Vol.2010-ARC-190 No.15
2010/8/3

ルータのアクセス傾向を利用したネットワークプロセッサ搭載用キャッシュ機構

永冨泰次[†]　石田慎一[†]　鯉渕道紘[††]
川島英之[†††]　西宏章[†/†††]

近年，web サービスならびにコンテンツの発展と増大によって，インターネットトラフィックは増加の一途を辿っている．爆発的に増加するビデオトラフィックに加え，IP 電話や会議システム，メッセンジャー，twitter のように，大量の小さなパケットにより構成されるサービスが今後増加し続けることが想定される．これらの膨大なネットワークトラフィックがバックボーンネットワークルータに集中すると，その処理能力が限界を超える可能性が否めない．大容量細粒度通信にバックボーンネットワークルータは対応をせまられている．我々は，過去に通過したパケットの処理をキャッシュに登録することで，極めて似たヘッダを持つ後続のパケットを高速に処理するネットワークプロセッサ, P-Gear を提案してきた．本稿では，その性能改善の鍵を握るキャッシュヒット率向上のために，パケットの転送履歴の解析結果を利用し，ネットワークトラフィックの時間的局所性に加えてトラフィックにおけるパケットの到着の動向を解析し，その特徴を利用することで，キャッシュエントリの生成，廃棄を制御する Multi-Context-Aware Cache の機構を提案する．

Cache Architecture Based Access Trend of Routers for Network Processor

Yasutsugu Nagatomi[†]　Shinichi Ishida[†]
Michihiro Koibuchi[††]　Hideyuki Kawashima[†††]
and Hiroaki Nishi[†/†††]

Recently, Internet traffic has increased due to the development of web services and the increase of contents. In particular, IP telephony, messengers, and twitter are composed by a large number of small packets, and these new services are expected to be increased in the future. For this reason, routers need to handle large-bandwidth fine-grain communication. We have suggested a network processor called P-Gear, which has a special cache mechanism in order to reduce the processing load by taking advantage of localities of the network traffic. In this report, the cache architecture of P-Gear is shown and an enhanced cache mechanism, Multi-Context-Aware cache, is proposed. Multi-Context-Aware cache can improve the speed of packet processing because it can control cache entry by analyzing the packet header.

1. イントロダクション

近年，パソコンや携帯電話などの端末の普及やそれに応じたサービスやコンテンツの増大によってインターネットトラフィックは増加の一途を辿っている．インターネットはクラウドコンピューティング時代を迎え，今後より多くの企業や個人がネットワークとその上に配置されるコンピュータやストレージを通じて大量のデータを利用することが予想される．また，爆発的に増加するビデオトラフィックに加え，IP 電話や会議システム，メッセンジャーのように大量の小さなパケットにより構成されるサービスが今後増加し続けることが想定される．これらの膨大なネットワークトラフィックがバックボーンネットワークルータに集中するとその処理能力が限界を超える可能性が否めない．大容量細粒度通信にバックボーンネットワークルータは対応を迫られている．

従来型のネットワークプロセッサの大多数は個々のパケットが独立に処理可能であることを利用し大量のプロセッシングエレメントの集積やマルチスレッド機構の搭載，もしくはその両方を実装することでパケットレベルの並列処理を行うことにより高スループット化を実現させてきた[1][2]. しかしながら半導体のスケーリング則を利用して半導体を微細化し，集積度を向上させることによって回路の並列化を行うことはリーク電流の面から困難になってきている．このためネットワークプロセッサの処理負荷軽減が必須である．

従来の研究では，パケット解析において比較的実行時間を必要とする処理であるルーティングテーブル検索に必要な処理をキャッシュにより解決する手法が提案されている[3][4][5]．この考え方を拡張した P-Gear と呼ばれるネットワークプロセッサアーキテクチャを我々は提案してきた．P-Gear は処理負担軽減のために時間的局所性を有効に活用しパケットに施される処理そのものをキャッシュする手法である．しかし，処理そのものをキャッシュすることはキャッシュするデータサイズが大きくなってしまいキャッシュできる容量に影響を及ぼす．

そこで本報告ではパケットの動向に着目することでキャッシュエントリ生成のコントロールを可能にし，同じキャッシュサイズでも高いヒット率を達成できる

[†]慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
[††]国立情報学研究所/総合研究大学院大学
National Institute of Informatics/The Graduate University for Advanced Studies
[†††]筑波大学大学院システム情報工学研究科 Graduate School of Systems and Information Engineering, University of Tsukuba

Multi-Context-Aware Cache を提案，評価する．従来の P-Gear ではあらゆるパケットに等しい優先度でキャッシュエントリを作成していたが Multi-Context-Aware Cache ではトラフィックのヘッダ解析を継続的に行い，今後必要となるエントリの準備や不要なエントリの非登録を行うことでキャッシュ領域の効率活用を可能にする．本報告では Multi-Context-Aware Cache の一部機能についてシミュレーションを行い，キャッシュヒット率の評価を行う．

2. 関連研究

パケットの解析において比較的実行時間を要する処理であるルーティングテーブル検索をキャッシュにより解決する手法が提案されている[3][4][5]．汎用プロセッサにおいてはキャッシュが性能に大きな影響を与える一方で，NP では，命令キャッシュは有効だがデータキャッシュは有効でない場合が多い[6]．NP で処理を行うデータの大半は，ネットワークインタフェースから入力されたパケットデータであり，パケットを処理した後は他のインタフェースか場合によってはホスト OS へ転送され，再利用される可能性が少ないためである．そのため，NP においては，パケットデータではなく NP が管理する内部情報に対してデータキャッシュが有効に働くと考えられるが, 特にルーティングテーブルは時間的にも空間的にも参照の局所性が期待できる．文献[4][7]では，レイヤ 3 転送におけるルーティングテーブルの高速検索手法に焦点を当て，キャッシュアーキテクチャを考察している．メモリキャッシュの構成においては，キャッシュのサイズ，ブロックサイズ，連想記憶の幅(associativity) が重要であり，高速な検索を実現しつつルーティングテーブルのエントリ数を効率的に圧縮する技術を提案している．文献[8]ではキャッシュを用いた NP アーキテクチャに焦点をあて，キャッシュメモリのマッピングの仕方， キャッシュサイズ，ブロックサイズに関して，マルチスレッドやマルチコアの機構と関連づけながら動作の最適な手法を考察している．

3. P-Gear

P-Gear は，我々が提案している通信の時間的局所性を利用したネットワークプロセッサである．ここでは，その構成と時間的局所性を利用するためのメカニズムについて説明する．

3.1 P-Gear のアーキテクチャ

ネットワーク通信には短時間に NP について同一の処理が必要であるとみなせるヘッダ情報を持つパケットが多数出現するという時間的局所性が存在する． P-Gear はこの時間的局所性を有効に活用することで，NP 内部に内包する PE (Processing Element) 数を増加させることなくスループットを向上させることができるアーキテクチャである． P-Gear は，P-Engine での処理結果と処理結果適用方法を併せて記録する PLC (Process Learning Cache)を C-Engine に備える．そして，PLC 登録済パケットと同一とみなせる後続のパケットには PU を利用せず， PLC の内容を適用することで高スループット化を実現する．また，あるパケットを PE で処理中にそのパケットと同一とみなすパケットが到着したときには，PLC に目的とするエントリが更新されるまで該当パケットを保持し，後続の別のパケットの処理を進めることにより消費電力の高い PE の利用, 浪費を抑止しつつ, PLC をノンブロッキング化する仕組みを CMH (Cache Miss Handler)として提供する [14].

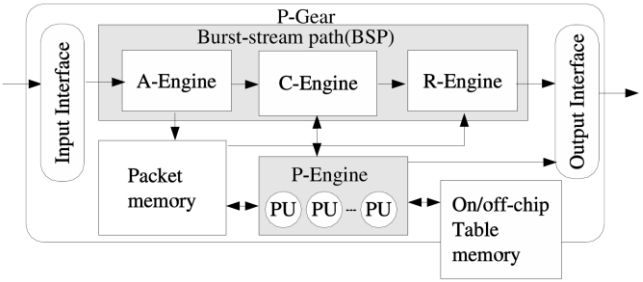

図 1: P-Gear のアーキテクチャ

図 1 に P-Gear のブロックダイアグラムを示す． P-Gear は PLC と CMH を組み込んだパイプライン処理型のネットワークプロセッサである．パイプライン部分を BSP (Burst Stream Path)と呼び，パケットを解析しトークンと呼ぶ内部情報列を生成する A-Engine，PLC を参照してトークンに処理結果と処理結果適用方法を付与および置換する C-Engine，トークンの内容を適用する R-Engine によって構成される．また，P-Gear は， BSP と並列動作する P-Engine と呼ばれる PE 群を集積した機能ブロックを備える．

P-Gear は，ワイヤーレートの高速処理を実現するために，各コントローラ部で利用する命令セットのレベルが異なる．特に，高速性を要求される BSP では，ビット列操作やフィールド検出, 専用ハードウェア処理の ON/OFF などの比較的簡易なマイクロコードレベルのプログラマビリティのみを A-Engine と R-Engine に与える．C-Engine はキャッシュ用の機能ブロックであり，受動的に命令を利用する．P-Engine は，BSP に比べてプログラマブルであり，ワイヤーレート達成の妨げとなりうる複雑

な処理を行う．例えば， BSP から与えられたトークンを操作して各種テーブルメモリの参照用アクセスキーを作成し，テーブルメモリから必要なデータを検索・取得するといった作業が該当する．テーブルメモリの種類としては，経路検索のためのルーティングテーブルやアクセス制限，優先度制御のための ACL (Access Control List)などがある．また，P-Engine は新規ヘッダ生成等も行い，PLC に格納する置換用トークンを生成する．

4. Multi-Context-Aware Cache の提案

P-Gear のアーキテクチャはパケットの時間的局所性を有効に活用し，少ない PU 数で大きなスループットを得ることを目指している．本章では，この P-Gear の性能の鍵を握る PLC の拡張としてキャッシュをより効率的に活用する Multi-Context-Aware Cache について述べ，中でもトラフィック傾向から後続パケットを予測してあらかじめキャッシュエントリの生成を行う Look-AheadCachee に加えて, 時間的局所性のないパケットに対してキャッシュエントリを生成しない機構 Rare-DstPort-Aware Cache, Attack-Aware Cache および P2P-Aware Cache を提案する．

4.1 目的

P-Gear の PLC では同一処理を行うパケットについて，エントリを 1 つ与え，ヒットした場合は専用の BSP と呼ばれるワイヤードロジックによって処理効率化を図っている．しかし，そのエントリの生成，廃棄は全てのパケットヘッダに対して同じ優先度で行われている．ネットワークのトラフィックには時間的な局所性の他にも様々な傾向があることが推測できるため，このように全てのパケットに対して一様にキャッシュエントリを生成するのではなく，変化を与えることで PLC の効率を上げることができると考えられる．例えば，不正なユーザーによってヘッダ情報が異なる大量のパケットを短時間に流布された場合は，キャッシュに無駄なエントリが挿入されるため，スラッシングによるキャッシュ効率低下が発生する．つまり，P-Gear はアタックが存在した場合など，特殊なケースにおいてはキャッシュヒット率が低下すると考えられる．そこで，キャッシュエントリをより柔軟に管理することでの処理効率を改善することが望まれる．このような観点から，従来 SIP-Aware Cache や FTP-Aware, IKE-Aware といった特定のプロトコルに焦点を置いた拡張機能が研究されてきたが，大量のトラフィックが流れている中で SIP/RTP や IKE のパケット数は極めて少なく，またプロトコルが特定されているため適応範囲が狭く，改善は限定的であった．

パケットのヘッダを連続的に解析することでトラフィックの時間的な傾向を探り，あるパケットの到着が後続する他のパケットの処理に影響を与えることが予想できる場合や，あるパケットが時間的局所性のない(単独で現れ，短時間に複数到着することがない)パケットであると推測できる場合に，キャッシュエントリの生成や廃棄を適切に制御し，トラフィックの特徴により影響を受けにくく，より高いキャッシュヒット率を達成する Multi-Context-Aware Cache を提案する．

4.2 適用例

Multi-Context-Aware Cache はトラフィックの傾向を推測しキャッシュエントリの生成，廃棄を制御する．キャッシュエントリの生成や廃棄の制御に有効なトラフィックの傾向とその時の処理として，例えば，パケットに時間的局所性がない場合にキャッシュ登録を避ける，パケットがストリームの最後のパケットである場合に，キャッシュエントリを削除する，あるパケットの到着により，後続するパケットヘッダを予測できる場合にエントリのインジェクションを行う，といったことが考えられる．これらの特徴的なパケットを見つける方法と，それらに対するキャッシュエントリの制御の方法を以降で説明する．

4.2.1 時間的局所性のないパケット

パケットに時間的局所性がない場合とは, そのヘッダ (宛先 IP, ポート, 送信元 IP, ポートの 4tuple)を持つパケットが短時間に一つしか現れないことを指す．その場合，PLC に処理情報を登録しても以後，そのエントリの再利用が期待できず，より再利用される確率が高いエントリが当該エントリにより上書きされる可能性がある．時間的局所性のないパケットが到着した場合は PLC への登録を回避するこで，このようなエントリの上書きを防ぎ，PLC を効率良く扱うことができる．時間的局所性のないパケットかどうかを判断する有効な手法として，Layer4 の TCP ヘッダ内にある宛先ポート調査があげられる．宛先ポートは各種通信サービス（アプリケーションプロトコル）の識別番号であり，提供するサービスと密接な関係がある．再利用性が高いか少ないかは提供するサービスの種類や傾向とも関係があると考えられるため，あるポート番号を宛先ポートとして持つパケットが短時間に連続して現れないことがトラフィック解析から推定できれば，当該ポート番号を有するパケットを時間的局所性がないパケットとして PLC への登録を回避する．このような推定は，宛先ポートを利用するだけでなく，送信元 IP によってもおこなえる場合がある．例えばウィルスに感染したソフトは短時間に大量の不正なパケットを不特定多数の IP アドレスに流布する．これらのパケットは特定の送信元 IP, 宛先ポート番号をもちながらそれぞれ異なる宛先 IP アドレスの情報をヘッダに持っているため，キャッシュのエントリを大きく撹乱すると考えられる．

4.2.2 パケットストリームにおける最後のパケット

パケットが一連の流れの最後であることが分かれば，新たにストリームを開始しない限りこのパケットと同一ヘッダを持つパケットが来ることはないと予測できる．この例として，TCP ヘッダ内に含まれるコードビットの内，FIN ビットが 1 で ACK ビットが 0 である場合が挙げられる．このようなパケットが到着した場合，該当するエントリを無効化すれば PLC の有効利用につながる．ただし， FIN ビットが 1 で ACK ビットが 0 であっても続けてその後に同一ヘッダを持ったパケットが来るようなアプリケーションプロトコルも数多く存在することが推測でき， FIN ビットだけでストリームの最後を判断するのは困難である．アプリケーションの通信特性に依存することが考えられるため， FIN ビットは宛先ポート番号等他のヘッダ情報と組み合わせることで，より効率よくストリームの最後を推測することができると考えられる．

4.2.3 後続するパケットヘッダを予測できるパケット

異なるパケットヘッダを持つ複数のパケット同士で，その片方の解析をすることにより，もう一方のパケットの到着を予測できる場合がある．例として上りのパケットが到着した際，下りのパケットのヘッダを推測することが考えられる．上りのパケットから下りのパケットを推測することは, 送信源と宛先の IP アドレスとポート番号をそれぞれ入れ替えることで予測することができる．その他，FTP (File Transfer Protocol) においては，制御用の通信 Port とデータ転送用の通信 Port の両方を用いて通信を行うため，互いのパケットヘッダは異なるが，制御用の通信 Port を送信元，または宛先 Port に含むパケットが到着した場合には，その後データ転送用のパケットが到着することが予測できる．また，データ通信用の Port 番号はアクティブモードでは固定であり，パッシブモードでも制御通信を行っている間に Port 番号の交換を行うので，その部分を解析することでデータ転送用パケットのヘッダを予測することができる．制御用の通信中にデータ転送用のヘッダ情報および処理情報を作成し，あらかじめ PLC に登録することにより，データ転送の一つめのパケットからキャッシュにヒットさせることができるようになる．このようなキャッシュエントリの準備をする機構については SIP-Aware Cache や FTP-Aware Cache が過去に研究されている[9].

4.3 実装方法

このような Multi-Context-Aware Cache のサブセットとして，時間的局所性のないパケットを動的に予測し，エントリを生成しないキャッシュ機構についてシミュレーションし，評価する．パケットが時間的局所性を持つものか否かは，到着するパケットのヘッダのうち，宛先ポートまた ，送信元 IP アドレスと宛先ポート番号との組み合わせ (以降宛先ポート /送信元 IP と記述する)を連続的に解析して判断する．

4.3.1 Rare-DstPort-Aware Cache

ごく低い頻度でしか現れない宛先ポート番号を見つけることにより，時間的局所性のないパケットを特定することができる．このキャッシュ機構を Rare-DstPort-Aware Cache と呼ぶ．Rare-DstPort-Aware Cache では以下の方法で時間的局所性のないパケットを逐次推測し，キャッシュ登録処理に適用する．

1, あるパケット数のウィンドウを設定し，そのウィンドウ内で到着するパケットそれぞれがヘッダに持つ宛先ポート番号をカウントする．
2, カウンタが一定の閾値に達しなかった宛先ポート番号をそのウィンドウにおける時間的局所性のないパケットを特徴づける宛先ポート番号とする．
3, 次のウィンドウで到着するパケットに対し，2 で候補に挙がっている宛先ポート番号を持つパケットに関しては，キャッシュエントリを作成しない．

4.3.2 Attack-Aware Cache

悪意を持ったユーザーが不特定多数の宛先 IP アドレスの特定ポートに向かって攻撃をすると，異なる宛先 IP アドレスを持ったパケットが短時間で大量に流布される．これらのパケットをキャッシュに登録しない機能を Attack-Aware Cache と呼ぶ．このようなパケットは同一送信元 IP アドレスから到着するという特徴を利用することで予測が可能である．また，これらの特徴を持つパケットに対してキャッシュ登録を防ぐことでキャッシュヒット率の改善が期待できる．Attack-Aware Cache における不正送信元を検出する方法としてハードウェアで実装可能な手法を 2 種類提案する．

手法 1 のアーキテクチャを図 2-4 に示す. 図 2 に示すようにウィンドウサイズを 512 パケットに設定し, それぞれのウィンドウでの解析結果を次のウィンドウに適用する．Phase1 では到着したパケットの送信元 IP アドレスを CRC ハッシングし, 7 ビットのハッシュ値とする．それぞれのハッシュ値毎に 5 ビットのカウンタを用意し，最初に桁が溢れたもの (32 を超えたもの)の送信元 IP アドレスのハッシュ値を保持する．Phase2 では， Phase1 で保持された送信元 IP アドレスのハッシュ値を持つパケットの宛先 IP アドレスを CRC ハッシングし, 3 ビットのハッシュ値とする． 3 ビットの値をアドレスとするメモリを用意し，到着するパケットに対し，生成した 3 ビットハッシュ値をアドレスに持つ領域に 1 を代入する．メモリ内に値が 1 のものが 5 つ以上存在した場合，その送信元 IP アドレスのハッシュ値を Phase3 でキャッシュ登録しない．

手法 2 のアーキテクチャを図 5 に示す．到着するパケットヘッダの送信元 IP アドレスと宛先 IP アドレスを CRC ハッシングし，7bit のハッシュ値とする．送信元 IP アドレスのハッシュ値をアドレスとするメモリに，宛先 IP アドレスのハッシュ値を格納する．次に同じアドレスへのアクセスがあった時に前回の宛先 IP アドレスのハ

ッシュ値と比較して，異なっていれば 3 ビットのカウンタをインクリメントする．カウンタの桁が溢れた時から，カウンタがリセットされるまではその送信元 IP アドレスをキャッシュ非登録の対象とする．

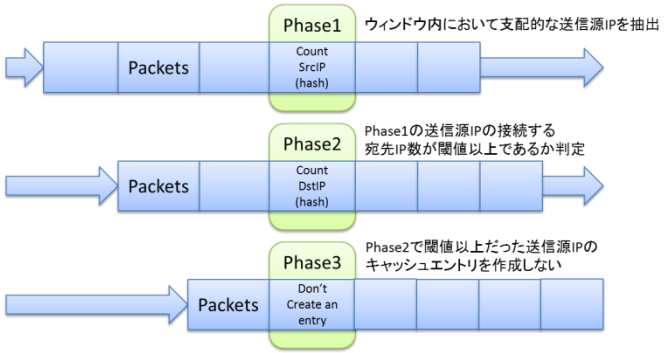

図 2:手法 1 のアルゴリズム

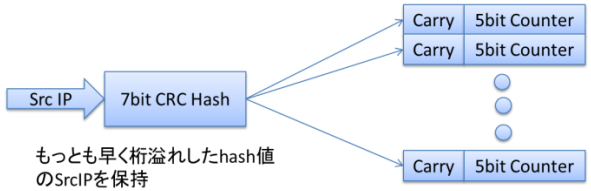

図 3:Phase1 のアーキテクチャ

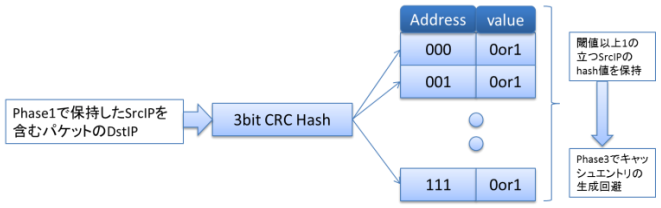

図 4:Phase2 のアーキテクチャ

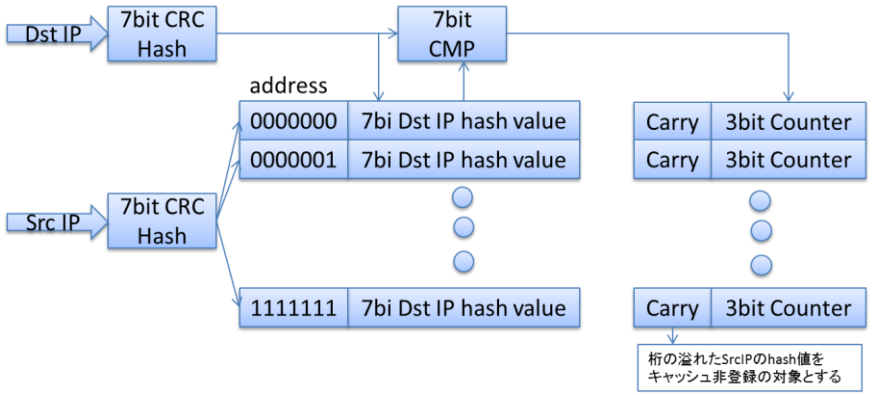

図 5: 手法 2 のアーキテクチャ

4.3.3 P2P-Aware Cache

BitTorrent のような P2P プロトコルは非常に多くのノードと接続する．また，一度限りのパケット転送もキャッシュを乱す要因となる．P2P プロトコルに用いられるポート番号を含むパケットを特定の条件の下でキャッシュエントリを作成しない機構を P2P-Aware Cache と呼ぶ．P2P においても連続転送されるパケットは存在する．このため，一度限りのパケットと連続転送されるパケットをヘッダ情報に含まれるパケット長を利用して区分けし，一度限りしか出現しないと予測したパケットのエントリを生成しないことで，キャッシュ空間の有効活用を行う．

4.3.4 Look-Ahead Cache

ルータではあるパケットが到着した際，そのパケットと逆方向のパケットを後に転送することがある．パケットが到着した際に，宛先と送信源のタプルを入れ替えたキャッシュエントリを作成する機構を Look-Ahead Cache と呼ぶ．図 6 のようにパケットがルータ内のラインカード A を通過した際に，宛先と送信源のタプルを入れ替えた情報と自身のラインカード番号をラインカード B に送信する．この情報をもとにキャッシュエントリを作成することで，後にラインカード B に先ほどと逆向きのパケットが到着した際に，プロセッシングすることなしにラインカード A にパケットを転送することが可能となる．

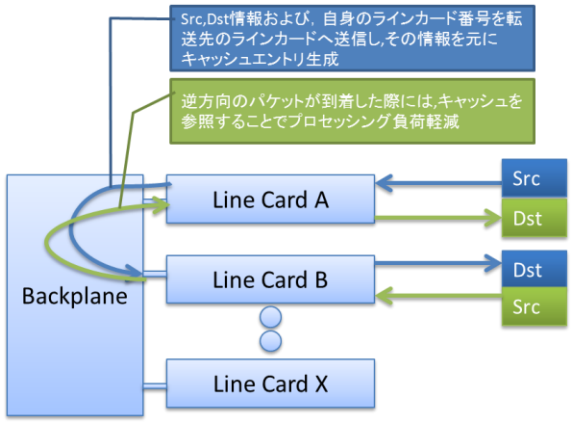

図 6: Look-Ahead Cache のアーキテクチャ

5. 評価

Multi-Context-Aware Cache のサブセットとして, 時間的局所性のないパケットのキャッシュエントリを生成しない Rare-DstPort-Aware Cache ， Attack-Aware Cache, P2P-Aware Cache に加えてトラフィック傾向から後続のパケットヘッダを予測して予めキャッシュエントリを作成する Look-Ahead Cache について命令レベル P-Gear シミュレータと実インターネットトレースを用いた評価を行った．本章ではこれらの評価結果について述べ，考察を行う．

5.1 Rare-DstPort-Aware Cache の評価結果

Rare-DstPort-Aware Cache のシミュレーション結果を示す．評価には WIDE の 2007 年 1 月 10 日の 0 時 0 分 0 秒からキャプチャされた実トレース 15 分間分(パケット数 13，344，532)を用いた．図 7 はウィンドウサイズとそのウィンドウ内で到着したパケット数 n の閾値を変えた時のキャッシュヒット率の変化である．全閾値において，ウィンドウサイズを大きくするにつれてキャッシュヒット率が向上するが，全ての宛先ポートに対してキャッシュエントリを生成した時のキャッシュヒット率を超えることはなかった．原因として，パケット数が極めて少ない宛先ポート番号を予測することは必要なメモリなどの資源が多く必要であり，今回実験した規模では予測が困難であったこと，あるウィンドウで極めて少数しかこなかったパケットが次のウィンドウでは複数現れ，キャッシュミス数が増大したことなどが挙げられる．例えば 5，000，000 パケットをウィンドウサイズとしたとき，そのなかで唯一現れるパケットのユニークな宛先ポート番号は 8，426 種類であった．この中のどれか 1 つのポート番号が次のウィンドウで複数連続して現れることがかなりの確率で発生し，キャッシュミスを増加させる原因となったと考えられる．これは，図 7 において，閾値設定を 0 以上 1 以下とした場合よりも 1 のみとした場合のキャッシュヒット率が高いことからも言える．つまり，この手法はキャッシュ登録を行わないことで得られるキャッシュヒット率の向上よりも，予測が当たらなかった場合に起きるキャッシュミスの増加によるペナルティのほうが大きい手法であると言える．

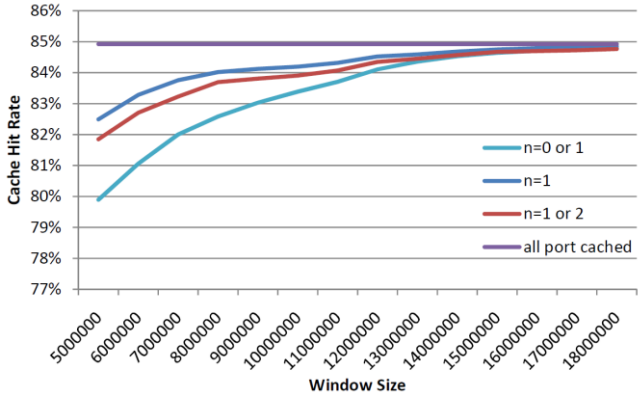

図 7: Rare-DstPort-Aware Cache を用いた際のキャッシュヒット率

5.2 Attack-Aware Cache の評価結果

Attack-Aware Cache のシミュレーション結果を図 8，9 に示す．ウィンドウサイズは 512 パケットとし，この中で 32 パケット以上存在する同一送信源 IP について，その送信元 IP から閾値以上の異なる宛先 IP へ接続していた場合を不正なパケットと見なし，キャッシュエントリの生成を回避した．シミュレーションでは閾値を 4-32 の間で段階的に変化させ評価した．この結果，提案したアルゴリズムでアタック主からのパケット予測が可能となり, 実トレース環境において 0.02%, ポートスキャン攻撃を 50%の割合で混入させたトレース環境においては 4.5%程度キャッシュヒット率が向上した．

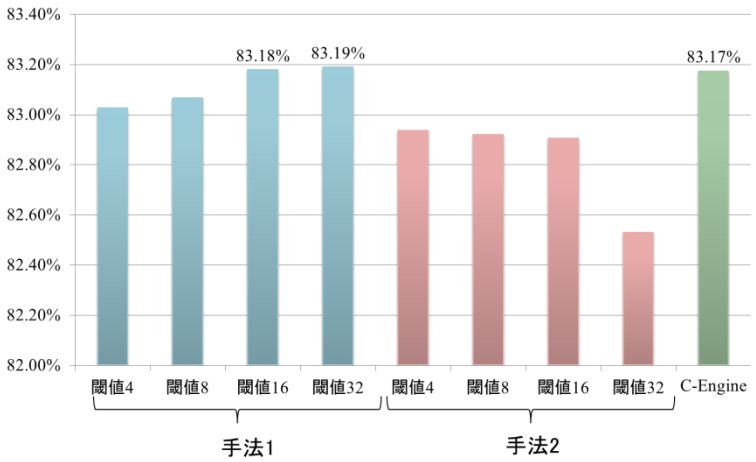

図 8: 実トレースにおける Attack-Aware Cache を用いた際のキャッシュヒット率

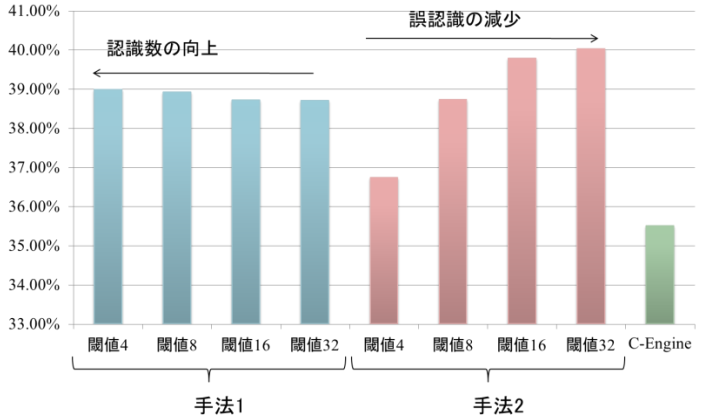

図 9: 50%アタックトレースにおける Attack-Aware Cache を用いた際のキャッシュヒット率

5.3 P2P-Aware Cache の評価結果

P2P-Aware Cache のシミュレーション結果を図 10-12 に示す．評価には WIDE の 2009 年 4 月 2 日にキャプチャされた実トレースを用いた．(以下同様)このシミュレーションでは，P2P プロトコルとして BitTorrent を標的にキャッシュ生成を回避することを行った．比較対象は次の 4 つである．

1. C-Engine のみのケース
2. BitTorrent でよく用いられる 6881 番ポートのみを対象にしたケース(図中では P2P_6881)
3. BitTorrent で用いられる 6881-6999 番ポート全てを対象としたケース (P2P_6881-6999)
4. 6881 番ポートのみを対象とするが，連続的にやりとりされ得るパケットについては，パケット長によって識別しキャッシュエントリを生成させるケース（P2P_P-Length）

BitTorrent では，6881 番ポートから 6999 番ポートまでの任意のポート番号を通信に用いる．しかしながら，このポート範囲は BitTorrent 以外のアプリケーションにも用いられている．そこで，BitTorrent でよく用いられる 6881 番ポートのみを対象にしたケースと 6881-6999 番ポート全てを対象としたケースを別々に評価した．また，BitTorrent において，連続的にやりとりされるパケットに対してもキャッシュエントリを作成しない場合，キャッシュヒットレートが低減する可能性がある．このため，パケット長を用いることで，一度限りのパケット転送であるのか，それとも連続的にやりとりされるパケットであるのかを識別することを行った．この際のパケット長の閾値は 256 バイトとした．

図 10 に各ケースにおけるキャッシュヒットレートの結果を示し，図 11 に C-Engine のキャッシュヒットレートを 1 とした際の各ケースにおけるキャッシュヒットレートの割合を示す．また，図 12 にベストケース，ワーストケースおよび平均値を示す．これらの図より，ケース 3 のヒットレートが概して低い一方で，ケース 2 およびケース 4 のヒットレートは概ねケース 1 より高い値を示していることがわかる．ケース 3 のヒットレートが低い原因として，キャッシュ生成の回避対象としたポート範囲が広く，BitTorrent 以外の，例えば Windows Live Messenger のファイル転送といったキャッシュするべきポート番号も含まれていたためと考えられる．また，図 12 より BitTorrent における連続転送の割合が高い場合に，ケース 2 では 3%程度ヒットレートが低減する場合が見受けられるが，パケット長を考慮したケース 4 において，その影響が 1/3 程度に軽減されていることがわかる．これらの手法におけるキャッシュヒットレートの向上幅はケース 2 では 0.5%程度，ケース 4 では 0.3%程度となった．

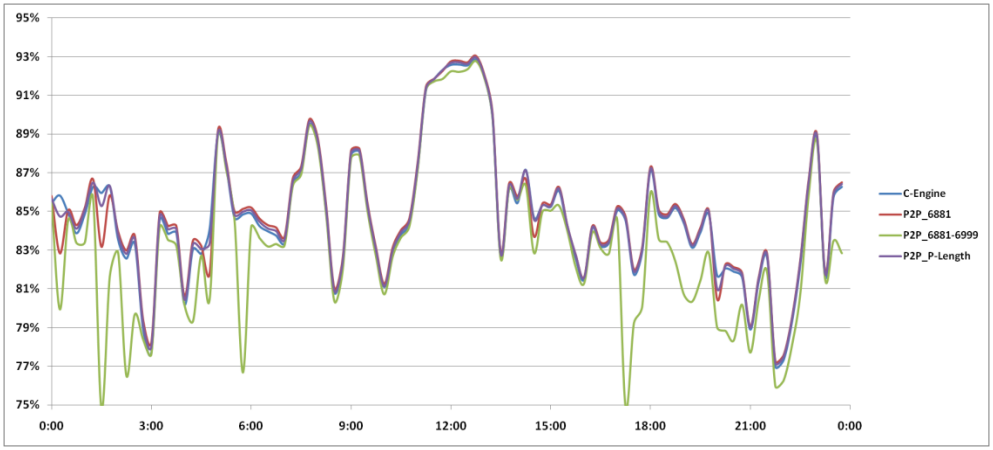

図 10:　各手法におけるキャッシュヒットレート

図 11: C-Engine を基準とした際の各手法におけるキャッシュヒットレート

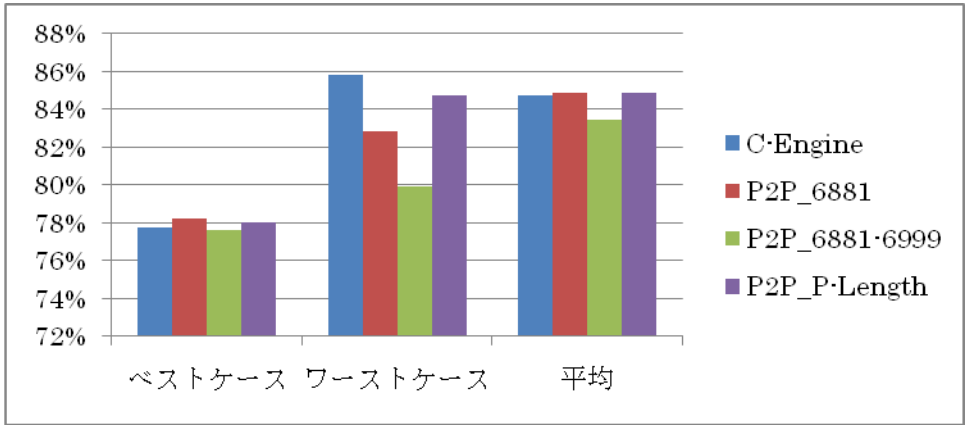

図 12: 各手法における特徴的なキャッシュヒットレート

5.4 Look-Ahead Cache の評価結果

図 13 に Look-Ahead Cache のキャッシュヒットレートを示す．この手法では対応するラインカードとの通信が必要であるため，内部ブロックおよび IO などの遅延を考慮してシミュレーションを行った．具体的には，システムチップの動作クロックを 200MHz とし，IO 遅延を 10 クロック，各チップの内部ブロックの動作遅延をそれぞれ 5 クロックとし，100ns の遅延を見込んだ．図 13 からこの手法は，C-Engine のみの時よりも概ね高いキャッシュヒットレートを示すことがわかる．この手法を利用することによるキャッシュヒットレートの低減は殆どなく，低減の生じた場合でもその幅は 0.1%程度である．その一方で，本手法におけるヒットレート向上幅は 2%程度と，キャッシュミスヒット率を 1 割程度改善可能であり，パケット転送パフォーマンスを 10%程度向上できることが示された．

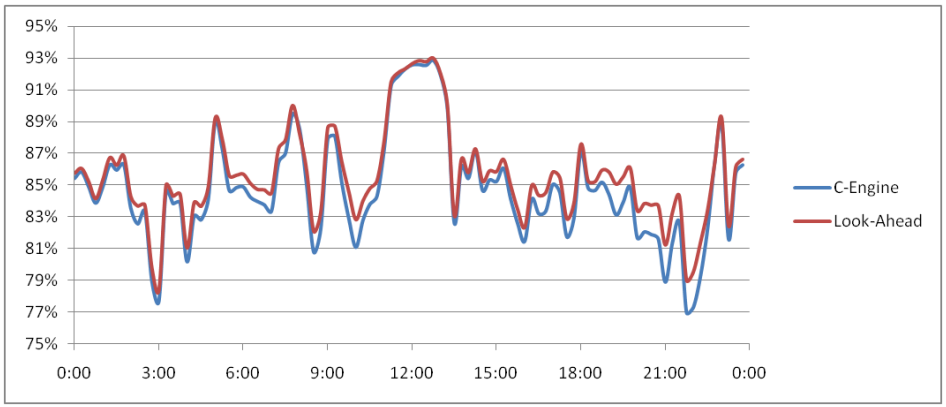

図 13:　Look-Ahead Cache におけるキャッシュヒットレート

6. 結論

本研究室ではキャッシュ機構を用いることでトラフィックの時間的局所性を有効活用し，少ない PE 数で高いスループットを実現できる P-Gear が研究してきた．本論文では，P-Gear の性能の鍵を握る PLC のキャッシュヒット率を向上し，キャッシュ領域をより有効に活用する新しいキャッシュ機構として Multi-Context-Aware Cache を提案し，その一部機能について評価を行った．Multi-Context-Aware Cache はパケットのヘッダを連続的に解析することによりトラフィックの傾向を逐次推測し，キャッシュエントリのインジェクションや不要なキャッシュエントリの廃棄により，キャッシュエントリを適切に管理し，キャッシュヒット率を改善する．本報告では，Multi-Context Aware Cache の一部機能として，トラフィック傾向から後続のパケットヘッダを予測して予めキャッシュエントリを作成する Look-Ahead Cache，時間的局所性のないパケットに対し不要なエントリを生成しない機構を提供する Rare-DstPort-Aware Cache，さらに Attack-Aware Cache および P2P-Aware Cache についてシミュレーションを行い，評価した．

Rare-DstPort-Aware Cache では，到着頻度が極めて少ない宛先ポート番号をもったパケットは同様のパケットが後続する可能性が低いと判断してキャッシュ登録処理を回避することでキャッシュヒット率の向上を図る．Attack-Aware Cache では短時間に同一の送信元 IP アドレスから大量の宛先 IP アドレスを持って到着する不正なトラフィックのパケットに対して，その送信元からは同じ宛先 IP アドレスを持ったパケットが後続する可能性が低いと推測して，キャッシュ登録を回避することでキャッシュヒット率の向上を図る．評価の結果，Rare-DstPort-Aware Cache ではキャッシュヒット率の向上が見られなかったことから，到着頻度が極めて少ない宛先ポート番号は予測が難しく，より高い精度の予測が必要であることがわかった．また，Attack-Aware Cache では，不正なトラフィックの送信元 IP アドレスの一部を特定しキャッシュエントリの作成を回避することで，キャッシュヒット率が 0.02%から 4%程度向上した．より高い精度で不正送信元 IP アドレスを複数特定することが可能になれば，より高いキャッシュヒット率の向上が期待できる．P2P-Aware Cache では，多数のノードと短期間に接続し，キャッシュを乱す BitTorret のパケットに対しキャッシュエントリを生成しないことで，ヒット率が 0.3-0.5%程度向上した．BitTorrent プロトコルを用いるパケットにおいて，連続的に通信するパケットの区分け精度を向上させることができればより高いヒット率を期待できる．Look-Ahead Cache では後続のパケットを予測することでキャッシュヒット率が 2%程度向上した．これはミスヒット率の 1 割程度に該当し，パケット転送のパフォーマンスを 10%程度向上できる可能性を示した．後続パケットを予測する手法の拡充や予測精度向上が今後の課題である．

謝辞　この研究は，独立行政法人情報通信研究機構の高度通信・放送研究開発委託研究「新世代ネットワーク技術戦略の実現に向けた萌芽的研究」および文部科学省科学技術研究費補助金基盤研究 C「安心・安全な情報提供を可能とするインターネット基盤の構築に関する研究」の一環として行われた．

参考文献

[1] P. Crowley and M.A. Franklin and H. Hadimioglu and P.Z. Onufryk. Network Processor Design Issues and Practices Volume 1. Morgan Kaufmann, 2002.

[2] Douglas E. Comer. Network Systems Design using Network Processors (IXP1200Version). Pearson Prentice Hall, 2004.

[3] Tzi Cker Chiueh and Prashant Pradhan. High-Performance IP Routing TableLookup Using CPU Caching. In Proceedings of INFOCOM'99, pp. 1421–1428, 1999.

[4] Tzi Cker Chiueh and Prashant Pradhan. Cache Memory Design for Network Processors. In Proceedings of IEEE High Performance Computer Architecture Conference(HPCA) 2000, pp. 409–419, Jan. 2000.

[5] Bryan Talbot, Timothy Sherwood, and Bill Lin. IP Caching for Terabit SpeedRouters. In Proceedings of Global Communication Conference (Globecom'99), pp.1565–1569, 1999.

[6] 河合栄治, 門林雄基, 山口英. ネットワークプロセッサ技術の研究開発動向. 情報処理学会論文誌　コンピューティングシステム, Vol. 45, No. SIGI(ACS4), Jan. 2004.

Vol.2010-ARC-190 No.15
2010/8/3

[7] K.Gopalan and T.C.Chiueh. Optimizations for Network Processor Cache. In Proceedings of SC2002 High Performance Networking and Computing, 2002.
[8] Zhen Liu, Jia Yu, Xiaojun Wang, Bin Liu, and Laxmi Bhuyan. Revisiting the Cache Effect on Multicore Multithreaded Network Processors. In Proceedings of the 2008
[9] 河合 満. Cache を利用したネットワークプロセッサの処理効率を向上させる MultiAware Cache の実装と評価. Master’s thesis, 慶應義塾大学大学院理工学研究科総合デザイン工学専攻, Mar. 2008.