輪講資料

ICE(Interactive Connectivity Establishment)

名城大学　理工学部　情報工学科

渡邊研究室　B4

140441043　鴨下友馬

ICE 基本事項

- RFC5245で標準化されているUDPベースのNAT越えプロトコル
- オファー／アンサーモデルを利用する任意のプロトコルで使用可能である.
  - **オファー／アンサーモデル**：セッションを開始するために，一方が自らの視点から情報を提供し（オファー），もう一方が相手に対応する情報を返答する（アンサー）モデル．RFC3264で標準化.
  - オファー／アンサーモデルを利用するプロトコルの例：
    - **SIP(Session Initiation Protocol)**：音声や映像，テキストメッセージの交換などを行うために必要なセッションの生成・変更・切断を行うプロトコル
- STUN，TURNという2つのプロトコルを使用する.
  - STUNおよびTURNについては，以降で解説する.

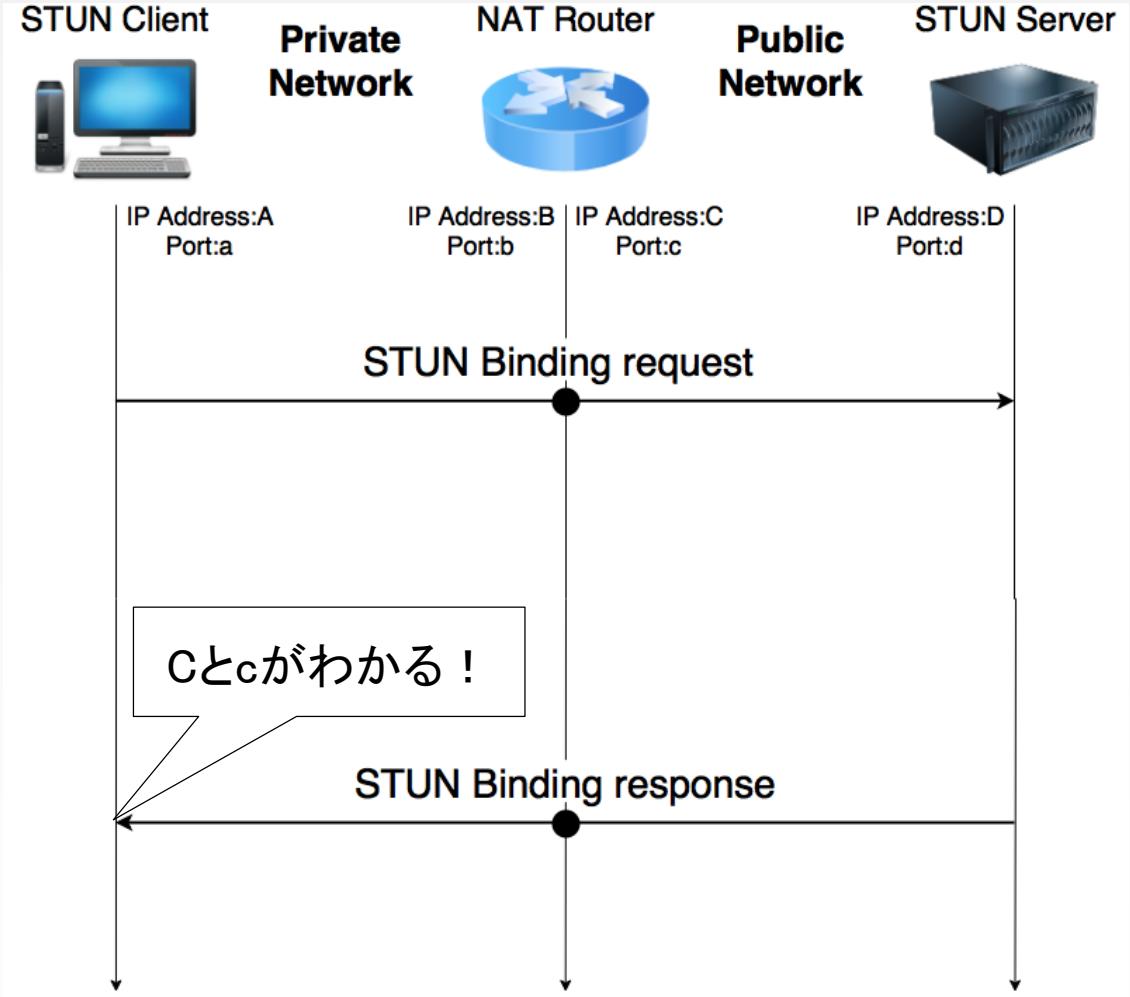
STUN

(Session Traversal Utilities for NAT)

STUN 基本事項

- RFC5389で標準化されているクライアントサーバプロトコル
  - オリジナルはRFC3489（Simple Traversal of User Datagram Protocol Through NAT）だが，これの登場により廃止．
- NAT越え問題を解決するために使用される．
  - ただし，STUN自身がNAT越え問題の解決策というわけではない．
  - STUNはNAT越え問題を解決するための一部として利用される．
- サーバ側で，NATのグローバルIPアドレスとポート番号を写像し，クライアントに渡す．
- クライアントは，自身がサーバ側ではどのように見えているかを知ることができる．

STUN 動作概要



- STUN Binding request
  - Bindingの要求を行う.
  - Serverに到着すると，送信元のIPアドレスおよびポート番号は，NATによりそれぞれCおよびcとなっている.
- STUN Binding response
  - 上記のCおよびcをパケットのデータ部にコピーしてClientに送信する.
  - Clientに到着すると，宛先のIPアドレスおよびポート番号は，NATによりAおよびaとなるが，内部データは変更されない.

TURN

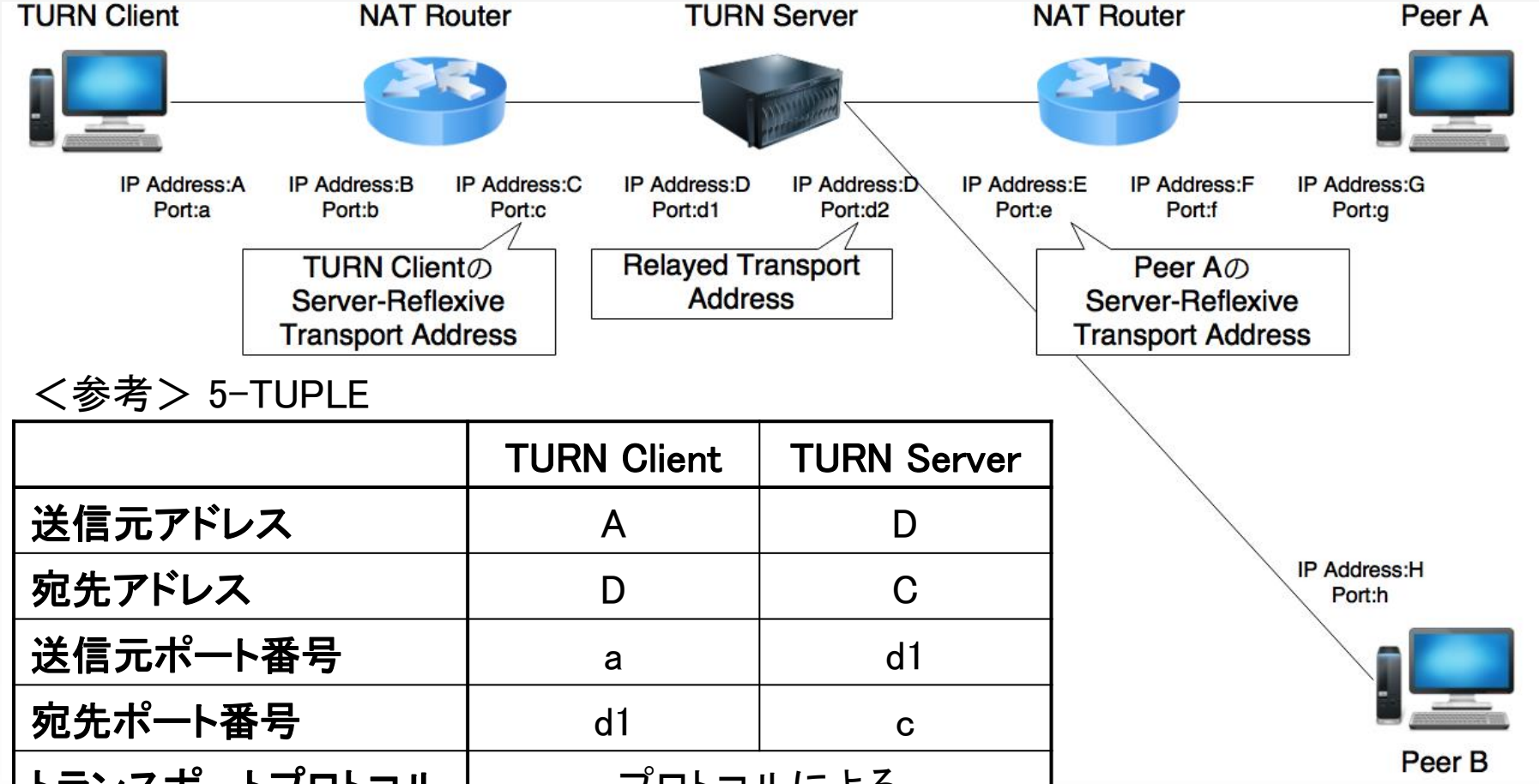
(Traversal Using Relays around NAT)

TURN 基本事項

- RFC5766で標準化されているクライアントサーバプロトコル
- STUN（前述）の拡張版
  - TURNメッセージのほとんどは，STUNメッセージと同一形式である．
- クライアントが，サーバに対してリレー通信を要求する．
- ファイアウォールセキュリティを迂回する可能性があるため，アクセス許可を必要とする．
- クライアントがサーバにデータパケットを送信すると，サーバは自身を送信元としてピアに中継する．
- クライアントとサーバは，5-TUPLEと呼ばれる情報を持つ．
  - **5-TUPLE**：送信元アドレス，宛先アドレス，送信元ポート番号，宛先ポート番号，トランスポートプロトコル（UDP, TCP, TLSなど）から成る情報．

TURN 動作概要 (1)

以下のようなネットワーク構成を仮定する.

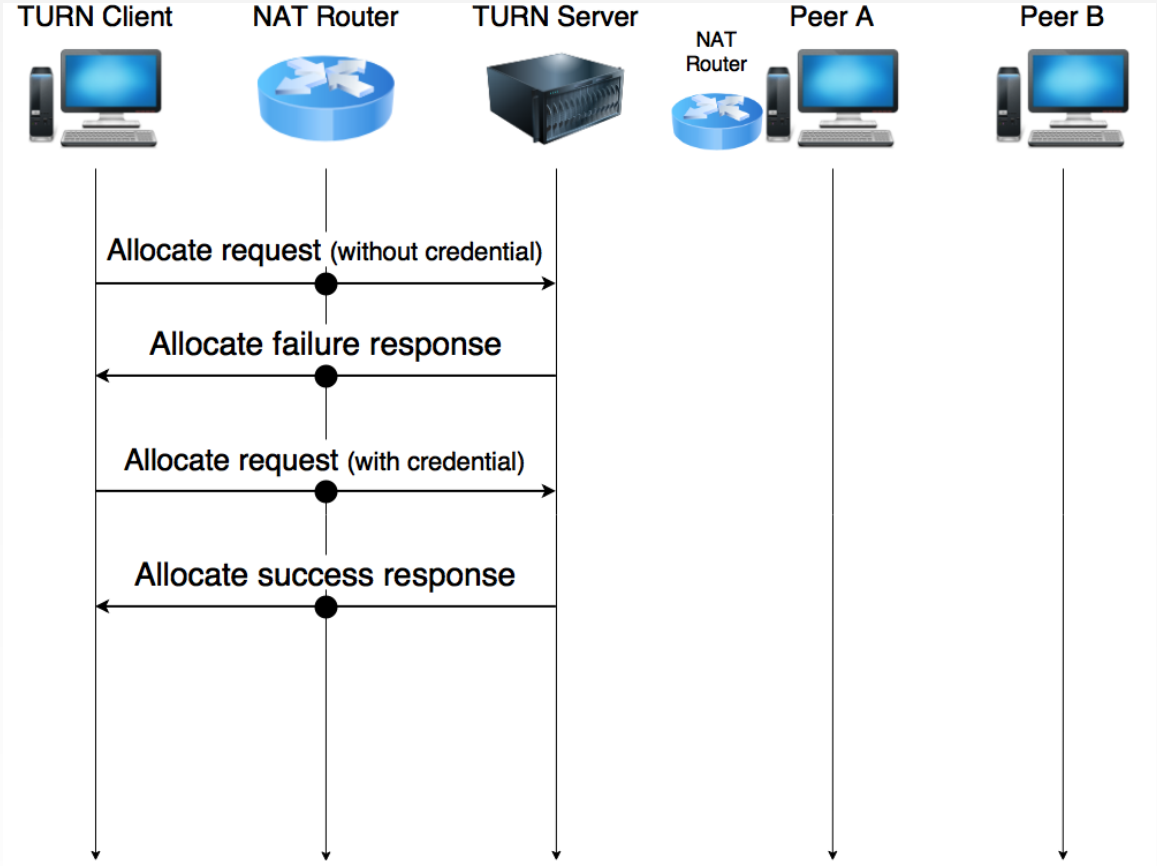


＜参考＞ 5-TUPLE

	TURN Client	TURN Server
送信元アドレス	A	D
宛先アドレス	D	C
送信元ポート番号	a	d1
宛先ポート番号	d1	c
トランスポートプロトコル	プロトコルによる	

TURN 動作概要 (2)

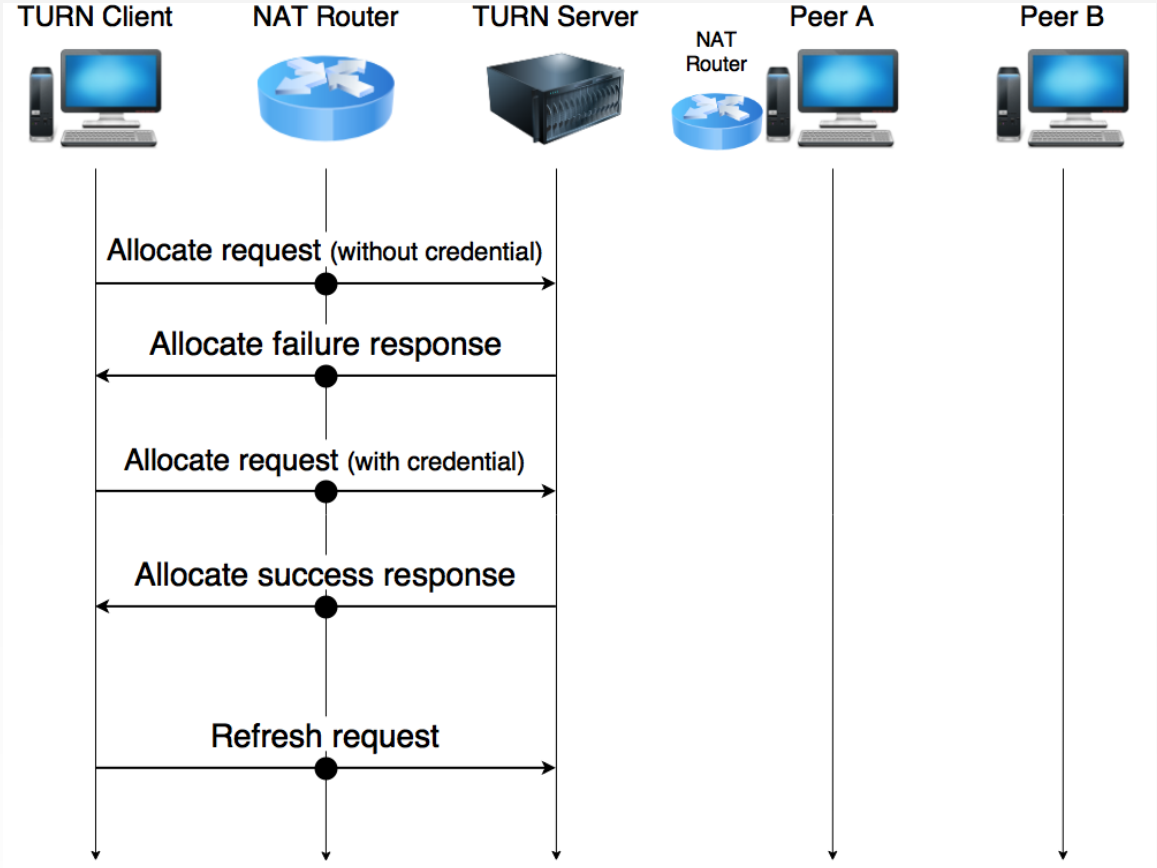
- Allocation（割り当て）
  TURNでは，サーバ上に割り当てを行う必要がある.



- credential（資格情報）
  - ユーザ名とパスワード
  - Serverを使用する権限を持っていることを示すため，これを用いて自身を認証する.
- Allocate response
  - Clientの認証に失敗したら，失敗応答を返す.
  - Clientの認証に成功したら，Relayed Transport Address（前出のDとd2）を返す.

TURN 動作概要 (3)

- Allocation（割り当て）

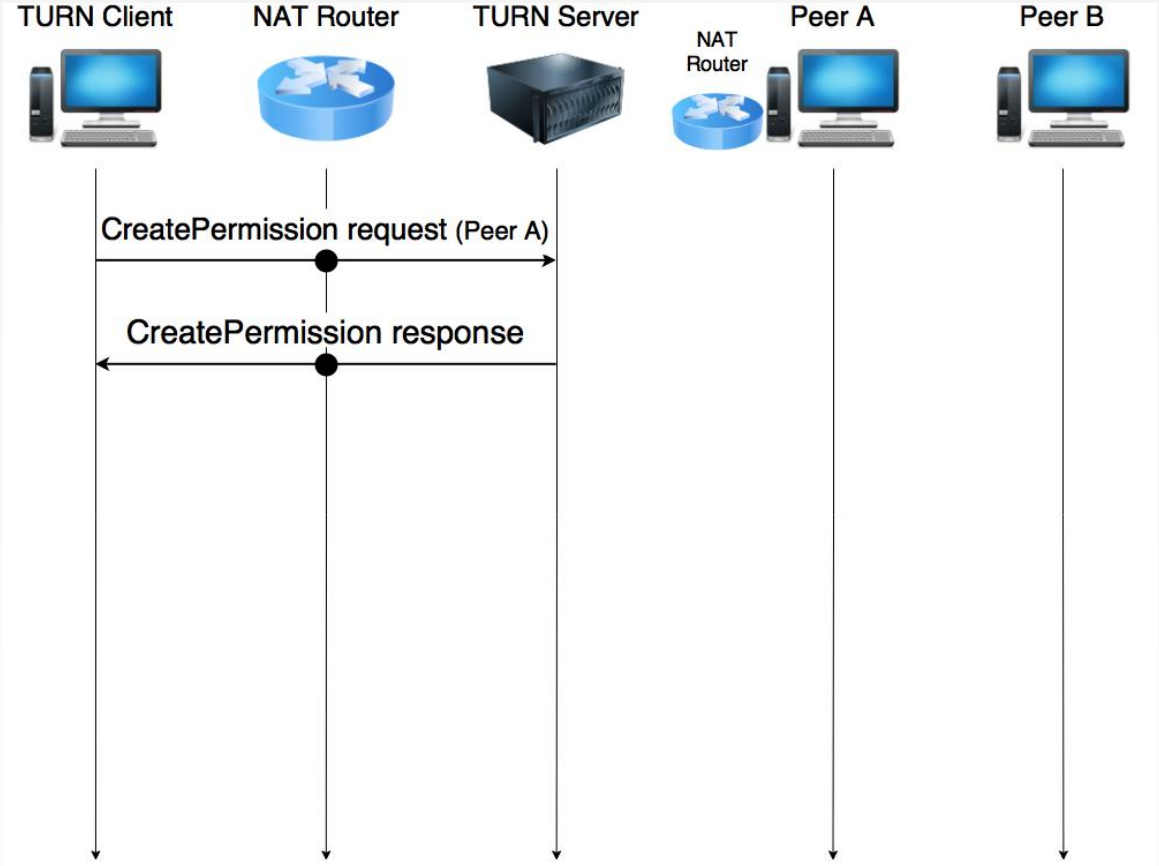


- Refresh request
  - 割り当てを有効に保つため，ClientがServerに定期的に送信する．
  - 「割り当ての有効時間内に再度Refresh requestを送信しない」あるいは「有効時間0のRefresh requestを送信する」と，割り当てが終了される．

TURN 動作概要 (4)

- Send Mechanism（送信メカニズム）

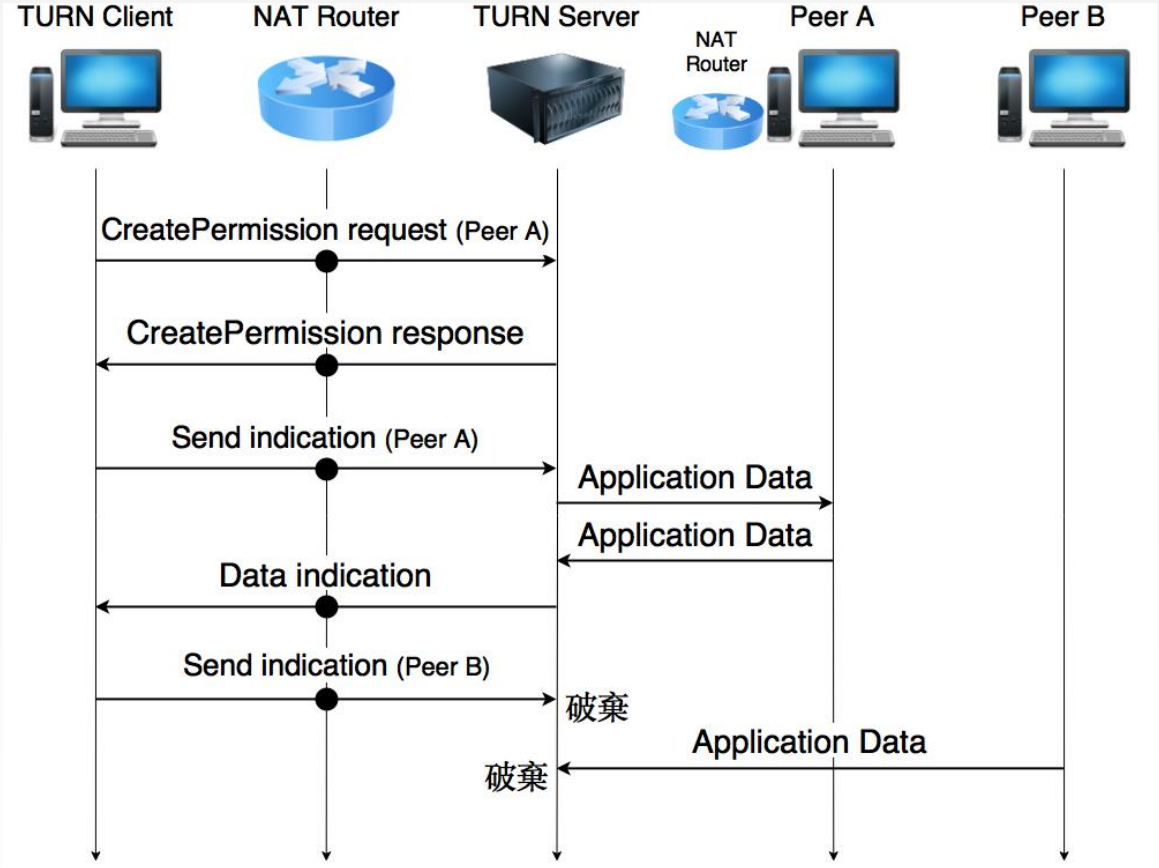
  TURN ClientとPeerが，TURN Serverを使用してデータ交換をする方法.



- CreatePermission request
  - ここでは，通信相手としてPeer Aを指定する.
  - Clientがこのアクセス許可を行わない場合，認証していないPeerとの送受信は全て無効となり，Serverによって破棄される.
- CreatePermission response
  - Serverがこれを返すことで，アクセスが許可される.

TURN 動作概要 (5)

- Send Mechanism（送信メカニズム）

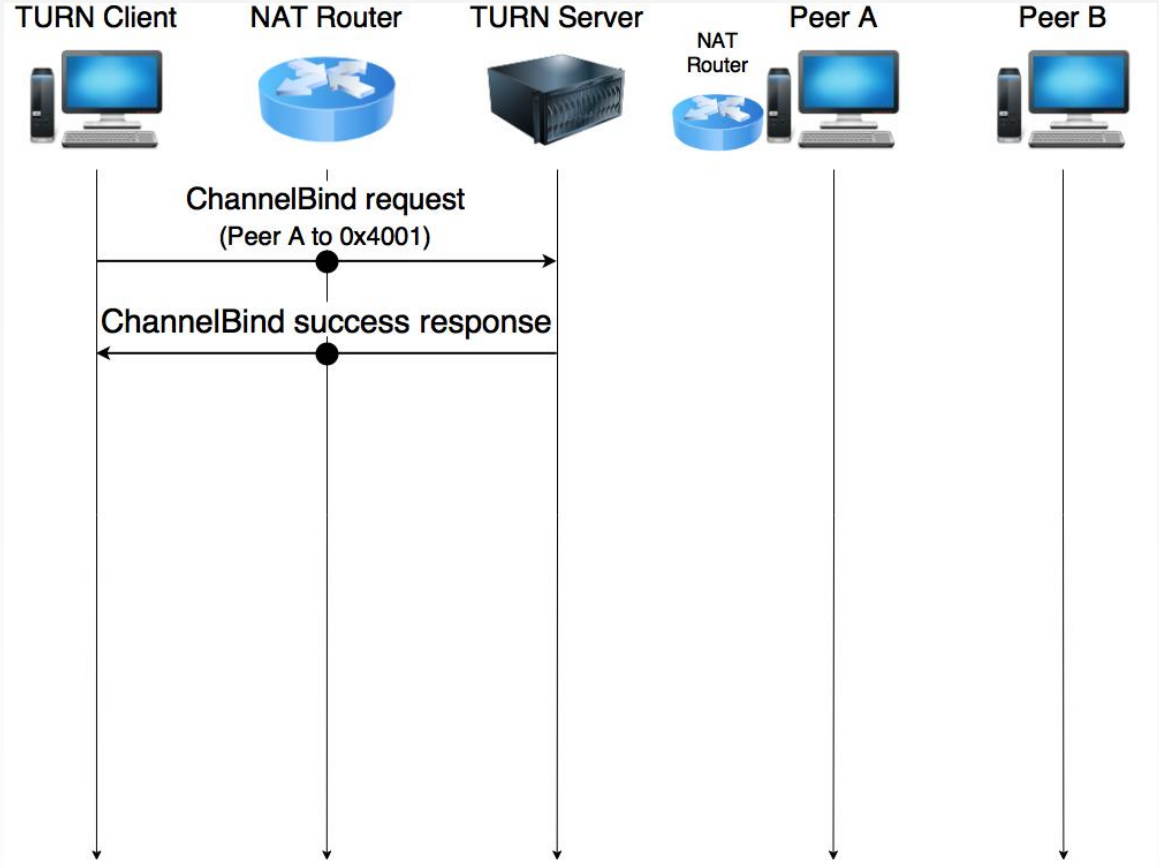


- Send indication
  - Serverは，Relayed Transport Address（前出のDとd2）を送信元として，受理したアプリケーションデータをPeer Aに送信する（UDP通信）.
- Data indication
  - Peer Aから受信したデータをClientに送信する.

TURN 動作概要 (6)

- Channel（チャネル）

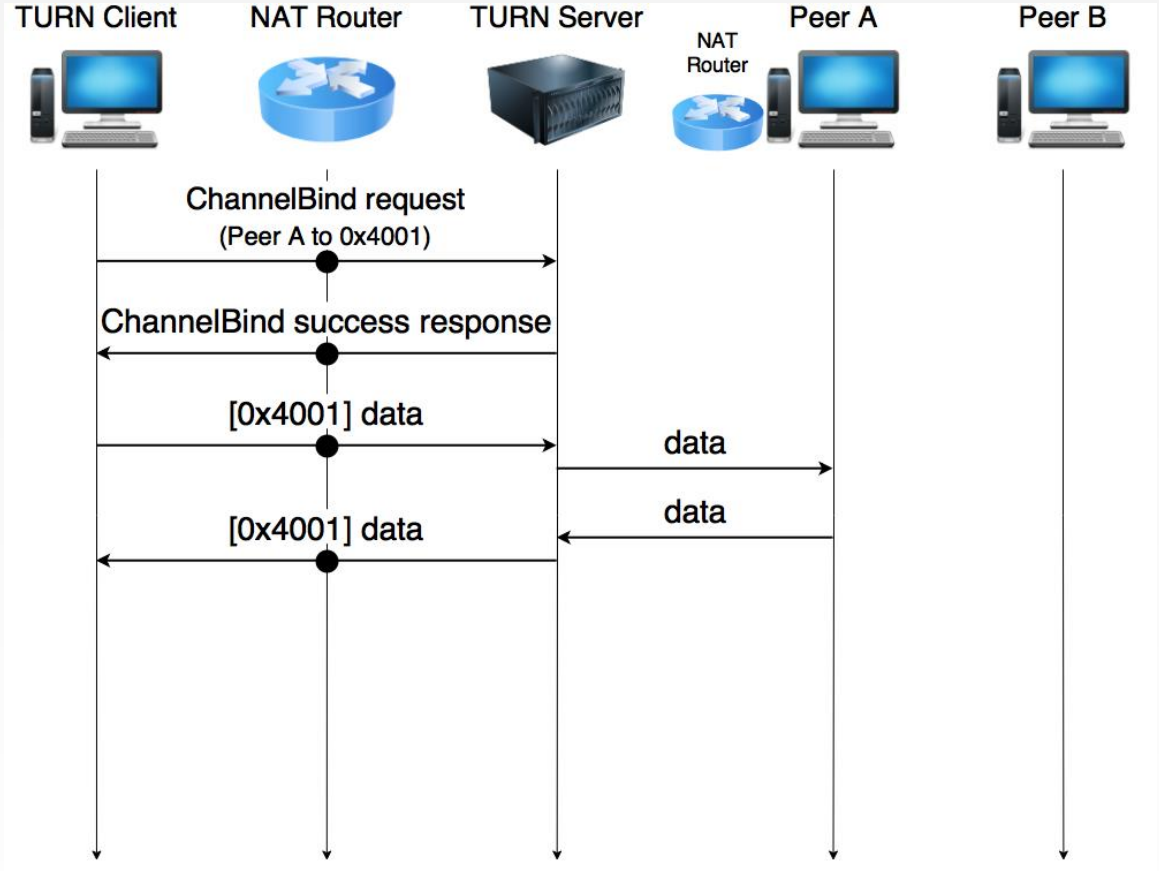
  ChannelData message（STUNメッセージとは異なるフォーマット）を用いる.



- ChannelBind request
  - ここでは，Channel番号0x4001とPeer AのServer-Reflexive Transport Address（前出のEとe）を指定する.
  - 割り当てとは異なり，ChannelBindingを明示的に終了する方法は存在しないため，タイムアウトを待つ.

TURN 動作概要 (7)

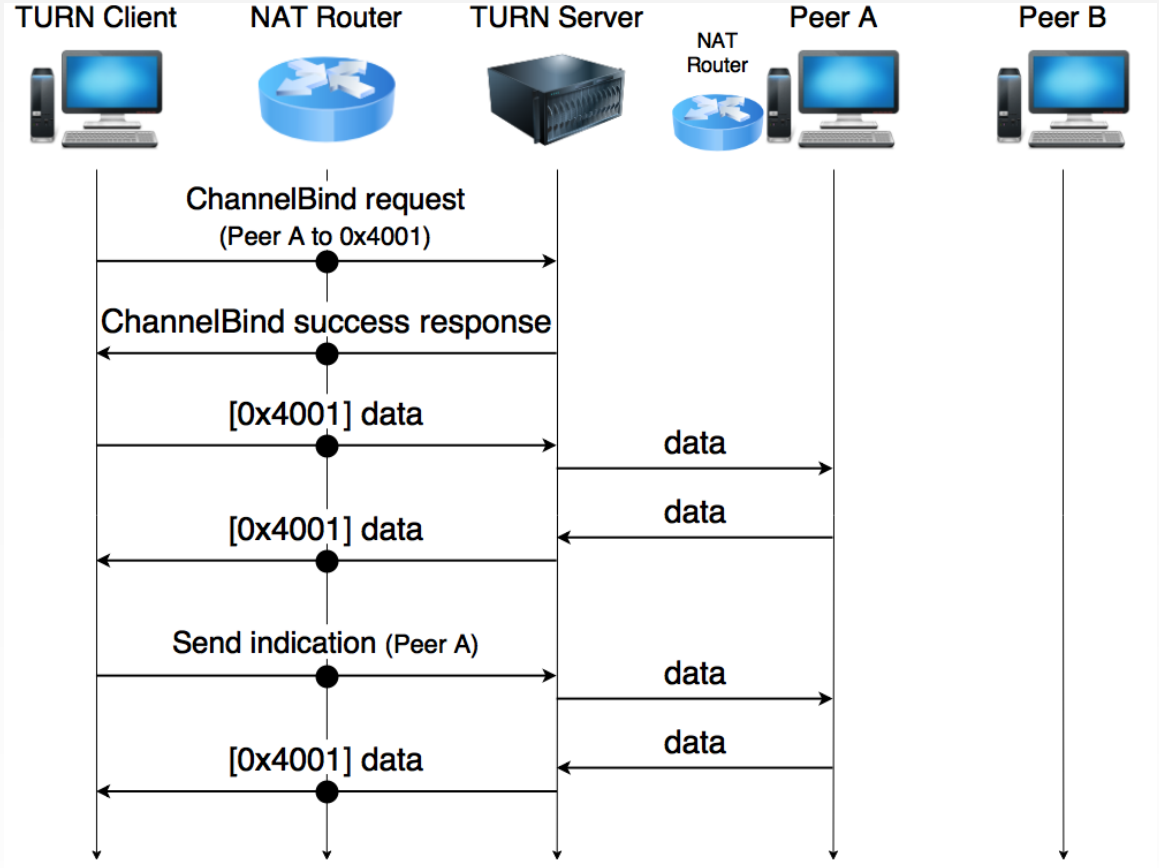
- Channel（チャネル）



- [0x4001] data
  - ChannelData messageに，アプリケーションデータをカプセル化したもの．
  - ServerがClientから受信すると，Relayed Transport Address（前出のDとd2）を送信元として，dataをPeer Aに送信する（UDP通信）．
  - ServerはPeerから受信したdataをClientに送信するとき，ChannelData messageにカプセル化する．

TURN 動作概要 (8)

- Channel（チャネル）



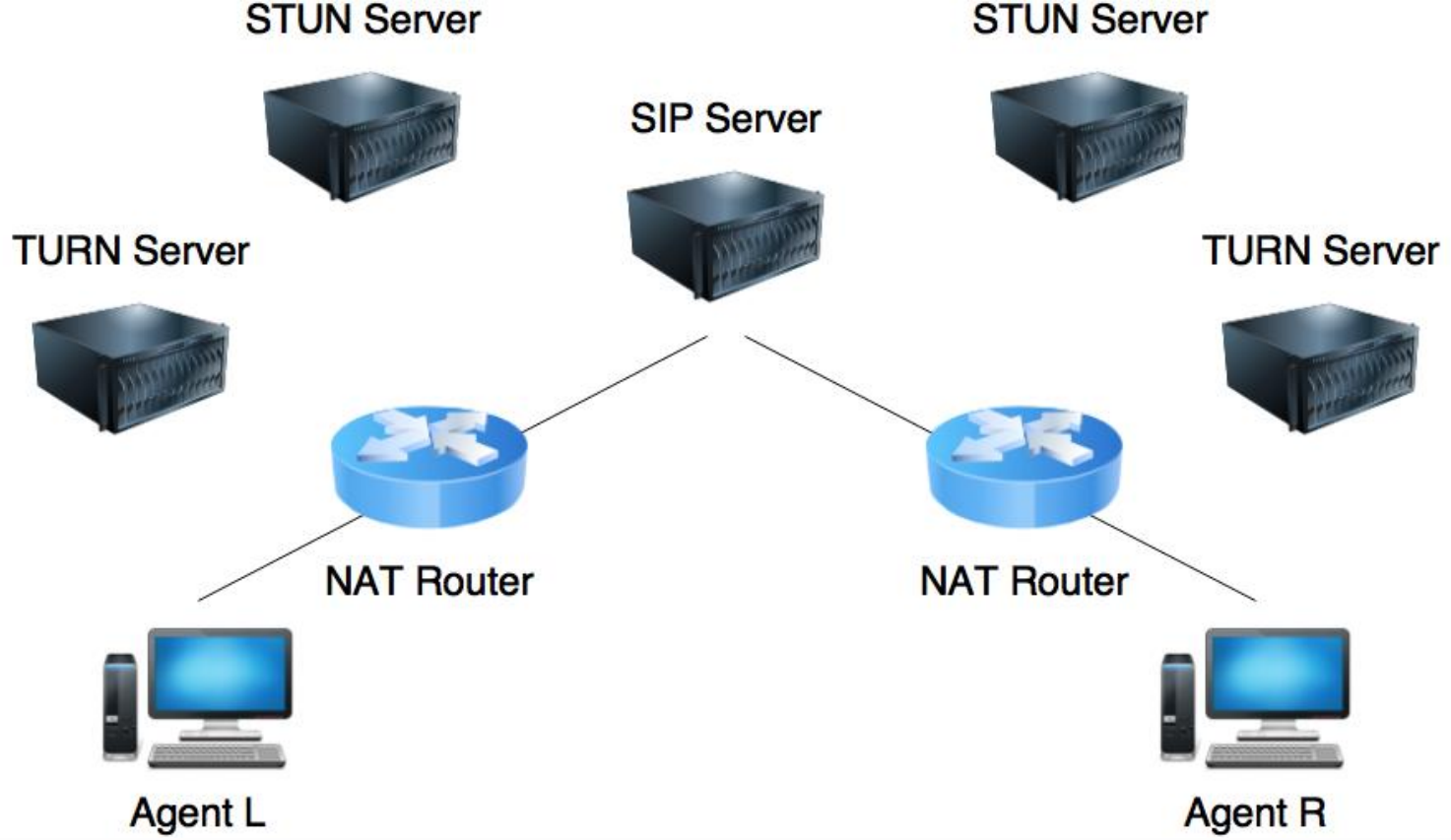
- Send indication
  - Send Mechanismで使用できるものと同様のもの.
  - Client側では，Channelを BindingしたPeerに対して，ChannelData messageと Send indicationの両方を用いることが可能.
  - Server側では，ChannelData messageと Send indicationのどちらを受け取っても，ChannelData messageを使用する.

ICE 動作概要

ネットワーク構成

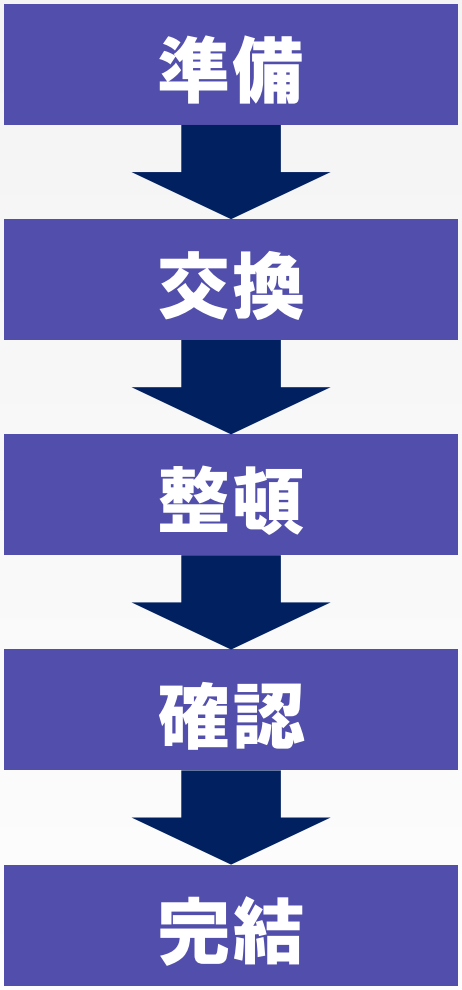
以下のようなネットワーク構成を仮定する.

- 参考：Agent LおよびRが用いるSTUN ServerおよびTURN Serverは，同じものでも違うものでもよい.



ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

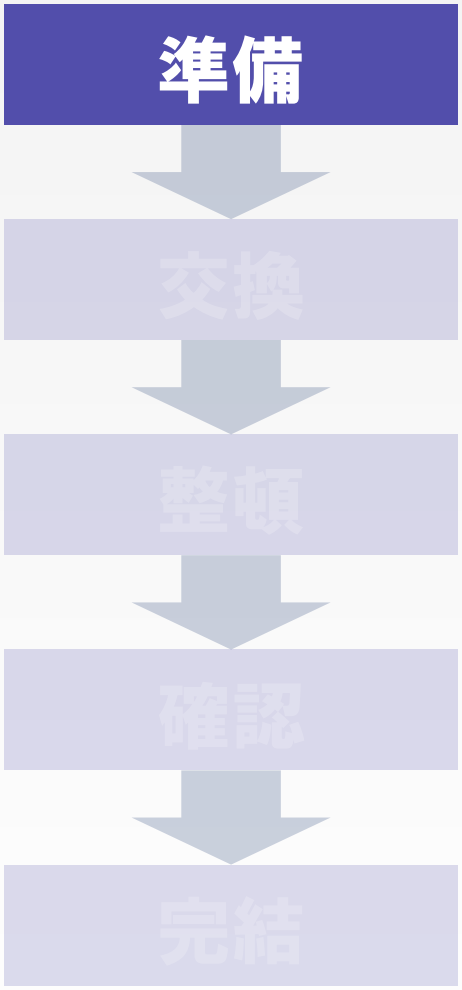
互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

どのペアを使用するのか決定

ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

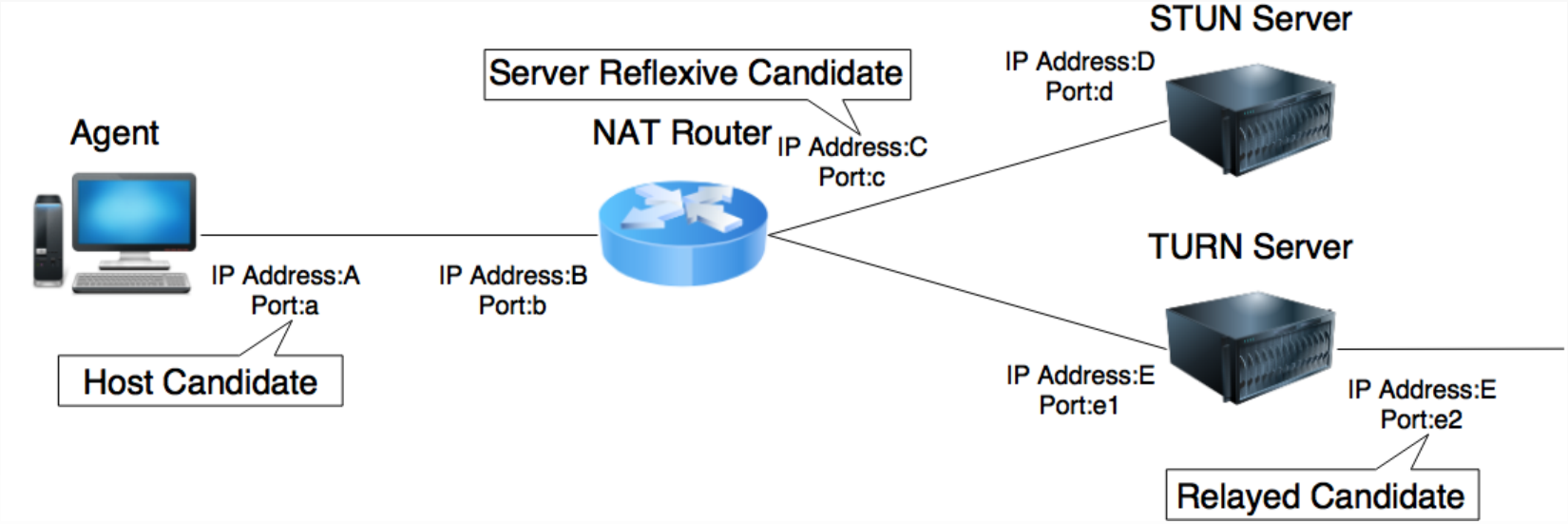
互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

どのペアを使用するのか決定

【準備】候補（Candidate）の収集

- エージェントのローカルアドレスを取得
  - この候補をHost Candidateという.
- NATのグローバルIPアドレスとポート番号を，STUNにより取得
  - この候補をServer Reflexive Candidateという.
- TURNサーバの中継用グローバルIPアドレスとポート番号を取得
  - この候補をRelayed Candidateという.



【準備】候補の優先順位付け

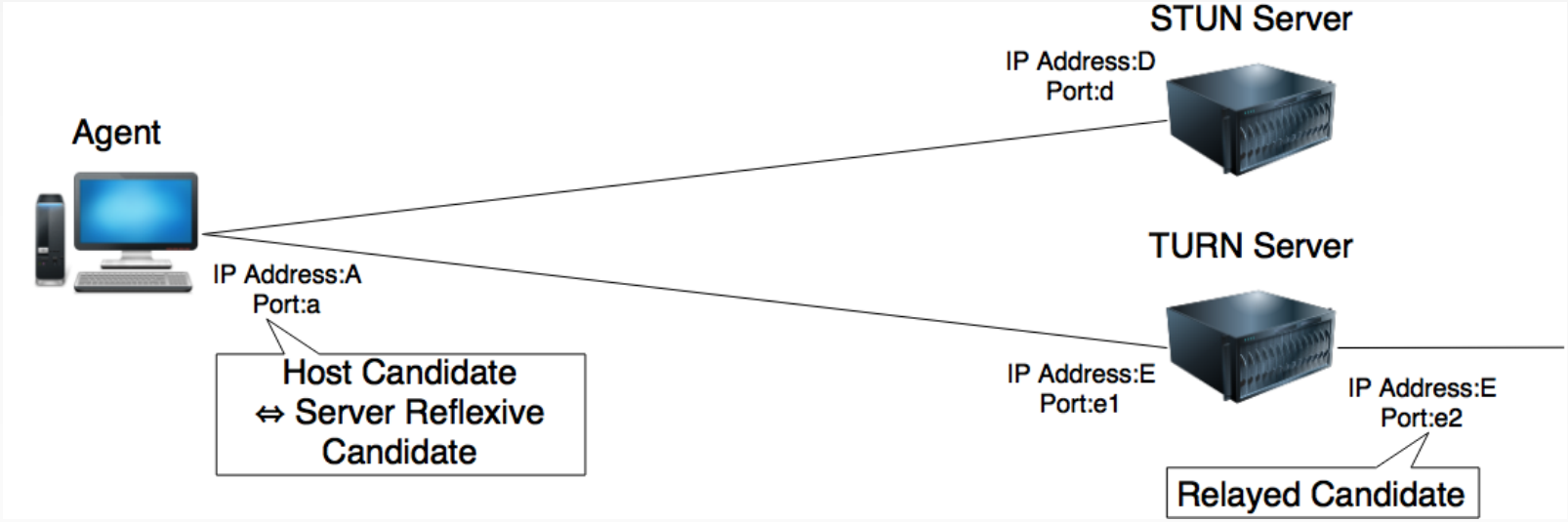
- Priorityの計算
  - Priority：接続チェック順，候補の優先順位
  - 以下の数式で計算される．

$$\begin{aligned}\text{priority} = {} & 2^{24} \times (\text{type preference}) + \\ & 2^{8} \times (\text{local preference}) + \\ & 2^{0} \times (256 - \text{component ID})\end{aligned}$$

  - **type preference**：Candidateのtype（HOST，SERVER REFLEXIVE，RELAYEDのいずれか）の優先度
    - 推奨値：HOST…126（最高値），SERVER REFLEXIVE…100，RELAYED…0（最低値）
  - **local preference**：Agentがマルチホームである場合（Server Reflexive Candidateが複数存在する場合など）の，CandidateのIPアドレスの優先度
    - 最低値…0，最高値…65535
  - **Component ID**：1から256までの間の値
    - RTP(Realtime Transport Protocol)に基づくメディアストリームの場合，RTPは1，RTCP(Realtime Transport Control Protocol)は2である．

【準備】冗長な候補の削除

- ICEでは，AgentはNAT配下にあるか否かを考慮していない．
- AgentがNAT配下にない（すなわち，グローバル環境内にある）場合，Host CandidateとServer Reflexive Candidateが一致する．



- これらを両方とも保持しておくのは無駄であるため，Priorityの値が低いものを削除する．

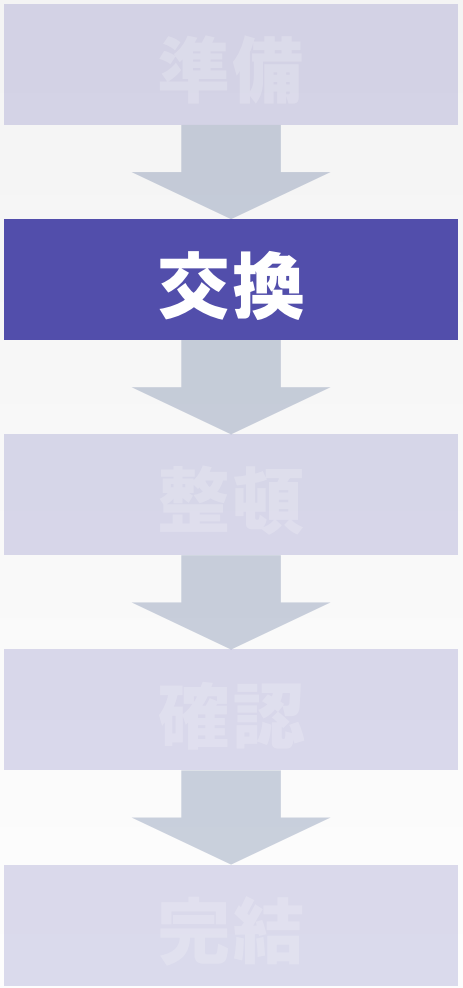
【準備】Default Candidateの選択

- Default Candidate（デフォルトの候補）
  - ICEを使用しないPeerとの通信において用いられる候補．そのPeerからのターゲットとなることから，Default Destinationともいう．
  - 通信に用いられる候補の可能性に基づいて選択することが推奨される．
    - Relayed Candidateが利用可能な場合…Relayed Candidate
    - Relayed Candidateが利用不可能な場合
      - Server Reflexive Candidateが利用可能…Server Reflexive Candidate
      - Server Reflexive Candidateが利用不可能…Host Candidate

- ICEにおいて，最終的に選択された候補がDefault Candidateと異なる場合，これらが一致するようにICEの処理を再度行う必要がある．

ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

どのペアを使用するのか決定

【交換】候補情報の交換（1）

- 候補リスト（全候補を含むリスト）の情報が揃ったら，これを相手に送信する．
- 送信するメッセージの記述には，SDPというプロトコルを用いる．
  - SDP(Session Description Protocol)：セッションを開始する際，メディアの詳細情報やトランスポートアドレスなどのセッション記述メタデータを，参加者に伝達するために用いるプロトコル．RFC4566で標準化．
- 相手がこの情報を受け取ると，自らの役割を決定した上で，送信側と同様の処理を行い，候補リストを返信する．
  - 役割…2つのエージェントを，Controlling Agent（候補ペアの最終選択を行うエージェント）と，Controlled Agent（候補ペアの最終選択を待つエージェント）とする．
- 相手からの返信を受け取ると，両者が，自分と相手の候補リストを保持していることになる．

【交換】候補情報の交換 (2)

以下では，Agent LをControlling Agent，Agent RをControlled Agentとする．

- Agent LおよびAgent Rの候補リスト（一部）を，以下のものとする．

Agent Lの候補リスト

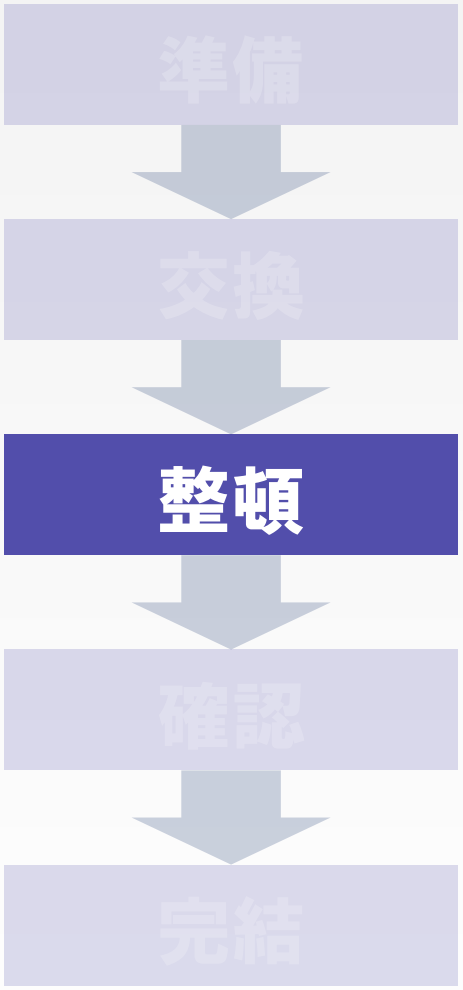
IP Address	Port	Protocol	Type
A_L	a_L	UDP	HOST
C_L	c_L	UDP	SERVER REFLEXIVE
E_L	$e2_L$	UDP	RELAYED

Agent Rの候補リスト

IP Address	Port	Protocol	Type
A_R	a_R	UDP	HOST
C_R	c_R	UDP	SERVER REFLEXIVE
E_R	$e2_R$	UDP	RELAYED

ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

どのペアを使用するのか決定

【整頓】Candidate Pairの作成

- Agent LおよびAgent Rは，自身の候補と相手の候補のそれぞれをペアにする．このペアをCandidate Pairという．

IP Address	Port	Protocol	Type	IP Address	Port	Protocol	Type
A_L	a_L	UDP	HOST	A_R	a_R	UDP	HOST
A_L	a_L	UDP	HOST	C_R	c_R	UDP	SERVER
A_L	a_L	UDP	HOST	E_R	$e2_R$	UDP	RELAYED
C_L	c_L	UDP	SERVER	A_R	a_R	UDP	HOST
C_L	c_L	UDP	SERVER	C_R	c_R	UDP	SERVER
C_L	c_L	UDP	SERVER	E_R	$e2_R$	UDP	RELAYED
E_L	$e2_L$	UDP	RELAYED	A_R	a_R	UDP	HOST
E_L	$e2_L$	UDP	RELAYED	C_R	c_R	UDP	SERVER
E_L	$e2_L$	UDP	RELAYED	E_R	$e2_R$	UDP	RELAYED

※TypeのSERVERは，SERVER REFLEXIVEの略．

【整頓】Candidate Pairの優先順位付け

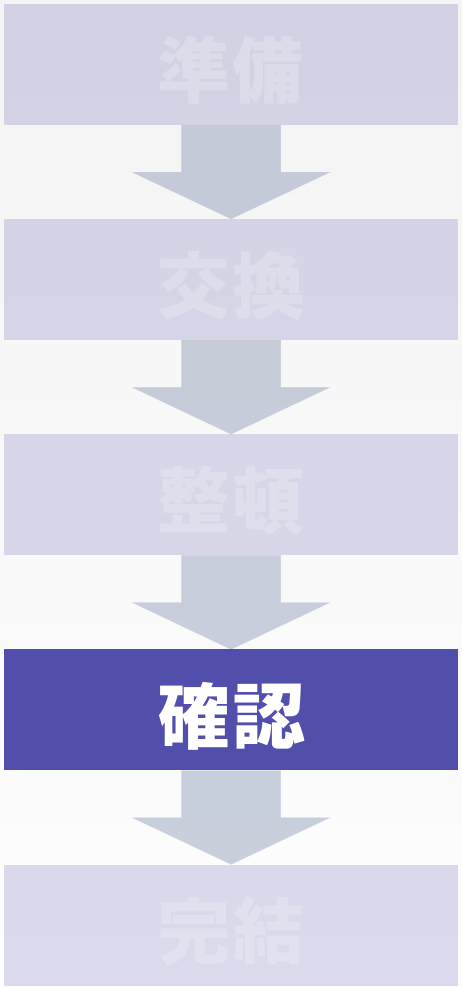
- Pair Priorityの計算
  - Pair Priority：候補ペアの接続チェック順，候補ペアの優先順位
  - Agent Lの候補のPriorityをG，Agent Rの候補のPriorityをDとすると，以下の数式で計算される．

$$\begin{aligned}\text{pair priority} = {} & 2^{32} \times \mathrm{MIN}(G, D) + \\ & 2 \times \mathrm{MAX}(G, D) + \\ & (G > D \ ? 1 : 0)\end{aligned}$$

  - ここで，$(G > D \ ? 1 : 0)$は，$G > D$のとき1，$G \leq D$のとき0となる式．
    - CやJavaなどの三項演算子と同義
- Candidate Pairを，Pair Priorityの降順に並べる．
  - Pair Priorityの優先度が高いものから接続可能性の確認を行う．
  - Pair Priorityの値が同値となるペアが複数存在する場合，それらの順序は任意である．

ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

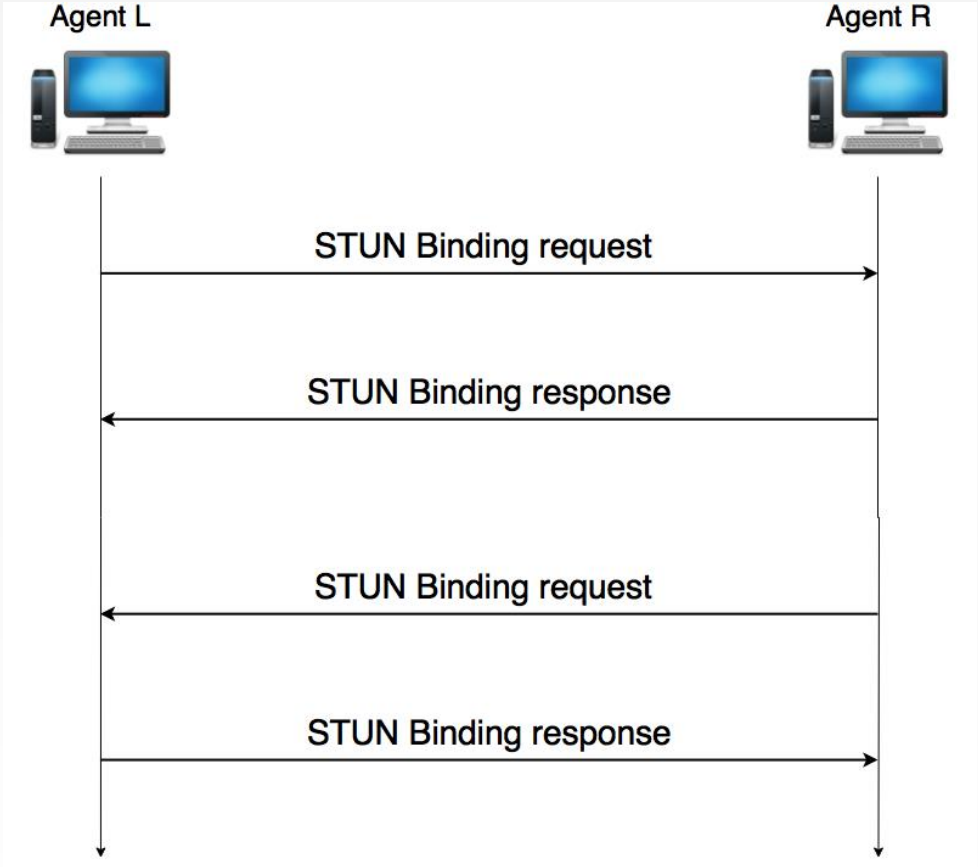
どのペアを使用するのか決定

【確認】接続可能性の確認 (1)

- Relayed Candidatesのアクセス許可を行う.
  - TURNの範疇であるため, CreatePermission request/response(前述)を使用する.
- 各候補間の接続性を, STUN Binding request/responseにより確認する.
  - 一方のエージェントがSTUN Clientとして, 他方のエージェントがSTUN Serverとして機能する.
- STUN Binding responseが以下の全ての条件を満たす場合, 接続可能と判定する.
  - success response(成功応答)である.
  - 送信元IPアドレスとポート番号が, requestの宛先IPアドレスとポート番号に一致する.
  - 宛先IPアドレスとポート番号が, requestの送信元IPアドレスとポート番号に一致する.

【確認】接続可能性の確認 (2)

- 1組のCandidate Pairについての接続可能性の確認は，以下のように行う．これを全てのCandidate Pairに対して行う．



- この図では，NATやSIP Serverなどは省略している．

- 最初のrequest/responseにより，Agent L側からの確認を行う．
  - この間，Agent LがSTUN Clientとして，Agent RがSTUN Serverとして動作する．
- 次のrequest/responseにより，Agent R側からの確認を行う．
  - この間，Agent RがSTUN Clientとして，Agent LがSTUN Serverとして動作する．

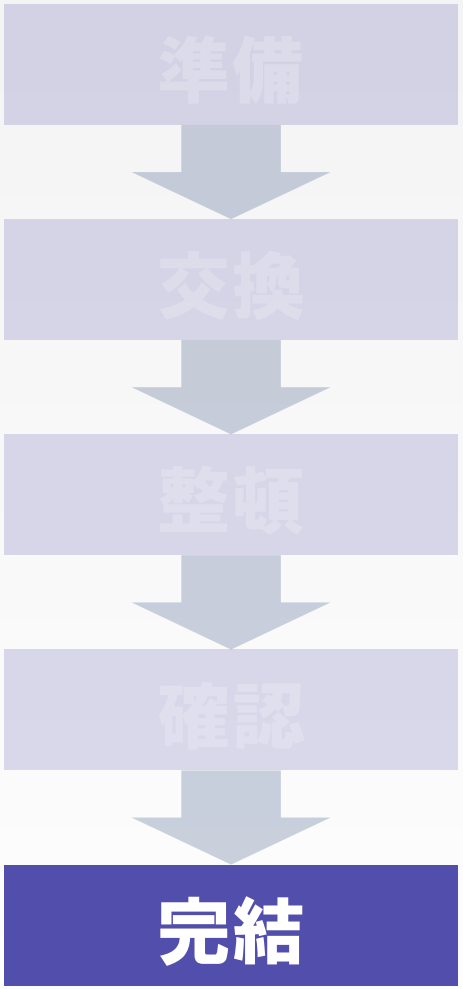
【確認】未知の候補の出現

- STUN Client側
  - STUN Binding responseには，サーバ側から見た自身のアドレスが写像されている（前述）
  - 写像されたアドレスが，既知の自身の候補と一致しない場合がある．
    - これをPeer Reflexive Candidateとして，自分側の候補リストに追加する．
    - この候補は，相手の候補との間でペアになっていない．ただし，瞬時にペアを作成し，これを含めたリストを共有することが可能である（必須ではない）．

- STUN Server側
  - STUN Binding requestの送信元IPアドレスとポート番号が，既知の相手の候補と一致しない場合がある．
    - これをPeer Reflexive Candidateとして，相手側の候補リストに追加する．
    - この候補は，自身の候補との間でペアになっていないが，ペアにする必要はない．

ICEの動作の流れ

- 基本的には以下のように進む.



ICEを使用するための候補を準備

自身と相手の候補リストを交換

互いの候補をペアにして整頓

全てのペアについて接続可能性を確認

どのペアを使用するのか決定

【完結】使用するペアの決定

- 確認の結果，接続可能なペアがいくつか判明する．
- Controlling Agentが，以後の通信において使用するペアを決定し，Controlled Agentに通知する．
  - 使用するペアの決定方法には，Regular NominationとAggressive Nominationの2種類がある．
  - **Regular Nomination**：使用するペアを決定した後，Flag付きSTUN Binding requestにより通知する．
    - 使用するペアは，接続可能なペアのうちのいずれかとなる．
  - **Aggressive Nomination**：確認段階でFlag付きSTUN Binding requestを送信し，接続可能なものが発見された場合は確認を終了し，そのペアを使用する．
    - 使用するペアは，接続可能なペアのうち，Pair Priorityが最高のものとなる．
- 使用するペアをSelected Pairと呼ぶ．

まとめ

- ICEとは，RFC5245で標準化されているUDPベースのNAT越えプロトコルであり，STUNとTURNを使用する．
  - STUNとは，RFC5389で標準化されているクライアントサーバプロトコルであり，NAT越え問題を解決するための一部として利用される．
  - TURNとは，STUNの拡張版であり，RFC5766で標準化されているクライアントサーバプロトコルである．サーバ経由でのリレー通信を行う．
- ICEでは，各エージェントが通信に用いるアドレスの候補を用意し，どのアドレスにより通信を行うかを決定する．
- ICEの基本的な動作の流れは，準備，交換，整頓，確認，完結という5つのステップで説明できる．

参考文献 (英文)

- RFC 5245 “Interactive Connectivity Establishment (ICE) : A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols”
  - J. Rosenberg, https://tools.ietf.org/html/rfc5245, 2017/04/06参照
- RFC 5389 “Session Traversal Utilities for NAT (STUN)”
  - J. Rosenberg他, https://tools.ietf.org/html/rfc5389, 2017/02/25参照
- RFC 5766 “Traversal Using Relays around NAT (TURN) : Relay Extensions to Session Traversal Utilities for NAT (STUN)”
  - R. Mahy他, https://tools.ietf.org/html/rfc5766, 2017/03/04参照

参考文献 (和文)

- WebRTCのICEについて知る
  - 岩瀬 義昌, http://www.slideshare.net/iwashi86/webrtcice, 2017/04/06参照
- 壁を越えろ！WebRTCでNAT/Firewallを越えて通信しよう
  - がねこまさし, https://html5experts.jp/mganeko/5554/, 2017/03/04参照

付録

【STUN】アドレス写像方法

- 写像には，MAPPED-ADDRESS属性，またはXOR-MAPPED-ADDRESS属性のいずれかを使用する．
  - MAPPED-ADDRESS属性：NATのグローバルIPアドレスとポート番号をそのままコピーする方法．
  - XOR-MAPPED-ADDRESS属性：NATのグローバルIPアドレスとポート番号をXOR関数により難読化する方法．RFC5389で追加された属性．
- RFC5389では，基本的に後者を使用する．
- 写像については，パケットフォーマットの解説で述べる．

【STUN】パケットフォーマット (1)

- STUNメッセージヘッダのフォーマット

0 0	STUN Message Type （ 14bit ）	Message Length （ 16bit ）
Magic Cookie		
Transaction ID （ 96bit ）		

  - 先頭2bitは，同じポート番号を用いる他のプロトコルとの区別に使用.
  - STUN Message Typeは以下のように定義.

M_{11}	M_{10}	M_9	M_8	M_7	C_1	M_6	M_5	M_4	C_0	M_3	M_2	M_1	M_0

    - メッセージメソッドは12bit（$M_{11}M_{10}$ … M_1M_0）で表現.
      - 0000 0000 0001：Binding
    - メッセージクラスは2bit（C_1C_0）で表現.
      - 00：request, 01：indication, 10：success response, 11：error response
    - STUN Message Typeはこれらの組み合わせにより表現する.
      - 例：Binding request…Bindingは「0000 0000 0001」, requestは「00」より, 00000000000001
      - 例：Binding response…Bindingは同上, (success) responseは「10」より, 00000100000001

【STUN】パケットフォーマット (2)

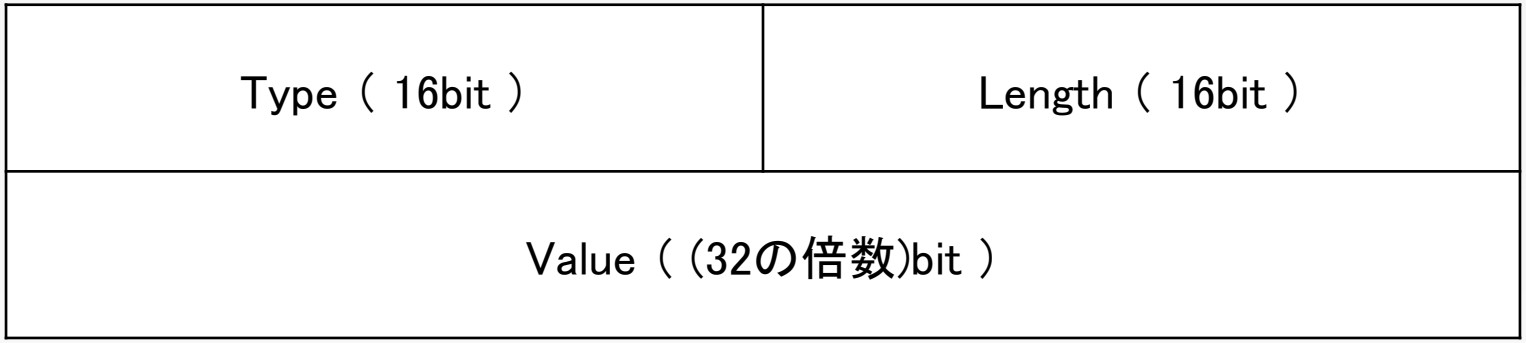
- STUNメッセージヘッダのフォーマット

00	STUN Message Type（ 14bit ）	Message Length（ 16bit ）
Magic Cookie		
Transaction ID（ 96bit ）		

  - Magic Cookieは0x2112A442（固定値）
    - すなわち，0010 0001 0001 0010 1010 0100 0100 0010
    - RFC5389で追加された属性（XOR-MAPPED-ADDRESS属性など）がクライアントに通じるかを，サーバが検出する際に使用．
    - 同じポート番号を用いる他のプロトコルとの区別が可能．
  - Transaction IDはSTUNトランザクションを一意に識別するために使用．
    - 送信側が選択し，受信側は返信時にそのまま返す．
    - 0から2^{96-1}（10進数）までのうち，任意の値をランダムに選択．
    - STUN ClientとSTUN Serverが同じポート番号で通信する際には，送信側のrequestのTransaction IDと受信側のrequestのTransaction IDは無関係．

【STUN】パケットフォーマット (3)

- STUNメッセージヘッダ（前述）の後には，0個以上の属性が続く.
- STUN属性のフォーマット



  - Typeの例
    - 0x0001（0000 0000 0000 0001）：MAPPED-ADDRESS属性
    - 0x0020（0000 0000 0010 0000）：XOR-MAPPED-ADDRESS属性
  - Lengthは，パディング前のValueの長さ（単位：Byte）
  - Valueは，各属性固有のデータ（次スライド参照）
    - 長さが(32の倍数)bitでない場合は，任意の値でパディングする.

【STUN】パケットフォーマット (4)

- STUN属性フォーマットのValueの内容例
  - MAPPED-ADDRESS属性

0 0 0 0 0 0 0 0	Family (8bit)	Port (16bit)
Address (32bit or 128bit)		

  - XOR-MAPPED-ADDRESS属性

x x x x x x x x	Family (8bit)	X-Port (16bit)
X-Address (32bit or 128bit)		

    - いずれも，NATのグローバルIPアドレスとポート番号を写像する際に使用される.
    - Familyは以下の値となる.
      - 0x01(0000 0001) : IPv4
      - 0x02(0000 0010) : IPv6

【TURN】パケットフォーマット（1）

- ほとんどのTURNメッセージはSTUNメッセージと同一形式
- STUNメッセージヘッダのフォーマット（再掲）

00	STUN Message Type （ 14bit ）	Message Length （ 16bit ）
Magic Cookie		
Transaction ID （ 96bit ）		

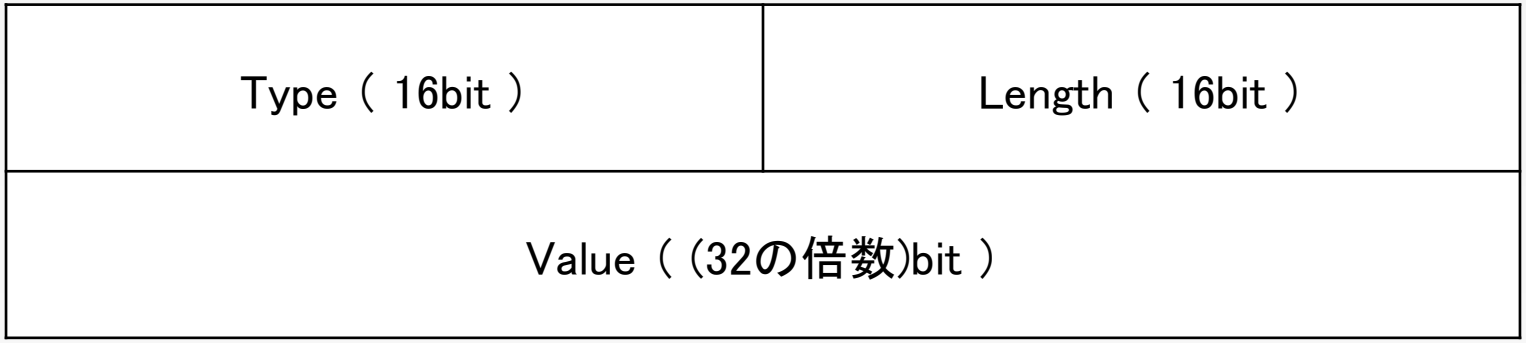
  - STUN Message Type（再掲）

M_{11}	M_{10}	M_9	M_8	M_7	C_1	M_6	M_5	M_4	C_0	M_3	M_2	M_1	M_0

    - TURNで新たに定義されたメッセージメソッド（$M_{11}M_{10}\ldots M_1M_0$）
      - 0000 0000 0011：Allocate（メッセージクラスはrequest, responseのみ）
      - 0000 0000 0100：Refresh（メッセージクラスはrequest, responseのみ）
      - 0000 0000 0110：Send（メッセージクラスはindicationのみ）
      - 0000 0000 0111：Data（メッセージクラスはindicationのみ）
      - 0000 0000 1000：CreatePermission（メッセージクラスはrequest, responseのみ）
      - 0000 0000 1001：ChannelBind（メッセージクラスはrequest, responseのみ）

【TURN】パケットフォーマット (2)

- STUN属性のフォーマット（再掲）



- TURNで新たに定義されたTypeの例
  - 0x000C（0000 0000 0000 1100）：CHANNEL-NUMBER属性
  - 0x000D（0000 0000 0000 1101）：LIFETIME属性
  - 0x0012（0000 0000 0001 0010）：XOR-PEER-ADDRESS属性
  - 0x0013（0000 0000 0001 0011）：DATA属性
  - 0x0016（0000 0000 0001 0110）：XOR-RELAYED-ADDRESS属性

【TURN】パケットフォーマット (3)

- STUN属性フォーマットのValueの内容例
  - CHANNEL-NUMBER属性

Channel Number （ 16bit ）	REFU （ 16bit ）

    - Channel Numberはチャネルの番号（符号なし整数）.
    - REFU(Reserved For Future Use)は0x0000（固定値）で送信し，受信側は無視.
    - ChannelのBindingを行う際にはこれを用いるが，データをやり取りする場合はChannelData message（後述）を用いる.

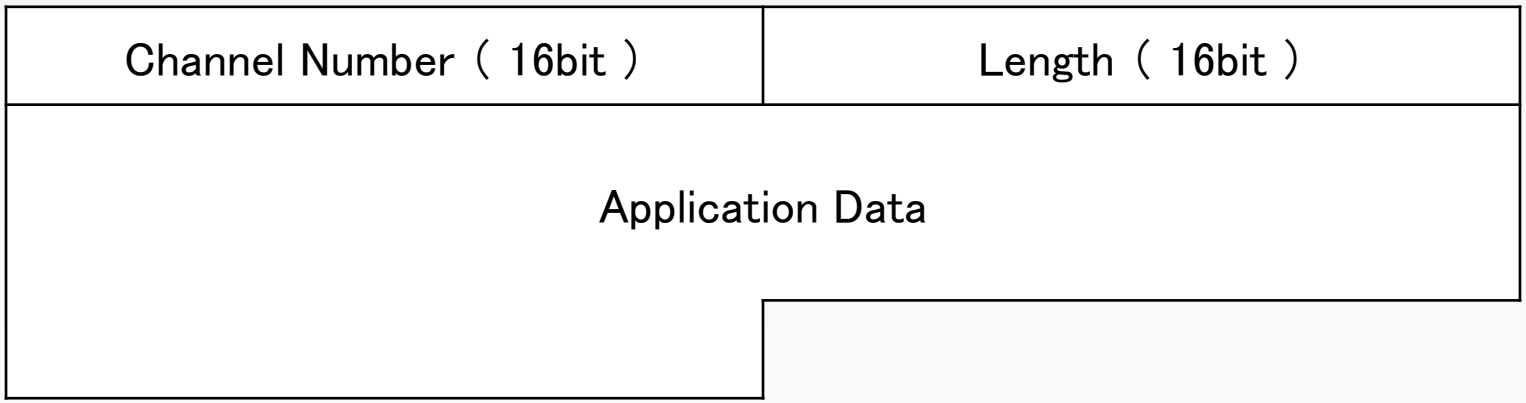
  - LIFETIME属性
    - サーバがAllocation（割り当て）を維持する時間（初期設定）.
    - Value部は割り当てを維持する時間（単位は秒，32bit符号なし整数）.

【TURN】パケットフォーマット (4)

- STUN属性フォーマットのValueの内容例
  - XOR-PEER-ADDRESS属性
    - PeerのIPアドレスとポート番号（PeerがNAT配下にある場合は，Server-Reflexive Transport Address）を指定.
    - 使用方法はXOR-MAPPED-ADDRESS属性（前述）と同様.
  - DATA属性
    - Send indicationおよびData indicationにおいて必ず存在する.
    - アプリケーションデータで構成されるため可変長となる．データの長さが(32の倍数)bitでない場合は，任意の値でパディングする.
  - XOR-RELAYED-ADDRESS属性
    - Relayed Transport Address（サーバのIPアドレス，およびPeerに中継するポート番号）を指定.
    - 使用方法はXOR-MAPPED-ADDRESS属性（前述）と同様.

【TURN】パケットフォーマット (5)

- ChannelData message
  - TURNで用いるメッセージのうち，STUNメッセージヘッダを用いない例.
  - ChannelのBindingが完了したPeerとデータをやり取りする場合には，Channel Numberで宛先を特定できるためChannelData message以外は不要.
  - フォーマットは以下の通り.



  - Channel Numberは，データをやり取りするチャネルの番号.
  - Lengthは，Application Dataの長さ（単位：Byte）
  - Application Dataは，アプリケーションデータの本体.

【ICE】パケットフォーマット（1）

- ICEにおいてもSTUNメッセージを使用する
- STUN属性のフォーマット（再掲）

Type（ 16bit ）	Length（ 16bit ）
Value（ (32の倍数)bit ）	

  - ICEで新たに定義されたTypeの例
    - 0x0024（0000 0000 0010 0100）：PRIORITY属性
    - 0x0025（0000 0000 0010 0101）：USE-CANDIDATE属性
    - 0x8029（1000 0000 0010 1001）：ICE-CONTROLLED属性
    - 0x802A（1000 0000 0010 1010）：ICE-CONTROLLING属性

【ICE】パケットフォーマット (2)

- STUN属性フォーマットのValueの内容例
  - PRIORITY属性
    - Peer Reflexive Candidate（32頁参照）の優先度を示すもの.
  - USE-CANDIDATE属性
    - ICEにより決定したCandidate Pair（Selected Pair）を，以降の通信において使用することを示すフラグメント.
    - Lengthフィールドは0となる.
  - ICE-CONTROLLED属性
    - 自身がControlled Agentであることを示すもの.
  - ICE-CONTROLLING属性
    - 自身がControlling Agentであることを示すもの.

【ICE】Candidateのプロパティー覧（1）

- Transport Address
  - IPアドレス，ポート番号，トランスポートプロトコル
- Type
  - HOST，SERVER REFLEXIVE，RELAYED
- Component
  - 単一のトランスポートアドレスを必要とするメディアストリームの一部．
    RTPに基づくメディアストリームの場合，RTP用（Component ID：1）とRTCP用（Component ID：2）がある．
    - **RTP**(Realtime Transport Protocol)：音声や動画などのデータストリームをリアルタイムに配送するためのデータ通信プロトコル
    - **RTCP**(Realtime Transport Control Protocol)：RTPに制御機能を備えたプロトコル
- Priority
  - 接続チェック順，候補の優先順位

【ICE】Candidateのプロパティ一覧 (2)

- Foundation：任意の値を持つ識別子．ある2つの候補について以下の全てが成立する場合，Foundationは同値となる．
  - Typeが同じである．
  - BASEのIPアドレスが同じである．
  - トランスポートプロトコルが同じである．
  - 接続しているSTUNサーバまたはTURNサーバが同じIPアドレスを持つ．
- Base
  - Server Reflexive CandidateのBase・・・それに対応するエージェント（Host Candidate）
  - Host CandidateのBase・・Host Candidate自身
  - Relayed CandidateのBase・・Relayed Candidate自身
  - 接続可能性の確認の際，Server Reflexive Candidateからは要求を送信できないため，Server Reflexive CandidateをBaseに置換する．
    - すなわち，Host Candidateから送信したものとみなす．

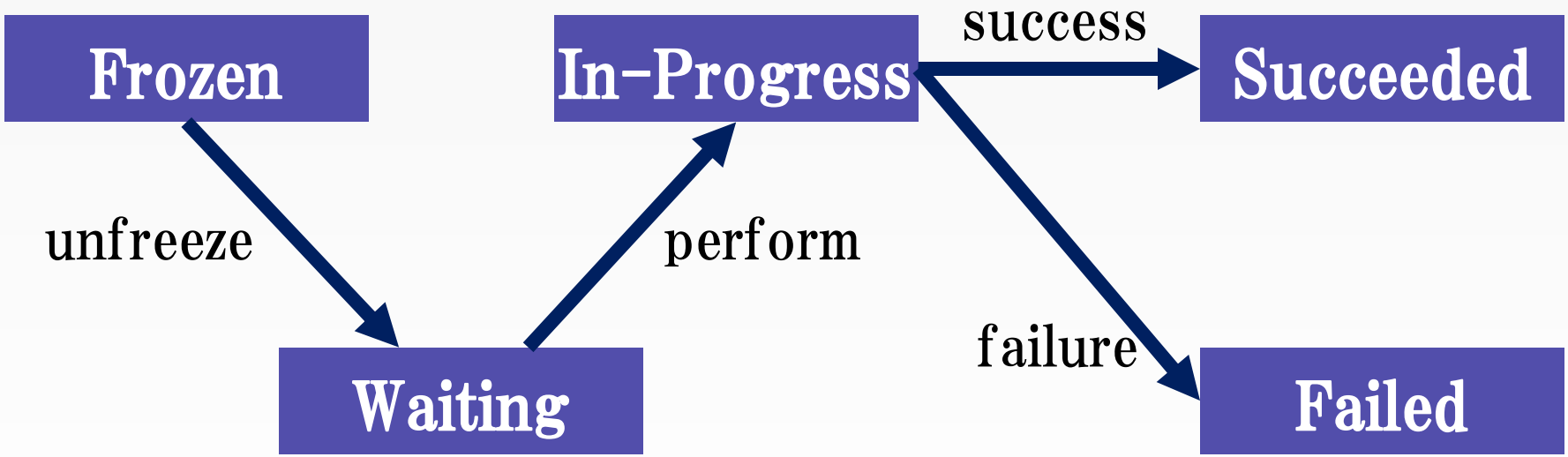
【ICE】チェックリスト

- **チェックリスト**：両者のCandidateおよびそのペアのFoundation，Stateをまとめたもの．
  - Foundation：Local Candidates，Remote Candidatesのそれぞれの Foundationを組み合わせたもの．
  - Stateについては次頁参照
- 自身の候補，相手の候補をそれぞれLocal Candidates，Remote Candidatesとすると，チェックリスト（部分抜粋）は以下のようになる．
  - それぞれの候補の詳細については27頁参照

Local Candidate				Remote Candidate				Foundation	State
A_L	a_L	UDP	HOST	A_R	a_R	UDP	HOST	…	…
A_L	a_L	UDP	HOST	C_R	c_R	UDP	SERVER	…	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
E_L	$e2_L$	UDP	RELAYED	E_R	$e2_R$	UDP	RELAYED	…	…

【ICE】ステートマシン (1)

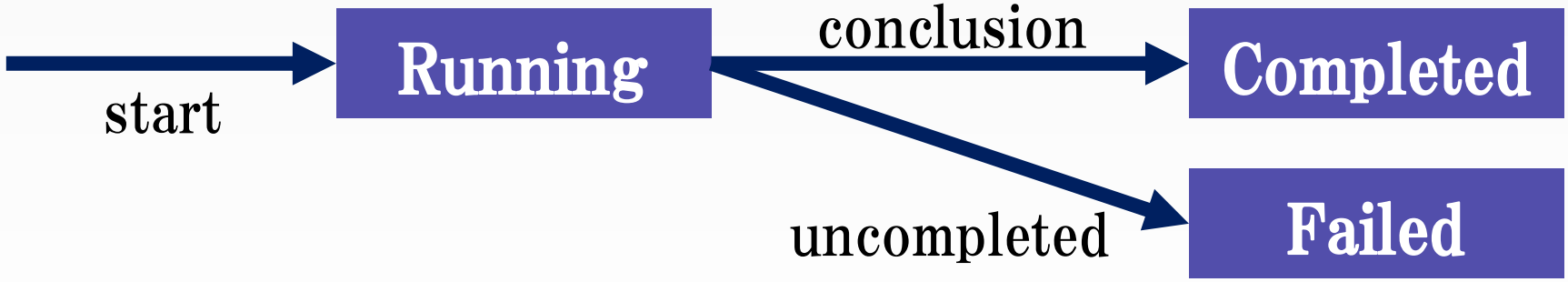
- State：各ペアのstateには，以下の5つがある．
  - Frozen：チェックが実行されていない初期状態．
  - Waiting：チェックは実行されていないが，チェックリスト上で最もPair Priorityの高いWaitingペアになるとすぐに実行できる状態．
  - In-Progress：チェックは送信されたが，トランザクションが進行中である状態．
  - Succeeded：チェックが完了し，成功したことを示す状態．
  - Failed：チェックが完了し，失敗したことを示す状態．



【ICE】ステートマシン (2)

- **アクティブチェックリスト**：チェックリスト内のペアのうち，少なくとも1つのペアのStateがWaitingであるチェックリスト.
- **フリーズチェックリスト**：チェックリスト内の全てのペアのStateがFrozenであるチェックリスト.

- チェックリスト自体のStateには，以下の3つがある.
  - Running：ICEのチェックが進行中である状態.
  - Completed：使用するペアを決定し，ICEが完結した状態.
  - Failed：ICEのチェックが正常に完了していない状態.



【ICE】使用しない候補の解放（1）

- Selected Pairが決定しても，選択されなかったペアに対して接続確認をすることがあるが，それは不要なものである．
- 全てのCandidate Pairについて接続確認が完了した後，3秒間待った段階で，使用しない候補の解放を行うことができる．
  - Host CandidateおよびRelayed Candidateの解放は，ただ放置することで可能となる．
  - Server Reflexive Candidateの解放は，キープアライブをなくすことで可能となる．
    - **キープアライブ**（keepalive）：ネットワーク上で，データのやり取りをしていない状況で，接続が有効であることを確認するために行う通信のこと．

【ICE】使用しない候補の解放 (2)

- 接続確認完了後に使用しない候補の解放を行うことができるが，このとき解放までに3秒間待つ理由
  - 接続確認はPair Priorityの高いものから順に行うことになる．
  - しかし，Aggressive Nominationを用いた場合，予め全てのペアが接続確認の準備が完了している．すなわち，1組のペアについての接続確認終了直後に，次のペアについて接続確認を行うことになる．
  - そのため，あるペアがSelected Pairとして決定するより前に，次のペアの接続確認が開始されてしまう．
  - それが接続可能であるとすると，本来は用いないペアがSelected Pairとして選択される状況が起こる．
  - すなわち，Selected Pairが短時間で変化してしまい，安定しない．
  - したがって，Selected Pairが安定する（本来用いるペアに変更する）までの時間として設定されている．

参考：Lite Implementation

- 一部の機能を省略した特別なタイプの実装方法.
  - 常時パブリックネットワーク上にあるようなエージェントに使用される.
  - これまで述べてきた実装方法を，「Lite Implementation」に対して「Full Implementation」と呼ぶ場合がある.
- Full Implementationとの違い
  - 候補の収集は行わず，使用する候補はHost Candidateのみ.
    - エージェントは常時パブリックネットワーク上にあるため，そのグローバルIPアドレスおよびポート番号を用いる.
  - チェックリスト，ステートマシンは使用しない.

参考：Trickle ICE (1)

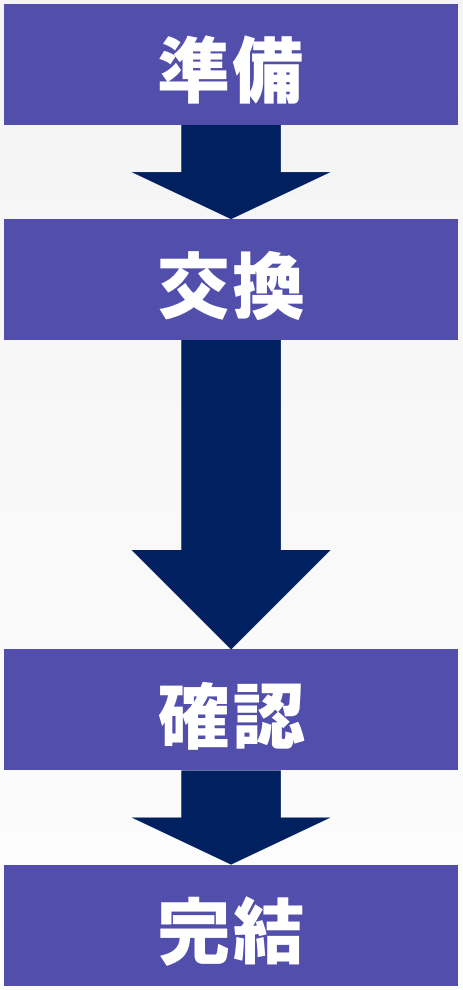
- **Trickle ICE**
  - 候補が発見されたら，収集段階であっても相手と交換し，ただちに接続可能性を確認することで，処理時間を短縮する方法．
  - RFCで標準化されておらず， Internet-Draftで公開されている．
  - 動作の流れは次頁参照

  - Trickleの意味
    - [動] 少しずつ流れる
    - [名] 少量

- Trickle ICEに対して，通常のICEを「Vanilla ICE」と呼ぶこともある．
  - Vanillaの意味
    - [形] 基本的な，普通の，ありふれた
    - [名] バニラ

参考：Trickle ICE (2)

- Trickle ICEの動作の流れ



ICEを使用するための候補を発見次第
直ちに相手の候補を交換する

交換が完了次第
直ちに接続可能性を確認する

参考：Trickle ICE (3)

- Trickle ICEについて，詳細は以下を参照

- “Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol draft-ietf-ice-trickle-07”
  - E. Ivov, https://tools.ietf.org/html/draft-ietf-ice-trickle-07, 2017/04/12参照
  - 2017/04/12現在の最新版，2017/08/31失効（予定）
- Iwashi.co “TrickleICEとは - WebRTCの要素技術 -”
  - 岩瀬 義昌, http://iwashi.co/2014/05/13/trickleice, 2017/04/12参照