2004年 4月 8日 (木)

Project JXTA

- loosely-consistent DHT with limited-range walker

首藤一幸
産業技術総合研究所 グリッド研究センター

資料

- http://www.jxta.org/project/www/white_papers.html
  - [1] Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, Bill Yeager, “Project JXTA 2.0 Super-Peer Virtual Network”, May 2003
  - [2] Bernard Traversat, Mohamed Abdelaziz, Eric Pouyoul, “Projext JXTA: A Loosely-Consistent DHT Rendezvous Walker”, March 2003
- **JXTA関連メーリングリストのメール**
- **JXTAの使用経験**

What's JXTA？ www.jxta.org

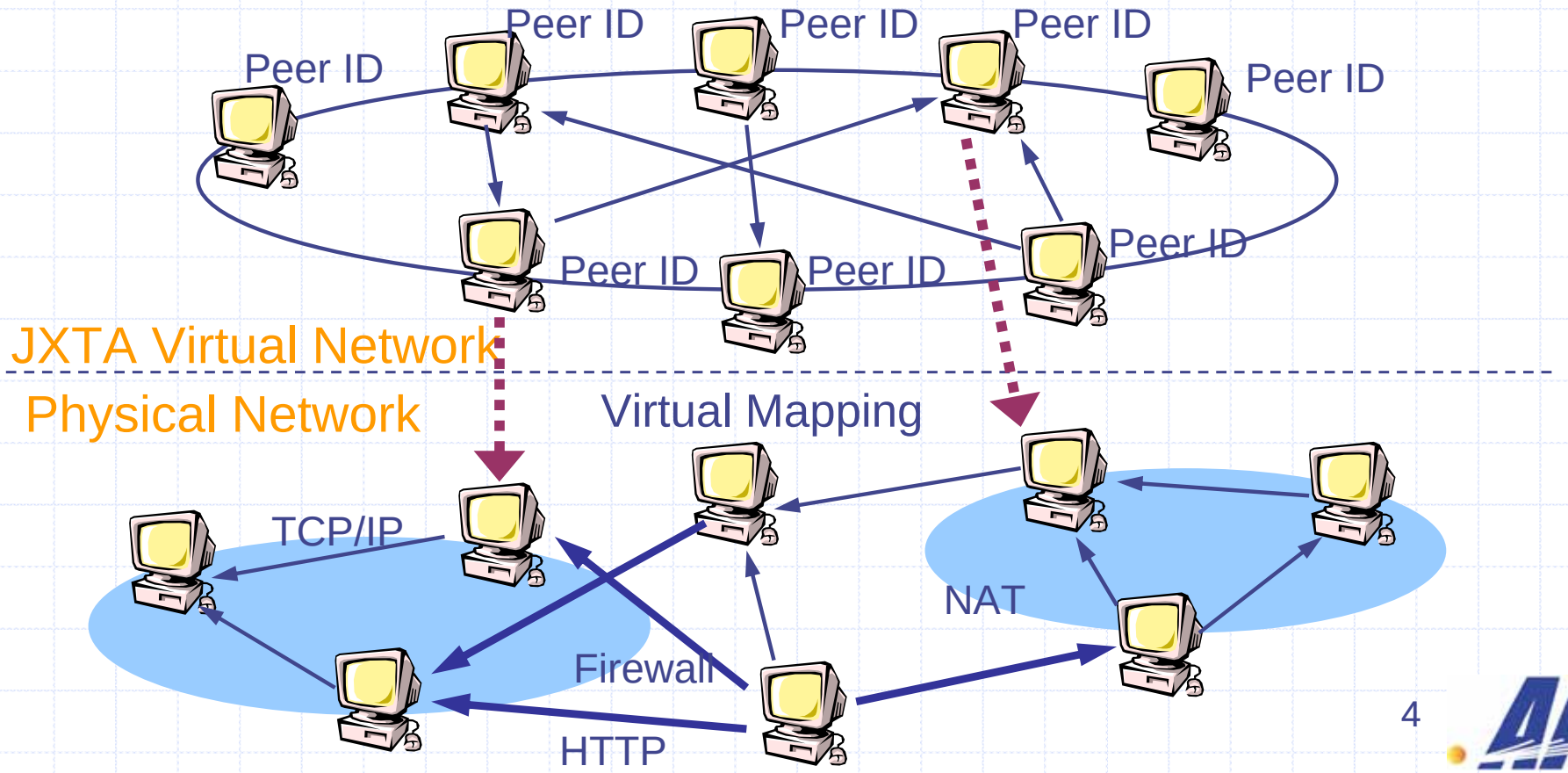
- P2Pソフトウェアに共通の機能を提供するプロトコル、参照実装 (Java & C)、プロジェクト
  - JXTAを使って、様々なアプリケーション、再利用可能サービスが実装されている。jxta.orgは数十のプロジェクトをホストしている。
    - コラボレーション: MyJXTA2, JXCube, ...
    - 分散処理: JNGI, P3
    - ファイル共有、ストリーミング、ゲーム、...
  - 携帯電話器 (J2ME) 向けの実装もある。
  - APIは、2001年末の時点でほぼ固定されている。
- 語源は "Juxtapose" (並置する)
  - Client-Serverモデルといった既存モデルと並存。
    別に競合するわけではない。
- 2001年 4月 25日、Sun Microsystems社からアナウンス
- オープンソースコミュニティ
  - Apache Software License 1.1 ほとんどそのままのライセンス条文
  - <name>@jxta.org の登録者は 17,000名 (2004年4月)。
  - ダウンロード数 200万以上 (2003年12月)。

AIST

JXTA の提供する Network Overlay

"Project JXTA 2.0 Super-Peer Virtual Network" 中の図

- 計算機はピアIDで識別される。
  - JXTAの実装は通信にTCPやHTTP, IPマルチキャストなどを用いるが、JXTA の利用者はそれを意識せずに、任意のピアを相手に通信を行う。
- ピアはピアグループを作成でき、複数のピアグループに参加できる。



Abstractions

- JXTA ID
  - あらゆる資源に割り振られる ID
    - peer ID, peer Group ID, pipe ID, …
  - 参照実装では、128 bitのUUID。乱数で作られる。
  - peer ID は、位置 (IPアドレス等) とは独立。
- Peer
  - 何か。だいたい、1台の機械だと思っておけばよい。
- Peer Groups (PG)
- Advertisements (adv, 広告)
  - 各資源を説明するXML文書。resource descriptors。
    - peer adv, PG adv, pipe adv, route adv, peer endpoint adv, …
    - 自由に、新たな種類の adv を作成できる。
  - 例: PG へjoinする手順
    - 1. PG advを用意する → (PG adv を publish する) → joinする
    - 2. PG advを入手 → joinする
  - 例: pipe を使った通信の手順
    - pipe adv を用意する → pipe advを元にして input pipe を作る → pipe adv を publish する → 他のピアが pipe adv を発見する → そのピアが output pipe を作る → 通信開始
- Pipes
  - 通信チャネル
  - 3種類ある:
    - Unicast　　一方向、信頼性なし
    - Propagate　　PG内にブロードキャスト
    - Secure (and reliable)
  - 最近は、pipe の上に実装されている Jxta Socket がよく使われる。
- Resolver
  - binding機構
  - JXTA においては、binding とはすなわち adv の発見 (discovery)。

History - Java の参照実装

- 1.0 と 2.0以降で、かなり中身が違う。
  - 2.0からDHTが導入された。

Date	Release Version	コード名	CVS Tag
2001/4/25	Build 14d, 04-21-2001		STABLE_20010422_0200
...	...		...
2002/9/24			STABLE_20020924T1446PDT
2003/3/1	2.0		JXTA_2_0_Stable_20030301
2003/6/9	2.1		JXTA_2_1_00
2003/9/16	2.1.1		JXTA_2_1_1_00
2003/12/15	2.2	Timpani	JXTA_2_2_00
2004/3/15	2.2.1	Churrasco	JXTA_2_2_1_00
2004/6/14 (予定)	?	Jambalaya	?
2004/9/13 (予定)	?	Yaprakh	?

JXTA 1.0

JXTA 2.x

徐々に
品質向上

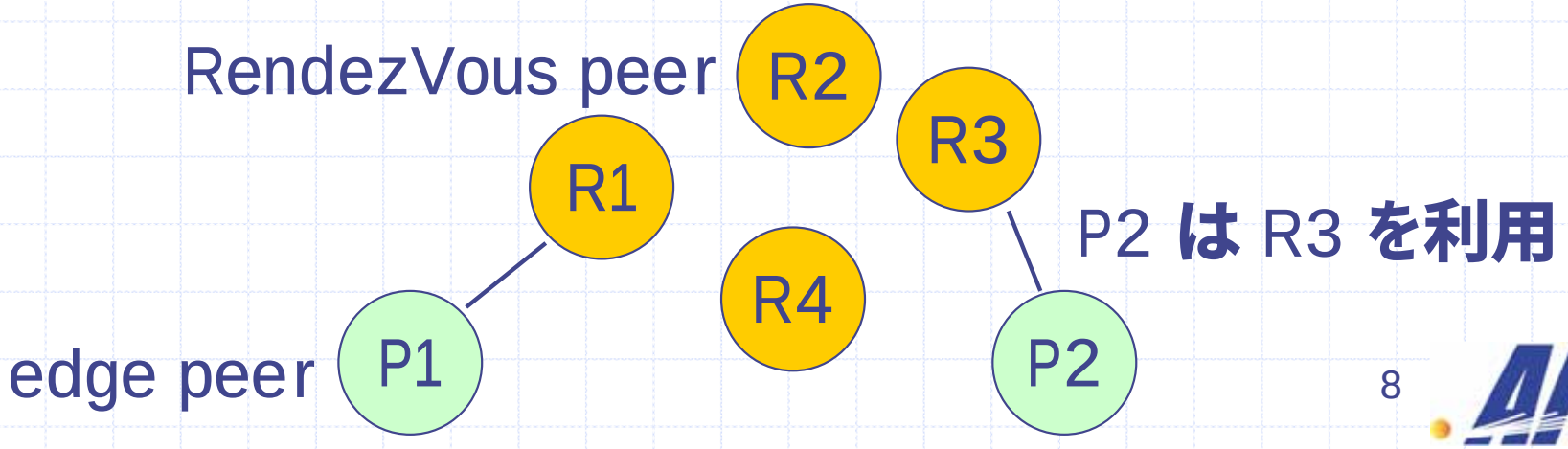
AIST

Resolverの変遷とJXTAのDHT

- JXTA 1.0
  - Propagate
    - 要は、flooding。Gnutellaと同じ。
      - flood [動詞]: 氾濫させる, 水浸しにする。
    - (ピアグループ内の) 全ピアへのブロードキャスト。
- JXTA 2.x
  - A loosely-consistent DHT
    - Consistent viewの維持を頑張りすぎずに、維持のコストを、若干、検索時に転化する。
      - ピアの出入りが激しい状況では、consistent viewの維持コストは非常に高くなってしまう。
      - cf. CAN, Chord, Brocade
  - with a limited-range (RendezVous) walker
    - DHTでは直接発見できなかった場合に用いられる。

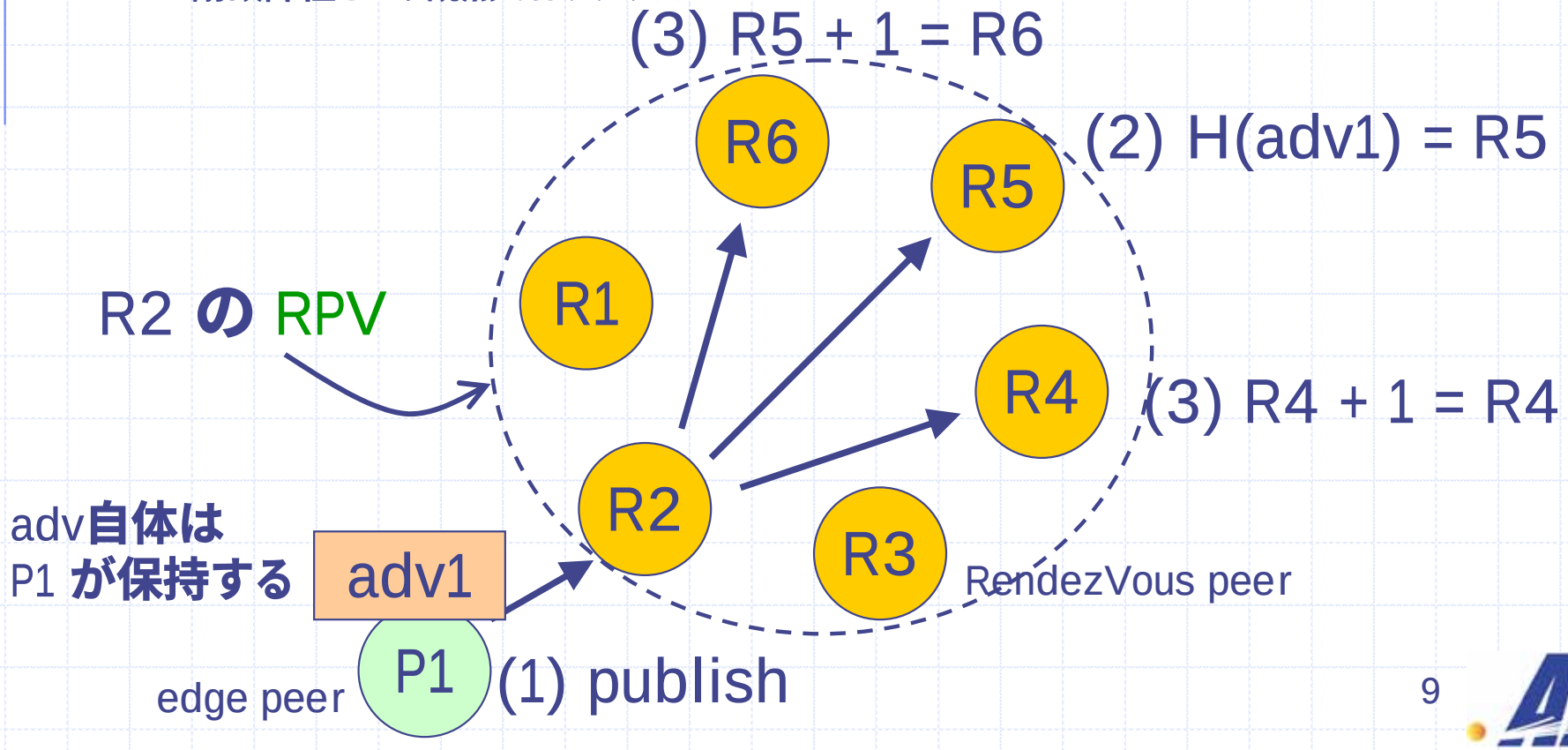
RendezVous Super-Peers

- Peerの属性
  - RendezVous か否か
  - Relay か否か (後述)
- RendezVous について、ピアは2種類に分けられる。
  - RendezVous peer (rdv)
    - resolver、すなわち advの発見に協力するピア。
      advのインデックスを保持して、問い合わせに応える。
      - advのインデックス: advのハッシュ値 → advを保持するピアのID
    - adv 自体を保持するわけではない。
  - Edge peer
    - rdvにぶら下がる。
    - どの rdv にぶら下がるかは、初回実行時は、設定ファイルや GUI からの設定として与える。2度目以降の起動では、前回の rdv (群) を覚えている。
    - 利用していたrdvと通信できなくなったら、他のrdvに乗り換える。
    - rdvとの接続を制御 (確立、切断、確認) する API も用意されている。
- rdv 間に恒常的なコネクションや、トポロジがあるわけではない。
- edge peerが自動的にrdvに化けることや、その逆もある。
- rdvであるということは、ある PG における局所的な属性。
  - i.e. peer group A の rdv が peer group Bでは rdv とは限らない。



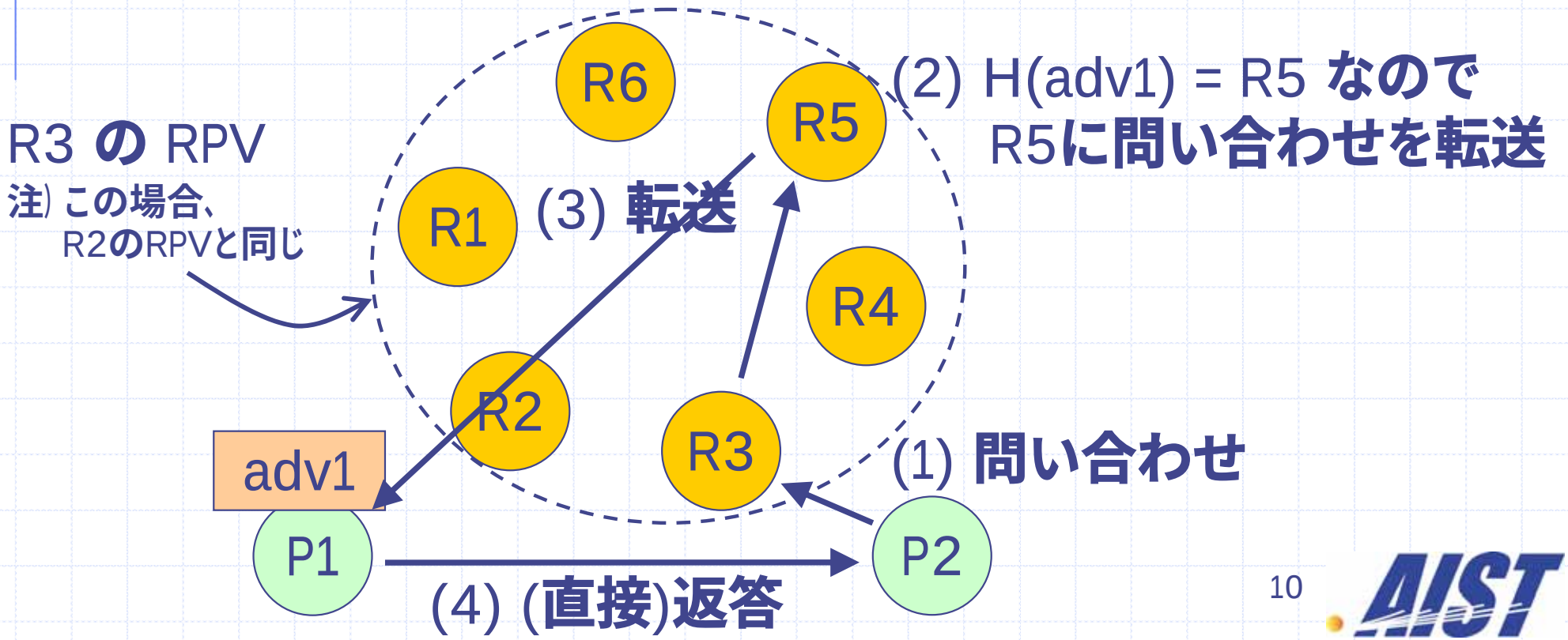
advertisement の publish

- ハッシュ関数は、advを入力として、peer IDを返す。
- RPV: RendezVous Peer View
  - ある rdv から見た rdv 群。順序付きリスト。peer IDでソートされている。
  - RPVの管理は、後述。
- R5だけでなく、その周辺 (+1, -1) にもインデックスを保持してもらう。
  - rdvの加入、脱退後も、そのadvが発見されやすいように。
  - 例: R5が抜けたら? いくつかrdvが加入して、H(adv1) が現在のR4を指すようになったら?
- peer IDの遠近は、ネットワーク的な遠近とは無関係。
  - R4とR5は、それぞれ地球の裏側にあるかもしれない。
  - 耐故障性という観点では、プラス?



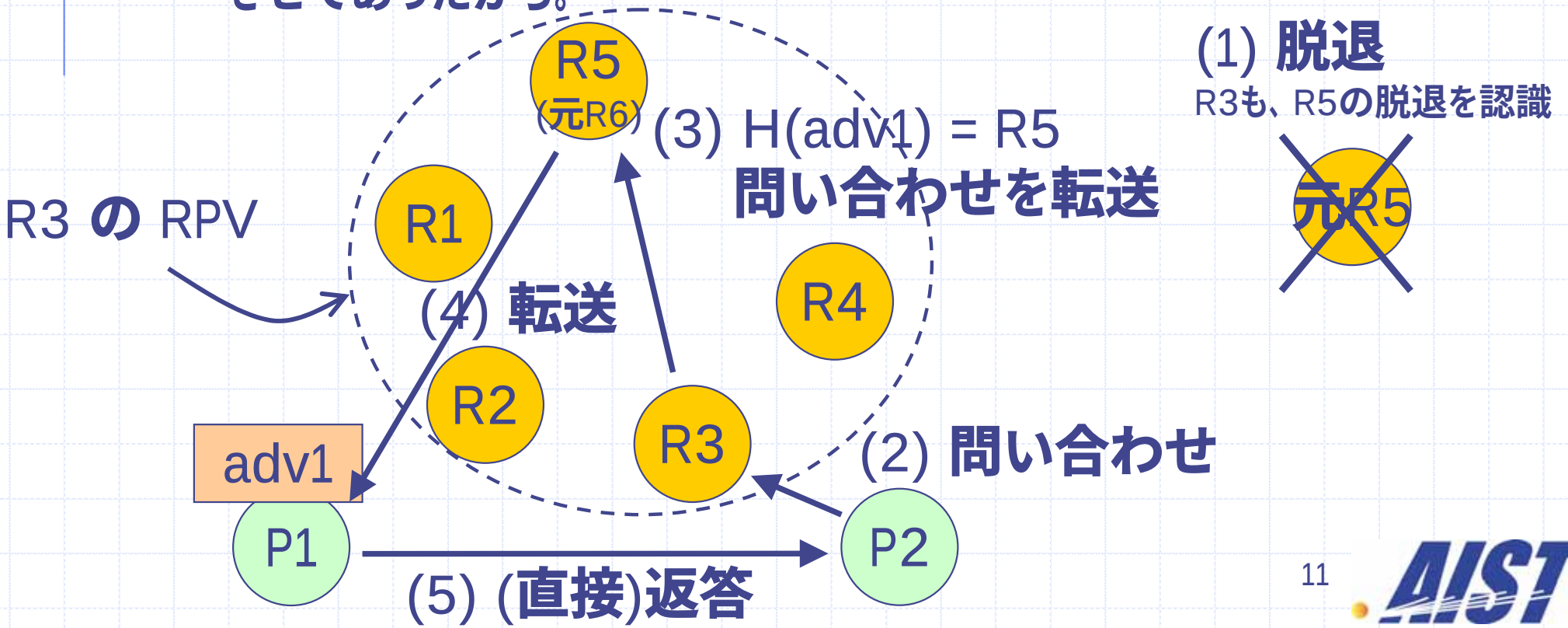
adv の発見- DHT

- P2が adv1 を検索。DHTで発見できる。
  - 検索キーとして、いくつかの属性を使える。
  - 値にはワイルドカードを使える。
  - 例: key = “Name”, value = “*foo*”
    key = “PeerID”, value = “urn:jxta:uuid-4C885537240……..”
- 不明な点:
  - 実際のハッシュ関数。ワイルドカードを使った検索をどうやって実現しているか?
  - advをpublishする際、検索キーの種類ごとに、異なる rdv にインデックスを保持させている?



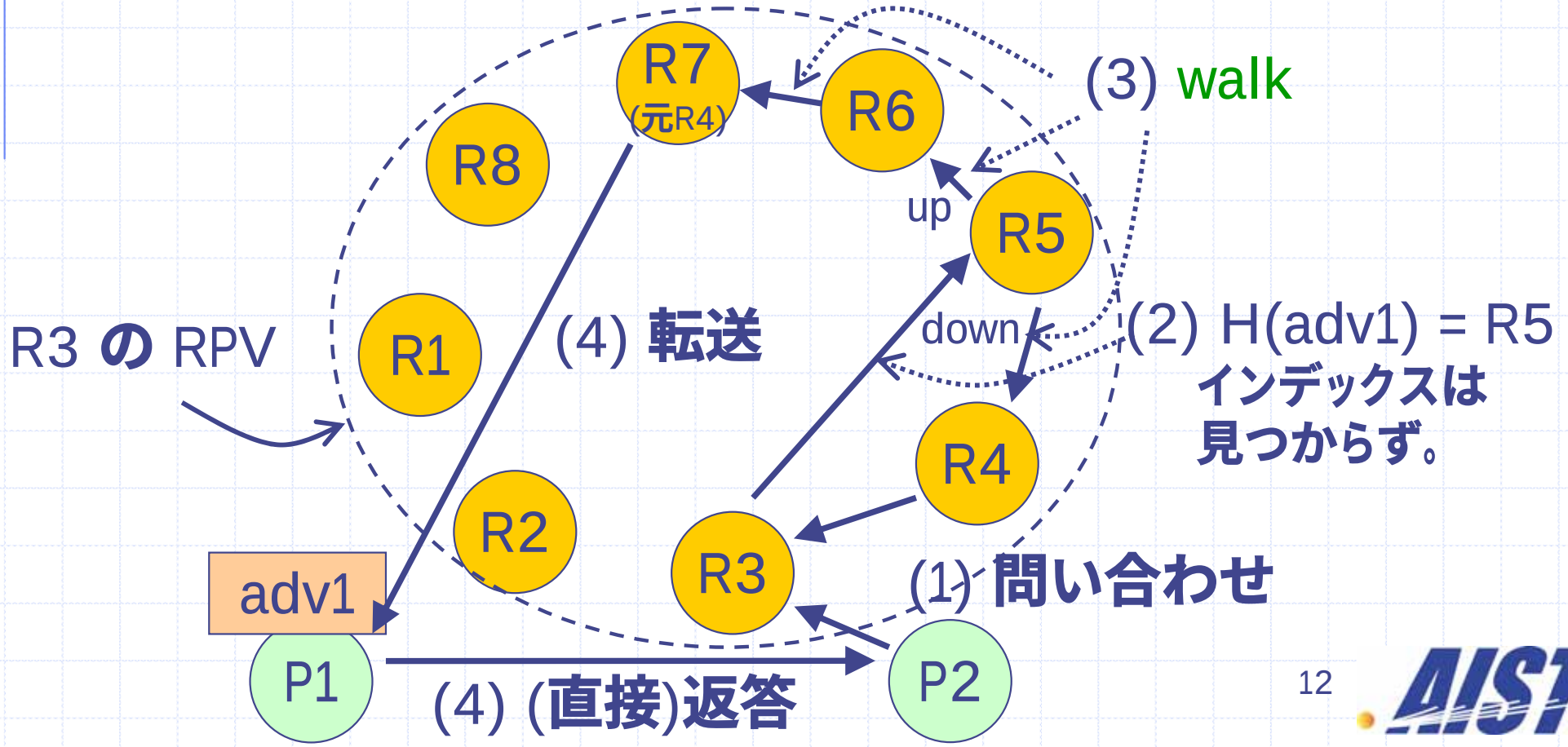
adv の発見- rdv構成が変化

- R5がダウン。まだ、DHTで発見できる。
  - R3のRPVは、R1 ～ R5 になる。
- P2がadv1を検索。
  - 現 R5 (元 R6) は adv1 のインデックスを保持している。
    なぜなら、インデックスは、元R5だけでなく、元R4と元R6にも保持させてあったから。



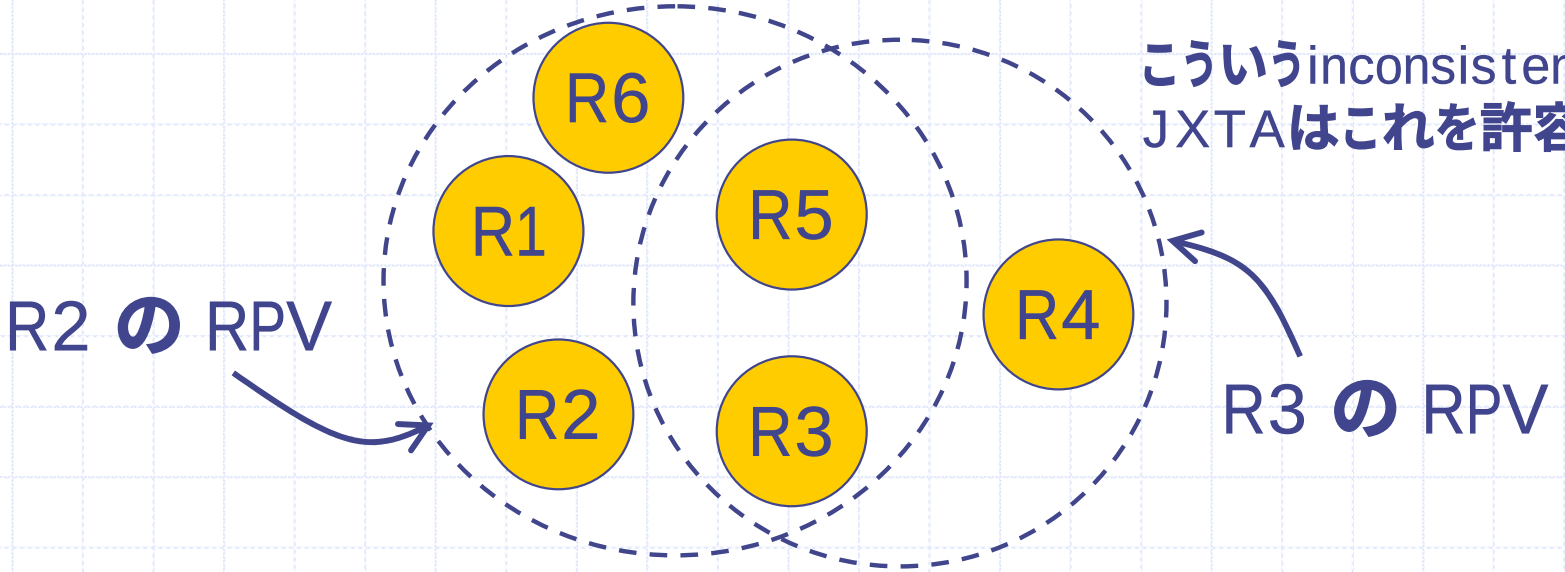
adv の発見 - rdv構成が激しく変化

- たくさんの rdv が加入。
- P2が adv1 を検索。DHTでは発見できず、とうとう walk が起こる。
  - 現 R5 はインデックスを持っていない。
  - up, down 双方向に、walk が始まる。
    - up方向は、R7 (旧R4) がインデックスを保持しているので、そこで walk 停止。
    - down 方向は、R3まで戻った時点で walk 停止。
- 不明な点
  - walk の最大ホップ数
  - rdv群は、リストというより ring として扱われる? (i.e. R7 → R8 → R1)
  - walk は、R5 の RPV だけに基づいて行われる? (e.g. source-routed)
    それとも、R5 の RPV で次は R6 だと決まり、R6 の RPV で次は R7 だと決まる?



RPV のメンテナンス

- RPV: RendezVous Peer View
  - ある rdv から見た rdv 群。順序付きリスト。peer IDでソートされている。
- 全 rdv が同一の (consistent) RPV を持つことにはあまりこだわらず、頑張りすぎない。
  - ゆえに、"loosely-consistent" DHT。
  - cf. CAN, Chord, Brocade
- RPV のメンテナンス:
  - rdvは、定期的に、与えられた数のrdvを自分のRPVからランダムに選び出し、ランダムなrdvに送る。
    - → いつかは、全 rdv が全 rdv を把握できる。
  - 近隣 (+1, -1) の rdv にハートビートを送る。
    - 返答がなければ、RPV から削除する。
- 不明な点:
  - RPV からの削除は、ハートビートに未返答の場合のみ?
    他の rdv の生存を確認できる機会は他にもあるか? 例えば、rdv リストの送信時など。



こういうinconsistentな状態も起こり得る。
JXTAはこれを許容する。

Relay Super-Peers

- NA(P)T, firewall越えや、異なる下位プロトコルをまたいだ通信を可能とする仕掛け。
  - 例えば、TCPとBluetooth

- To be written

その他の話題

- adv の lifetime と expiration time
- seeding RendezVous
- edge peer の自動 rdv 化とその逆。
  - rdvにぶら下がれなければ、自分がrdvになる。
- routing
  - route advertisement
- IPマルチキャストの使用。
  - propagateやdiscoveryに、IPマルチキャストも使える。
  - 設定次第。
  - rdvは、マルチキャストで通信できる相手が多いと、CPU負荷が非常に高くなる。

議論

- ハッシュ関数は、RPVの変化、つまり rdv の出入りを補償する。 by 文献 [1]

議論

- rdv の加入･脱退によって、H(adv1) と実際に adv1 のインデックスを持つ rdv が大きくズレてしまった場合、自動的な補整はない?
  - 検索のたびに、ホップ数の多い walk が要ることになってしまう。
  - JXTAを使うアプリ側が、再 publish するしかない？
- walk で adv1 が見つかったら、その時点で H(adv1) = R5 にインデックスを持たせるといいかもしれない。
  - さもないと、検索のたびに walk が必要。

議論 - 現実装？将来も？

- 複製 (replica) のサポートはない。
- adv自体は、advをpublishしたピアが持ち続ける。
  - 配置の最適化、移動といったサポートはない。
  - もしJXTAを使うとしたら、コンテンツの在りかをadvで表現することになる？
- ネットワーク的な遠近を考慮していない。
  - 逆に、adv がネットワーク的、地理的に分散されるという耐故障上のメリットもある。
- コンシューマ機器にも rdv をやらせる?
  - やらせないとしたら、インデックスの(分散)管理は誰がやる？
- P2P 的な分散管理はやっぱり大変。
  - 並行して、集中管理 + 負荷分散 + DB一貫性管理も要検討？

JXTA 現実装の不安 - Java用参照実装 2.2.1

- 性能
  - 通信のスループット
    - 1回あたりの送受信量を調整しても、GbE上のPC間で、100 Mbps 程度。
    - 現在のFTTHを埋めるくらいはできる、とも考えられる。
  - 通信遅延
    - GbE上で、片道 4.5 msec。
      - cf. 生のTCPでは、0.06 msec。
- スケーラビリティ
  - 掲げられている数値目標:
    - 1,500,000ピアのうち、300,000ピアがアクティブ
  - rdvが面倒を看ることができる edge peer 数は 20程度?
- 機能的な問題 ?
  - rdv上でJXTAアプリケーションを動作させると、discoveryなどの機能を正常に利用できない?

AIST