P2Pアプリケーションにおける Symmetric NAT Traversal の研究

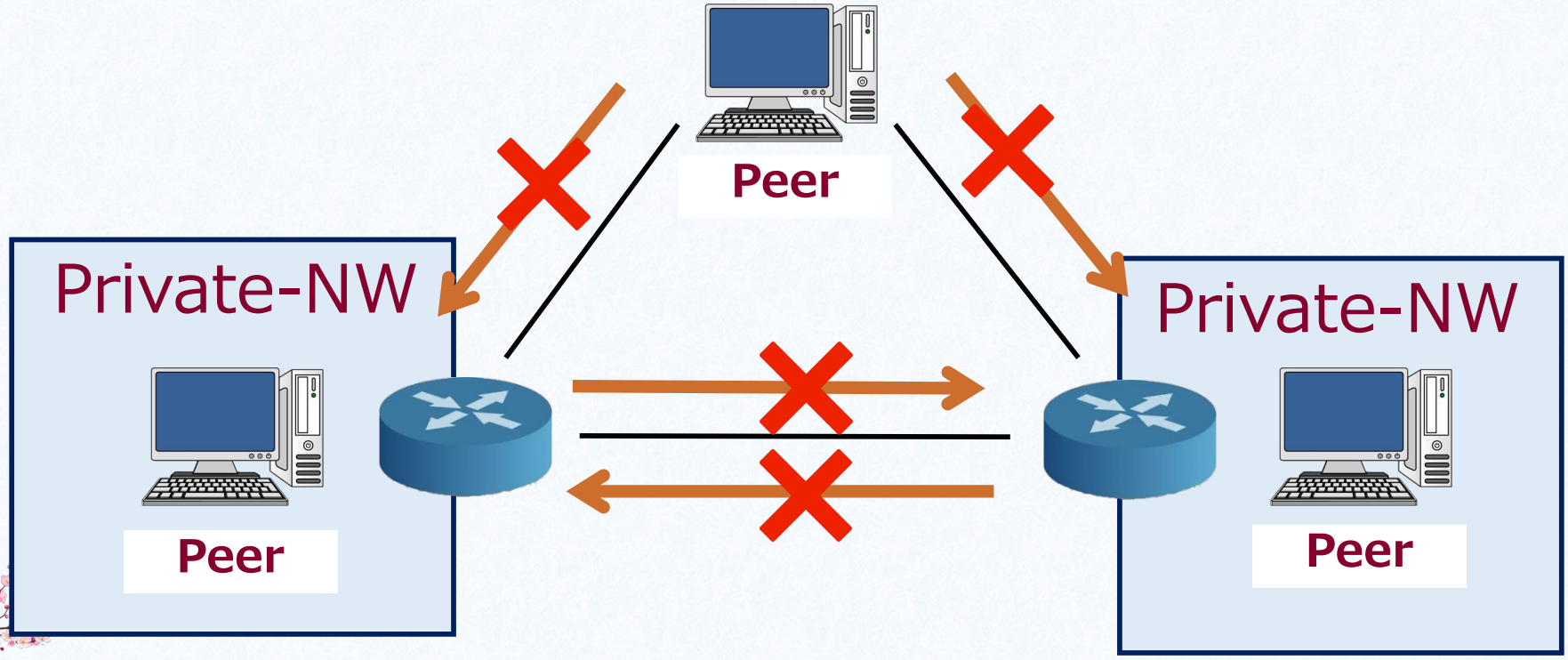
Yong Wang, Zhao Lu, Junzhong Gu

内藤研究室
K18039 B4 後藤 廉

2021.07.14 論文ゼミ

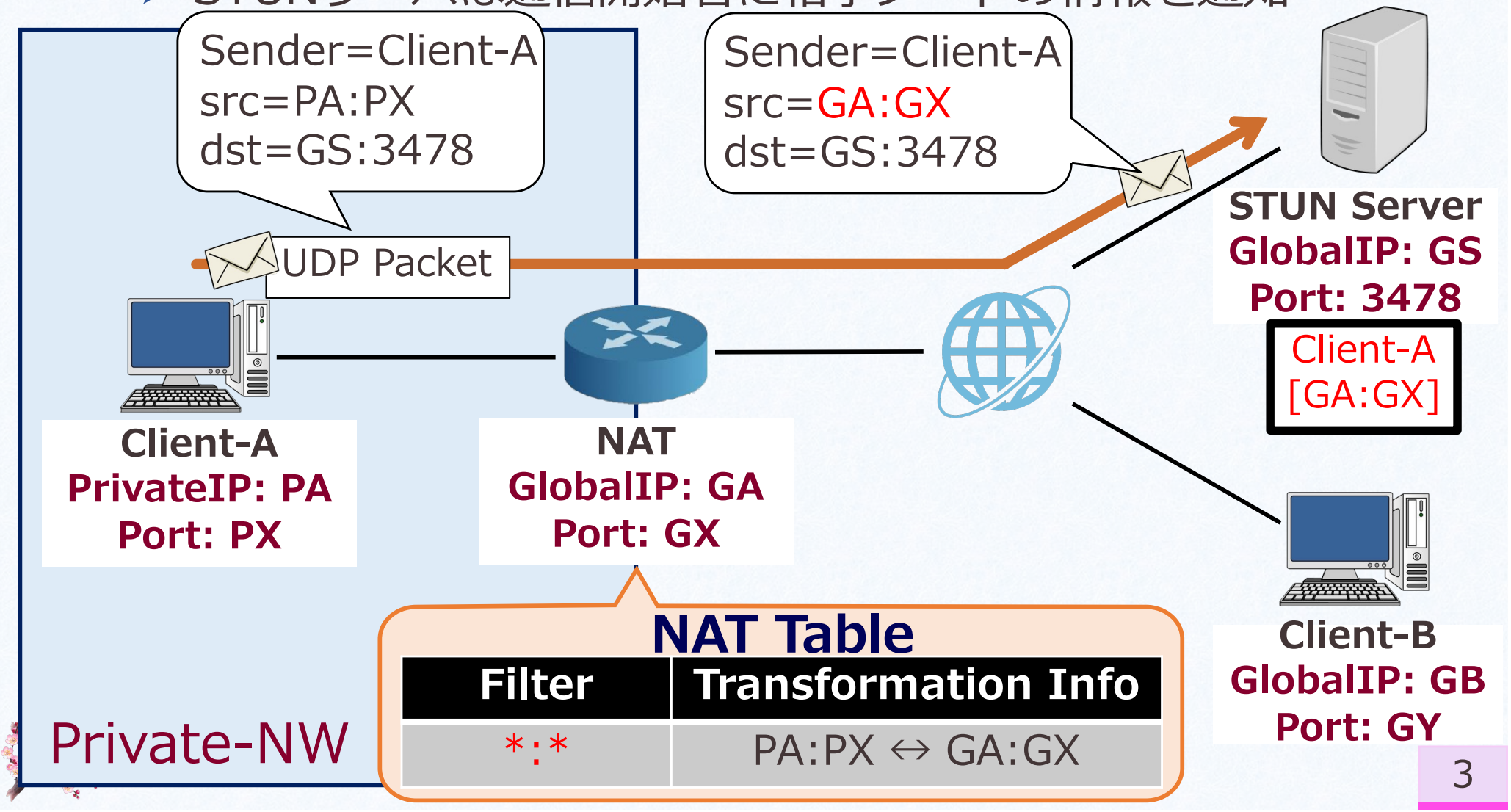
P2Pによるクライアント間通信

- P2P(Peer-to-Peer)通信：ピア間で直接送受信
  - 大量のデータを送信する様々なアプリケーションで利用
    - ✓ 例：）ファイル共有, VoIP, IM, ビデオ会議
  - 集中型サーバのコストを実質的に排除し帯域消費を抑制
- P2P通信の諸問題（NAT越え問題）
  - NATがピア間のデータストリームをブロック



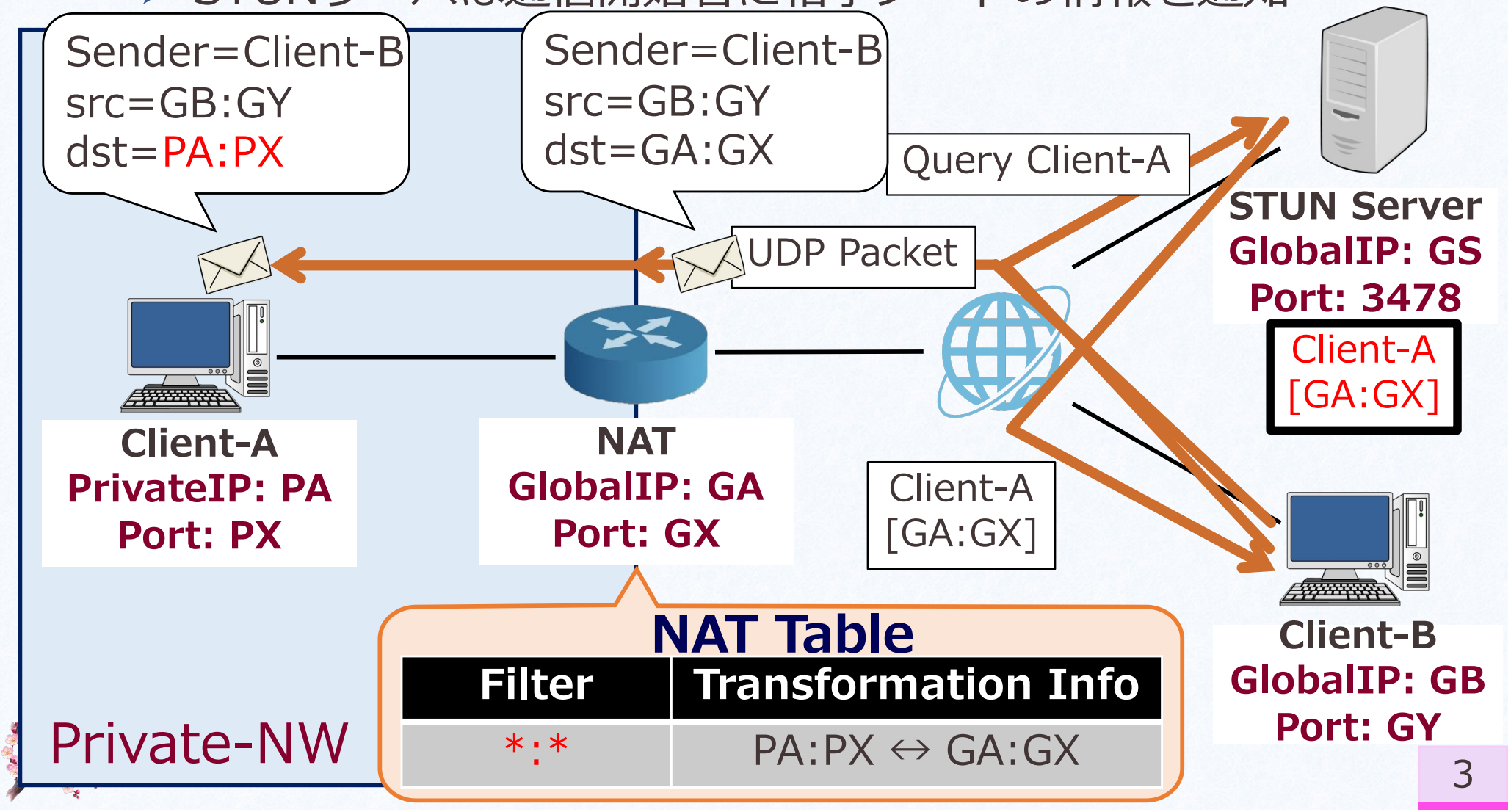
STUN(Session Traversal Utilities for NAT)

- STUNサーバを用いて変換情報を管理
  - **単一**のアドレス変換を生成してSTUNサーバに登録
  - STUNサーバは通信開始者に相手ノードの情報を通知



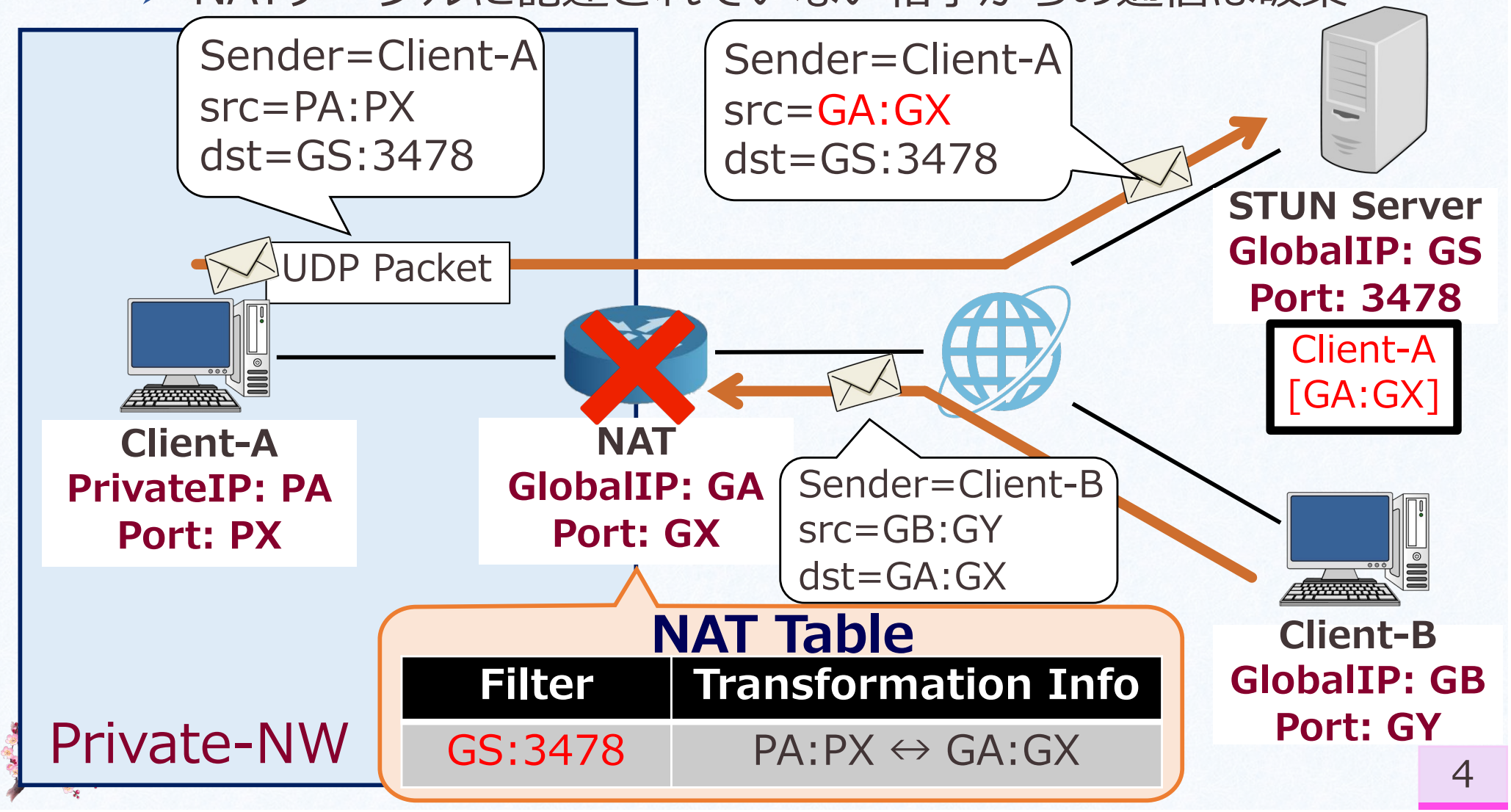
STUN(Session Traversal Utilities for NAT)

- STUNサーバを用いて変換情報を管理
  - **単一**のアドレス変換を生成してSTUNサーバに登録
  - STUNサーバは通信開始者に相手ノードの情報を通知



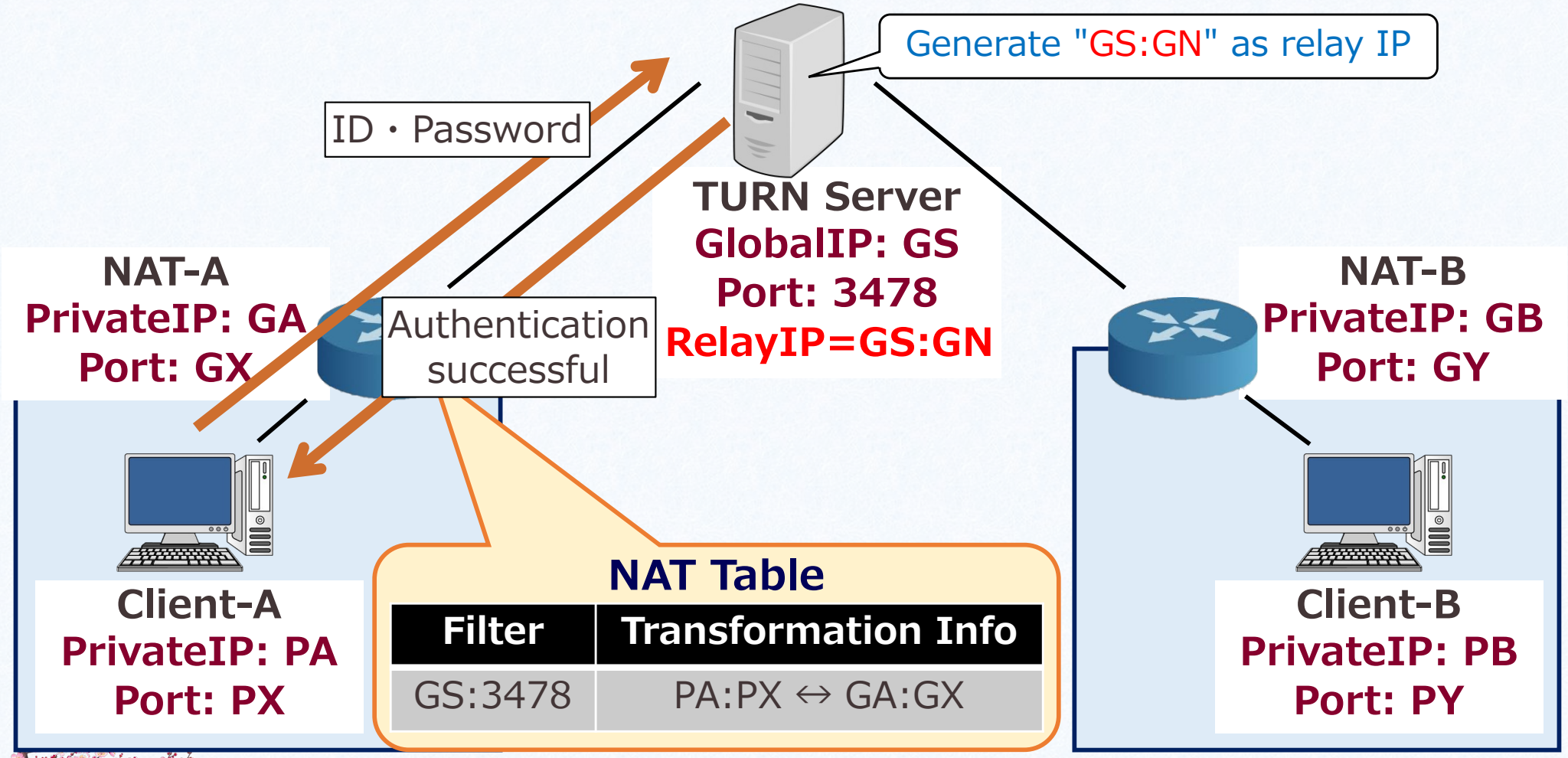
STUNの問題点

- Symmetric NAT環境下ではトラバーサル不可能
  - **宛先毎**にアドレス変換を生成
  - NATテーブルに記述されていない相手からの通信は破棄



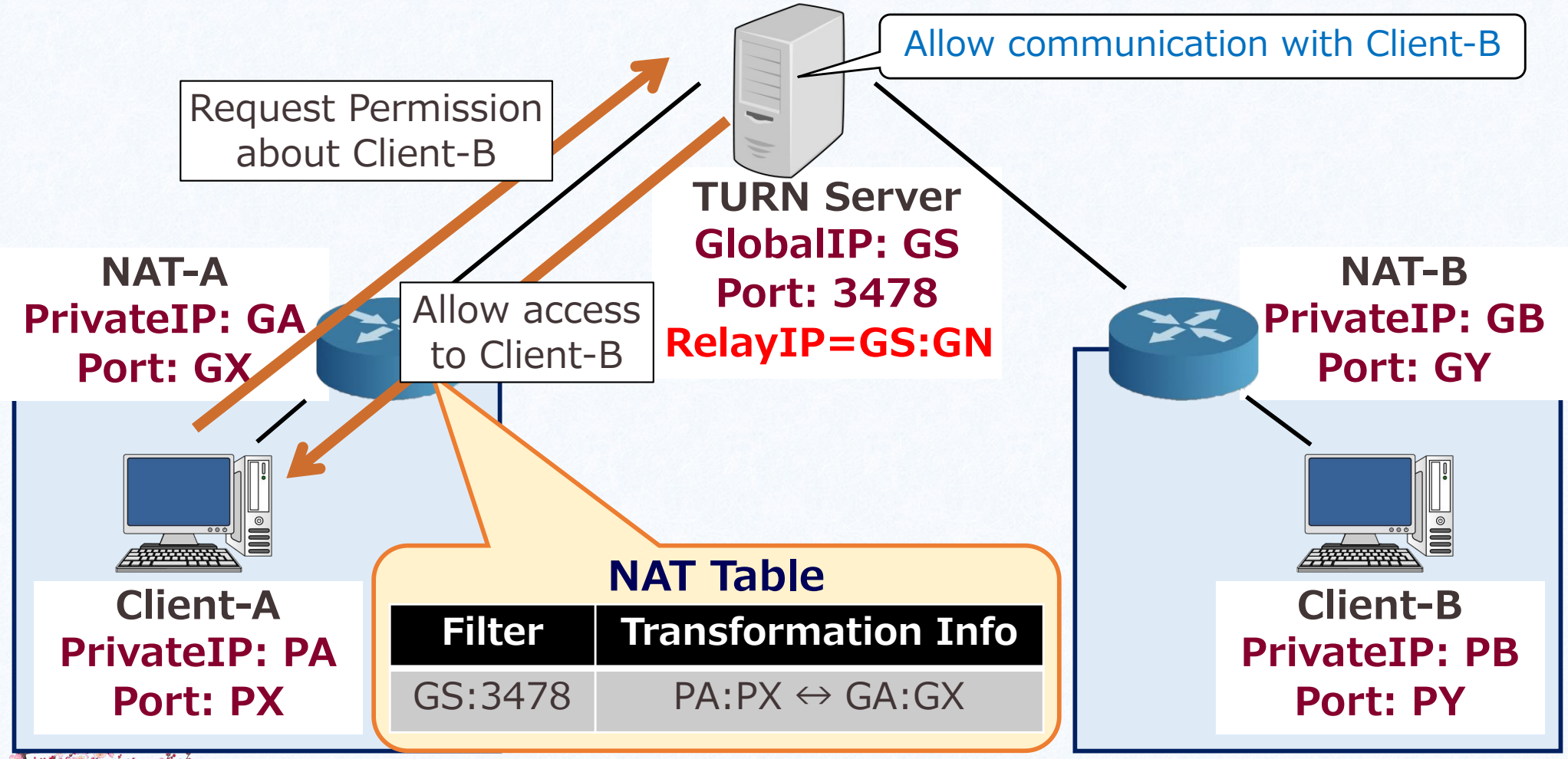
TURN(Traversal Using Relays around NAT)

- TURNサーバが両者間で通信を中継してトラバース
  - Symmetric NAT環境下でのデバイス間通信をサポート
  - 通信時は常にTURNサーバを介した送受信が必要



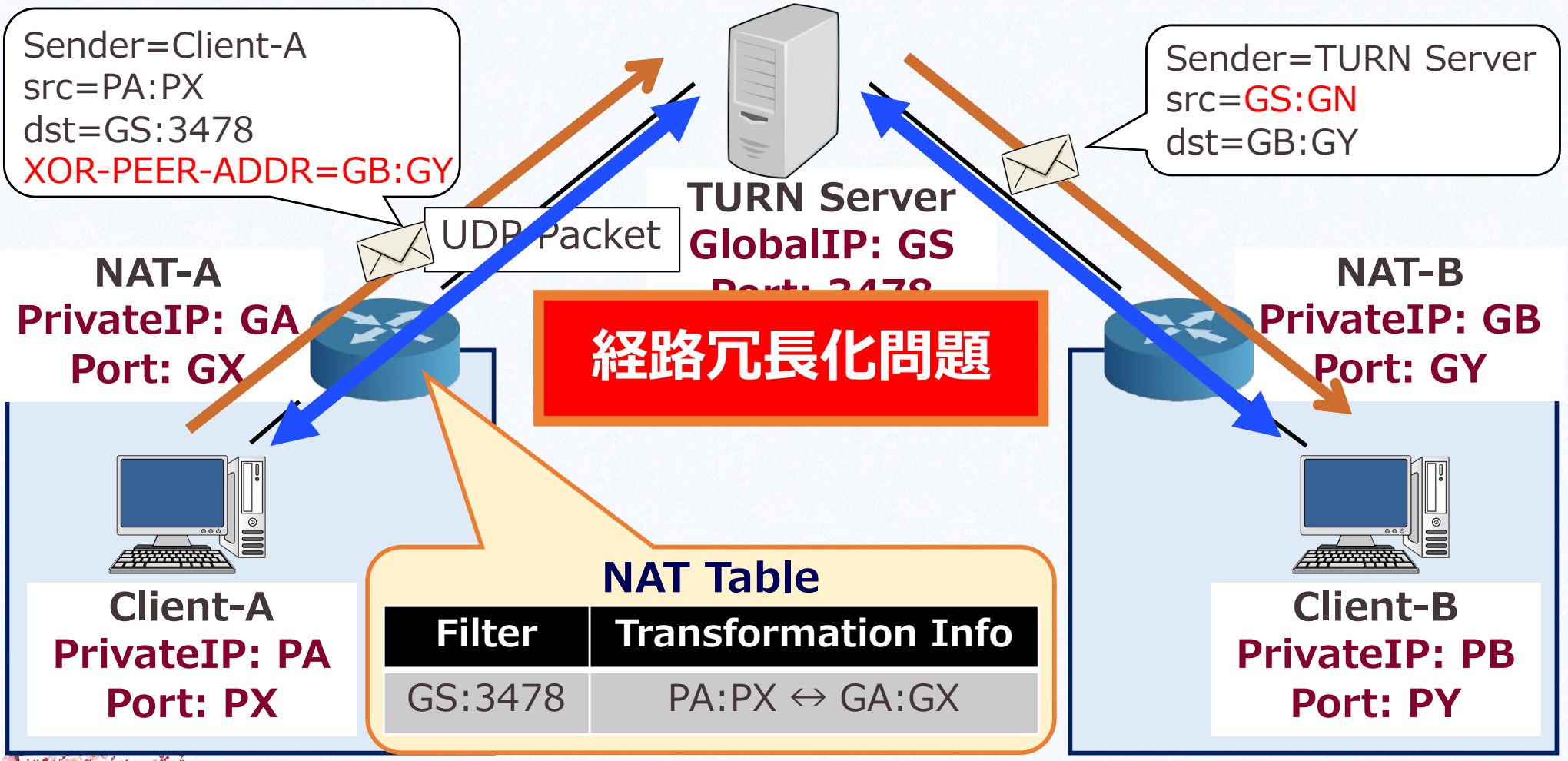
TURN(Traversal Using Relays around NAT)

- TURNサーバが両者間で通信を中継してトラバース
  - Symmetric NAT環境下でのデバイス間通信をサポート
  - 通信時は常にTURNサーバを介した送受信が必要



TURN(Traversal Using Relays around NAT)

- TURNサーバが両者間で通信を中継してトラバース
  - Symmetric NAT環境下でのデバイス間通信をサポート
  - 通信時は常にTURNサーバを介した送受信が必要



分散型P2Pテクノロジの課題

P2P通信は集中型サーバのコストを実質的に排除し，
ネットワーク帯域幅の消費を抑えることが可能

しかし，多くのクライアントはLAN内に存在するため
NATがデータストリームをブロック

STUN及びTURNを用いたNAT Traversalには
問題が介在し，効果的に対処することが不可能

これらの課題を解決する新たな NAT Traversal が必要

提案システム

STUNが有効でない状況においてSymmetric NATを
トラバースしてP2Pで直接通信を実現する **PS-STUN** の提案

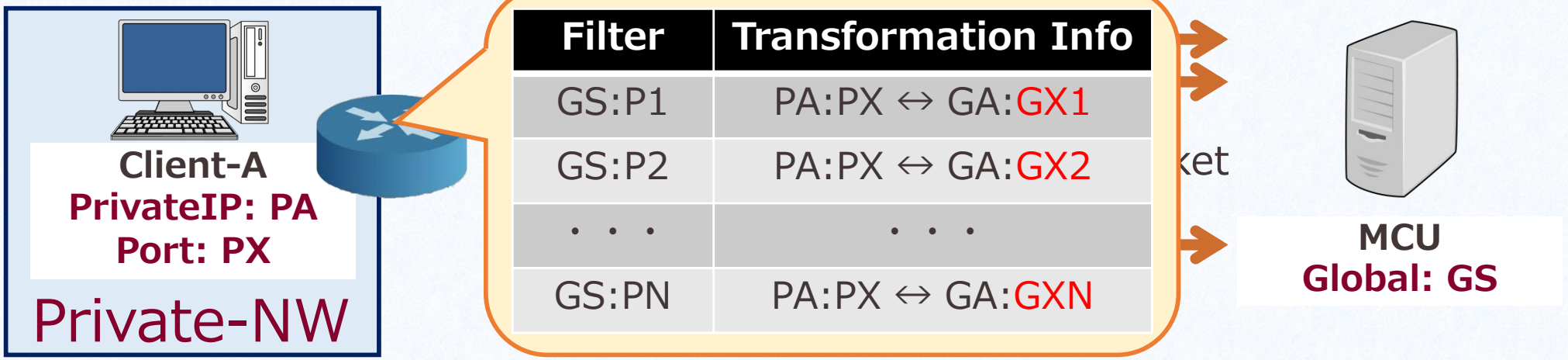
- “予測”と”走査”に基づくNAT Traversal
  - **P**redicting：NATが次に使用するポート番号を予測
  - **S**canning：ポート番号を段階的に変化させてパケットを送信
- 中央のサーバを介さずにSymmetric NATをトラバース
  - リレーによって発生していた経路冗長化問題を排除
  - 最終的にUDP Hole PunchingによりP2Pで直接通信を実現

UDP Hole Punching

内部から通信を行なった後, 一定期間のみ外部ネットワークからの
通信を許可するNATの性質を利用して通信用の穴を開ける。

Symmetric NATの分類

- NATによる変換ポートの生成規則から分類

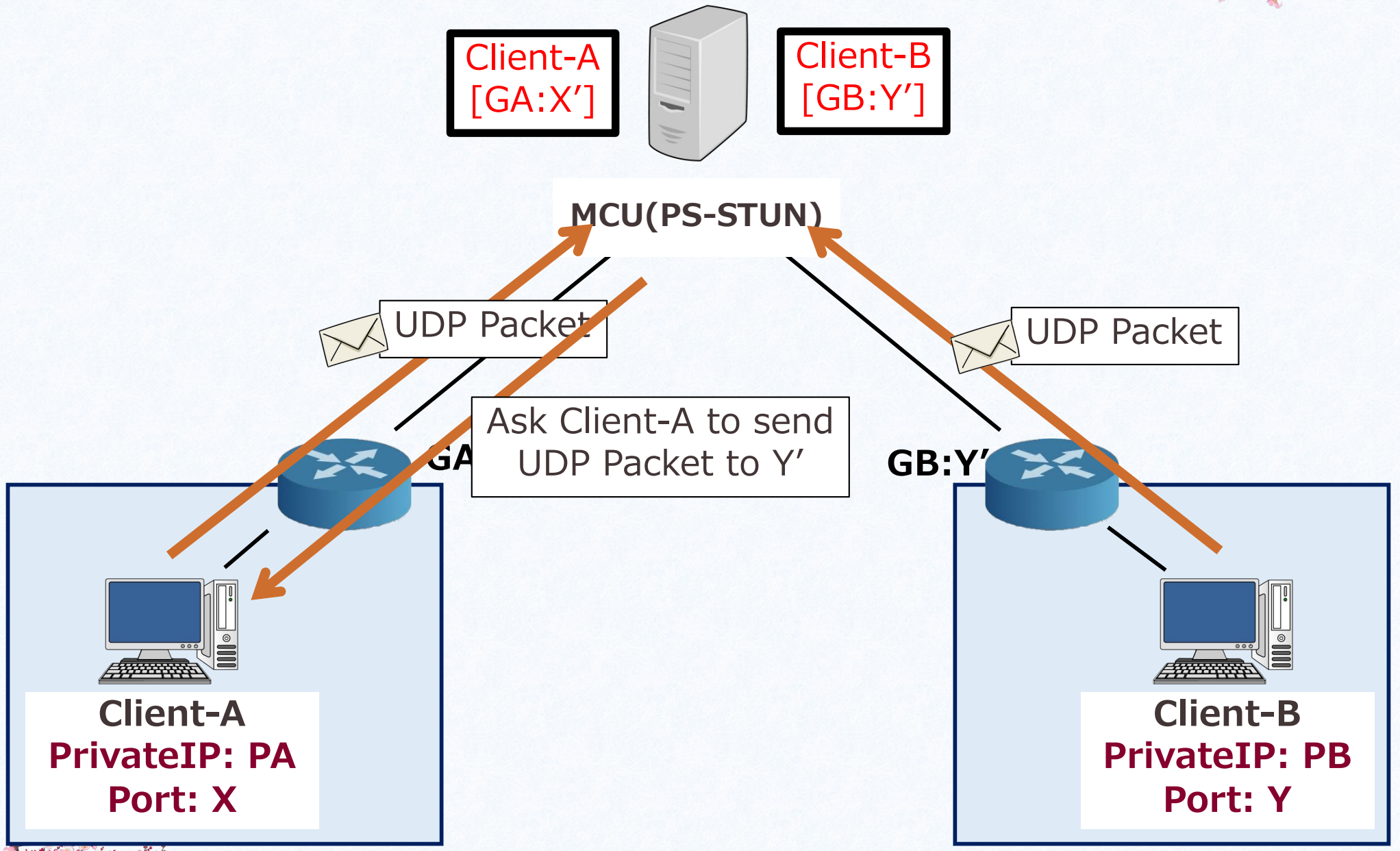


P(Progressive)型 Symmetric NAT：ポート番号が連続的
R(Random)型 Symmetric NAT：ポート番号がランダム

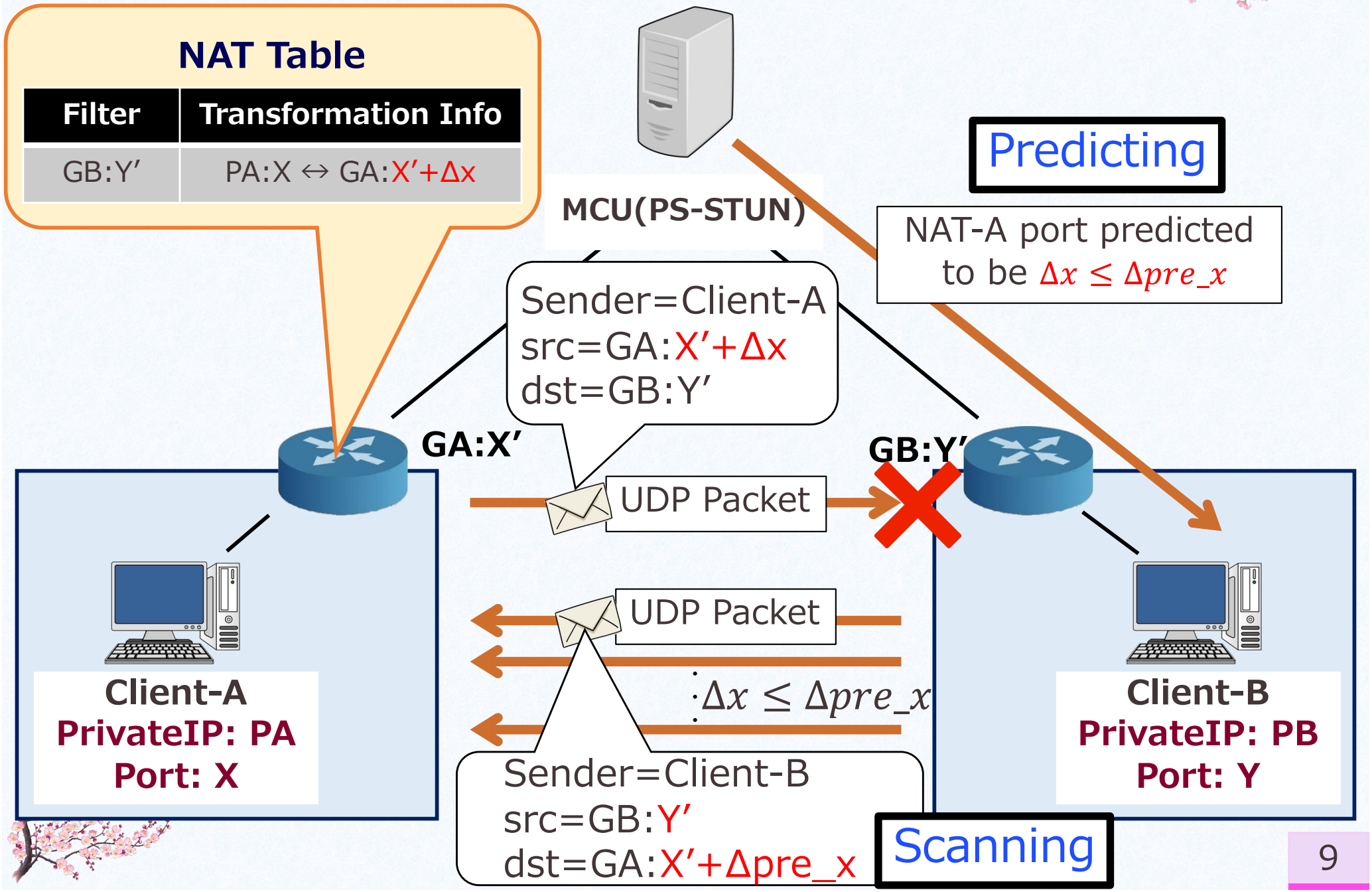
- PS-STUNによるNAT Traversalが実現されるケース

Class	NAT-A	NAT-B	Traversal Probability
A	P-type Symmetric	IP or Port Restricted	Very good
B	R-type Symmetric	IP or Port Restricted	Good
C	P-type Symmetric	P-type Symmetric	Very good

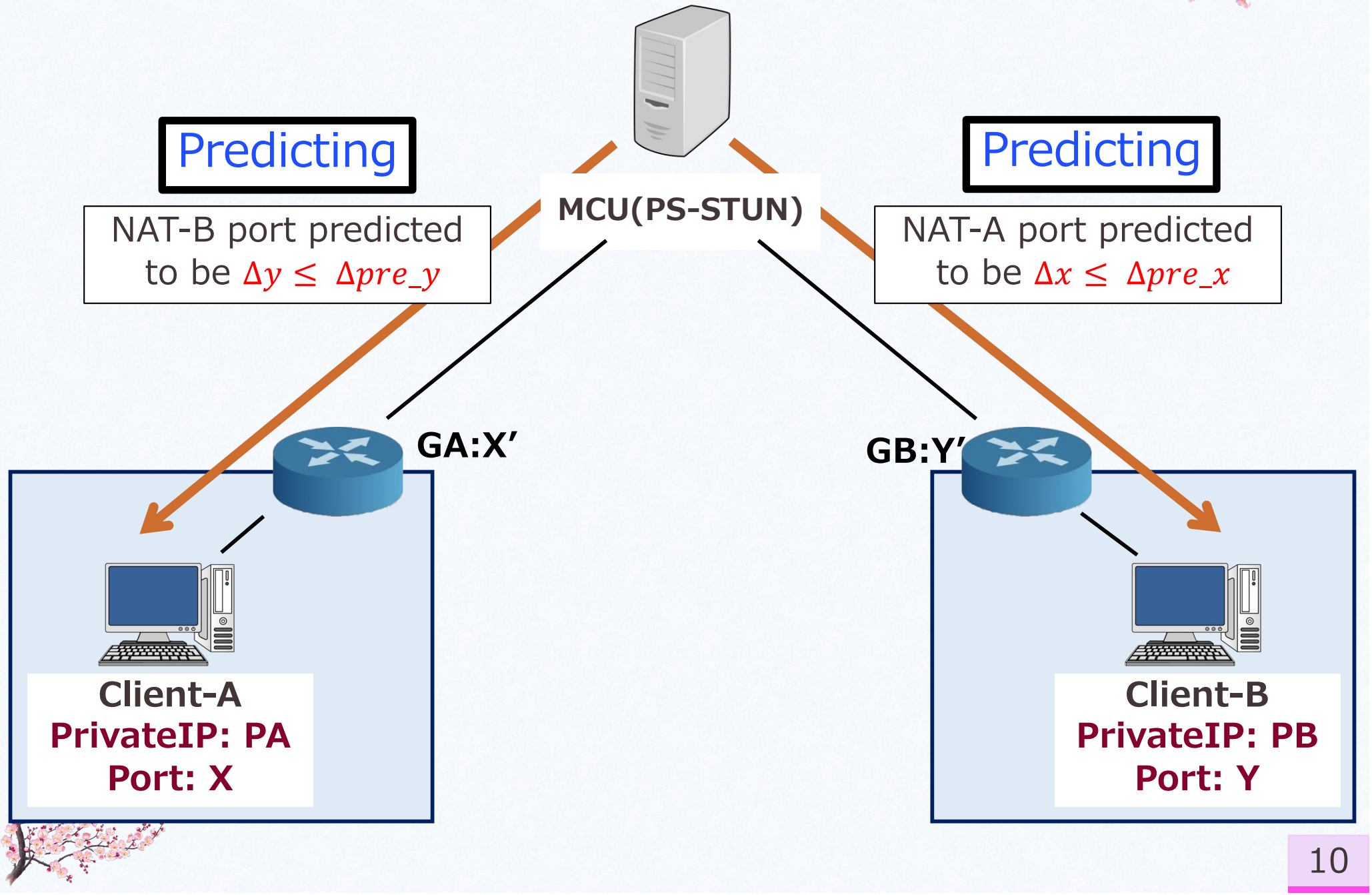
クラスA・クラスBにおけるPS-STUN



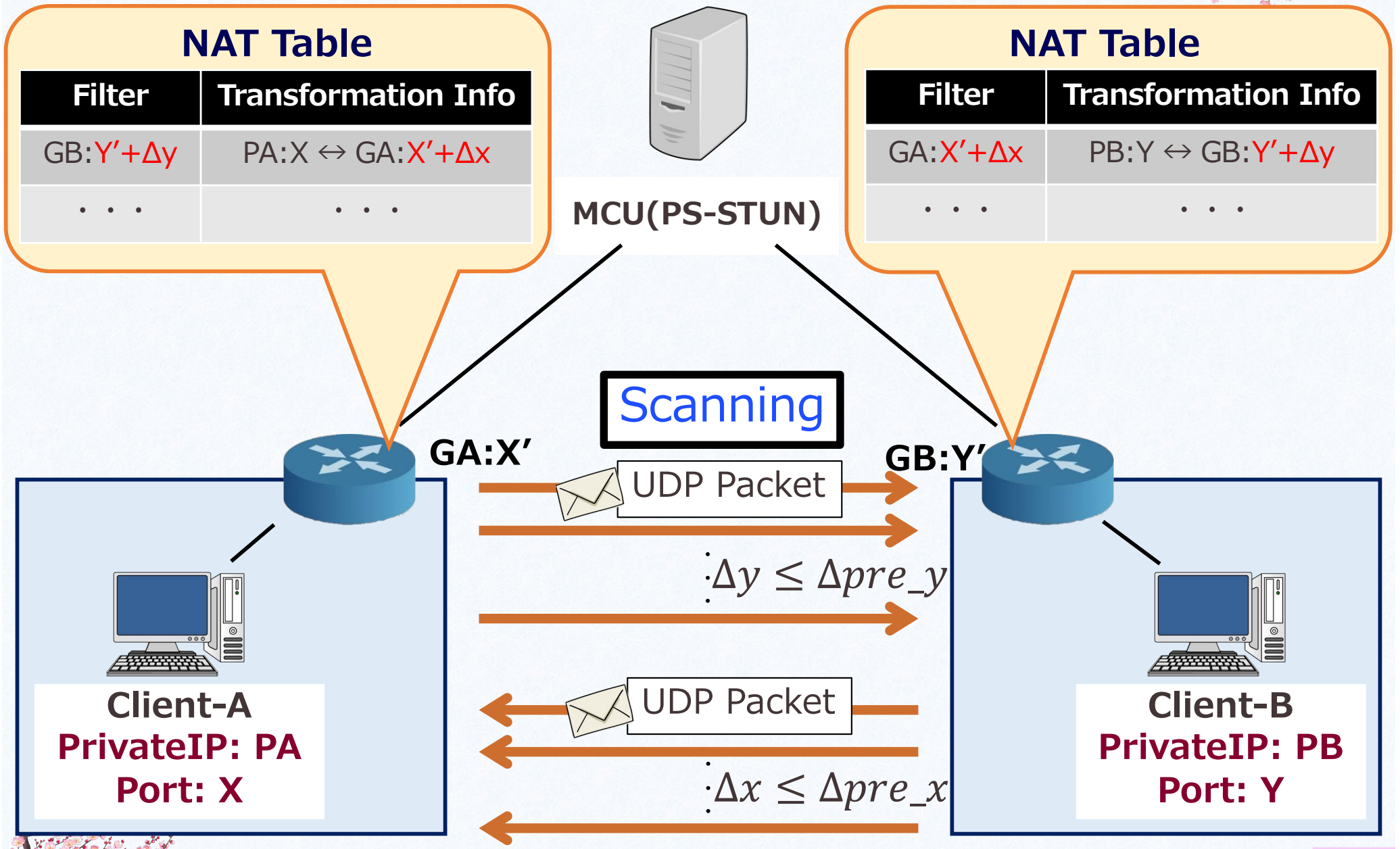
クラスA・クラスBにおけるPS-STUN



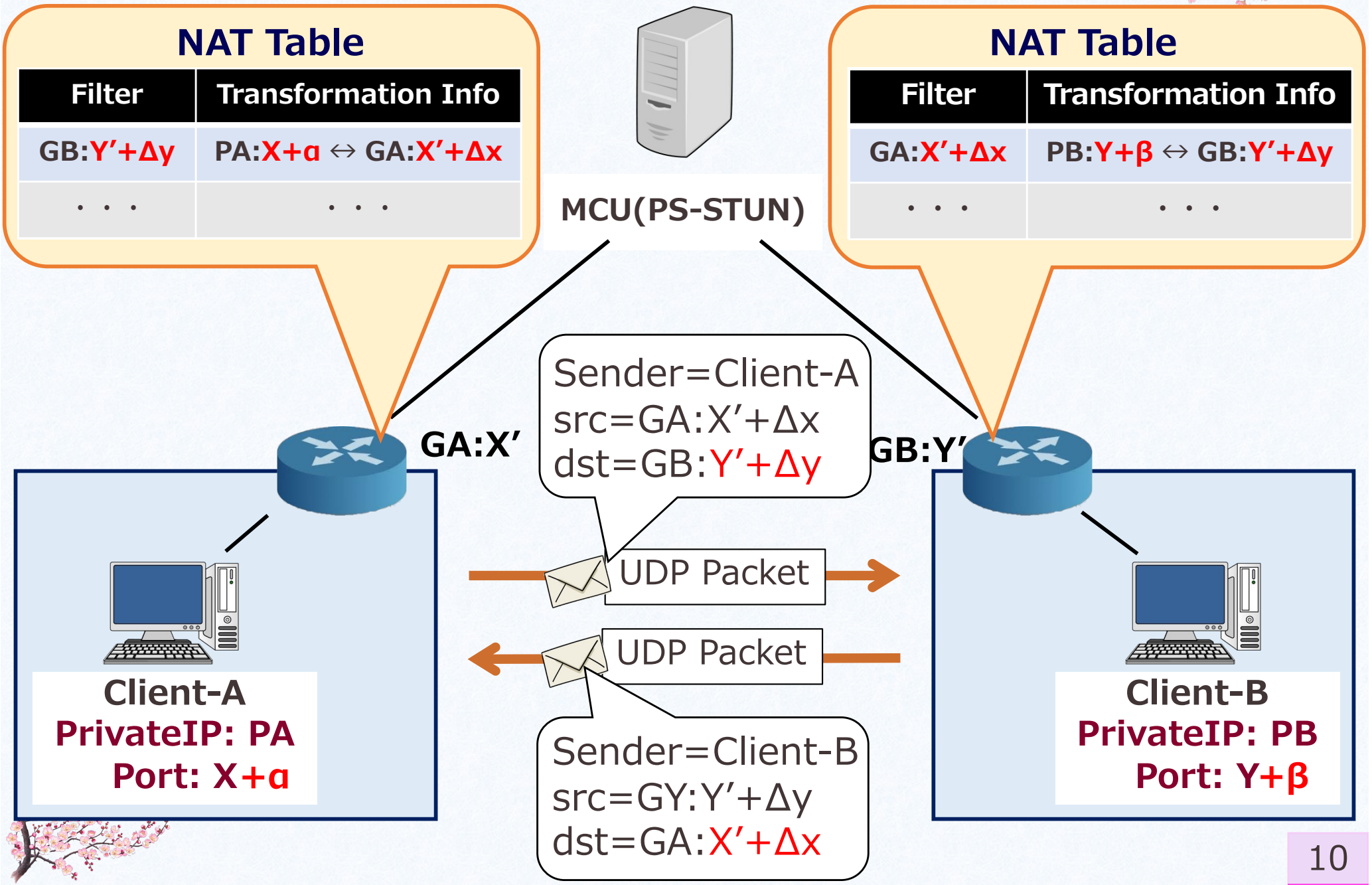
クラスCにおけるPS-STUN



クラスCにおけるPS-STUN



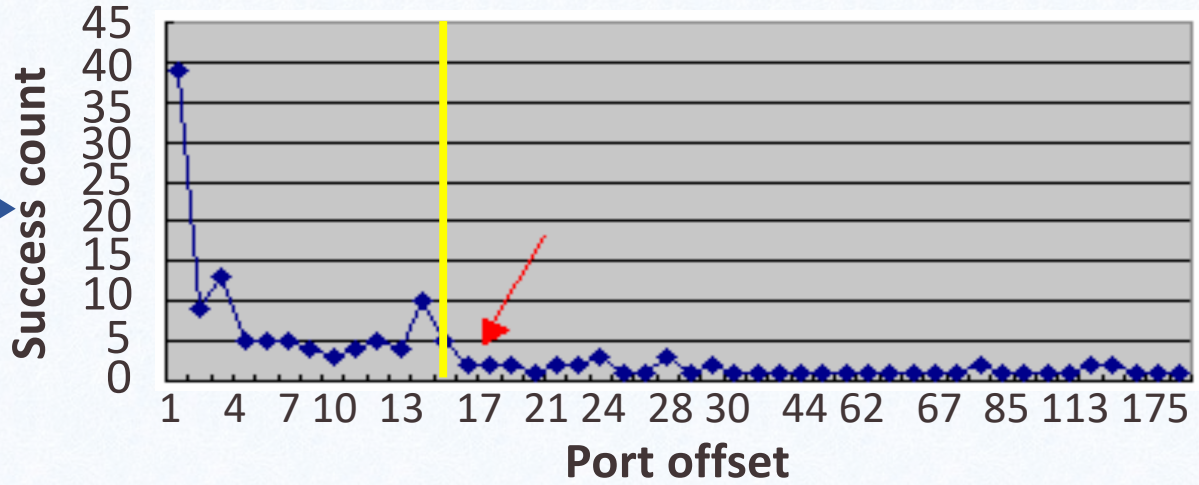
クラスCにおけるPS-STUN



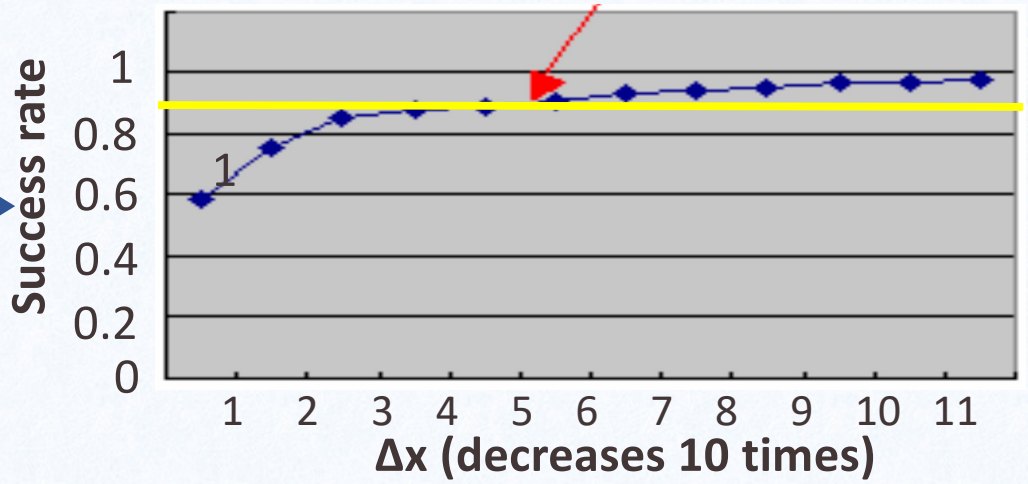
PS-STUNの評価

■ クラスA：P型Symmetric NAT ⇔ Restricted NAT

オフセットと
スキャンニング成功回数の関係



NAT Traversalの成功率

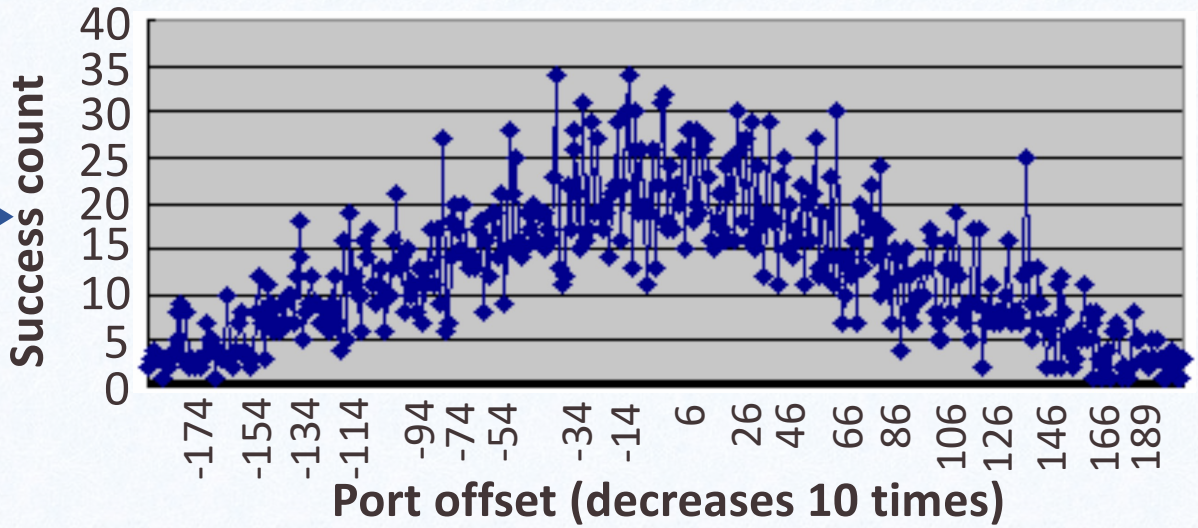


オフセットが15以下と小さく，比較的簡単に
スキャンニングが成功し，NAT Traversalを実現可能

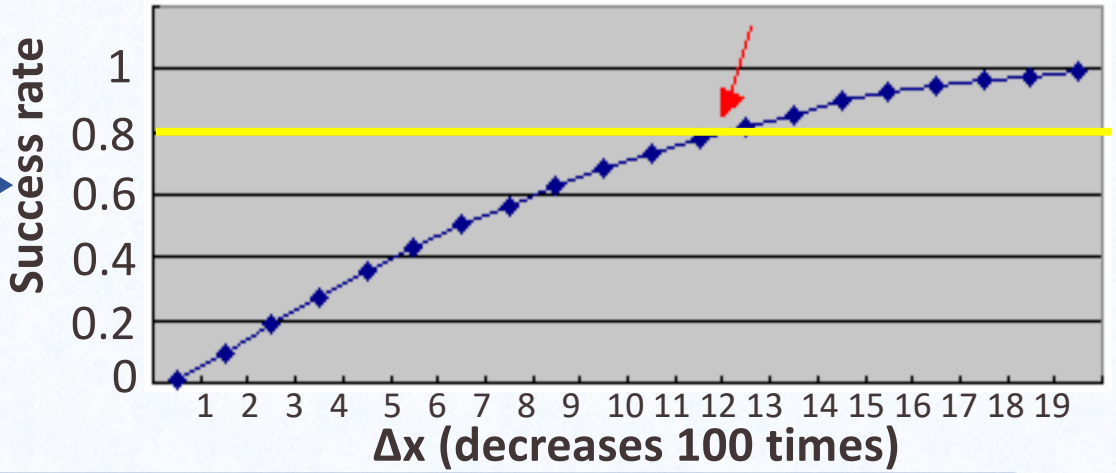
PS-STUNの評価

- クラスB：R型Symmetric NAT ⇔ Restricted NAT

NAT-Aのマッピングポートに
連続して割り当てられた2つの
ポート間のオフセット分布



NAT Traversalの成功率



ヒット率向上のためにはより多くの
スキャンニングを実行することが効果的

PS-STUNの評価

- クラスC：P型Symmetric NAT ⇔ P型Symmetric NAT
  - すべてのポートに対して1回のみスキャンした場合

Workload \ Δx	5	10	20	30	60
Busy	19	26	34	44	46
Idle	37	46	45	50	50

送信間隔：500ms

NATのワークロードがビジー状態にある時
予測される Δx を増大することで対処可能

  - すべてのポートに対して2回スキャンした場合

Workload \ Δx	5	10	20	30	60
Busy	18	28	44	47	47
Idle	38	48	48	50	50

送信間隔 1回目：100ms

送信間隔 2回目：1000ms

オフセット100以上で2つの Δx を予測して
スキャンした場合，トラバーサル成功率がほぼ100%

PS-STUNの改善

- P型Symmetric NATにおけるΔxの値の予測
  - Δxの最小値はNATの現在の最大作業負荷Wをテストして推定

$$W = MAX\left(X^{i+1} - X^{i}\right) / T \ (i > 0, i < test\ times)$$

X^i ：i 番目のパケットに対してNATにより割り当てられたポート番号

T ：クライアントからMCUへのUDPパケットの送信間隔

  - 現状のMCUでは $\Delta x \geq W$ と推定

$\Delta x = W + \alpha$（$\Delta\alpha \geq 0$：誤差許容係数）
➡ テストの不正確さやNATの作業負荷のバーストを改善

- クライアントにおけるスキャンパケットの送信速度
  - 高速：UDPパケットの一部がルータで廃棄される可能性を有す
  - 低速：$X' + \Delta x$ を他のクライアントに使用される可能性を有す

NATのワークロードを監視し，パケットの送信間隔を変動させるようクライアントに依頼

まとめ

STUNはSymmetric NATをトラバースすることが
不可能であり, TURNによる転送処理は経路冗長化問題が介在

Symmetric NATの特性に基き新たな
トラバーサルソリューションを提供する **PS-STUN** を提案

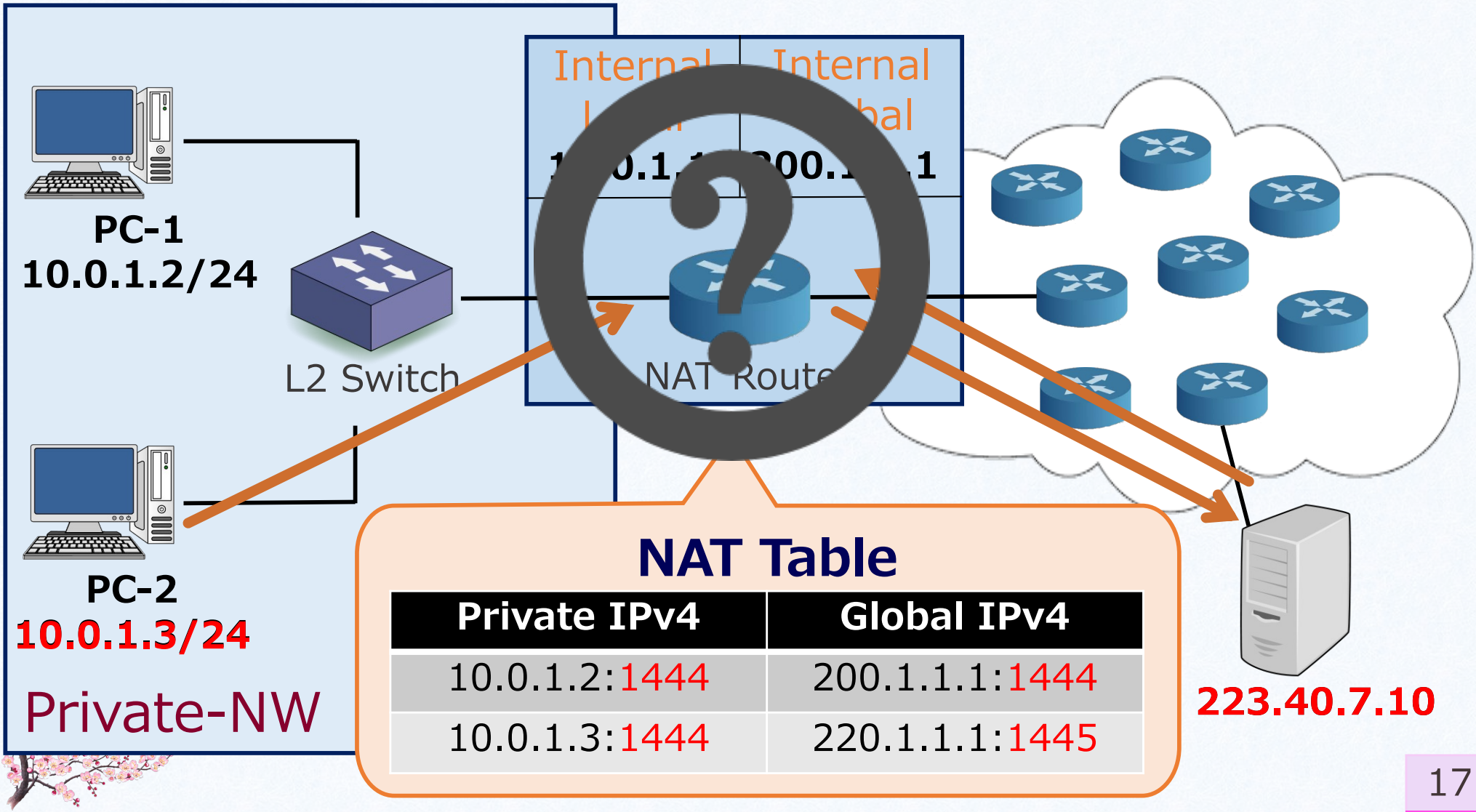
検証より, 多くの状況でクライアント間の直接接続を
実現することが可能であると証明

今後, NAT Traversal能力とシステム性能を
大幅に向上させることが見込める

以下，参考スライド

NAT (Network Address Translation)

- グローバルIPとプライベートIPを変換
  - NAT越え問題が介在

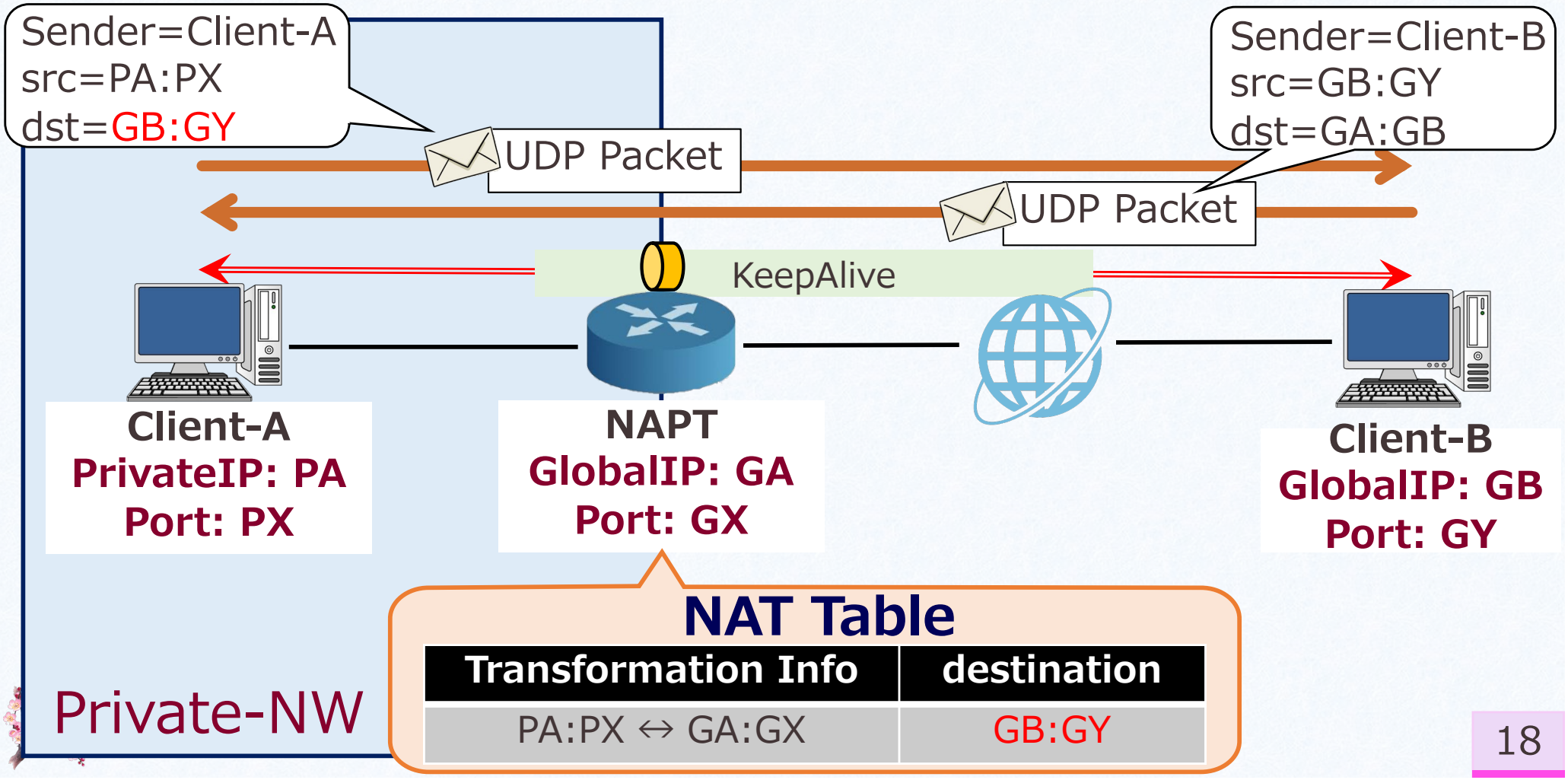


NAT Table

Private IPv4	Global IPv4
10.0.1.2:1444	200.1.1.1:1444
10.0.1.3:1444	220.1.1.1:1445

UDP Hole Punching

- UDP Hole Punchingは代表的なNAT Traversal
  - 内部から通信を行なった後, 一定期間のみ外部ネットワークからの通信を許可するNATの性質を利用
  - keep-aliveパケットを併用することによりコネクションを維持



UDP Hole Punchingの問題点

- 両デバイスがNAT配下に存在する場合トラバースは不可能
  - NATによって生成された変換情報を互いに知ることが困難

UDP Hole Punching のみで
NAT Traversalを実現するのは非現実的

↓

両デバイスの間に変換情報を記録するサーバが必要

↓

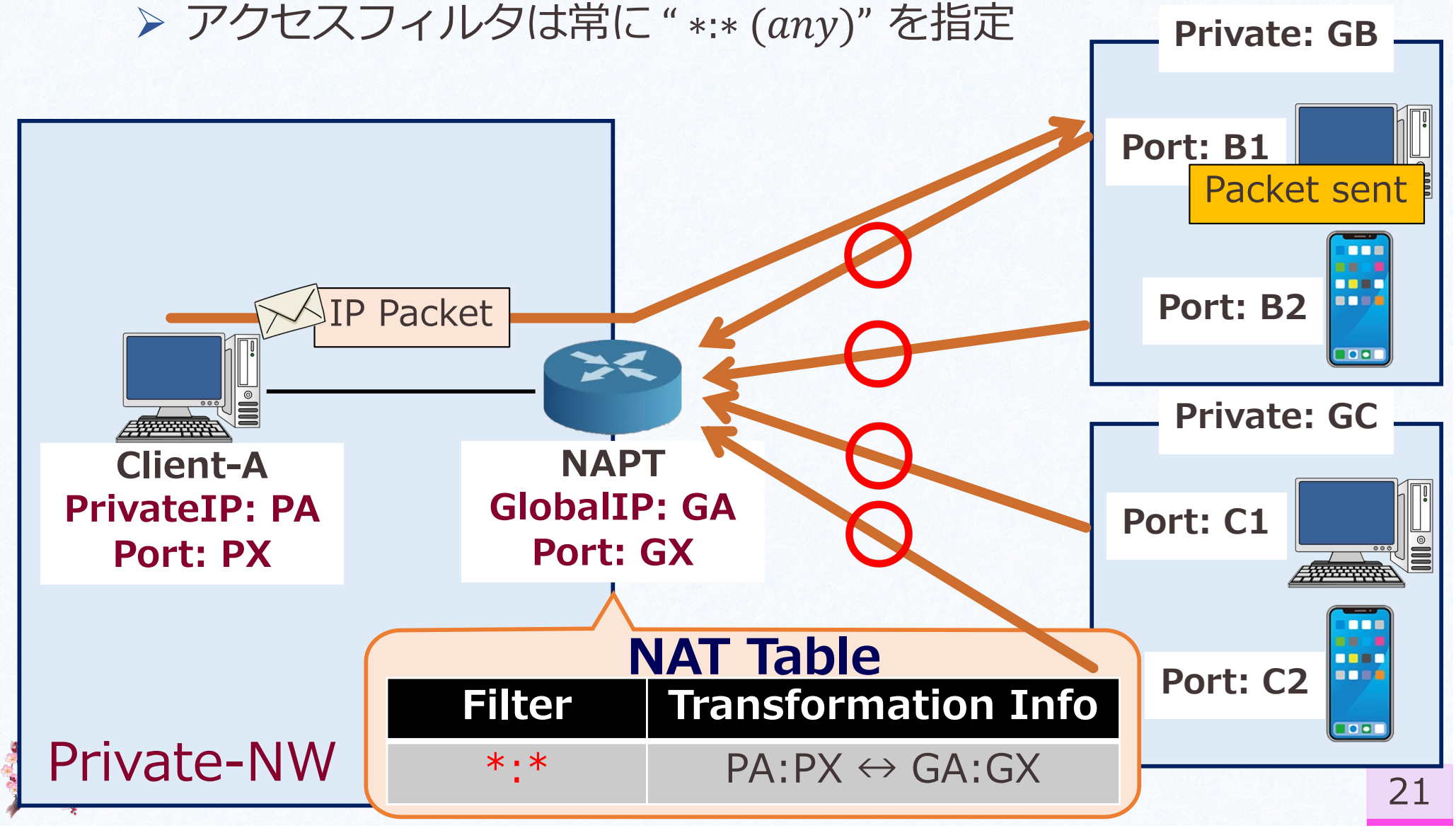
STUN (Session Traversal Utilities for NAT) の登場

NATの分類

- マッピングとフィルタリングの特性より4種類に分類
  - マッピング特性
    - ✓ NATテーブルに保持されるエントリの変換規則
  - フィルタリング特性
    - ✓ NATの外部から内部に充てられたパケットのフィルタリング規則
- Cone型NAT：**単一の**エントリを作成
  - **Full Cone NAT**
    - ✓ 一度も送信したことのないWAN側からのパケットも受信可能
  - **Restricted Cone NAT**
    - ✓ 一度送信したことのある “IP” からのパケットであれば受信可能
  - **Port Restricted Cone NAT**
    - ✓ 一度送信したことのある “IP及びPort” からのパケットであれば受信可能
- Symmetric型NAT：**宛先毎に**エントリを作成
  - **Symmetric NAT**
    - ✓ マッピングが存在する宛先デバイスからのパケットであれば受信可能

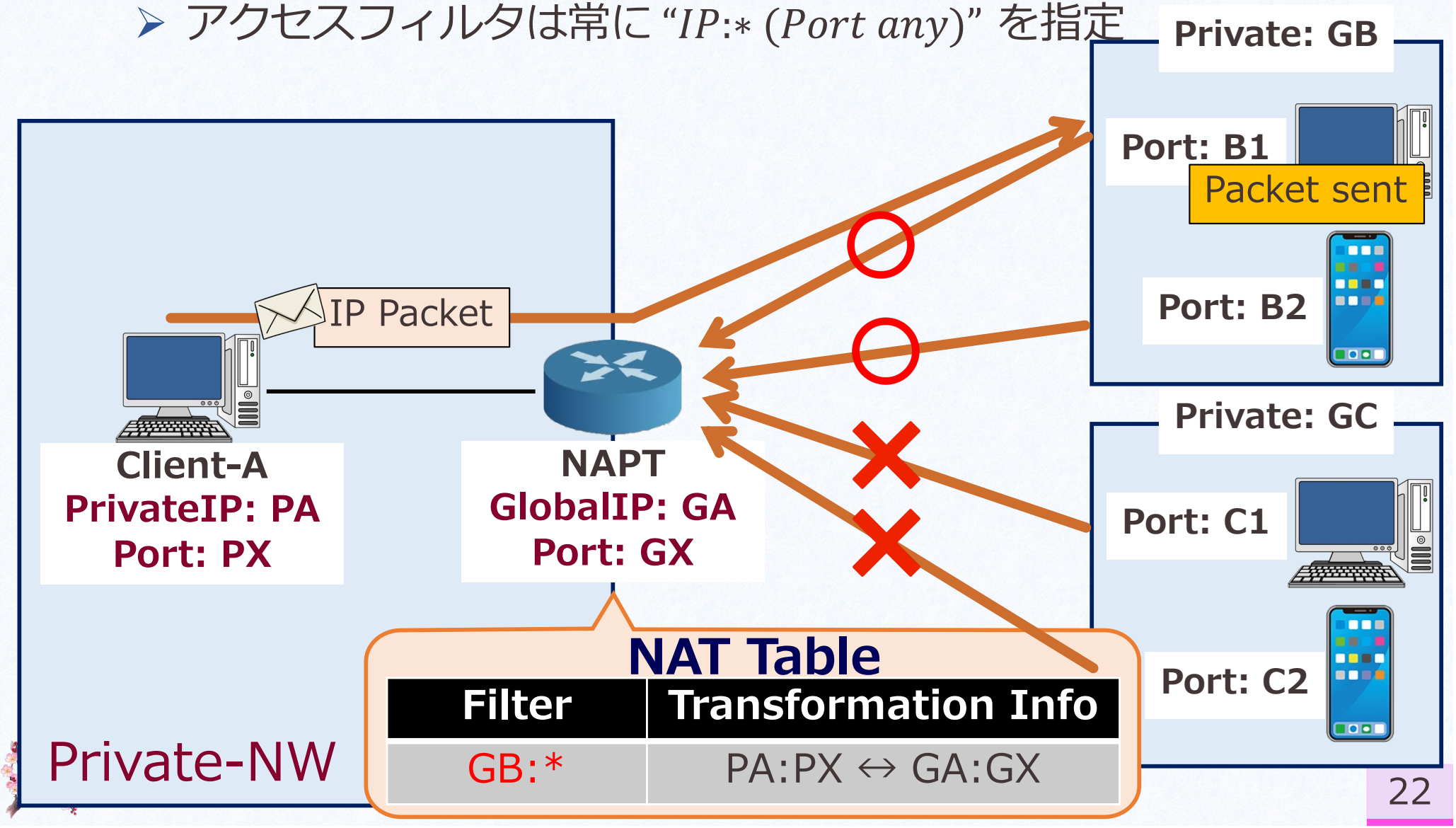
Full Cone NAT

- 一度も送信したことのないWAN側からのパケットも受信可能
  - パケットは変換情報が存在すれば内部のIP/Port宛てに転送
  - アクセスフィルタは常に“ *:* (*any*)” を指定



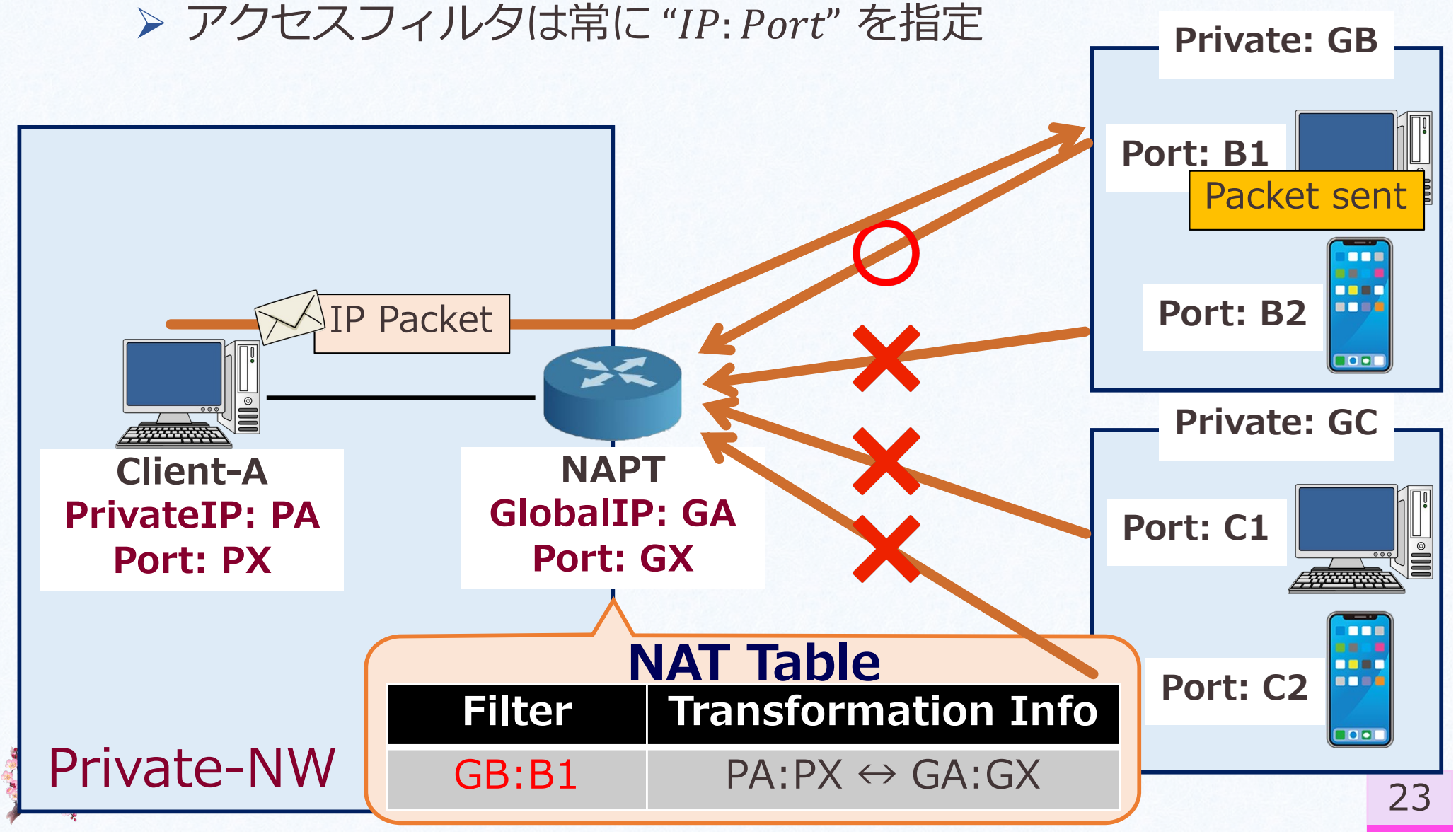
Restricted Cone NAT

- 一度送信したことのあるIPからのパケットであれば受信可能
  - パケットは変換情報が存在すれば内部のIP/Port宛てに転送
  - アクセスフィルタは常に“IP:* ($Port\ any$)” を指定



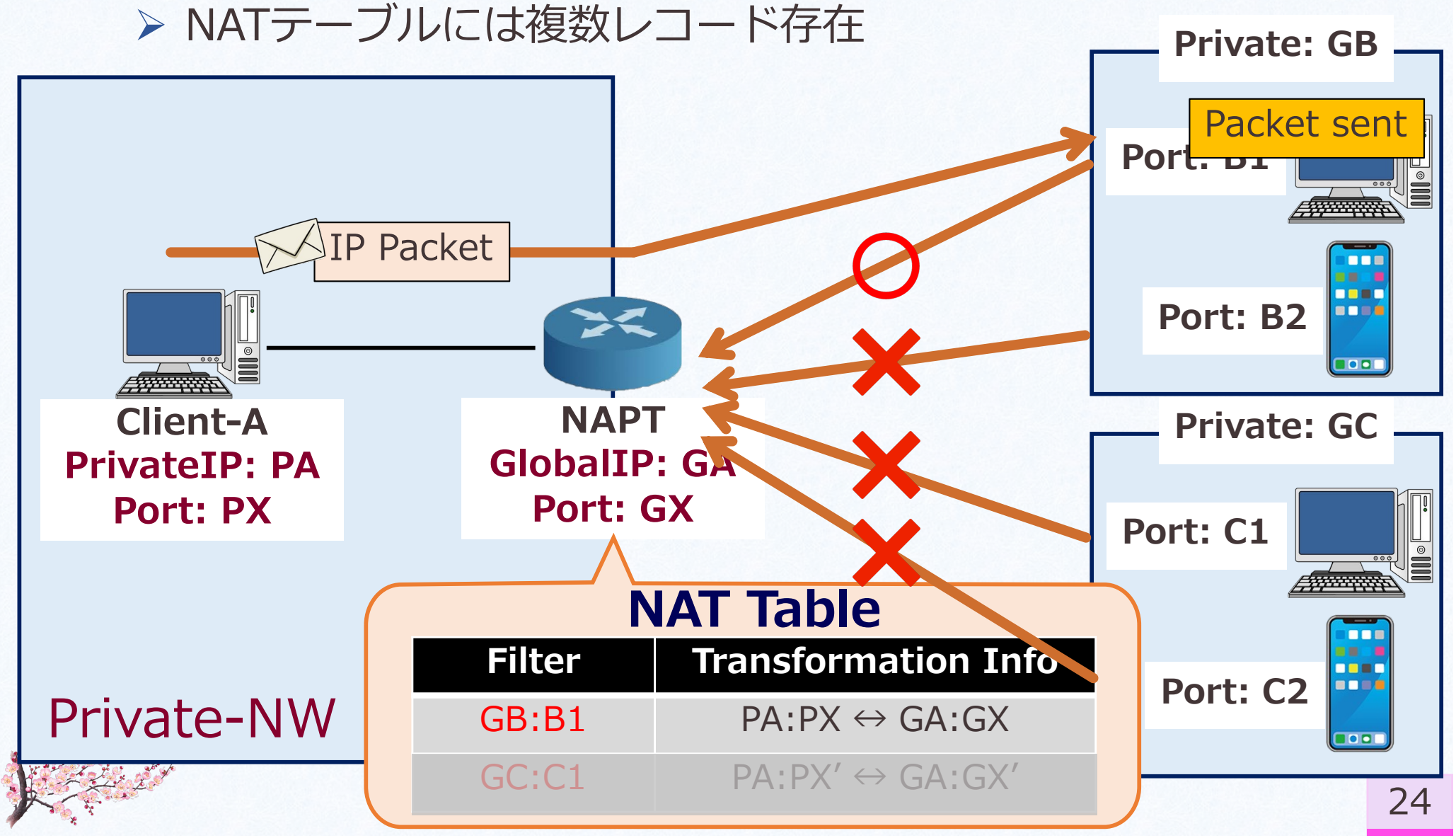
Port Restricted Cone NAT

- 一度送信したことのあるIP及びPortからのパケットであれば受信可能
  - パケットは変換情報が存在すれば内部のIP/Port宛てに転送
  - アクセスフィルタは常に“$IP:Port$” を指定



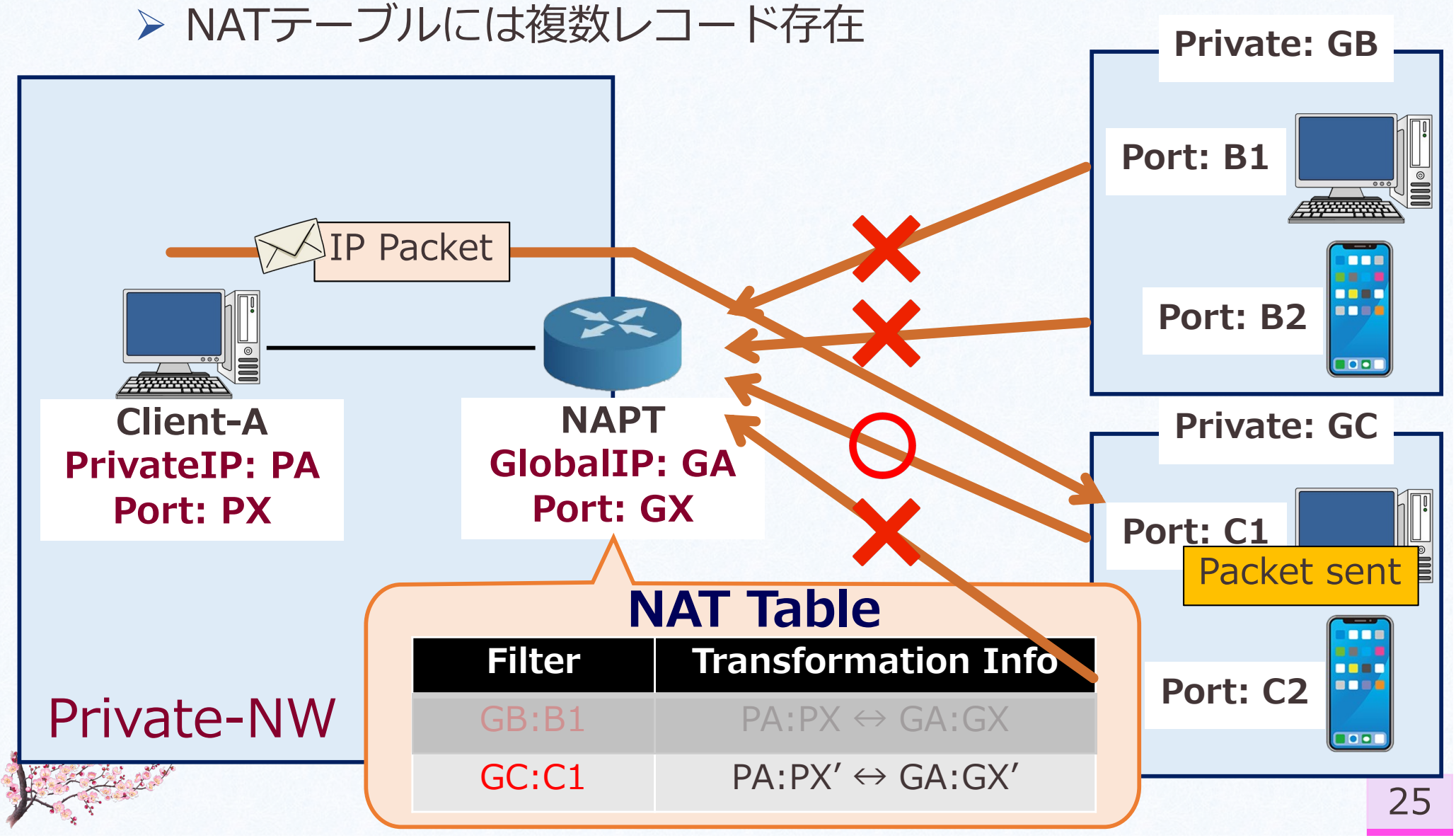
Symmetric NAT

- マッピングが存在する宛先デバイスからのパケットであれば受信可能
  - ーデバイスが複数の変換情報を保有
  - NATテーブルには複数レコード存在



Symmetric NAT

- マッピングが存在する宛先デバイスからのパケットであれば受信可能
  - ーデバイスが複数の変換情報を保有
  - NATテーブルには複数レコード存在



NATの種類とトラバーシング

- **STUN (Session Traversal Utilities for NAT)**
  - STUNサーバを導入して変換情報を記録
  - 相手デバイスに通知することでNAT Traversalを実現
- **TURN (Traversal Using Relays around NAT)**
  - TURNサーバが両者の間で通信を中継
  - TURNを介することでSymmetric NATに対応可能
- **ICE (Interactive Connectivity Establishment)**
  - STUNやTURNなど複数のNAT Traversalを応用
  - デバイスが存在するNW環境に最適なNAT越えを実現

	Cone型			Symmetric型
NAT Traversal	Full Cone	Rest-Cone	Port-Rest-Cone	Symmetric
STUN	○	○	○	×
TURN	○	○	○	○
ICE	○	○	○	○

論文・参考文献

[1] Research on Symmetric NAT Traversal in P2P applications

©2006 IEEE - Yong Wang, Zhao Lu, Junzhong Gu

[2] RFC 5128 (UDP Hole Punching)

https://tex2e.github.io/rfc-translater/html/rfc5128.html

[3] RFC 8489 (STUN)

https://tex2e.github.io/rfc-translater/html/rfc8489.html

[4] RFC 8656 (TURN)

https://tex2e.github.io/rfc-translater/html/rfc8656.html

[5] RFC 8445 / RFC 8839 (ICE)

https://tex2e.github.io/rfc-translater/html/rfc8445.html

https://tex2e.github.io/rfc-translater/html/rfc8839.html