パケット処理キャッシュにおける応答予測キャッシュの最適化

代表研究者　　八　巻　隼　人　　電気通信大学大学院　情報理工学研究科　准教授

1 はじめに

近年，動画ストリーミングやクラウドサービス，IoT（Internet of Things）の普及が進み，インターネット通信量が増大している．報告[1]では，全世界の通信総量は 2017 年から 2022 年にかけて 1.5ZB から 4.8ZB に増加すると予測されており，今後もこの傾向が続くと予想される．このような中，パケットの中継を担うインターネットルータには，処理スループットの向上に加えて，省電力化が要求されている．ルータの消費電力は，処理パケット数に応じて増大する．そのため，今後も現在と同程度の電力バジェットでルータを動作させるためには，スループットを向上しつつ，より省電力にパケットを処理しなければならない．

ルータにおいて，パケット処理のスループットを決定付ける要因の一つはテーブル検索である．ルータはパケットの転送先や転送優先度，転送可否等を決定するために，専用のテーブル(e.g.，ルーティングテーブル)を検索する．テーブル検索に必要とされるメモリアクセスは，一般に，チェックサム計算等の他のパケット処理に比べ長い時間を要する．そのため，近年のハイエンドルータは高速な検索用途メモリの TCAM（Ternary Content Adressable Memory）を用いてテーブル検索を高速化している．しかし，TCAM は，検索ごとに検索対象のデータと TCAM 内の全データとの一致比較を行うため，SRAM 等の他のメモリと比べて消費電力が大きい[2]．また，TCAM を用いたルータは，最短パケット長のパケットが連続した場合には 50～100Gbps 程度のスループットが限界であり,今後実現される 1Tbps の超大容量回線に対応していくことが困難である．

この問題を解決する手法として，パケット処理キャッシュが提案されている．パケット処理キャッシュでは，RAM 等の高速かつ省電力なメモリにパケットの転送に必要なデータを格納し，テーブル検索の際は TCAM の代わりに上記のメモリをアクセスすることによってルータの省電力化や高速化を達成している．パケット処理キャッシュを用いたテーブル検索では，パケット処理キャッシュのミス率の改善がスループット，消費電力の改善に直結する．パケット処理キャッシュのミス率を改善する手法として，応答予測キャッシュが提案されている[3]．応答予測キャッシュは，インターネット通信にリクエスト・レスポンスモデルが多いことに着目し，到着した要求パケットから，対応する応答パケットのテーブル検索結果を計算する．そして，計算した結果から応答予測エントリを構成し，これをパケット処理キャッシュに投機的に登録することでパケット処理キャッシュのミスを削減する．

既存の応答予測キャッシュでは,応答予測エントリの追い出し時間が通信の往復遅延に対して短い場合に，応答予測エントリが応答パケットの到着前に追い出されてしまうことがある．特に小容量のパケット処理キャッシュはキャッシュエントリの追い出し時間が短く,応答予測キャッシュによるミス率改善が困難である．本研究では，応答予測キャッシュの投機的なキャッシュエントリの登録を適切に遅延させることによってパケット処理キャッシュのミス率改善を実現する．具体的には，パケット処理キャッシュにおける応答予測エントリの追い出し時間，往復遅延の値を用いて適切な遅延時間を算出し，その遅延時間を用いてエントリ登録を遅延させる機構を提案する．

2 パケット処理キャッシュ（PPC: Packet Processing Cache）

2-1 PPC の概要

前述したように，TCAM によるテーブル検索はルータのパケット処理におけるスループットのボトルネックとなっており，かつ電力消費が大きい手法である．この TCAM によるテーブル検索を改善する手法として，パケット処理キャッシュ(PPC: Packet Processing Cache)が提案されている．パケット処理キャッシュでは，ネットワーク通信の時間的局所性に着目し，TCAM によるテーブル検索結果をキャッシュすることでテーブル検索の高速化を実現する．具体的には，先述した 5 タプルによってパケットをフローに分類し，各フローに対するテーブル検索結果をキャッシュに格納する.テーブル検索結果は 5 タプルによって一意に定まるため，あるパケットのフローに対応する検索結果がキャッシュに保存されていれば，キャッシュを検索することで TCAM によるテーブル検索を代替できる．

公益財団法人電気通信普及財団
研究調査助成報告書　第37号　2022年度

図 1 にパケット処理キャッシュのアーキテクチャを示す．パケット処理キャッシュを用いたテーブル検索では，テーブル検索の際にまずパケット処理キャッシュにアクセスする．パケット処理キャッシュがヒットした場合には前述した 4 つのテーブルの検索結果が返され，これをもってテーブル検索を終了する．パケット処理キャッシュがミスした場合には，TCAM 内に複数存在するテーブルにアクセスし，それぞれ検索を行う．その後，得られた複数テーブルの結果をパケット処理キャッシュに登録する．このように，1 回のパケット処理キャッシュへのアクセスは 4 回の TCAM アクセスを代替する．パケット処理キャッシュには小規模の SRAM が用いられるため，TCAM のみでのテーブル検索よりも省電力かつ高速なテーブル検索を実現できる．パケット処理キャッシュがキャッシュミスした場合には TCAM にアクセスしてしまうので，パケット処理キャッシュのミスを削減することが肝要となる．

パケット処理キャッシュのエントリは，キャッシュタグとなる 5 タプルと，テーブル検索結果のデータから構成される．5 タプルは，4 バイトの宛先 IP アドレス，4 バイトの送信元 IP アドレス，2 バイトの送信元ポート番号，2 バイトの宛先ポート番号，1 バイトのプロトコル番号からなり，合計 13 バイトである．テーブル検索結果のデータは，出力先ポートとして 1 バイト，送信元，宛先 MAC アドレスとして 12 バイト，ACL によるフィルタリング結果として 1 バイト，QoS 情報として 1 バイトの合計 15 バイトである．この合計 28 バイトがキャッシュエントリのサイズとなる．

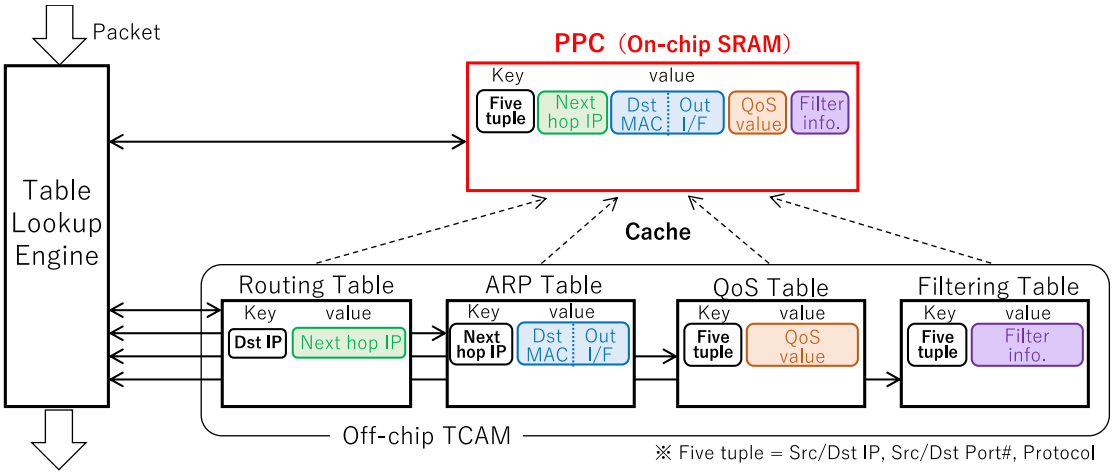

図1. パケット処理キャッシュのアーキテクチャ

2-2 PPC ミスとテーブル検索処理性能

パケット処理キャッシュでは，ミス率の改善がテーブル検索処理のスループットと省電力性の向上に直結する．パケット処理キャッシュのミス率改善において重要な要素である，キャッシュミスの分類について説明する．パケット処理キャッシュのミスは以下の三種類に大別できる．

1．容量性ミス

容量性ミスはパケット処理キャッシュに要求したデータがすでにキャッシュから追い出されていることによって起こるミスである．初期参照ミスと異なり，事前に書き込まれたデータが追い出されることによって起こるので，キャッシュ容量を増やすことによって改善できる．

2．競合性ミス

ダイレクトマッピング方式，セットアソシアティブ方式のキャッシュでは，同一キャッシュインデックスデータの競合が起こる．競合性ミスは，この競合によって必要とするデータがすでに追い出されていたことに起因するミスである．競合性ミスの改善には，キャッシュエントリ置換方法の改善や，同一キャッシュインデックスで保持できるエントリ数を示すウェイ数を増やすことが有効である．

3．初期参照ミス

初期参照ミスはパケット処理キャッシュに対する初回アクセスの際，キャッシュに対して要求したデータが存在しないことによって起こるミスである．このミスはキャッシュの容量やエントリの置換方法に関わらず発生するミスであり，削減のためにはキャッシュへの初回アクセスより前に該当データをキャッシュに書き込む必要がある．

2-3 関連研究

パケット処理キャッシュのミス率改善を目的とした関連研究について紹介する.

関連研究[4]では,パケット処理キャッシュを多階層化し,多階層化されたパケット処理キャッシュの各階層の容量を最適化している．一般にキャッシュの容量を大きくすれば，容量性ミスを削減できる．しかし，キャッシュ容量が増加するとキャッシュのアクセス遅延や消費電力が増大する．そのため，多階層キャッシュでは最初にアクセスする L1 キャッシュに高速な小容量メモリを，L2，L3 キャッシュに大容量キャッシュを用いることで,パケット処理キャッシュの容量性ミスの削減と高速化を実現した.結果的に3階層の PPC を用いて，1Tbps を超えるスループット(1 階層の PPC の 2.5 倍)を達成しつつ消費電力を 27.8%削減できることを示した.

関連研究[5]では,少量パケットから構成されるフローを迅速に追い出し,大量のパケットから構成されるフローを長時間保持するための手法として，ELC(Elevator Cache) を提案した．一般にパケット処理キャッシュでキャッシュの追い出しアルゴリズムとして，LRU(Least Recently Used)が用いられている．LRU アルゴリズムでは挿入の際には MRU(Most Recently Used)に挿入する．この場合挿入されたエントリは，追い出されるまでに(キャッシュのウェイ数-1)サイクルを要する.ELC では,新たに挿入するエントリを LRU(Least Recently Used) 位置に挿入し，参照されるごとに MRU 側のエントリと置換する．これによってヒット可能性が高い，パケット数の多いフローに対応するエントリを優先的にパケット処理キャッシュに残すことで，競合性ミスを削減した.

関連研究[3]で提案されている応答予測キャッシュは,初期参照ミスを削減する手法である.応答予測キャッシュは本研究において重要な研究であるため，次章で後述する.

2-3 応答予測キャッシュ

応答予測キャッシュ(RPC: Response Prediction Cache)[3]はパケット処理キャッシュにおける初期参照ミスの削減を主な目的とした手法である．初期参照ミスはパケット処理キャッシュにおいて，初回アクセス時に要求データがキャッシュに存在しないために起こるミスである．通常，この初期参照ミスは初回アクセスの際に必ず発生する．初期参照ミスを防ぐには今後要求されるであろうデータをあらかじめ予測し，投機的にキャッシュに登録する必要がある.

RPC では，インターネットにおけるサーバ・クライアント通信のうち高い割合を占める，リクエスト・レスポンスモデルに着目した応答予測を行う．リクエスト・レスポンスモデルの通信では，クライアントがサーバに要求パケットを送り，それに対してサーバが応答パケットを送る形で通信が実現される．サーバ・クライアント通信においてパケットの中継を行うルータでは，サーバ，クライアントを 1 往復する通信でリクエスト，応答パケットの二種のパケットを中継する．このようなリクエスト・レスポンスモデルに則った初回の通信においては,要求パケット,応答パケットともにパケット処理キャッシュに保存されていないため，計 2 回の初期参照ミスが発生する.

リクエスト･レスポンスモデルの通信では,応答パケットの 5 タプルは要求パケットの 5 タプルから予測できる．応答パケットの宛先 IP アドレス，送信元 IP アドレスはそれぞれ要求パケットの送信元 IP アドレス，宛先 IP アドレスに対応している．同様に，宛先ポート番号，送信元ポート番号も要求パケットの宛先ポート番号，送信元ポート番号を入れ替えた値に対応する．プロトコル番号は，要求パケット，応答パケットで同じ値が使用される．加えて，応答パケットのテーブル検索結果は要求パケットのテーブル検索結果から算出できる．応答パケットのルーティングテーブルの検索結果は，ルータが要求パケットを受け取ったインターフェースとなる．また，応答パケットの ARP テーブルの検索結果は，要求パケットの送信元 MAC アドレスに対応する．さらに，ACL，QoS テーブルの検索結果は，要求パケットの ACL，QoS テーブルの検索結果と同一である.

したがって,RPC では,要求パケットのデータから応答パケットのエントリ(応答予測エントリ)を作成し，事前にパケット処理キャッシュに登録しておくことで,TCAM のアクセス回数を増やすことなくパケット処理キャッシュのキャッシュミスを削減できる．実ネットワークにおけるトラフィックでは，およそ 70%が送信元と宛先の IP アドレスとポート番号を入れ替えた，対称なフローによって構成されている[6]．そのため，この手法によって多くの初期参照ミスを削減することが期待できる.

応答予測キャッシュのアーキテクチャを図 2 に示す．図 2 では，あるネットワークからラインカード 1 で受信したパケットをラインカード 2 から別のネットワークへ送信する場合の応答予測キャッシュの処理フロ

ーを示している．パケット処理キャッシュ，RPC モジュールは各ラインカードのパケット転送 ASIC に実装される．

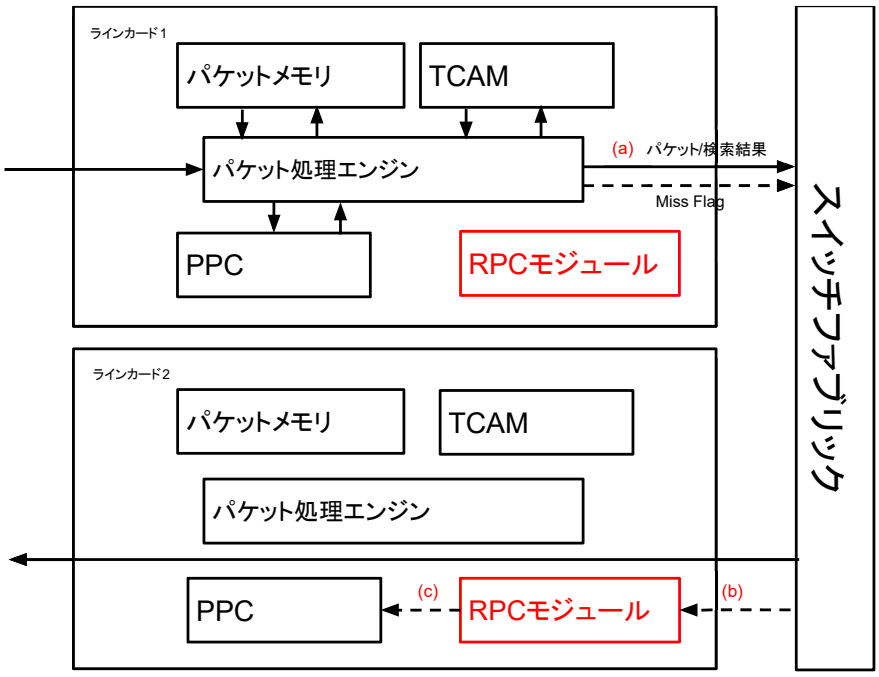

図２．応答予測キャッシュの全体アーキテクチャ

応答予測エントリの作成と登録は次の手順で行われる．

１．ラインカード 1 のパケット処理エンジンは到着パケットを受け取り，該当パケットが PPC へのアクセスでのヒット/ミスの結果を得る．PPC アクセスでのヒット/ミスにかかわらず，パケット処理エンジンはパケットと検索結果をラインカード 2 に転送する． PPC アクセスでミスした場合，テーブル検索結果と PPC Miss Flag を応答パケットが到着するラインカード 2 に転送する．

２．ラインカード 2 の RPC モジュールがラインカード 1 から検索結果と PPC Miss Flag を受け取る．受け取った PPC Miss Flag が有効であった場合，RPC モジュールは到着パケットに対応する応答パケットの 5 タプルとテーブル検索結果を計算する．計算した結果から応答予測エントリを構成し， RPC モジュールのキューに保存する．

３．最後にデータを RPC モジュールのキューから受け取り， PPC に登録する．

これによって，応答パケットのテーブル検索結果を保持する応答予測エントリが，応答パケットの到着するラインカードの パケット処理キャッシュに応答パケット到着前に登録される．

次に，PPC，パケット処理エンジンを含めた応答予測キャッシュのアーキテクチャの詳細を図 3 に示す．RPC モジュールは別のラインカードから送られてきたテーブル検索結果，Miss Flag を受け取り，応答予測エントリを作成する． この RPC モジュールは Aligner，Hash モジュール，キューで構成される．Aligner はキャッシュタグと，要求パケットに対応する応答パケットのデータを作成する． このデータの作成は先述した通り，応答パケットの情報を用いることでなされる．Hash モジュールは応答パケットのハッシュ値を計算する．キューは未登録の応答予測エントリを保持する．RPC モジュールのバッファに保持された応答予測エントリは，アービタを介してパケット処理キャッシュに登録される．

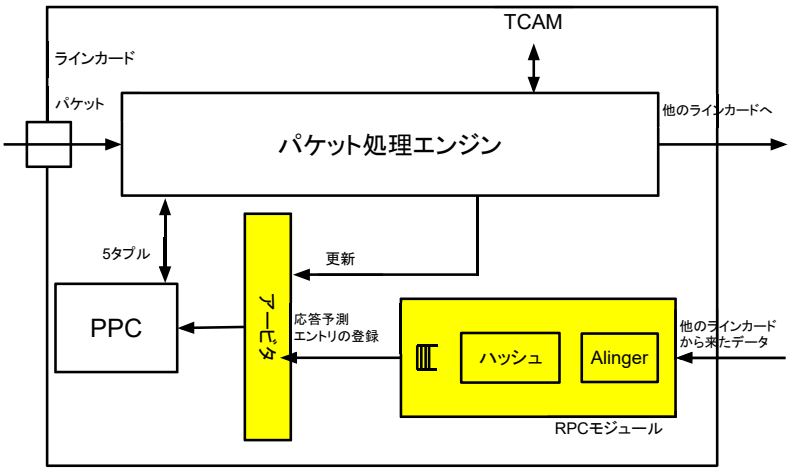

図３．RPC モジュールのハードウェアブロック図

3 遅延型応答予測キャッシュ（Delayed-RPC: D-RPC）の提案

3-1 RPC の課題

応答予測キャッシュでは，毎回のキャッシュミス時に該当パケットの応答パケットのエントリを作成し，パケット処理キャッシュに登録する．そのため，キャッシュミスの増加に伴って，RPC モジュールによって作られる応答予測エントリの数も増加する．小容量のパケット処理キャッシュを用いた場合，登録されるエントリに対するパケット処理キャッシュのエントリ数が少ない．そのうえ，容量性ミスの増加に伴い，応答予測エントリの数が増加する．そのため，応答予測エントリがヒットする前に追い出されてしまうことが懸念される．

そこで，ある実ネットワークトレースについて，各フローにおける最初の要求パケットが来てから応答パケットが到着するまでの往復遅延(RTT)について調査した．それに加えて，パケット処理キャッシュにおいて，応答予測エントリが登録されてから追い出されるまでの時間(応答予測エントリの追い出し時間)を測定した．これらによって，応答予測エントリがパケット処理キャッシュにある時間と，要求パケットがルータに到着してから応答パケットが到着するまでの時間について考察した．

あるトレースにおける往復遅延と応答予測エントリの追い出し時間を図示したのが図４である．図４では，あるネットワーク中における通信の往復遅延の累積分布を示すグラフと，追い出し時間の中央値を示している．図からも，従来の RPC では，多くの応答予測エントリが応答パケットの到着までにキャッシュから追い出されてしまうことがわかる．

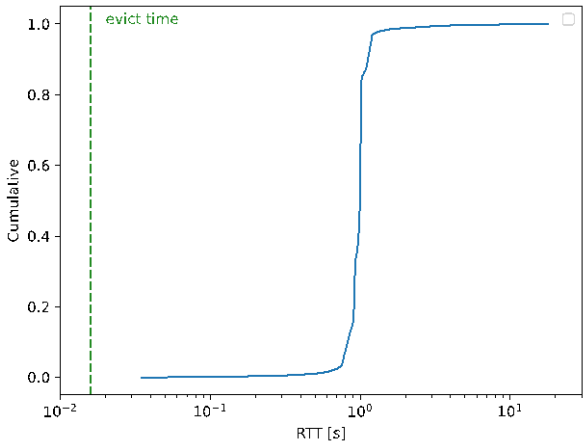

図４．あるネットワークにおける通信往復遅延の累積分布

3-2 D-RPC のコンセプト

D-RPC では，応答予測エントリの登録を一定時間遅延させることで，応答パケットの到着前に応答予測エントリが追い出されることを防ぐ．この遅延時間については，ネットワークの往復遅延を事前に計算し，それに基づいて決定する．提案手法の概要とその効果を，前節で分析したパケット処理キャッシュのエントリ追い出し時間に合わせて説明する．

図 5 では，図 4 と同じグラフを用いて提案手法のコンセプトを示した．提案手法では，要求パケットが到着してから一定時間応答予測エントリの登録を遅延させる．具体的には，図 4 のトレースについて，追い出し時間の中央値は 0.0159 秒である．そこで，このトレースについては 1.0 秒の遅延時間を設けるとする．その場合，要求パケットが到着してから 1.0 秒後に応答予測エントリが登録され，大多数の応答パケットが到着する，およそ 1.01 秒まで応答予測エントリがキャッシュに残る．これによって，従来手法と比べ，RPC によって作成された応答予測エントリでより多くの初期参照ミスを削減できる．

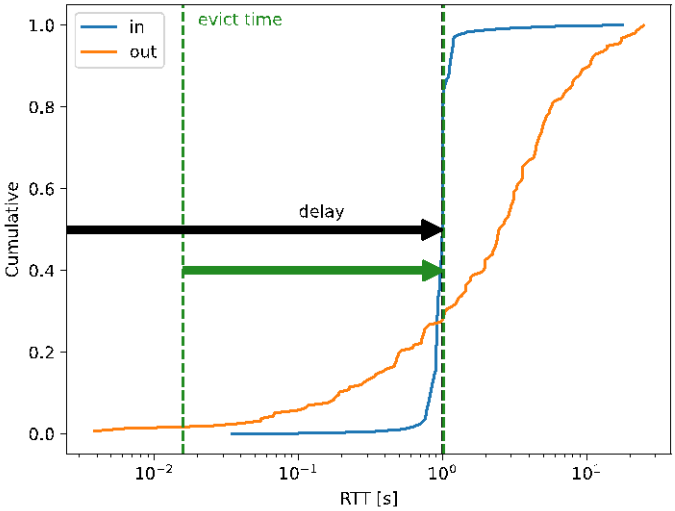

図５．D-RPC の適用イメージ

3-3 D-RPC の実装

D-RPC の実装を図 6 に示した．D-RPC は，ネットワークの往復遅延から RPC の予測応答エントリに対する最適な遅延時間を計算する遅延計算部（図中の左下部分）と，計算した遅延を RPC に適用する遅延適用部（図中の右下部分）に分かれる．以下ではそれぞれについて詳述する．

（１）遅延計算部

この部分では，RPC の予測応答エントリに対する最適な遅延時間の計算を行う．この計算には，3.1 節で説明した追い出し時間と往復遅延のデータを用いる．追い出し時間，往復遅延のデータは各ラインカードで一定時間収集され，ラインカードの DRAM にあるログメモリに保存される．なお，これらの計算は図中に示したラインカードの CPU を用い，計算した遅延時間は遅延適用部に転送する．

（２）遅延適用部

この部分では，遅延計算部で計算した遅延時間を用いて応答予測エントリの登録を遅延させる．遅延適用部では，新たにタイマ，D-RPC バッファを設ける．D-RPC バッファでは，遅延させる応答予測エントリを保存する FIFO である．D-RPC バッファのエントリは，PPC のエントリに RPC に登録される時刻のタイムスタンプを合わせたものである．タイマは現在の時刻を保持しており，D-RPC バッファにポーリングする．その際，D-RPC エントリのタイムスタンプ値が現在の時刻を上回っていれば，そのエントリを RPC モジュールのキューに転送する．D-RPC バッファに登録されるエントリは，RPC モジュールから受け取った応答予測エントリから構成される．加えて，タイマから得た現在時刻とレジスタに格納された遅延時間(Delay)を加えた値をタイムスタンプとして D-RPC バッファのエントリを構成し，D-RPC バッファに登録される．

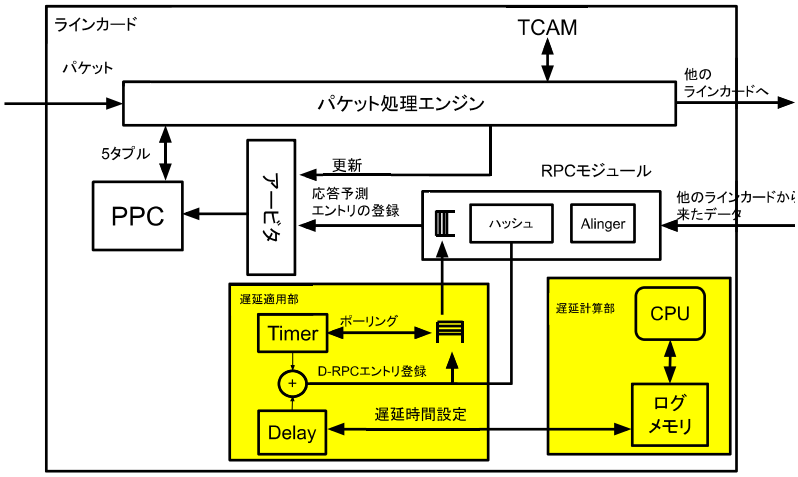

図５．D-RPC のハードウェアブロック図

4 評価

4-1 評価方法

提案手法を評価するために,提案手法のシミュレータと 10 種の実ネットワークのパケットトレースを用いて評価する[7,8]．今回用いたパケットトレースの情報を表 1 に示す.

表１．評価に使用したネットワークトレースの詳細

トレース名	地点	パケット数	時間[s]
BWY	Columbia University (BroadWaY)	1,616,336	90
Core	An academic network (not public)	51,940,543	90
FRG	Front Range GigaPOP	2,600,472	90
MEM	University of Memphis	92,926	90
MRA	Merit Abilene	3,716,471	90
ODU	Old Dominion University	628,146	90
PSC	Pittsburgh Supercomputing Center	1,435,482	90
TXG	Texas GigaPOP	1,109,851	90
UFL	University of Florida	4,388,338	90
WIDE	samplepoint-F	52,840,127	900
UPCB	Unknown	1,774,339	300

評価に用いたシミュレータは，論文[4]で使用されたものを提案手法に沿う形に改変したものである．具体的には，シミュレータを応答予測キャッシュのシミュレーションを実装した上で，提案手法の機構を実装した．このシミュレータは，パケット処理キャッシュの構成とネットワークトレースを与えることで，パケット処理キャッシュのヒット率，ミス率を算出できる．本評価ではセットアソシアティブ方式の一階層キャッシュを用いた．シミュレータでは，パラメータとしてパケット処理キャッシュのウェイ数，エントリ数，新規エントリの挿入位置(LRU/MRU)を選択できる．今回の評価では，ウェイ数を 4 とし，LRU 位置にエントリ挿入するようにした．また，エントリ数は 256，512，1024 について評価した.

提案手法の有効性の評価については，様々なネットワークトレースにおいて提案手法を適用した場合のキャッシュミス率を計測し，キャッシュミス改善率，提案手法によるスループット向上と消費電力削減について見積もる．評価では，表 1 のトレースをタイムスタンプで二つに分割し，一方を遅延時間の算出に用い，もう一方のトレースと算出した遅延時間を用いて手法を評価する.

スループットの評価には文献[4]で用いられている許容スループットを用いる．許容スループット T_PPC[Gbps]は,パケットロスを発生させることなくテーブル検索処理を行うことができるルータの最大入力トラフィック容量である.具体的には次式で表される.

$$T_{PPC} = \min\left(\frac{l}{d_{SRAM}}, \frac{l}{d_{TCAM} \cdot m_{SRAM}}\right) \tag{1}$$

ここで，*l* はパケットサイズ，*d_SRAM*, *d_TCAM* はそれぞれ，SRAM(パケット処理キャッシュ)および TCAM のサイクル時間を示している．また，*m_SRAM* は PPC のミス率である．今回の評価では TCAM のアクセス遅延を 10.07ns とした.最悪ケースを想定し，全てのパケットが最短パケット長の 64byte とすると，TCAM のパケット処理性能は 47.85Gbps となる.

次に，消費電力 *P_PPC* [W/Gbps]は次の式で表される.

$$P_{PPC} = \frac{E_{SRAM} \cdot n + (a \cdot E_{TCAM} \cdot m_{SRAM}) \cdot n + S_{SRAM} + S_{TCAM}}{T_{PPC}} \tag{2}$$

パラメータ E は動的消費電力を表し，下付き添え字がメモリの種類を表している．また，パラメータ P も同様に静的消費電力を表している．n は 1 秒あたりの平均パケット数，a は 1 パケットのテーブル検索に要する TCAM アクセス数である．キャッシュメモリの静的消費電力，動的消費電力の見積には，CACTI 6.5[?]を使用した．スループットの増大に比例してパケット数が増加すると仮定し，1 秒あたりの平均パケット数をスループットから算出した．本評価では，文献[4]を参考に，TCAM のアクセスエネルギーを 10.5nJ，リーク電力を 9.404mW とした．なお，1 回のテーブル検索において 4 回の TCAM アクセスが生じることから，1 回のテーブル検索における TCAM のアクセスエネルギーは 42nJ である.

4-2 評価結果

スループット，消費電力の算出結果を表 2,3,4 に示す．また，PPC エントリ数が 256 の際の PPC ミス率とその内訳，ミスの改善率を図 6,7,8 に示す．図中の initialmiss は初期参照ミスを，othermiss は競合性，容量性ミスを示す．評価結果より，一部のトレースで高い改善率を示していることがわかる．特に，エントリ数 512 のパケット処理キャッシュを用いた場合の，MEM の in 側のトレースについては，50%以上のミス改善率が得られており，高いスループット向上率および消費電力削減率を示した．全体を通して，提案手法はスループット向上および消費電力削減に寄与している．これは主に初期参照ミスを削減できている効果が大きい.

5 おわりに

インターネット通信量は年々増大しており，インターネットルータへのパケット転送の向上が重要視されている．テーブル検索処理はインターネットルータのパケット転送処理におけるスループット上のボトルネックかつ電力消費の主要因となっている.

パケット処理キャッシュでは，テーブル検索結果を小規模な RAM に保存し，TCAM での検索を代替することでルータの省電力化と高スループット化を実現する．このパケット処理キャッシュでは，ミス率の改善がスループットや消費電力に直結するため，ミス率の改善を目的とした研究がこれまでに多くされてきた.

応答予測キャッシュは，インターネット通信の多くがリクエスト・レスポンスモデルの通信であることに着目した．要求パケットから応答パケットに対応する応答予測エントリを作成し，これをパケット処理キャッシュに登録することで初期参照ミスの削減を目指した．しかし，この応答予測キャッシュでは小容量キャッシュを用いた場合に，多くの応答予測エントリが応答パケットの到着前に追い出されてしまう.

そこで，本稿では，応答予測エントリの登録を遅延させることで，応答予測エントリによるミス率改善を試みた．結果として，一部のトレースでパケット処理キャッシュのミスを 58%改善した．また，消費電力を 57%削減し，スループットを 2.35 倍に改善した.

表２．スループットおよび消費電力の評価結果（PPC エントリ数 256）

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	21.4	35.1	6.42	15.8	38.1	38.1	15.1	31.7	37.6	52.2	33.1
	T_PPC	196	120	658	267	110	110	263	133	112	80.4	127
提案	P_PPC	16.9	35.1	6.42	6.75	38.0	37.1	16.1	27.8	35.6	51.6	33.1
	T_PPC	250	120	658	626	110	113	262	151	118	81.4	127

(a) Inbound

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	25.7	23.4	14.8	14.8	34.1	25.0	26.5	22.6	21.3	37.1	13.1
	T_PPC	163	180	284	284	123	168	159	187	197	113	321
提案	P_PPC	26.1	22.2	11.2	14.8	32.3	24.9	24.6	22.5	21.0	35.4	13.1
	T_PPC	160	189	377	284	130	169	171	187	200	119	321

(b) Outbound

表３．スループットおよび消費電力の評価結果（PPC エントリ数 512）

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	16.5	28.1	3.97	7.41	34.8	24.7	12.7	23.3	31.6	48.1	28.2
	T_PPC	255	150	856	570	121	171	332	180	133	87.3	149
提案	P_PPC	12.3	28.1	3.97	3.65	34.8	24.1	12.8	21.5	28.4	46.8	28.2
	T_PPC	342	150	856	856	121	175	329	195	148	89.8	149

(c) Inbound

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	19.4	19.5	7.59	10.9	29.8	18.3	18.5	18.6	16.6	27.1	10.8
	T_PPC	217	216	556	386	141	230	228	226	254	155	392
提案	P_PPC	19.8	18.4	6.34	11.0	27.9	18.3	17.6	18.6	16.5	26.7	10.8
	T_PPC	213	228	667	384	151	230	239	226	255	157	392

(d) Outbound

表４．スループットおよび消費電力の評価結果（PPC エントリ数 1,024）

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	11.6	22.9	2.18	2.98	30.8	14.1	8.91	13.8	24.0	43.1	21.7
	T_PPC	364	184	856	856	137	298	474	305	175	97.6	194
提案	P_PPC	8.88	22.9	2.18	2.47	30.8	14.0	8.91	13.7	20.9	40.5	21.7
	T_PPC	475	184	856	856	137	300	474	308	201	104	194

(e) Inbound

		BWY	Core	FRG	MEM	MRA	ODU	PSC	TXG	UFL	WIDE	UPCB
従来	P_PPC	13.2	16.3	3.12	7.71	24.1	11.2	11.2	15.1	13.1	21.8	8.47
	T_PPC	319	258	856	548	175	378	377	279	320	193	498
提案	P_PPC	13.2	15.4	2.98	7.71	23.1	11.1	10.9	15.1	13.1	21.8	8.47
	T_PPC	319	273	856	548	182	379	386	279	321	193	498

(f) Outbound

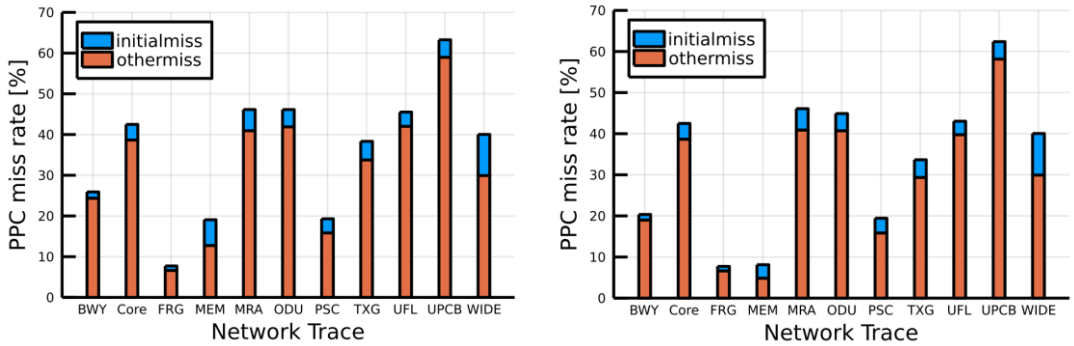

図６．PPC ミス率とその内訳（PPC エントリ数 256，Inbound，左：従来，右：提案）

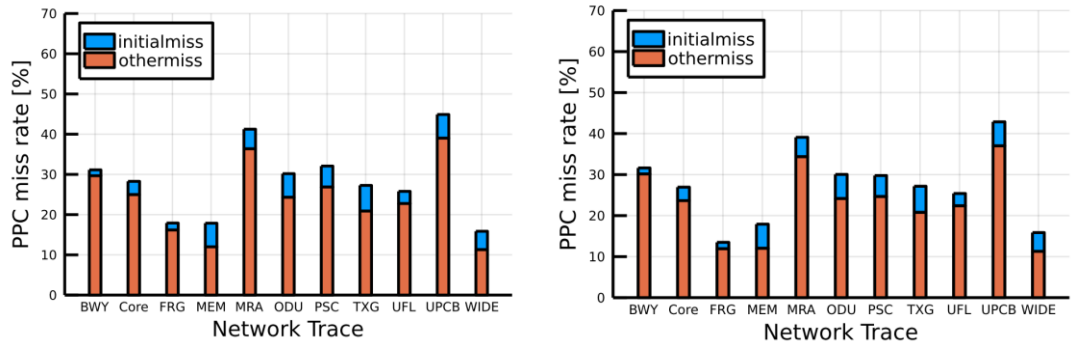

図７．PPC ミス率とその内訳（PPC エントリ数 256，Outbound，左：従来，右：提案）

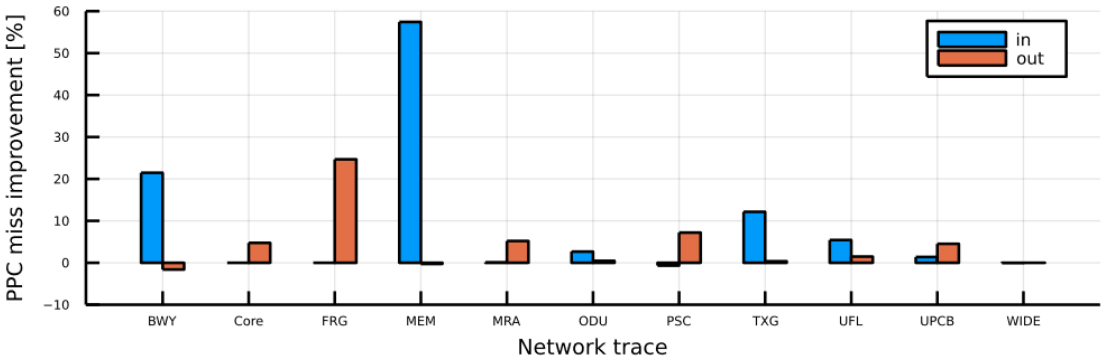

図８．従来手法に対する D-RPC の PPC ミス改善率（PPC エントリ数 256）

【参考文献】

[1] Cisco, Cisco visual networking index (vni)：予測とトレンド、2017 ～ 2022 年 ホワイトペーパー – cisco. https://www.cisco.com/c/ja_jp/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-cii-741490.html, 2019. (Accessed on 11/23/2019).

[2] Masami Nawa, Kenzo Okuda, Shingo Ata, Yasuto Kuroda, Yuji Yano, Hisashi Iwamoto, Kazunari Inoue, and Ikuo Oka, “Energy-efficient high-speed search engine using a multi-dimensional tcam architecture with parallel pipelined sub-divided structure,” *In 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 309–314 (2016).

[3] H. Yamaki, H. Nishi, S. Miwa, and H. Honda, “RPC: An Approach for Reducing Compulsory Misses in Packet Processing Cache,” *IEICE Transactions on Information and Systems*, Vol.E103-D, No.12, pp.2590-2599 (2020).
[4] K. Tanaka, H. Yamaki, S. Miwa, and H. Honda, “Evaluating Architecture-Level Optimization in Packet Processing Caches,” *Computer Networks, Elsevier*, Vol.181, No.107550, pp.1-10 (2020).
[5] H. Yamaki, “Effective Cache Replacement Policy for Packet Processing Cache,” *International Journal of Communication Systems*, Wiley, Vol. 33, Issue. 14, pp.1-13 (2020).
[6] H. Yamaki, H. Nishi, S. Miwa, and H. Honda, “Data Prediction for Response Flows in Packet Processing Cache,” *in Proc. 55th ACM/EDAC/IEEE Design Automation Conference (DAC 2018)*, pp.1-6, Jun. (2018).
[7] Mawi working group traffic archive, https://mawi.wide.ad.jp/mawi/ (Accessed on 11/23/2019).
[8] Nlanr amp data | ripe labs, https://labs.ripe.net/datarepository/datasets/nlanr-amp-data/. (Accessed on 01/20/2022).