ハイパフォーマンス
コンピューティング 93−13
（２００３．３．１２）

プロセスの優先度に基づいてサービス品質を制御するネットワーク処理機構

尾崎亮太† 中山泰一†

インターネットの発展とユーザの増加に伴い，サーバシステムに対する要求が高まっている．最近ではサービス内容に従ったサービス品質の制御がサーバシステムに対する新たな要求となっている．UNIX 系 OS では，優先度を用いて個々のプロセスに対するサービス品質の違いを実現している．現在の UNIX 系 OS では，ネットワークの受信処理において優先度が反映されていないという問題が存在する．その問題は SMP 型計算機で特に顕著に現れる．

本研究では，パケットの選択破棄と割込みの抑制という手法を導入することでこの問題に対応した．linux-2.4.18 上にシステムを実現し，SMP 型計算機上で性能評価実験を行なったところ，既存のネットワーク処理機構と比較して，優先度をより正しく反映した処理を行なうことを確認した．

A Network Processing System for QoS Control Based on Process Priority

Ryota OZAKI† and Yasuichi NAKAYAMA†

The explosive growth of the Internet and the number of its users places interesting new demand for the server systems. Service managements according to contents of service become new requirements for server systems. UNIX based operating systems provide service managements to individual processes using priority. On network receive processing, however, priority does not work adequately. In paticular, this problem have a much greater impact on SMP machines.

To overcome this problem, we propose two techniques, *dropping lower-priority packets* and *suppression of interrupts*. We have designed and implemented proposed system on linux-2.4.18, and have evaluated comparing with original network processing system. Experimental results show that our system have an advantage over original network processing system.

1. はじめに

インターネットの発展とユーザの増加に伴い，種々のサービスを提供するサーバシステムへの要求が高まっている．サーバシステムには多数のクライアントの要求に応える処理性能やシステムの安定性が必要である．そのような要求に対し，SMP 型計算機などの導入による処理性能の強化や，オペレーティングシステム (以下 OS) のネットワーク処理機構の改善による対応がなされている[1)~6)]．最近では，サービスの品質を制御できることがサーバシステムに対する新たな要求となっている．

UNIX 系 OS では，優先度を用いて個々のプロセスに対するサービス品質の違いを実現している．通常は CPU 資源の利用時間に差を持たせ，高い優先度をもつプロセスには他のプロセスより多くの CPU 資源を割り当てる．ところが，現在の UNIX 系 OS では，ネットワークの受信処理において優先度が反映されないという問題が存在する．この問題は特に SMP 型計算機において顕著に現れる．

本研究では優先度を適切に反映するネットワーク処理機構を実現することを目的とする．ネットワークの負荷が高いとき破棄するパケットの選択に優先度が考慮されていない問題に対して，パケットの選択破棄という手法を導入する．また割込みにより優先度の高いプロセスの処理が阻害されている問題に対して，割込みの抑制という手法を導入する．

パケットの選択破棄では受信したパケットの優先度を調べ，優先度の高いパケットを優先的に残し，優先度の低いパケットを代わりに破棄する．割込みの抑制では，優先度の高いプロセスへの割込みを禁止しプロ

† 電気通信大学 情報工学科
Department of Computer Science, The University of Electro-Communications

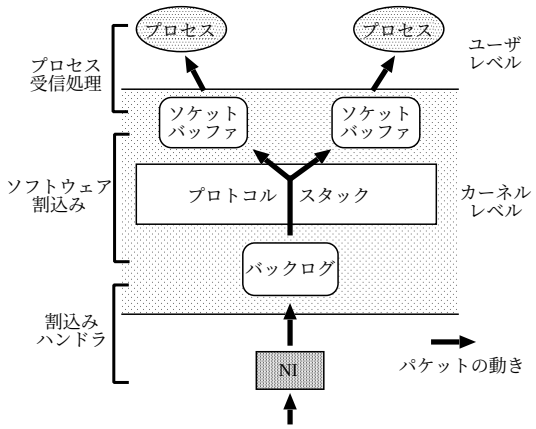

図 1 パケット受信処理
Fig. 1 The process of receiving packets

セスのパケット受信処理が割込みにより阻害されることを防ぐ．

これらの手法を導入した試作システムを設計・実現した．SMP 型並列計算機上で性能評価実験を行なったところ，既存のネットワーク処理機構 (Linux 2.4.18) と比べて，サーバプロセスの優先度を反映していることが確認された．

2. ネットワーク処理機構

2.1 概　　要

ネットワーク処理機構とは OS においてネットワーク資源を管理する部分を指す．

OS は計算機を効率的に制御・管理しユーザに使いやすい環境を提供するためのソフトウェアの集合体である．OS の中において CPU やメモリ，ハードディスクなどのハードウェア資源の管理などを行ない，アプリケーションプログラムに資源を提供する，機能的中核部分をカーネルという．UNIX 系 OS におけるカーネルでは CPU やメモリなどのハードウェア資源の他にプロセスやファイルシステムなどの仮想化された資源などを管理している．ネットワーク資源はそれらと同様カーネルで管理されている資源の一つである．

2.2 パケット受信処理

外部から送られて来たパケットは割込みハンドラ，ソフトウェア割込みで必要な処理を施されプロセスへ渡される (図 1)．

ネットワークインタフェース (以下 NI) にパケットが到着すると割込みハンドラが起動し，パケットに対する処理を行なう．パケットは一時的にバックログへ格納される．しかる後 OS の遅延処理機構であるソフトウェア割込みによりプロトコル処理を施されソケットバッファへ格納される．

2.2.1 バックログ

バックログとは，パケットを一時的に保存しておくため用意されたバッファである．バックログに格納できるパケットの数には最大値が設定され，その数を越えるパケットは破棄されるのが一般的である．

割込みハンドラは送られて来たパケットを一旦バックログに格納し処理を終える．バックログに格納されたパケットは，後でソフトウェア割込みによって取り出され処理される．

2.2.2 ソケットバッファ

ソフトウェア割込みではパケットに対しプロトコル処理をはじめ様々な処理を行なう．パケットの宛先が自ホストであることが判るとそのパケットは対応するソケットが持つバッファに格納される．

カーネル内部においては外部から送られて来たパケットは，このソケットが終点である．パケットはソケットに用意されたバッファに溜められ，プロセスの受信処理を待つ．バックログと同様，バッファのサイズの最大値を越える場合新たなパケットは破棄される．

2.3 問　題　点

UNIX 系 OS では重要度の違うプロセスに優先度を付け資源割当てに差を持たせている．しかしネットワーク処理機構においては，この優先度がうまく機能していない．

2.3.1 パケット破棄ポリシ

ネットワーク処理機構では外部から送られて来るパケットをすべて同等に扱うため，優先度の高いプロセスへのパケットが優先度の低いプロセスへのパケットに CPU 時間を不当に取られるという問題がある．負荷が高くなくプロセッサの能力に余裕がある場合，処理量の不適切さは問題ではない．しかしネットワークの負荷が上がりネットワーク処理の占める割合が増えた場合，この不適切さがプロセスと対応するパケットに対する CPU 時間のバランスに大きな影響を及ぼす．

割込みハンドラでは処理時間の短縮のため到着したパケットをバックログに入れる処理のみ行なう．もしバックログが満杯であった場合，パケットは即破棄される．たとえ到着したパケットの優先度がバックログにあるパケットのものより高い場合でも，到着したパケットが破棄される．

2.3.2 割込みによるプロセスの処理の阻害

優先度の高いプロセスは優先度の低いプロセスに比べ CPU が割り当てられる頻度が高く割当て時間が多くなる．しかし，プロセスの優先度が高く CPU への割当て時間が多いとしても，負荷が高くなり割込みの

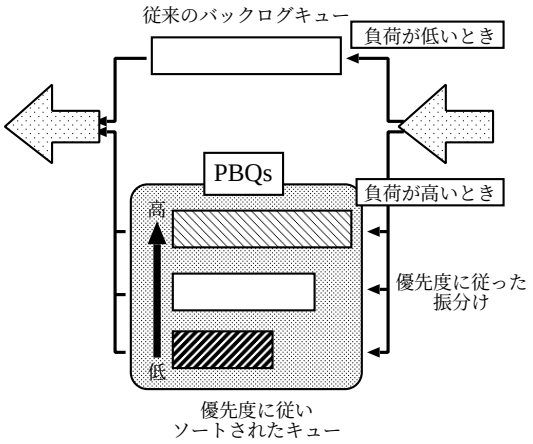

図 2 提案するバックログの構造
Fig. 2 Structure of proposed backlog

発生数が多くなればそれだけ実行が邪魔されてしまう．パケットがソケットバッファにたどり着いたとしても，プロセスが受信処理を行なう前にソケットバッファが満杯になりパケット破棄が起き，その結果としてプロセスのパケット受信量は低下する．

3. 提案するネットワーク処理機構

3.1 設計方針

前章で提起した問題点を受け，プロセスの優先度を適切に反映するネットワーク処理機構を提案する．本研究ではパケットの選択破棄と割込みの抑制という手法を導入する．バックログとソケットバッファにおいて，捨てられるパケットを調整することで優先度に従ったパケット制御を実現する．本来捨てられてしまうパケットを調節することで，無駄なパケット落ちをなくし，性能が低下することを防ぐ．

3.2 パケットの選択破棄

通常プロトコルに従いパケットの処理を行なう場合，そのパケットが送られるべきプロセスが判明するのは，処理の最後であるソケットバッファに繋ぐところである．この段階でパケットの帯域制御を行ったとしても，既に CPU 時間を大量に消費してしまっている．帯域制御に対するその効果はほとんどないどころか，スループットを落とすことになる．

そこで NI からパケットを受け取った段階で送信先プロセスを判別しそのパケットの優先度を割り出す．CPU 時間を消費していない段階で，割出した優先度を基づき破棄するパケットを決定する．

本システムではこの機構を実現するため，バックログに新たなキュー構造を導入する (図 2)．以下では導入するキュー構造を PBQs (Priority Based Queues) と呼ぶ．負荷が高くなったときに PBQs を利用する．PBQs は優先度に従いソートされた複数のキューを持ち，パケットの振分けを行なう．それぞれのキューにあるパケットを調べることにより，破棄すべきパケットを決定する．負荷の低いときは従来のバックログキューを利用し，バックログにおけるパケット振分けのコストがかからないようにする．

以下ではバックログに対するパケットの格納と取出し処理について述べる．

3.2.1 バックログに対する処理

図 3 でバックログに対するパケットの格納処理，図 4 でバックログに対するパケットの取出し処理について，疑似コードで示す．以下は疑似コードのための諸定義である．

- $P_{<id>}$: パケットを表す．$<id>$ は識別名である．
- $Q_{<id>}$: パケットを格納するキューを表す．$<id>$ は識別名である．特に従来のバックログキューを Q_{orig} と表記する．
- $P \leftarrow_{head} Q$: Q の先頭から P を取り出すことを表す．
- $P \leftarrow_{tail} Q$: Q の末尾から P を取り出すことを表す．
- $Q \leftarrow_{tail} P$: Q の末尾にパケット P を格納することを表す．

パケットの格納処理では，優先度を割り出し対応するキューにパケットを格納する．PBQs が満杯であった場合，パケットの優先度より低い優先度のキューから一つ選んでその中のパケットを破棄する．ただしキューの中にパケットがある一定の数ないときは，新しいパケットの方を破棄する．これにより，ある優先度の低いパケットを過剰に破棄して性能が低下することを防ぐ．

パケットの取出しは優先度の高いキューから先に取って行く．この処理は Linux の $O(1)$ スケジューラと同様の手法を用いており，計算量は $O(1)$ である．

3.3 割込みの抑制

ネットワークの負荷が高くなり，割込みの発生が多くなると実行中のプロセスへの割り込みの回数が増加する．割り込まれる回数はプロセスの実行頻度に比例するため，優先度の高いプロセスほど割り込まれる回数も多くなる．

本研究では割込みの発生回数が増加したとき，プロセスの優先度に従い割込みを抑制する処理を導入する．パケットを待つプロセスに処理が移行したときプロセスの優先度に比例した割合で割込みを禁止にする．

これにより優先度の高いプロセスに多く CPU 時間が割り当てられ，より多くのパケットを受信することが可能となる．

```
1:  function Backlog_enqueue(P_in)
2:    if (負荷が低い) then
3:      Q_orig ←_tail P_in
4:    else
5:      PBQ_enqueue(P_in)
6:    endif
7:  end

8:  function PBQ_enqueue(P_in)
9:    // P_in の優先度 prio を割り出す
10:   prio := determine_priority(P_in)
11:   // prio を基にキュー Q_in を決定する
12:   Q_in := determine_queue(prio)
13:   if (PBQs が満杯) then
14:     // prio より優先度の低いキューの中から
15:     // ランダムにキューを決定し Q_sac とする
16:     Q_sac := determine_sacrifice(prio)
17:     if (Q_sac に一定以上のパケットがない) then
18:       P_in を破棄する
19:       return
20:     endif
21:     P_sac ←_tail Q_sac
22:     P_sac を破棄する
23:   endif
24:   Q_in ←_tail P_in
25: end
```

図 3 バックログへのパケットの格納 (割込みハンドラ)
Fig. 3 Store packet into backlog (Interrupt handler)

3.3.1 割込み禁止判定

コンテクストが切り替わるときに割込み禁止判定を行なう．本研究では以下の条件が満たされたとき割込みを禁止する．

- 負荷が高い．
- プロセスがサービスを提供している．
- プロセスの優先度に従った確率による判定に成功した．
- ソケットバッファのサイズが閾値を越えている．

4. 性能評価実験

受信パケット数，パケット破棄比率，CPU 時間を計測する評価実験を行なった．また受信パケット数，パケット破棄比率の実験のおいては既存のシステムとの性能低下率を計測した．

4.1 実験環境

実験環境として，ネットワークで接続された 1 台のサーバマシンと 8 台のクライアントマシンから成るネットワーク環境を構築した (表 1)．

サーバマシンとしては 4 台のプロセッサと 2 枚の Gigabit Ethernet カードを持つ SMP 型計算機を用いた．クライアントマシンとしては Fast Ethernet カードを持つ 8 台の PC を利用した．サーバマシンとクライアントマシンはスイッチを介して接続されており，サーバマシンの NI 1 つに対し 4 台のクライアントマシンが接続される．サーバ・スイッチ間は Gigabit Ethernet で接続され，クライアント・スイッチ間は Fast Ethernet で接続されている．

```
1:  function Backlog_dequeue()
2:    if (Q_orig にパケットがある) then
3:      P_out ←_head Q_orig
4:      return P_out
5:    else
6:      return PBQ_dequeue()
7:    endif
8:  end

9:  function PBQ_dequeue()
10:   if (PBQs にパケットがある) then
11:     // パケットを持つキューのうち
12:     // 最も優先度の高いキュー Q_out とする
13:     Q_out := determine_max_priority_queue()
14:     P_out ←_head Q_out
15:     return P_out
16:   else
17:     return NULL
18:   endif
19: end
```

図 4 バックログからのパケット取出し (ソフトウェア割込み)
Fig. 4 Pick up packet from backlog (Software interrupt)

サーバマシンには UDP パケットを受信し続けるサーバプログラムを用意する．一方クライアントマシンには UDP パケットをサーバプログラムに送信し続けるクライアントプログラムを用意する．

4.2 受信パケット数

サーバマシンでは優先度の異なる 10 個のサーバプ

表 1 実験環境
Table 1 Experimental environments

ネットワーク	
スイッチ	CentreCOM FS909GT V1 1000BASE-T ポート ×1 100BASE-TX ポート ×8

クライアント	
プロセッサ	Intel Celeron 800MHz
メモリ	256MBytes
OS	RedHat Linux 7.1 (カーネル 2.4.7)
NI カード	DEC 21140 chip 100Mbps

サーバ	
プロセッサ	Intel PentiumIII Xeon 500MHz (2 次キャッシュ 512KB) × 4
メモリ	256MBytes
OS	RedHat Linux 8.0 (カーネル 2.4.18)
NI カード	Intel PRO/1000XT PWLA8490XT 1Gbps

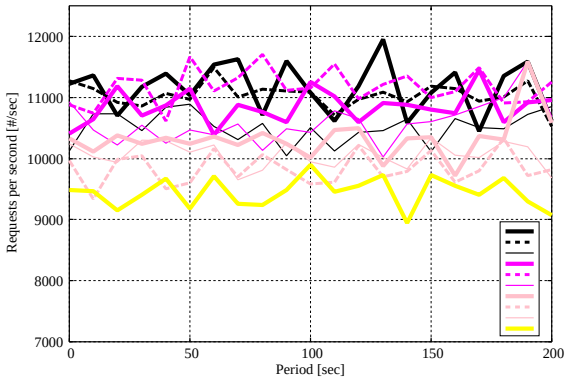

図 5 受信パケット数 (既存のシステム)
Fig. 5 Number of received packets (original system)

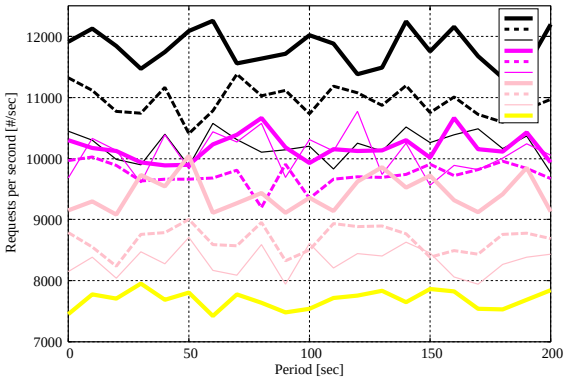

図 6 受信パケット数 (本システム)
Fig. 6 Number of received packets (our system)

ログラムを実行させ，クライアントプログラムはクライアントマシン毎にサーバプログラムの数だけ実行させ，全体としてそれぞれのサーバプログラムへ均等な量のパケットを送信する．

この実験ではそれぞれのサーバプログラムが受信したパケットの数をカウントし，比較する．優先度の高いサーバプログラムへより多くのパケットが到着し，優先度の低いサーバプログラムへのパケット数が高いサーバプログラムの受信パケット数を越えないことが，優先度を正しく反映していると言える．

本研究で提案したネットワーク処理機構の実験結果を図 6 に示す．比較として既存のネットワーク処理機構の結果を図 5 に示す．

縦軸がサーバプログラムが受け取ったパケット数，横軸が時間の推移を表している．それぞれの折れ線は優先度の違うサーバプログラムが受け取ったパケット数を表している．優先度が適切に反映されているならば折れ線は交差することがない．

図 5 を見ると交差している折れ線が多く見受けられる．一方，図 6 では交差の数が少ない．これにより，本研究で提案したネットワーク処理機構が既存のシステムに比べ，優先度の違いを適切に処理していることが判る．

4.3 パケット破棄比率

優先度の異なる 4 個のサーバプロセスを動かし，バックログにおけるパケット落ちの比率を調べる (図 7)．縦軸は優先度の最も高いプロセスのパケット落ち数を 1 としたときのパケット落ち数の比率を表している．

本システムでは低い優先度の方がより多くパケットが落ちていることが見て取れる．それに対し既存のシステムでは，パケット落ちの比率に差が見られない．これにより本システムが優先度を反映した処理を行なっていることが判る．

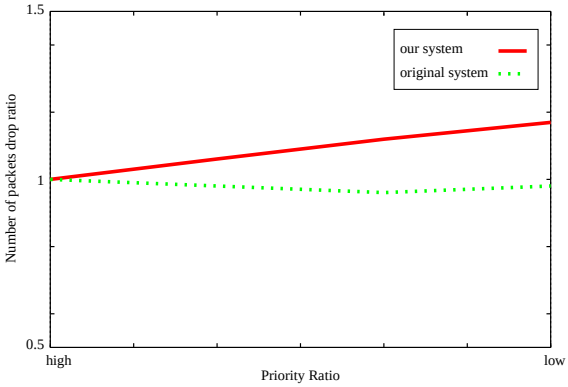

図 7 パケット破棄比率
Fig. 7 Ratio of dropped packets

4.4 CPU 時間

割込みの抑制が正しく働いているか確かめるためプロセスごとの CPU 時間を計測した (図 8)．横軸はプロセス優先度，縦軸はプロセスが消費した CPU 時間を表わす．この実験では 4 つのサーバプロセスを用いた．

割込みの抑制が正しく働けば，優先度の高いプロセスにより多くの CPU 時間が割り当てられるはずである．図 8 を見ると，本システムの方がプロセスに対し CPU 時間を多く割り当てていることが判る．特に優先度の高いプロセスに対して高い効果を示している．

これにより割込みの抑制が正しく働き，優先度の高いプロセスにパケット受信の機会を多く与えていることが判る．

4.5 性能低下率

4.2 節と 4.3 節の 2 つの実験における性能低下率を調べた．4.2 節における性能低下率は 7%，4.3 節における性能低下率は 3%であった．この結果から本システムのオーバヘッドは実システムにおいても影響が小

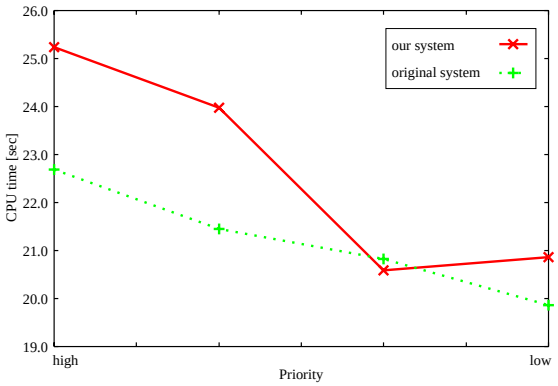

図 8 CPU 時間
Fig. 8 CPU time

さいと思われる．

5. 関連研究

LRP[6] ではパケットを待つプロセスが実行中である場合のみ対応するパケットの受信を許すことで，プロセスの優先度を間接的にパケット受信処理に反映させている．この手法によりあるプロセスに対するネットワーク負荷が高くなっても他のプロセスのネットワーク処理に影響を与えない．しかし本研究と異なり SMP 型計算機に対する考慮はされていない．

また Scout OS[1] では Path と呼ばれる通信路を概念化した構造を導入している．Path を用い静的にルーティングを決定しておくことで通信の最適化を図っている．しかし本研究の手法のように他の優先度のパケットを落とすなど，互いの通信路が影響を及ぼすような処理は行なっていない．

Banga ら[2] は Resource Container と呼ばれる実行主体と資源とを分離するための概念を導入している．これにより資源割当てを正しく行なうことを目的としている．しかし本研究のように割込みに対する考慮はなされていない．

6. おわりに

本研究では優先度を適切に反映するネットワーク処理機構を設計・実現した．ネットワーク処理機構について述べ，その問題点の言及を行った．パケットの選択破棄と割込みの抑制という手法を導入し，問題点を克服するネットワーク処理機構の設計・実現，およびその評価を行った．その結果本システムの有効性を確認出来た．

ネットワーク負荷が高い状況において，ネットワーク処理機構の受信時にプロセス優先度が正しく反映されないという問題が存在する．

パケットの受信処理においては，その優先度は反映されておらずすべてのパケットが同等に処理される．そのためバックログが満杯の時に発生するパケット破棄では，優先度の高いパケットが不当に破棄されるという問題が起こる．本研究ではバックログにおいてパケットの選択破棄を行なうことでその問題に対応した．

またパケット到着などのイベント時に起こる割込みが優先度の高いプロセスの処理を阻害する問題がある．本研究では割込み発生を制御することでこの問題に対応した．

評価実験を行ない本システムが既存のシステムに比べ，パケット受信処理においてより優先度を反映させていることを確認した．

参考文献

1) Mosberger, D. and Peterson, L.L.: Making Paths Explicit in the Scout Operating System, USENIX Symposium on Operating Systems Design and Implementation, pp.28–31 (1996).
2) Banga, G., Druschel, P. and Mogul, J.C.: Resource Containers: A New Facility for Resource Management in Server Systems, In Proceedings of ACM Symposium on Operating Systems Design and Implementation, pp.45–58 (1999).
3) Brustoloni, J., Gabber, E., Silberschatz, A. and Singh, A.: Signaled Receiver Processing, In Proceedings of the USENIX 2000 Annual Technical Conference, pp.211–223 (2000).
4) Mogul, J.C. and Ramakrishnan, K.K.: Eliminating Receive Livelock in an Interrupt-Driven Kernel, ACM Transactions on Computer Systems, pp.217–252 (1997).
5) Hansen, J. S. and Jul, E.: Latency Reduction Using a Polling Scheduler, In Proceedings of the Second Workshop on Cluster–Based Computing, pp.27–31 (2000).
6) Druschel, P. and Banga, G.: Lazy Receiver Processing (LRP): A Network Subsystem Architecture for Server Systems, In Proceedings of Second USENIX Symposium on Operating Systems Design and Implementation (OSDI '96), pp.261–276 (1996).