



P2Pアプリケーションの実例

～ 工夫と課題『アリエル・プロジェクトA』と『Skype』～



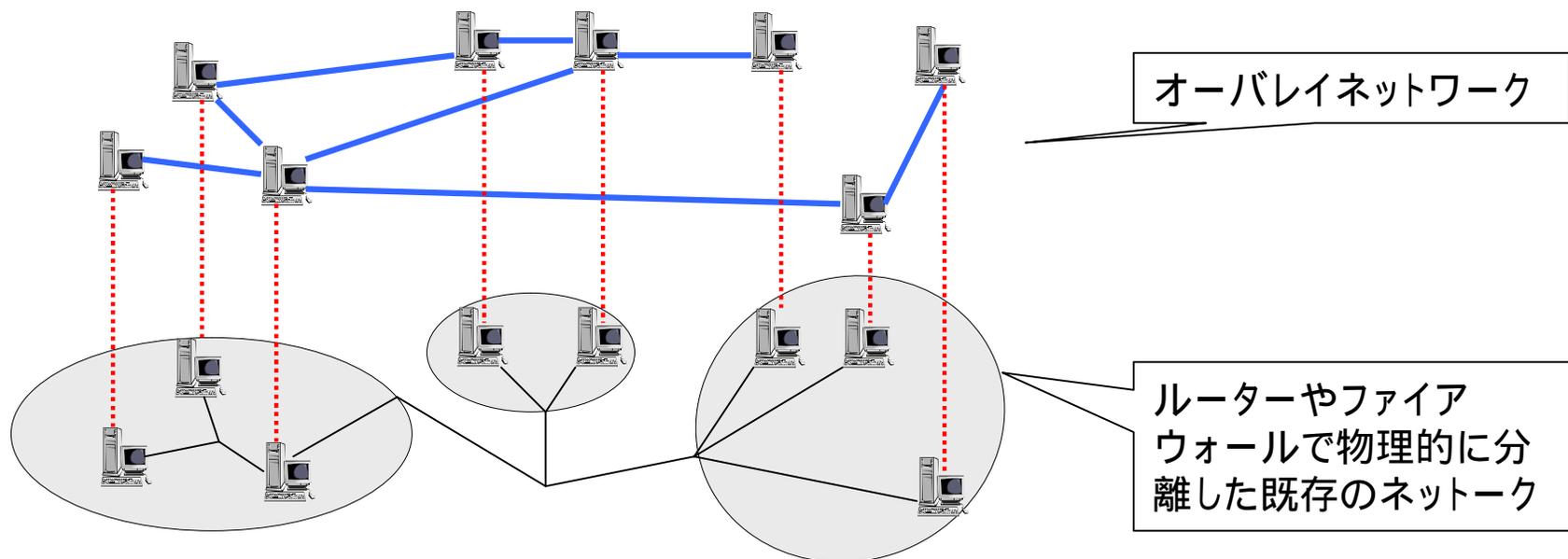
プロダクト・マネージャ 岩田真一

アジェンダ

1. 本講演の概要
2. プロジェクトA と Skype のシステム上の違い
3. プロジェクトA のシステム
4. Skype のシステム
5. P2Pの課題
6. 質疑応答(ディスカッション)

本講演の概要

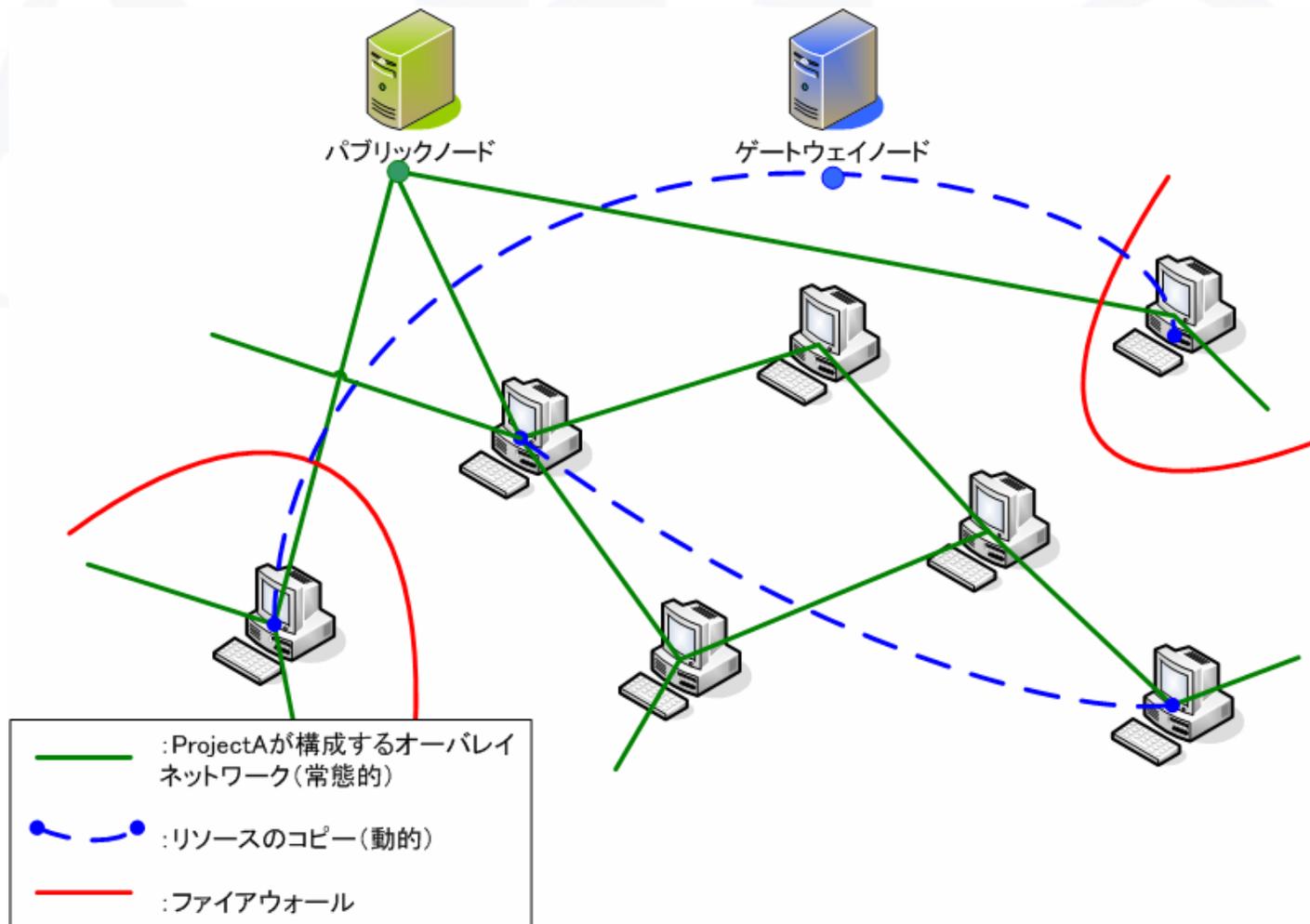
1. P2P技術再考
 1. 「盲信」と「毛嫌い」からの脱却
 2. 具体的なアプリケーション2つを通して可能性を共有したい
2. 本講演におけるP2P
 1. ソフトウェアで何でもやろうとする
 2. オーバレイネットワーク ハードウェアを抽象化
 3. アプリケーション層で実装されたネットワークライブラリ



プロジェクトA と Skype のシステム上の違い

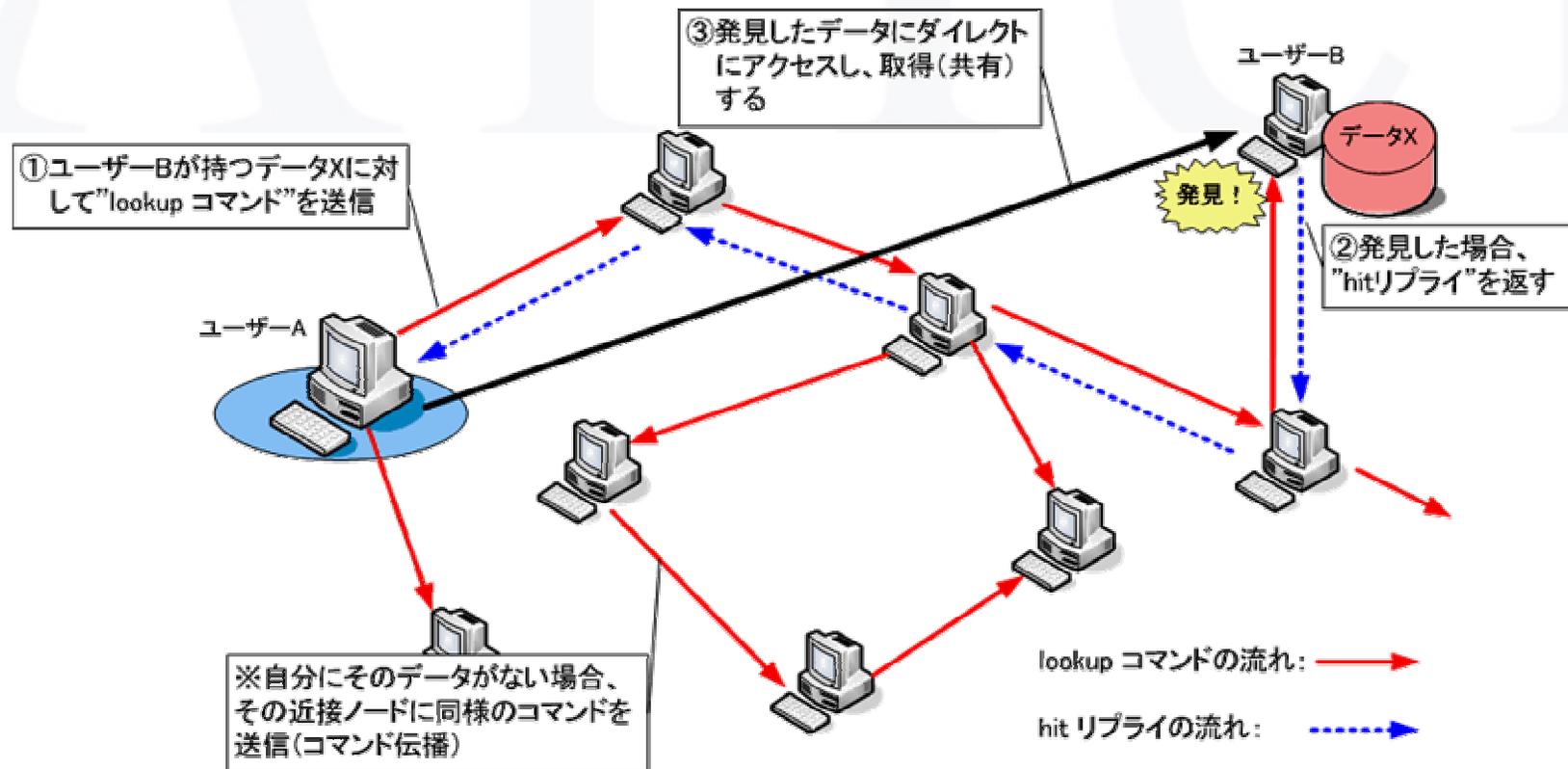
		
扱うデータ	静的データ 可逆性必須	リアルタイムデータ 不可逆で良い
適したプロトコル	TCP	UDP
FW越え技術	1. 逆接続リクエスト 2. ゲートウェイノードによる中継 3. HTTP Bridge	1. UDP Hole Punching 2. リレーノードによる中継
データ検索	バケツリレー方式 (Lookup-Hit)	スーパーノード群による分散ハッシュ (DHT) ライクな「Global-Index」
検索対象データ	グループウェアの一般的データ (予定や掲示板など)	ユーザー情報 (ID、ホスト情報)

システム構成

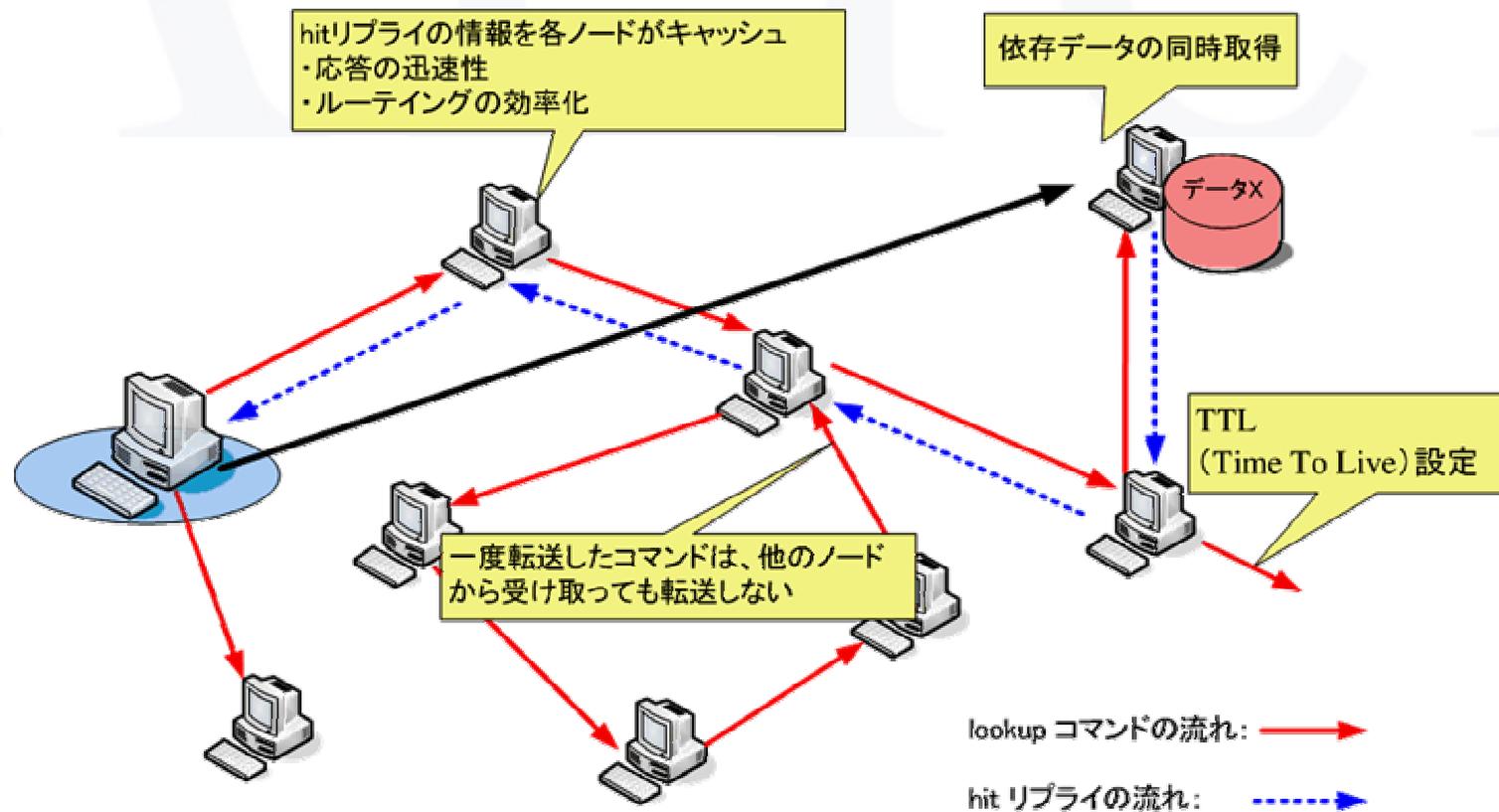


- 常態的な接続(手をつないでいる常態)とダイナミックなデータ取得のネットワークを使い分けている。
- パブリックノードを露出させているので、すべてのノードは手をつなげる(常態的なオーバーレイネットワーク)
 - Http-Bridge で 80 ポートしか空いていない環境にも対応

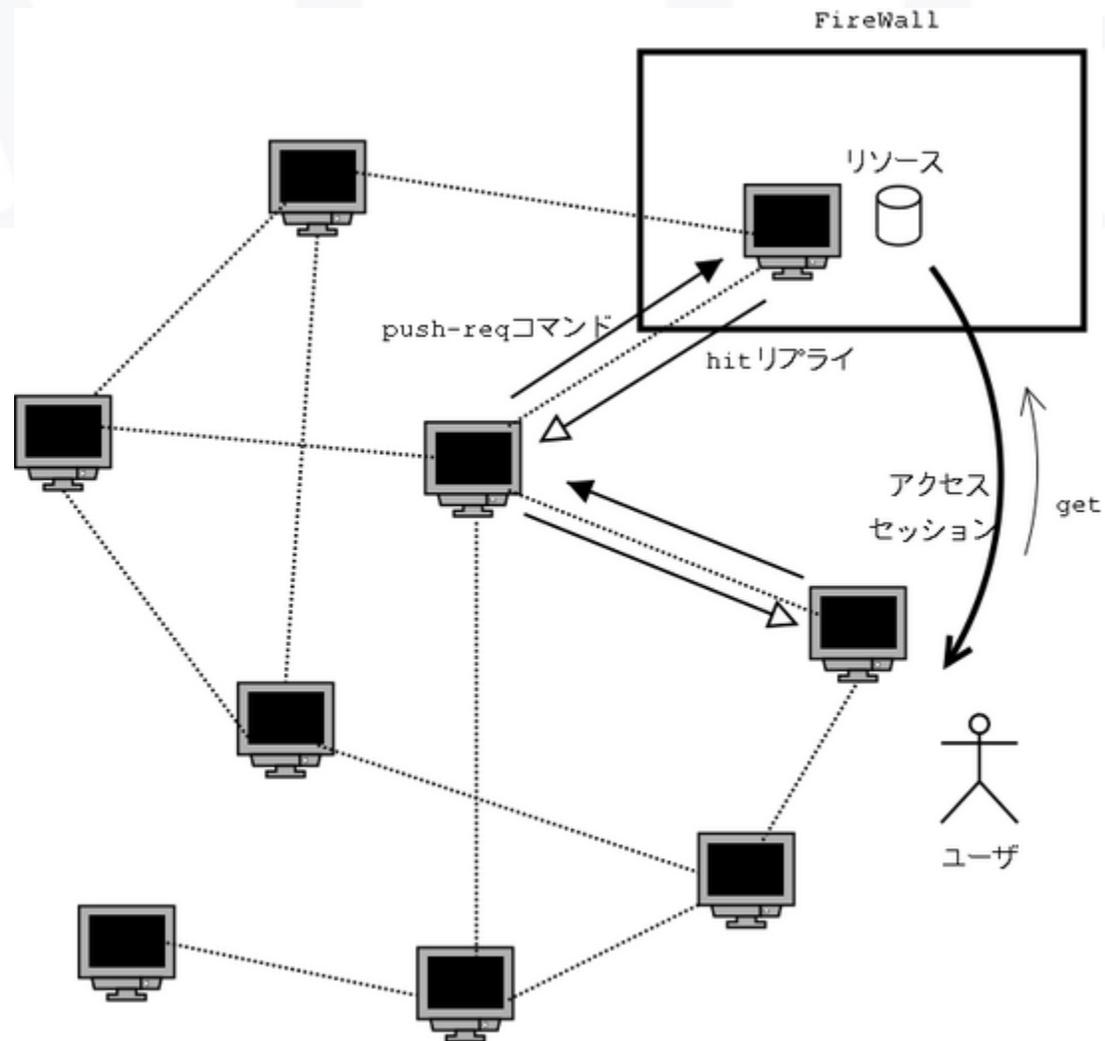
- バケツリレー方式による「Lookup-Hit」



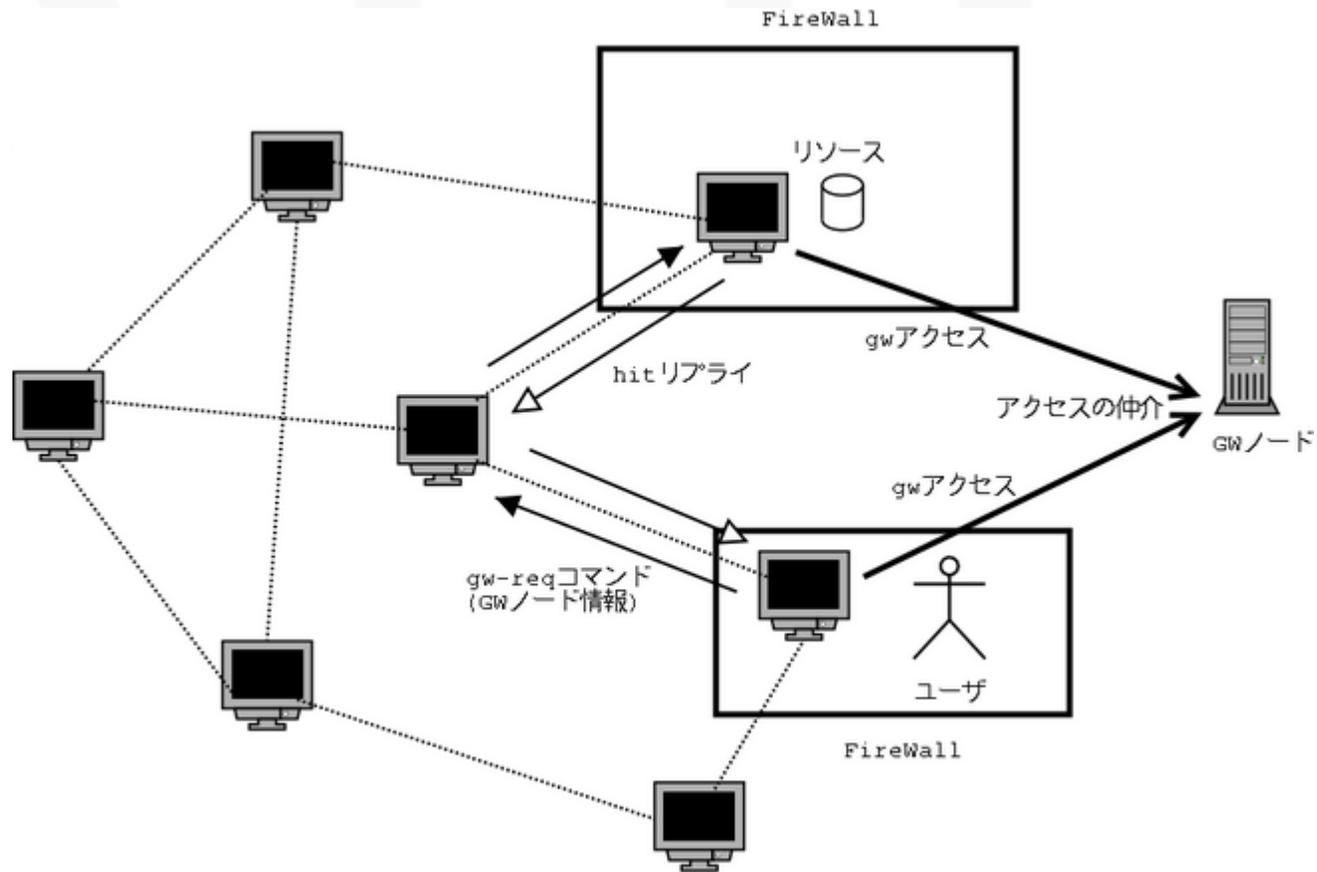
- Lookup-Hit モデルの効率化



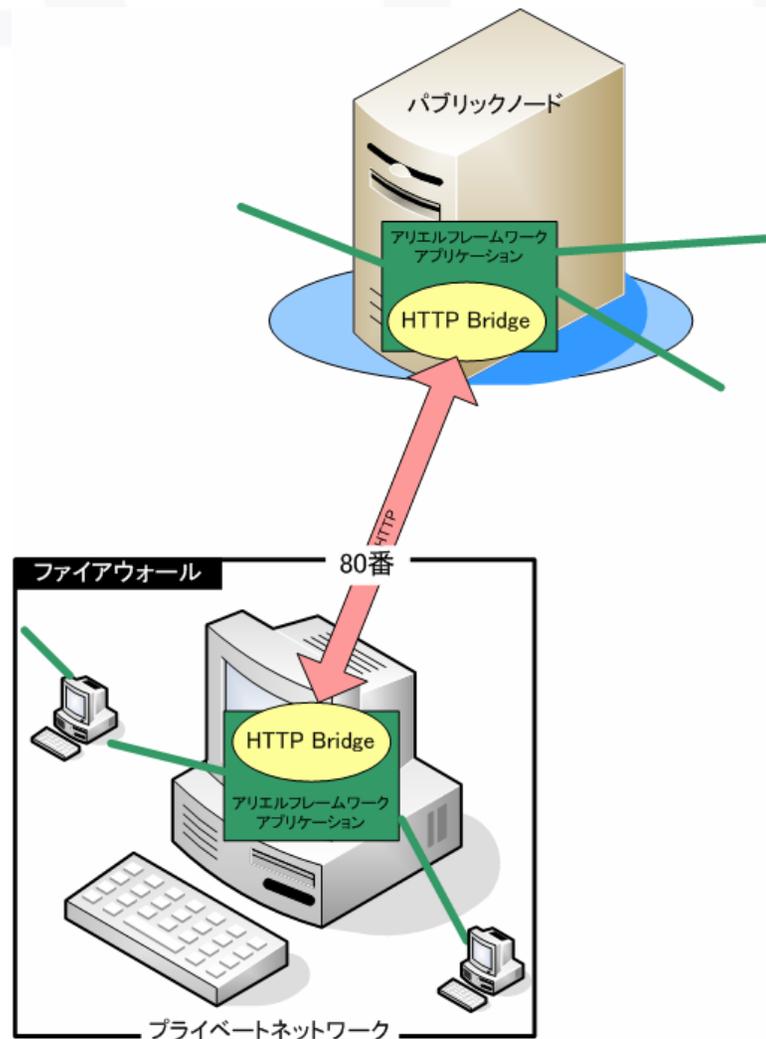
- 逆接続リクエスト



- ゲートウェイ(中継ノード)接続リクエスト



- HTTP Bridge



- 各ノードでHTTPカプセル化し、パブリックノードにてアンカプセル化。またはその逆。
- 80番、8080番、8081番しか空いていないプロキシで有効
- プロキシ情報はIEから取得 (手動設定も可) => 一括管理の必要性 => 企業版
- UPnP 対応は開発中

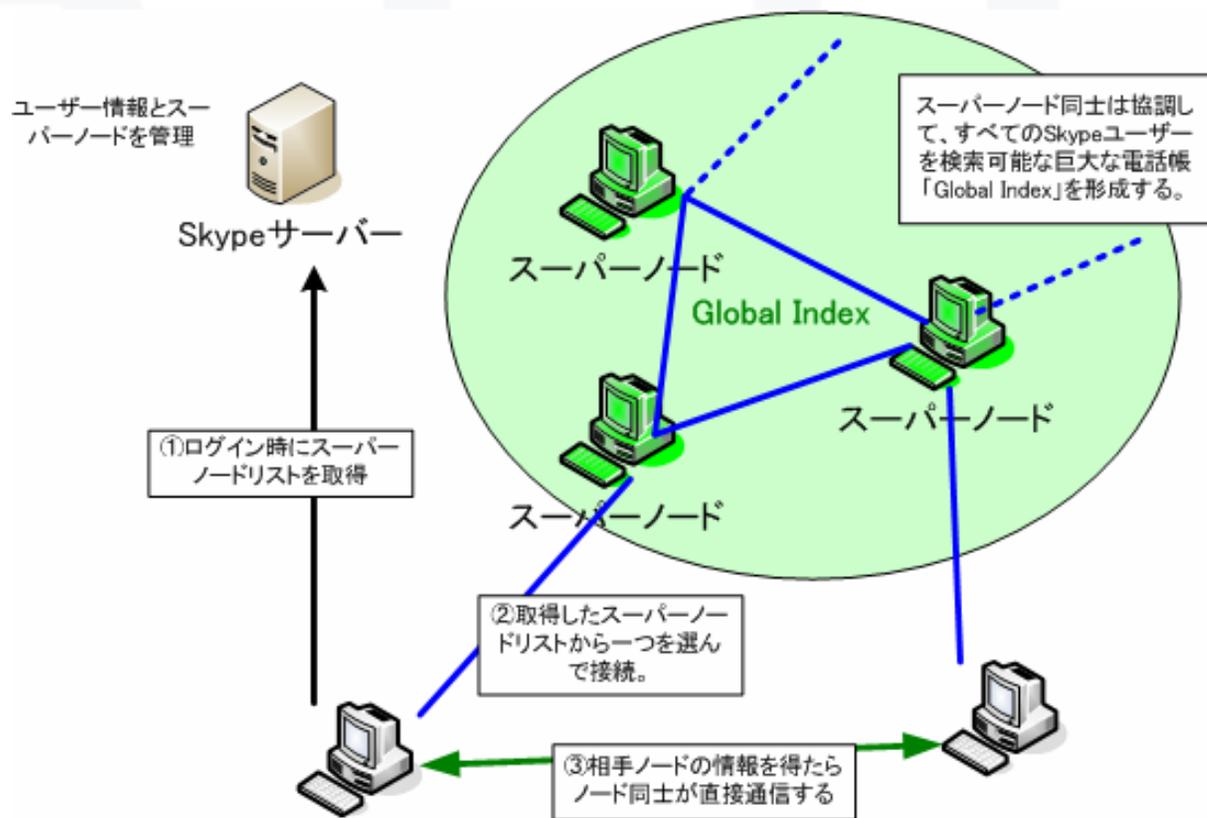
- コンフリクト
 - コンフリクトを防ぐ: ACLをつけて編集できる人を制限(本質的ではない)
 - コンフリクトを解決: マージ(一般的にはノーツのように「競合文書」)
- ノード間の時間のズレ
 - Ariel Framework では6時間以上時間がずれているノードとはセッションを張らない。(パブリックノードとのズレを検知し警告)
- ナローバンド
 - 依然として上りは遅い(Edge CacheとしてのP2Pに向いていない)
- 検索コマンドの帯域
 - P2Pシステムはリソースのコピーよりも検索コマンドの伝播による帯域の方がシビア
 - Hop数制限
 - Index-Docアーキテクチャに基づく peek 取得
- リソース取得にばらつき
 - 非同期取得(取得と閲覧を非同期にできる。メールのメタファ)
- ローカルディスクサイズの消費

- POSTに関する制限
 - POSTメソッドを通さない
 - POSTメソッドをキャッシュする
 - POSTデータを改変する
 - POSTデータサイズの上限
 - レスポンスデータサイズの上限
 - 大きなPOSTデータと小さいレスポンスのHTTPセッションを不正と見なす
- ヘッダに関する制限
 - 未知のUser-Agentヘッダのメソッドを破棄する
 - 未知のContent-Typeヘッダのメソッドを破棄する
 - Cookieヘッダを破棄する
- その他
 - HOSTヘッダが空のメソッドを破棄する(未確認)
 - HTTP1.1のみを通す
 - Digest認証
 - 透過プロキシ



システム構成

セミハイブリッド P2P

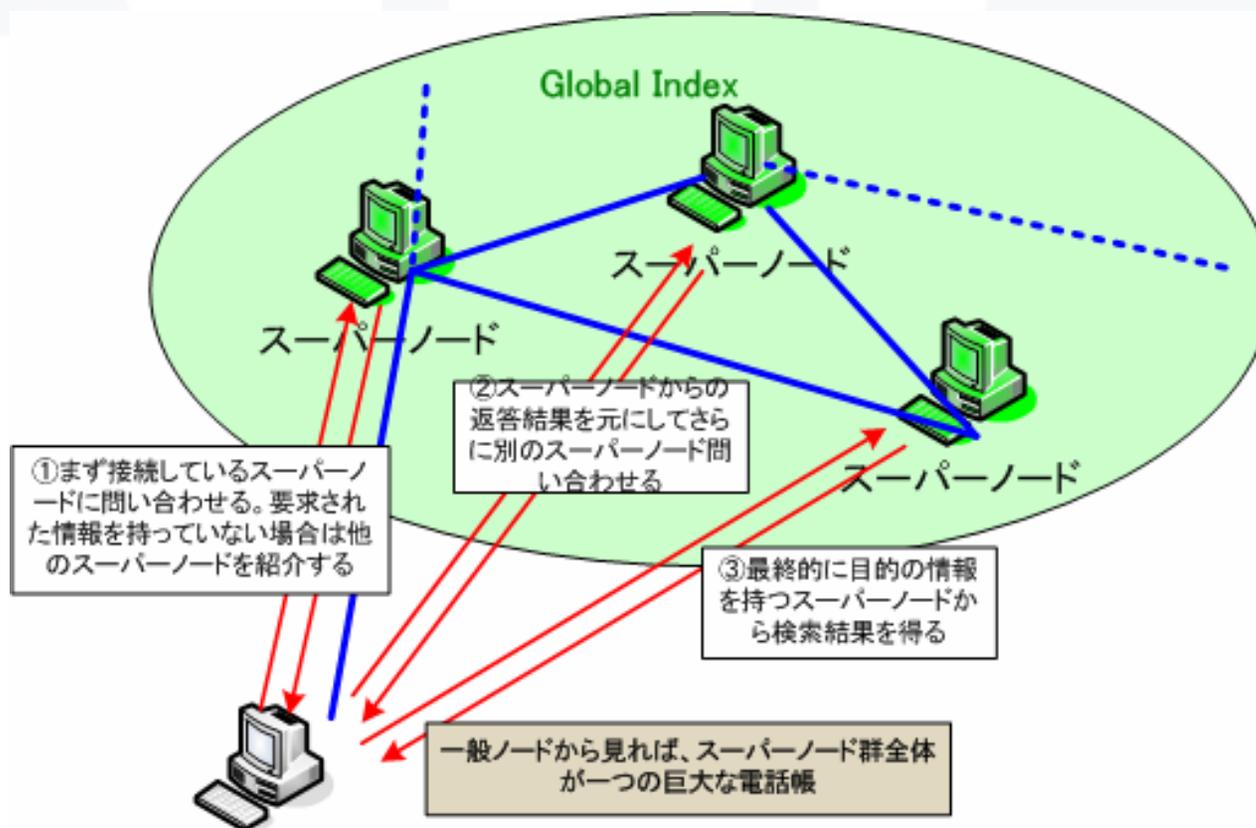


- 一般ノードは検索には参加しない(すべ他のノードに一樣に負担をかけることはしない)
- 検索はクラスタ化(Grid化)されている(スーパーノード同士の世界で協調している)
- ボランティアはスーパーノード/リレーノードなどの選ばれたノード(1%に調整される機構)



検索の仕組み

- スーパーノード群が構成する Global Index



- バケツリレーと比べて無駄なコマンド伝搬が少ない(分散ハッシュライクな収束の仕方)
- Global Index は 3G-P2P テクノロジーのひとつと定義している (Skype社)



検索の仕組み ~ パケットキャプチャ

Skype-UserSearch - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: not (http or arp or dns) Expres

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.10.149.245	143.107.232.124	UDP	Source port: 3463 Destination port: 58231
2	0.000197	10.10.149.245	c-8d5e73d5.023-2014-7	UDP	Source port: 3463 Destination port: 44516
3	0.000343	10.10.149.245	82-199-189-132.vallne	UDP	Source port: 3463 Destination port: 29278
4	0.000481	10.10.149.245	h9n3fls306o1049.telia	UDP	Source port: 3463 Destination port: 60802
5	0.000614	10.10.149.245	68.187.142.89	UDP	Source port: 3463 Destination port: 16806
10	0.195853	68.187.142.89	10.10.149.245	UDP	Source port: 16806 Destination port: 3463
11	0.204223	10.10.149.245	dhcp098076.res-hall.n	UDP	Source port: 3463 Destination port: 21215
12	0.311312	82-199-189-132.vallne	10.10.149.245	UDP	Source port: 29278 Destination port: 3463
13	0.317242	c-8d5e73d5.023-2014-7	10.10.149.245	UDP	Source port: 44516 Destination port: 3463
14	0.319722	10.10.149.245	herning.hostero.pil.d	UDP	Source port: 3463 Destination port: 5600
15	0.319932	10.10.149.245	D5E00A40.	UDP	Source port: 64859 Destination port: 3463
16	0.325992	h9n3fls306o1049.telia	10.10.149.245	UDP	Source port: 3463 Destination port: 3463
17	0.328349	143.107.232.124	10.10.149.245	UDP	Source port: 58231 Destination port: 3463
18	0.334389	10.10.149.245	ca-santaanahub-cuda1-	UDP	Source port: 3463 Destination port: 21651
19	0.375913	dhcp098076.res-hall.n	10.10.149.245	UDP	Source port: 21215 Destination port: 3463
20	0.526986	ca-santaanahub-cuda1-	10.10.149.245	UDP	Source port: 21651 Destination port: 3463
21	0.527447	10.10.149.245	b-195-90.cable.kpy.cu	UDP	Source port: 3463 Destination port: 60394
22	0.615648	D5E00A40.kabel.telene	10.10.149.245	UDP	Source port: 64859 Destination port: 3463
23	0.616182	10.10.149.245	rei.postech.ac.kr	UDP	Source port: 3463 Destination port: 15505
24	0.651022	herning.hostero.pil.d	10.10.149.245	UDP	Source port: 5600 Destination port: 3463
25	0.651531	10.10.149.245	cm-net-poa-C8B0C5A8.b	UDP	Source port: 3463 Destination port: 29021
26	0.854680	b-195-90.cable.kpy.cu	10.10.149.245	UDP	Source port: 60394 Destination port: 3463
27	0.855149	10.10.149.245	user-0cettgv.cable.mi	UDP	Source port: 3463 Destination port: 19380
28	0.985602	cm-net-poa-C8B0C5A8.b	10.10.149.245	UDP	Source port: 29021 Destination port: 3463
29	0.986051	10.10.149.245	68.187.142.89	UDP	Source port: 3463 Destination port: 16806
30	1.027266	user-0cettgv.cable.mi	10.10.149.245	UDP	Source port: 19380 Destination port: 3463
31	1.027733	10.10.149.245	dhcp16621146.indy.rr.	UDP	Source port: 3463 Destination port: 56422

まず5つのスーパーノードに対して検索コマンドを送信しているようだ

初期の問い合わせの返答を受けて次々に問い合わせを行っている。

Differentiated Services Field: 0x00 (DSCP: 0x00, Default, ECN: 0x00)
Total Length: 48
Identification: 0xc3e2 (50146)
Flags: 0x00
Fragment offset: 0

パケットはすべて暗号化されるため、検索しているユーザーIDはわからない。

```

0000 b2 c9 fd 85 a9 c3 00 80 6d 69 06 ac 08 00 45 00 .....mi...E.
0010 00 30 c3 e2 00 00 68 11 c5 44 c7 4a 62 4c 0a 0a .0....h..D..bL..
0020 95 f5 52 df 0d 87 00 1c e6 11 dc 50 02 6d 44 c0 ..R.....P.mD.
0030 6f 20 4e 76 a3 10 9a 4a 02 3b a1 cd 2d 2f      o Nv...J ;...-/

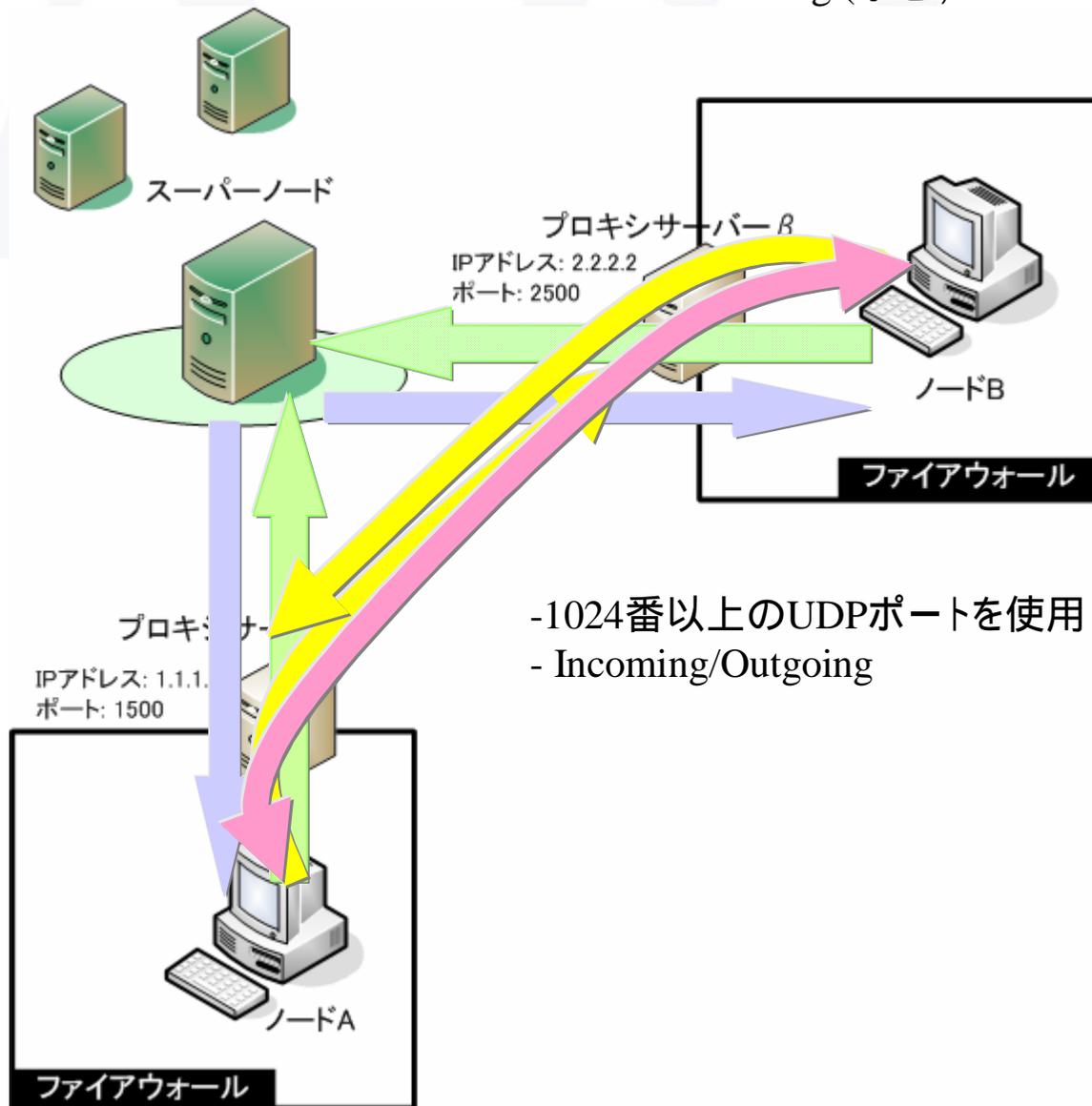
```

File: Skype-UserSearch 14 KE | P: 72 D: 38 M: 16



通信の仕組み1

- UDP Hole Punching (予想)



プロキシサーバーのアドレスとポート番号をスーパーノードに教える

相手のプロキシサーバーのアドレスとポート番号をスーパーノードから知る

相手のプロキシサーバーに向けてUDPパケットを投げまくる
このときに外向きに穴が開く(ルーティング情報が一定期間プロキシに記録される)

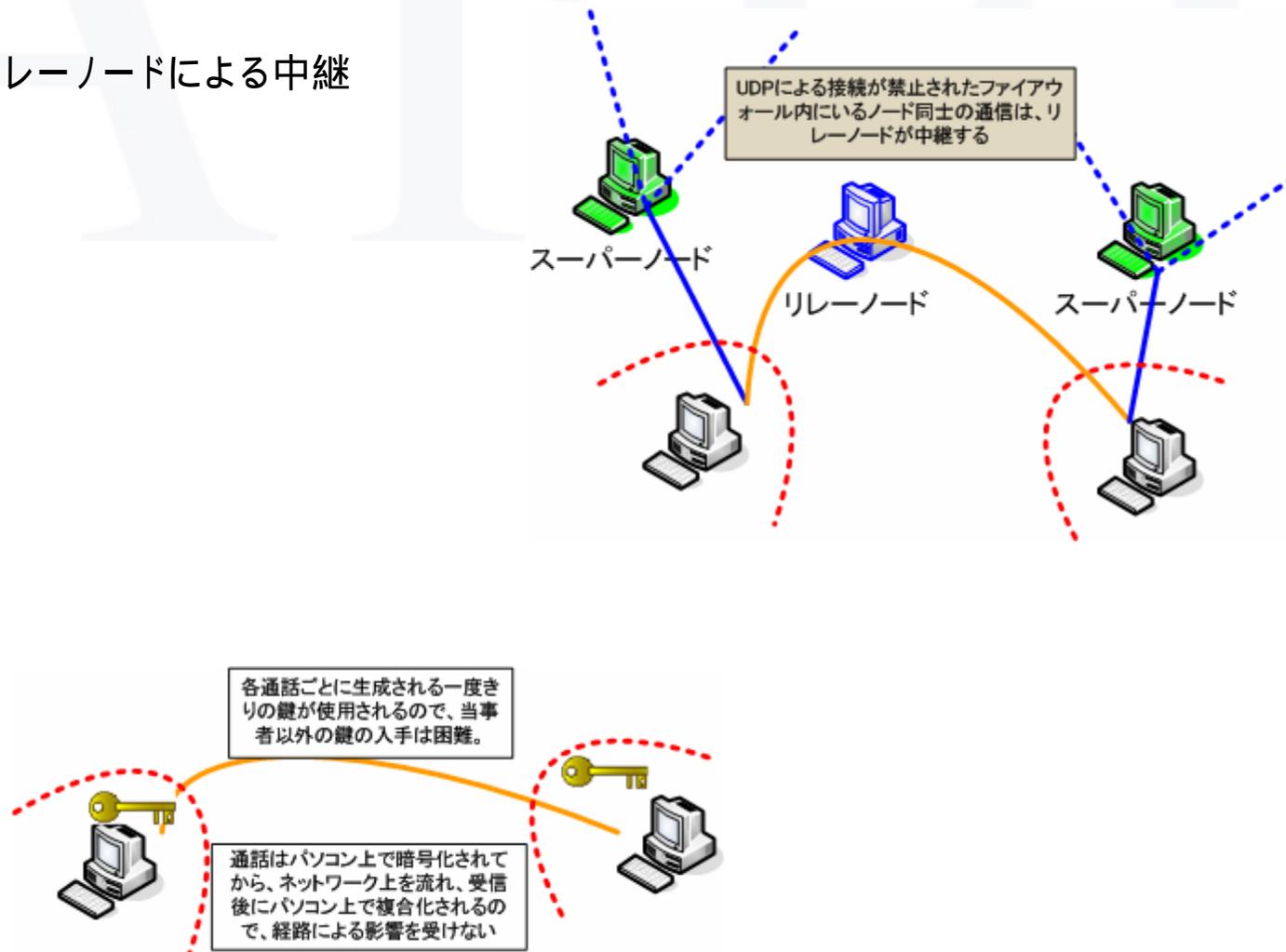
UDPの対称性により、ルーティング情報がプロキシに記録されている間はUDPパケットが直接届く

- 1024番以上のUDPポートを使用
- Incoming/Outgoing



通信の仕組み2

- リレーノードによる中継



P2Pの課題

1. より現実的なシステムへ -> いいとこ取りのハイブリッドP2P
 - 学術系、研究に寄りすぎるとおもちゃになる
 - 例:「自立的、自動的(開発者)」=「勝手に(管理者)」
 - 環境の問題(リテラシーも含めた)
 - 低スペックPCの存在という現実
 - 24hオンラインが不可能という現実
 - ローカルインストール

2. アンチP2Pと歩み寄り理解を求める
 - 著作権
 - 疎結合
 - ネットワーク管理者
 - 帯域制御装置の併用
 - UPnPの実装
 - システム管理者
 - Audit機能の充実
 - ローカルへのインストール(およびアップグレード)

3. キラーアプリ
 - (部分的にでも)P2Pでしか実現不可能なアプリの登場

P2Pの特徴再考 (P2P勉強会資料より)

データを持つ中心がない

- 匿名通信路 (関連技術: オニオンルーティングなど)
- ネットワーク冗長性

ローカルにデータを持つ

- データ冗長性
 - 分散させて持ちバックアップ
- **非同期性**
 - データの閲覧と取得を非同期にできる

ローカルで動作するモジュールを持つ

- 情報共有の手間を省く
 - (例) FTP(書き込み) + HTTP(閲覧) => P2Pモジュールに押し込む
 - 「P2Pはプロトコルである」(by Fukutommyさん)

すべてのノードがオンラインである状況が望ましい

- いつもオンラインのノード=>サーバー
 - (例) 北海道の病院(サーバー同士)をP2Pで結ぶ(情報通信研究機構)。

流通力

- ファイル交換ソフトで証明されたもっとも明らかなP2Pテクノロジーの効用

質疑応答

