ネットワーク アーキテクチャ

システムアーキテクチャ

- エンド ツー エンド
  - トランスペアレントな基盤の上に、自由なサービスの展開

  (Q. Proxy/Cache は、エンド・ツー・エンドか?)
- ピア・ ツー・ ピア
  - 情報通信機器 が主役の アーキテクチャ
- クライアント サーバ
  - サービスプロバイダが主役のアーキテクチャ

CS vs P2P

(client-server) vs (Peer-to-Peer)

- とちらも、”Transparent” な情報通信基盤
- “Server” は、「点」である必要はない。
  - “Server” のネットワーク化
  - “Proxy/Cache”もネットワーク化の一種???
- Client-Server
  - “Server” での機能/処理の共有
    - コスト削減 、高品質サービス、サービスの継続性
  - ISPもIT部門設備(企業/大学)も “Server” の一つ
- Peer-to-Peer
  - すべての機器が、サーバにもクライアントにもなる。

放送、電話、インターネットの技術比較

	放送	インターネット	電話
クライアント・サーバ or ピア・ツー・ピア	クライアント・サーバ	ピア・ツー・ピア	ピア・ツー・ピア
エンド・ツー。エンド or ゲートウェイ	ゲートウェイ	エンド・ツー・エンド	ゲートウェイ
オーバレイ or ピア	ピア	オーバレイ	ピア
保証型 or ベストエフォート型	保証型	ベストエフォート型	保証型 (+ベストエフォート)
シグナリング	なし	インバンド	アウトバンド
ハードステート or ソフトステート	なし	ソフトステート	ハードステート

シグナリング(制御信号)

- アウトバンドシグナリング
  - ユーザデータ(Data-Plane) と 制御データ(Control-Plane) が分離している。
  - 例: 電話、MP3プレイヤ、放送
- インバンドシグナリング
  - ユーザデータ(Data-Plane) と 制御データ(Control-Plane) が縮退している。
  - 例: インターネット

状態管理 ポリシー

- ハードステート
  - コネクション型サービスにおいては、通信を開始する前に、シグナリング手順を実行し、通信する情報機器の間に、仮想的な通信回線(＝コネクション)を確立する。多くの場合(インターネットにおけるTCPは例外とみることができる)、データが転送される経路は、シグナリング時に決定され、通信路が解放されるまで、同一の経路が維持されるのが、一般的。ハシステムの状態は、インスタンスが発生して、消滅するまで状態を変化させない。電話サービスにおいては、電話網内でコネクションが利用する経路上の情報機器(＝交換機)や通信回線に障害が発生した場合、基本的には、通信を継続することができない。

状態管理 ポリシー(続)

- ソフトステート
  - コネクションレス型サービスにおいては、シグナリング手順を持たず、ネットワークの状態に応じて、必要な時には、ディジタル小包の転送経路を動的に変化させる。システムの状態を、常に更新(Update)しながら運用するシステムを、ソフトステートのシステムと呼ぶ。典型例は、TCP/IP技術を用いたインターネット。IPパケットの転送経路は、動的に変化することを前提として、データ通信手法が設計されているために、ある通信路やルータに障害が発生しても、自動的に、IPパケットの転送サービスを実現可能な他の経路を探し出し、サービス提供を継続させる。

インターネットの実体

- 1960年代: コンピュータ”らしき” もの
- 1970年代: UNIXの開発、まだ”鎮座”する計算機
- 1980年代: 僕の体重の数倍の計算機
- 1990年代: パソコン
- 2000年代: 携帯電話とゲーム機
- 2010年代: ?????

「あちら側」と「こちら側」の議論

1. CS：メインフレーム
2. P2P：分散コンピューティング
3. CS：インターネット(ISP/ASP）
4. P2P：ファイル共有
5. CS： Google情報加工工場
6. **P2P：??????**

インターネットの実体

- 1960年代: コンピュータ”らしき” もの
  - IBM for 政府
- 1970年代: UNIXの開発、まだ”鎮座”する計算機
  - HP for 研究機関
- 1980年代: 僕の体重の数倍くらいの計算機
  - サンマイクロシステムズ for 大企業
- 1990年代: パソコン
  - マイクロソフト for ビジネスマン
- 2000年代: 携帯電話 とゲーム機
  - Google for everyone
- 2010年代: ?????

Peer-to-Peerシステムの役割

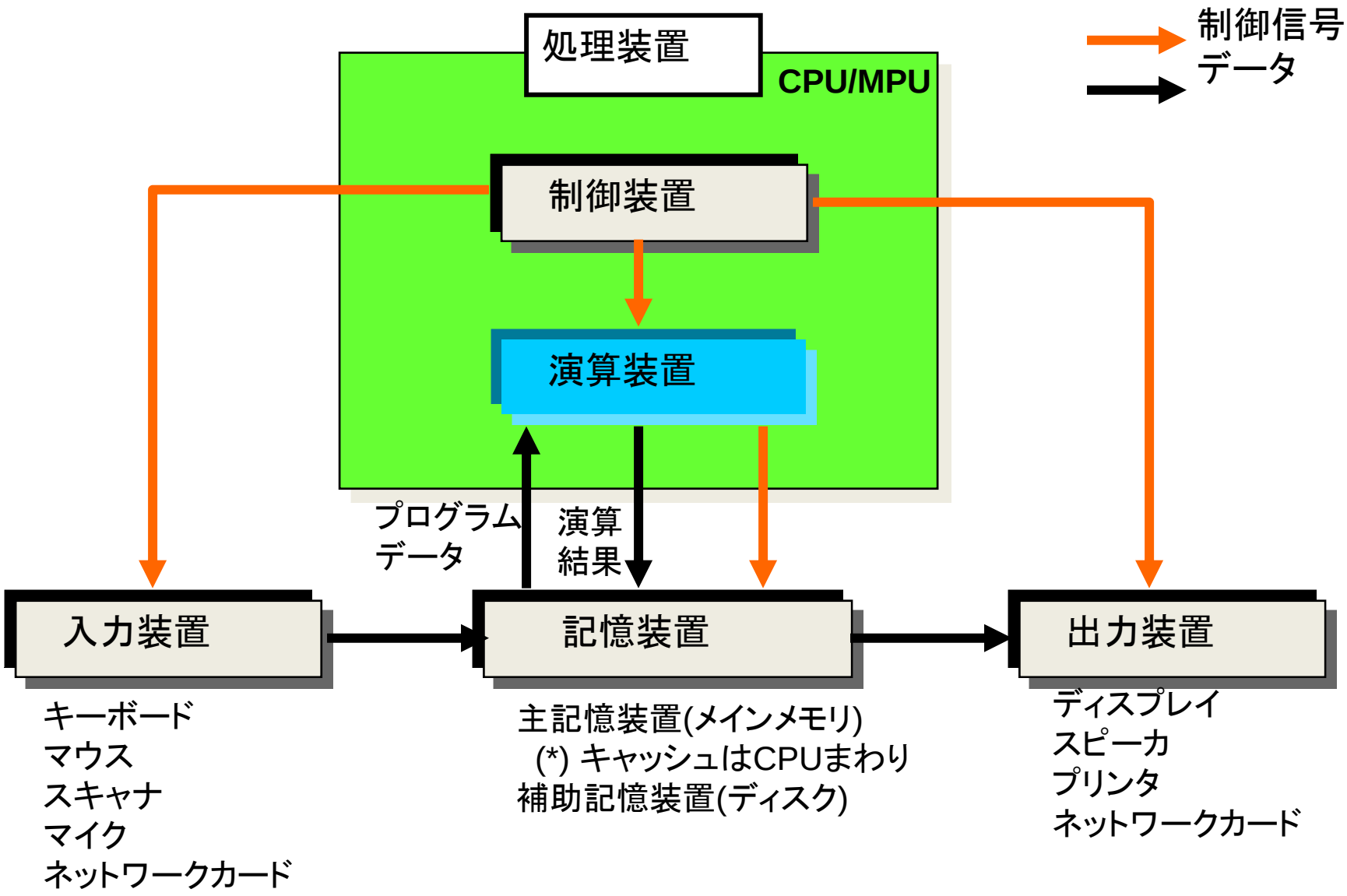
1. キャッシュ(Cache) と Proxy の導入
2. DMA (Direct Memory Access) の導入
3. 仮想記憶システムの導入 (by DHT)

   コンテンツハンドラ(識別子) と 実アドレスの分離
4. コンテンツの抽象化 (by DHT)

   {ファイル名、ファイル拡張子、等} を隠蔽し、単純な数値で表現。

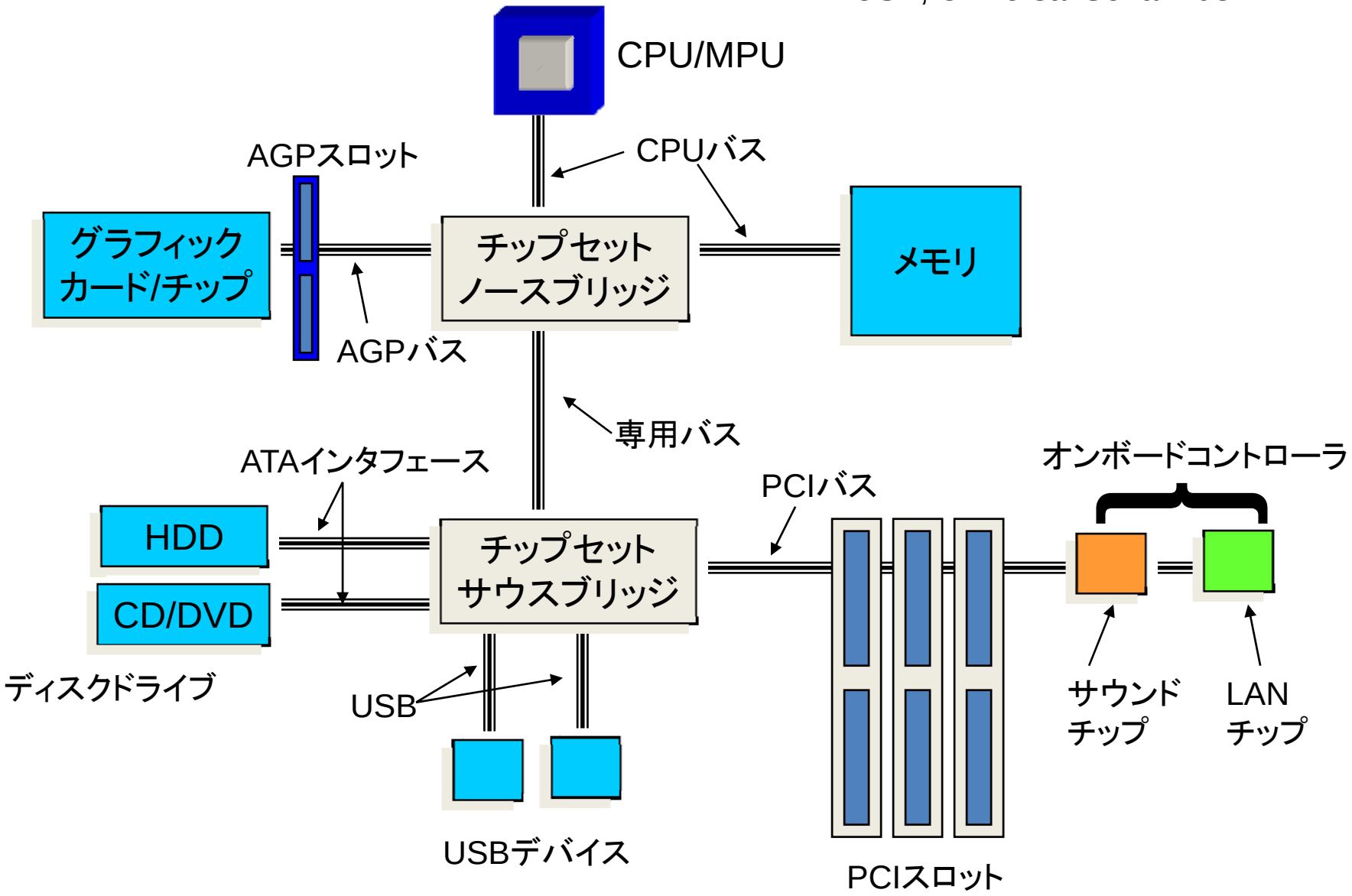
(*) 仮想メモリ
仮想的なメモリ機構によって生成される、仮想的なメモリ 領域(とても大きな記憶空間)。仮想メモリは、最終的には適当な物理メモリにマップされる。 物理メモリ量を超える仮想メモリ 空間を作り出したり、複数の仮想空間を作り出したりする。

コンピュータの基本構成

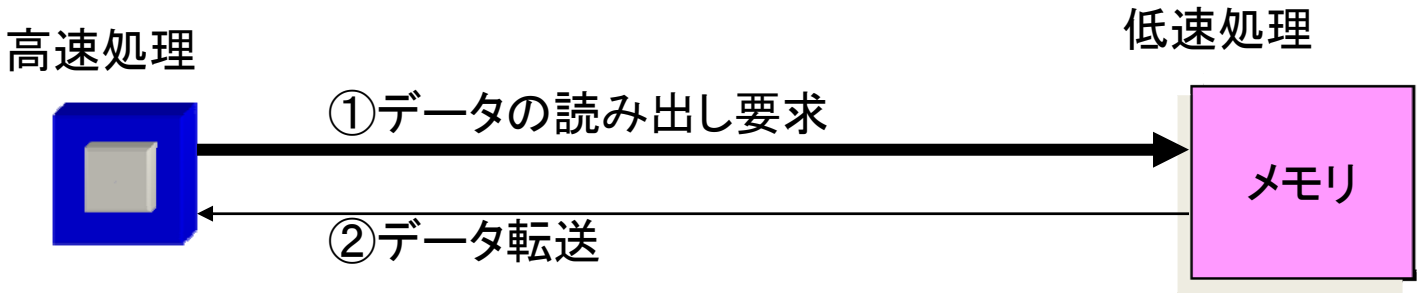


<< PCの基本構成 >>

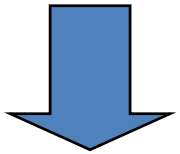
(*) AGP; Accelerated Graphics Port
PCI; Peripheral Component Interconnec
USB; Universal Serial Bus



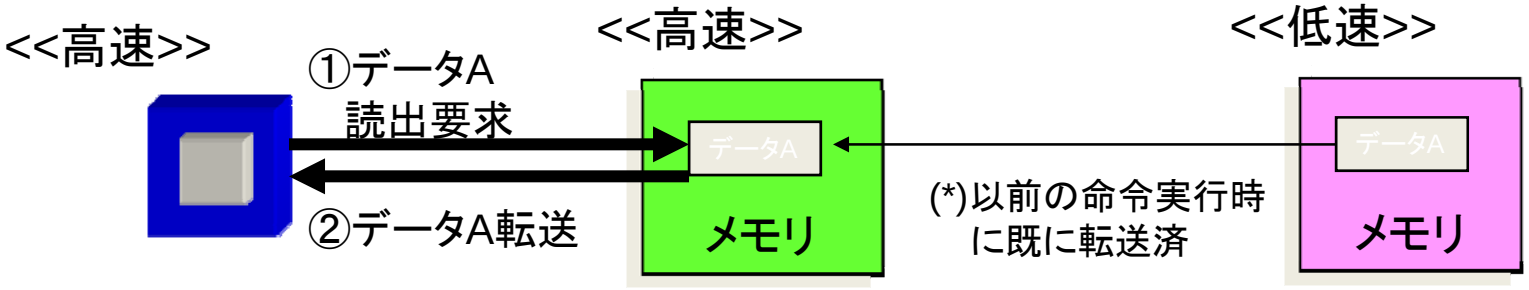
キャッシュメモリ



(*) ②が遅い、、、➔ CPUのアイドル時間が発生。。。。

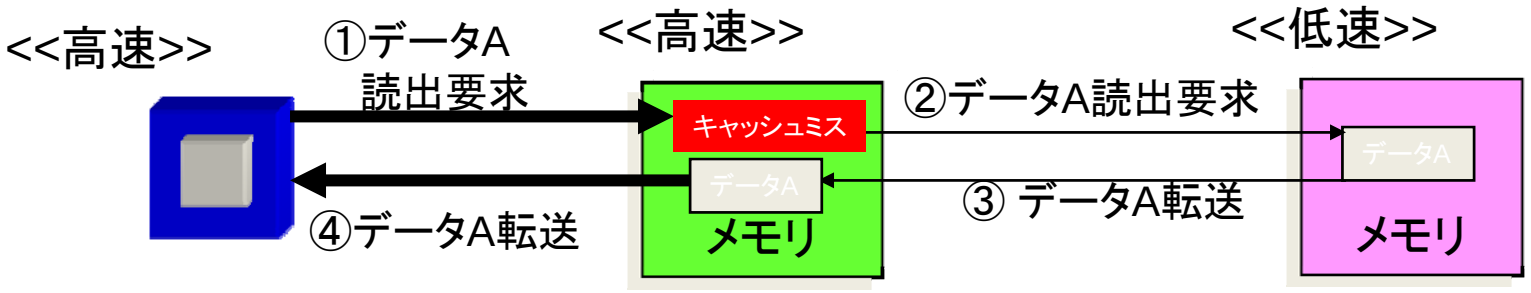


キャッシュメモリの導入

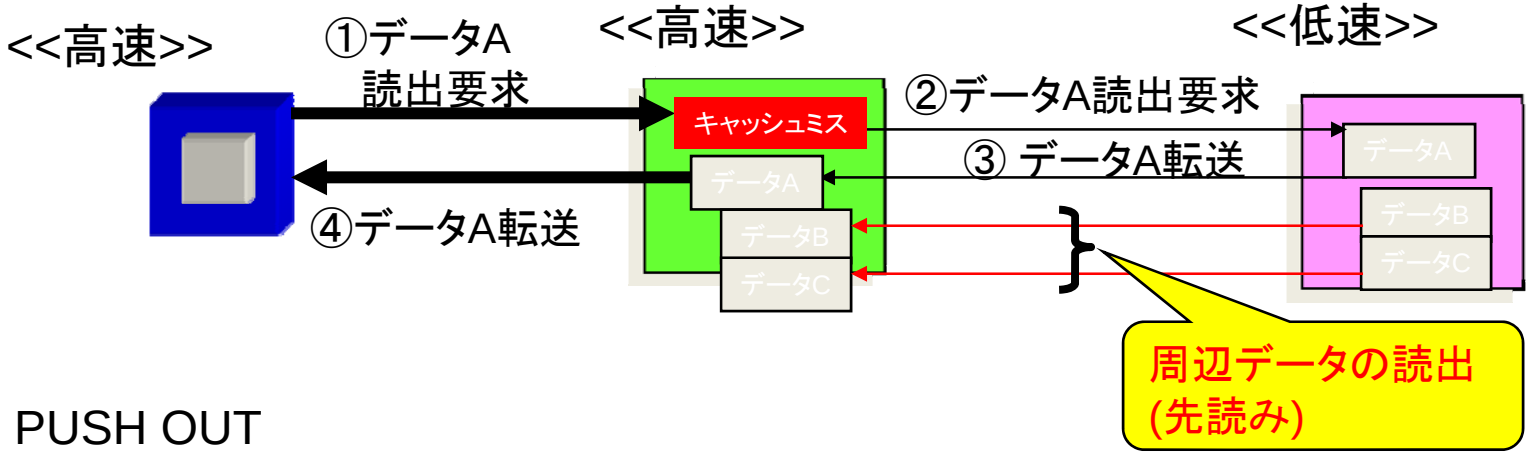


キャッシュメモリ(続)

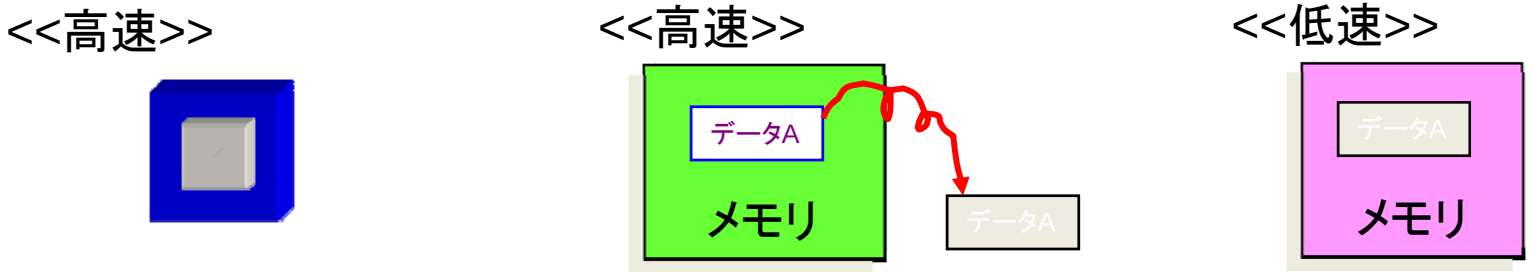
(1) キャッシュミス



(2) 先読み (Reverse Cache)

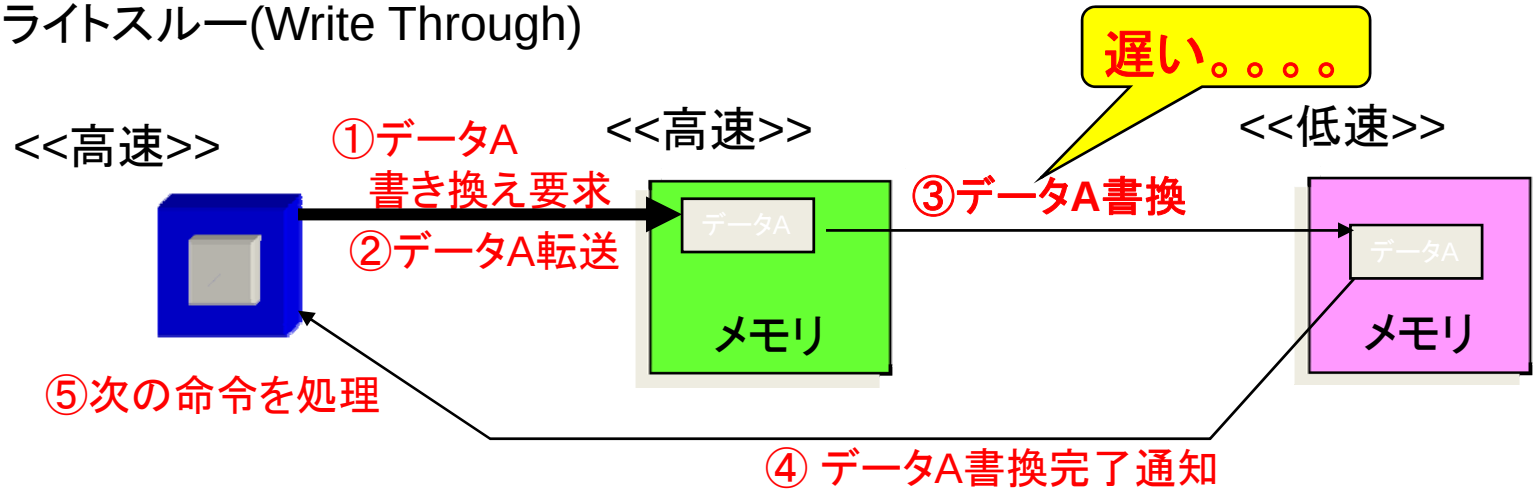


(3) PUSH OUT

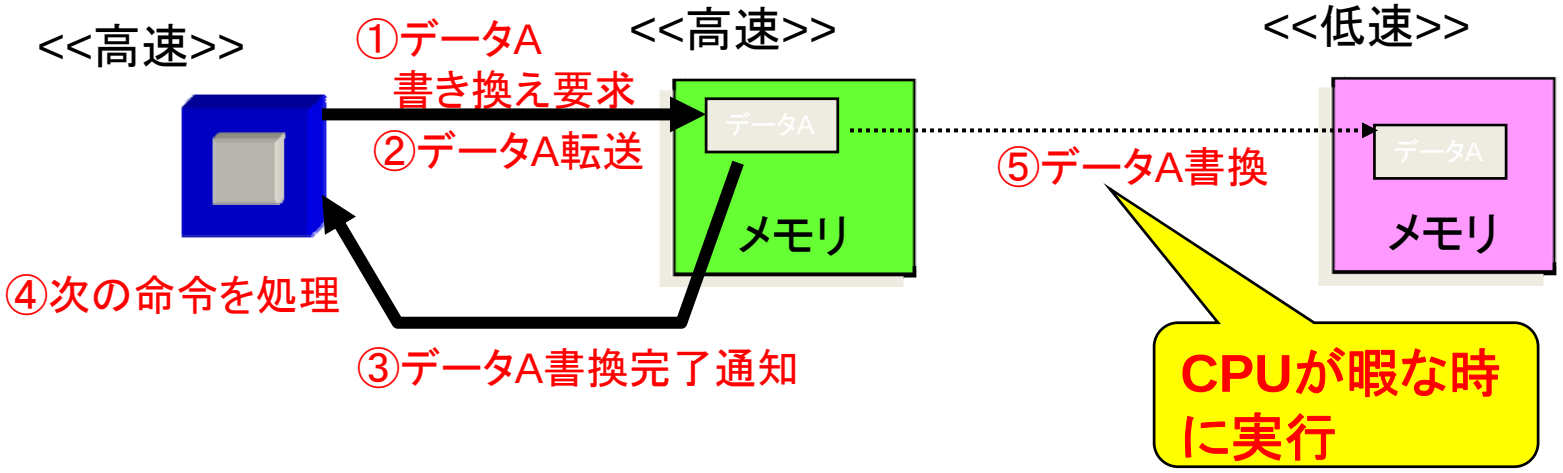


キャッシュメモリ(書き換え)

(1) ライトスルー(Write Through)



(2) ライトバック (Write Back)



(*) データの不一致に注意が必要！

Peer-to-Peerシステムの役割

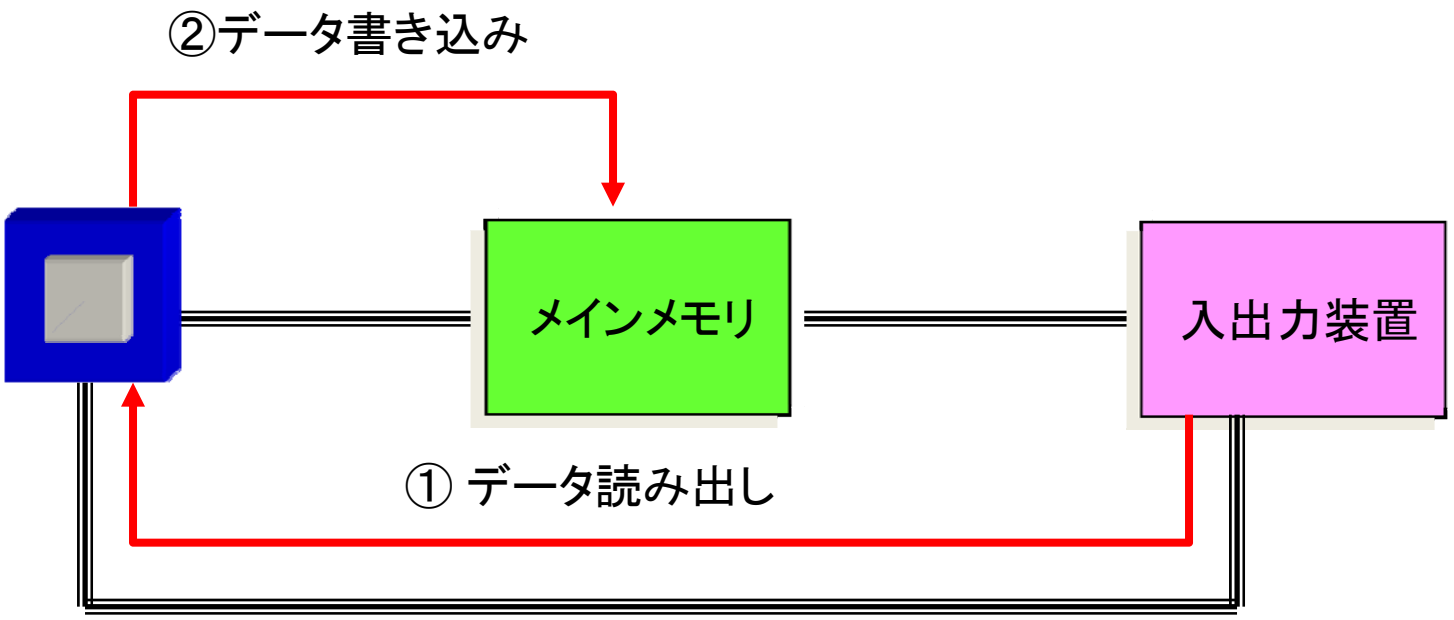
1. キャッシュ(Cache) と Proxy の導入
2. DMA (Direct Memory Access) の導入
3. 仮想記憶システムの導入 (by DHT)

   コンテンツハンドラ(識別子) と 実アドレスの分離
4. コンテンツの抽象化 (by DHT)

   {ファイル名、ファイル拡張子、等} を隠蔽し、単純な数値で表現。

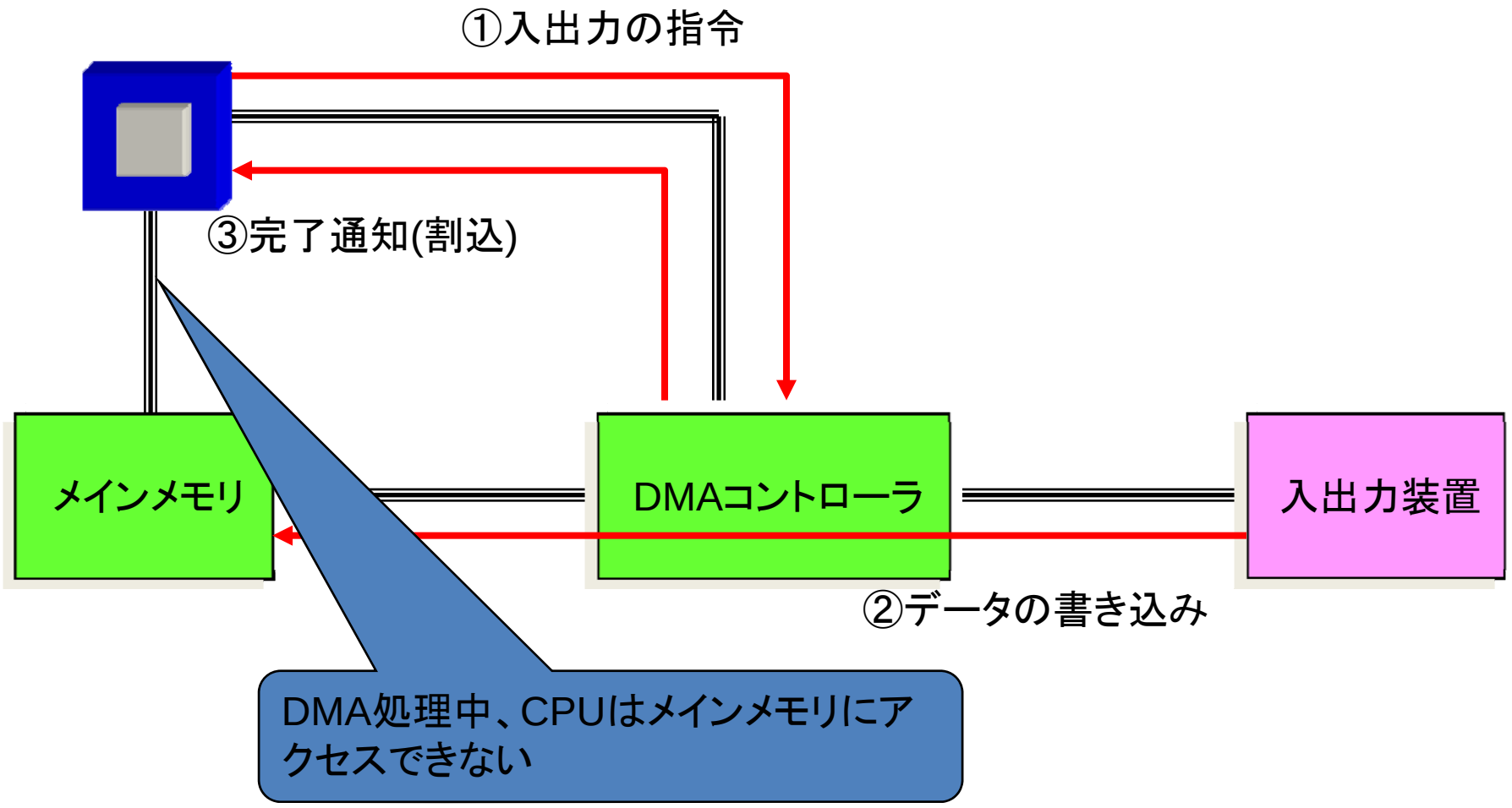
(*) 仮想メモリ
仮想的なメモリ機構によって生成される、仮想的なメモリ 領域(とても大きな記憶空間)。仮想メモリは、最終的には適当な物理メモリにマップされる。 物理メモリ量を超える仮想メモリ 空間を作り出したり、複数の仮想空間を作り出したりする。

直接制御方式



(*) CPUは、データの読み出し、書き込みが
完了するまで、他の処理をできない

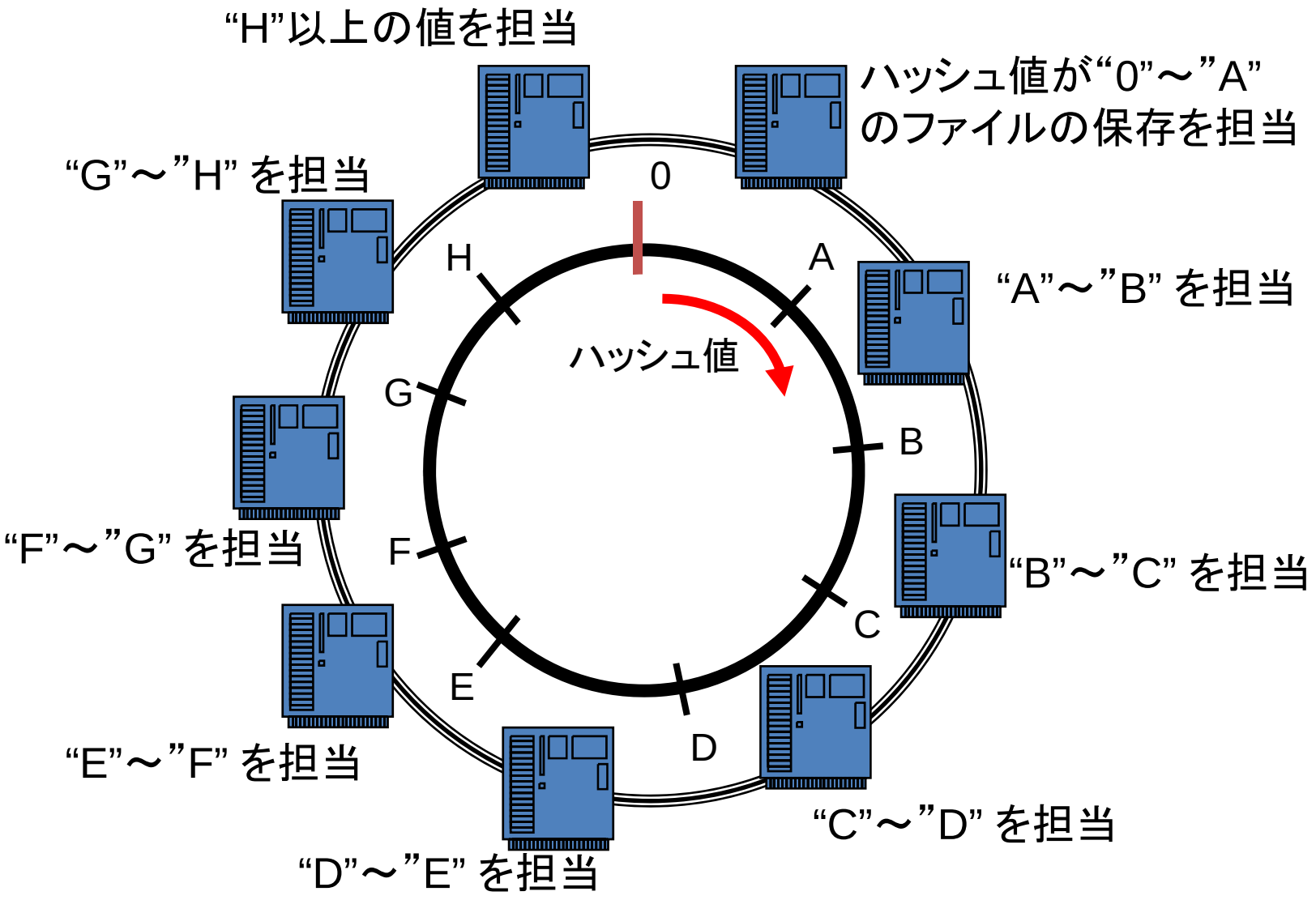
DMA方式


①入出力の指令
③完了通知(割込)
メインメモリ
DMAコントローラ
入出力装置
②データの書き込み
DMA処理中、CPUはメインメモリにアクセスできない

Peer-to-Peerシステムの役割

1. キャッシュ(Cache) と Proxy の導入
2. DMA (Direct Memory Access) の導入
3. **仮想記憶システムの導入 (by DHT)**
   コンテンツハンドラ(識別子) と 実アドレスの分離
4. **コンテンツの抽象化 (by DHT)**
   {ファイル名、ファイル拡張子、等} を隠蔽し、単純な数値で表現。

(*) 仮想メモリ
仮想的なメモリ機構によって生成される、仮想的なメモリ 領域 (とても大きな記憶空間)。仮想メモリは、最終的には適当な物理メモリにマップされる。 物理メモリ量を超える仮想メモリ 空間を作り出したり、複数の仮想空間を作り出したりする。



DHTにおける 分散ファイル保存

P2P型ファイル分散共有

- Napster
  - IndexとStorageを分離
    - アウトバンドシグナリングの導入による DMA的動作
    - ポインター渡しによる コピーオーバーヘッドの削減
- Freenet/Winny
  - ファイル名とLocationを分離
    - 仮想メモリ的動作
  - ファイル保有・送信・受信の匿名性を実現した
    - 仮想メモリ的 動作
  - ネットワークキャッシュの導入
    - ページング、キャッシュ、リバースキャッシュ、先読み

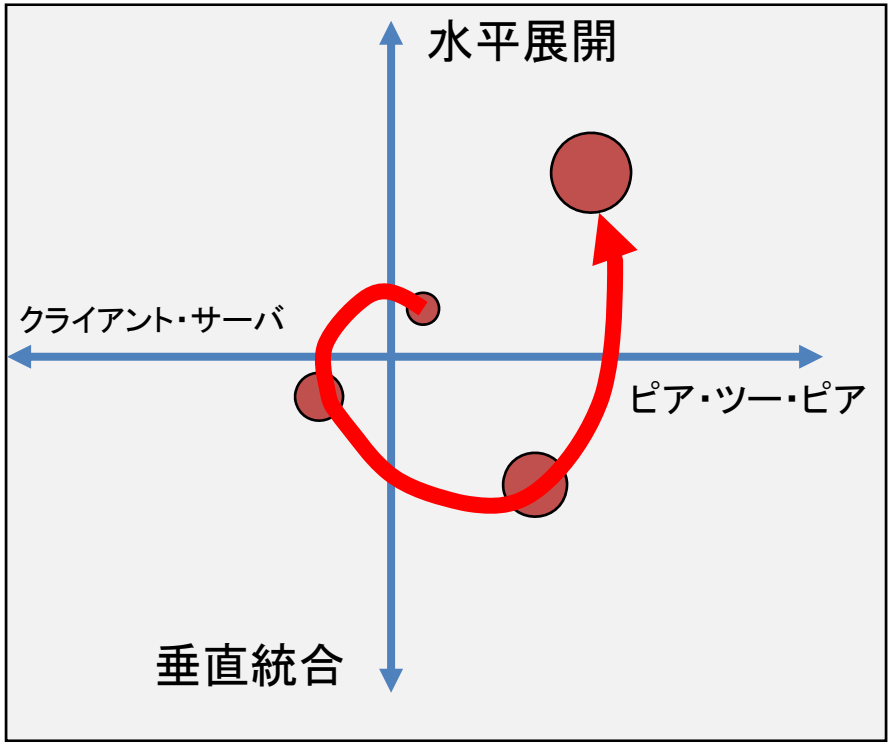
典型例

- Peer-to-Peerシステム
  - 電話システム
- Client-Serverシステム
  - 放送システム

- 混在システム
  - インターネット

通信モデル

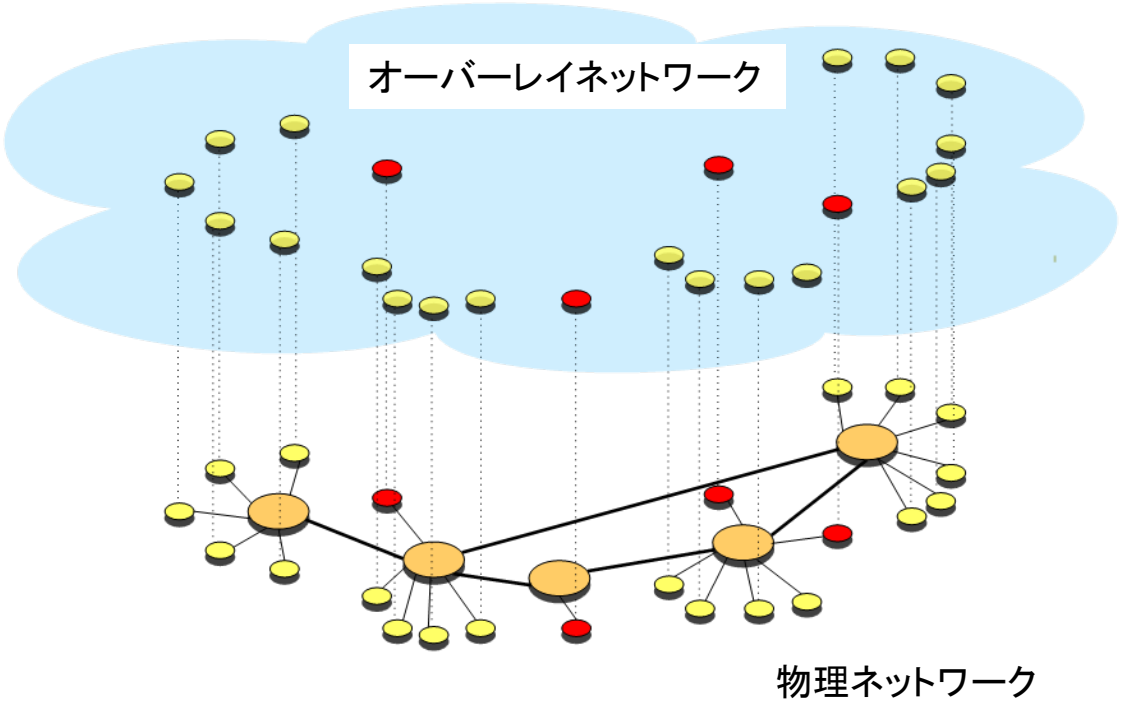
- client - serverモデル
  - 二者間の通信モデル
    - 最初にどのように通信をはじめるかの役割分担
      - 能動側と受動側
- peer to peerモデル
  - 二者間の通信モデル
    - 機能的には全てのエンティティがクライアントとサーバを兼ねる
    - 全く対等な関係でどちらからも通信をはじめられる
- End to Endモデル
  - 二者間の通信モデル
    - 代理人を介さず直接通信の両端がコミュニケーションする
    - インターネットの大原理



ピア・ツー・ピア と クライアント・サーバの正のスパイラル

Proxy、Overlay、Gatewayの違い

- Overlay： 論理的なネットワークを、あるプラットフォーム(e.g., IPネットワーク) の上に仮想的に構築する。
  ➔ システムの最適化が難しい。。。。。
- Proxy： “Transparent” な通信を、Intermediate Nodeで、なりすましをしたり、盗み見をしたりして、効率を向上させる。
  ➔ 効率向上は、データの移動パターンに大きく依存
- Gateway： 違うプロトコル(=言語)を翻訳変換してあげる。
  ➔ もっとも、非効率。。。。大きなシステム同士を相互接続させる場合には、巨大な Stateful Machine を必要とする場合が多い。



オーバーレイネットワーキングの概念図

Proxy、Overlay、Gatewayの違い

- Overlay： 論理的なネットワークを、あるプラットフォーム(e.g., IPネットワーク) の上に仮想的に構築する。

  ➔ システムの最適化が難しい。。。。。

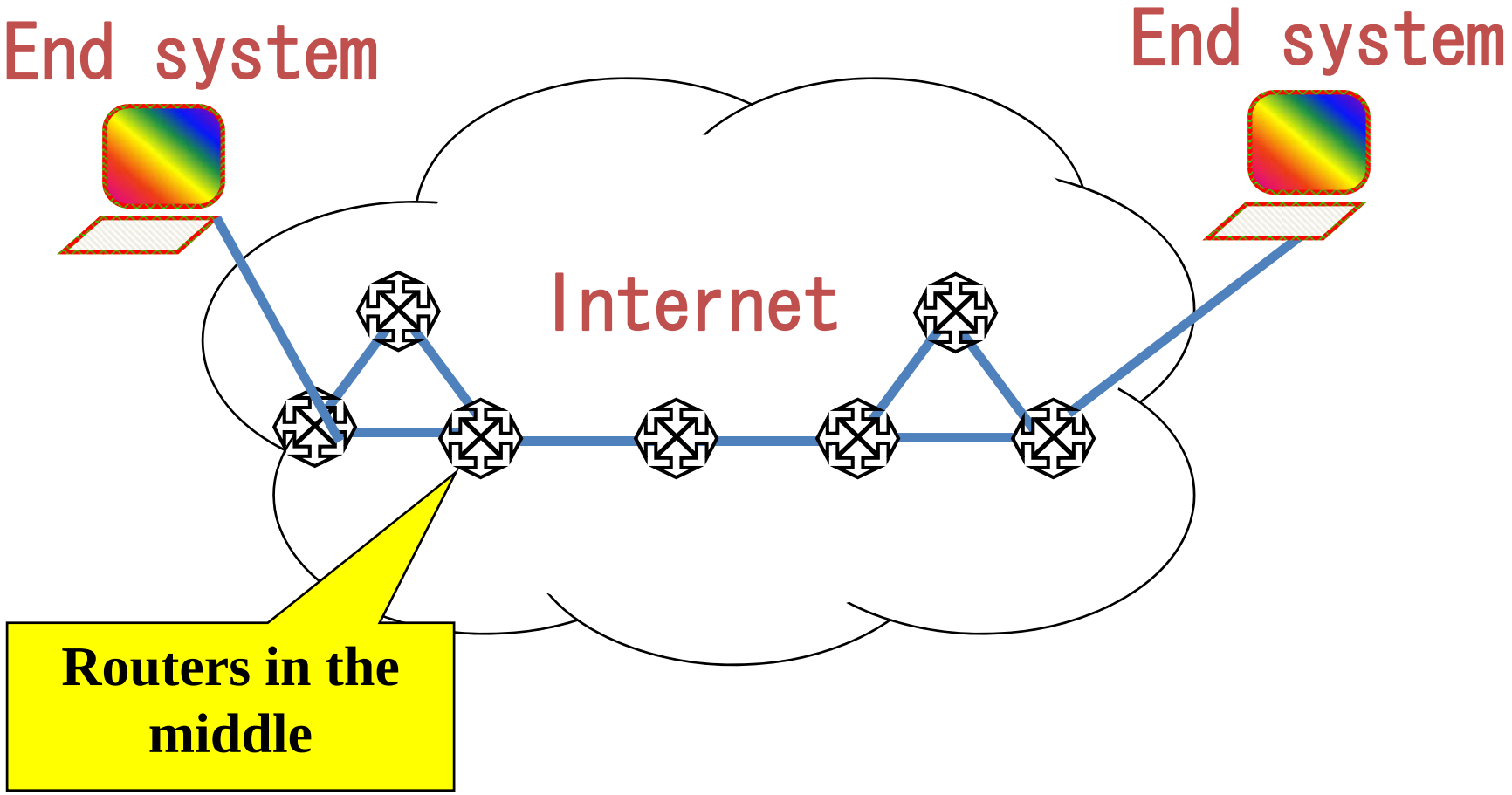
- Proxy： “Transparent” な通信を、Intermediate Nodeで、なりすましをしたり、盗み見をしたりして、効率を向上させる。

  ➔ 効率向上は、データの移動パターンに大きく依存

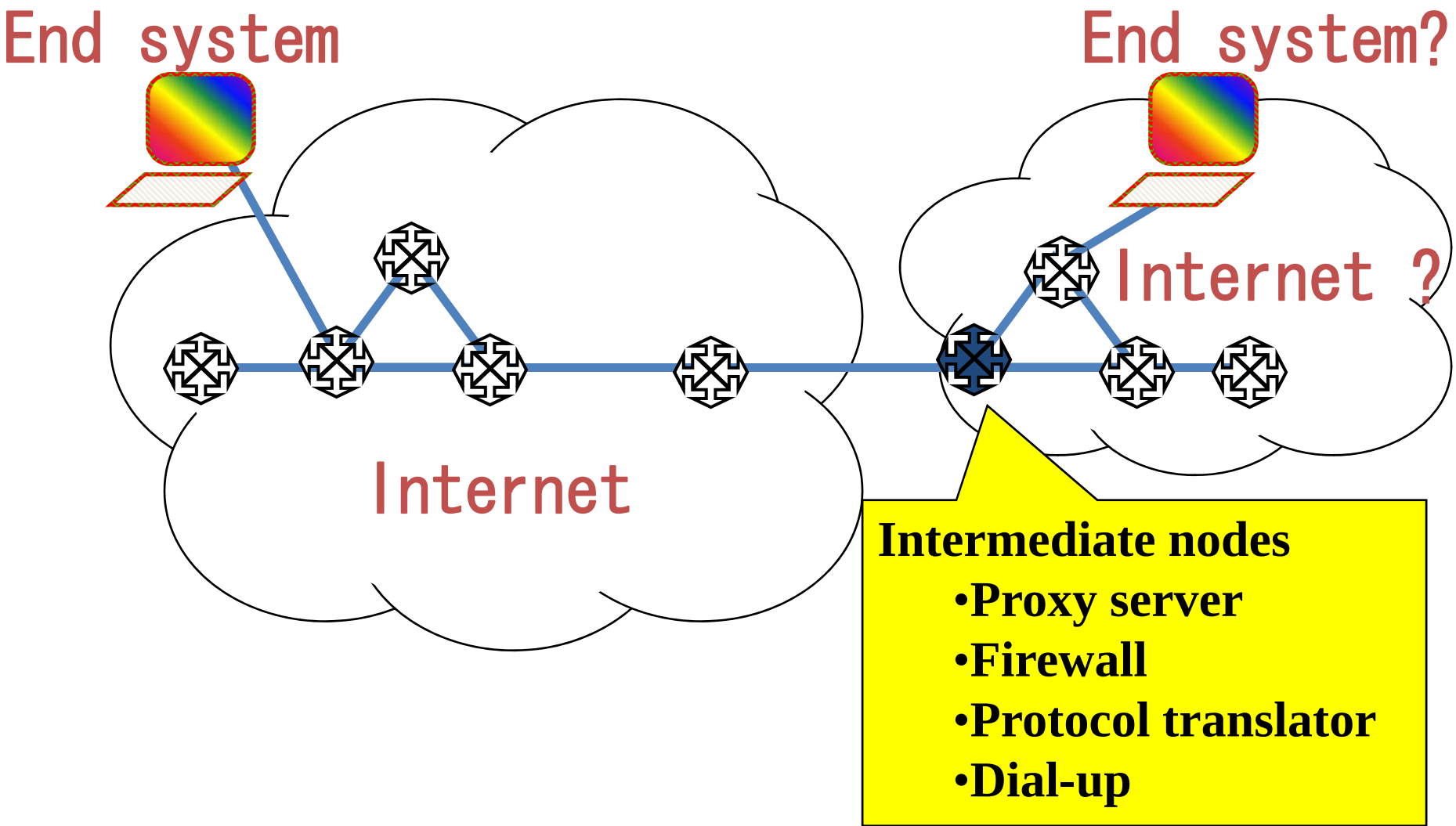
- Gateway： 違うプロトコル(=言語)を翻訳変換してあげる。

  ➔ もっとも、非効率。。。。大きなシステム同士を相互接続させる場合には、巨大な Stateful Machine を必要とする場合が多い。

“End-to-End Model”

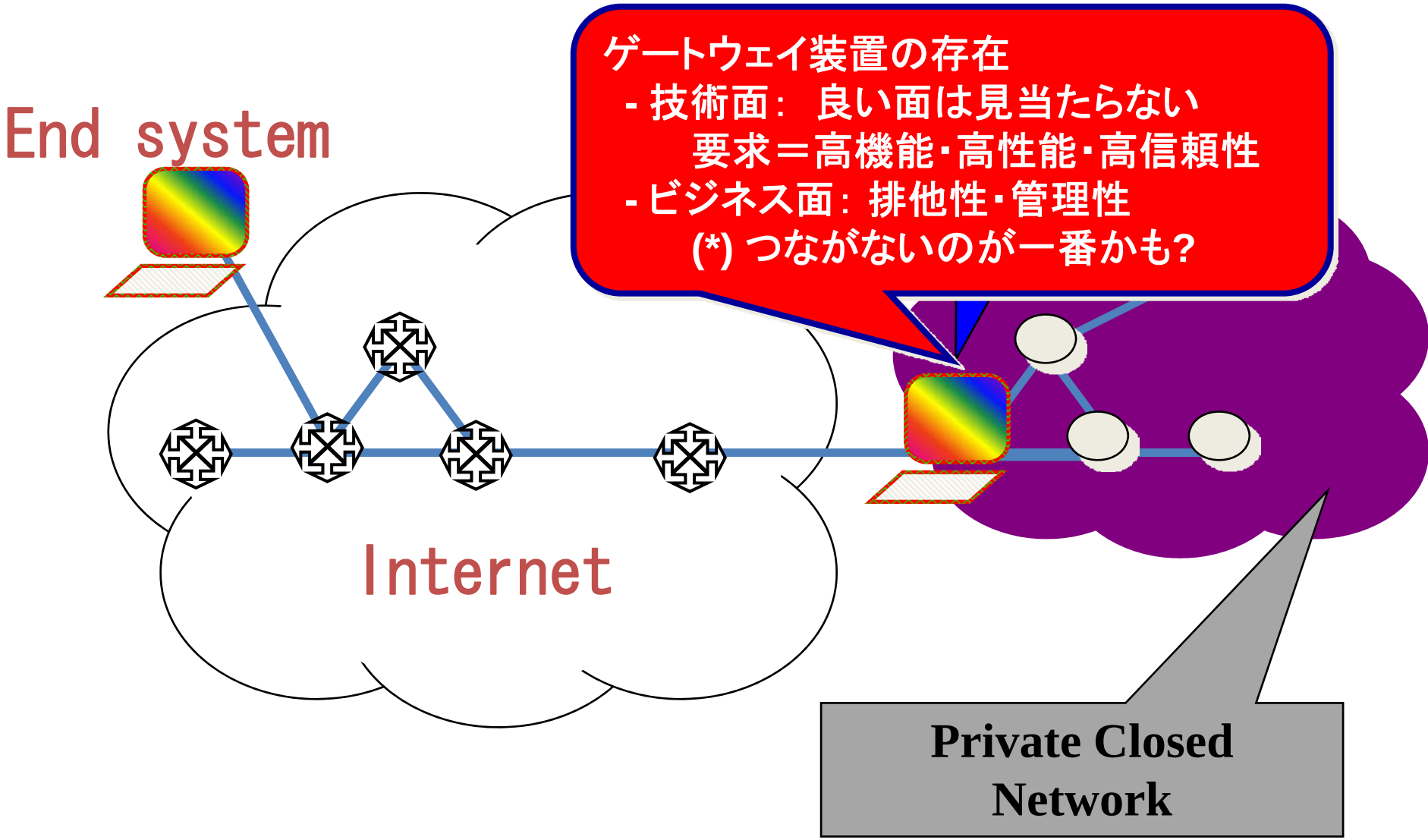


ネットワークを分割したくなった?



大変な苦労を抱え込みます。

(*)でも“苦労”は、ビジネスチャンス?



Proxy、Overlay、Gatewayの違い

- Overlay： 論理的なネットワークを、あるプラットフォーム(e.g., IPネットワーク) の上に仮想的に構築する。

  ➔ システムの最適化が難しい。。。。。

- Proxy： “Transparent” な通信を、Intermediate Nodeで、なりすましをしたり、盗み見をしたりして、効率を向上させる。

  ➔ 効率向上は、データの移動パターンに大きく依存

- Gateway： 違うプロトコル(=言語)を翻訳変換してあげる。

  ➔ もっとも、非効率。。。。大きなシステム同士を相互接続させる場合には、巨大な Stateful Machine を必要とする場合が多い。

Overlay Model vs Peer Model

- Overlay Model
  - 自由なトポロジーを定義・形成可能
  - 自由なプロトコルを定義・運用可能

- Peer Model
  - トポロジーを直接反映させた最適化が可能
  - システムの 動作とCritical Point の把握が可能

最近の 傾向

1. マルチプロセッサ型の計算機アーキテクチャの導入

   分散処理(機能分散、地理的分散)

2. キャッシュ技術の導入

   CDN、P2Pなど

3. 仮想化技術の導入

   Virtulization、Overlayネットワーク

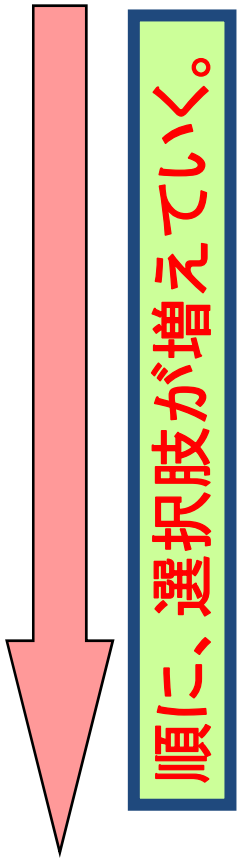
処理負荷の分散手法

1. 水平分散
   - 複数のサーバに並列的に処理を実行させる。
   - 特定の機能ごとに、専門化したサーバを用意し、適切に Dispatch する。
2. 垂直分散
   - Proxy機能や Cache機能を提供し、クライアントに近い仮想ノード/分身ノードが、データ処理を分担する。
   - データの変更時の Consistency の維持が課題となる。

技術の上下関係。。。。。

- フレームワーク
  - どのような 方針でシステムを設計・動作さすか。
- アーキテクチャ
  - どのような 原理/技術 でシステムを構築するか。
- プロトコル、インターフェース
  - 具体的な、仕組み、決め事、アルゴリズム
- 実装
  - 具体的にどのように実現するか。

電話サービスを例にして。



- フレームワーク
  - エンドーエンドに専用の通信パイプを提供する。
- アーキテクチャ
  - 空間分割 ➔ 時分割多重 ➔ パケット多重
- プロトコル
  - お願い(声) ➔ ダイヤル ➔ 小包の宛先札
- 実装
  - 手作業接続 ➔ 交換機 ➔ パケットスイッチ