Portable Jini Services

Tetsuro Kimura

Toshiba R&D Center

Outline

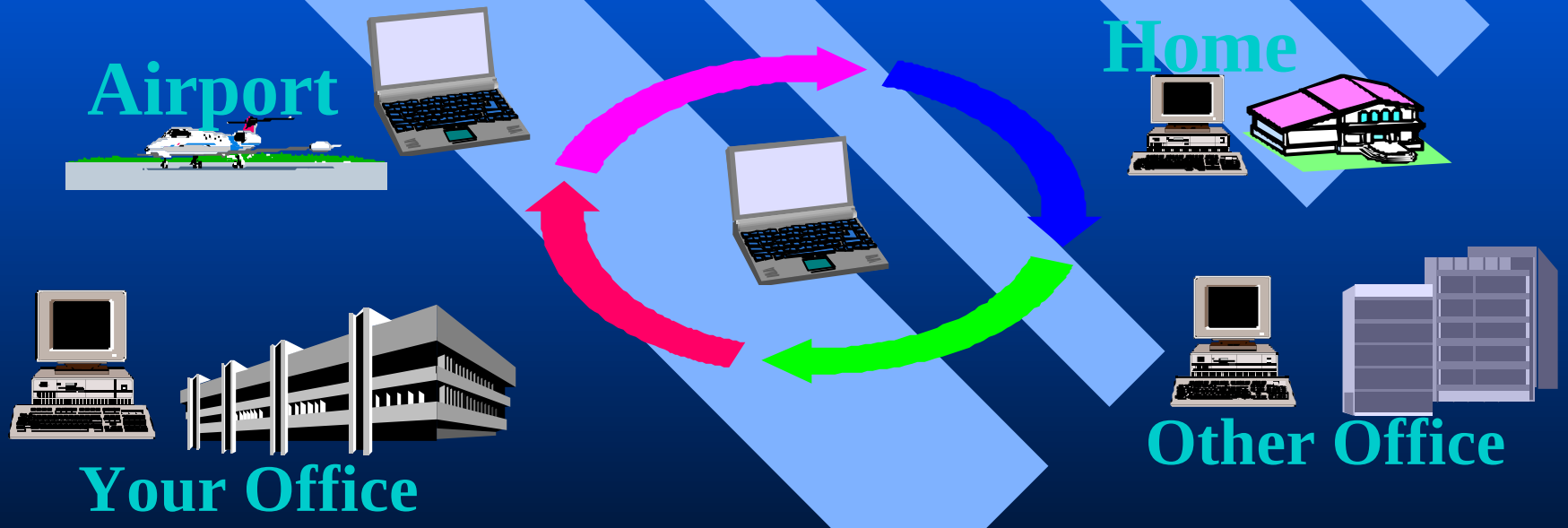
- Portable Services
- Portable Jini Services
- モバイルでの問題点について
- モバイル拡張について
- まとめ

Portable Services

- モバイルコンピューティングの時代
  - 携帯型PCの小型化、性能の向上
- 操作性　VS　携帯性
  - 携帯性の重視、操作性の犠牲
  - 他の機器との連携
- 個人用のサービス
  - 多数の情報機器を併用する時代へ
  - 個人環境の核となるサーバ
- サーバも移動する時代へ

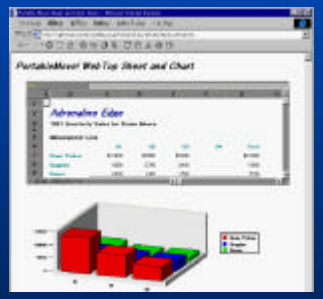
利 用 形 態

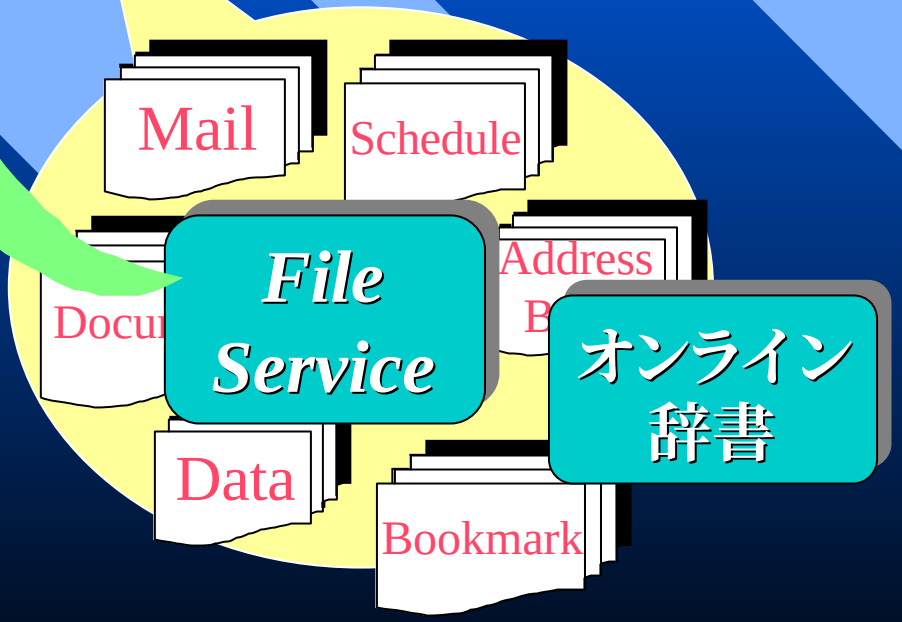
- 常に携帯型PCを持ち歩く
- さまざまなネットワークに接続して利用



携帯するサービス

移動先でも自分の環境を





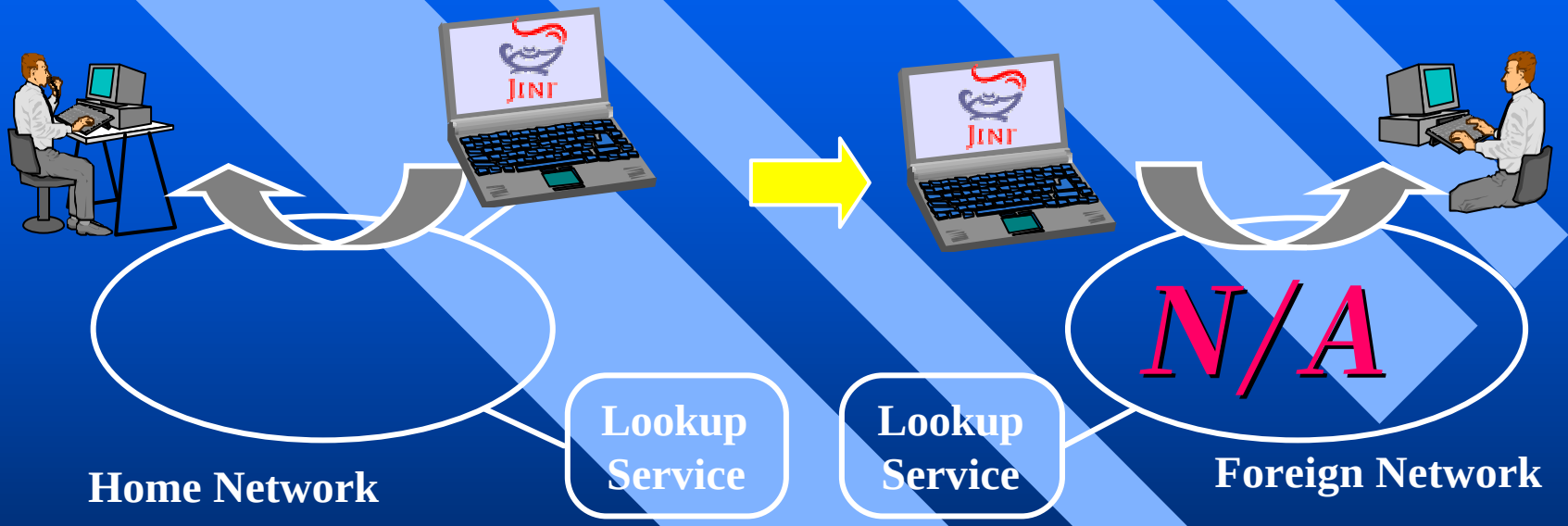
Key Points

- 接続先のネットワークへのアダプテーション
- 他の機器からのサービス発見・利用
  - **Jini Technology**の活用

Portable Jini Services

- モバイル機上でJini Serviceが稼動
  - ユーザとともに移動、移動先のネットワークに接続
  - 個人用サービスの提供
- 外出先の情報機器は、Jini Client
  - モバイル機と連携

利用イメージ



- 残念ながら、移動先のネットワークでは、サービスが利用できない

利用パターンの特徴

- モバイル機・Jini Serviceはリブートせず
  - 稼動したままの状態での移動
  - サスペンド・リジュームの活用

↓

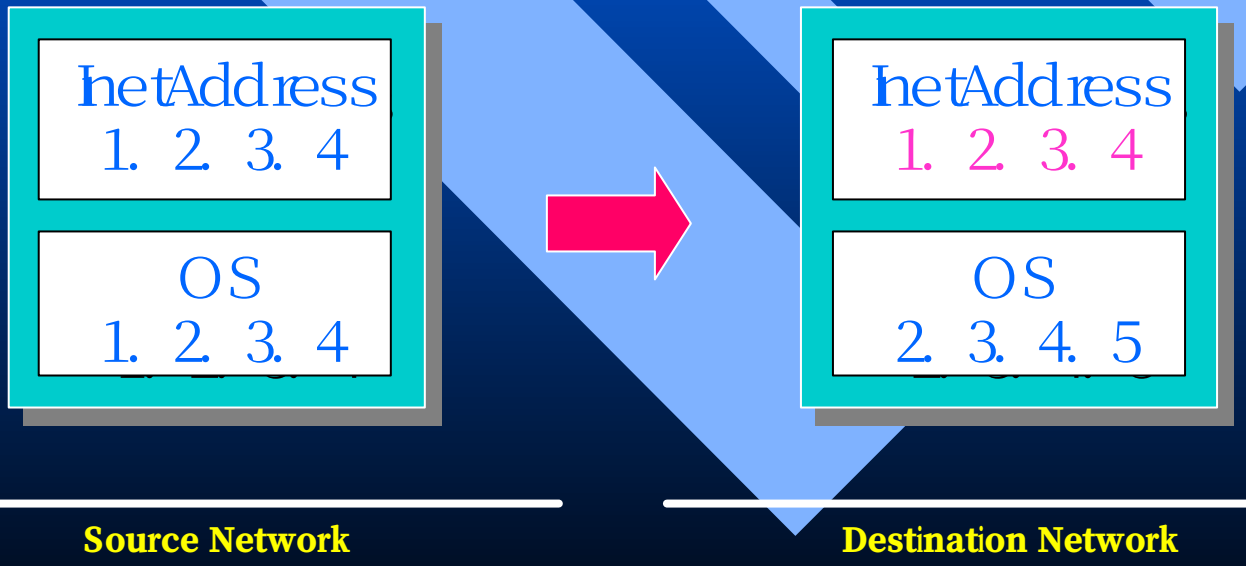
  - On-the-flyにIPアドレスが変更される

Jini適用上の問題点

- Java/Jiniは、IPアドレスの動的な変更に対応できない
  - IPアドレスの取得
  - リモート参照
  - コードベース

IPアドレスの取得

- 問題点:
  - IPアドレスを獲得するには java.net.InetAddress.getLocalHost()
  - 新 IPアドレス獲得後も、古いアドレスを返す

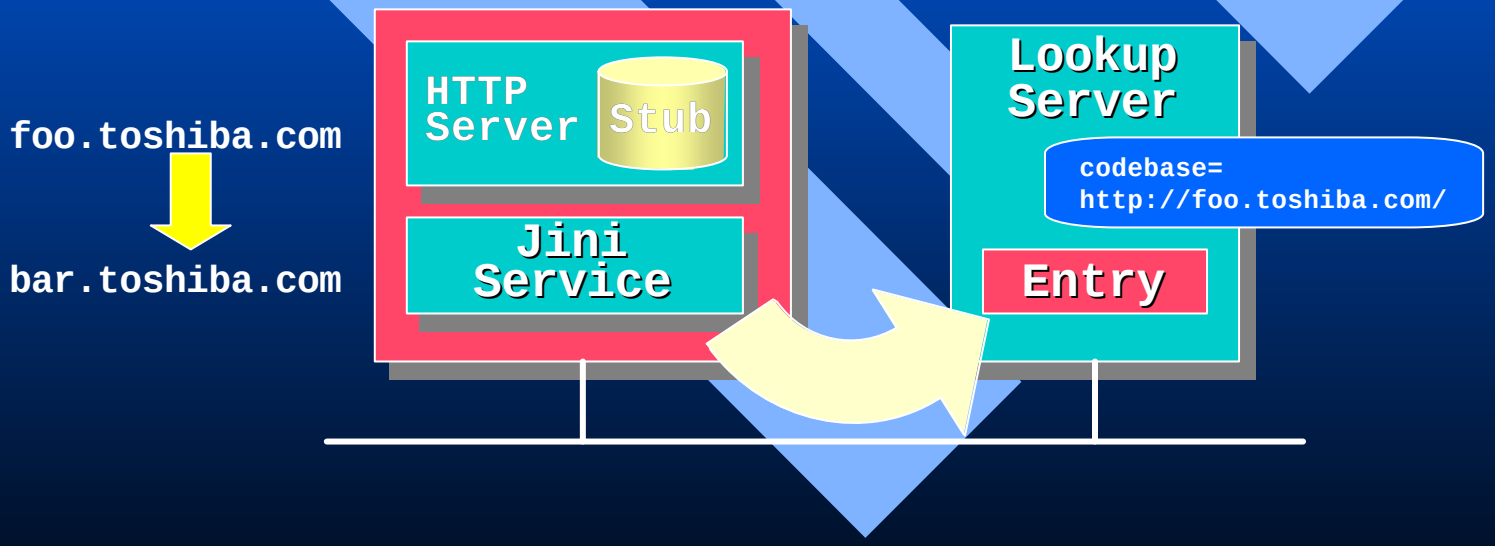


リモート参照

- 問題点:
  - ネットワーク移動のあと、古いIPアドレスを含んだリモート参照をLookupServerに登録してしまう

コードベース

- 問題点:
  - 移動前のアドレスを含んだコードベースがオブジェクトに付与されてしまう

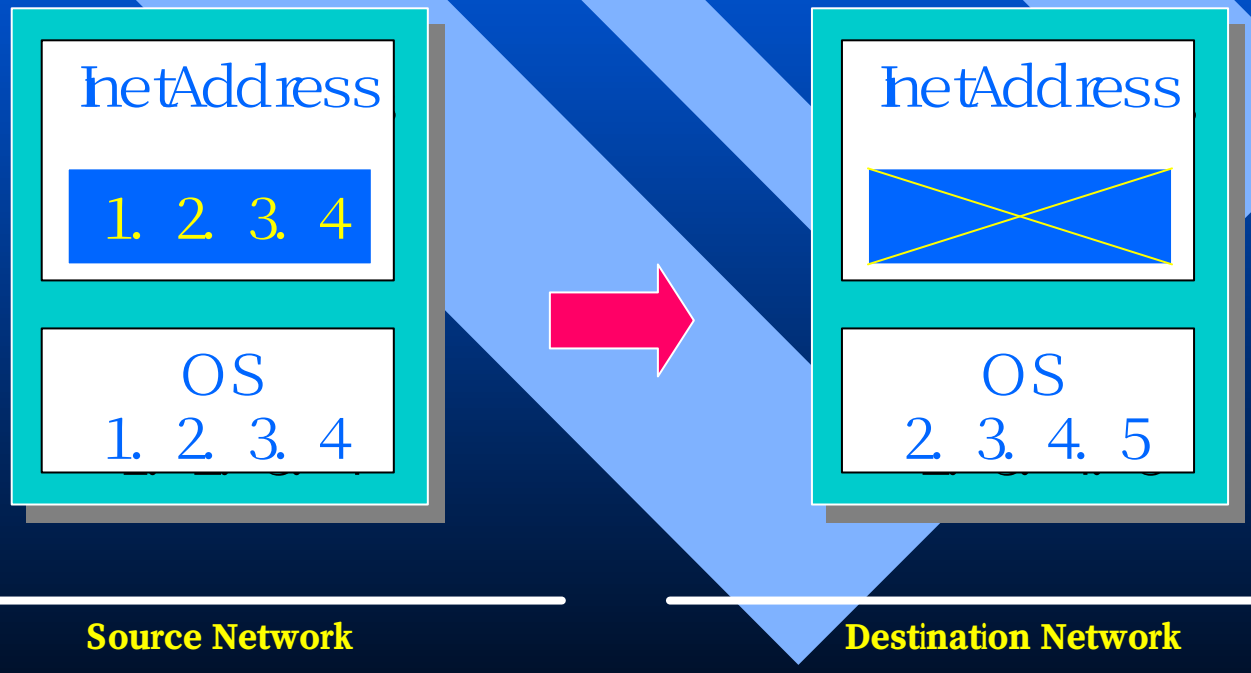


モーバイル拡張

- Portable Jini Servicesを実現する上で障害となる個所の洗い出しと、簡単な改良

IPアドレスの取得

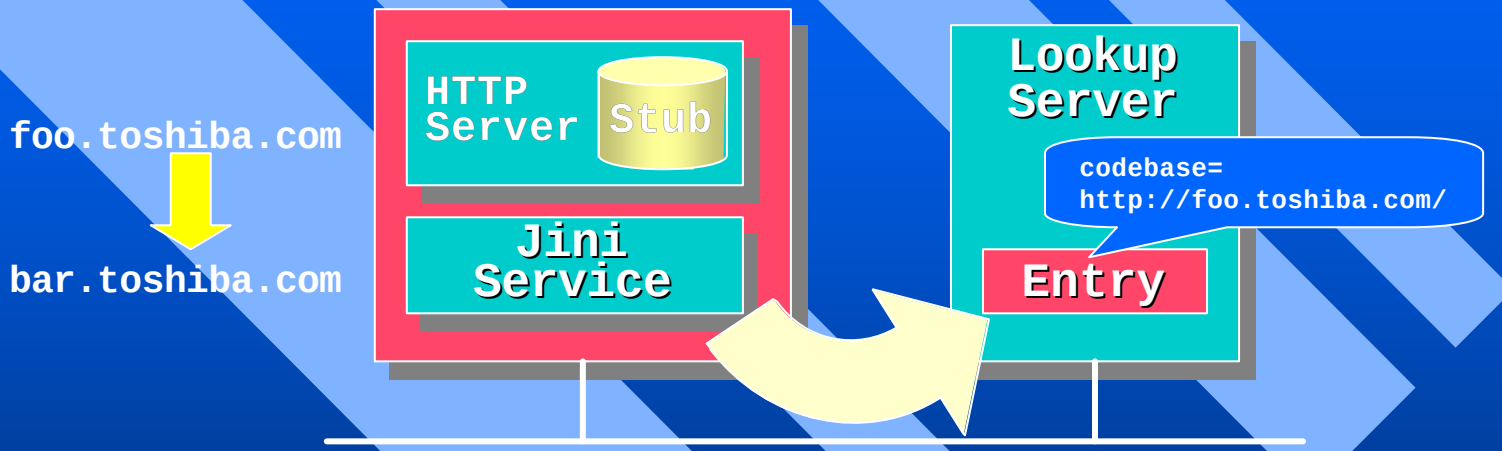
- 解決法:
  - java.net.InetAddress内のキャッシュを無効に



リモート参照

- 解決法:
  - 新しいアドレス獲得後に、すべてのlocalEndPointを更新するメソッド“refreshLocalHost()”を、sun.rmi.transport.tcp.TCPEndpointクラスに追加

コードベース

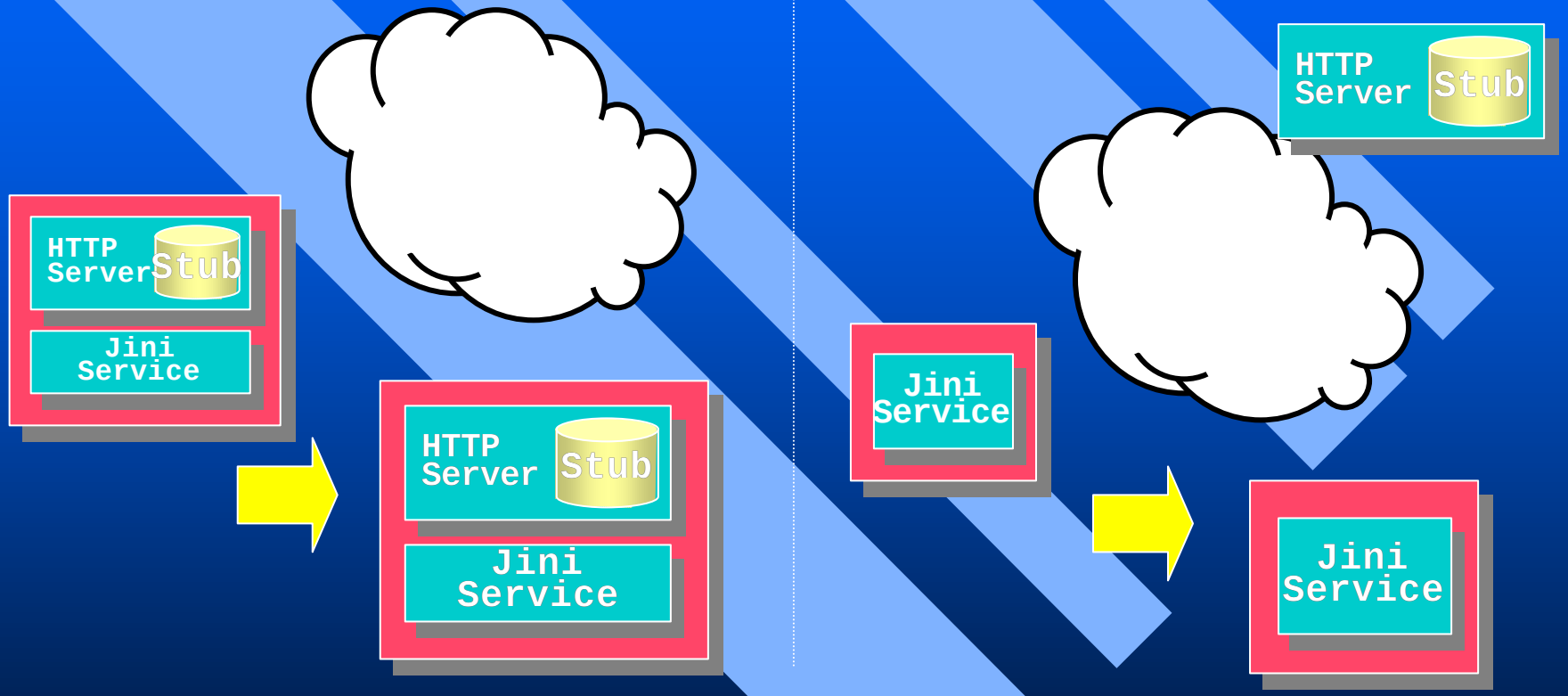


- 解決法:
  - 新しいアドレス獲得後にローカルコードベースを更新するメソッド“refreshCodebase()”をsun.rmi.server.LoaderHandlerクラスに追加

コードベース

コードベース更新が必要

コードベース更新が不要



- 両者が簡単に区別できるよう、mobileCodebaseプロパティを導入

モーバイル拡張の適用

- サーバが移動する場合
- クライアントが移動する場合
- PPP接続する場合

まとめ

- 現状：
  - 簡単なJini Serviceで、動作確認
  - Jini Community Meetingで、プロジェクト立ち上げのアナウンス

- 今後：
  - 他の問題点の洗い出し
  - Jini Communityでプロジェクト立ち上げ