先が記述される．プロキシサーバとは異なり，リダイレクトサーバは SIP メッセージを転送することはない．

レジストラ

レジストラ (Registrar) は，UA の現在位置を登録するリクエスト (REGISTER リクエスト) を受け取り，ロケーションサーバのデータベースに反映する．この情報は同一の管理ドメイン下にあるプロキシサーバやリダイレクトサーバによって，SIP メッセージの適切なルーティングを行うために利用される[2]．

2.2.3 ロケーションサーバ

ロケーションサーバ (Location Server) は，レジストラから登録される UA の情報を保持し，それをプロキシサーバやリダイレクトサーバによって利用されるデータベースとして提供する．SIP ではロケーションサーバへのアクセス方法を規定していないため，レコードに対する操作には別プロトコルが利用される．

[1] レスポンスの宛先の決定には，SIP メッセージ内の Via ヘッダフィールド (SIP メッセージのシグナリングパスを記録する) が利用される．プロキシサーバを経由する限り，レスポンスはリクエストと逆の経路をたどる．

[2] 一般に，レジストラはプロキシサーバやリダイレクトサーバと同じホスト上で動作する．

2.3 SIP メッセージ

2.3.1 SIP URI

SIP による通信では，リソースの特定に URI (Uniform Resource Identifier) を利用する．SIP で新たに定義された URI には SIP URI と SIPS URI の 2 つがあり，この他にも tel URI が利用できる．ここで SIPS URI とは SIP のセキュアな URI であり，TCP のようなコネクション指向のトランスポートプロトコルを利用する通信に対して TLS (Transport Layer Security)[5] による暗号化が施される．RFC3261[1] では，SIP のトランスポートプロトコルとして TCP と UDP が必須とされている．しかし実際には，プロキシのようにシグナリングパス上のサーバにとって，長時間存続するコネクションを維持することが困難な場合がある．そのため実際には多くの場合に，UDP がトランスポートプロトコルとして利用される．このような理由から，本論文でも SIP のトランスポートプロトコルとして UDP を想定する．

SIP URI と SIPS URI の形式は RFC2396[6] のガイドラインに従って定義されている．SIP URI の一般的な形式は次の通りである．SIPS URI はプロトコルスキームが”sips” になることを除けば SIP URI と同じ形式である．

表 2.2: SIP URI の一般形式

sip:user:password@host:port;uri-parameters?headers

下線で示した部分は必ず指定しなくてはならない．各要素の意味は次の通りである．

user　宛先となるホストでのユーザ ID．ユーザ ID を指定せずに単にホストを宛先とする場合には，’@’ までを省略することができる．宛先のホストが電話番号を扱える場合には，電話番号を指定することもできる．

password　ユーザ ID に関連付けられたパスワード．ただし認証情報を平文で記述することになるため，RFC3261[1] では使用を推奨していない．実装上は “user:password” を単一の文字列として扱うことができる．

host　SIP リソースを提供するホスト．FQDN (Fully Qualified Domain Name) または IP アドレスのいずれでも指定することができるが，FQDN を利用することが推奨される．

port　リクエストの宛先ポート番号．UDP，TCP，および SCTP (Stream Control Transmission Protocol)[7] を使用する”sip” スキームでは 5060，TCP，および TLS 上の TCP を利用する”sips” スキームでは 5061 がデフォルトで利用される．

uri-parameters　URI パラメータ．”< パラメータ名 >=< パラメータ値 >” の形式で，各組を’;’ で区切ることで複数指定することができる．定義済みパラメータ名には transport， user， method， maddr， ttl， lr などがあり，独自に追加定義することも可能である．

headers　リクエストのヘッダフィールド．”< ヘッダ名 >=< 値 >” の形式で，各組を’&’ で区切ることで複数指定することができる．特別なヘッダ名として body が定義されており，この値は SIP メッセージのボディとなる．

2.3.2　メッセージの構成

SIP メッセージはテキストベースのプロトコルであり，文字コードセットとして UTF-8 (Universal Transformation Format, 8-bit Form) [8] を利用する． UTF-8 とは，文字コードポイント (文字の符号化用に割り当てられた整数値) を， ASCII との互換性を持たせつつ，他の文字を可変長のバイト列で符合化する方法である[3]． Unicode または USC (Universal Character Set) で符合化された文字を，インターネットで転送する際の変換に利用される．

SIP メッセージは，クライアントからサーバへのリクエスト，またはサーバからクライアントへのレスポンスのいずれかである．これらは詳細なシンタックスは異なるが，共に RFC2822[9] の基本フォーマットを利用する．これらのメッセージは表 2.3に示すように，スタートライン，複数行のヘッダフィールド，ヘッダフィールドの終了を表す空白行，およびオプションのボディからなる．メッセージ内の各行は CRLF で区切られる．

表 2.3: SIP メッセージの構造

構成要素	内容
スタートライン	リクエストとレスポンスを区別するヘッダ
ヘッダフィールド	複数行からなる，ヘッダとそれに対する値の組
空白行	ヘッダフィールドの終わりを示す．ボディの有無に関わらず必須
ボディ	オプションの付加情報．セッション情報の記述などに利用される

スタートラインは，メッセージがリクエストの場合に特にリクエストラインと呼ばれる．またメッセージがレスポンスの場合には特にステータスラインと呼ばれる．

2.3.3　リクエスト

リクエストは，メッセージの先頭行にリクエストライン (Request-Line) を持つ．リクエストラインは 1 つの空白文字 (SP) で区切られた，メソッド名 (Method)，リクエスト URI (Request-

[3]ASCII は従来通り 1 バイト，他の文字は 2 または 3 バイトで符合化される．

URI)，およびプロトコルのバージョン (SIP-Version) からなる．リクエストラインの一般形式を次に示す．

表 2.4: リクエストラインの一般形式

Request-Line = Method SP Request-URI SP SIP-Version CRLF

各要素の意味は次の通りである．

Method　リクエストの種別を表す．RFC3261[1] では，UA のコンタクト情報を登録するための REGISTER，セッションを構築するための INVITE，ACK，CANCEL，セッションを終了するための BYE，およびサーバに能力を問い合わせるための OPTIONS の 6 種類のメソッドが定義されている．また表 2.5に示すように，これらのメソッドに加えて，付随するその他の RFC では拡張メソッドも定義されている．

表 2.5: SIP メソッド

メソッド	内容	RFC
INVITE	セッション参加リクエスト	RFC3261[1]
ACK	INVITE に対する最終レスポンスの確認	RFC3261[1]
BYE	セッションの終了	RFC3261[1]
CANCEL	進行中のセッションのキャンセル	RFC3261[1]
REGISTER	ユーザの URI の登録	RFC3261[1]
OPTIONS	オプション機能や能力についての問い合わせ	RFC3261[1]
INFO	ミッドコールシグナリング	RFC2976[10]
NOTIFY	要請されているイベント通知の伝送	RFC2848[11]
SUBSCRIBE	イベント通知の要請	RFC2848[11]
UNSUBSCRIBE	イベント通知の終了	RFC2848[11]
UPDATE	SDP によるメディアネゴシエーションの更新	RFC3311[12]
MESSAGE	メッセージボディを利用した IM の伝送	RFC3428[13]
REFER	別の URI への呼の転送	RFC3515[14]
PRACK	暫定レスポンスに対する確認リクエスト	RFC3262[15]

Request-URI　リクエストの宛先となるユーザやサービスのロケーションを表す．SIP URI，SIPS URI，または RFC2396[6] で定義される一般的な URI として記述される．

SIP-Version　メッセージの記述に利用する SIP プロトコルのバージョンを表す．RFC3261[1] に準拠する場合は，“SIP/2.0” となる．

2.3.4 レスポンス

レスポンスは，メッセージの先頭行にステータスライン (Status-Line) を持つ．ステータスラインは 1 つの空白文字 (SP) で区切られた，プロトコルのバージョン (SIP-Version)，ステータスコード (Status-Code)，および関連付けられたフレーズ (Reason-Phrase) からなる．ステータスラインの一般形式を次に示す．

表 2.6: ステータスラインの一般形式

Status-Line = SIP-Version SP Status-Code SP Reason-Phrase CRLF

各要素の意味は次の通りである．

SIP-Version　リクエストラインの場合と同様，メッセージの記述に利用する SIP プロトコルのバージョンを表す．

Status-Code　100 番台から 600 番台までの 3 桁の整数からなる，リクエストに対する結果を表すコード．表 2.7に，各区分のステータスコードの意味を示す．

表 2.7: SIP ステータスコード

ステータスコード	意味	内容
1xx	暫定	リクエストが受信され，そのリクエストの処理を継続中
2xx	成功	リクエストが成功
3xx	リダイレクト	ユーザの新たな場所の情報，代替サービスに関する情報を与える
4xx	クライアントエラー	リクエストに誤りが含まれているか，指定されたサーバではこのリクエストを実行できない
5xx	サーバエラー	サーバがリクエストの実行に失敗
6xx	グローバルエラー	リクエストはどのサーバでも実行できなかった

ここで 1xx はプロビジョナル (暫定) レスポンス， 2xx 〜 6xx はファイナル (最終) レスポンスに分類される．

Reason-Phrase　テキストからなる，リクエストに対する結果についての短い説明． Status-Code がオートマトンによる使用を意図しているのに対して， Reason-Phrase は人間のユーザによる使用を意図したものである．

2.3.5 シグナリングの構成単位

SIP によるシグナリングでは，連続する複数のメッセージによって意味のあるまとまりが形づくられる．本論文ではその中でも特に重要なトランザクションとダイアログについて触れる．

トランザクション

トランザクションとは，1 つのリクエストとそれに対するレスポンス (ゼロ個以上の暫定レスポンス，および 1 つ以上の最終レスポンス) の組である．最終レスポンスが 2xx でない場合に限り，トランザクションには ACK も含まれる．最終レスポンスが 2xx である場合には，ACK はトランザクションの一部とは見なされない．

ダイアログ

ダイアログとは，トランザクション進行中に持続される，2 つの UA 間でのピアツーピアの関係である．INVITE リクエストに対する，100 Trying を除く 1xx レスポンス，および 2xx レスポンスによって構築される．前者の手続きによって構築されたダイアログは early ダイアログと呼ばれ，後者の手続きによって構築されたダイアログは confirmed ダイアログと呼ばれる[4]．

SIP を利用した通信では，必ずしもサーバ側で状態を保持する必要がないため，端末側でそれぞれの結び付きを一意に保持しなくてはならない．ダイアログはこれを実現するためのものであり，次の 3 つの要素によって定義される．

- リモート tag (相手端末が作成した tag)
- ローカル tag (自端末が作成した tag)
- 呼識別子

これらは，具体的には後述する To tag，From tag，および Call-ID に関連付けられ，3 つの要素を組み合わせたものはダイアログ ID と呼ばれる．

[4]early ステートにあるダイアログは，2xx レスポンスを受け取ることによって confirmed ステートに移行する．

2.3.6 ヘッダフィールド

SIP メッセージには，スタートラインに続いて ”< ヘッダ名 >: < ヘッダ値 >” の形式で複数のヘッダフィールドが続く．ヘッダフィールドは数多くの種類が定義されているが，本論文では特に利用頻度の高いものを選択して取り上げる．ヘッダフィールドの中でも， To， From， Via， Max-Forwards， CSeq，および Call-ID の 6 種類は，すべての SIP リクエストメッセージに必須である．

To

To ヘッダフィールドは，リクエストの論理的な受信者を指定する．オプションの表示名 (受信者名) は，ユーザインタフェースによって表示されることを意図している． tag パラメータ[5]は，ダイアログの構築に利用される．このヘッダフィールドのコンパクトフォームは t である．

表 2.8: To ヘッダフィールド

To: The Operator <sip:operator@cs.columbia.edu>;tag=287447
t: sip:+12125551212@server.phone2net.com

From

From ヘッダフィールドは，リクエストのイニシエータ (生成元) を示す． To ヘッダフィールド同様，オプションの表示名 (送信者名) は，ユーザインタフェースによって表示されることを意図している．ただしクライアントの身元が隠される場合には，表示名として”Anonymous” を使用するべきとされている． tag パラメータ[6]は， To ヘッダフィールド同様，ダイアログの構築に利用される．このヘッダフィールドのコンパクトフォームは f である．

表 2.9: From ヘッダフィールド

From: “A. G. Bell” <sip:agb@bell-telephone.com> ;tag=a48s
From: sip:+12125551212@server.phone2net.com;tag=887s
f: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8

[5]To tag は， UAS によって 100 Trying を除く 1xx レスポンス，および 2xx レスポンスで生成される．
[6]From tag は， UAC によって INVITE リクエストで生成される．

Via

Via ヘッダフィールドは，リクエストが経由したパスを示す．これはレスポンスが返される時にたどるべきパスとなる．”z9hG4bK”(マジッククッキー) で始まる branch パラメータは，トランザクション ID として，ループ検知のためにプロキシサーバによって利用される． Via ヘッダフィールドは，この他にも maddr， ttl， received などのパラメータを含むことができる．このヘッダフィールドのコンパクトフォームは v である．

表 2.10: Via ヘッダフィールド

```
Via: SIP/2.0/UDP erlang.bell-telephone.com:5060;branch=z9hG4bK87asdks7
Via: SIP/2.0/UDP 192.0.2.1:5060 ;received=192.0.2.207;branch=z9hG4bK77asjd
```

Record-Route

Record-Route ヘッダフィールドは，ダイアログ中のそれ以降のリクエストを，指定したプロキシを経由して転送させるために，プロキシによってリクエストに挿入される．

表 2.11: Record-Route ヘッダフィールド

```
Record-Route: <sip:server10.biloxi.com;lr>, <sip:bigbox3.site3.atlanta.com;lr>
```

Route

Route ヘッダフィールドは， Record-Route ヘッダフィールドにリストされたプロキシのセットを経由してリクエストを転送させるために利用される．

表 2.12: Route ヘッダフィールド

```
Route: <sip:bigbox3.site3.atlanta.com;lr>, <sip:server10.biloxi.com;lr>
```

Max-Forwards

Max-Forwards ヘッダフィールドは，リクエストをダウンストリームサーバに転送できるプロキシの数を制限するために利用される． Max-Forwards ヘッダフィールドの値は，リクエストが転送されることを認められている残り回数を示す 0 〜 255 の整数である．このカウントは，そのリクエストを転送する各プロキシでデクリメントされる．推奨される初期値は 70 である．

表 2.13: Max-Forwards ヘッダフィールド

```
Max-Forwards: 6
```

Contact

Contact ヘッダフィールドは，ユーザがプロキシを経由せずに直接通信するための URI 情報を示す． RFC3261[1] ではパラメータとして， q (品質値)，および expires (有効期限値) を定義している．このヘッダフィールドのコンパクトフォームは m である．

表 2.14: Contact ヘッダフィールド

```
Contact: “Mr. Watson” <sip:watson@worcester.bell-telephone.com>
        ;q=0.7; expires=3600,
        “Mr. Watson” <mailto:watson@bell-telephone.com> ;q=0.1
m: <sips:bob@192.0.2.4>;expires=60
```

Expires

Expires ヘッダフィールドは，メッセージまたはコンテンツがそれ以降期限切れになる相対時間を示すが，正確な意味はメソッドに依存する． Expires ヘッダフィールドの値は，リクエストの受信時から計測された 0 から $2^{32} - 1$ の間の，秒を表す 10 進整数である．

表 2.15: Expires ヘッダフィールド

```
Expires: 5
```

CSeq

CSeq ヘッダフィールドは， 10 進数のシーケンス番号とリクエストのメソッド名の組み合わせで構成される．シーケンス番号は 32 ビットの符合なし整数で表現される． CSeq ヘッダフィールドの値によって，トランザクションが順序付けられる．そのため新規リクエストとリクエストの再送を区別するために利用することができる．

CSeq シーケンス番号は，原則として新しいリクエストごとにインクリメントされる．ただし例外的に ACK と CANCEL リクエストに対しては，振り方が異なる． ACK リクエストのシー

ケンス番号は，確認応答の対象となる INVITE リクエストのそれと同じである．また CANCEL リクエストのシーケンス番号は，キャンセルの対象となるリクエストのそれと同じである．

表 2.16: CSeq ヘッダフィールド

CSeq: 4711 INVITE

Call-ID

Call-ID ヘッダフィールドは，特定の招待または特定のクライアントのすべての登録を一意に識別する． Call-ID の値はダイアログの構築に利用される．このヘッダフィールドのコンパクトフォームは i である．

表 2.17: Call-ID ヘッダフィールド

Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
i:f81d4fae-7dec-11d0-a765-00a0c91e6bf6@192.0.2.4

Content-Type

Content-Type ヘッダフィールドは，受信者に送られたメッセージボディのメディアタイプを示す．メディアタイプのエレメントは IANA (Internet Assigned Numbers Authority) によって登録されている． Content-Type ヘッダフィールドは，ボディが空でない場合には必ず存在しなければらない．このヘッダフィールドのコンパクトフォームは c である．

表 2.18: Content-Type ヘッダフィールド

Content-Type: application/sdp
c: text/html; charset=ISO-8859-4

Content-Length

Content-Length ヘッダフィールドは，受信者に送られたメッセージボディのサイズを，オクテット単位の 10 進数で示す．ここでメッセージボディのサイズにはヘッダフィールドとボディを分ける CRLF は含まない．ボディが存在しない場合， Content-Length ヘッダフィールドの値は 0 に設定されなければならない．このヘッダフィールドのコンパクトフォームは l である．

表 2.19: Content-Length ヘッダフィールド

Content-Length: 349
l: 173

2.4 各種プロトコルとの連携

SIP を利用した通信では，SIP 以外にも複数のプロトコルが連携して動作する．ここではその中でも，メディアセッションのセットアップに利用される SDP，およびメディアによる通信の際に利用される RTP/RTCP について触れる．

2.4.1 SDP

SDP (Session Description Protocol)[16] [17] は，もともとマルチキャスト実験ネットワーク MBone (Multicast Backbone) 上で，ユーザをマルチメディア会議に召集したり，ユーザに対してセッションに関する情報を通知するために策定されたプロトコルである． SDP は SIP と同様にテキストベースであり，次のような情報が記述される．

- **セッション記述**
  セッションの名前， ID，目的など，セッションを識別する情報
- **時間記述**
  開始時刻，終了時刻，繰り返し回数など，セッションの有効期間に関する情報
- **メディア記述**
  IP アドレスやポート番号，メディアの種別やコーデックなど，メディアに関する情報

2.4.2 RTP/RTCP

RTP (Real-time Transport Protocol)[18] は，音声や映像のような実時間性 (リアルタイム性) が要求されるデータを利用して，多人数マルチメディア会議のようなサービスをインターネットで実現することを目的に設計されたプロトコルである．現在では IP ネットワークでメディアをリアルタイムに転送する目的で広く利用されている．

一方， RTCP (RTP Control Protocol)[18] は， RTP を制御するプロトコルとして RTP と共に規定されているプロトコルである． RTCP は上位アプリケーションに対する， RTP パケットのフロー制御通知や，送受信端末間でのクロック同期などを行う．

RTP/RTCP はいずれもトランスポートプロトコルとして UDP を利用する．また，いずれもバイナリで記述される．

第 3 章

SIP を利用した通信での問題

3.1 セッション異常時からの復旧

一般に SIP エンティティ[1]は，トランザクション，またはセッションの開始から終了までを単位として，呼の状態を保持するように — つまりステートフルに — 実装される．また一方でエンティティの主要な動作は，シグナリングによって決定される．つまりエンティティは，シグナリングがなければ，その動作や状態を変化させることができない．

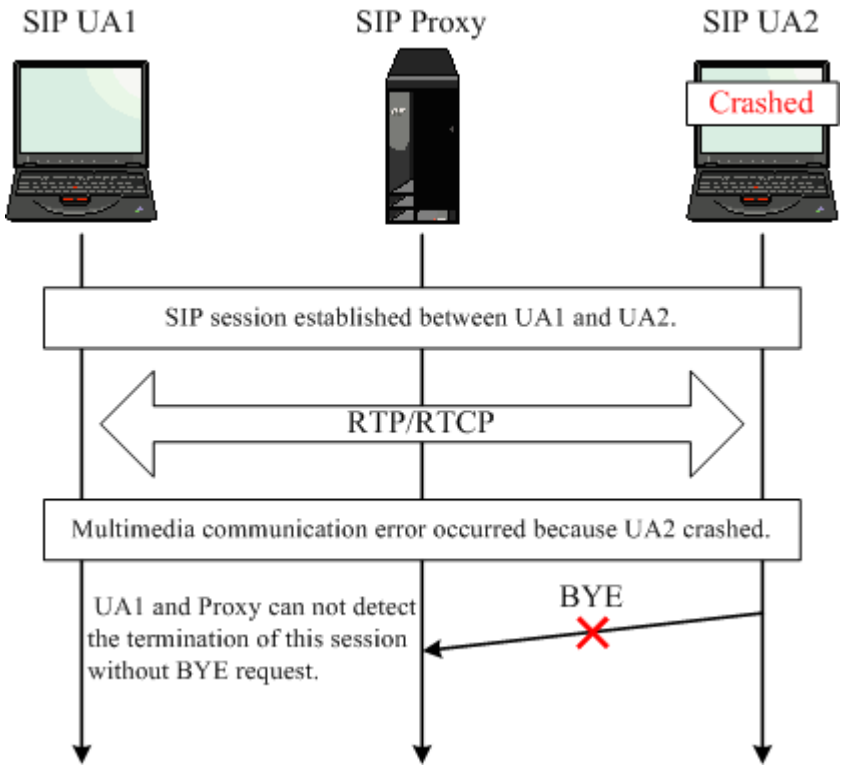

図 3.1: セッション異常時からの復旧に関する問題

[1]UA やプロキシのように， SIP による通信の構成要素となるホストを指す．

これは SIP による通信の途中で，UA が強制終了したり，ユーザ端末がクラッシュするなど，メッセージの再送を行っても復旧できないエラーが発生したときに問題となる．つまりこのとき SIP による適切なシグナリングがないため，パス上のエンティティは初期状態に移行するための通知を受けることができない．これによって，エンティティは復旧までの間，様々なリソースを浪費することになってしまう．

例えば図 3.1のような場合を考える．UA1 と UA2 との間ではセッションが確立し，メディアによる通信を開始しているものとする．ここで UA2 がクラッシュし，セッション終了リクエストである BYE を生成せずに異常終了してしまうと，UA1 とプロキシはこのセッションの終了を検知することができない．その結果，UA1 とプロキシは，それぞれこのセッションのために確保したリソースを，無効となった後も保持し続けなければならなくなってしまう．

従来まで，この問題に対する解決策は，エンティティに対して独自にタイマを持たせることでしか実現されていなかった [1]．そのため前述のエラーによってセッションに異常が発生してしまうと，エンティティはリソースの解放をタイムアウトまで個別に待つ必要があった．またエンティティに対して定義されているオートマトンには，タイマが設けられていない状態も存在していた．そのため特定の状態で前述のエラーが発生してしまうと，エンティティはそれを検知することができず，リソースを解放することができなかった．

3.2 異種ネットワーク間でのセッション確立

SIP によるシグナリングでは，メッセージの受信ホストがレスポンスの宛先を判断するために IP および TCP/UDP ヘッダ部分だけでなく，ペイロード部分にも送信ホストのアドレスが記述される [1]．しかし通信を行うエンティティが，NAT を介して異なるアドレス体系[2]に存在する場合には，これが障害となる．

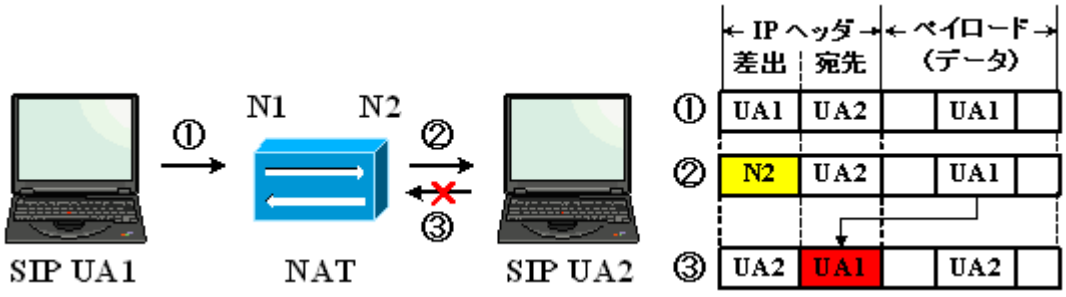

図 3.2: 異種ネットワーク間でのセッション確立に関する問題

[2]ここでは自分自身と通信相手がそれぞれのネットワークアドレスを利用して，直接通信することができない環境を指す．具体的には IPv4 プライベート / グローバル間，IPv4/IPv6 間ネットワークがこれに当たる．

通常の NAT では，パケットを転送する際に，IP および TCP/UDP ヘッダ部分に記述されるアドレスを書き換えるが，ペイロード部分ついては関知しない．そのため図 3.2に示すように，リクエスト送信側の UA が，レスポンスの宛先としてペイロード内にローカルなアドレス[3]を記述してしまうと，NAT で変換されないまま転送される．その結果，受信側の UA は適切にレスポンスを返すことができなくなってしまう．つまり，アドレス体系の異なるネットワーク内に存在する 2 つのエンティティは，通常の NAT を経由するだけでは通信を行うことができない．

この問題の解決策として，SIP を利用した通信で NAT を実現するアプローチには複数のものが存在する．それぞれにメリットやデメリットがあり，現時点 (2004 年) では必ずしも完成されたものではない．本論文では SIP で採用されている手法のうちのいくつかを取り上げる．

3.2.1 UPnP

UPnP (Universal Plug and Play)[19] とは，UPnP Forum で規定された規格で，PC や AV 機器など家庭内のいろいろな機器をネットワークを通じて接続し，相互に機能を提供しあう技術である．UPnP を利用すると，プライベートネットワーク内のホストから UPnP 対応のルータに対して，ルータのもつパブリックな IP アドレスの割り当てや，ポート番号のマッピングを要求することができる．

ルータに加えて SIP を利用するホストでも UPnP に対応すれば，プライベートネットワークとパブリックネットワークとの間で，SIP を利用して通信することができるようになる．ただし現時点では UPnP に対応したルータ (UPnP サーバ) は数多く存在するものの，UA やプロキシなど SIP アプリケーションで UPnP に対応したもの (UPnP クライアント) は非常に少ない．

3.2.2 STUN

STUN (Simple Traversal of UDP through NATs)[20] とは，アプリケーションがパブリックネットワークとの間の NAT，およびファイアウォールの存在とタイプを発見することを可能にする軽量プロトコルである．STUN は，NAT がプライベートネットワーク内のホストに割り当てたパブリックな IP アドレスやポート番号を，アプリケーションに通知することができる．

ただしこの方法を利用するためには，パブリックネットワーク内に STUN サーバが必要となり，プライベートネットワーク内の SIP を利用するホストでも STUN に対応する必要がある．これに加えて，途中に介在する NAT の動作がコーン型[4]でなければならない．つまり，NAT の

[3]具体的には SIP ヘッダ部の Via，Record-Route，Route および Contact ヘッダフィールド，また SIP ボディ部のメディアホストのアドレスがこれに当たる．

[4]最もセキュリティレベルの高い Symmetric NAT を除く，Full cone NAT，Restricted cone NAT，および Port-restricted cone NAT を指す．

内側の同じ IP アドレス，ポート番号から外側にパケットを送信する際に，NAT の外側に割り当てられる IP アドレス，ポート番号が常に同じである必要がある．

3.2.3 B2BUA

B2BUA (Back to Back User Agent)[1] とは，リクエストを受け取り，UAS としてそれを処理する論理的なエンティティである．同時に，そのリクエストに対してどのようなレスポンスを返すかを決定するために UAC として動作し，リクエストを生成する．つまり，プライベートな UA とパブリックな UA が結合されたもので，それぞれのネットワークに対して独立した UA として動作する．プロキシとは違い，ダイアログの状態を保持し，それが確立したすべてのリクエストに関与しなければならない．動作に関する明示的な定義はされていない．

この方法を利用するメリットとしては，UA やプロキシでは特別な処理を一切行う必要がない点がある．逆にデメリットとしては，B2BUA に対して負荷が集中するため，高い処理性能が求められるという点がある．

3.2.4 ALG

ALG (Application Level Gateway)[21] [22] [23] [24] [25] [26] とは，NAT またはファイアウォール上で通過する SIP メッセージを解析し，メッセージ内のプライベートなアドレスとパブリックなアドレスの変換を一括して行う方法である．動作に関する明示的な定義はされていないが，多くの検討がされてきている．

ALG は B2BUA と同様，サーバサイドに限定したアプローチであり，これらはタスクについても多くの共通点がある．実際，この方法を利用するメリットおよびデメリットは，B2BUA の場合と同様である．ただし B2BUA は，UA をベースとしているため，UA に対して定義されているオートマトンに基づく必要がある．それに対して，ALG にはその必要がないという相違点がある． ALG を利用することによって，B2BUA よりも柔軟な実装が可能になる．

第 4 章

状態正常化シグナリングに基づく SIP-ALG

4.1 状態正常化シグナリング

SIP でのセッション異常時からの復旧問題を解決するために，本論文では状態正常化シグナリングに基づく SIP-ALG を提案する．ここで状態正常化シグナリングとは，サーバがセッションの異常を検知したときに，適切なシグナリングを行うことによって，セッションを正常に終了させる手続きをいう．これによって，パス上のエンティティに対して，迅速かつ適切な状態遷移を促すことが可能となる．各エンティティは初期状態へ移行し，無効となったセッションのために確保されていたリソースを解放する．

本論文では，セッションを監視し，状態正常化シグナリングを行うサーバとして SIP-ALG を利用する．この理由は， ALG ではシグナリングとメディアの両方を監視することができるためである． ALG を利用することによって異種ネットワーク間接続が可能となるが， ALG はセッションを通して，シグナリングとメディアの両方が必ず経由するポイントともなる．本論文では [26] に基づく SIP-ALG に対して，自律的シグナリング機能，およびメディアモニタリング機能を付加することによって，状態正常化シグナリングを実現することを考える．

4.1.1 自律的シグナリング

自律的シグナリングとは，セッションの状態に応じて，適切なメッセージ (リクエストおよびレスポンス) を生成する機能モジュールである．従来まで SIP-ALG は，受け取ったメッセージを適切に変換し，送り出すだけであった．そのため ALG は，呼の状態を保持し，特定のオートマトンに基づいて動作する — つまりステートフルである — 必要はなく，ステートレスな実装で十分であった．また ALG は外部からの入力を受けてはじめて，その出力としてシグナリングを行うため，自身でメッセージを生成する必要もなかった．ここで， ALG が自律的シグナリングを行えるようにするためには，次の 2 点で拡張が必要となる．

セッションの状態管理

対象となるセッションの状態を把握することができるようにするために，シグナリングを監視し，セッションの継続中はその状態を保持する必要がある．

トランザクションの生成

自身でシグナリングを行うことができるようにするために，メッセージ (リクエストおよびレスポンス) を動的に生成し，トランザクションを適切に処理する必要がある．

4.1.2 メディアモニタリング

メディアモニタリングとは，メディアセッションを監視し異常を検知した場合に，自律的シグナリングモジュールに対してそれを通知する機能モジュールである．これによりセッションを通して，シグナリングだけではなくメディアも監視の対象とすることが可能となる．

4.2 セッションを構成するシグナリングのフェーズ分割

本論文では，INVITE - 100 Trying/180 Ringing - 200 OK - ACK で開始し，BYE - 200 OK で終了する最も一般的なセッションについて，状態正常化シグナリングのコールフローを検討する．セッションを終了し SIP エンティティを初期状態に移行させるための手続きは，その時点でのセッションの状態によって異なる．そこで本論文では，一連のセッションをそれぞれ状態の異なる 5 つのフェーズに分割する．その上で，どのフェーズでシグナリングまたはメディアの異常が発生したかによって，その後に行うべきシグナリングを決定する．図 4.1にセッションの各フェーズを示す．ここで F1，F3，F7，F9，および F11 は，それぞれのフェーズに移行するトリガとなるメッセージである．

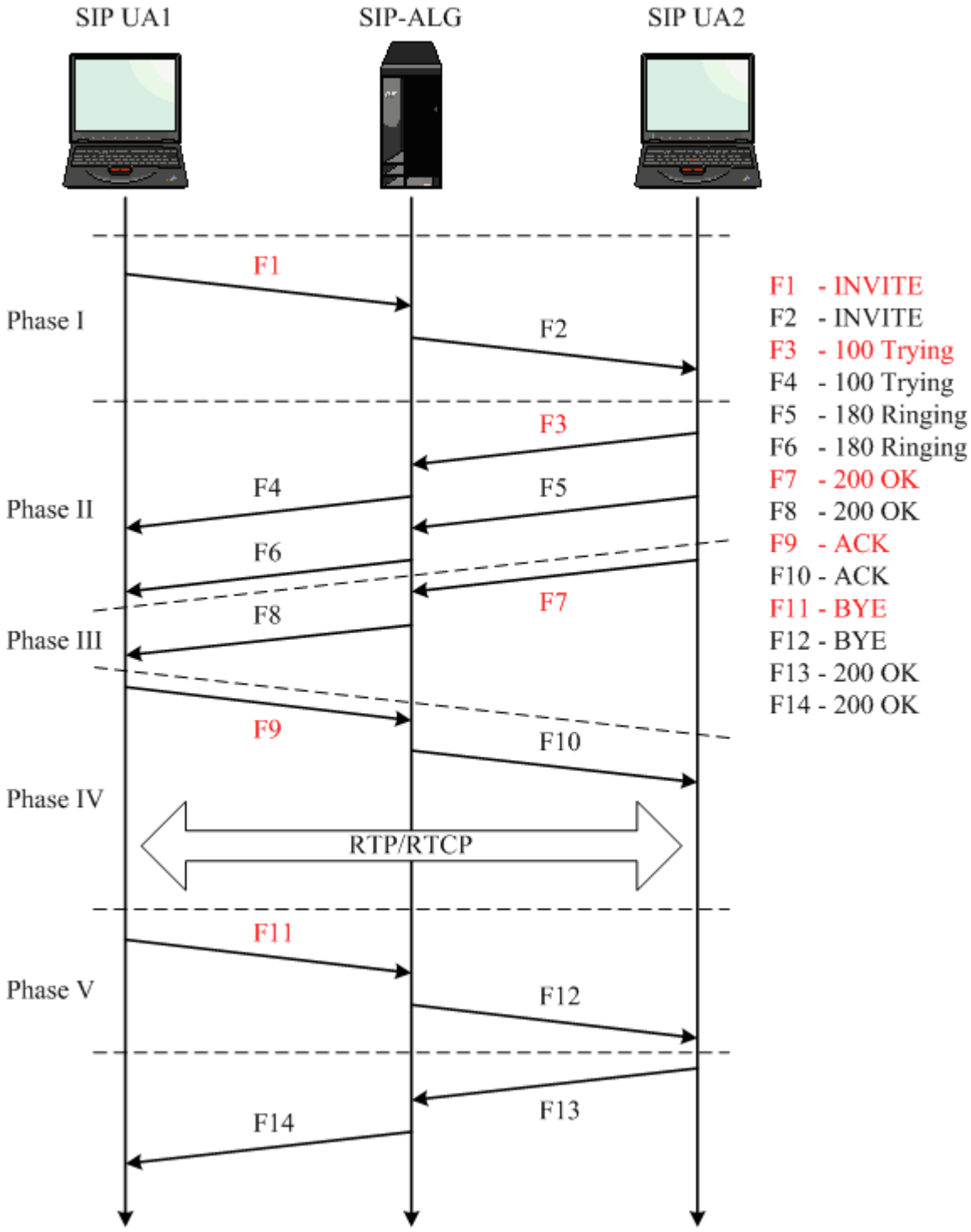

図 4.1: 状態正常化シグナリングのフェーズ

4.2.1 フェーズ I

SIP-ALG は初期状態 (セッションが開始されていない状態) にあるとき，SIP UA1 からの F1 - INVITE を確認することによってフェーズ I に移行する．フェーズ I に移行した後，SIP UA2 から一定時間内にレスポンスがないことを検知すると，セッションに異常が発生したとみなし，状態の正常化シグナリングを開始する．図 4.2にフェーズ I から復旧するコールフローを示す．

ALG は UA1 に対して，まずリクエスト F1 に対する最終エラーレスポンス F3 を返す．続いて UA1 からの F4 によって，ALG と UA1 間のトランザクションが正常形で終了する．

また ALG は UA2 に対しては，シグナリングを行わない．これは ALG と UA2 との間でダイアログが確立されていない[1]ことに基づいている．この状態では CANCEL によるリクエストのキャンセルや，BYE によるダイアログの終了は，RFC3261[1] では認められていない[2]．

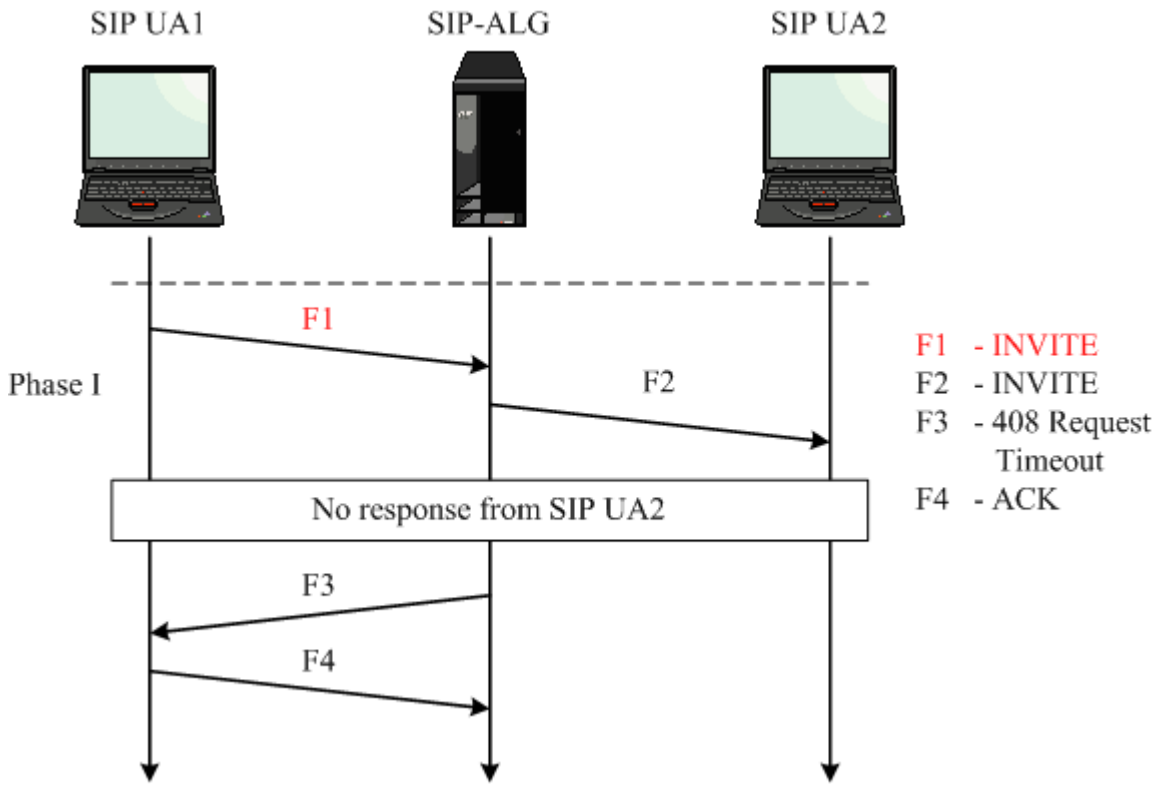

図 4.2: フェーズ I からの復旧コールフロー

[1]early ダイアログを確立するためには，INVITE に対する，100 Trying を除く 1xx レスポンスが必要となる．また confirmed ダイアログを確立するためには，INVITE に対する 2xx レスポンスが必要となる．

[2]CANCEL は early ダイアログ上でのみ，また BYE は confirmed ダイアログ上でのみ認められている．

4.2.2 フェーズ II

SIP-ALG はフェーズ I にあるとき， SIP UA2 からの F3 - 100 Trying，およびその他の暫定レスポンスを確認することによってフェーズ II に移行する．フェーズ II に移行した後， UA2 から一定時間内に最終レスポンスがないことを検知すると，セッションに異常が発生したとみなし，状態の正常化シグナリングを開始する．図 4.3にフェーズ II から復旧するコールフローを示す．

ALG は UA1 に対して，フェーズ I と同様のシグナリングを行う．また ALG は UA2 に対しては， F8 によってリクエスト F1 のキャンセルを試みる．もしも UA2 からのレスポンスがあれば F10 〜 F12 によって， ALG と UA2 間のトランザクションが正常形で終了する．

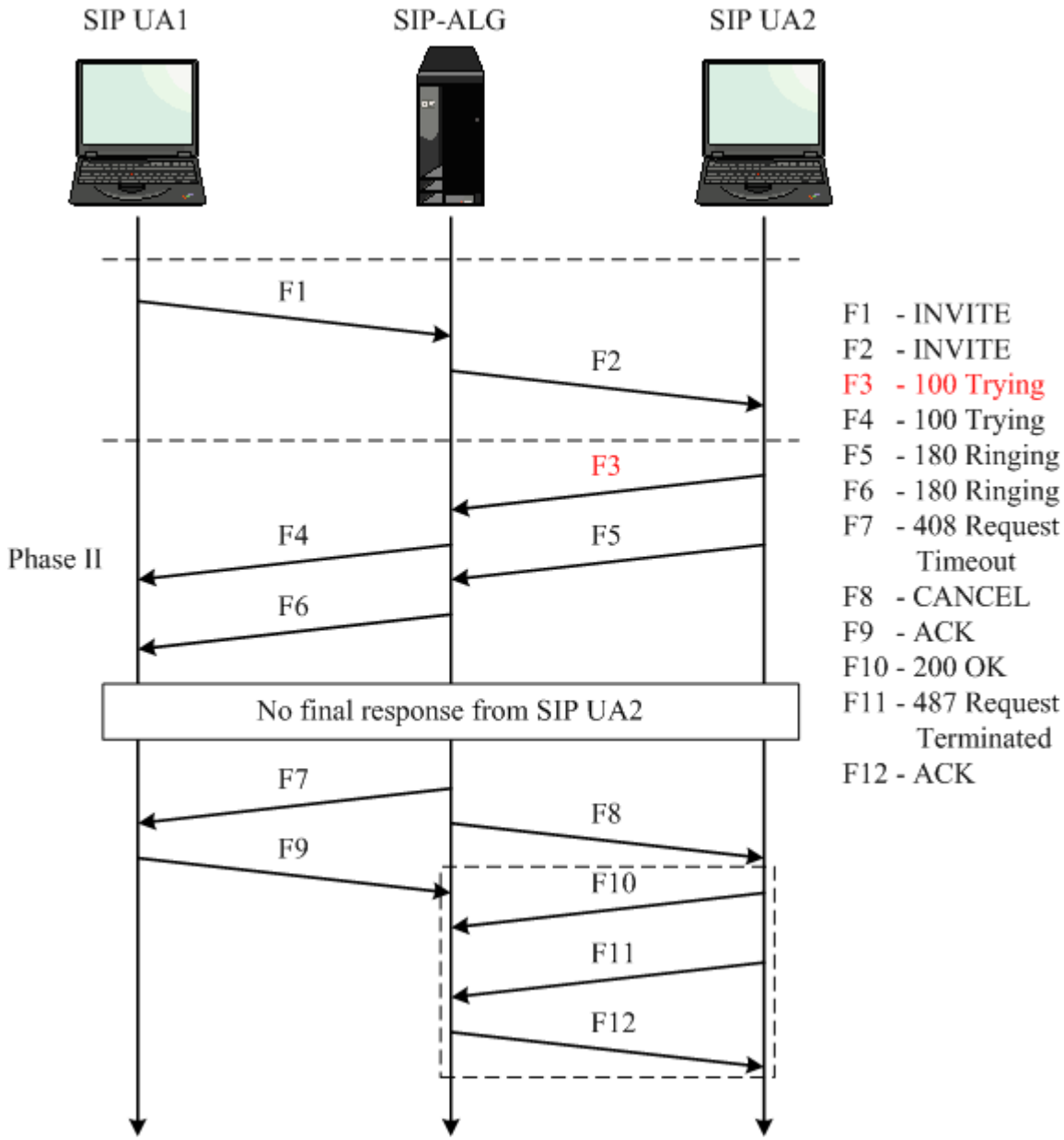

図 4.3: フェーズ II からの復旧コールフロー

4.2.3 フェーズ III

SIP-ALG はフェーズ I またはフェーズ II にあるとき，SIP UA2 からの F7 - 200 OK を確認することによってフェーズ III に移行する．フェーズ III に移行した後，SIP UA1 から一定時間内にトランザクションを終了する ACK リクエストがないことを検知すると，セッションに異常が発生したとみなし，状態の正常化シグナリングを開始する．図 4.4にフェーズ III から復旧するコールフローを示す．

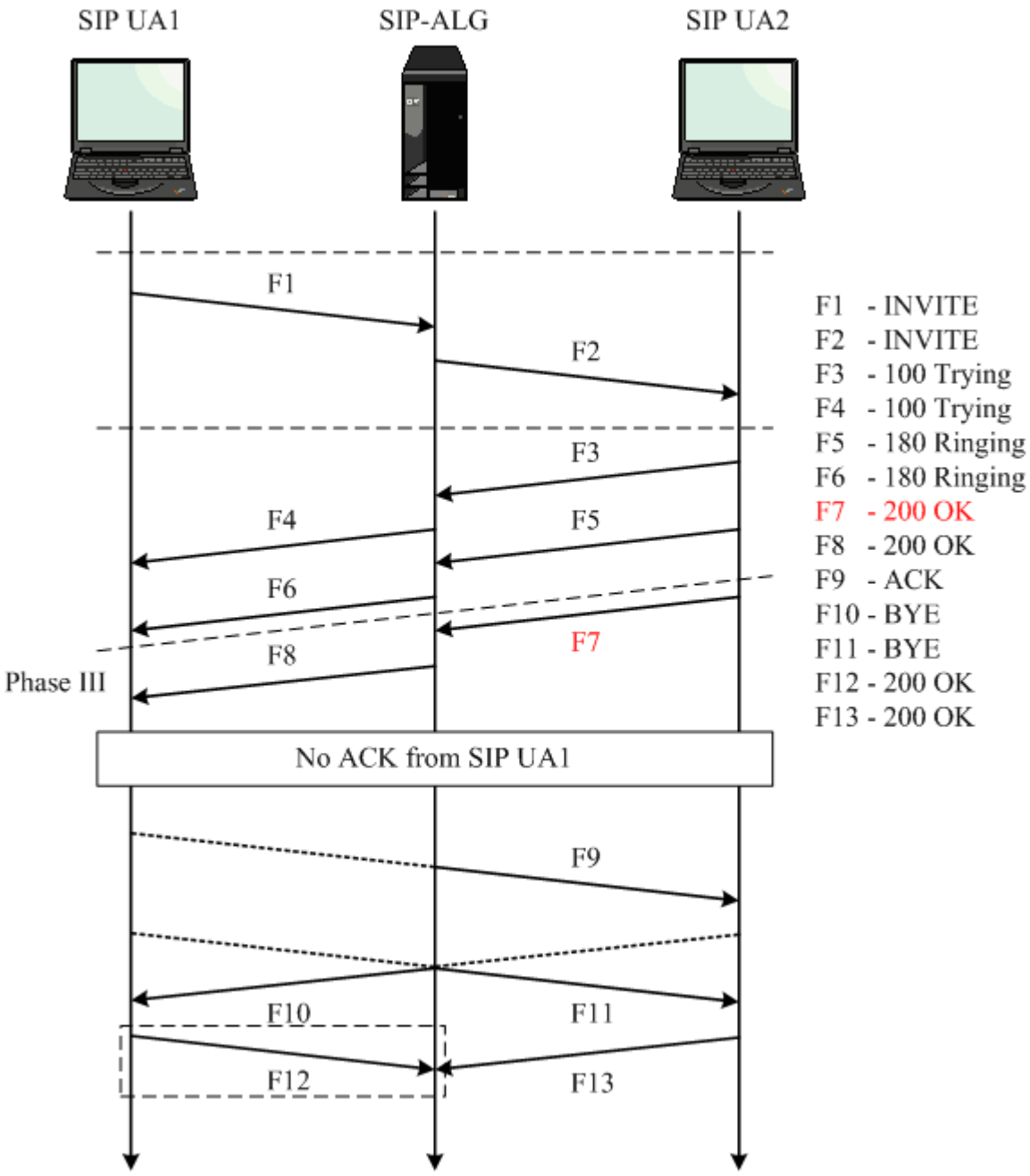

図 4.4: フェーズ III からの復旧コールフロー

まず ALG はリクエスト F1 およびレスポンス F7 によって構築されたトランザクションを終了させる．ALG はあたかも UA1 からのメッセージであるかのようにして F9 を生成し，UA2 に対してトランザクションが正常形で終了したように見せる[3]．

次に ALG は UA1，UA2 に対してそれぞれ F10，F11 によって，確立したセッションを終了させる．ALG はこれらをそれぞれあたかも UA2，UA1 からのメッセージであるかのようにして生成する．ここで UA1 は ACK を送り出す前に BYE を受け取ることになるが，RFC3261[1] では，UAS は confirmed ダイアログ上では BYE を送ることが認められている[4]．

最後に ALG は UA1，UA2 からの最終成功レスポンス F12，F13 を傍受する．これにより，F10，F11 によって開始されたトランザクションがそれぞれ正常形で終了する．ただしこのとき UA1 はシグナリングが行えない状態にあると考えられるため，F12 は実際には行われない可能性が高い．

4.2.4 フェーズ IV

SIP-ALG はフェーズ III にあるとき，SIP UA1 からの F9 - ACK を確認することによってフェーズ IV に移行する．フェーズ IV に移行した後，片方向または両方向からメディアが一定時間ないことを検知すると，セッションに異常が発生したとみなし，状態の正常化シグナリングを開始する．図 4.5にフェーズ IV から復旧するコールフローを示す．

ALG によって行われるシグナリング F11 〜 F14 は，フェーズ III からの復旧での F10 〜 F13 と同様である．ただしこのとき UA1 または UA2 はシグナリングが行えない状態にあると考えられるため，F13 または F14 は実際には行われない可能性が高い．

4.2.5 フェーズ V

SIP-ALG はフェーズ IV にあるとき，SIP UA1 (または SIP UA2) からの F11 - BYE を確認することによってフェーズ V に移行する．フェーズ V に移行した後，UA2 (または UA1) から一定時間内に最終レスポンスがないことを検知すると，セッションに異常が発生したとみなし，状態の正常化シグナリングを開始する．図 4.6にフェーズ V から復旧するコールフローを示す．

ここでは，ALG はリクエスト F11 によって開始されたトランザクションを終了させるだけである．ALG はあたかも UA2 からのメッセージであるかのようにして F13 を生成し，UA1 に対してトランザクションが正常形で終了したように見せる．

[3]ただしここでは confirmed ダイアログが確立されているため，すぐに BYE を生成することもできる．本論文ではまず，UA2 からの，INVITE に対する 200 OK の再送を確実に停止させるため，ACK を生成することとした．

[4]UAS が early ダイアログ上で BYE を送ることは認められていない．

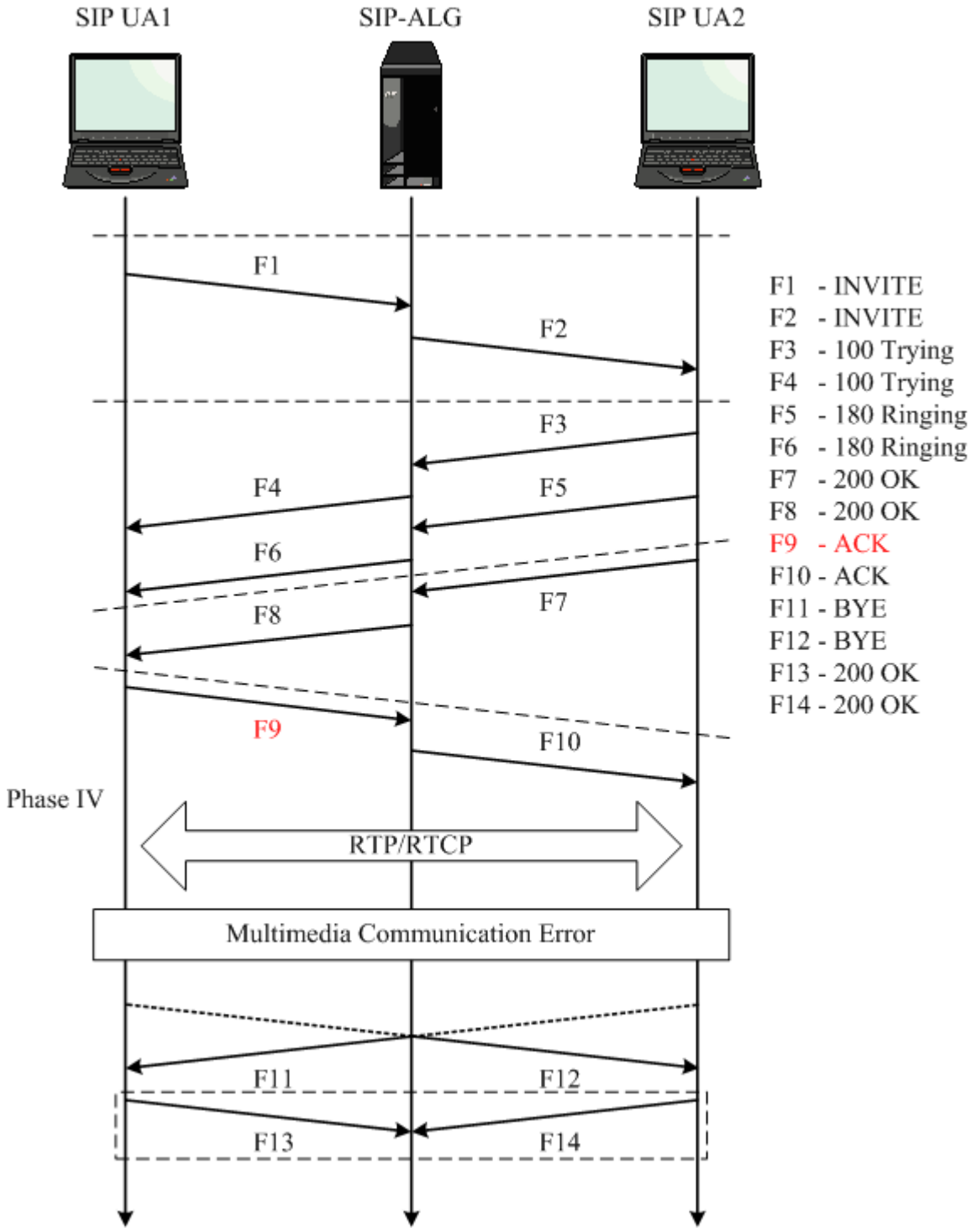

図 4.5: フェーズ IV からの復旧コールフロー

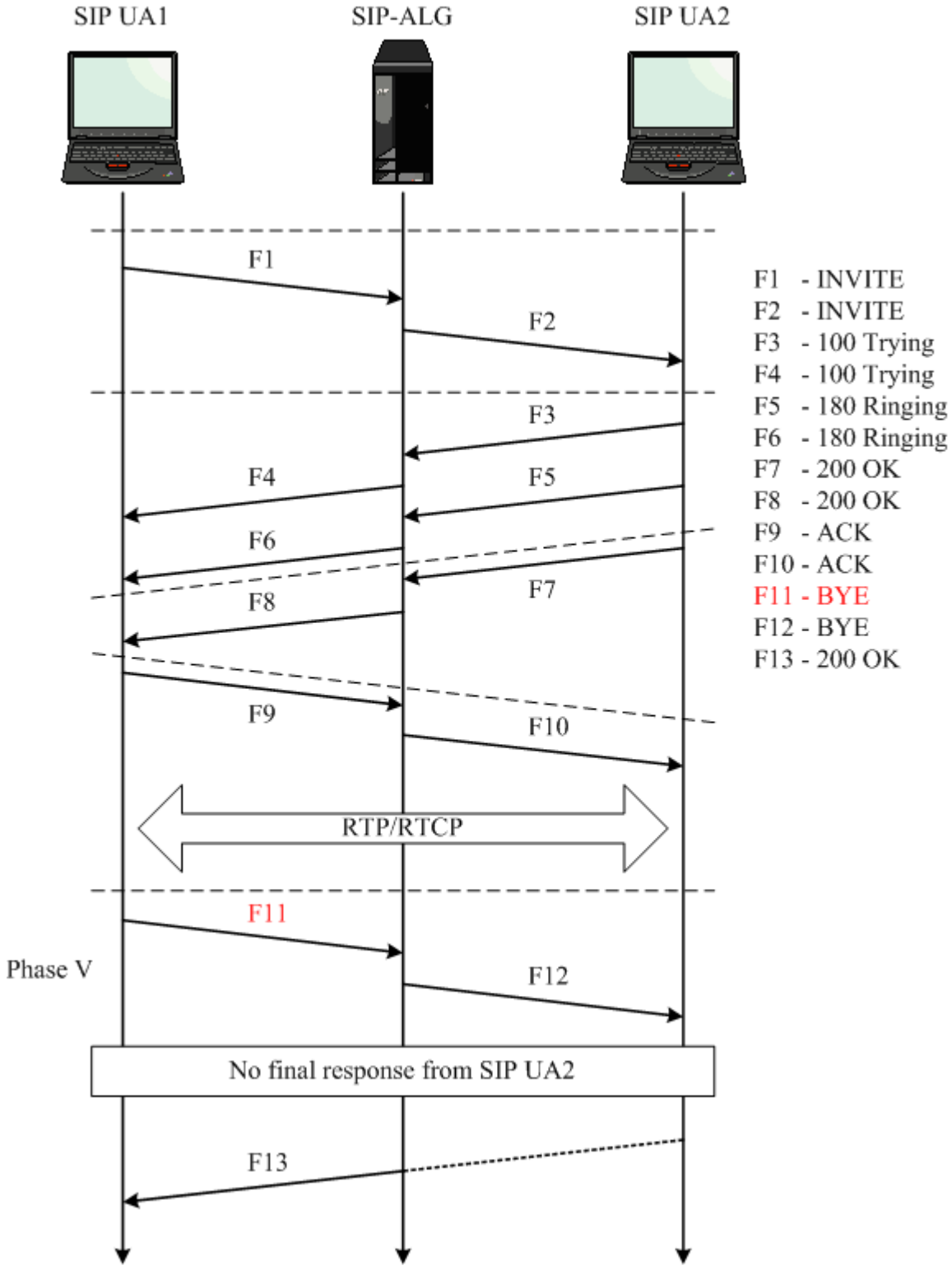

図 4.6: フェーズ V からの復旧コールフロー

第 5 章

検証実験

5.1 実験環境

本論文では提案方式に基づいて状態正常化シグナリングを行う SIP-ALG を実装し，図 5.1のような環境で検証実験を行った． SIP UA にはソフトフォンとして， SJphone[27] を利用した．実験では， UA1 と UA2 との間で ALG を介して SIP による通信を行い，そこでフェーズ I から V までの異常を発生させた．本論文では各フェーズにおいて， ALG によって適切に現在の状態を正常化するシグナリングが行われていることを検証する．

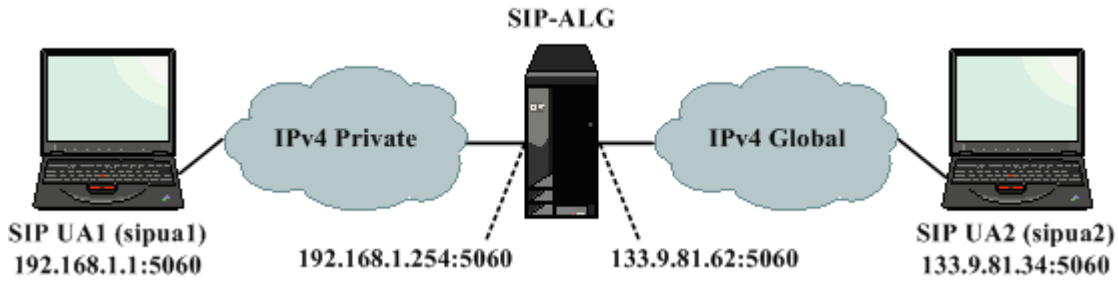

図 5.1: 実験環境

5.2 各フェーズからの復旧

5.2.1 フェーズ I

フェーズ I での異常は，図 4.2において F2 が行われた時点で， UA2 が存在しないか，またはシグナリングが行えない状態になっているときに発生する．本論文では， UA2 上で SIP フォンが稼働していないときに， UA1 からセッション開始リクエストを生成することによって，この状況を発生させる．

このフェーズについての ALG の実装では，UA1 から最後に送られた F1 を転送した後，2 秒以内に UA2 からの何らかのレスポンスがない場合に，状態正常化シグナリングを開始する．図 4.2において，ALG によって状態の正常化が開始された後に行われるシグナリングのメッセージ F3，および F4 を次に示す．

F3 - 408 Request Timeout

```
SIP/2.0 408 Request Timeout
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc82bb0000289100000003
To: <sip:sipua2@133.9.81.34>;tag=20041224235959
From: "sipua1"<sip:sipua1@192.168.1.1>;tag=7016867031613
Call-ID: AAAE2808-16B8-4641-87E0-D374F71FEFF3@192.168.1.1
CSeq: 1 INVITE
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.
```

F4 - ACK

```
ACK sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc82bb0000289100000003
To: <sip:sipua2@133.9.81.34>;tag=20041224235959
From: "sipua1"<sip:sipua1@192.168.1.1>;tag=7016867031613
Call-ID: AAAE2808-16B8-4641-87E0-D374F71FEFF3@192.168.1.1
CSeq: 1 ACK
Max-Forwards: 70
Content-Length: 0
```

フェーズ I からの復旧では，以上のシグナリングによって UA1 および ALG が初期状態に遷移し，セッションが正常に終了することを確認した．

5.2.2 フェーズ II

フェーズ II での異常は，図 4.3において F3 または F5 が行われた後に，UA2 がシグナリングを行えない状態になったか，またはユーザが時間内に応答しなかったときに発生する．前者からの復旧は，後者からの復旧の一部分であるため，ここでは後者の状態を想定する．本論文では，まず UA1 からセッション開始リクエストを生成する．次に UA2 上でユーザが時間内に応答しないことによって，この状況を発生させる．

このフェーズについての ALG の実装では，UA2 から最後に送られた F3 の後，または最後に送られた F5 を転送した後，10 秒以内に UA2 からの最終レスポンスがない場合に，状態正常化

シグナリングを開始する．図 4.3において， ALG によって状態の正常化が開始された後に行われるシグナリングのメッセージ F7 〜 F12 を次に示す．

F7 - 408 Request Timeout

SIP/2.0 408 Request Timeout
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc8353000000cd00000003
To: <sip:sipua2@133.9.81.34>;tag=20041224235959
From: "sipua1"<sip:sipua1@192.168.1.1>;tag=7018392186616
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 INVITE
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.

F8 - CANCEL

CANCEL sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: <sip:sipua2@133.9.81.34>
From: "sipua1"<sip:sipua1@192.168.1.1>;tag=7018392186616
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 CANCEL
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.

F9 - ACK

ACK sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc8353000000cd00000003
To: <sip:sipua2@133.9.81.34>;tag=20041224235959
From: "sipua1"<sip:sipua1@192.168.1.1>;tag=7018392186616
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 ACK
Max-Forwards: 70
Content-Length: 0

F10 - 200 OK

SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: "sipua2"<sip:sipua2@133.9.81.34>;tag=82151456220360
From: <sip:sipua1@192.168.1.1>;tag=7018392186616
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 CANCEL
Server: SJLabs-SJphone/1.30.252
Content-Length: 0

F11 - 487 Request Terminated

SIP/2.0 487 Request terminated
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc8353000000cd00000003
To: "sipua2"<sip:sipua2@133.9.81.34>;tag=82151456220360
From: <sip:sipua1@192.168.1.1>;tag=7018392186616
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0

F12 - ACK

ACK sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: "sipua2"<sip:sipua2@133.9.81.34>;tag=82151456220360
From: <sip:sipua1@192.168.1.1>;tag=7018392186616
Call-ID: FB69984B-F9BC-425C-9BFD-81782B1B4E3E@192.168.1.1
CSeq: 1 ACK
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.

フェーズ II からの復旧では，以上のシグナリングによって各 UA および ALG が初期状態に遷移し，セッションが正常に終了することを確認した.

5.2.3 フェーズ III

フェーズ III での異常は，図 4.4において F8 が行われた時点で，UA1 がシグナリングを行えない状態になっているときに発生する．本論文では，まず UA1 からセッション開始リクエストを生成し，その直後に UA1 上で SIP フォンを強制終了させる．次に UA2 上でユーザが時間内に応答することによって，この状況を発生させる．

このフェーズについての ALG の実装では，UA2 から最後に送られた F7 を転送した後，2 秒以内に UA1 からの ACK がない場合に，状態正常化シグナリングを開始する．図 4.4において，ALG によって状態の正常化が開始された後に行われるシグナリングのメッセージ F9 〜 F11，および F13 を次に示す．

F9 - ACK

```
ACK sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: ”sipua2”<sip:sipua2@133.9.81.34>;tag=82166889019138
From: <sip:sipua1@192.168.1.1>;tag=7019935785394
Call-ID: 3135DCD1-588D-4D5E-8D04-A6A31D7A6436@192.168.1.1
CSeq: 1 ACK
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.
```

F10 - BYE

```
BYE sip:sipua1@192.168.1.1 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.254:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: <sip:sipua1@192.168.1.1>;tag=7019935785394
From: <sip:sipua2@133.9.81.34>;tag=82166889019138
Call-ID: 3135DCD1-588D-4D5E-8D04-A6A31D7A6436@192.168.1.1
CSeq: 2 BYE
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.
```

F11 - BYE

```
BYE sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: <sip:sipua2@133.9.81.34>;tag=82166889019138
From: <sip:sipua1@192.168.1.1>;tag=7019935785394
Call-ID: 3135DCD1-588D-4D5E-8D04-A6A31D7A6436@192.168.1.1
CSeq: 2 BYE
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.
```

F13 - 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: "sipua2"<sip:sipua2@133.9.81.34>;tag=82166889019138
From: <sip:sipua1@192.168.1.1>;tag=7019935785394
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: 3135DCD1-588D-4D5E-8D04-A6A31D7A6436@192.168.1.1
CSeq: 2 BYE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0
```

フェーズ III からの復旧では，以上のシグナリングによって UA2 および ALG が初期状態に遷移し，セッションが正常に終了することを確認した．

5.2.4 フェーズ IV

フェーズ IV での異常は，図 4.5において F10 が行われた後に，セッション終了リクエストを伴わずに UA1 または UA2 からのメディアが一定時間以上なくなったときに発生する．本論文では，まず UA1 と UA2 との間でメディア通信を確立する．次に UA2 上で SIP フォンを強制終了させ，セッション終了リクエストを伴わずにメディア通信を終了することによって，この状況を発生させる．

このフェーズについての ALG の実装では，UA 間でメディア通信が確立した後，セッション終了リクエストを伴わず，UA1 または UA2 からのメディアが 1 秒以上なかった場合に，状態正常化シグナリングを開始する．図 4.5において，ALG によって状態の正常化が開始された後に行われるシグナリングのメッセージ F11 〜 F13 を次に示す．

F11 - BYE

BYE sip:sipua1@192.168.1.1 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.254:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: <sip:sipua1@192.168.1.1>;tag=70223306225261
From: <sip:sipua2@133.9.81.34>;tag=8219083756238
Call-ID: F25F344E-09C2-4C8F-B6B1-99BDC283E96E@192.168.1.1
CSeq: 2 BYE
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.

F12 - BYE

BYE sip:sipua2@133.9.81.34 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: <sip:sipua2@133.9.81.34>;tag=8219083756238
From: <sip:sipua1@192.168.1.1>;tag=70223306225261
Call-ID: F25F344E-09C2-4C8F-B6B1-99BDC283E96E@192.168.1.1
CSeq: 2 BYE
Max-Forwards: 70
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.

F13 - 200 OK

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.254:5060;branch=z9hG4bKd9da2f22eaecb3a8cd9192ac5d0cace
To: ”sipua1”<sip:sipua1@192.168.1.1>;tag=70223306225261
From: <sip:sipua2@133.9.81.34>;tag=8219083756238
Contact: <sip:sipua1@192.168.1.1:5060>
Call-ID: F25F344E-09C2-4C8F-B6B1-99BDC283E96E@192.168.1.1
CSeq: 2 BYE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0

フェーズ IV からの復旧では，以上のシグナリングによって UA1 および ALG が初期状態に遷移し，セッションが正常に終了することを確認した.

5.2.5 フェーズ V

フェーズ V での異常は，図 4.6において F12 が行われた時点で， UA2 がシグナリングを行えない状態になっているときに発生する．本論文では，まず UA1 と UA2 との間でメディア通信を確立する．次にメディアモニタリングの機能を無効とした状態で， UA2 上で SIP フォンを強制終了させる．最後に UA1 からセッション終了リクエストを生成することによって，この状況を発生させる．

このフェーズについての ALG の実装では, UA1 から最後に送られた F11 を転送した後, 2 秒以内に UA2 からの最終レスポンスがない場合に，状態正常化シグナリングを開始する．図 4.6において， ALG によって状態の正常化が開始された後に行われるシグナリングのメッセージ F13 を次に示す．

F13 - 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.1;branch=z9hG4bK850951210131c9b141cc8a8d00007cd200000025
To: <sip:sipua2@133.9.81.34>;tag=82335084315787
From: <sip:sipua1@192.168.1.1>;tag=70367554627690
Call-ID: 47141749-9815-483D-8A45-EF78A7FF4188@192.168.1.1
CSeq: 2 BYE
Server: SIP-ALG with AS eXtension (i386/linux)
Content-Length: 0
Warning: Now, AS is activated because of signaling error.
```

フェーズ V からの復旧では，以上のシグナリングによって UA1 および ALG が初期状態に遷移し，セッションが正常に終了することを確認した．

5.3 検証結果についての評価と考察

実験により，すべてのフェーズにおいて， ALG による状態正常化シグナリングが，パス上の UA に対して作用することを確認した．次に，各フェーズにおける状態正常化シグナリングの有効性について評価し，その上で最も効率的な状態正常化の手法について考察する．検討のために， RFC3261[1] で UA に対して定義されているオートマトンを利用する[1]．

[1]オートマトンの表記について: 枠内は状態を表し，矢印は状態の遷移を表す．各矢印には状態が遷移するためのイベントが関連付けられており，'#' 以下はそのとき取られるアクションを表す．

5.3.1 フェーズ I

フェーズ I では，INVITE を生成した UA (UAC) に対して，状態正常化シグナリングが作用する．ここでは INVITE トランザクションにおける UAC のオートマトン (図 5.2) を利用して，UAC に対する状態正常化の効果を検討する．本論文で提案した，フェーズ I における状態正常化シグナリングでは，UAC に対して，Calling ステートから Completed ステートへの遷移を促すことが可能になる．フェーズ I でセッションに異常が発生する場合，図 5.2において，UAC は Calling ステートにある．ここで UAC を迅速に Terminated ステートに遷移させる必要がある．

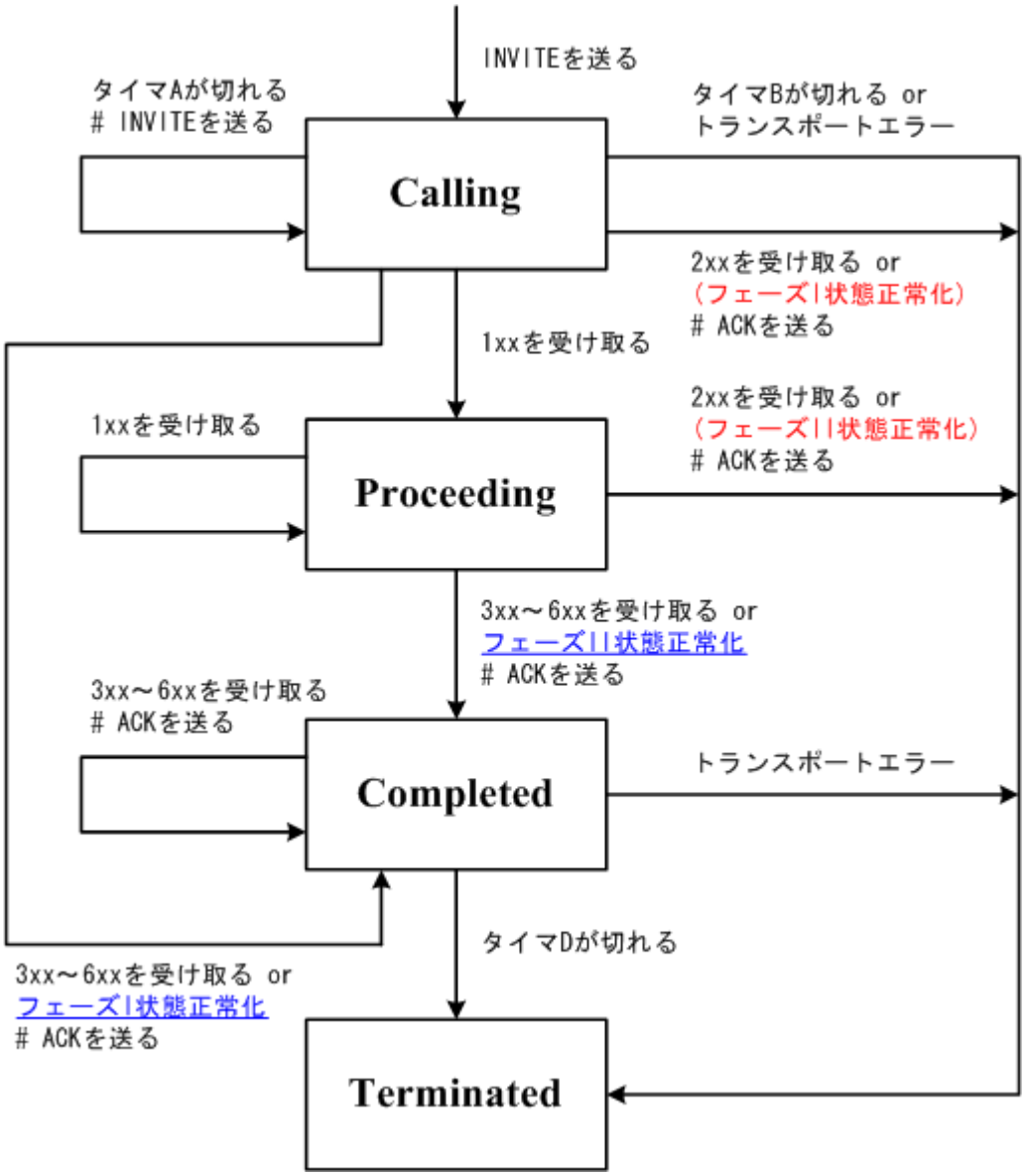

図 5.2: INVITE トランザクションにおける UAC オートマトン

フェーズ I における状態正常化シグナリングを行わないとき，UAC は Timer B [2] (以下 T(B)) が切れることによって，Calling ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T(B) sec である．

それに対して，フェーズ I における状態正常化シグナリングを行うとき，UAC はまず ALG によってフェーズ I の状態正常化が開始されるまで待機する必要がある．この時間，すなわち ALG のもつフェーズ I 状態正常化タイマのタイムアウトを T'1 sec とする．状態正常化シグナリングによって，UAC は Calling ステートから Completed ステートに移行する．これに続いて，UAC は Timer D [3] (以下 T(D)) が切れることによって，Completed ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T'1 + T(D) sec である．

以上のことから，状態正常化シグナリングを行う場合，$T'1+T(D) < T(B)$ となるように T'1 を決定することで，UAC の迅速な状態遷移が可能となる．しかしここで SIP のトランスポートプロトコルとして UDP を利用する場合，必ず $T(D) > T(B)$ が成立する．そのためどのように T'1 を決定しても，状態正常化シグナリングによって，デフォルトのタイムアウトよりも復旧が早くなることはない．したがって，フェーズ I においては状態正常化シグナリングを行わず，UA においてデフォルトのタイムアウトを発生させるほうが効率的である[4]．

フェーズ I からの復旧 – 別手法の検討

フェーズ I からの復旧では，図 4.2において提案したシグナリングとはまた別の方法で，状態正常化を実現することができる．ただし，別手法に基づくフェーズ I 状態正常化シグナリングでは，UA に対して無駄なオーバヘッドが伴う．そのため，本論文では別手法については扱わないこととする．別手法についての詳細な検討結果については，付録の表 A.1に示す．

5.3.2 フェーズ II

フェーズ II では，INVITE を生成し，それに対する 1xx を受け取った UA (UAC) に対して，状態正常化シグナリングが作用する．またユーザが時間内に応答しなかった場合には，INVITE に対する 1xx を生成した UA (UAS) に対しても，状態正常化シグナリングが作用する．

[2]INVITE トランザクションのタイムアウトタイマ

[3]応答の再送のための待ち時間

[4]検証実験に利用した UA である SJphone[27] は，INVITE クライアントトランザクションがタイムアウトした後，新たに BYE を生成することによって，独自に状態正常化を図っていた．このとき復旧に要する時間は T(B) + T(F) (T(F) については後述) である．そのため $T'1 + T(D) < T(B) + T(F)$ となるように T'1 を決定することで，UAC の迅速な状態遷移が可能となり，本論文で提案した状態正常化シグナリングが有効であった．しかし UA に対するこのような実装は RFC3261[1] に基づいたものではなく，一般的なものではないため，本論文では扱わないこととした．BYE の生成は confirmed ダイアログ上でのみ認められており，フェーズ I のようにダイアログが確立していない状態では認められていない．

UAC に対する効果

ここではフェーズ I からの復旧の場合と同様に，INVITE トランザクションにおける UAC のオートマトン (図 5.2) を利用して，UAC に対する状態正常化の効果を検討する．本論文で提案した，フェーズ II における状態正常化シグナリングでは，UAC に対して，Proceeding ステートから Completed ステートへの遷移を促すことが可能になる．フェーズ II でセッションに異常が発生する場合，図 5.2において，UAC は Proceeding ステートにある．ここで UAC を迅速に Terminated ステートに遷移させる必要がある．

フェーズ II における状態正常化シグナリングを行わないとき，UAC は Proceeding ステートからどのステートにも移行することができない．これは Proceeding ステートにはタイマが全く設けられていないためである．したがって，このとき復旧することは不可能である．

それに対して，フェーズ II における状態正常化シグナリングを行うとき，UAC はまず ALG によってフェーズ II の状態正常化が開始されるまで待機する必要がある．この時間，すなわち ALG のもつフェーズ II 状態正常化タイマのタイムアウトを T'2 sec とする．状態正常化シグナリングによって，UAC は Proceeding ステートから Completed ステートに移行する．これに続いて，UAC は T(D) が切れることによって，Completed ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T'2 + T(D) sec である．

以上のことから，従来までのオートマトンでは，フェーズ II においてセッションに異常が発生した場合，UAC は Proceeding ステートから遷移できないため，復旧が不可能であった．しかし，本論文で提案した状態正常化シグナリングによって，UAC は Proceeding ステートからの状態遷移が，T'2 + T(D) sec の時間で必ず可能となる．したがって，フェーズ II においては UAC に対する状態正常化シグナリングが有効である．

UAS に対する効果

ここでは INVITE トランザクションにおける UAS のオートマトン (図 5.3) を利用して，UAS に対する状態正常化の効果を検討する．本論文で提案した，フェーズ II における状態正常化シグナリングでは，UAS に対して，Proceeding ステートから Completed ステートへの遷移を促すことが可能になる．フェーズ II でユーザが時間内に応答しなかった場合，図 5.3において，UAS は Proceeding ステートにある．ここで UAS を迅速に Terminated ステートに遷移させる必要がある．ユーザが応答しないことはセッションの異常ではないが，リソースの浪費となる可能性がある．公衆電話の場合，一定回数の呼出音が鳴ることで，留守番電話や FAX への自動切り替えが行われることを考慮すれば，SIP でもこのような措置は妥当であると考えられる．

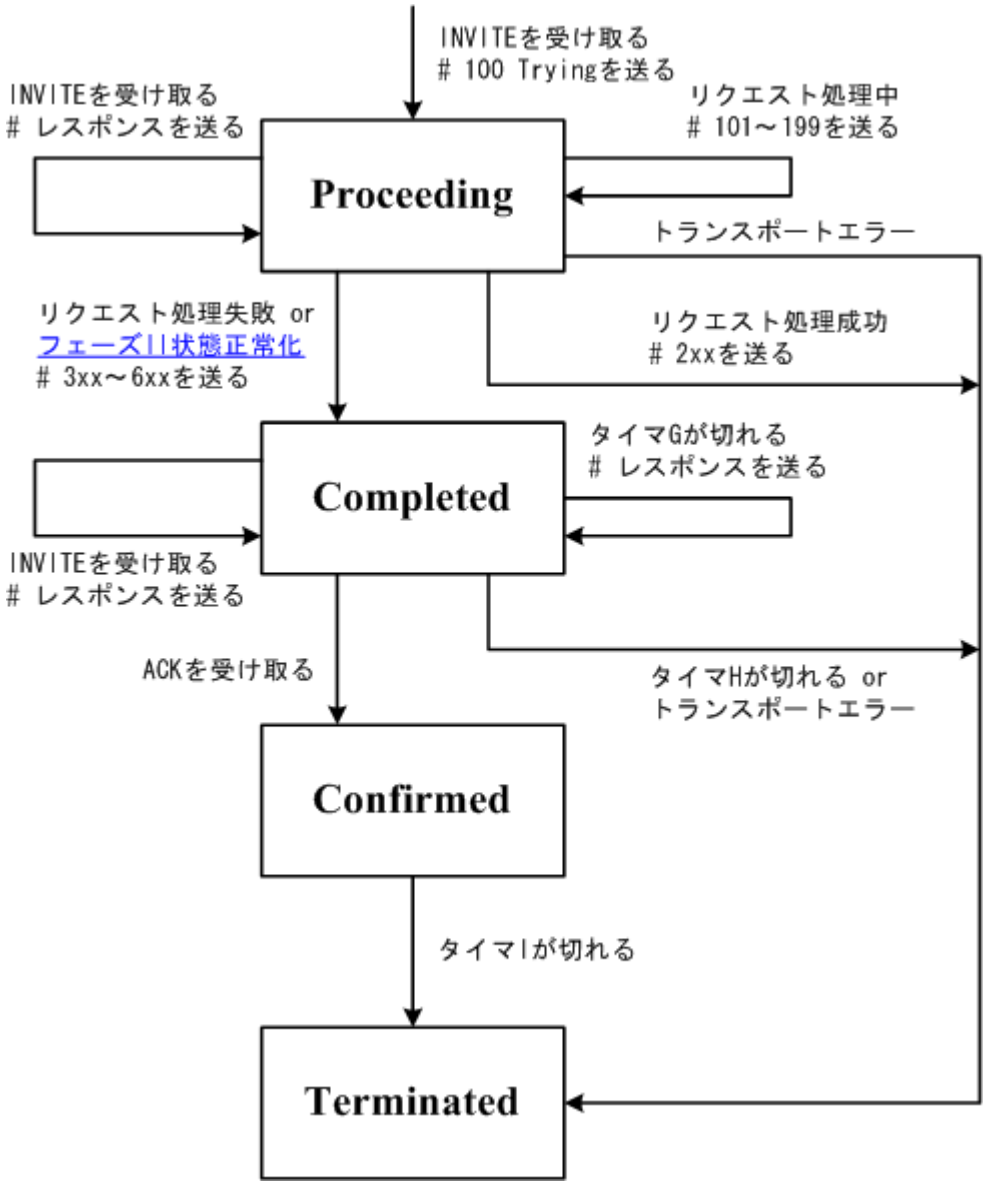

図 5.3: INVITE トランザクションにおける UAS オートマトン

フェーズ II における状態正常化シグナリングが行われないとき， UAS は Proceeding ステートからどのステートにも移行することができない．これは Proceeding ステートにはタイマが全く設けられていないためである．したがって，このとき UAC からセッションの取消を行わない限り， UAS では時間の制限なくユーザの呼出を継続する．

それに対して，フェーズ II における状態正常化シグナリングを行うとき， UAS はまず ALG によってフェーズ II の状態正常化が開始されるまで， T’2 sec 待機する必要がある．状態正常化シグナリングによって，はじめに UAS は Proceeding ステートから Completed ステートに移行する．これに続いて， ALG から ACK を受け取ることによって， UAS は Completed ステート

から Confirmed ステートに移行する．最後に，UAS は Timer I [5] (以下 T(I)) が切れることによって，Confirmed ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T'2 + T(I) sec である．

以上のことから，従来までのオートマトンでは，フェーズ II においてユーザが応答しなかった場合，UAS は Proceeding ステートから遷移できないため，時間の制限なくユーザの呼出が継続していた．しかし，本論文で提案した状態正常化シグナリングによって，UAS は Proceedng ステートからの状態遷移が，T'2 + T(I) sec の時間で必ず可能となる．したがって，フェーズ II においては UAS に対する状態正常化シグナリングが有効である．

ただしこの場合，どのような呼であっても，リソースの浪費を抑制するために，呼出のための最長時間が T'2 sec として決定される．しかし各 UA (UAC と UAS) では，状態正常化によって初期状態に移行した後であれば，再発信することも，またしないこともできる．そのためユーザに対して選択肢を与えることができ，利便性にも影響はないと考えられる．

フェーズ II からの復旧 – 別手法の検討

フェーズ II からの復旧では，図 4.3において提案したシグナリングとはまた別の方法で，状態正常化を実現することができる．ただし，別手法に基づくフェーズ II 状態正常化シグナリングでは，UA に対して無駄なオーバヘッドが伴う．そのため，本論文では別手法については扱わないこととする．別手法についての詳細な検討結果については，付録の表 A.2に示す．

5.3.3 フェーズ III

フェーズ III では，INVITE に対する 2xx を生成した UA (UAS) に対して，状態正常化シグナリングが作用する．ただしここで INVITE トランザクションにおける UAS オートマトン (図 5.3) において，UAS は既に Terminated ステートにある．

しかしメディア通信は確立しているため, 開始したセッションを終了するために, 新しく BYE トランザクションが生成される．これによって，INVITE トランザクションにおける UAS は，新たに BYE トランザクションにおける UAS となる．そのため，ここでは非 INVITE トランザクションにおける UAS のオートマトン (図 5.4) を利用して，UAS に対する状態正常化の効果を検討する．本論文で提案した，フェーズ III における状態正常化シグナリングでは，UAS に対して，初期状態 (BYE トランザクションが開始されていない状態) から Trying ステートへの遷移を促すことが可能になる．フェーズ III でセッションに異常が発生する場合，図 5.4において，UAS は初期状態にある．ここで UAS を迅速に Terminated ステートに遷移させる必要がある．

[5] ACK 再送のための待ち時間

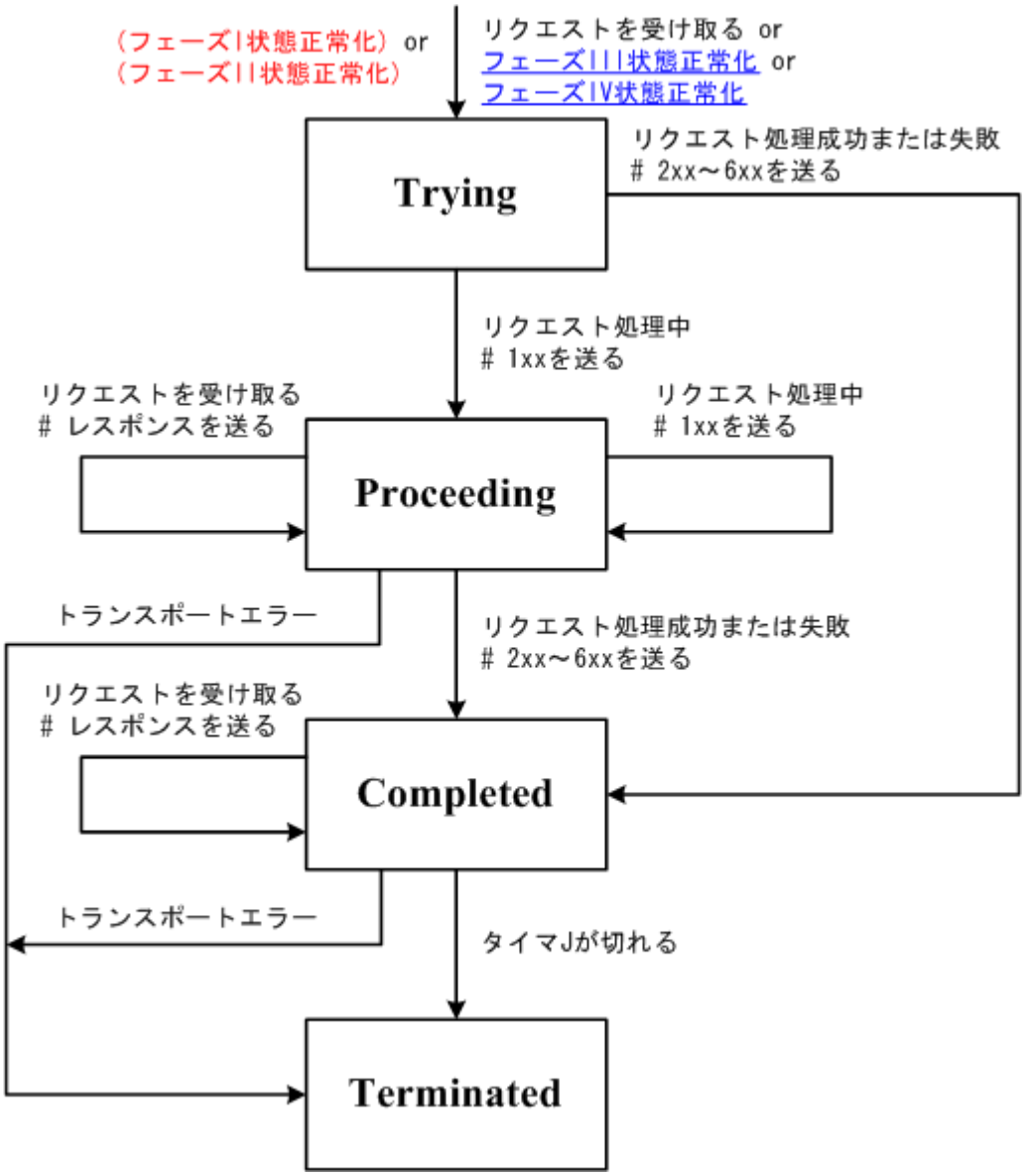

図 5.4: 非 INVITE トランザクションにおける UAS オートマトン

フェーズ III における状態正常化シグナリングを行わないとき， UAS は， INVITE トランザクションを終了し，メディア通信を開始する．しかしこのとき UA は特定のオートマトンに基づいて動作することはない．そのため UAS はセッションの異常を検知することができず，メディア通信を終了することができない．したがって，このとき復旧することは不可能である．ここで，RFC3261[1] では，「UAS が 2xx を生成したのにいつまでも ACK を受け取らない場合， UAS はダイアログを終了するために BYE を生成するべきである」とされている．しかし実装は必須ではないため，すべての UA に対して期待できる機能ではない．

それに対して，フェーズ III における状態正常化シグナリングを行うとき，UAS はまず ALG によってフェーズ III の状態正常化が開始されるまで待機する必要がある．この時間，すなわち

ALG のもつフェーズ III 状態正常化タイマのタイムアウトを T'3 sec とする．状態正常化シグナリングによって， UAS は ALG からの BYE を受け取る．それを契機として，はじめに UAS は Trying ステートに入る．これに続いて， BYE に対する 2xx を生成することによって， UAS は Trying ステートから Completed ステートに移行する．最後に，UAS は Timer J [6] (以下 T(J)) が切れることによって，Completed ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T'3 + T(J) sec である．

以上のことから，従来までのオートマトンでは，フェーズ III においてセッションに異常が発生した場合， UAS はメディア通信を終了できない．しかし，本論文で提案した状態正常化シグナリングによって， UAS はこの状態からの復旧が， T'3 + T(J) sec の時間で可能となる．したがって，フェーズ III においては UAS に対する状態正常化シグナリングが有効である．また UAS が独自に BYE を生成する場合，復旧に要する時間は T(B) + T(J) sec である．そのため $T'3+T(J) < T(B)+T(J)$，つまり $T'3 < T(B)$ となるように T'3 を決定することで，この場合でも UAS の迅速な状態遷移が可能となる．

5.3.4 フェーズ IV

フェーズ IV では，異常が発生したセッションで，メディア通信を継続している UA に対して，状態正常化シグナリングが作用する．一般に UA は，通信相手である UA からのメディアをモニタリングすることはない．

フェーズ IV における状態正常化シグナリングを行わないとき，メディア通信を継続している UA は，通信相手である UA からのメディアがなくなっても，セッションの異常とみなすことはなく，メディア通信を終了することができない．したがって，このとき復旧することは不可能である．フェーズ III との決定的な相違点として， INVITE トランザクションにおける UAS が，独自に BYE を生成することはないということがある．

それに対して，フェーズ IV における状態正常化シグナリングを行うとき，メディア通信を継続している UA は，まず ALG によってフェーズ IV の状態正常化が開始されるまで待機する必要がある．この時間，すなわち ALG のもつフェーズ IV 状態正常化タイマのタイムアウトを T'4 sec とする．この後行われる状態正常化シグナリングの流れと，その効果は，フェーズ III からの復旧の場合と同様である．したがって，このとき復旧に要する時間は T'4 + T(J) sec である．

以上のことから，従来までのメディア通信では，フェーズ IV においてセッションに異常が発生した場合， UA は復旧が不可能であった．しかし，本論文で提案した状態正常化シグナリングによって， UA はこの状態からの復旧が， T'4 + T(J) sec の時間で可能となる．したがって，フェーズ IV においては UA に対する状態正常化シグナリングが有効である．

[6]非 INVITE リクエストの再送のための待ち時間

5.3.5 フェーズ V

フェーズ V では, BYE を生成した UA (UAC) に対して, 状態正常化シグナリングが作用する. ここでは非 INVITE トランザクションにおける UAC のオートマトン (図 5.5) を利用して, UAC に対する状態正常化の効果を検討する. 本論文で提案した, フェーズ V における状態正常化シグナリングでは, UAC に対して, Trying ステートから Completed ステート, または Proceeding ステートから Completed ステートへの遷移を促すことが可能になる. フェーズ V でセッションに異常が発生する場合, 図 5.5において, UAC は Trying ステートにある. ここで UAC を迅速に Terminated ステートに遷移させる必要がある.

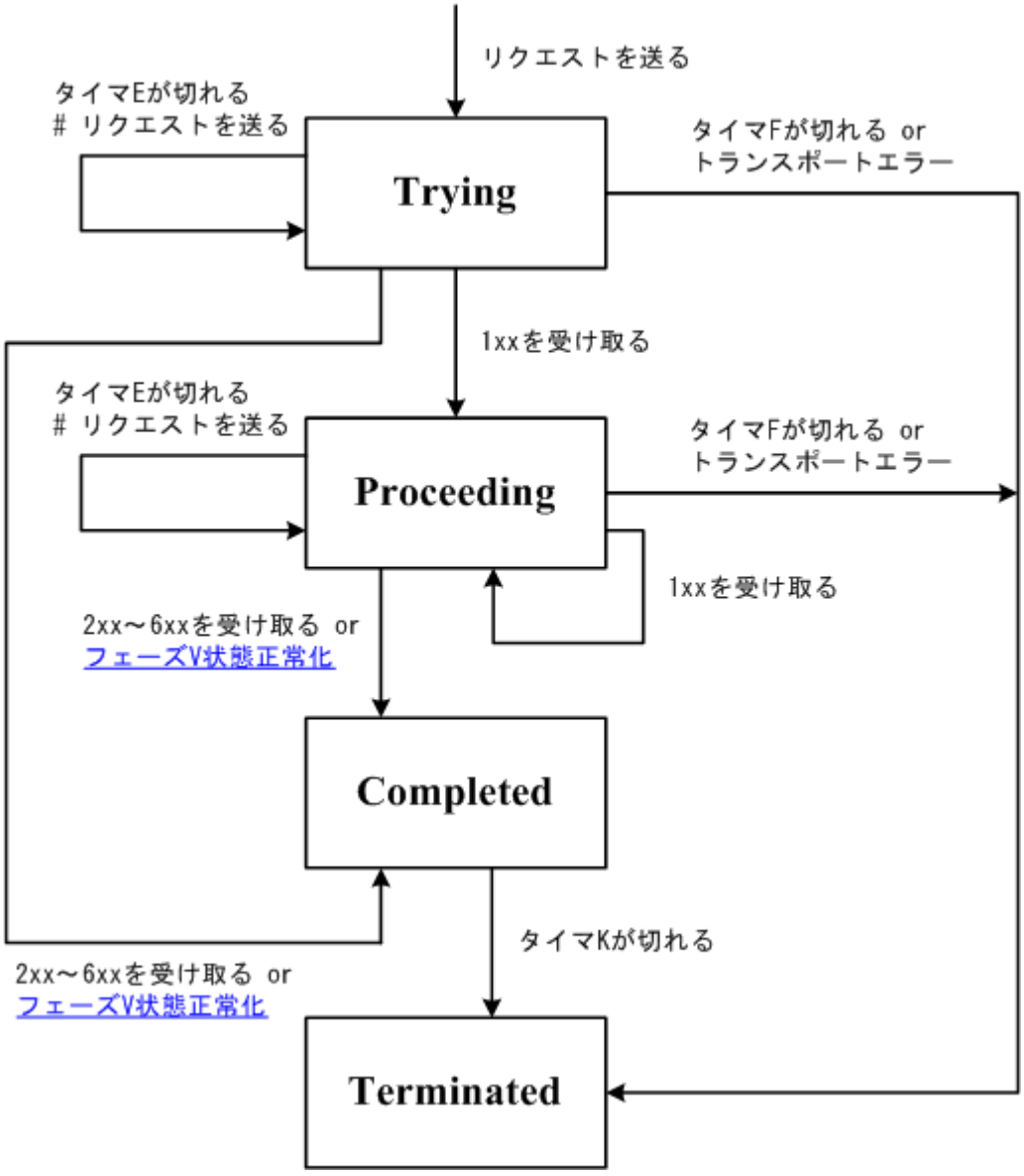

図 5.5: 非 INVITE トランザクションにおける UAC オートマトン

フェーズ V における状態正常化シグナリングを行わないとき，UAC は Timer F [7] (以下 T(F)) が切れることによって, Trying ステートから Terminated ステートに移行する．したがって, このとき復旧に要する時間は T(F) sec である．

それに対して, フェーズ V における状態正常化シグナリングを行うとき, UAC はまず ALG によってフェーズ V の状態正常化が開始されるまで待機する必要がある．この時間, すなわち ALG のもつフェーズ V 状態正常化タイマのタイムアウトを T'5 sec とする．状態正常化シグナリングによって， UAC は Trying ステートから Completed ステートに移行する．これに続いて, UAC は Timer K [8] (以下 T(K)) が切れることによって， Completed ステートから Terminated ステートに移行する．したがって，このとき復旧に要する時間は T'5 + T(K) sec である．

以上のことから，状態正常化シグナリングを行う場合， $T'5+T(K) < T(F)$ となるように T'5 を決定することで， UAC の迅速な状態遷移が可能となる．したがって，フェーズ V においては UAC に対する状態正常化シグナリングが有効である．

[7]非 INVITE トランザクションのタイムアウトタイマ
[8]応答の再送のための待ち時間

5.3.6 まとめ

各フェーズにおける状態正常化シグナリングの有効性についての検討の結果を，表 5.1に示す．検討によって，フェーズ I を除くすべてのフェーズにおいて，本論文で提案した状態正常化シグナリングが，従来までの UA での個別タイムアウトよりも，有効な復旧手法であることを明らかにした．このことから，実際への応用を考えた場合，フェーズ I では従来通り UA での個別タイムアウトを発生させ，フェーズ II 以降でのみ状態正常化シグナリングを行うことによって，セッション異常時からの最適な復旧を実現することができる[9]．

表 5.1: 各フェーズにおける状態正常化シグナリングの有効性

フェーズ	復旧の対象	RFC3261[1] における従来手法 — UA での個別タイムアウト		本論文における提案手法 — 状態正常化シグナリング	
		復旧の可否	所要時間 (sec)	復旧の可否	所要時間 (sec)
I	INVITE-UAC	可能 ○	$T(B)$ ○	可能 ○	$T'1 + T(D)$ △ $T'1+T(D) > T(B)$
II	INVITE-UAC	不可能 × UA のオートマトンにタイマが定義されていない	∞ ×	可能 ○	$T'2 + T(D)$ ○
	INVITE-UAS	可能 △ ユーザが着信に応答する場合に限り可能	不定 △ ユーザが着信に応答するまでの時間	可能 ○	$T'2 + T(I)$ △ 一定時間内にユーザからの応答がなければ異常とみなす
III	INVITE-UAS	可能 △ UAS が独自に BYE を生成する場合に限り可能	∞ または $T(B) + T(J)$ △	可能 ○	$T'3 + T(J)$ ○
IV	UA	不可能 × 一般に，UA に対してメディアモニタリングは実装されない	∞ ×	可能 ○	$T'4 + T(J)$ ○
V	BYE-UAC	可能 ○	$T(F)$ △ $T(F) \geq T'5+T(K)$	可能 ○	$T'5 + T(K)$ ○

[9]ただしフェーズ I においても，UA に対してセッションの異常を通知するためには，復旧に要する時間に関わらず，状態正常化シグナリングを行うべきであると考えられる．

第 6 章

結論

6.1 むすび

本論文では，SIP でのセッション異常時からの復旧問題を解決するために，状態正常化シグナリングに基づく SIP-ALG を提案した．提案方式に基づく SIP-ALG は，セッションの異常状態から速やかにシグナリングの正常形を構築し，パス上のエンティティに対して適切な状態遷移を促す．これにより，無効となったリソースを迅速に解放させることができる．

本論文では，まず SIP における典型的なセッションを，それぞれ状態の異なる 5 つのフェーズ I ～ V に分割し，復旧のシグナリングフローを定義した．次に提案方式に基づいて SIP-ALG の実装を行い，実験によってその動作を検証した．最後に，検証結果についての評価と考察を通して，フェーズ I を除くすべてのフェーズにおいて，状態正常化シグナリングが有効であることを示した．従来手法 (UA での個別タイムアウト) と提案手法 (状態正常化シグナリング) を組み合わせることによって，セッション異常時からの最適な復旧を実現することができる．

本論文で提案した状態正常化シグナリングの機能は，SIP-ALG にしか実装できないものではない．モジュールとして，既存の SIP サーバにも組み込むことができるものである．特に実際のサービスにおいては，非常に多くのセッションの状態を管理する必要があるステートフルサーバ (プロキシ) が存在する．そのようなサーバに対して，状態正常化シグナリングの機能を組み込むことによって，セッションに異常が発生しても，無効なリソースの発生を最小限に抑えることができるようになる．

6.2 今後の課題

本論文における今後の課題として，次のような点が挙げられる．これらは，SIP における状態正常化シグナリングを実用化するために克服すべき課題でもある．

汎用性のある状態正常化モデルの定義

本論文では，INVITE によって開始され，BYE によって終了する典型的なセッションについて，状態正常化シグナリングのコールフローを提案した．しかし実際にセッションに異常が発生するのは，これらのトランザクションに限定されない．そのため，状態正常化のための汎用性のあるモデルを定義し，未考慮のトランザクションにも対応できるようにする必要がある．

プロキシを経由したシグナリングの検証

本論文では，UA と ALG はプロキシを介さず直接通信を行う構成をとったが，プロキシを経由した場合であっても，ALG が処理すべきメッセージの種類に違いはない．そのため本論文で提案した状態正常化シグナリングは，パス上に存在するプロキシに対しても有効であると考えられるが，これについては追加検証を行う必要がある．

状態正常化シグナリング開始契機の検討

本論文ではフェーズ I ～ V のそれぞれにおいて，セッションに異常が発生したかどうかを判断するためのタイマ T'1 ～ T'5 を設けた．これらのタイムアウト値は，状態正常化シグナリングを開始するかどうかを決定するための重要な値である．そのためこれらは条件を満たす範囲内で，明確な根拠に基づいて決定する必要がある．

またその一方で，タイマを利用しないアプローチも考えられる．SIP のトランスポートプロトコルとして UDP を利用する場合，本論文で想定したように，UA がシグナリングやメディア通信を行えない状態にあれば，そのホストからは，ICMP エラーメッセージ Type 3 (Destination Unreachable) が返される．そのため ALG では，これを契機として，状態正常化シグナリングを開始することもできる．この場合，タイマを利用する場合と比較して，さらに迅速な復旧が可能になると考えられる．

謝辞

本研究の遂行にあたり，多大なるご指導をいただいた後藤滋樹教授に深く感謝の意を表します．また研修を通して研究の機会を与えてくださいました，株式会社イメージパートナー天野潔代表取締役社長をはじめ，数多くの貴重なご助言をいただきました田中信顕氏，安川文隆氏，そして社員の皆様方に深く感謝の意を表します．最後に，日頃よりお世話をいただいている後藤研究室諸氏に感謝の意を表します．

参考文献

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC3261, Jun. 2002.

[2] ITU, H.323: Packet-based multimedia communications systems, `http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-H.323`

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC2616, Jun. 1999.

[4] J. Klensin, "Simple Mail Transfer Protocol", RFC2821, Apr. 2001.

[5] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC2246, Jan. 1999.

[6] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Idetifiers (URI) : Generic Syntax", RFC2396, Aug. 1998.

[7] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol", RFC2960, Oct. 2000.

[8] F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC3629, Nov. 2003.

[9] P. Resnick, "Internet Message Format", RFC2822, Apr. 2001.

[10] S. Donovan, "The SIP INFO Method", RFC2976, Oct. 2000.

[11] S. Petrack, L. Conroy, "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services", RFC2848, Jun. 2000.

[12] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method", RFC3311, Sep. 2002.

[13] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, “Session Initiation Protocol (SIP) Extension for Instant Messaging”, RFC3428, Dec. 2002.

[14] R. Sparks, “The Session Initiation Protocol (SIP) Refer Method”, RFC3515, Apr. 2003.

[15] J. Rosenberg, H. Schulzrinne, “Reliability of Provisional Responses in the Session Initiation Protocol (SIP)”, RFC3262, Jun. 2002.

[16] M. Handley, V. Jacobson, “SDP: Session Description Protocol”, RFC2327, Apr. 1998.

[17] J. Rosenberg, H. Schulzrinne, “An Offer/Answer Model with the Session Description Protocol (SDP)”, RFC3264, Jun. 2002.

[18] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, RFC3550, Jul. 2003.

[19] UPnP: Universal Plug and Play, `http://www.upnp.org/`

[20] J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC3489, Mar. 2003.

[21] B. Bigs, “A SIP Application Level Gateway for Network Address Translation”, Internet Draft (expired), Mar. 2000.

[22] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, K. Summers, “Session Initiation Protocol (SIP) Basic Call Flow Examples”, RFC3665, Dec. 2003.

[23] 林和仁, 柴田高穂, 諏訪裕一, 小幡洋昭, 黒木純一郎, “SIP アプリケーションレベルゲートウェイ (SIP-ALG) におけるアドレス変換エントリの管理方式に関する一考察”, 信学技報, vol.102, No.512, NS2002-188, pp.25–28, Dec. 2002.

[24] 黒木純一郎, 林和仁, 諏訪裕一, 柴田高穂, “SIP アプリケーションレベルゲートウェイ (SIP-ALG) におけるファイアウォール制御方式に関する一考察”, 信学技報, vol.102, No.691, NS2002-248, pp.105–108, Mar. 2003.

[25] 林和仁, 諏訪裕一, 柴田高穂, “SIP アプリケーションレベルゲートウェイ (SIP-ALG) における呼状態モデルに関する一検討”, 信学技報, vol.103, No.121, NS2003-33, pp.37–40, Jun. 2003.

[26] 匂坂岳志, 田中信顕, 後藤滋樹, “SIP-ALG におけるセッション管理方式の検討”, 信学技報, vol.103, No.506, NS2003-208, pp.5–8, Dec. 2003.

[27] SJ Labs VOIP software, http://www.sjlabs.com/

[28] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC1321, Apr. 1992.

[29] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, “HTTP Authentication : Basic and Digest Access Authentication”, RFC2617, Jun. 1999.

[30] 千村保文, 村田利文 監修, 『IDG 情報通信シリーズ SIP 教科書』, 株式会社 IDG ジャパン, 2003.

[31] Henry Sinnreich, Alan B. Jhonston 共著, 株式会社ソフトフロント 阪口克彦 監訳, 『マスタリング TCP/IP SIP 編』, オーム社, 2002.

[32] Gonzalo Camarillo 著, 新堀 幸一, 西尾 知子 監訳, 『SIP 入門』, 翔泳社, 2002.

[33] 大久保榮, 川島正久 監修, MCR (Multimedia Communications Research) 編, 『要点チェック式 H.323/MPEG-4 教科書』, IE インスティテュート, 2001.

[34] Colin Perkins 著，小川晃通 監訳, 『マスタリング TCP/IP RTP 編』, オーム社, 2004.

[35] Columbia University Department of Computer Science, SIP: Session Initiation Protocol, http://www.cs.columbia.edu/sip/

[36] Jonathan Rosenberg’s Home Page, http://www.jdrosen.net/

[37] iptel.org SIP Server: SIP Express Router, http://www.iptel.org/ser/

[38] SIP Express Media Server (Sems), http://sems.berlios.de/

[39] Asterisk - The Open Source Linux PBX, http://www.asterisk.org/

[40] .NET Messenger Service – 無料インスタント メッセージング サービス, http://www.microsoft.com/windows/messenger/ja/default.asp

[41] The VOIP Wiki - a reference guide to all things VOIP, http://www.voip-info.org/

[42] ソフトフロント，SIP 関連 RFC／ドラフト，http://www.softfront.co.jp/tech/sip_rfc_draft.html

[43] 日経 BP 社, IP 電話 start, http://itpro.nikkeibp.co.jp/denwa/

付録 A

状態正常化シグナリング – 別手法の検討

A.1 フェーズ I

図 A.1に，本論文で利用した手法とは別の手法に基づいて，フェーズ I から復旧するコールフローを示す.

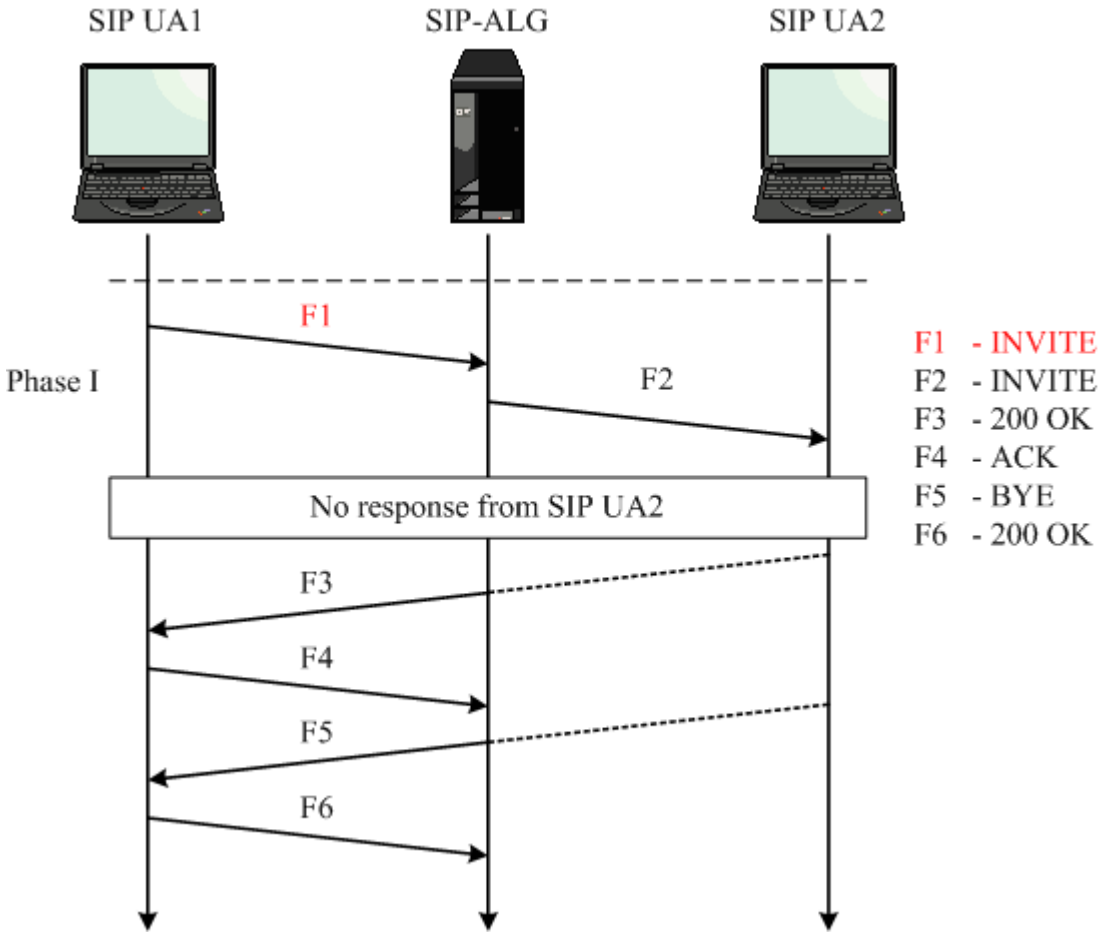

図 A.1: 別手法に基づくフェーズ I からの復旧コールフロー

A.1.1 シグナリングの詳細

ALG は UA1 に対して，まずリクエスト F1 に対する最終成功レスポンス F3 を返す．続いて UA1 からの F4 によって， ALG と UA1 間の INVITE トランザクションが正常形で終了する．ここで ALG と UA1 間でメディア通信が確立する．次に ALG は UA1 に対して， F5 によって，確立したメディア通信を終了させる． ALG は F5 をあたかも UA2 からのメッセージであるかのようにして生成する．最後に ALG は UA1 からの最終成功レスポンス F6 を傍受する．これにより， F5 によって開始された BYE トランザクションが正常形で終了する．

また ALG は UA2 に対しては，シグナリングを行わない．これは本論文で利用した手法に基づくフェーズ I からの復旧と同様である．

A.1.2 状態正常化の効果

別手法に基づくフェーズ I 状態正常化シグナリングでは，はじめに INVITE トランザクションにおける UAC (図 5.2) に対して， Calling ステートから Terminated ステートへの遷移を促す．続いて開始したメディア通信を終了させるため， ALG によって新しく BYE トランザクションが生成される．これによって， INVITE トランザクションにおける UAS は，新たに BYE トランザクションにおける UAS (図 5.4) となる． BYE トランザクションにおける UAS はこのとき Trying ステートにあり，自身で 2xx を生成することによって，Terminated ステートに移行する．このとき復旧に要する時間は T'1 + T(J) sec である．

本論文で利用した手法との比較

表 A.1に，別手法に基づくフェーズ I 状態正常化シグナリングの有効性についての検討結果を示す．別手法では，僅かながら迅速な復旧が可能となる．しかしその一方で， UA に対して一旦メディア通信を確立させるためのオーバヘッドが生じる．状態正常化ではリソースの浪費を最小限に抑えることが目的であるため，このような副作用は発生させるべきではない．そのため状態正常化シグナリングの手法としては，本論文で利用した手法が適切である[1]．

表 A.1: 別手法に基づくフェーズ I 状態正常化シグナリングの有効性

フェーズ	復旧の対象	本論文で利用した状態正常化シグナリング		別手法に基づく状態正常化シグナリング	
		復旧の可否	所要時間 (sec)	復旧の可否	所要時間 (sec)
I	INVITE-UAC	可能 ○	$T'1 + T(D)$ △ $T(D) \geq T(J)$	可能 ○	$T'1 + T(J)$ △ メディア通信確立のオーバヘッドを伴う

[1]ただし実際への応用では，フェーズ I では状態正常化シグナリングを行わず， UA での個別タイムアウトを発生させたほうが迅速な復旧が可能となる (表 5.1).

A.2 フェーズ II

図 A.2に，本論文で利用した手法とは別の手法に基づいて，フェーズ II から復旧するコールフローを示す．ただしここで ALG と UA1 間，および ALG と UA2 間のシグナリングは，いずれも図 A.2の順序であればよく，同期する必要はない．

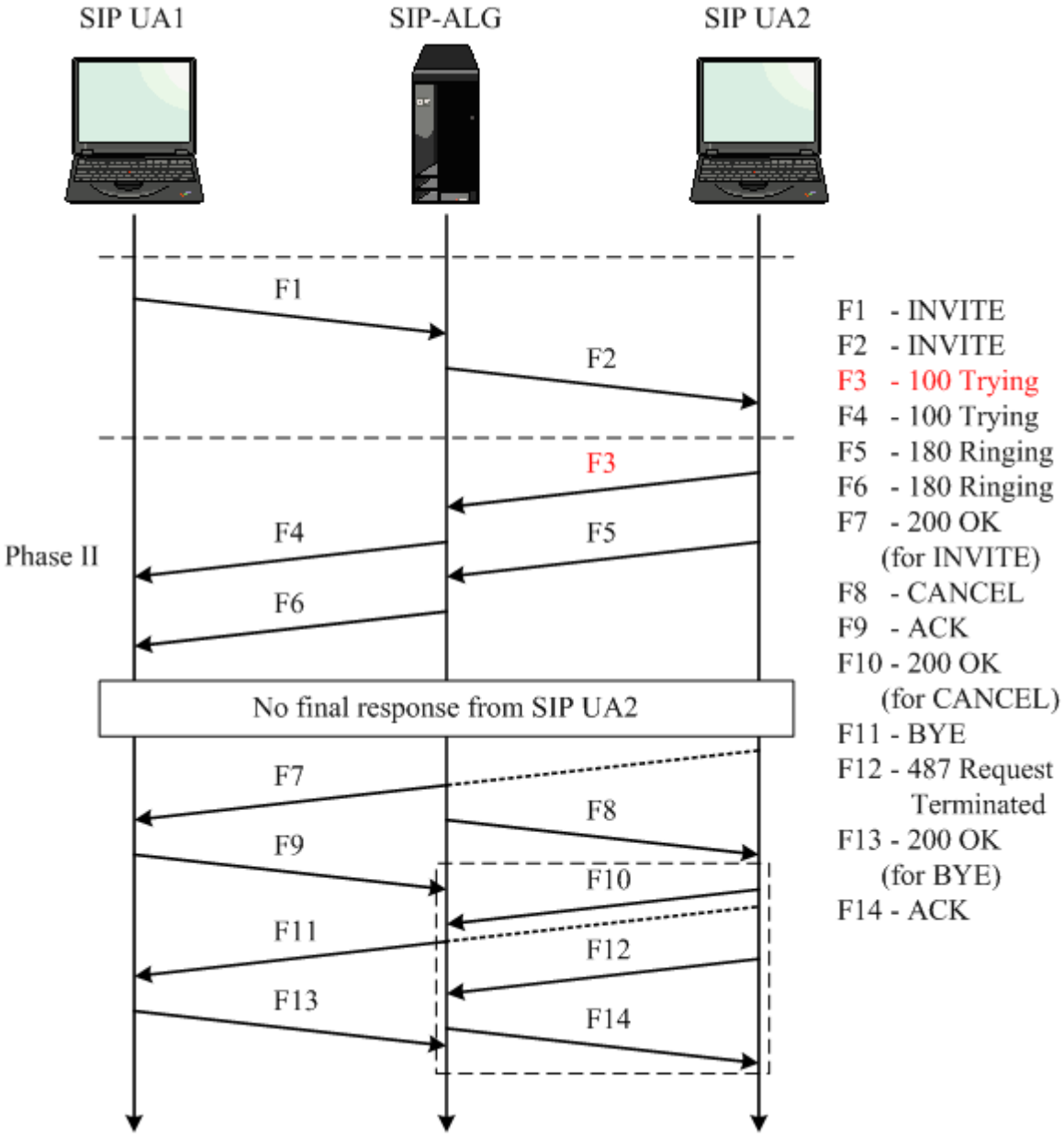

図 A.2: 別手法に基づくフェーズ II からの復旧コールフロー

A.2.1 シグナリングの詳細

ALG と UA1 間での状態正常化シグナリングは，別手法に基づくフェーズ I からの復旧と同様である．つまり F7， F9， F11， F13 はこの順序で，図 A.1での F3， F4， F5， F6 に対応する．

また ALG と UA2 間の状態正常化シグナリングは，本論文で利用した手法に基づくフェーズ II からの復旧と同様である．つまり F8， F10， F12， F14 はこの順序で，図 4.3での F8， F10， F11， F12 に対応する．

A.2.2 状態正常化の効果

UAC に対する効果

別手法に基づくフェーズ II 状態正常化シグナリングでは， INVITE トランザクションにおける UAC (図 5.2) に対して， Proceeding ステートから Terminated ステートへの遷移を促す．開始したメディア通信を終了させる手続きは，別手法に基づくフェーズ I からの復旧の場合と同様である．このとき復旧に要する時間は T'2 + T(J) sec である．

UAS に対する効果

別手法に基づくフェーズ II 状態正常化シグナリングでは， INVITE トランザクションにおける UAS (図 5.3) に対する状態正常化の効果は，本論文で利用した手法に基づくフェーズ II からの復旧の場合と同様である．このとき復旧に要する時間は T'2 + T(I) sec である．

本論文で利用した手法との比較

表 A.2に，別手法に基づくフェーズ II 状態正常化シグナリングの有効性についての検討結果を示す．別手法に基づくフェーズ I からの復旧の場合と同様に，状態正常化シグナリングの手法としては，本論文で利用した手法が適切である．

表 A.2: 別手法に基づくフェーズ II 状態正常化シグナリングの有効性

フェーズ	復旧の対象	本論文で利用した状態正常化シグナリング		別手法に基づく状態正常化シグナリング	
		復旧の可否	所要時間 (sec)	復旧の可否	所要時間 (sec)
II	INVITE-UAC	可能 ○	$T'2 + T(D)$ △ $T(D) \geq T(J)$	可能 ○	$T'2 + T(J)$ △ メディア通信確立のオーバヘッドを伴う
	INVITE-UAS	可能 ○	$T'2 + T(I)$ △ 一定時間内にユーザからの応答がなければ異常とみなす	可能 ○	$T'2 + T(I)$ △ 一定時間内にユーザからの応答がなければ異常とみなす

付録 B

SIP 詳細仕様

B.1 SIP ステータスコード一覧

B.1.1 暫定レスポンス

Status-Code	Reason-Phrase	内容
100	Trying	試行中
180	Ringing	呼び出し中
181	Call Is Being Forwarded	呼が転送されている
182	Queued	キューに入れられた
183	Session Progress	セッションの進捗状況

B.1.2 成功レスポンス

Status-Code	Reason-Phrase	内容
200	OK	成功

B.1.3 リダイレクトレスポンス

Status-Code	Reason-Phrase	内容
300	Multiple Choices	複数の選択肢がある
301	Moved Permanently	恒久的に移動した
302	Moved Temporarily	一時的に移動した
305	Use Proxy	プロキシを使用せよ
380	Alternative Service	代替サービス

B.1.4 クライアントエラーレスポンス

Status-Code	Reason-Phrase	内容
400	Bad Request	不正なリクエスト
401	Unauthorized	認可されていない
402	Payment Required	料金支払いが必要
403	Forbidden	禁止
404	Not Found	見つからない
405	Method Not Allowed	メソッドが許可されていない
406	Not Acceptable	受け入れられない
407	Proxy Authentication Required	プロキシ認証が必要
408	Request Timeout	リクエストがタイムアウトした
410	Gone	リソースが既に存在しない
413	Request Entity Too Large	リクエストのエンティティが大きすぎる
414	Request-URI Too Long	リクエスト URI が長すぎる
415	Unsupported Media Type	サポートされていないメディアタイプ
416	Unsupported URI Scheme	サポートされていない URI スキーム
420	Bad Extension	不正な拡張
421	Extension Required	拡張が必要
423	Interval Too Brief	間隔が短すぎる
480	Temporarily Unavailable	一時的に利用不可
481	Call/Transaction Does Not Exist	呼またはトランザクションが存在しない
482	Loop Detected	ループが検知された
483	Too Many Hops	ホップが多すぎる
484	Address Incomplete	アドレスが不完全
485	Ambiguous	不明瞭
486	Busy Here	ここは現在ビジー
487	Request Terminated	リクエストが終了させられた
488	Not Acceptable Here	ここでは受け入れ不能
491	Request Pending	リクエストペンディング
493	Undecipherable	解読不能

B.1.5 サーバエラーレスポンス

Status-Code	Reason-Phrase	内容
500	Server Internal Error	サーバ内部エラー
501	Not Implemented	実装されていない

Status-Code	Reason-Phrase	内容
502	Bad Gateway	不正なゲートウェイ
503	Service Unavailable	サービスを利用できない
504	Server Time-out	サーバタイムアウト
505	Version Not Supported	サポートされていないバージョン
513	Message Too Large	メッセージが大きすぎる

B.1.6 グローバルエラーレスポンス

Status-Code	Reason-Phrase	内容
600	Busy Everywhere	どの場所もビジー
603	Decline	辞退
604	Does Not Exist Anywhere	どこにも存在しない
606	Not Acceptable	受け入れ不能

B.2 SIP ヘッダフィールド一覧

ヘッダフィールド	内容
Accept	受け入れ可能なメディアタイプを示す
Accept-Encoding	受け入れ可能なエンコーディングを示す
Accept-Language	受け入れ可能な言語を示す
Alert-Info	警告情報を示す
Allow	UA がサポートするメソッドの組を示す
Authentication-Info	相互認証に HTTP ダイジェスト認証を使用することを示す
Authorization	UA の認証情報を示す
Call-ID	呼識別子を示す
Call-Info	呼に対する付加的な情報を示す
Contact	ユーザの表示名や URI を示す
Content-Disposition	メッセージボディが UA によってどのように解釈されるかを示す
Content-Encoding	メッセージボディのエンコーディングを示す
Content-Language	メッセージボディの言語を示す
Content-Length	メッセージボディの長さを示す
Content-Type	メッセージボディのタイプを示す
CSeq	メッセージのシーケンス番号を示す
Date	日付を示す

ヘッダフィールド	内容
Error-Info	エラーステータスについての付加的な情報を示す
Expires	メッセージの有効期限を示す
From	リクエストのイニシエータ (生成元) を示す
In-Reply-To	呼が参照する，または折り返される Call-ID を示す
Max-Forwards	リクエストを転送できるダウンストリームサーバの数を示す
Min-Expires	最小の登録更新間隔を示す
MIME-Version	MIME バージョンを示す
Organization	メッセージを生成するエンティティが所属する組織の名称を示す
Priority	クライアントが認識するリクエストの緊急度を示す
Proxy-Authenticate	プロキシに対する認証チャレンジを示す
Proxy-Authorization	プロキシに対する UA の認証情報を示す
Proxy-Require	プロキシが理解する機能を示す
Record-Route	それ以降のリクエストが経由すべきプロキシを示す
Reply-To	論理的な折り返し先 URI を示す
Require	使用する SIP の拡張機能を示す
Retry-After	サービスに対するリクエストの再試行間隔を示す
Route	リクエストが経由すべきプロキシを示す
Server	UAS のソフトウェアの情報を示す
Subject	呼の概要または性質を示す
Supported	UA がサポートするすべての拡張機能を示す
Timestamp	リクエストが送られた時間を示す
To	リクエストの論理的な受信者を示す
Unsupported	UAS がサポートしない機能を示す
User-Agent	UAC のソフトウェアの情報を示す
Via	リクエストとレスポンスのシグナリングパスを示す
Warning	レスポンスステータスについての付加的な情報を示す
WWW-Authenticate	WWW 認証チャレンジを示す

B.3 コールフローの例

B.3.1 ロケーションの登録

REGISTER - 401 Unauthorized - REGISTER - 200 OK

UA はセッションの開始に先立ち，自身のロケーション情報をサーバに登録する．プロキシによる認証を伴ったロケーション登録時のコールフロー[1]を図 B.1に示す．

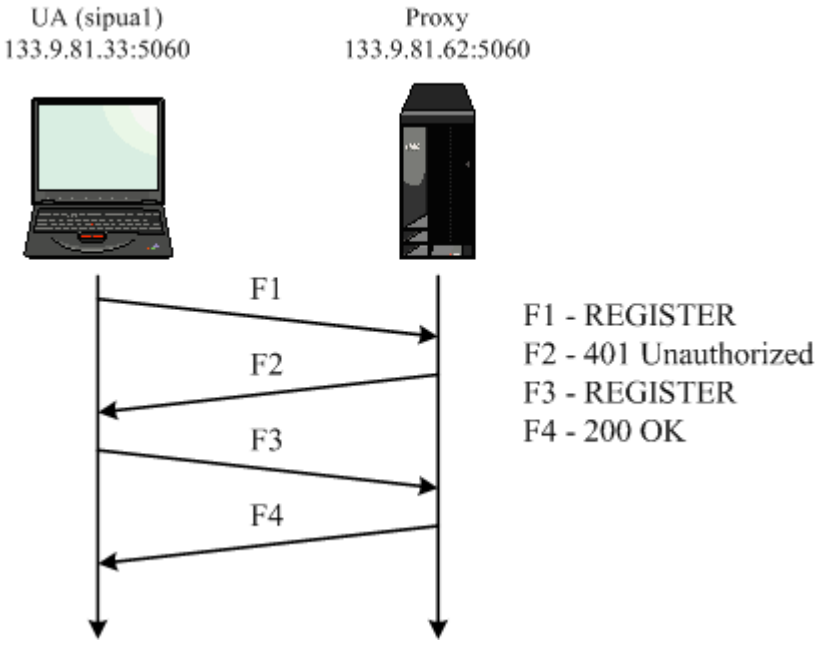

図 B.1: ロケーション登録時のコールフロー

F1 - REGISTER

```
REGISTER sip:133.9.81.62 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.33;rport;branch=z9hG4bK850951210131c9b14185b5c60000389e00000001
To: <sip:sipua1@133.9.81.62>
From: <sip:sipua1@133.9.81.62>;tag=23591393724876
Contact: <sip:sipua1@133.9.81.33:5060>;events="message-summary"
Call-ID: A16C9A6E-3D2A-443F-B5D6-ABACC8638BDF@133.9.81.33
CSeq: 1 REGISTER
Max-Forwards: 70
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0
```

[1]ここに挙げた例ではプロキシがレジストラとしての機能も備えている．

UA (sipua1) はまず認証情報をもたない通常の REGISTER リクエストによって，プロキシに対して自身のコンタクト SIP URI (sip:sipua1@133.9.81.33:5060) の登録を試みる．

F2 - 401 Unauthorized

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185b5c60000389e00
000001
To: <sip:sipua1@133.9.81.62>;tag=b27e1a1d33761e85846fc98f5f3a7e58.db9e
From: <sip:sipua1@133.9.81.62>;tag=23591393724876
Call-ID: A16C9A6E-3D2A-443F-B5D6-ABACC8638BDF@133.9.81.33
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="sipsrv.goto.info.waseda.ac.jp", nonce="4185b73e7d9b3e0fe8a8
b72b1897cd6cec2478df"
Server: Sip EXpress router (0.8.14 (i386/linux))
Content-Length: 0
```

ここでプロキシは UA からのロケーション登録にダイジェスト認証 [28] [29] を要求する．プロキシは UA に対して, WWW 認証チャレンジ (WWW-Authenticate ヘッダフィールドに記述される) を含んだレスポンスを返す．

F3 - REGISTER

```
REGISTER sip:133.9.81.62 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.33;rport;branch=z9hG4bK850951210131c9b14185b5c6000010b300000003
To: <sip:sipua1@133.9.81.62>
From: <sip:sipua1@133.9.81.62>;tag=23591403111740
Contact: <sip:sipua1@133.9.81.33:5060>;events="message-summary"
Call-ID: A16C9A6E-3D2A-443F-B5D6-ABACC8638BDF@133.9.81.33
CSeq: 2 REGISTER
Authorization: Digest username="sipua1",realm="sipsrv.goto.info.waseda.ac.jp",nonce="4185b73e7
d9b3e0fe8a8b72b1897cd6cec2478df",uri="sip:133.9.81.62",response="0cee87b52778ea3f7ddd115700f
6fdc2"
Max-Forwards: 70
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0
```

UA はプロキシからの WWW 認証チャレンジに基づいて，ダイジェスト認証のためのレスポンスパラメータを生成する． UA は認証情報 (Authorization ヘッダフィールドに記述される) を

付加した新しい REGISTER リクエストによって，プロキシに対して再度ロケーションの登録を試みる．

F4 - 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185b5c6000010b300
000003
To: <sip:sipua1@133.9.81.62>;tag=b27e1a1d33761e85846fc98f5f3a7e58.f375
From: <sip:sipua1@133.9.81.62>;tag=23591403111740
Contact: <sip:sipua1@133.9.81.33:5060>;q=0.00;expires=3600
Call-ID: A16C9A6E-3D2A-443F-B5D6-ABACC8638BDF@133.9.81.33
CSeq: 2 REGISTER
Server: Sip EXpress router (0.8.14 (i386/linux))
Content-Length: 0
```

プロキシは UA からの認証情報をチェックし，正しいことを確認した後，ロケーションサーバへの登録を行う．例では UA のコンタクト SIP URI (sip:sipua1@133.9.81.33:5060) は，論理的な SIP URI (sip:sipua1@133.9.81.62) に有効期限 3600 秒 (=1 時間) で関連付けられている．

B.3.2 セッションの開始

INVITE - 100 Trying/180 Ringing - 200 OK - ACK

UA はプロキシに対するロケーションの登録が完了すると，プロキシを利用したセッションの確立が可能となる．プロキシによる認証を伴わないセッション開始時のコールフロー[2]を図 B.2に示す．

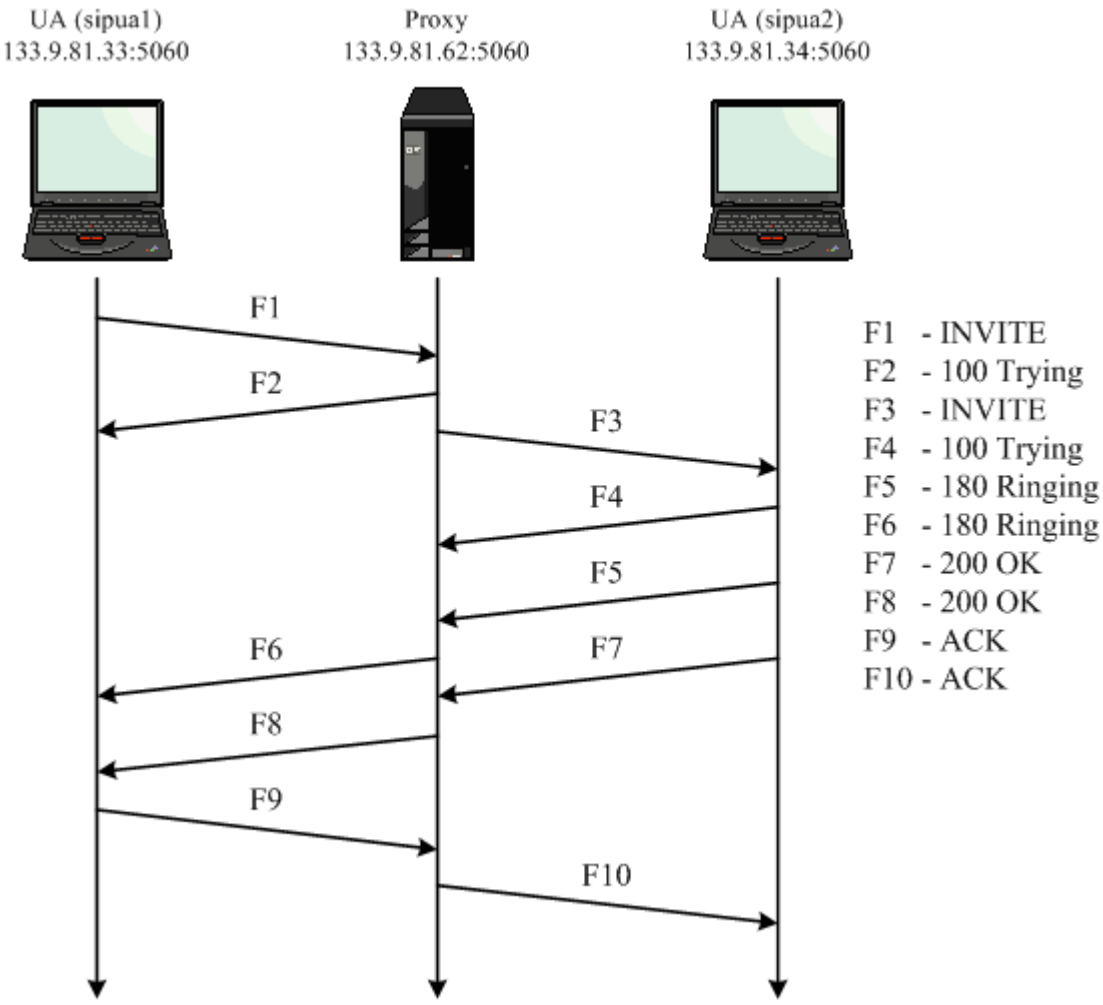

図 B.2: セッション開始時のコールフロー

[2]ここでは UA (sipua1 および sipua2) は，いずれもプロキシに対するロケーションの登録が事前に完了しているものとする． sipua1 は sip:sipua1@133.9.81.62， sipua2 は sip:sipua2@133.9.81.62 をそれぞれ論理的な SIP URI としてもつ．

F1 - INVITE

```
INVITE sip:sipua2@133.9.81.62 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.33;rport;branch=z9hG4bK850951210131c9b14185bc7900001f73000000ad
To: <sip:sipua2@133.9.81.62>
From: “sipua1”<sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Max-Forwards: 70
User-Agent: SJLabs-SJphone/1.30.252
Content-Type: application/sdp
Content-Length: 264

v=0
o=- 3308272376 3308272376 IN IP4 133.9.81.33
s=SJphone
c=IN IP4 133.9.81.33
t=0 0
a=direction:active
m=audio 16384 RTP/AVP 3 8 0 101
a=rtpmap:3 GSM/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11,16
```

UAC (sipua1) は UAS (sipua2) に対して，INVITE リクエストによってセッションの開始を要求する．メッセージのボディ部には UAC からの SDP オファー記述が含まれる．

F2 - 100 Trying

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: <sip:sipua2@133.9.81.62>
From: “sipua1”<sip:sipua1@133.9.81.62>;tag=23762854628776
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: Sip EXpress router (0.8.14 (i386/linux))
Content-Length: 0
```

プロキシは UAC に対してリクエストが試行中であることを通知する暫定レスポンスを返す.

F3 - INVITE

```
INVITE sip:sipua2@133.9.81.34:5060 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bK026f.154e4ae2.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: <sip:sipua2@133.9.81.62>
From: “sipua1”<sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Max-Forwards: 69
User-Agent: SJLabs-SJphone/1.30.252
Content-Type: application/sdp
Content-Length: 264

(ボディ部は省略)
```

プロキシは UAC からリクエストを受け取ると，ロケーションサーバを利用して適切な宛先を決定して送り出す．例では To ヘッダフィールドの値 (sip:sipua2@133.9.81.62) から，プロキシはこのリクエストが sipua2 宛のものであることを判断し，事前に sipua2 のロケーションとして登録されているドメイン (133.9.81.34:5060) へ送り出す.

F4 - 100 Trying

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bK026f.154e4ae2.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: “sipua2”<sip:sipua2@133.9.81.62>
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0
```

UAS はプロキシに対してリクエストが試行中であることを通知する暫定レスポンスを返す.

F5 - 180 Ringing

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bK026f.154e4ae2.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0
```

UAS は UAC からセッション開始リクエストがあったことをユーザに知らせるために呼び出し音を鳴らす．UAS はプロキシに対して，呼び出し音を鳴らしたことを通知する暫定レスポンスを返す．ここで early ダイアログが確立される．

F6 - 180 Ringing

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0
```

プロキシは UAS からのレスポンスを受け取ると，リクエストが辿ったシグナリングパスを戻るように適切な宛先を決定して送り出す．例では Via ヘッダフィールドの値 (133.9.81.33) からプロキシはレスポンスの宛先を判断し，sipua1 のドメイン (133.9.81.33:5060) へ送り出す．

F7 - 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bK026f.154e4ae2.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Type: application/sdp
Content-Length: 216

v=0
o=- 3308272340 3308272340 IN IP4 133.9.81.34
s=SJphone
c=IN IP4 133.9.81.34
t=0 0
a=direction:active
m=audio 16384 RTP/AVP 3 101
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11,16
```

ユーザが呼び出しに応じると，UAS はプロキシに対してユーザがリクエストを了承したことを通知する最終レスポンスを返す．メッセージのボディ部には UAS からの SDP アンサー記述が含まれる．ここで confirmed ダイアログが確立される．

例では F1 - INVITE および F7 - 200 OK のトランザクションによって，sipua1 と sipua2 との間でメディアセッションのセットアップに関するオファー / アンサーが行われている．ここで，sipua1 はメディアセッションでの IP アドレスとして 133.9.81.33 を利用し，RTP に 16384 番ポートを利用する．一方，sipua2 はメディアセッションでの IP アドレスとして 133.9.81.34 を利用し，RTP に 16384 番ポートを利用する．両者共通のコーデックには GSM を利用する．

F8 - 200 OK

SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc7900001f7300
0000ad
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 INVITE
Server: SJLabs-SJphone/1.30.252
Content-Type: application/sdp
Content-Length: 216

(ボディ部は省略)

プロキシは UAS からのレスポンスを受け取り， UAC に対して送り出す.

F9 - ACK

ACK sip:sipua2@133.9.81.34:5060 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.33;rport;branch=z9hG4bK850951210131c9b14185bc80000020e9000000b1
To: <sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 ACK
Max-Forwards: 70
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0

UAC はプロキシを経由して UAS からのレスポンスを受け取ると，トランザクションを終了させるための ACK リクエストをプロキシに対して送り出す.

F10 - ACK

```
ACK sip:sipua2@133.9.81.34:5060 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62;branch=0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc80000020e900
0000b1
To: <sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 1 ACK
Max-Forwards: 69
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0
```

プロキシは UAC からのリクエストを受け取り，UAS に対して送り出す．この後，UA 間で RTP を利用したメディアによる双方向通信が可能となる．

B.3.3 セッションの終了

BYE - 200 OK

UA 間でメディアによる通信を終えるためには，確立したセッションを終了するための手続きをとる必要がある．セッション終了時のコールフローを図 B.3に示す．

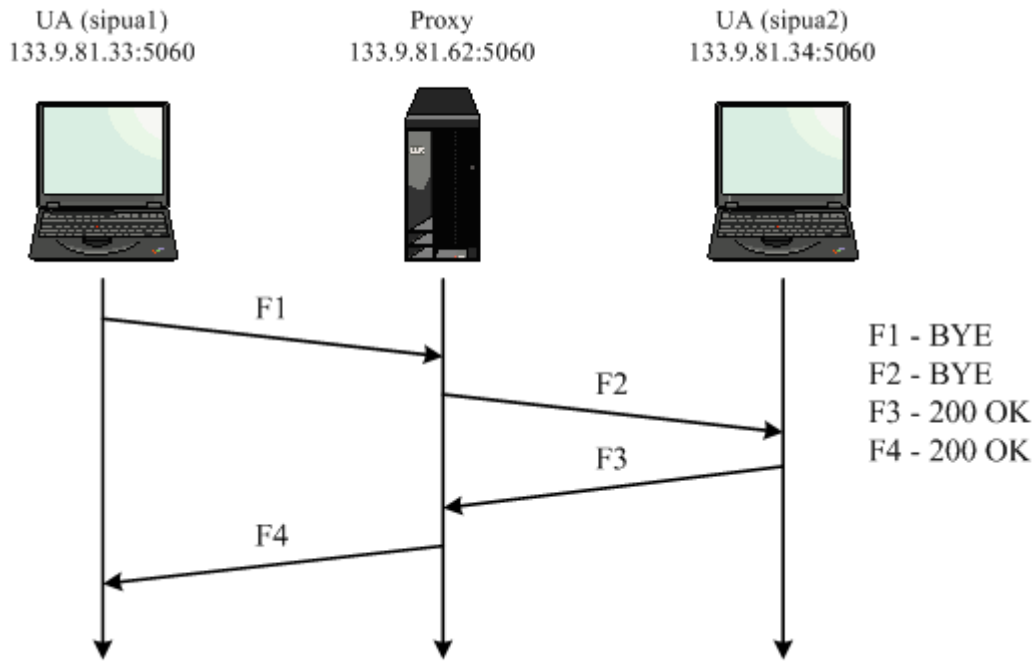

図 B.3: セッション終了時のコールフロー

F1 - BYE

```
BYE sip:sipua2@133.9.81.34:5060 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.33;rport;branch=z9hG4bK850951210131c9b14185bc8a00007604000000b4
To: <sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 2 BYE
Max-Forwards: 70
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0
```

UAC (sipua1) は UAS (sipua2) に対して， BYE リクエストによってセッションの終了を要求する．この時点で， sipua1 での RTP 送受信は停止される．

F2 - BYE

```
BYE sip:sipua2@133.9.81.34:5060 SIP/2.0
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bKd16f.fe67f817.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc8a0000760400
0000b4
To: <sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua1@133.9.81.33:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 2 BYE
Max-Forwards: 69
User-Agent: SJLabs-SJphone/1.30.252
Content-Length: 0
```

プロキシは UAC からのリクエストを受け取り，UAS に対して送り出す．

F3 - 200 OK

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.62;branch=z9hG4bKd16f.fe67f817.0
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc8a0000760400
0000b4
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 2 BYE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0
```

UAS はプロキシを経由して UAC からのセッション終了リクエストを受け取ると，(ユーザによる操作を伴わず) プロキシに対して，リクエストを了承したことを通知する最終レスポンスを返す．この時点で，sipua2 での RTP 送受信は停止される．

F4 - 200 OK

SIP/2.0 200 OK
Via: SIP/2.0/UDP 133.9.81.33;rport=5060;branch=z9hG4bK850951210131c9b14185bc8a0000760400
0000b4
To: “sipua2”<sip:sipua2@133.9.81.62>;tag=18069668719403
From: <sip:sipua1@133.9.81.62>;tag=23762854628776
Contact: <sip:sipua2@133.9.81.34:5060>
Call-ID: F3985577-FCE3-4CE4-A025-1E39F865A2AF@133.9.81.33
CSeq: 2 BYE
Server: SJLabs-SJphone/1.30.252
Content-Length: 0

プロキシは UAS からのレスポンスを受け取り，UAC に対して送り出す．セッション開始時とは異なり，最終レスポンスに対する ACK リクエストは生成されない[3]．

[3]INVITE トランザクションとは異なり，非 INVITE トランザクションは 2xx レスポンスに対する特別なハンドリングがない．

B.4 各種タイマ

タイマ	値	意味
T1	500ms default	RTT 予測値
T2	4s	非 INVITE リクエストおよび INVITE に対する応答のための最大送信間隔
T4	5s	メッセージがネットワーク上に残存する最大期間
Timer A	initially T1	UDP だけのための INVITE リクエストの再送間隔
Timer B	64*T1	INVITE トランザクションのタイムアウトタイマ
Timer C	> 3min	プロキシの INVITE トランザクションのタイムアウト
Timer D	> 32s for UDP 0s for TCP/SCTP	応答の再送のための待ち時間
Timer E	initially T1	UDP だけのための非 INVITE リクエストの再送間隔
Timer F	64*T1	非 INVITE トランザクションのタイムアウトタイマ
Timer G	initially T1	INVITE に対する応答の再送間隔
Timer H	64*T1	ACK 受信のための待ち時間
Timer I	T4 for UDP 0s for TCP/SCTP	ACK 再送のための待ち時間
Timer J	64*T1 for UDP 0s for TCP/SCTP	非 INVITE リクエストの再送のための待ち時間
Timer K	T4 for UDP 0s for TCP/SCTP	応答の再送のための待ち時間