# 並列分散コンピューティング
# (13)負荷分散, P2P, DHT

大瀧保広

# 今日の内容

今日の内容は試験範囲に含まれません。

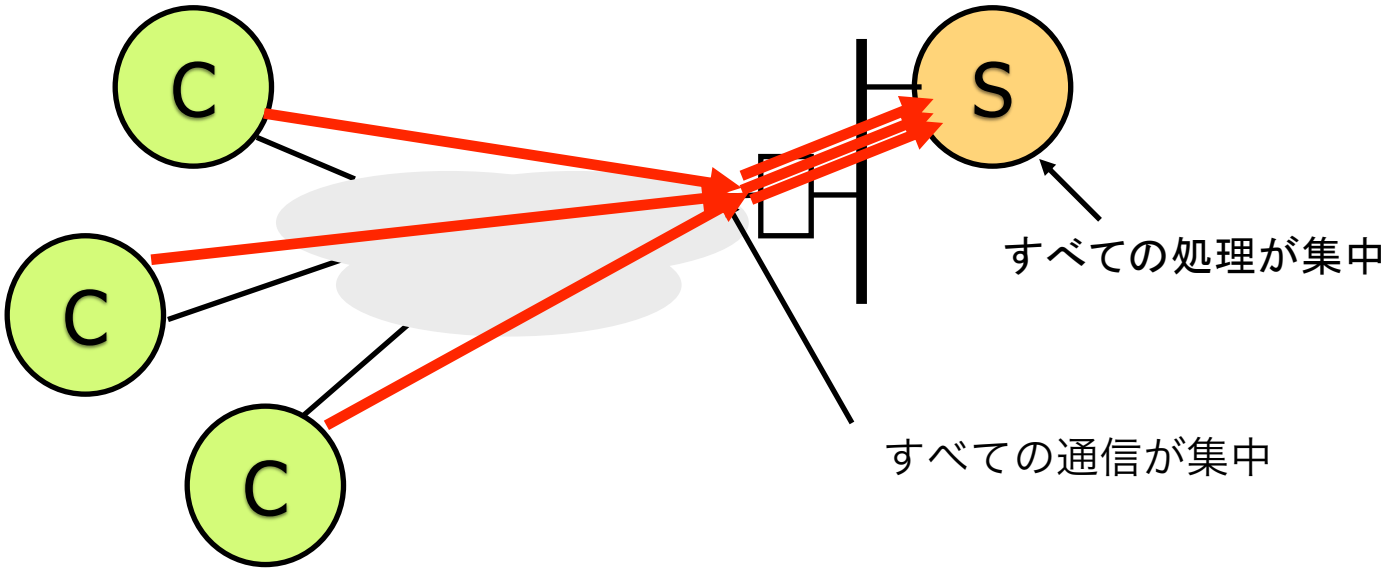
分散システムを実際に運用する場合に
どのようなことが問題になるのか、
いくつか例を見ていきます。

- 負荷分散手法
- P2Pにおける資源探索（と匿名性）
- DHT (Distributed Hash Table)
  - Chord

# 負荷分散の手法

# Serverへの負荷集中の問題

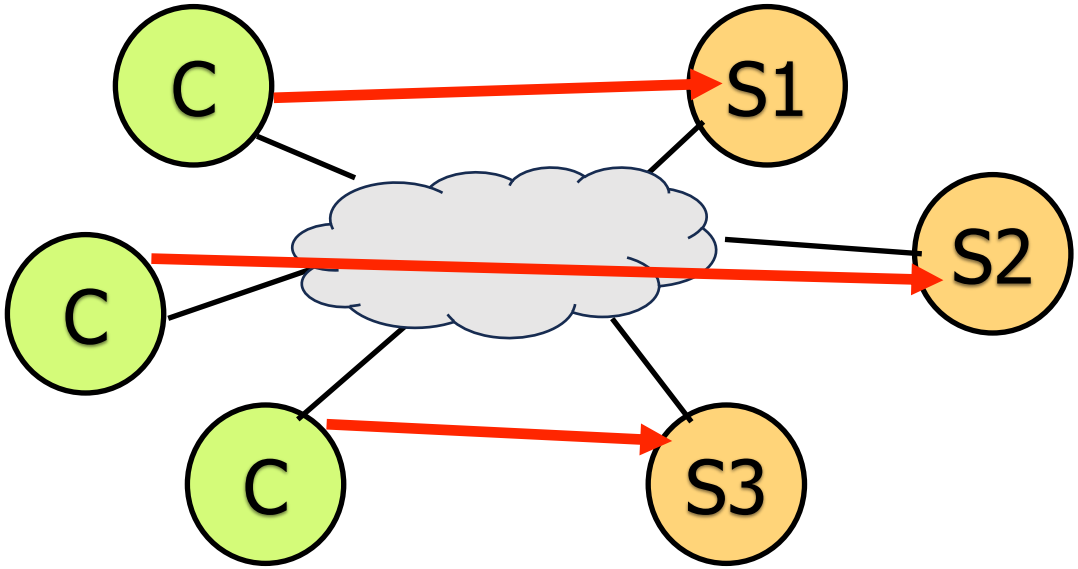
- サーバにはクライアントからのすべてのリクエストが集中。
- 世界規模でサービスを提供するサービスでは、
  すべてのクライアントからの要求を処理できない。



# 対策１：Mirror server

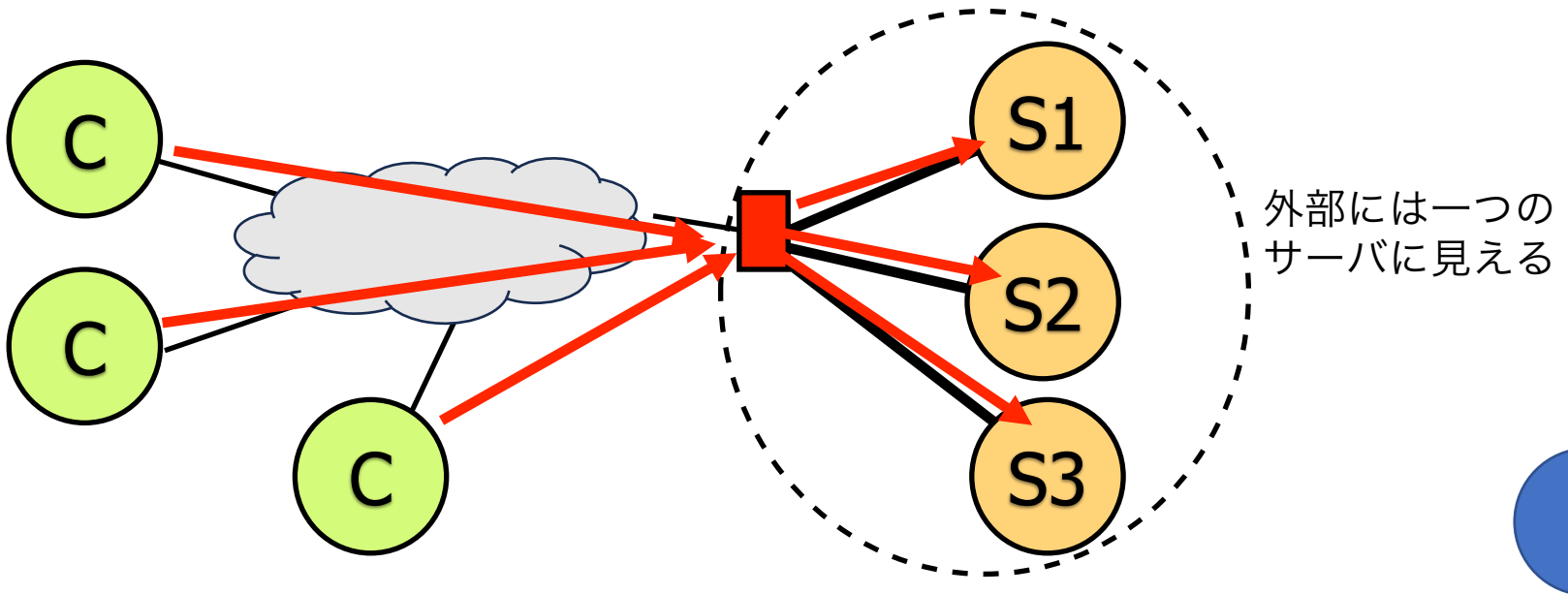
同一のサービスを提供するサーバを複数台 配置する。

- Internetの離れた地域にMirrorサーバを分散配置することで、ネットワーク通信を分散させることができる。
- 複数のサーバのどれにアクセスするかは、クライアントが明示的に選択する。
- 欠点：クライアントに対して “透明” Transparent でない。



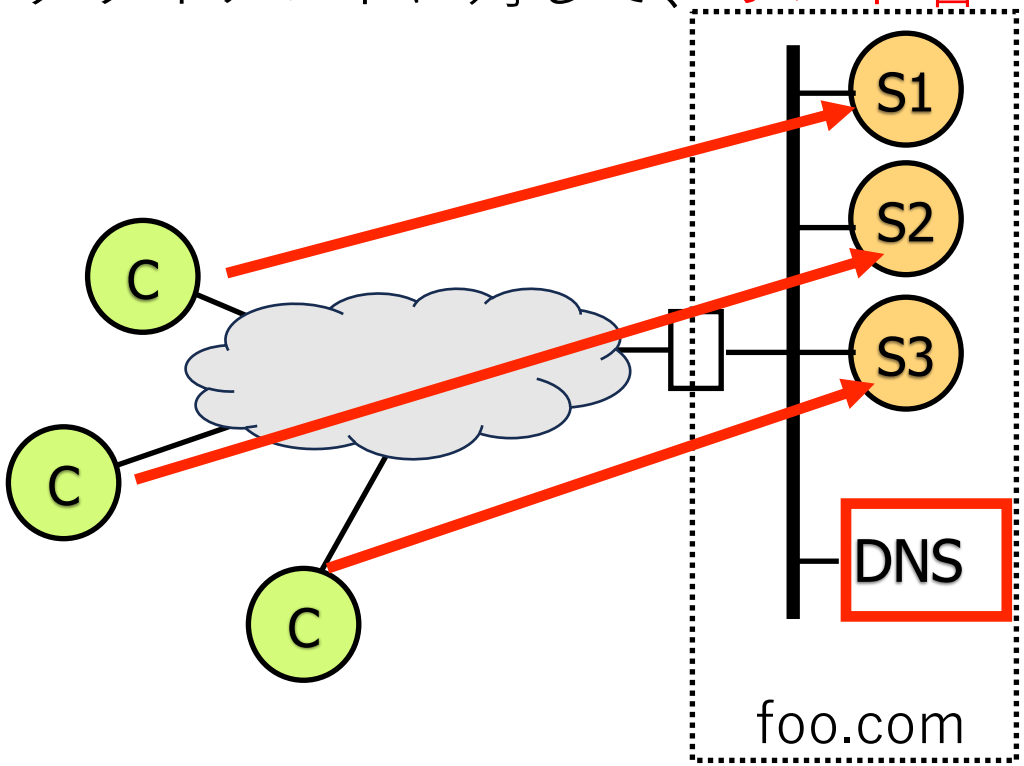
# 対策２：Load Balancer

- 複数のサーバを同一のIPアドレスに対応づけることができる。
- 各サーバの負荷に応じてコネクションを振り分ける。
- クライアントに対して完全に透明に実装できるが、
  特殊なハードウェア機器であるため高コスト
- 原理的にインターネット上に分散配置することができない。



# 対策3：Round Robin DNS

- ホスト名からIPアドレスを取得するために
  DNSに問い合わせたときに、組織のDNSが
  毎回異なるIPアドレスを答える。
- 同じホスト名でも異なるサーバにアクセスさせる。
- クライアントに対して、ホスト名レベルで透明。



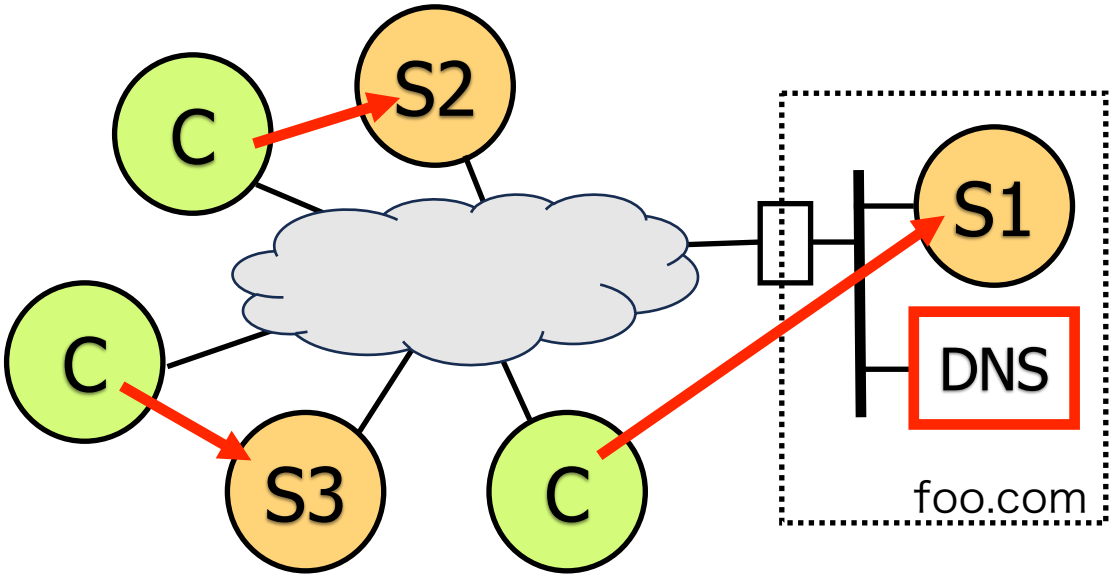

www.foo.comのIPを聞かれたら、
S1のIPを答える。

www.foo.comのIPを聞かれたら、
S2のIPを答える。

www.foo.comのIPを聞かれたら、
S3のIPを答える。

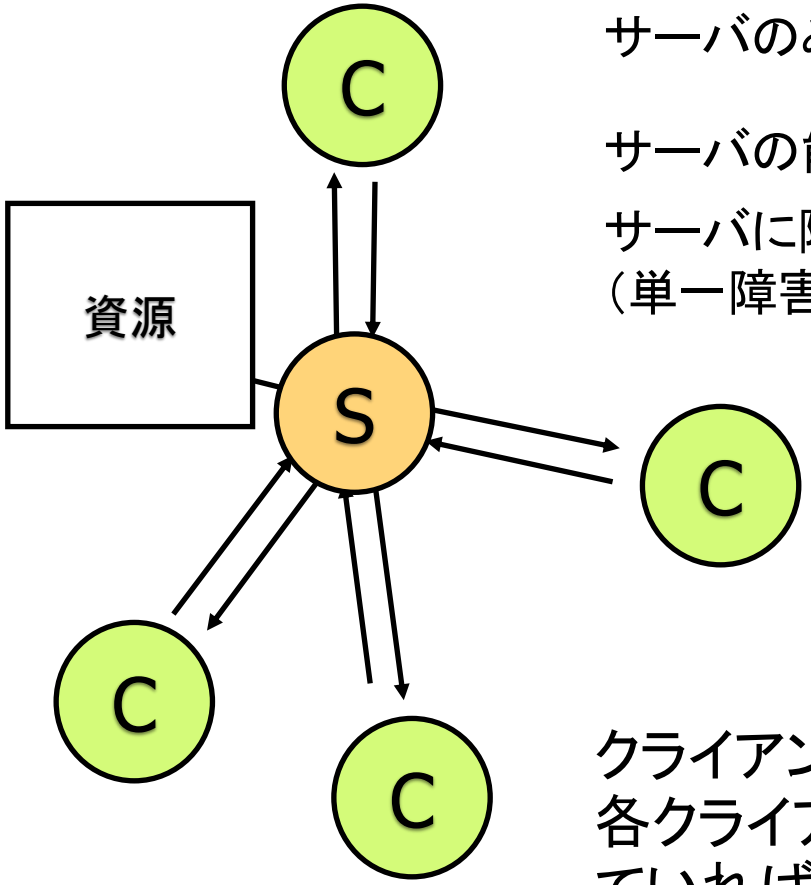
# 対策3：Round Robin DNS （つづき）

- 各サーバをミラーサーバとしてインターネット上に分散配置できれば、通信も分散させることができる。
- DNSが、クライアントのIPアドレスにネットワーク的に近いミラーサーバのIPアドレスを答えるように構成できると通信の軽減に効果絶大。



# P2Pにおける資源探索（と匿名性の問題）

# Peer-to-Peer (P2P)




サーバのみが資源（情報、計算能力）を持つ

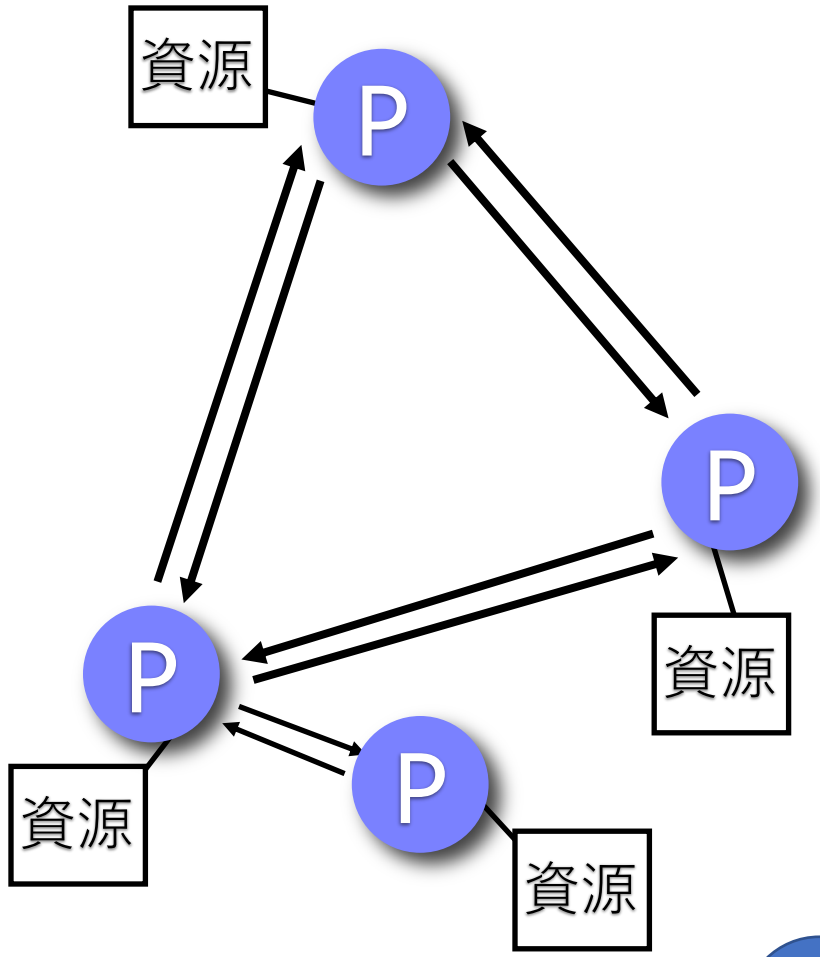
サーバの能力＞＞クライアントの能力

サーバに障害が起きると、システム全体が機能停止。
（単一障害点）

クライアントの実装は比較的容易。
各クライアントは、サーバに関する情報を知っていれば、資源を利用できる。

# Peer-to-Peer (P2P)モデル

- 処理能力と通信能力の向上によって、 クライアントだけでも様々な処理が実行可能となった。
- 関係が対等なので「ピア」と呼ぶ。
  peer：同等の者、対等の者
- 一つのピアは、
  他のピアからのアクセスを
  受け付けるサーバの機能と
  他のピアにアクセスするための
  クライアントの機能を合わせ持つ。



# 問題：P2Pでの資源発見はどうする？

P2Pにおいて、ピアが[ある資源]を利用したいと思った時：

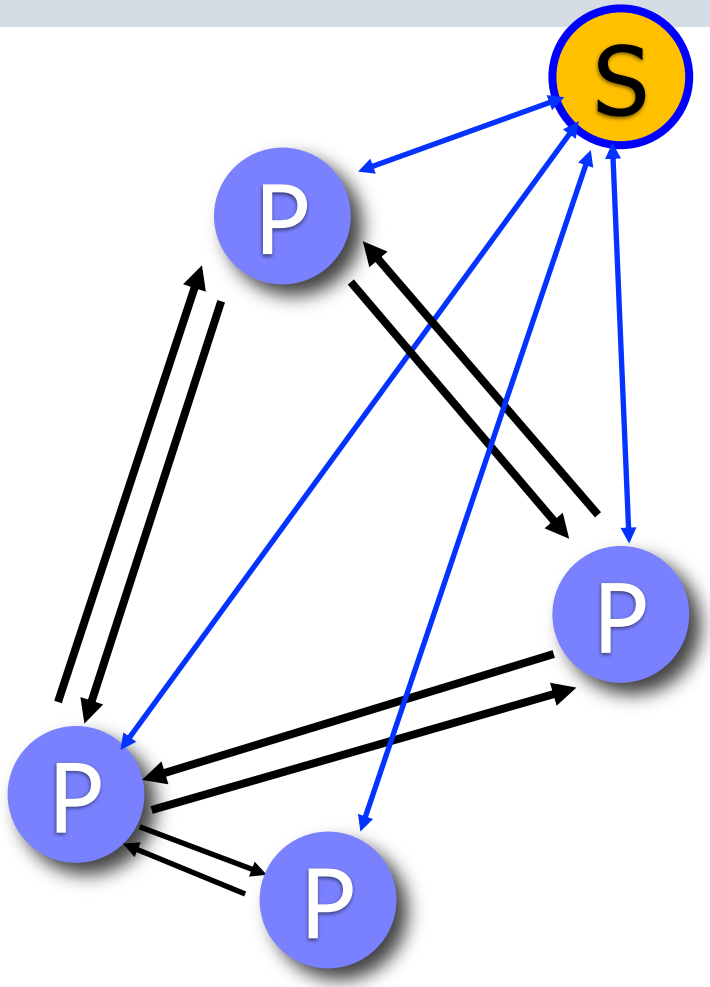
- その資源はどのピアが持っているのか？
- そもそも そのピアはどこにいるのか？

- Hybrid P2P
  「ピア（＆そこにある資源）に関する情報」を管理する
  サーバ（インデックスサーバ）を置く方法
- PureP2P
  インデックスサーバを置かず、
  **ピア同士の情報交換のみで解決**する方法

# Hybrid P2P

各ピアに関する情報を管理する
**インデックスサーバ**がある。

- 各ピアは、サーバの情報を参照して他のピアと通信を行う。



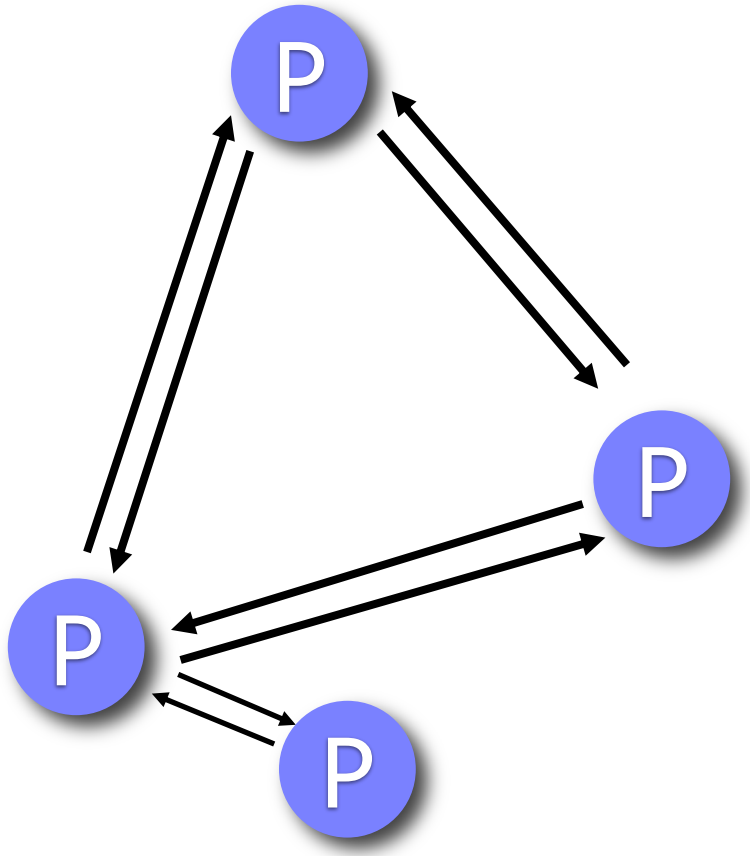

- メリット：無駄な通信が少ない
- デメリット：
  サーバに障害が発生すると
  システム全体が機能停止。
  （単一障害点）

# Pure P2P

ピアのみでシステムを構成する。

- 他のピアの情報は、最寄りのピアから教えてもらう。

- メリット：あるピアに障害が発生しても、そのピアが機能しなくなるだけで 他のピアには影響がないように構成できる。
- デメリット：ピア数の増加に伴い、通信量が加速度的に増加する傾向がある。



# PureP2Pの例：Gnutella（ぬてら）

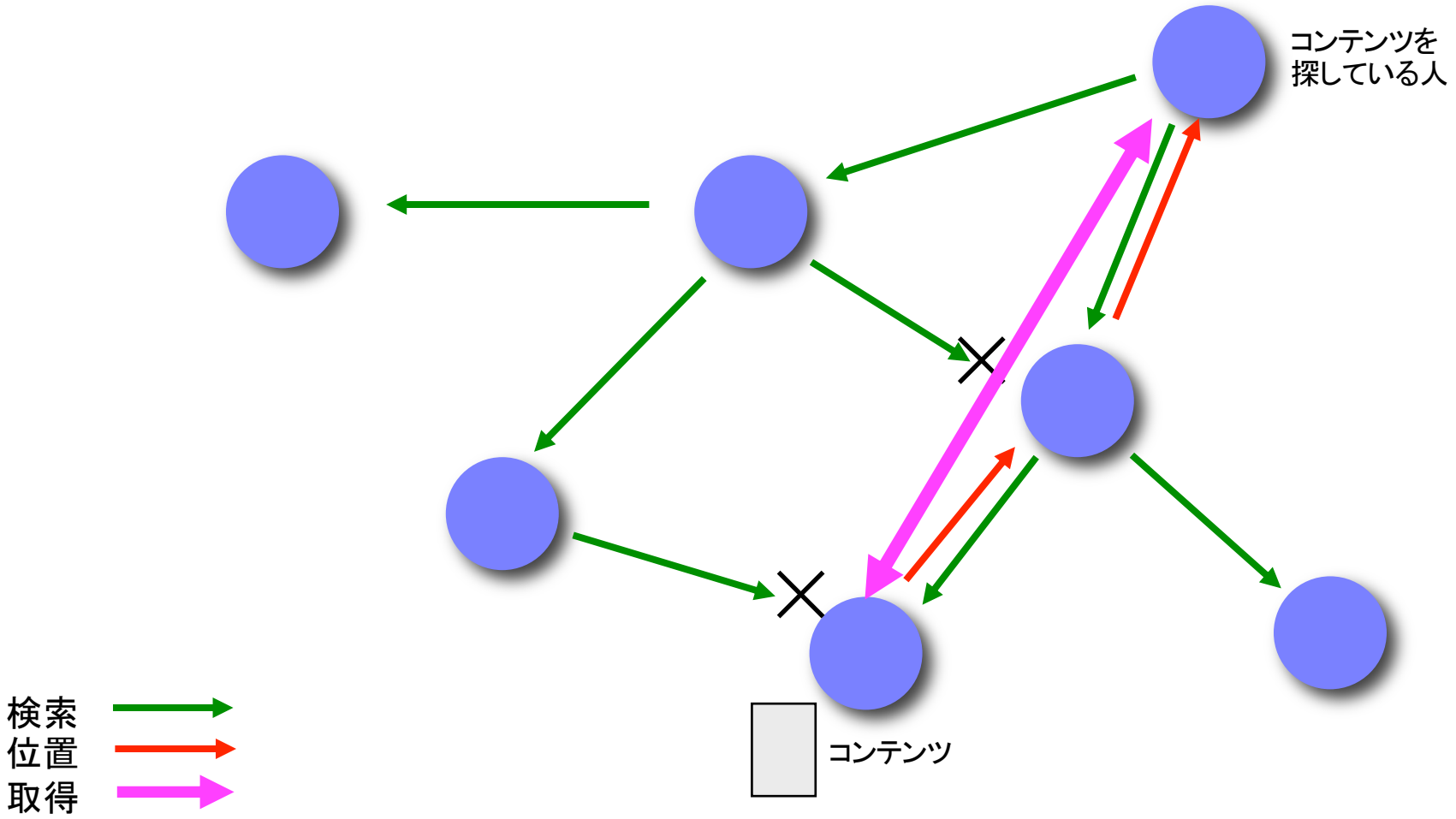

Gnutella：ファイル共有システム

- 各ピアはごく少数の近接ピアへの経路だけを保持

Flooding と呼ばれる原始的な資源探索アルゴリズム

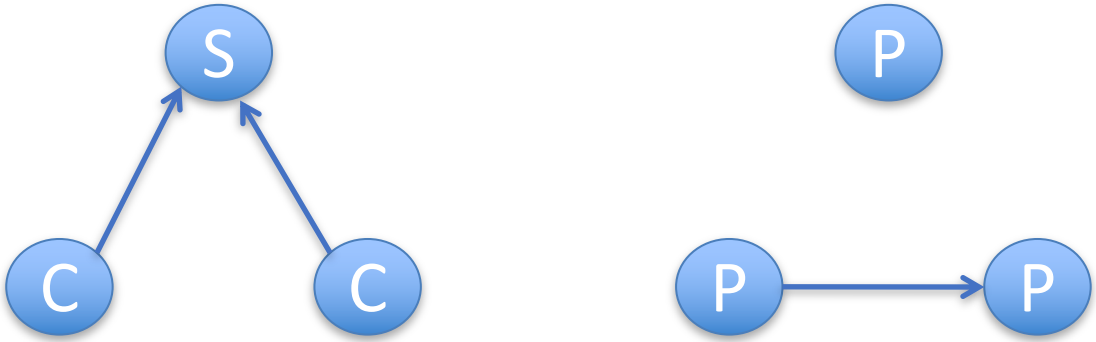
- 近隣ピアへ一斉に探索要求を送信。
  各ピアはさらに近隣ピアへ一斉に転送（幅優先探索）
- 探索要求にはTTL(Time-To-Live)があり、
  転送するごとにデクリメントする。（無限転送の防止）
- 探索要求にはIDを付与し、ループを検出する。
- 探索成功時には、探索結果（位置情報）を
  探索要求がきた経路に沿って逆向きに伝搬。

# Flooding



# P2Pと匿名性

- Client-Serverモデルのシステムでは、
  サービスに参加するClientが
  他のClientの情報を得ることはなかった。
- P2Pシステムでは、
  サービスの送受信はPeer同士が直接通信を行う。
  - 通信相手に自分の情報が知られるのではないか？
    暗号通信を使えば大丈夫か？



# 暗号通信（TLS）では匿名性は保護されない

- TLS(Transport Layer Security)は、Transport層での暗号化。
  ネットワーク層の情報は守られていない。


- 想定される脅威
  - 誰と誰が通信しているか が**第三者に**推測される
    →結果的に、何をしようとしているか
    を　推測される可能性がある
  - **受信者には**送信者の情報がわかってしまう

# Freenet

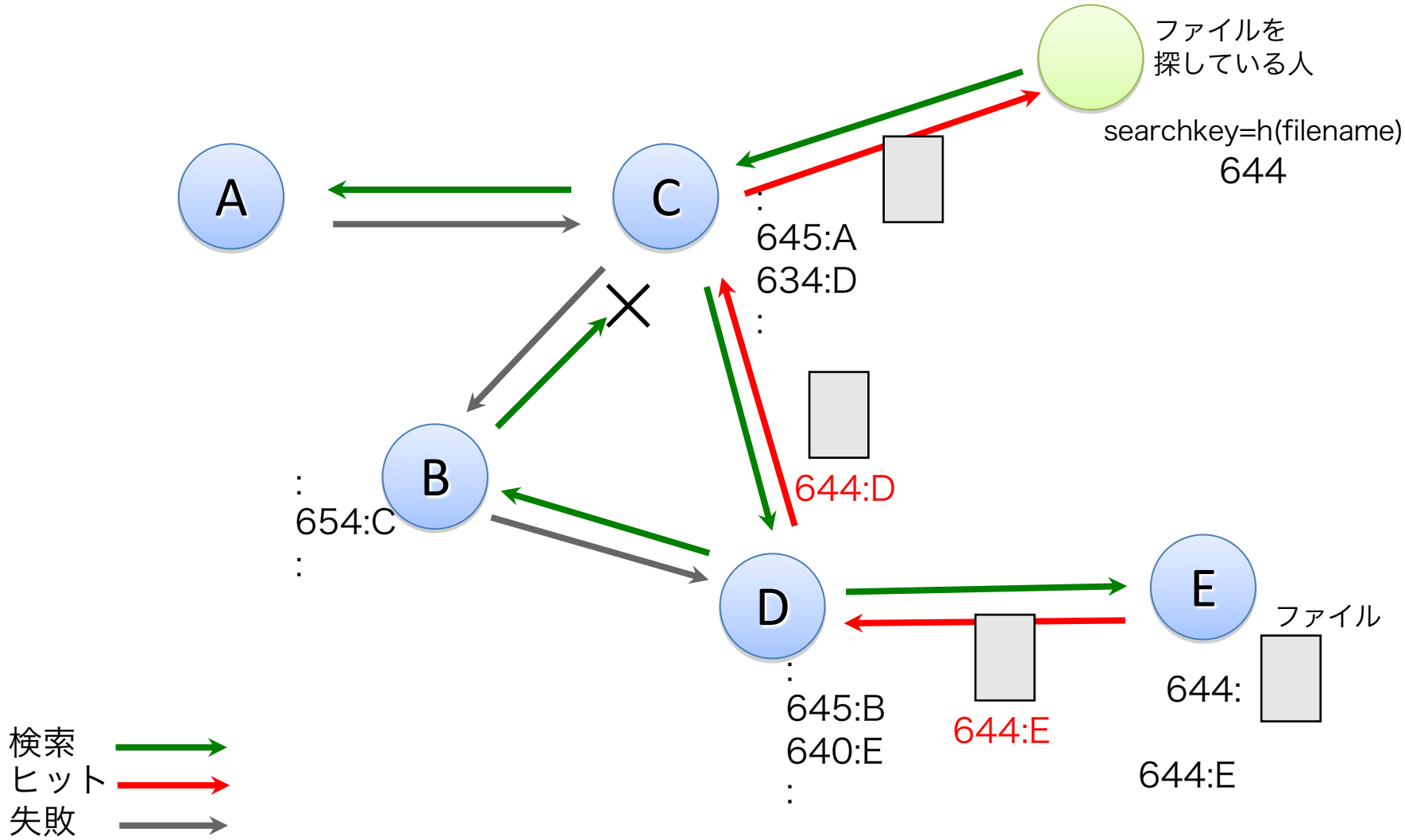
- 目的
  - 比較的サイズの小さいドキュメントの流通を目的としたPureP2Pシステム
  - 「言論の自由」の保護を目的とした、匿名性の厳密な保証。（転送効率や利便性を犠牲にしている）

- Freenetの特徴
  - 探索キーとして、ファイル名そのものではなく、ファイル名から生成されたハッシュ値のみを利用する。
  - 各ノードは、近接ノードのIPアドレスと探索キーのペアの集合を持つ。

# Freenetの探索アルゴリズム

- 基本的な動きはGnutellaに似ている。
- 探索キーと数値的に最も近いキーを経路表から探し、
  その近接ノードに探索要求を転送。
- TTL終了かループ検出したら、探索失敗を逆向きに送信。
- 探索失敗が来たら、次に近いキー値のところに転送。
  (深さ優先探索)
- 探索が成功（＝探索キーに一致するファイルを発見）したら
  - ファイル本体は、探索経路に沿って逆向きに転送される。
    直接探索ノードに直接送信しない→中継ダウンロード
  - 各中継ノードでは、経路表を更新する。
    （「探索キーと経路上のノード」を経路表に追加する）
    ←人気のあるファイルほど探索転送数が減少していく。

# Freenetでの探索とファイル転送



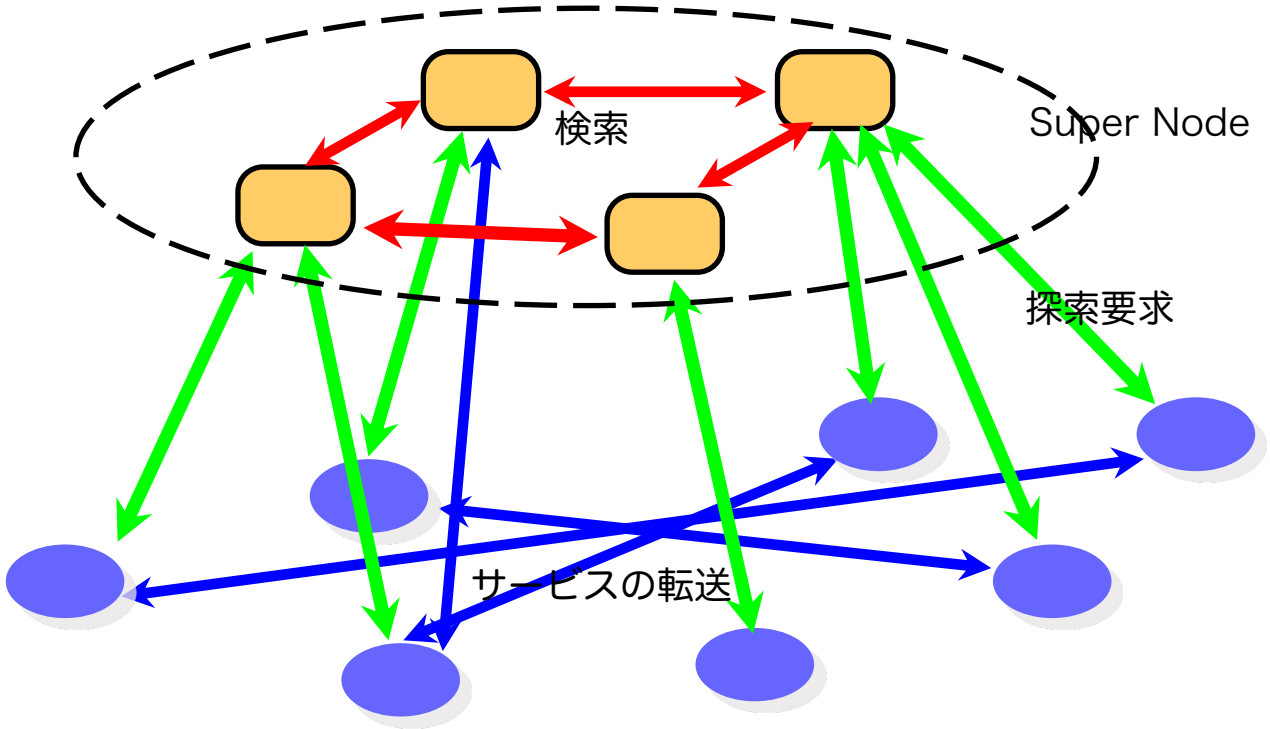
# Pure P2Pにおける資源探索

- Pure P2Pモデルのシステムでは、資源の探索が一番の問題
  - Index Serverと同等の機能をどのように実現するか？
  - ノード情報を取得するための仕組みが必要。
    IPアドレス、URL、サービスのメタ情報などなど。

- Unstructured P2P.... Flooding (非常に効率が悪い)
- Structured P2P
  - P2P上にサービスを管理する構造を組み込む。
    一部のPeerにIndex Server の機能を持たせる。
  - Super Nodeは固定せずノードの参加／離脱を認める。

# Structured P2P

- Nodeに階層構造を導入
  - Super Node: Indexを分散管理するノード
  - 一般Node:その他のノード



# SuperNodeで管理する情報

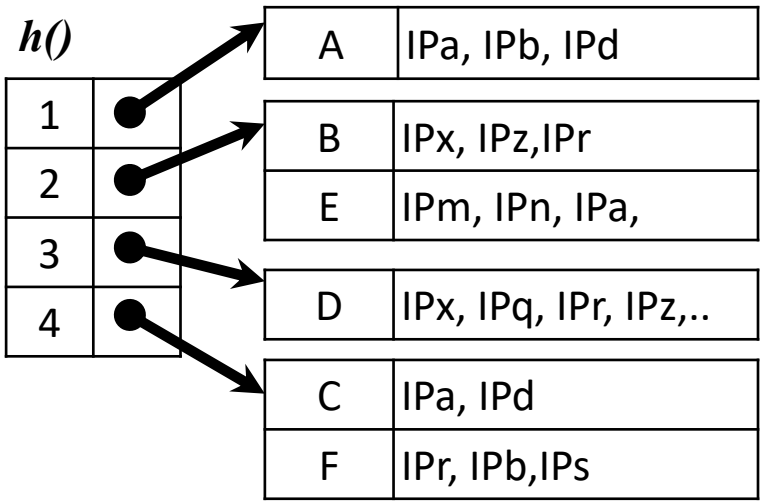
- どのコンテンツが
  どのノードにあるか
- Hash Table
  - {Key, Value}を保存
  - Keyをもとに
    Valueを取り出す

| contents | host IP |
|---|---|
| A | IPa, IPb, IPd |
| B | IPx, IPz,IPr |
| C | IPa, IPd |
| D | IPx, IPq, IPr, IPz,.. |
| E | IPm, IPn, IPa, |
| F | IPr, IPb,IPs |

*h()*

| | |
|---|---|
| A | *1* |
| B | *2* |
| C | *4* |
| D | *3* |
| E | *2* |
| F | *4* |

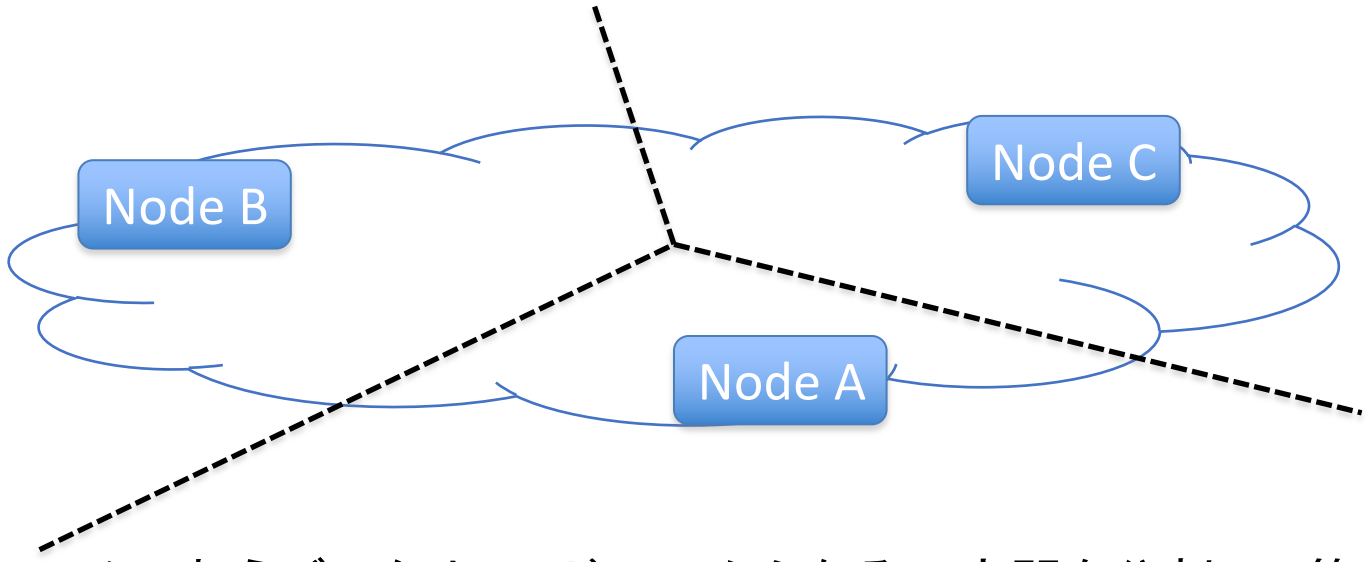
- この情報を複数ノード上に
  保存、検索、取得する
  仕組みが必要
  →
  Distributed Hash Table



# Distributed Hash Table

# どのSuperノードに保存するのか?

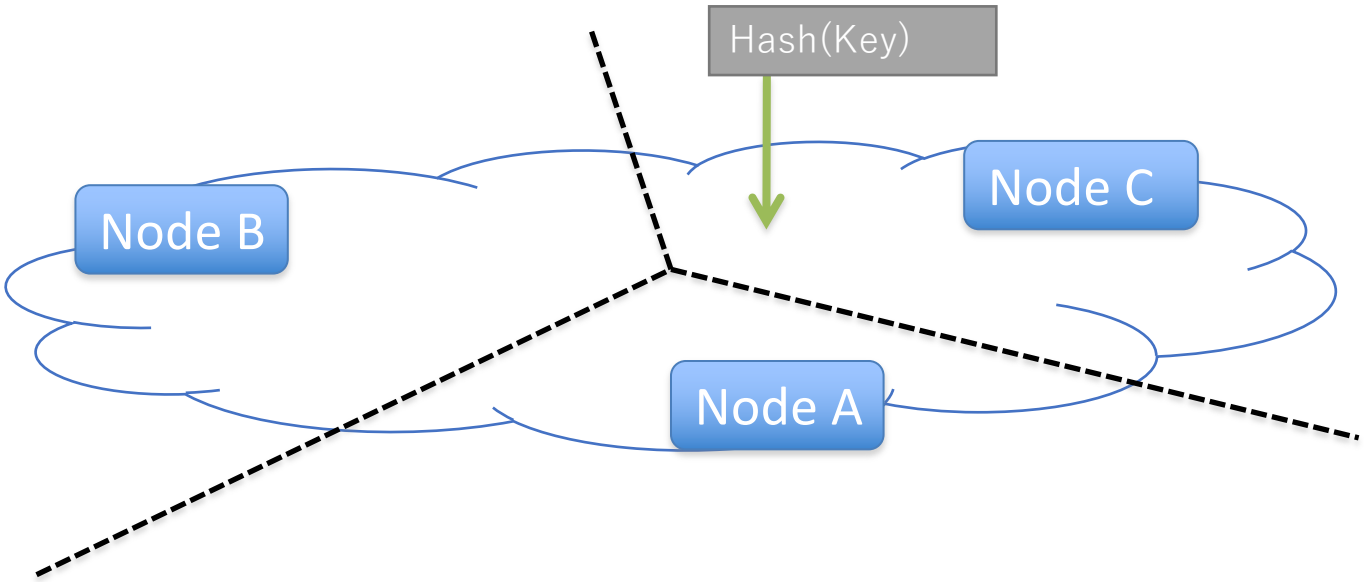
- すべてのノードは、ユニークなノードIDを持つ
- ノードIDに応じて "ID空間上の位置" が決まる
- ID空間上の位置によって、
  そのノードの「ID空間の担当領域」が決まる。




P2Pシステムで扱うデータやノードのIDからなるID空間を分割して管理する

# どのノードに保存するのか?

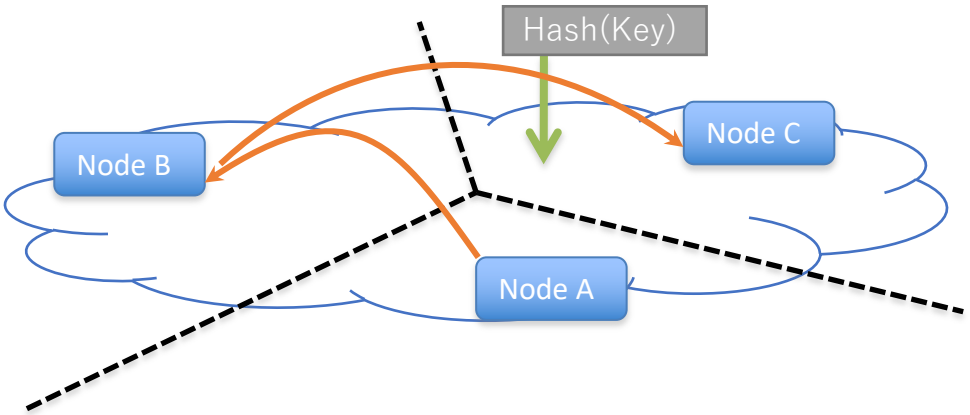
- {Key, Value}を保存するとき
  - Keyを「ID空間」へマップする。
    データID = Hash(Key)
  - データID を担当領域に含むノードが担当ノードとなり、
    {Key, Value}を保管する。



# 担当ノードへたどり着くには?

- 全ノードは、経路表（ノードIDとIPアドレスの対応表）を持つ。
- データが欲しいノードは経路表を見て、データIDに最も「近い」ノードIDをもつノードに処理を丸投げする。
- 丸投げされたノードでも自分が担当でなければ同じことを繰り返す。
- 最終的に担当ノードへたどり着く。

| ノードID | IPアドレス |
|---|---|
| 0x0111 | 157.80.11.1 |
| 0x0222 | 157.80.22.2 |
| … | … |
| … | … |



# DHTのアルゴリズム

- 必要な処理
  - ノードIDの決め方
  - ID空間の構成方法
  - 担当領域の決め方
  - 距離の決め方

- アルゴリズム
  - Chord
  - CAN
  - Pastery
  - Tapestry

STOICA, I., MORRIS, R., LIBEN-NOWELL, D., KARGER, D. R., KAASHOEK, M. F., DABEK, F., AND BALAKRISHNAN,H.
Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications.
IEEE/ACM Transactions on Networking 11, 1, 17–32. 2003.

# ノードIDとID空間

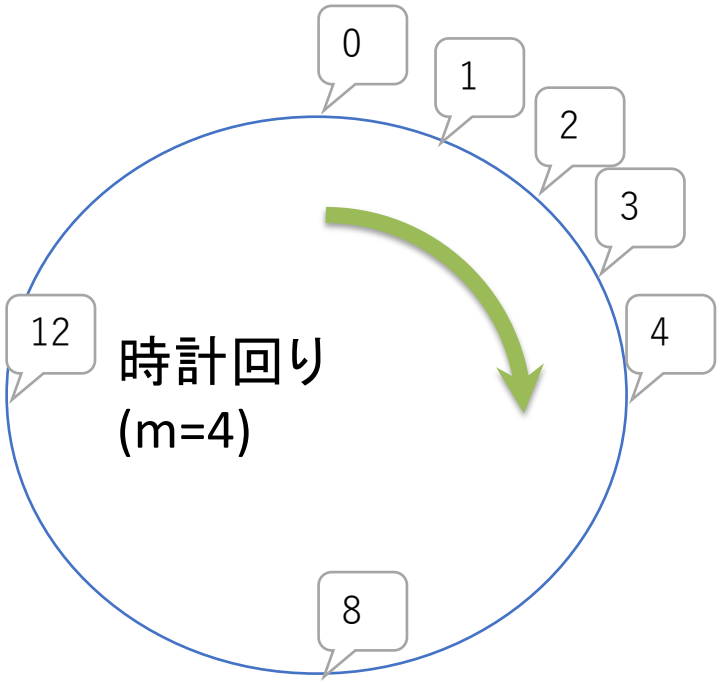
■ノードIDの決め方の例

ノードID = hash( ノードのIPアドレス )

ノードIDが mビットのとき、0～ $2^m-1$ の整数値となる。

■ID空間

■ノードIDに基づいて、
リングネットワークを構成。
（$2^m-1$と0が隣接）

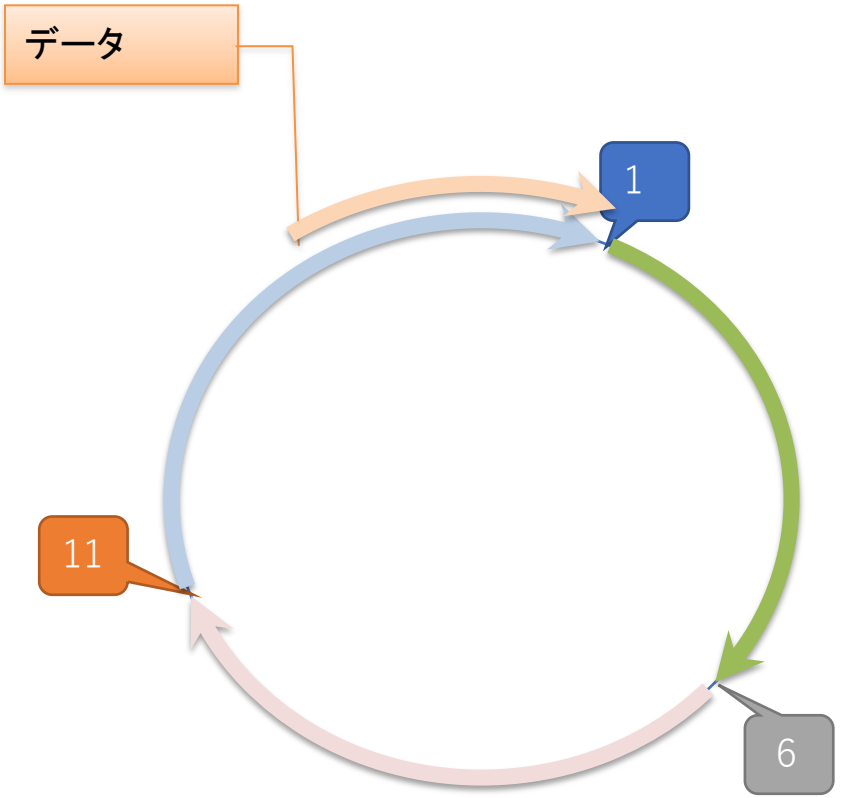


# ノードの担当領域

- データIDの位置からID空間を時計回りに進んで
  最初に見つかったノードが担当ノード（データを保持するノード）

- Successor
  あるIDから時計回りに進んで
  最初に当たるノードのこと。

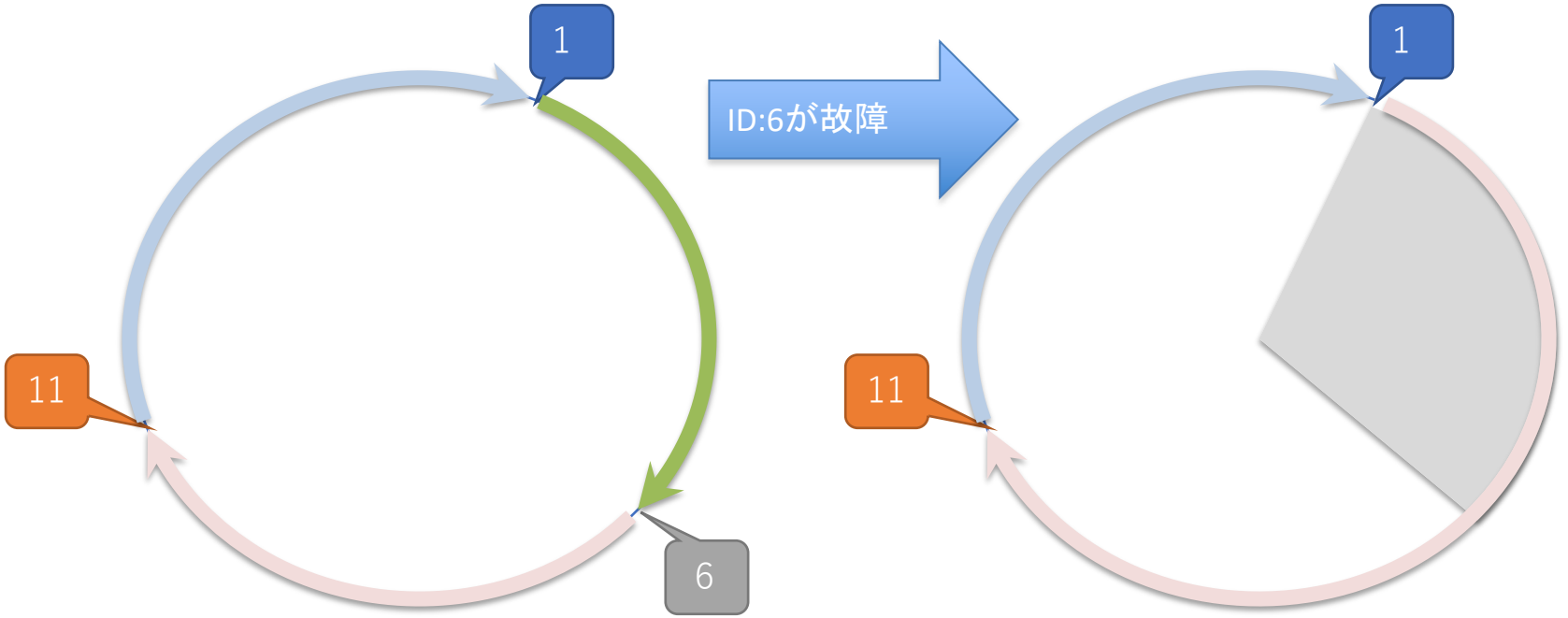
- 各ノードは、経路表に 自分の
  Successorの位置を記録。




| | ノードID | IPアドレス |
|---|---|---|
| Successor | 11 | … |

# 耐故障性

- あるノードが故障すると、
  そのノードが担当していた領域を担当するノードが変わる。
- しかし、残りの範囲は担当ノードに変化がない。
  （→データ移動の必要がない）

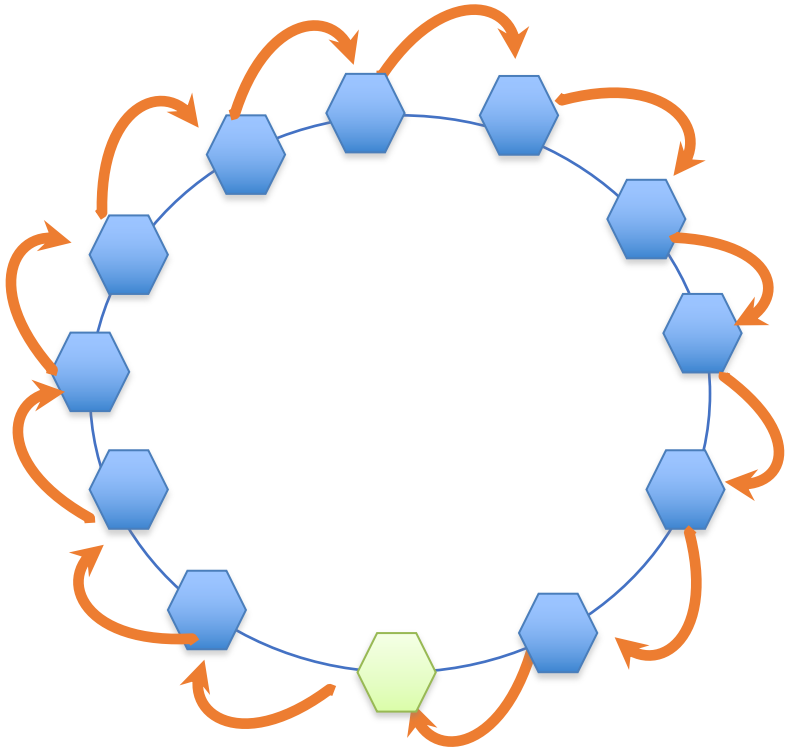


# 経路表：Successor (Only)

- ノードの経路表にはSuccessorの情報だけ

- Successorさえ経路表に正しく登録されていれば、確実に到達できる。
  ノードへの到達性を保証

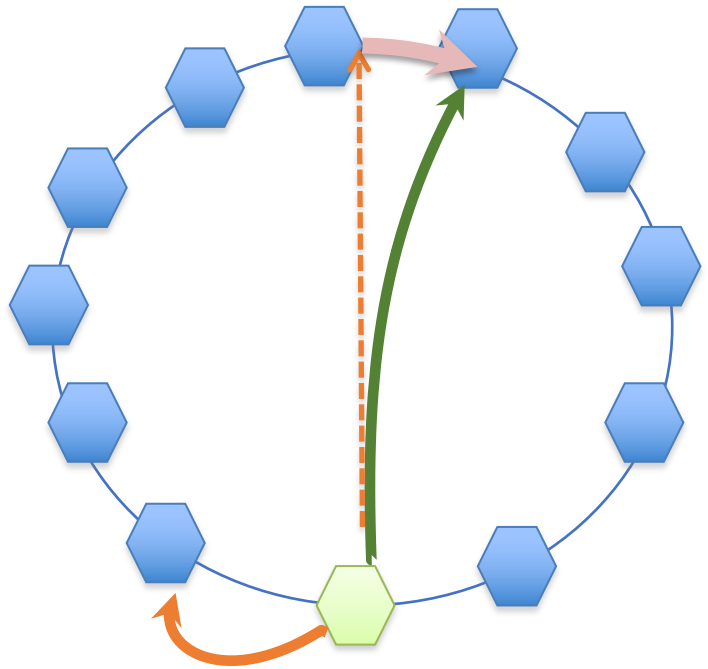
- 担当ノードに到達するまでの経路長(転送回数)は平均で $2^m/2$



successor

# 経路表：Successor + FingerTable

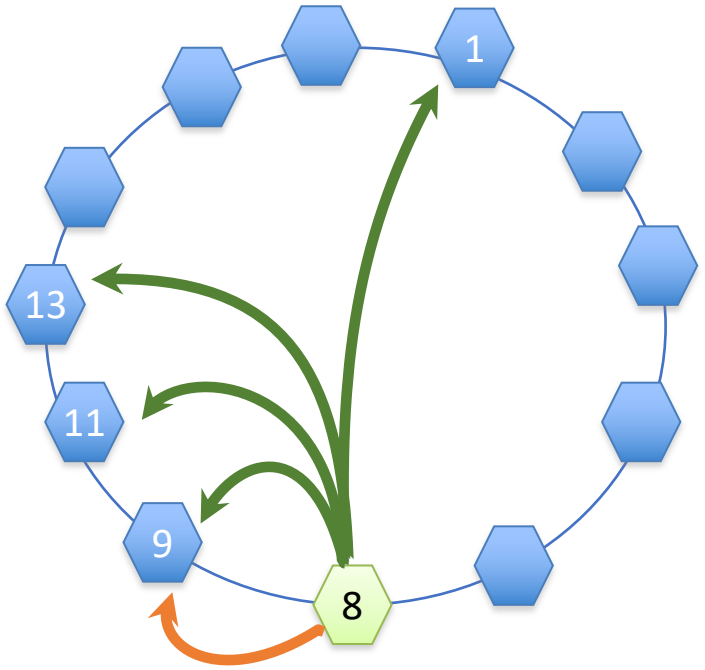
- FingerTable
  - 経路長を短くするため
  - Finger Tableは到達性を保証しないので、Successorと併用。
- Finger Tableに格納するデータ：
  - リングを1/2周行った先のIDの担当ノード



# 経路表：Successor + FingerTable

- FingerTableに格納するデータ：
  - リングを 1/2周行った先のIDの担当ノード
  - リングを 1/4周進んだ先のIDの担当ノード
  - …
  - リングを 1/($2^m$)周進んだ先のIDの担当ノー
- Successorと かぶっても構わない
- 1回転送するごとに必ず
  残りの距離が半分以下になる。

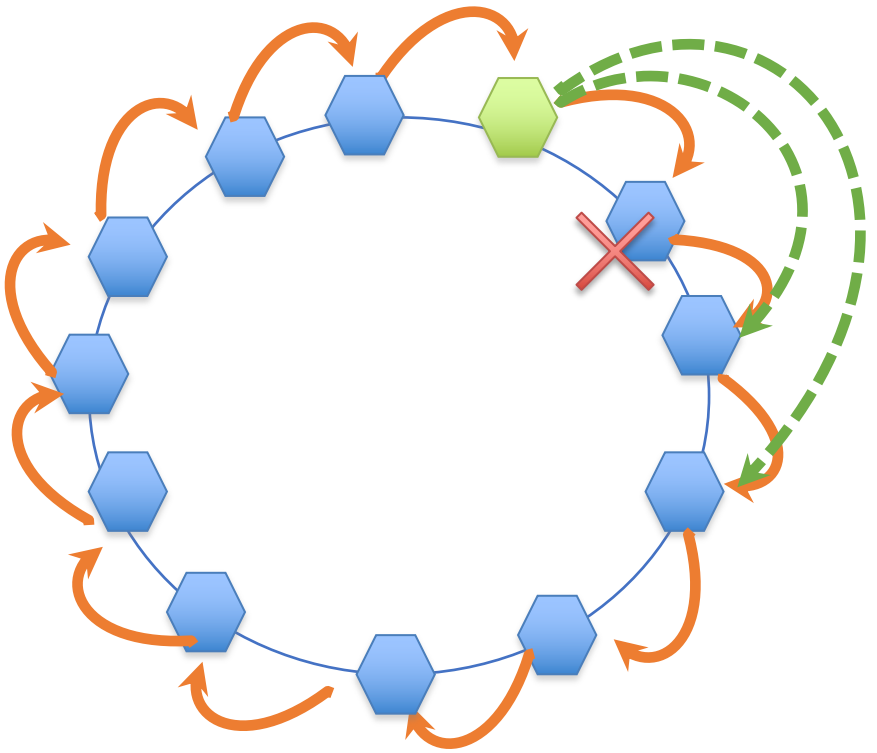
| (m=4) | ノードID | IPアドレス |
|---|---|---|
| Successor | 9 | … |
| Finger[1] | 9 | … |
| Finger[2] | 11 | … |
| Finger[3] | 13 | … |
| Finger[4] | 1 | … |



# 経路表：SuccessorList + FingerTable

- Successorによるノードへの到達性保証は、1ノードが故障すると失われてしまう。
- ノードが失われたときのために
  次のsuccessor
  次の次のsuccessor
  の情報も保持する。
  (一般にrノード)

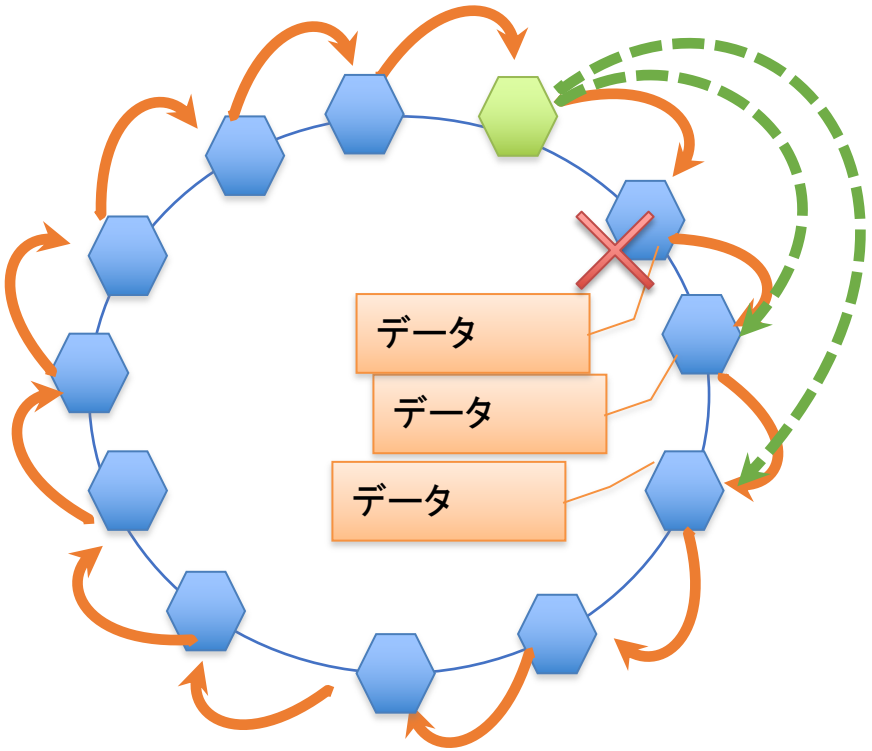
| | ノードID | IP |
|---|---|---|
| Successor[1] | 次 | … |
| Successor[2] | 次の次 | |
| Successor[3] | 次の次の次 | |
| Finger[1] | $2^0$先 | … |
| Finger[2] | $2^1$先 | … |
| … | … | … |
| Finger[m] | $2^{m-1}$先 | … |



# 安全性向上：Data Replication

■担当ノードが持っているデータを、
次のノード、次の次のノードにも持たせる。

■ノードが故障しても
ノードへの到達性保証だけでなく、
データへの到達性も保証される。



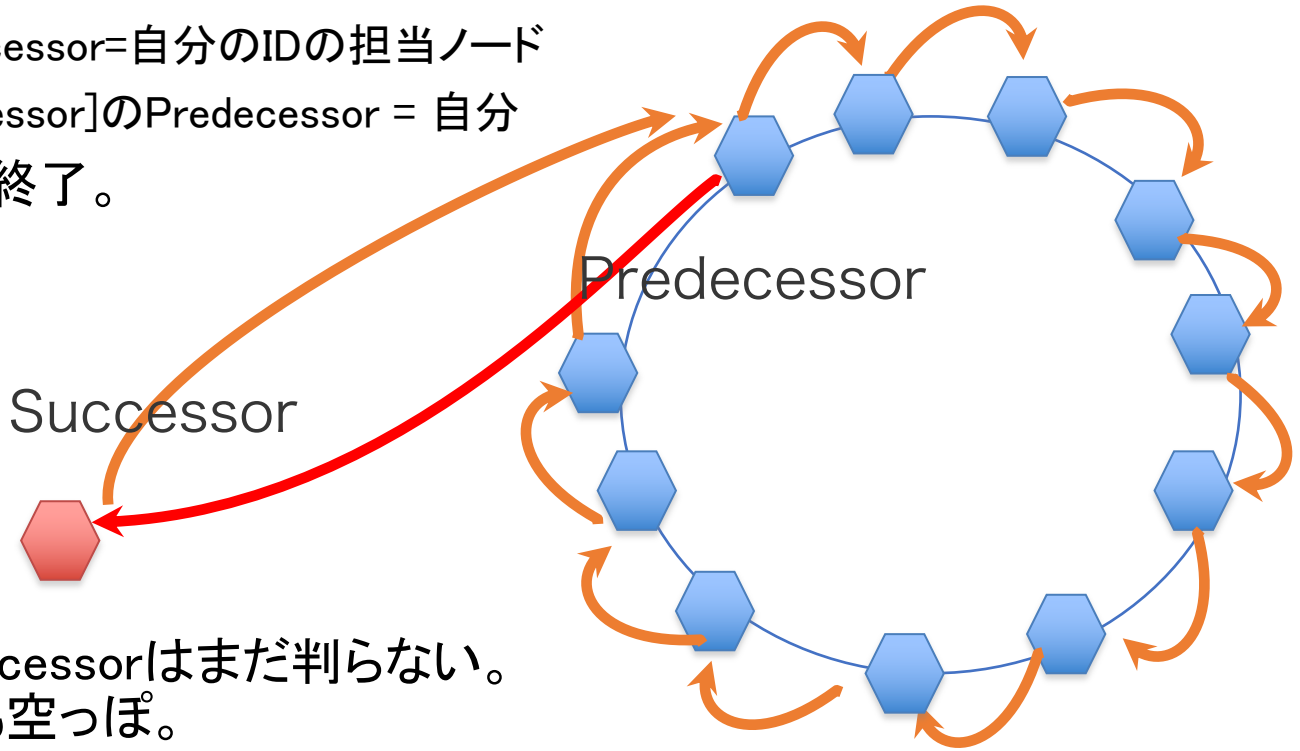
# 新規ノードの参加処理：Join

- ノードの追加に対応するために、補助的な経路 Predecessor を加える。
- 参加したいノードは、少なくとも一つのノードのIPアドレスを知っている。
- JOINでは、以下の2つの情報を更新

  [自分]のSuccessor=自分のIDの担当ノード

  [自分のSuccessor]のPredecessor = 自分

  Joinはここで終了。




- 自分のPredecessorはまだ判らない。
  FingerTableも空っぽ。
- しかし、既存ネットワークの
  ルーティングは維持されている。

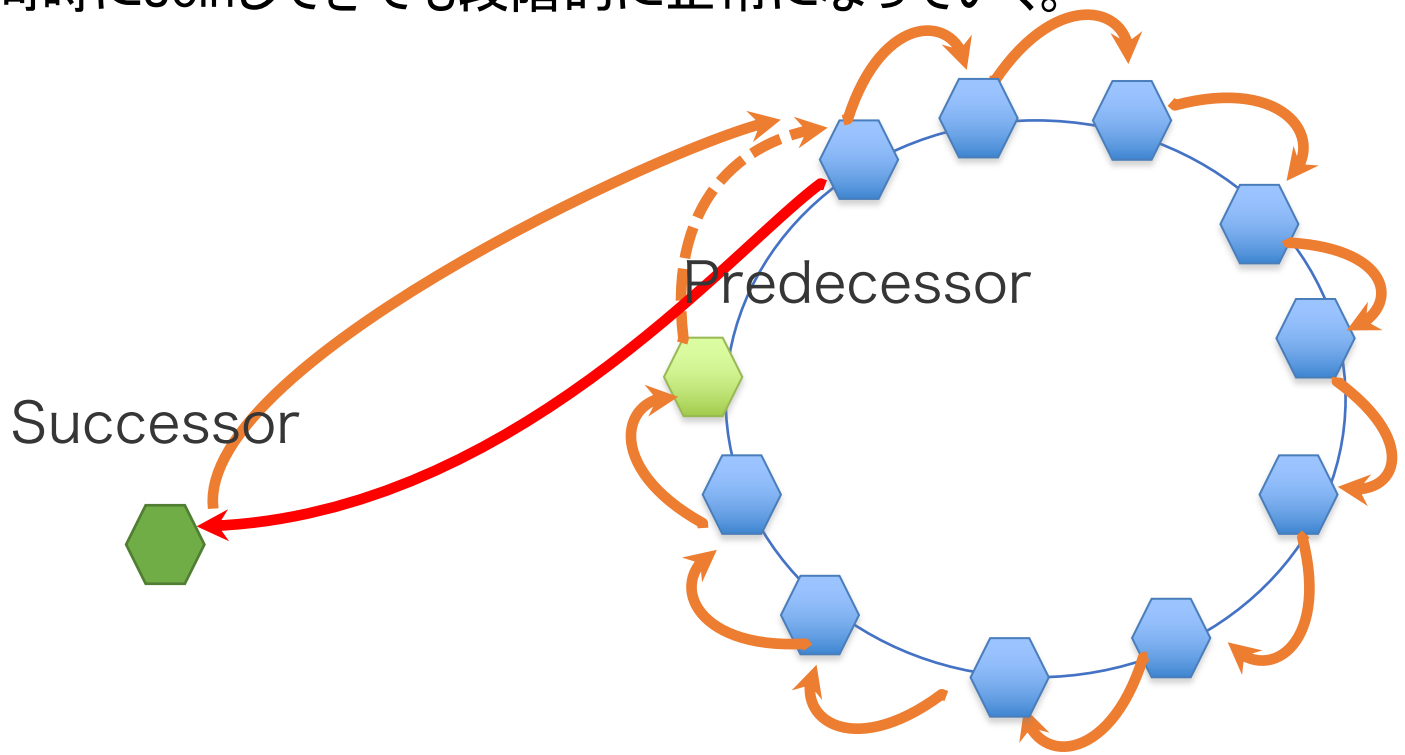
# メンテナンス処理：Stabilize

- 以下の2つをそれぞれを定期的に実行する
  - Successor用のStabilize
  - FingerTable用のStabilize
- 並列に実行される大量のJOINを、既存ネットワークのルーティングを妨げずに正しく完了できる.

| | ノードID | IP |
|---|---|---|
| Successor[1] | 次 | … |
| Successor[2] | 次の次 | |
| Successor[3] | 次の次の次 | |
| Finger[1] | $2^0$先 | … |
| Finger[2] | $2^1$先 | … |
| … | … | … |
| Finger[m] | $2^{m-1}$先 | … |

# Successor用Stabilize

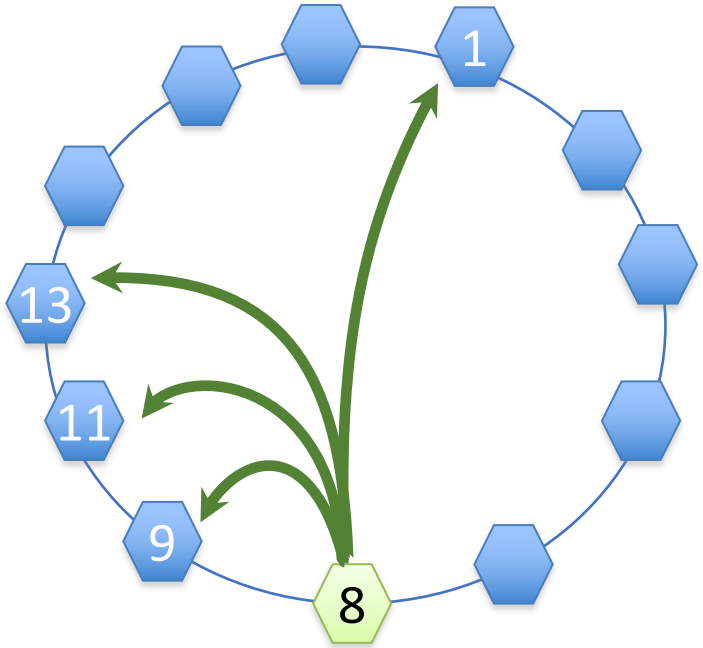
- 自分のSuccessorに,Predecessorを問い合わせる。
  間違いがあったら自分のSuccessor を更新する。
- 複数ノードが同時にJoinしてきても段階的に正常になっていく。



# FingerTable用Stabilize

- m個のFingerTableのうち、ランダムにi番目を選択し、「自分のID+$2^{i-1}$」をの担当ノードを探索し、情報を更新。

| | ノードID | IPアドレス |
|---|---|---|
| Finger[1] | 9 | … |
| Finger[2] | 11 | … |
| Finger[3] | 13 | … |
| Finger[4] | 1 | … |



# Chordの特徴

- 環状のID空間 （リングネットワーク）
- 経路表
  - Successorによる到達性保証
  - FingerTableによる経路短縮
  - Predecessor （Join用）
- 新規参加
  - Join
- 経路表の保守
  - Stabilize の定期的実行によるupdate

# 今日のまとめ

- 負荷分散手法
  - ミラーサーバ
  - ロードバランサ
  - ラウンドロビンDNS
- P2Pにおける資源探索（と匿名性）
  - Gnutella／Flooding
  - Freenet
- DHT (Distributed Hash Table)
  - Chordの例