The Guaranteed Network

いちばん近くで、もっと先へ。

パケットフォワーディングを支える技術
（2）ハードウェア処理ルータの内部詳解

2012年11月20日

アラクサラネットワークス株式会社

製品開発部

The Guaranteed Network

AlaxalA

自己紹介

はじめに

- ハードウェア処理ルータといえば
  - ワイヤーレートでのパケット処理が可能
- 使われている要素技術レベルで見るとサーバとの差は無い
  - SRAM：キャッシュメモリ ⇔ パケットバッファ
  - DRAM：メインメモリ ⇔ パケットバッファ、テーブルメモリ
  - CAM：キャッシュメモリ, TLB検索 ⇔ 転送テーブル検索
  - SerDes：PCI Express ⇔ XAUI, XLAUI, CAUI, Interlaken etc
  - クロスバースイッチ：CPU(コア)間接続 ⇔ プロセッサ間接続
  - 高速化手法：マルチコア / スーパースケーラ ⇔ 処理のパラレル化
  - パイプライン：共通の技術
- 何が違うのか？
  - 柔軟性を犠牲にすることで性能の最大化

PCI ExpressはPCI-SIGの商標または登録商標です。

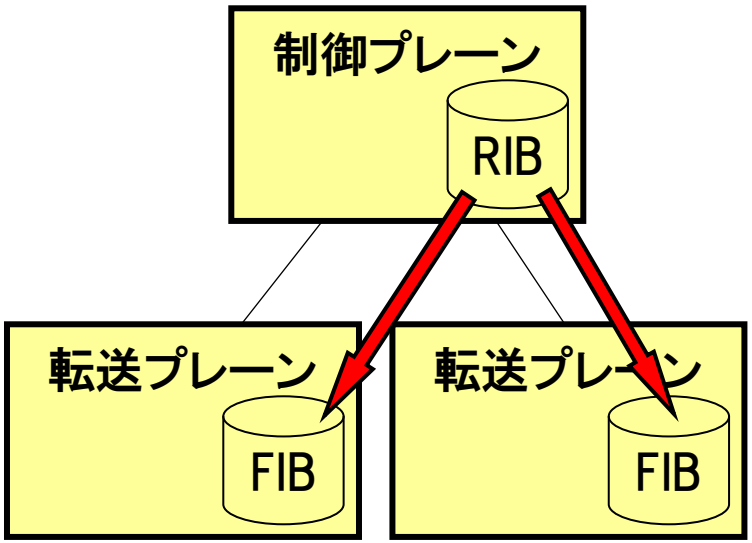
本セッションの内容

- ネットワーク装置におけるパケットのハードウェア処理
  - ハードウェア処理の基本的な考え方
  - 転送プレーンの構成方法
  - ハードウェアによるパケット転送処理の流れ
  - 高速テーブル検索手法

- 本セッションの対象外の内容
  - 経路テーブルの作成のためのプロトコル処理
  - 経路テーブルに基づいた転送テーブル作成・配布・アップデート
  - 経路収束性能の向上

ハードウェア処理の基本的な考え方

- 制御プレーンと転送プレーンの分離
  - 制御プレーンはソフトウェア処理に専念
    - 経路制御など
    - 高速性より柔軟性
  - 転送プレーンはハードウェア処理に専念
    - パケット転送など
    - 転送エンジンを増やすことで性能をn倍化
- 経路テーブル(RIB)と転送テーブル(FIB)の分離
  - 制御プレーンは他ルータ・スイッチと通信して経路情報 (BGP、OSPFなど)を交換し、RIBを構築
  - さらにRIBからFIBを算出し、転送プレーンに配布



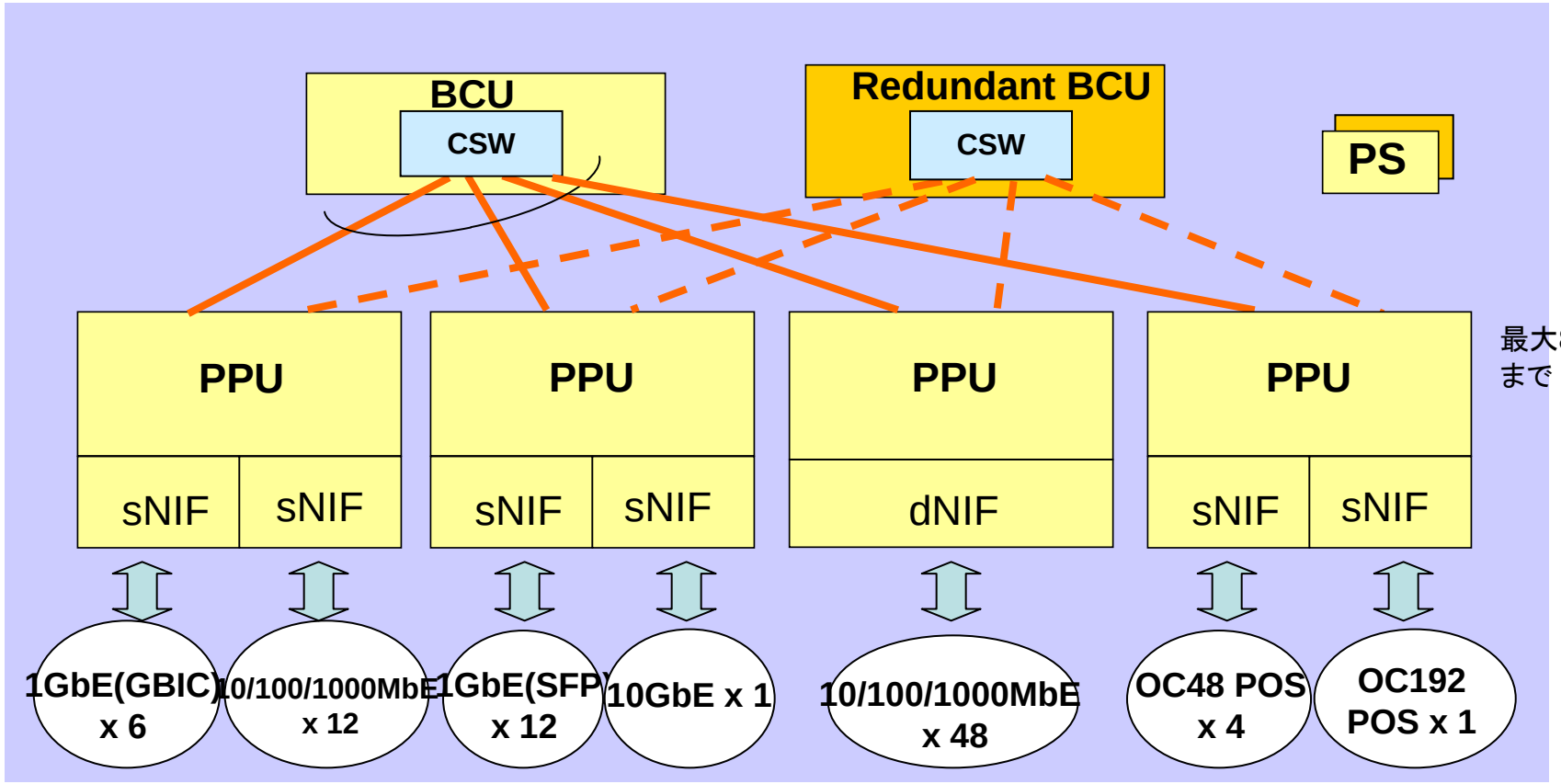
RIB：**Routing Information Base**
FIB: Forwarding Information Base

転送プレーンの配置

- 転送エンジンを複数持つことで転送性能をn倍化
  - 複数の転送エンジン持つと相互接続するスイッチが必要
    - 価格と性能のトレードオフ

	分散型	集中型
CPU	一つ	一つ
転送エンジン	複数	一つ
性能	転送エンジンを追加することで性能向上	一つの転送エンジンの性能で頭打ち
価格	高い	安い
主なターゲット	キャリア、データセンタ	企業

分散型アーキテクチャ（AX7800の例）

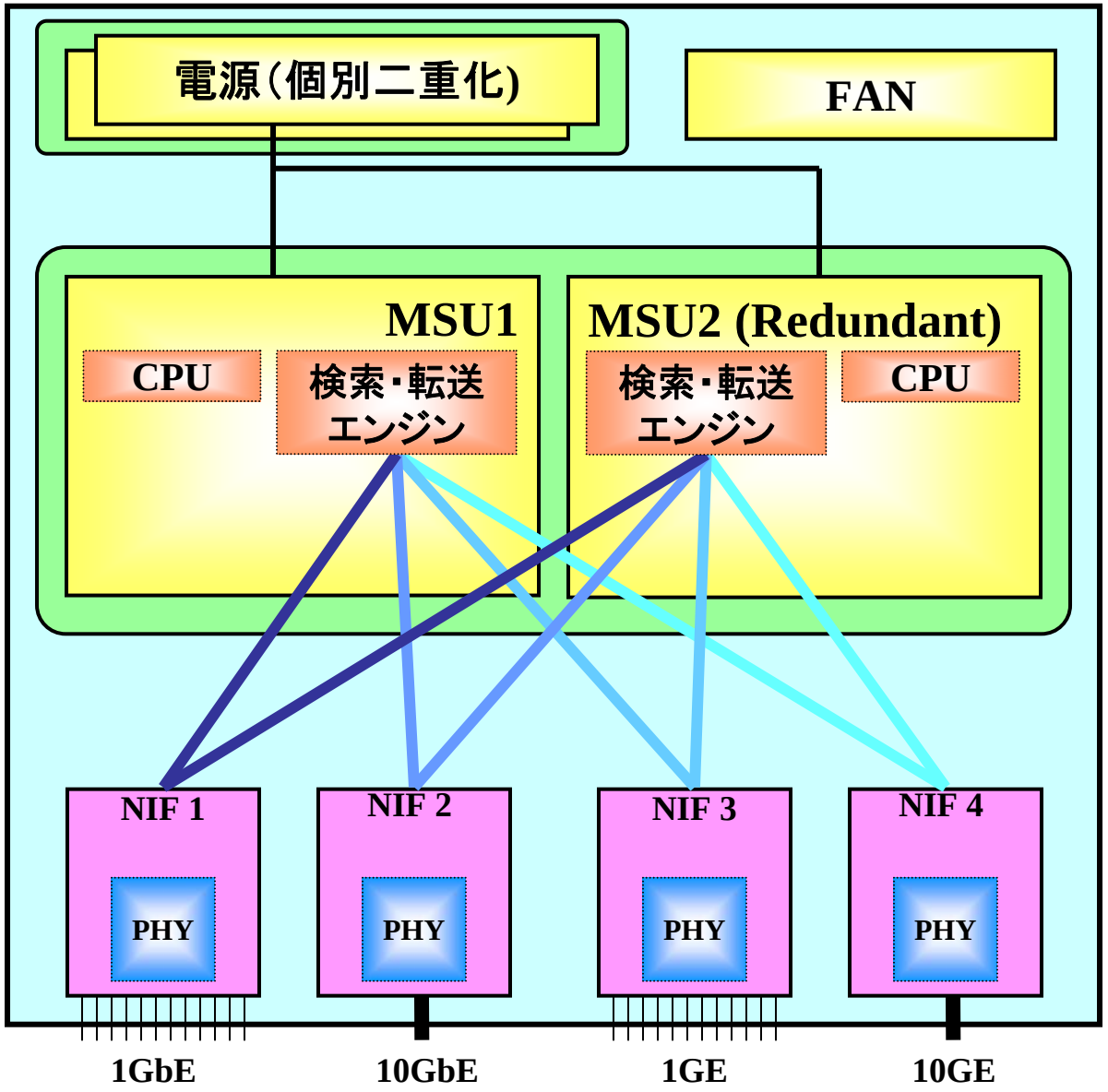


BCU:Basic Control Unit
PPU: Packet Processing Unit
CSW: Crossbar Switch

sNIF : single-size Network Interface
dNIF: double-size Network Interface
PS : Power Supply

集中型アーキテクチャ（AX6300の例）

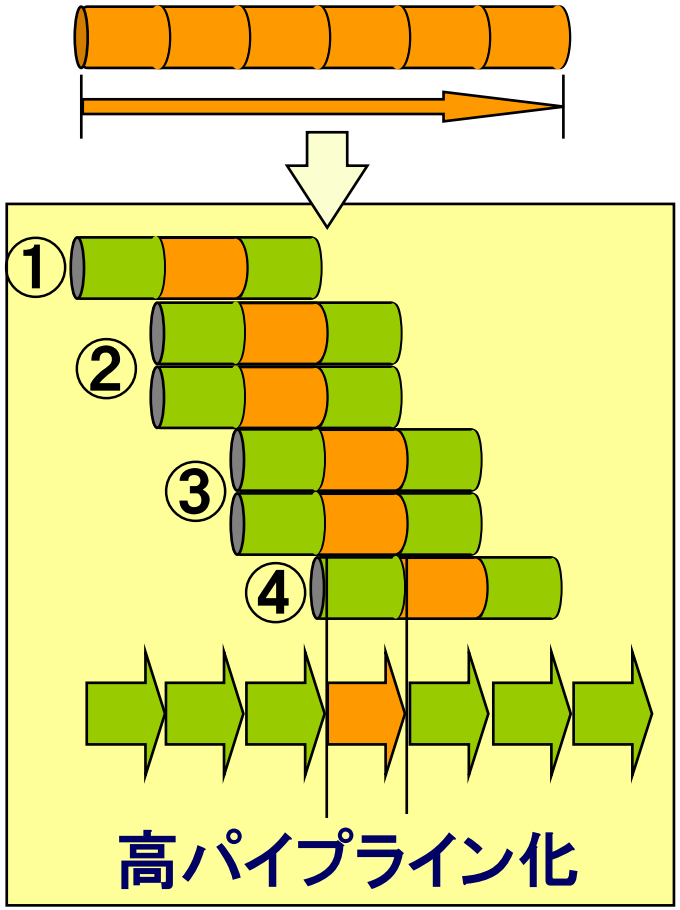
- CPUエンジンと転送エンジンが同一のモジュールで実装



高速化実現方式

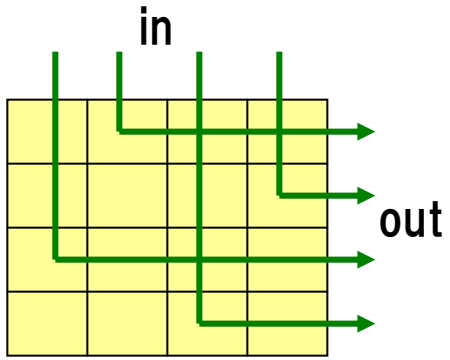
◆パケット転送処理のパラレル／パイプライン化

- □ 機能として4段のパイプライン
  - (1)受信処理
  - (2)入力検索(ルーティング・フィルタリング)
  - (3)出力検索(ルーティング・フィルタリング)
  - (4)送信処理
- □ それぞれの機能ブロックでさらに細分化されたパイプライン
  - ● 例: TCAM検索とRAM検索をパイプライン可
  - ● トータルで20段強



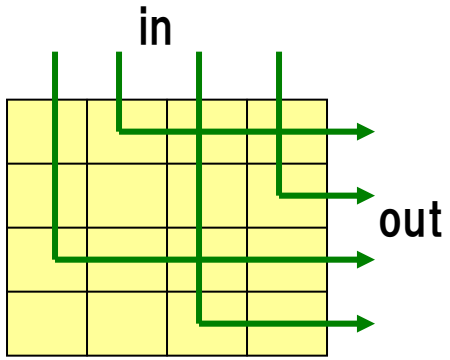
クロスバースイッチ

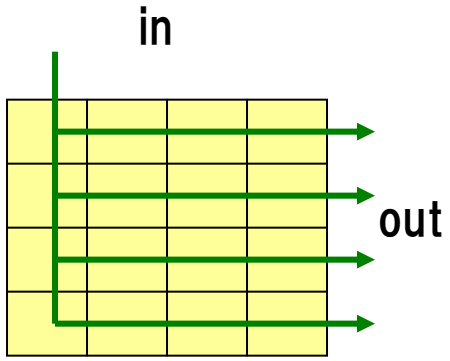
- マトリックススイッチ化されたバス
  - バックプレーンとも言う
  - 任意の転送エンジンから任意の転送エンジンへパケットを(内部的に)転送
  - Inとoutがぶつからない限り並行転送
- LSIレベルの誤り率と揺れの少ない遅延
  - 再送は考えない (考える必要がない)



クロスバースイッチ技術

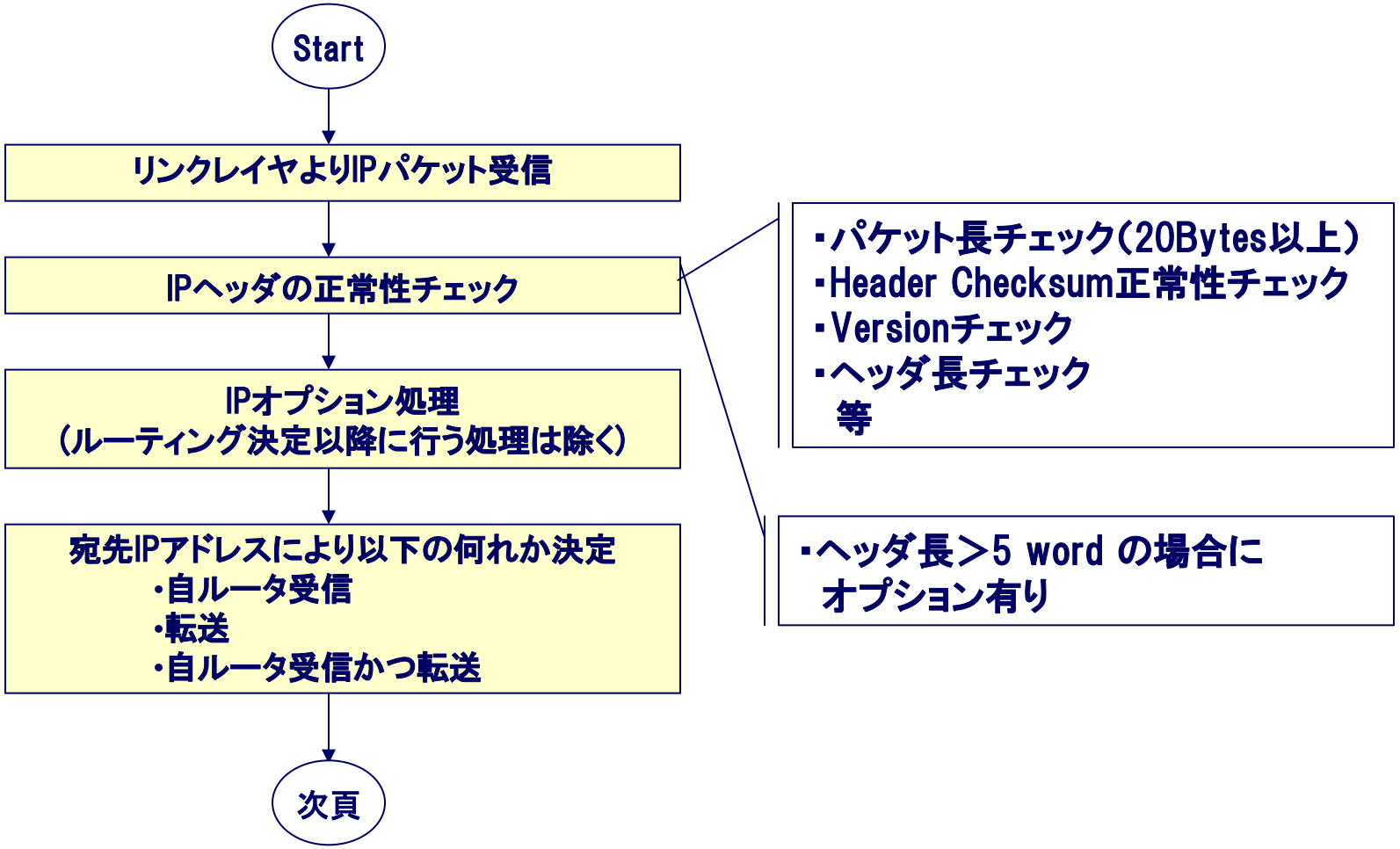
- パケットを32バイト単位に分割
  - 大きさを揃えて、クロスバースイッチの競合をなくす
  - 順序性は保存されてる
  - パケット落ちはない
- セル化することによってのデメリット
  - 内部ヘッダ分のオーバーヘッドがある
- 1対多通信も可能
  - マルチキャストで使用



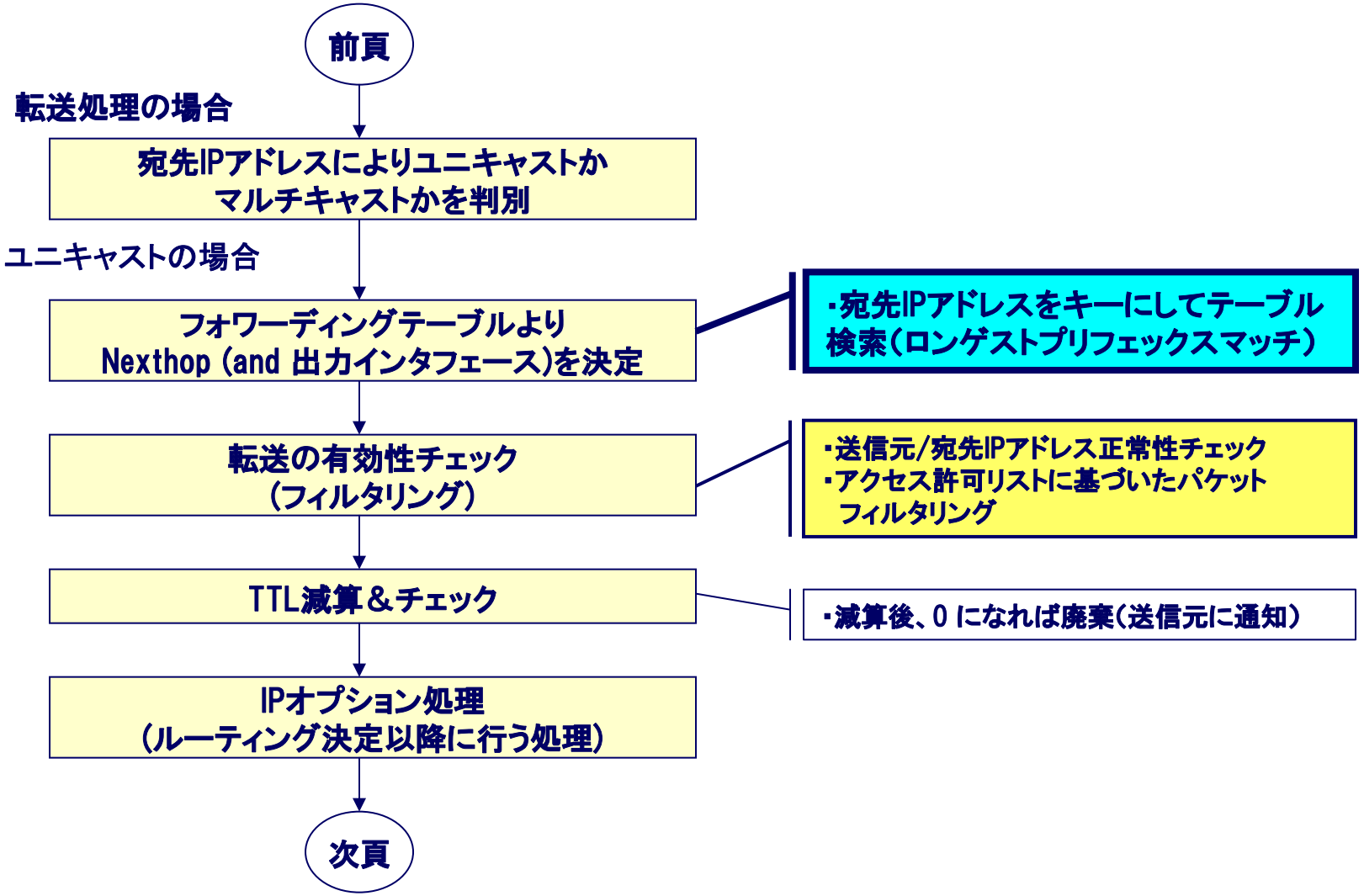


パケットフォワーディング処理の流れ（1/3）

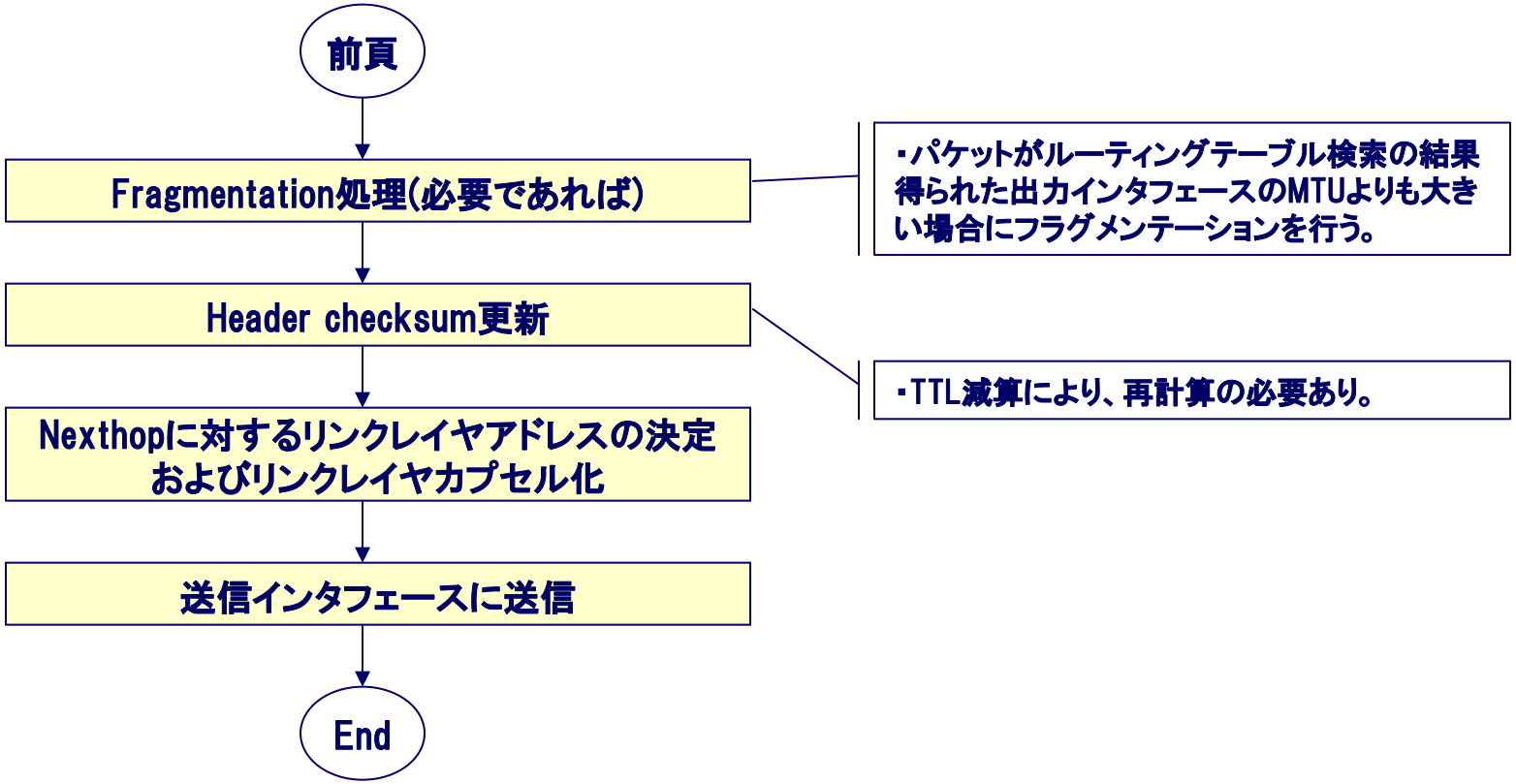
◆ 参考： RFC1812（ Requirements for IP Version 4 Routers ）



パケットフォワーディング処理の流れ（2/3）

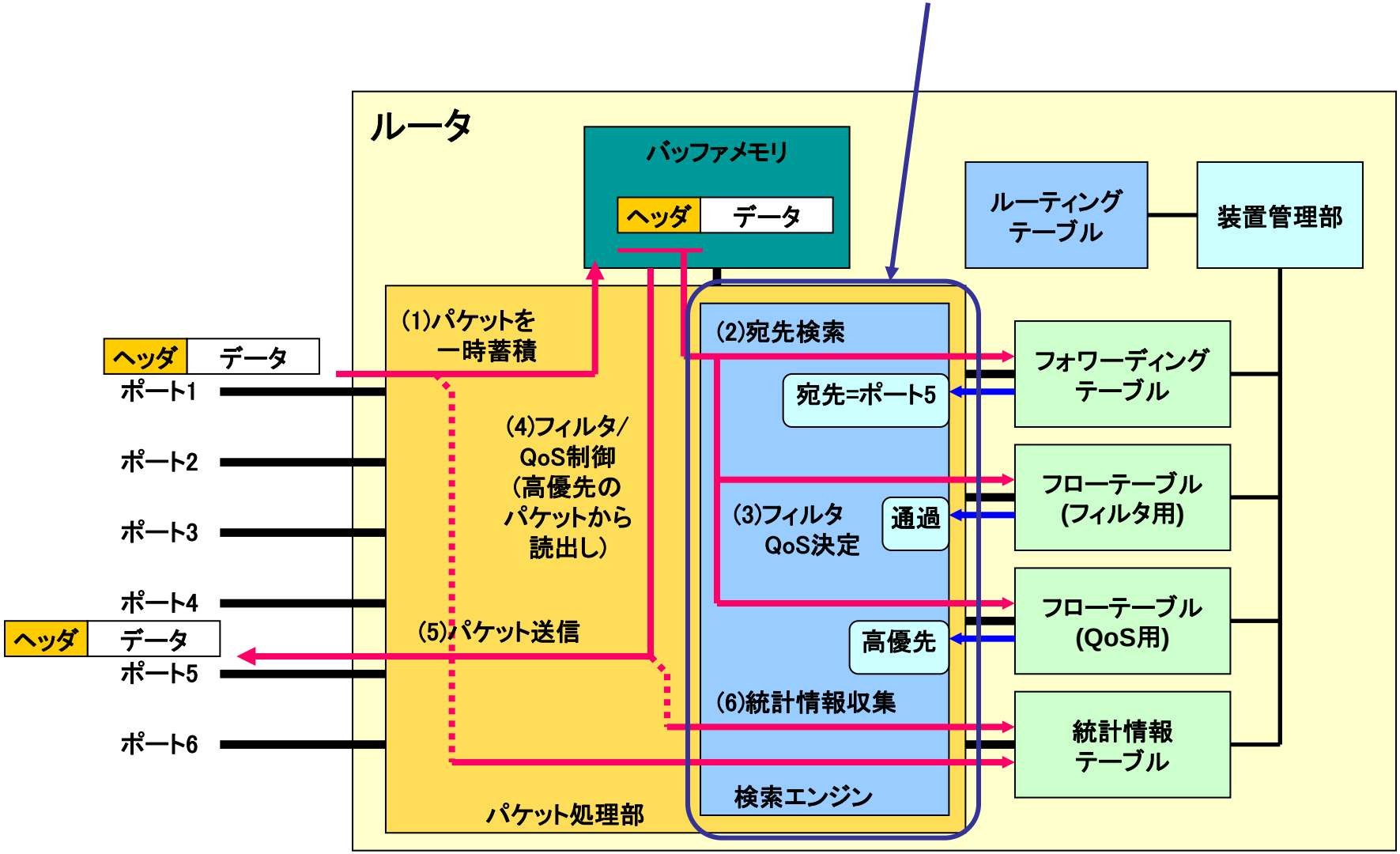


パケットフォワーディング処理の流れ（3/3）



高速アドレス検索エンジンの構成例

フォワーディング、フロー検索、統計採取を1チップで実現！

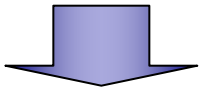


フォワーディングテーブル検索

◆フォワーディングは送信先ホストというより、送信先ネットワークに向けての転送である。
◆フォワーディングテーブルは、ルーティングテーブルから転送に必要な情報を抽出して構成される送信先ネットワーク毎の情報を持つテーブル

送信先ネットワーク	サブネットマスク	次ルータ(Next Hop)	出力IF	出力リンク(Etherの場合)
133.206.40.0	255.255.255.0	125.200.50.20	#0	00:00:80:30:37:d1
133.206.70.0	255.255.0.0	125.210.60.90	#8	00:00:80:42:4a:c3

◆ フォワーディングテーブルのインプリについて
- IPv4アドレスは 32bits → 2の32乗(≒4.3ギガ個)のアドレス空間
- 直接アドレッシングによる単純なテーブルで構成しようとすると、1エントリ当たり1Byteとしても、4.3GBytes 必要！→単純なメモリアクセスでの実現不可



圧縮した(エントリ数に制約を設けた)テーブルおよび
そのテーブルを高速に検索する手段が必要

Longest Prefix Matchとは

◆Longest Prefix Match

- CIDR(Classless Inter-Domain Routing)、VLSM(Variable Length Subnet Mask)により、受信IPパケットのネットワークプレフィクス(言いかえればマスク長)が一意的に決まらない。

- マッチするマスク長が一番長い(Longest Matchする)送信先ネットワークを検索する必要あり → 処理の複雑化

◆Longest Prefix Matchの具体例

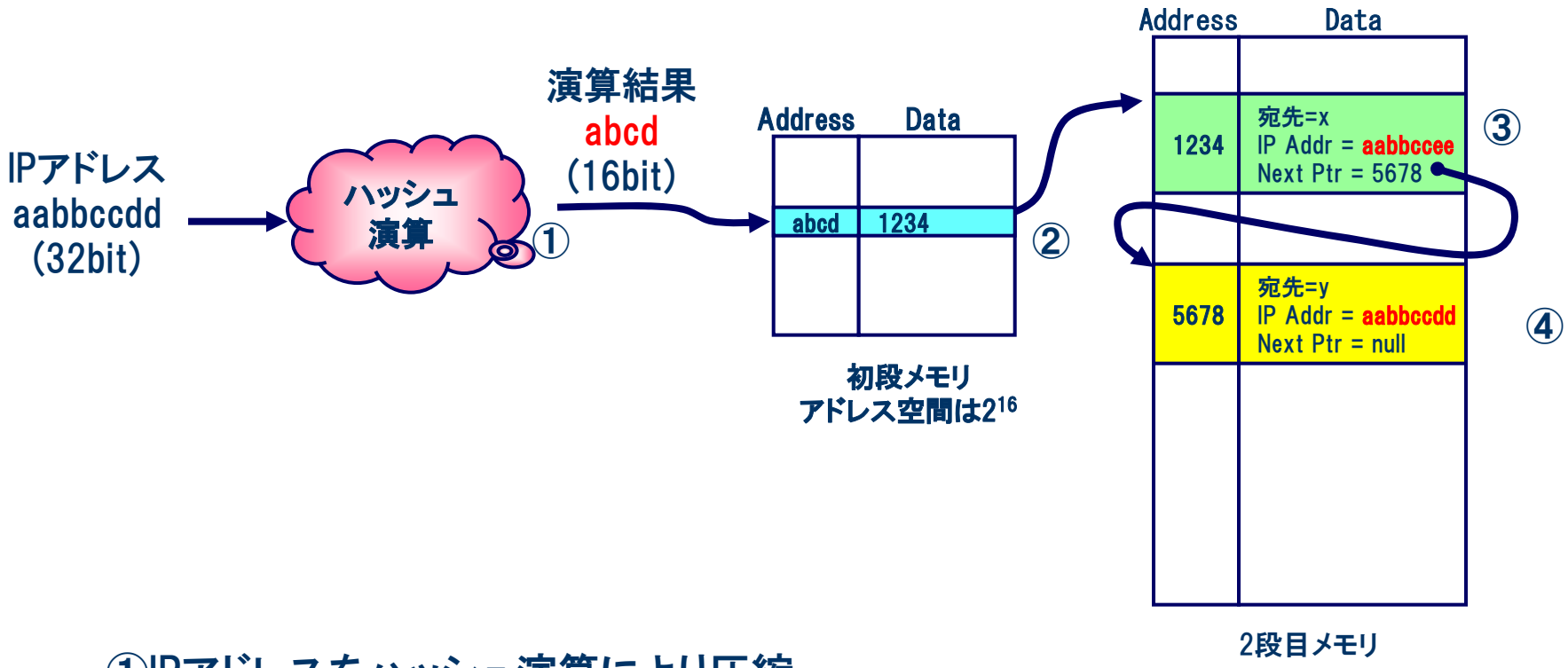
133.209.123.10行きのIPパケットに対しては下記の(B)を選択

- (A) 133.209.0.0　　(マスク:　255.255.0.0)
- (B) 133.209.123.0　(マスク:　255.255.255.0)
- (C) 133.209.125.0　(マスク:　255.255.255.0)

高速テーブル検索手法

- ハッシュ演算を用いたテーブル圧縮技術
  - IPアドレス(32bit)を特定の演算(わり算やCRC等)を用いて圧縮し(ex.16bit)、テーブル自体のサイズをコンパクト化
- 二分木検索アルゴリズム
  - IPアドレスを0,1のbit列に分解した木構造のテーブルによる検索
  - パトリシアンツリー等
- 多分木検索アルゴリズム
  - 二分木検索の多ビットへの拡張
  - Controlled Prefix Expansion
- CAM (連想記憶メモリ)の採用
  - データ入力に対し対応するアドレスを出力する特殊なメモリ
    (通常のメモリはアドレス入力するとデータを出力)
  - IPアドレス検索等をターゲットにしたLongest Prefix Match対応のCAMが出てきている

ハッシュによる検索テーブルの圧縮技術（1/2）



①IPアドレスをハッシュ演算により圧縮
②初段のメモリリードにより宛先情報テーブルのポインタを得る
③2段目のメモリリードにより宛先情報と該当する受信アドレス情報を得る
④受信アドレスに関するものでない場合には次の情報を読み出し
⑤以下③と④を繰り返し、受信アドレスによる検索結果を得る

ハッシュによる検索テーブルの圧縮技術（2/2）

- 検索キーをある特定の演算（わり算やCRC等）の結果に置き換えることにより、ビット幅を圧縮
  - Ex. IPv4 DA(32bit)をCRC16により16ビット化
- 実装が比較的容易かつ性能を出しやすい
  - 前頁の例では最短2回のメモリアクセスで宛先解決可能
- ハッシュ演算結果をもとに宛先情報を引き出す
  - このときたまたま演算結果が同じになるエントリが複数あった場合にはその中のどれか1つを選ぶ必要有り
    - 処理の複雑化，遅延の増大（最低保証値が見積もれない）
      →最善は100M sps(※)以上だが、最悪は数百k sps？
- LPM検索ができないため、経路集約を行うことができず結果的にエントリ数が膨大になる

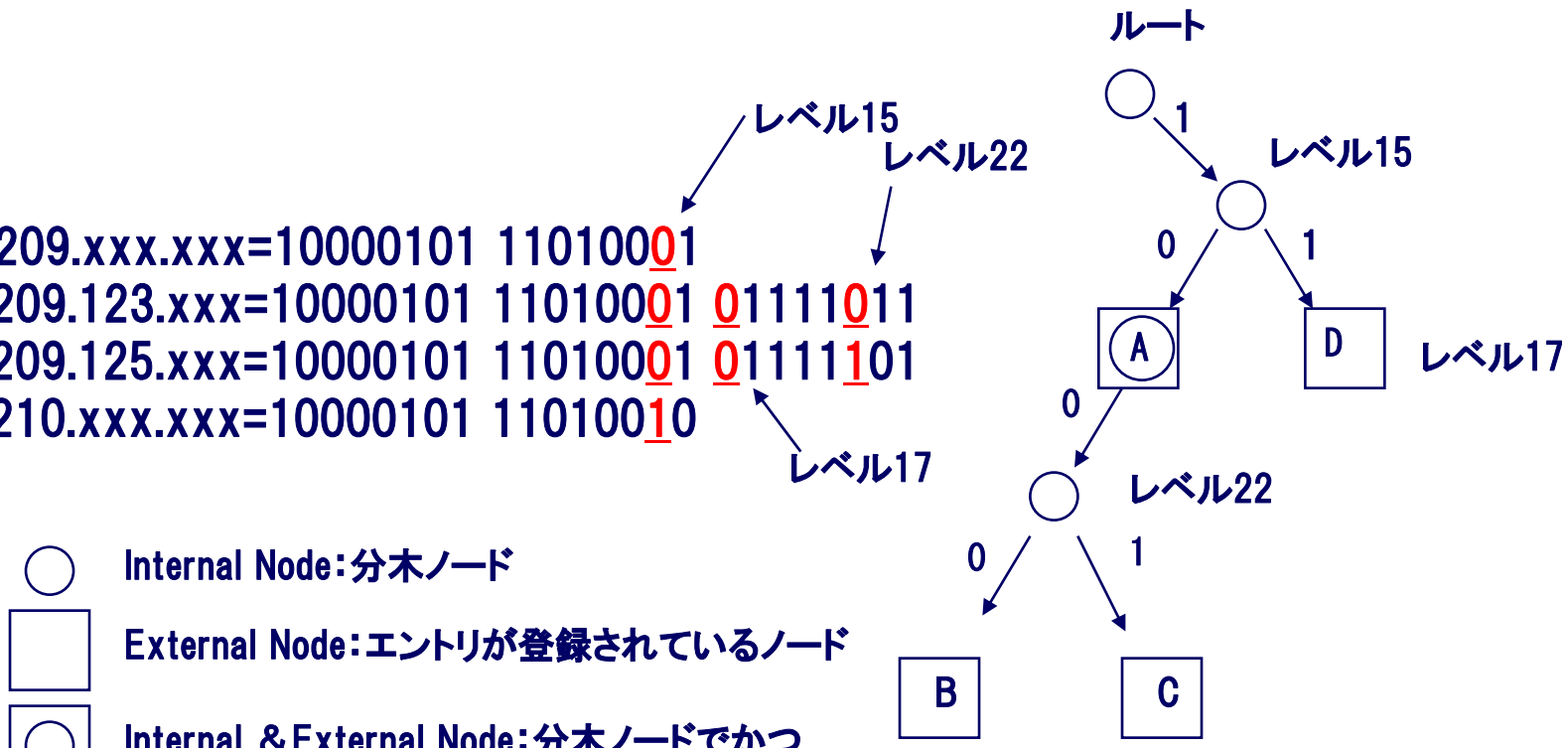
※sps (search per second) 1秒あたりの検索回数

パトリシアンツリー検索（1/2）

◆パトリシアンツリーとは？

- 二分木検索法の一種。
- 空の分木ノードを省略し、検索回数を削減

(A)133.209.xxx.xxx=10000101 11010001
(B)133.209.123.xxx=10000101 11010001 01111011
(C)133.209.125.xxx=10000101 11010001 01111101
(D)133.210.xxx.xxx=10000101 11010010



パトリシアンツリー検索（2/2）

- 従来からUNIXで使用。
- 検索回数は平均：$Log_2(N)$、最大：L。
  - Nは登録エントリ数、Lはエントリのビット数。
  - IPv4で64Kエントリ登録の場合、平均:16回、最大:32回
  - エントリの分布で検索回数にばらつきがある。
  - 最悪を考慮すると検索性能は数百k～数M sps

Controlled Prefix Expansion（1/3）

◆ Controlled Prefix Expansionとは？

- プレフィックス(マスク)長を検索し易いように拡張する。
  - ツリー検索では1ビットずつ検索をしていく必要があったが、**複数ビット単位で**検索することで検索回数を削減する
    - 例：検索のビット幅を2ビット単位に拡張する場合、以下のようにエントリを拡張し、拡張したエントリは全て同様に扱う。

例）0010 01**というエントリがあった場合
0010 0100
0100 0101
0100 0110
0100 0111
という4つのエントリに拡張してそれらを全て同一エントリとして扱う

◆ メリット

- 検索回数を削減
  - 固定的に4ビット単位に拡張すると、IPv4の場合、最大でも8回(=32/4)の検索回数で解決する。
  - 検索性能としては単一メモリの使用で数M～数十M sps（メモリを複数個パイプライン的に並列アクセスすることで100Msps以上の性能を出すことも可能）

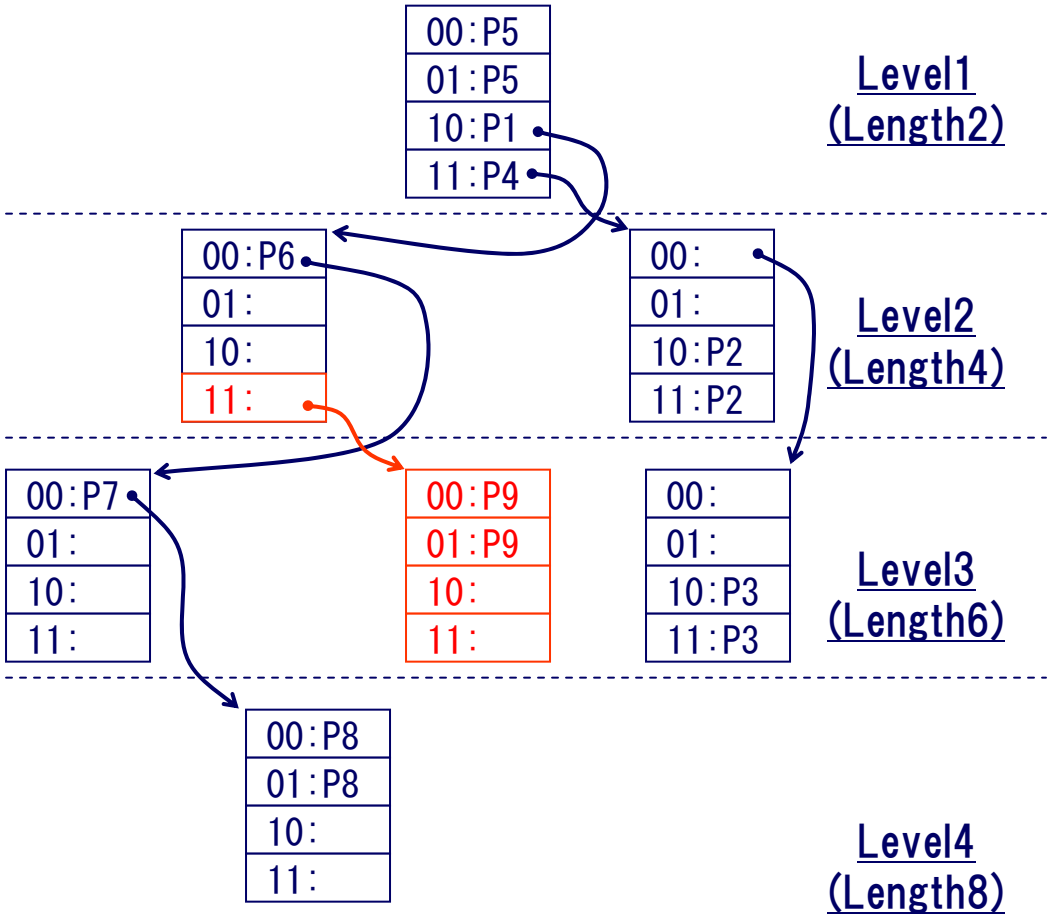
Controlled Prefix Expansion（2/3）

2ビット単位の拡張の例）
・エントリP9を追加する場合

エントリの拡張

Original	Expanded	
P5=0*	00 *(P5)	Length2
P1=10*	01 *(P5)	
P2=11 1*	10 *(P1)	
P3=11 00 1*	11 *(P4)	
P4=1*	11 10 *(P2)	Length4
P6=10 00 *	11 11 *(P2)	
P7=10 00 00 *	10 00 *(P6)	
P8=10 00 00 0*	11 00 10 *(P3)	Length6
P9=10 11 0*	11 00 11 *(P3)	
	10 00 00 *(P7)	
	10 11 00 *(P9)	
	10 11 01 *(P9)	
	10 00 00 00 *(P8)	Length8
	10 00 00 01 *(P8)	

実際のデータ構造



本来9個のエントリが14個に膨れ上がる

Controlled Prefix Expansion（3/3）

◆デメリット

- 拡張することにより、テーブル(メモリ)量の増大
  - 例えば、4ビット単位の拡張で00xx→0000 ,0001,0010,0011の場合、一つのエントリに対して4エントリの登録が必要。
- 容量が増えると同時にメンテナンスの負荷も増大
  - 登録/削除するエントリ数が増えると、転送処理に影響を与える場合がある
  - Prefix拡張したエントリを作成するためのCPU負荷がかかる

◆近年のメモリの大容量/高速化、Prefix拡張演算のオフロード化を行うことで容量的にもメンテナンス的にも実用できるレベルになりつつある

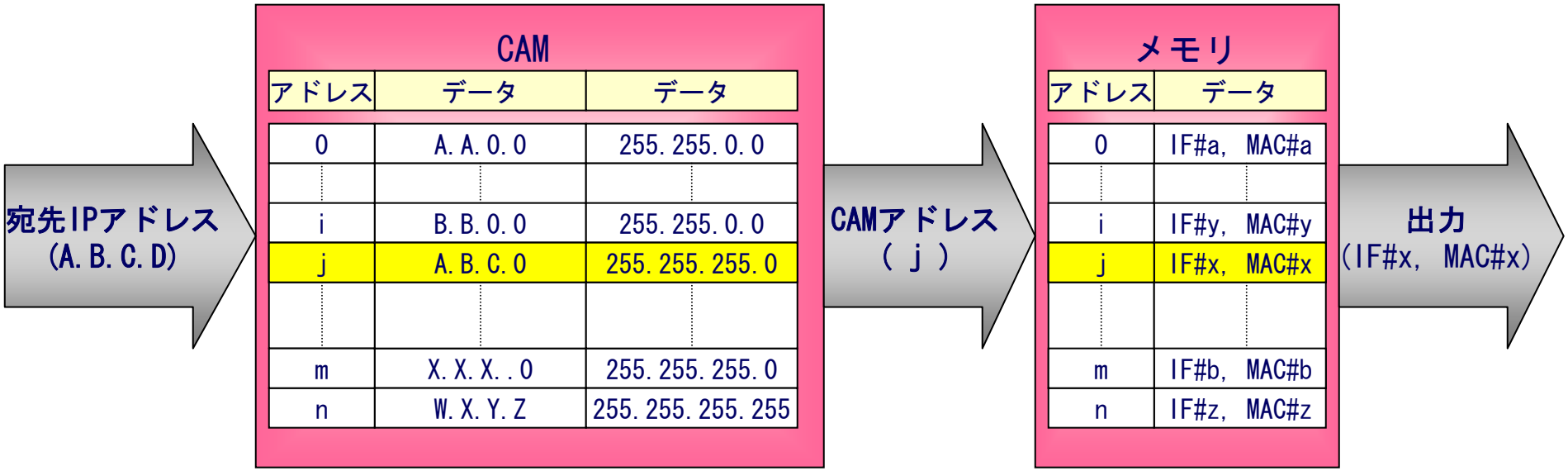
CAM（1/5）

- Contents Addressable Memory（連想記憶メモリ）
- データを入力するとアドレスを出力する特殊なメモリで、高速検索処理に特化
  （通常のメモリはアドレス入力するとデータを出力）
- Longest Prefix Match(LPM)が可能なCAM(Ternary CAM)も存在
- 検索回数は1回のみ。高速検索（数十M～数百M sps）
- 検索時間はビット幅（※）や登録エントリ数に依存しない
- エントリの管理に工夫が必要

※実際には制限あり

CAM（2/5）

- Ternary CAMによるアドレス検索動作の一例
  - CAMのデータとしてIPアドレス、およびネットマスクを登録
  - 検索時は宛先IPアドレスを検索データとしてCAMに入力
  - CAMは入力された検索データとマスクデータのandをとり、その後登録されているデータと一致するかどうか比較する
  - 一致するデータがある場合は登録されているアドレスを出力
  - CAMの出力アドレスでアクセスするメモリからテーブル内容を出力



◆Ternary CAMを用いた検索動作の課題

- □エントリの等力順序を工夫しないと複数ヒットする場合LPMにならなくなってしまう

検索キー
209.131.105.12

address	data
0	124.112.110.3
1	192.140.110.112
2	209.***.***.***
:	241.***.***.***
:	253.131.***.***
:	201.187.32.116
:	125.233.***.***
:	124.112.110.***
i	192.140.105.***
j	209.131.105.12
k	247.185.104.***
:	255.***.***.***
:	220.237.161.***
:	151.218.32.7
:	124.112.219.***
x	192.140.***.***
y	209.131.***.***
z	253.131.109.12

検索結果
Address = “2”

実際には“2”と“j”と“y”がヒットするが、通常のTernary CAMは複数ヒットした場合最も若いアドレスを検索結果として出力するため、LPMで考えれば“j”が検索結果となるべきところを“2”として出力してしまう

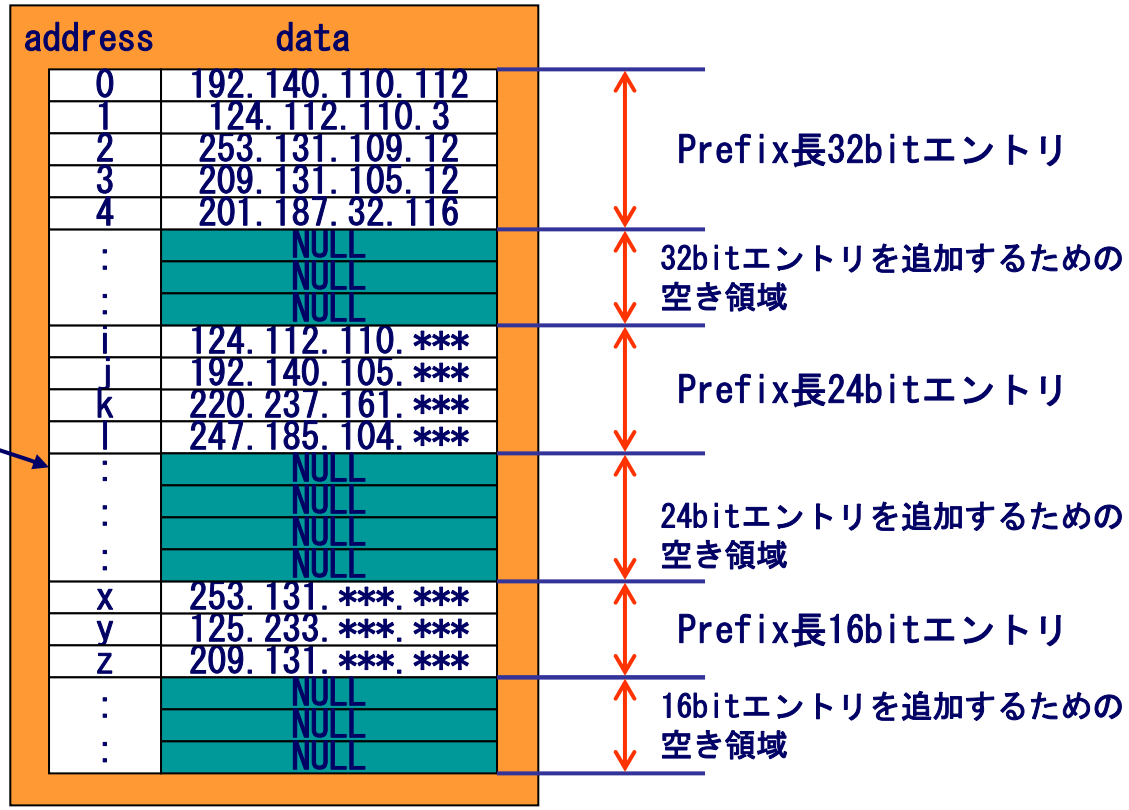
CAM（4/5）

◆ Ternary CAMでのエントリ管理方法の例

- □ Prefix長の長いものから並べる必要有り
- □ エントリを追加するための空き領域をあらかじめ用意しておく必要有り

例）新たに以下のエントリを追加
212.235.116.***

prefix長24bitの空き(NULL)領域の一番上(m番地)に登録



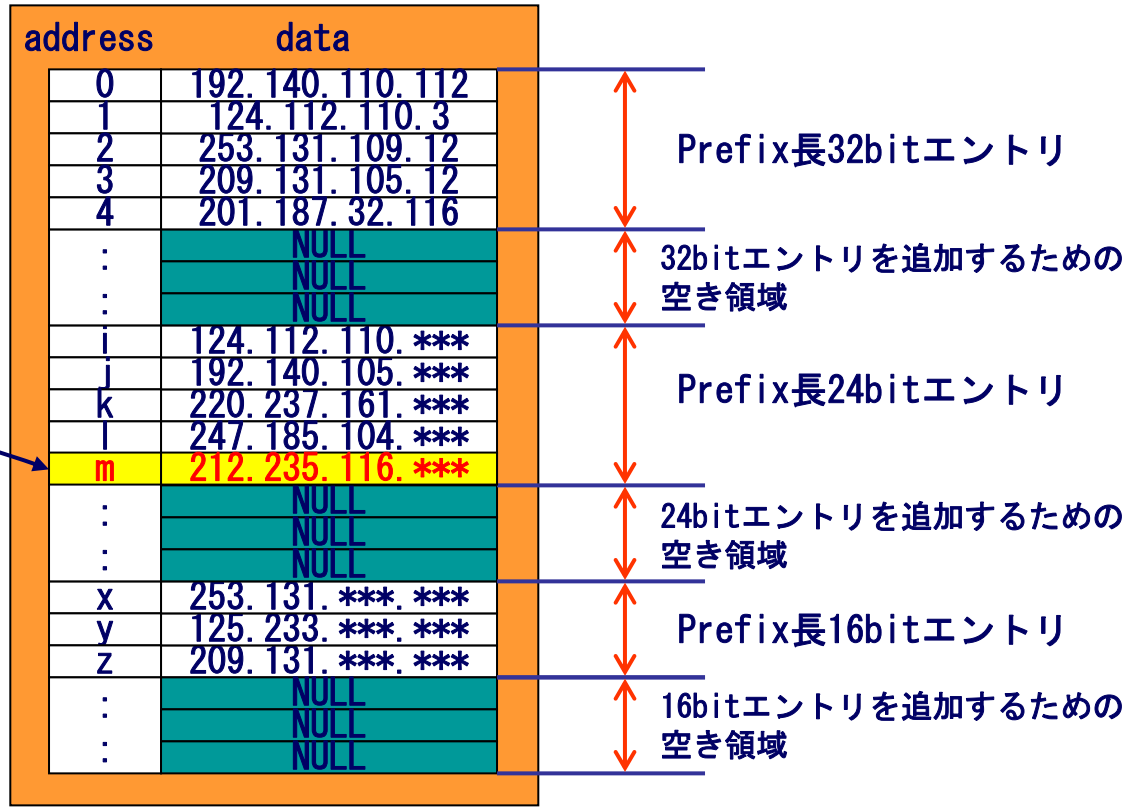
CAM（5/5）

◆ Ternary CAMでのエントリ管理方法の例

- □ Prefix長の長いものから並べる必要有り
- □ エントリを追加するための空き領域をあらかじめ用意しておく必要有り

例）新たに以下のエントリを追加
212.235.116.***

prefix長24bitの空き(NULL)領域の一番上(m番地)に登録



各種高速IPアドレス検索法の比較

◆ 各方式の特長をベースにした長所と短所で比較

	ハッシュ方式	二分木方式	他分木方式	CAM方式
機能拡張性	× LPM検索できず	△ 経路検索特化	△ 経路検索特化	○ 他テーブルマージ可能
性能レンジ	数百kpps～ 数百Mpps	数Mpps～ 数十Mpps	数十Mpps～ 数百Mpps	数十pps～ 数百Mpps
性能保証	×	△	△	○
メンテナンス	△	○ （Unixで実績有）	× （複雑）	△
部品点数	○	△ 性能保証のためにはメモリを多数使用	△ 性能保証のためにはメモリを多数使用	○
消費電力	○	○	○	△
価格	○ 特殊メモリ不要	○ 特殊メモリ不要	○ 特殊メモリ不要	×⇒○ 経路テーブル以外を収容可能
総合評価	×	△	△⇒○	○

各種高速IPアドレス検索法の比較(まとめ)

各手段とも、一長一短あり。

- ハードウェア(特にASIC)で処理する場合には固定ピッチで性能保証できるCAMが最も扱いやすく、現在はCAMを用いるのが主流であるが、大容量化に伴うコスト/消費電力の増加が課題となっている
- 上記課題に対応するために最近ではCAMメーカーも工夫をしている
  - 内部をサブブロック化してアクティブになるメモリセルを局所化(※1)
  - アルゴリズム検索のチップを開発(※2)
- 装置ベンダとしては技術トレンドを見極め一つの方式に拘りしすぎずに最適なインプリをしていく必要がある

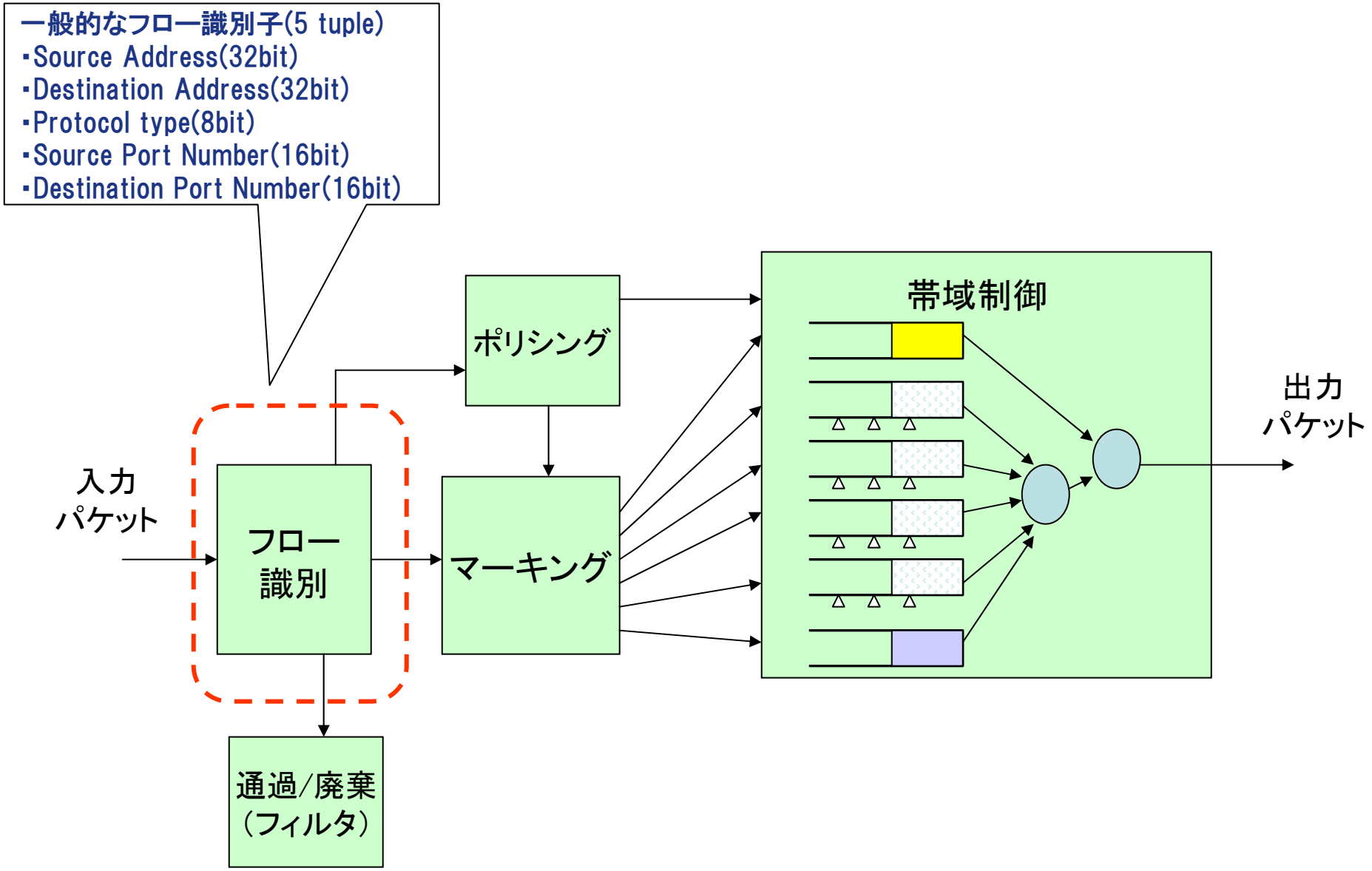
※1 http://www.anarg.jp/achievements/web2007/research_4rtg.html
※2 http://www.idt.com/products/getDoc.cfm?docID=18458762

フィルタリング/QoS技術

- フィルタリングとは
  - 送られてきたパケットを検査して通過させるかどうか判断する機能
- QoS(Quality Of Service)とは
  - 通信サービスの品質を指しており、トラフィック(アプリケーション)に応じ遅延、帯域、廃棄率などの品質を決めそれらを保証する
  - 従来のIP網は帯域を有効利用するため安価なベストエフォートが一般的であったが、VoIPや動画配信等遅延や廃棄を気にするアプリケーションが増えつつある

- **適切なフィルタリング/QoS処理を行うためにはフロー識別する機能が必須**
  - **フロー識別機能はパケットフォワーディングと同様検索が必要とされるため、高速化のための工夫が必要となる**

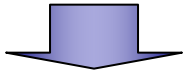
ハードウェアパケット転送時のフィルタ/QoS処理



フロー検出技術の高速化と課題

◆高速化

- □5 tuple(IPv4では96bit)＋αという多ビットの検索キーを高速に検索する技術が必要
  - ●性能を出すためには①ハッシュ、②CAMの適用が考えられるが、フロー検索ではexact matchではなく、任意フィールドを検索キーとすることが多く、ハッシュでは対応困難



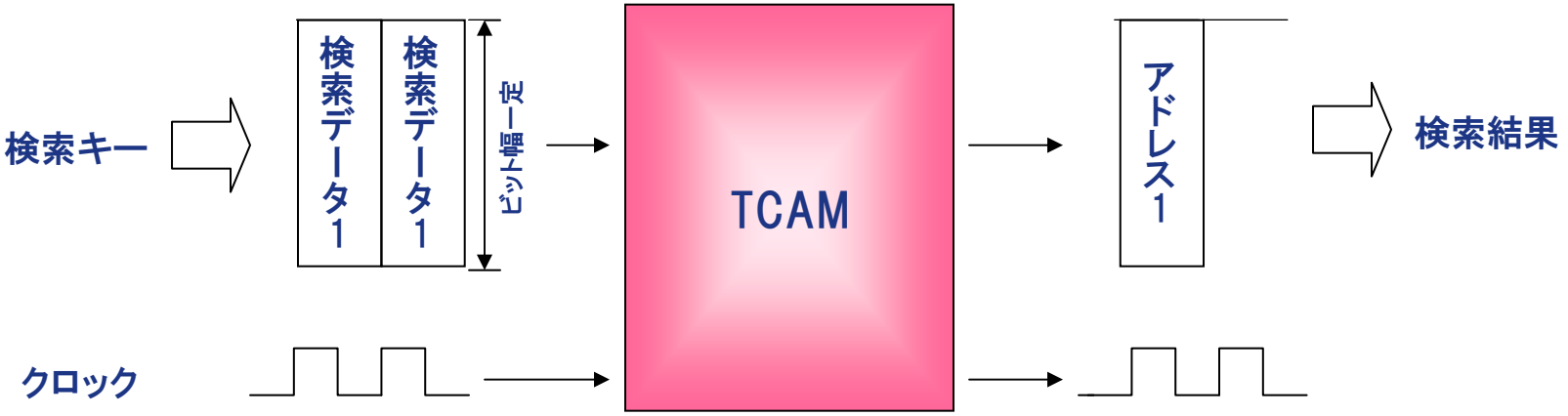
フロー検出にはCAMが最適

◆課題

- □検索キーの肥大化
  - ●従来の5 tupleに加え、Layer1、Layer2の情報やQoS情報までを検索キーに加わることで更に検索ビット数が増加

CAMを用いたフロー検出技術の課題

- CAMの検索性能は万全ではない
  - 特定のビット幅(1サイクルにCAMに与えることができる検索キーのビット数)までは検索性能を保持することが可能だが、それを超えると性能が1/2、1/4、、、と劣化する

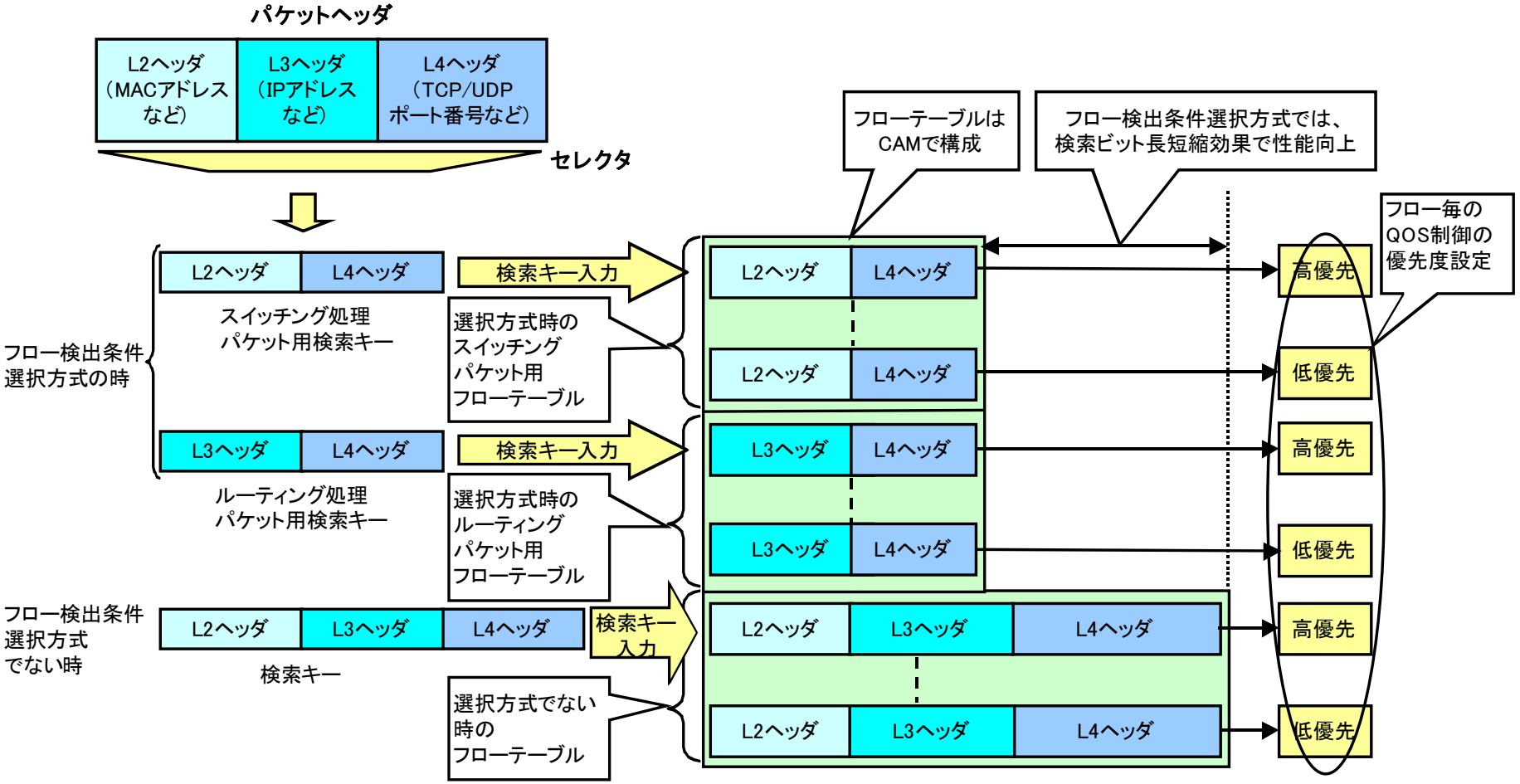


TCAMの検索では、1回のクロック動作(1サイクル)で検索できるビット幅が決まっており、それ以上ののビット幅では2サイクルかけて検索キーを渡す

検索性能が1/2に劣化

フロー検出高速化技術の例

◆ パケット毎にフロー検出に必要なフィールドを選択(圧縮)し、少ないビット幅で高速検索を実現



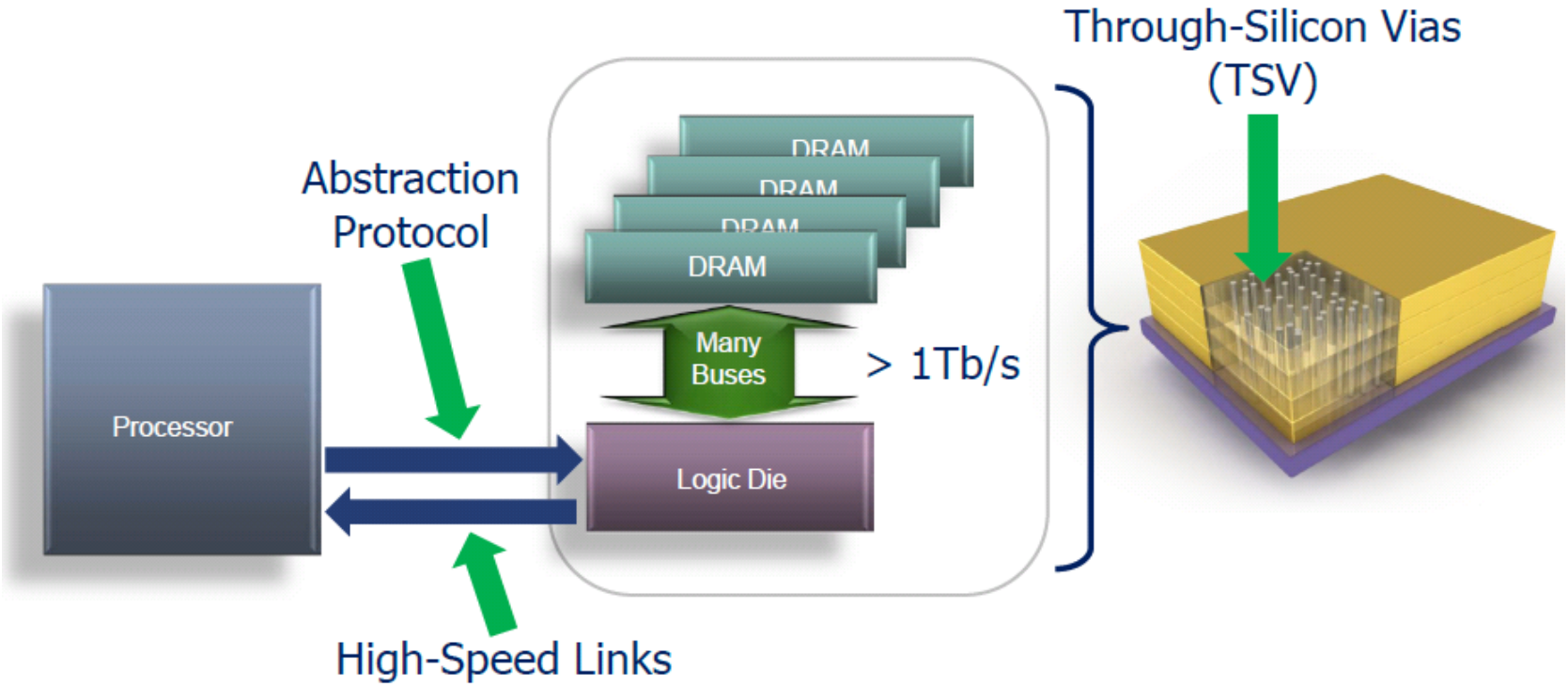
更なる性能向上に向けて

- パケット転送エンジンの性能を決めるのは
  - メモリの帯域幅：キューイング性能（bps）
  - メモリのランダムアクセス性能：テーブル検索性能（pps）
  - TCAMの容量：エントリー数
  - TCAMの検索性能：テーブル検索性能（sps / pps）
    - n個並列に接続すれば性能は上がるが、LSIのピン数増
  - メモリの容量：テーブル数 / バッファ容量
    - 容量を取れば、ランダムアクセス性能が悪化：DRAM
    - ランダムアクセス性能を取れば容量が減少：SRAM
  - コスト（≒部品代）
    - LSI：ダイサイズ＋ピン数（≒パッケージサイズ）に比例して上昇
    - DRAM：1T＋キャパシタ
    - SRAM：6T
    - TCAM：16T
- 性能の向上はコストに跳ね返るだけではない
  - bpsとppsとエントリ数はトレードオフの関係
  - 消費電力

次世代高速メモリ

- インターフェイスのSerDes化
- シリコン貫通電極(TSV)技術の導入

Hybrid Memory Cube (HMC)



http://www.hotchips.org/wp-content/uploads/hc_archives/hc23/HC23.18.3-memory-FPGA/HC23.18.320-HybridCube-Pawlowski-Micron.pdf

The Guaranteed Network

いちばん近くで、もっと先へ。