システムソフトウェアと
オペレーティング・システム 88−11
（２００１．７．２７）

ファイアウォールへの資源予約の適用と検証

梶原 史雄† 岡部 恵一† 盛合 敏‡
†NTT 情報流通プラットフォーム研究所 ‡株式会社ぷららネットワークス

昨今ではPC 能力の向上により安価なルータやファイアウォールシステムとしてPC 用 UNIX を流用する場合が増えてきた. しかし, DoS 攻撃 (Denial-of-Service Attack) などの原因により大量のパケットが到着すると割り込みによるパケット処理がPC の処理の大部分を占めるようになりルータやファイアウォールとして機能し得なくなる. 本論文では, プロクシ型ファイアウォールを主なターゲットとして, 処理するパケット数を予約してPC のパケット処理の負荷を軽減するパケット処理予約機構と同時にプロクシアプリケーションが利用するCPU 時間, ネットワーク帯域という資源を個々のサービスごとに予約することで, サービス毎の独立性を確保することを提案する. 本論文ではRT-Mach にネットワーク帯域予約機構とパケット処理予約機構を実現し, 代表的なWWW サービスを対象に評価を行って有効性を検証する.

Applying resource reservation to firewall

Masao Kajihara† Keiichi Okabe† Satoshi Moriai‡
†NTT Information Sharing Platform Laboratories ‡Plala Networks Inc.

PC can use as a firewall, because of increase of their power. However, a lots of packets make PC firewall malfunction, so they are processed by interrupt calls and they use most of processor power. We propose to apply resource reservation, which are processor, outgoing network and incoming network reservation, to PC firewall to avoid PC firewall malfunction. And we implement them on Real Time Mach.

1 はじめに

ファイアウォールはインターネットとイントラネット間など二つ以上のネットワークの境界に配置し, HTTP などのアプリケーション層あるいはIP などのネットワーク層での中継を行なう. 通常の運営方法ではネットワーク境界上のファイアウォールは１つであるが, ファイアウォールが中継を行なう対象のサーバは複数である.

ファイアウォールはホストに対するネットワーク経由の攻撃に対して有効に働き防御効果を発揮するが, ある種の状況下では逆にネットワーク麻痺の原因ともなる. ネットワーク層で中継するIP パケットがDoS 攻撃 (Denial of Service Attack)[5][6] などの原因により莫大な数になると, ファイアウォールはパケット処理にプロセッサ時間を浪費させられてしまい中継が行えなくなる. また, プロクシアプリケーションによりアプリケーション層での中継を行う場合はプロクシアプリケーションのバグを突いてファイアウォールのプロセッサ時間やネットワーク帯域などの資源を浪費させ中継を行えなくできる. このような状況下に置かれたファイアウォールはネットワーク間で中継がすべて行えなくなる.

このような資源の浪費に対して特定のプロセスによる資源の利用を制限する資源予約機構が研究されてきた. 資源予約機構はプロセッサ時間の予約, ネットワーク帯域の予約, ディスク帯域の予約などが研究されており, 従来の資源割り当て単位であるプロセスとは直交した予約ドメイン (reservation domain) を利用して資源を予約する仕組みとなっている場合が多い [1]. この資源予約機構によりプロクシアプリケーションによる資源の浪費は回避でき, ファイアウォールが中

継を行えなくなるという事態を回避できる.

しかしながら, 先に挙げたように莫大な数の IP パケットが到着した場合にファイアウォールがパケット処理のみでプロセッサ時間を浪費してしまう事態は回避できない. 大部分の資源予約機構はユーザ空間を対象としているためにカーネル内部の処理であるパケット処理に対する予約は考慮されていないためである.

本論文ではこのような問題に対し, パケット処理の大部分を資源予約対象とできるマイクロカーネル方式とパケットフィルタリング [12][13] を利用した入力パケット数予約機構とを組み合わせた手法を提案する.

2 関連研究

2.1 resource container

resource container[1] はスレッドやファイル, ソケットを最小単位とした細粒度の資源予約が可能な予約ドメインであり, 階層的な資源予約が行える. さらに LRP(Lazy Receiver Processing)[2] によりカーネル内でのネットワーク処理に費されるプロセッサ時間が正しく resource container にチャージされるように工夫がなされている.

resource container では明示的な入力パケット数あるいは入力ネットワーク帯域予約が行えない. また, NIC でのパケット処理が行えることを前提としているため, PC ファイアウォールなどが利用する廉価な NIC での実現は困難である.

2.2 Linux/RK

Linux/RK[3] はリアルタイム OS の枠組を利用して資源の予約を行うという Resource Kernels[4] に基いて Linux にプロセッサ時間の資源予約機構を加えたものであり, 高精度のプロセッサ時間予約が可能である.

Linux/RK ではユーザ空間のプロセスに対して資源予約を行なうため, カーネル空間内のパケット処理にかかるプロセッサ時間には適用し得ない.

3 ファイアウォールによるサービス中継時の問題

3.1 ファイアウォールの概要

ファイアウォールには大きく分けて 2 つの種類がある. ネットワーク層でフィルタによってパケットを中継したりブロックしたりするパケットフィルタリング方式とアプリケーション層でプロクシアプリケーションによって中継をするプロクシ方式である.

前者のパケットフィルタリング方式は基本的にカーネル内でのフィルタ処理と中継処理のみであるので, プロクシ方式と比較して中継時間が短くて済むという利点がある. 他方, パケットフィルタリング方式で設定可能な規則は IP アドレスやポート番号などのネットワーク層での情報に基くためにサーバに対する不正なサービス要求などアプリケーション層での攻撃を防ぐことはできない.

後者のプロクシ方式ではアプリケーション層でサービス要求の検証を行ってから中継を行うため, アプリケーション層での攻撃へも対応できるという利点がある. しかし, ユーザ空間に位置するプロクシアプリケーションが中継を行うため, パケットフィルタリング方式と比較して中継時間が長くなってしまう.

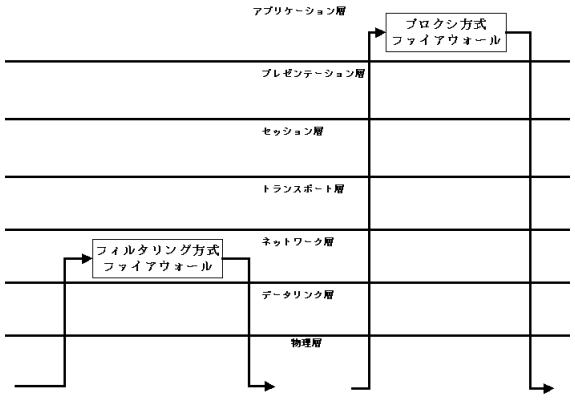

図 1: ファイアウォールの種類

商用のファイアウォールによってはパケットフィルタリング方式を改良し, ネットワーク層より上位の層をチェックするものもあるが, 基本はパケットフィルタリング方式である. ファイアウォールは一般的には土台となるオペレーティングシステム (以下, OS) の上に

実現されており, Linux や FreeBSD などの PC UNIX と呼ばれる UNIX 系 OS や WindowsNT が主に利用されている.

3.2 高負荷時のパケット処理の問題

ファイアウォールの土台に利用されている UNIX 系 OS では以下のような流れで到着したパケットの処理を行う.

パケットがネットワークインタフェースカード (以下, NIC) に到着すると NIC はハードウェア割り込みを発生させる (NIC によっては複数のパケットが到着してから発生させるものもある). ハードウェア割り込みが発生するとその NIC に対応したデバイスドライバの割り込み処理ハンドラが処理を行う. ここでは到着したパケットを取り込み, そのパケットが IP パケットであれば IP 用キューに繋いでからソフトウェア割り込みを発生させる. このソフトウェア割り込みにより, IP パケットのフラグメント解消などの IP 処理が行われてから TCP 処理あるいは UDP 処理が行われ, 最終的にソケットのキューに辿りつく. この時点でソケットのキューが埋まっているとパケットは廃棄される.

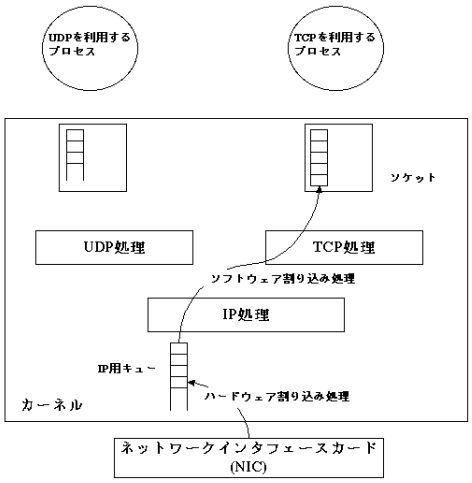

図 2: UNIX でのパケット処理

ソフトウェア割り込みは通常のプロセスよりは優先順位が高いのでパケットが到着するとソケットのキューに辿りつくまで, パケットの処理が行われる. 但し, あらたにハードウェア割り込みが発生した場合はハードウェア割り込みの処理が優先して行われる. もちろん, この間通常のプロセスは何もできない. つまり, パケットの数が一定数を越えると割り込み処理のみが行われてしまい通常プロセスは処理されなくなる. パケットフィルタリング方式のファイアウォールでもログの書き込みなどは通常プロセスにより処理されていることが多い. パケット数が増加するとこのようなログの書き込みなどが処理されなくなってしまうということになる.

さらにパケット数が増加するとハードウェア割り込みが多発し, ソフトウェア割り込みによるパケット処理も行われなくなってしまう. 通常のサーバ用途やクライエント用途であれば, このように大量のパケットが到着する可能性はまずありえないが, ルータ用途やファイアウォール用途に使う場合では以下のような様々な理由により大量のパケットが到着する.

1. サーバへのアクセスの集中
   ファイアウォールが中継を行うサーバに対してのクライエントからのサービス要求が増大することによって, ファイアウォールに到着するパケット数が増加する. 例えば, WWW サーバであれば多数の WWW クライエントから同時に接続要求がある. WWW サーバは TCP 接続を使っているためにパケットの紛失による再送がこれに拍車をかけることもある. また, ファイアウォールは通常複数のサーバのに対して中継を行なうので到着するパケット数はさらに増加する.

2. DoS 攻撃
   IP レベルまたはアプリケーションレベルでの DoS 攻撃によりファイアウォールに到着するパケット数が増加する. 例えば, SYN パケットを大量に送りつける DoS 攻撃である SYN flood が IP レベルでの DoS 攻撃の例として挙げられる. SYN flood によるメモリ浪費の問題は SYN クッキーなどの手法で対抗し得るが, パケットが大量に届くことを防ぐことはできない. また, アプリケーションレベルでの DoS 攻撃でもサーバに対して巨大なサービス要求や大量のサービス要求を送りつけたりするので, ファイアウォールに到着するパケット数が増加する.

パケット数の増加により割り込みが多発し, 処理が行われなくなる問題はパケットフィルタリング方式,

プロクシ方式いずれのファイアウォールでも一般的なOS を利用している限り起こる. 代表的な商用ファイアウォールでも一般的なOS あるいはその改良版を利用している場合は同様である.

3.3 資源浪費の問題

プロクシ方式のファイアウォールではプロクシアプリケーションがアプリケーション層での中継を行なう. プロクシアプリケーションではサービス要求の正当性検証などを行い中継するためアプリケーション層での攻撃を防ぐことができる. プロクシ方式ファイアウォールではアプリケーションごとに別個のプロクシアプリケーションを用意し, 個々に中継を行なう.

しかし, プロクシアプリケーションは処理が複雑であるためバグやDoS 攻撃などの原因によりプロセッサ時間やネットワーク帯域などの資源を浪費する場合がある. この場合, 資源を浪費するプロクシアプリケーション以外のプロクシアプリケーションが中継処理をするために必要な資源が極端に減少し, 中継が困難あるいは不可能となる.

すなわち, あるプロクシアプリケーションの異常動作により, そのファイアウォール上では全ての中継が行えなくなる. また, このファイアウォール上では全てのアプリケーションが利用不可能となるため, ログの取得など異常検出やIDS による管理者へのメール通知などの異常報告なども不可能となる.

4 ファイアウォールへの資源予約

4.1 資源予約機構と問題点

特定のプロセスによる資源浪費を防ぐ手法として資源予約機構が研究されている. プロセッサ時間, ネットワーク帯域, ディスク帯域などの資源の上限を予約する資源予約機構が研究されており, これらを統合して扱える統合的資源予約機構も研究されている.

このような資源予約機構をプロクシ方式のファイアウォールに対して適用すれば, あるプロクシアプリケーションがバグやDoS 攻撃などの原因により資源を浪費しようとしても資源の上限が予約されているため, 他のプロクシアプリケーションが中継するサービスには影響がない.

しかし, 資源予約機構にも以下のような問題点がある.

1. 処理できないパケットの増加
   アプリケーションにネットワークのパケットを処理させる場合, 処理の種類によるものの一定時間にアプリケーションが処理可能なパケットの数は上限がある. アプリケーションが処理しきれないパケットはソケットのキューに蓄積される. ソケットのキューは有限なので, この上限を越えて新たにソケットに到着したパケットは廃棄される. このアプリケーションに対してプロセッサ時間の資源予約を行なったりすると一定時間に処理できるパケット数は減少し, 処理できないパケット数は増加する.

2. パケット処理に対する資源予約が不可能
   大部分の資源予約機構はユーザ空間のプロセスやタスク, スレッドなどの資源割り当て単位に対して資源を予約するものであり, カーネル空間で処理される割り込み処理には適応されない. 通常, 割り込み処理により消費されたプロセッサ時間はパケットが到着する資源割り当て単位ではなく, 割り込まれた資源割り当て単位のプロセッサ時間を利用する. すなわち, 割り込み処理に費した時間は資源割り当て単位に正しく反映されず, 結果として資源予約が正常に働かない.

4.2 カーネル内処理の短縮

本論文では到着したパケットをユーザ空間で処理することで, パケット処理をも資源予約対象とすることを目指す.

パケット処理をすべてユーザ空間で処理させるためには割り込みをなくすのが理想であるが, 割り込みをなくすためにはNIC と OS の処理分担を変更する必要がある. 例えば, メモリアクセスとパケット処理の可能なNIC を用意してソケットのキューに繋ぐところまでNIC に処理させて, OS 側ではプロセスなどが定期的にソケットのキューにデータを取得するというような方法がある.

本論文ではPC ファイアウォールを前提としてこのような特殊なハードウェアを使うことを避けカーネル内でのパケット処理の時間を短くする方法で対応する. NIC へのパケット到着時のパケットの取り込みのみをカーネル内で処理し, その後のIP 処理やTCP やUDP などの上位層の処理をユーザ空間で処理する.

このようなユーザ空間におけるネットワーク処理

はマイクロカーネル構成の OS で実現されている．マイクロカーネルがハードウェア依存の NIC からのパケット取り込みを行い，その後の処理をユーザ空間で動作する OS サーバが処理する．OS サーバはパケットが IP パケットであれば IP パケットのフラグメント解消などの IP 処理を行い，その後 TCP 処理あるいは UDP 処理を行なう．モノリシックカーネル構成の OS でも NIC へ到着したパケットの取り込みを割り込み処理で行ない，以後の処理をユーザ空間のネットワーク処理プロセスに行わせることで実現できる．

以上のようにパケット処理の大部分をユーザ空間で行わせ，資源予約ドメインに含めることでパケット処理時間がより正しく反映できるようになる．

4.3 入力パケット数予約機構

資源予約による資源予約ドメインのプロセッサ時間の限定やネットワーク負荷の増加によりアプリケーションが一定時間に処理できるパケット数より多数のパケットがそのアプリーションのソケット到着した場合にはアプリケーションのソケットのキューに蓄積される．さらにこのソケットのキューの上限を越えて到着するパケットは廃棄されるので，廃棄されたパケットの処理に費やしたプロセッサ時間やメモリなどの資源が無駄となってしまう．

そこで，本論文ではネットワーク入力側での資源予約機構を検討する．ネットワーク出力側と同様にネットワーク帯域で資源の上限を決定する手法もあるが，ネットワーク帯域よりパケット数の方が消費されるプロセッサ時間を予測しやすいという利点がある．

本論文では予約ドメインに対してカーネルより送られるパケット数を予約できる入力パケット数予約機構を提案する．入力パケット数予約機構は NIC よりパケットを取り込んだ際に動作して，特定の条件に合ういパケットを選別して予約ドメインへ送り予約上限を越えたパケットは廃棄する．

この入力パケット数予約機構によりアプリケーションのソケットの段階で廃棄されるパケットをなくすことができる．

5 資源予約機構の実装

5.1 ファイアウォール方式とプラットフォームの選択

本論文ではファイアウォールの基本方式としてプロクシ方式を前提として実装を行なった．プロクシ方式はアプリケーションレベルでのサービス要求の正当性検証を行うので，パケットフィルタリング方式では防げないアプリケーションレベルでの攻撃を防ぐことができる．

プラットフォームとしてはプロセッサ時間予約などの資源予約機構が実装されているという条件以外にもパケット処理に利用されるプロセッサ時間を正しく資源予約対象に反映させる必要がある．本論文ではマイクロカーネル構成の OS である RT-Mach NR2 [10] を選択した．

RT-Mach NR2 ではユーザ空間で動作する FreeBSD 互換の OS サーバである Lites サーバ [11] が IP 処理や TCP や UDP などの上位層のパケット処理を行う．本論文ではプロクシアプリケーションと Lites サーバを合わせて 1 つの資源予約ドメインとして扱い，資源を予約するように各種資源予約機構を変更・実装した．

5.2 ALTQ のマイクロカーネルへの移植

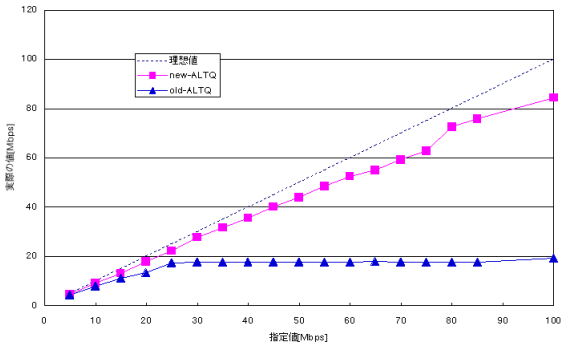

図 3: マイクロカーネルでの ALTQ の性能

RT-Mach NR2 ではネットワーク帯域を予約する資源予約機構がない．以前，Lites サーバへ FreeBSD でネットワーク帯域予約ができる ALTQ を移植したが，複数の Lites サーバが動作したときのネットワー

ク帯域予約をサポートしていない上に他のアプリケーションの影響を受ける Lites サーバでは精度の高い資源予約が行えなかった. そこで, 本論文では FreeBSD の ALTQ をマイクロカーネルに移植した.

ALTQ[7] は FreeBSD のパケット送出キューのスケジューリングアルゴリズムを実装できる汎用的な枠組を提供している. ALTQ では CBQ(Class Based Queueing)[8] などの代表的なキューイングアルゴリズムがあらかじめ実装してある.

CBQ は送出先アドレスと送出先ポート, プロトコル種別などでパケットをフィルタリングするフィルタを定義し, そのフィルタに対応するクラスに対して階層的に帯域を割り当てることでネットワーク帯域を制御できる.

本論文でも CBQ によるネットワーク帯域予約を利用する. オリジナルの CBQ では予約ドメインであるプロセス群とフィルタの対応付けがポートによるものでしか定義できない. 一方, RT-Mach ではタスク, スレッドに対して資源を予約する. 本論文では資源予約ドメインごとに別のデバイスポートを用意し, 個々のデバイスポートに対してネットワーク帯域予約が行えるように変更した.

また, この ALTQ の性能を図3に示す. new-ALTQ がマイクロカーネル上の ALTQ, old-ALTQ が Lites サーバ上の ALTQ の性能を示している.

5.3 入力パケット数予約機構の実装

入力パケット数予約機構では特定の条件に合うパケットを選別して予約ドメインへ送り, 予約上限を越えたパケットは廃棄する. パケットフィルタリング方式のファイアウォールが利用するパケットフィルタに似た機能であるが, パケット数の上限を予約できる.

RT-Mach ではパケットフィルタとして MPF[12] , BPF[13] などが実装されている. NIC から到着したパケットが条件に一致するパケットかどうかを判断する際にこれらのパケットフィルタを利用できるように入力パケット数予約機構を実装した. 資源予約ドメインはこの入力パケット数予約機構にパケットフィルタを登録し, 選別される.

Lites サーバは標準ではすべてのパケットを無制限に取り込む設定となっている. 入力パケット数予約機構を利用するためにはパケットの指定やパケット数の予約などより詳細な設定が必要となる. API としてはフィルタの設定と同じ device_set_filter を使うが, より

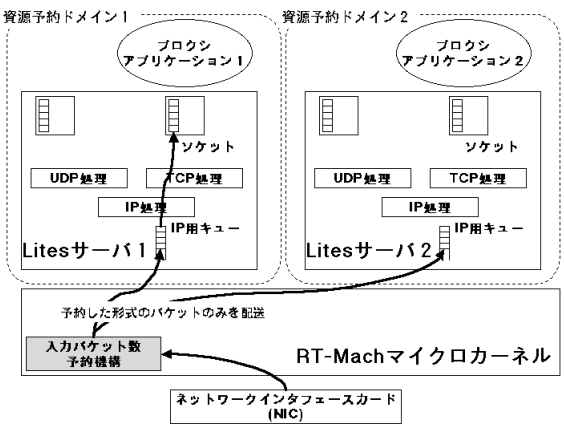

図 4: 入力パケット数予約機構の実装

簡易なコマンドラインインタフェースとして ipfw とほぼ同じ文法が使える mpf を実装した. この mpf では例えば

mpf fxp0 10 accept ip from any to any rate 1/10

のようにパケット数の予約を行う.

6 評価

6.1 評価環境

本論文ではファイアウォールに対して図5のような環境で資源予約を適用した場合の評価を行った. 評価環境ではファイアウォールを通じて WWW サーバへ HTTP リクエストを送ってから HTML を受け取るまでの応答時間を測定する.

1. 攻撃対象ホスト 1 台
   WWW サーバホスト 1 台
   - CPU Athlon 800Mhz
   - Memory 512Mbytes
   - FastEtherNIC(fxp)
   - OS : FreeBSD-4.3R

2. ファイアウォールホスト 1 台
   - CPU Celeron 533Mhz
   - Memory 512Mbytes
   - FastEtherNIC(fxp)
   - OS : FreeBSD-4.3R
     FreeBSD-2.2.8R

Linux-2.4.4
Solaris-7+FW-1
RT-Mach+Lites
(+資源予約機構)

3. RTT 測定用ホスト 1 台
   DoS 攻撃用ホスト 1 台
   ・CPU Athlon 950Mhz
   ・Memory 512Mbytes
   ・FastEtherNIC(fxp)
   ・OS : FreeBSD-4.3R

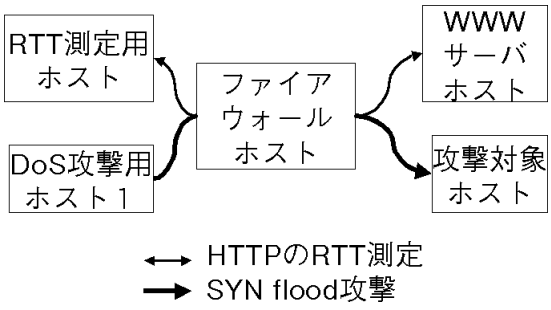

図 5: 資源予約効果の測定環境

このような環境においてファイアウォールを通じて DoS 攻撃用ホストから攻撃対象ホストに対して DoS 攻撃を行った場合の WWW サーバの HTTP 応答時間の変化を見る. なお, HTML 文書のサイズは 628[bytes] であり, 攻撃手法は Syn パケットを大量に送りつける SYN flood 攻撃である.

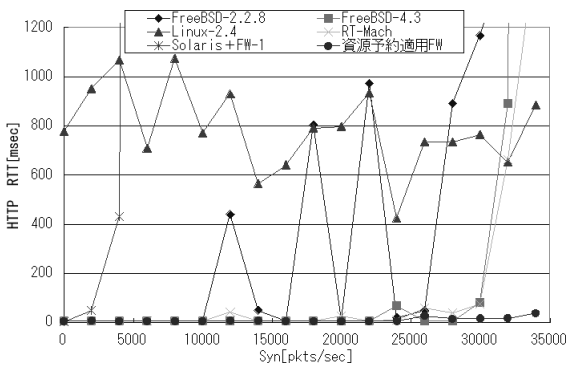

図 6: FW への資源予約効果

この結果を図 6に示す. 比較対象とした OS は FreeBSD-4.3R, FreeBSD-2.2.8R, Linux-2.4.4, RT-Mach NR2, さらに商用ファイアウォールと商用 OS である Solaris と FW-1, 今回実装した資源予約機構を適用した FW である.

資源予約はサービスごとに別個に計 3 つの Lites サーバを用意し, CPU, ネットワーク帯域を各 20%, BPF により SYN パケット入力を 1000[pakets/sec] に予約した. また, Solaris-7+FW-1 以外はサービスを中継するために simpleproxy[14] を用いている. 結果は 1 秒以上の間隔を空けて HTTP リクエストを 50 回行った際の応答時間の平均を取っている.

図より明らかなように Solaris-7+FW-1 が最も SYN flood に対する耐性が低く, 6000[packets/sec] でサービス不能状態に陥いる. これはパケット到着ごとに FW-1 が log を書き込むことが影響していると推測される.

FreeBSD-2.2.8R は 32000[packets/sec], FreeBSD-4.3R は 34000[packets/sec] でサービス不能状態に陥いる. 資源予約を行っていない RT-Mach NR2 も同様に 34000[packets/sec] でサービス不能となるが, マイクロカーネル内のネットワーク処理よりも複数の Lites サーバが同じパケットを処理する無駄が生じているためであると考えられる.

Linux-2.4.4 は FreeBSD に対して安定している. これはネットワークパケットの入力処理の割り込み処理が FreeBSD に比べて細分化され, 割り込みレベルの使い方も異なるために安定性が向上しているものと考えられる. 他方通常時でも応答速度が 800[msec] 前後であり他の OS の 5[msec] 前後と比較して明らかに遅い一因となっていると考えられる.

対して今回実装した資源予約を適用した FW では 34000[packets/sec] でも 10[msec] とほぼ影響がなく, かつ, Linux より 80 倍程度応答速度が早い. 入力パケット数予約を含めた資源予約がファイアウォール内でのサービスごとの資源の分離に有効に働いていると言える.

7 おわりに

本論文ではファイアウォールが DoS 攻撃に対してボトルネックとなり得るような状況に対し, ファイアウォールが中継するサービスごとに資源を分離, 予約することを提案した. また, 資源予約を適用したファイアウォールをマイクロカーネル構成の RT-Mach 上

に実装し, その評価を行った.

資源予約を適用したファイアウォールの有効性を評価するためにファイアウォールを通じて特定のホストを SYN flood 攻撃で攻撃した際の HTTP 応答時間を見ることでファイアウォールでの資源予約が有効であることの検証を行った. 検証の結果, FreeBSD や資源予約なしの RT-Mach が 34000[pakets/sec] 程度でサービス不能状態となるのに対し, 資源予約を適用したファイアウォールでは 10[msec] で応答を返すことができ資源予約によるサービスごとの資源分離が有効であることを実証できた.

今後の課題としては入力パケット数予約機構にキャッシュを適用してより高速化することなどが挙げられる.

参考文献

[1] G.Banga, P.Druschel, and J.Mogul, "Resource containers: A new facility for resource management in server systems.", In Proceedings of the USENIX 3rd Symposium on Operating System Design and Implementation New Orleans, LA, October 1999, Feburary 1999.

[2] G.Banga and P.Druschel, "Lazy receiver processing (LRP): a network subsystem architecture for server systems.", In Proceedings of the second USENIX symposium on Operating systems design and implementation,Seattle, WA, October 29 - November 1, 1996.

[3] S. Oikawa and R.Rajkumar, "Linux/RK: A Portable Resource Kernel in Linux.", 1998 RTSS Work-in-progress Session, Madrid, December 1998.

[4] R.Rajkumar, K.Juuva, A. Molano, and S. Oikawa, "Resource Kernels: A resource-centric approach to real-time systems.", In Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking, January 1998.

[5] Roger M.Needham, "Denial of Service.", In Proceedings of the 1st ACM Conference on Computer and Communication security, pp.151, 1993.

[6] "CERT/CC Denial of Service.", http://www.cert.org/tech_tips/denial_of_service.html

[7] K. Cho, "A framework for alternate queueing: Towards traffic management by pc-unix based routers.", In Proceedings of the USENIX 1998 Annual Technical Conference, New Orleans, Lousiana, June 1998.

[8] Floyd. S and Jacobson. V, "Link-Sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol.3, No.4(1995), pp.365-386.

[9] H. Tokuda, T. Nakajima and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System. ", In Proceedings of USENIX Mach Workshop, Otctober 1990. http://www.cs.cmu.edu/afs/cs/project/rtmach/public/papers/rtmach90.ps

[10] "Real-Time Mach NTT Release", http://info.isl.ntt.co.jp/rtmach/

[11] "UNIX under Mach The LITES Server", http://www.cs.hut.fi/ jvh/lites.MASTERS.ps

[12] M. Yahara, B. Bershad, C. Maeda, and E. Moss, "Efficient packet demultiplexing for multiple endpoints and large messages", Proceedings of the Winter 1994 USENIX Conference, 1994.

[13] S. McCanne and V. Jacobson, "The BSD packet filter: A new architecture for user-level packet capture.", USENIX Technical Conference Proceedings, pp.259-269, San Diego, CA, Winter 1993. USENIX.

[14] "simpleproxy" http://www.crocodile.org/software.html