並列IPルータCORErouterにおける並列パケットフィルタ機構の提案

5O－9

三栄武　小倉毅　丸山充　高橋直久
NTTソフトウェア研究所

1　はじめに

インターネットの普及にともない、パケット転送能力の高いIPパケットルータの実現が求められている。一方、外部からの侵入・攻撃から自組織のネットワークを保護するため、ルータのパケットフィルタ機能を用いたファイアウォールの構築が重要となっている。しかし、複雑な設定のパケットフィルタ処理は、ルータのパケット転送能力の低下を招いてしまう。

我々は、高性能並列IPルータCORErouter[2]の研究を進めており、本稿では、CORErouterのような機能分散型マルチプロセッサ構成のルータに適した並列フィルタ処理方式を提案する。本フィルタ処理方式では、ユーザが定義したフィルタを並列に解釈可能な記述に変換し、これを並列に処理することにより高速なフィルタリングを実現する。

2　従来のパケットフィルタリング

一般的なIPルータにおいてネットワーク管理者は、指定したパケットのルータ通過を許可または禁止するルールを複数記述して、パケットフィルタを設定する[1]。本稿では、このルールをアクセス制御規則と呼ぶ。アクセス制御規則はパケット条件とルータ動作の対で構成する。ルータのパケットフィルタ機構は、受信したIPパケットのヘッダ部とパケット条件を比較し、一致した場合にはルータ動作に従ってパケットを処理する。アクセス制御規則は複数行記述でき、一連のアクセス制御規則をアクセス制御プログラム(ACP)と呼ぶ。図1にアクセス制御プログラムの例を示す。従来のルータのフィルタ機構は、アクセス制御プログラムを上から順に解釈する。すなわち、i行目のアクセス制御規則のパケット条件を$P[i]$、ルータ動作を$A[i]$とすると、従来のフィルタ機構は次のように逐次的にアクセス制御プログラムを解釈する。

```
if P[1] then A[1]
  elsif P[2] then A[2]
      elsif P[3] then A[3]
          :
```

この時、パケット条件は、パケットヘッダとの比較パラメータとして、プロトコル種類、始点アドレス/ポート番号、終点アドレス/ポート番号およびTCPパケットの場合にはACKビット[3]の値からなる。これら比較パラメータを各々$p[i,1]$, …, $p[i,6]$として、i行目のアクセス制御規則のパケット条件$P[i]$を記述すると次のようになる。

$$P[i] = p[i,1] \cap p[i,2] \cap p[i,3] \cap p[i,4] \cap p[i,5] \cap p[i,6]$$

また、ルータ動作の値はpermitとdenyがあり、permitの場合にはルータはパケットを転送し、denyの場合にはパケットを廃棄する。

このように従来のフィルタ機構は、アクセス制御プログラムを逐次的に解釈するため、次のような問題が起こる。

問題1　アクセス制御プログラムの行数に比例してフィルタ処理時間が増加し、ルータのパケット転送能力が低下する。

問題2　パケット条件は比較パラメータのAND条件であるため、OR条件は複数行のアクセス制御規則を用いて記述する。これらのアクセス制御規則は、一部の比較パラメータのみが異なる記述となる。このようなアクセス制御規則を持つアクセス制御プログラムを、単純に逐次的に解釈すると、同じ比較演算を複数回行なってしまう。

問題3　パケット条件$P[i]$, $P[j]$, (ただし$i < j$)において、$P[j]$が$P[i]$に含まれる場合、実行不能経路(IFP: Infeasible Path[4])が存在することになる。この場合、ルータ動作$A[j]$は決して実行されないにも関わらず、フィルタ機構はパケット条件$P[j]$とパケットの比較を行なってしまう。

パケット条件						ルータ動作
プロトコル種類	始点アドレス	始点ポート	終点アドレス	終点ポート	ACKビット	
tcp	*	*	11.22.*.*	*	established	permit
tcp	*	*	11.22.*.*	53	*	permit
tcp	*	*	11.22.33.44	25	*	permit
tcp	*	*	11.22.55.66	119	*	permit
ip	*	*	*	*	*	deny

*はワイルドカード

図1: アクセス制御プログラム(ACP)の例

3　並列化パケットフィルタリング

前章で述べた問題点を解決するため、アクセス制御プログラムにおける依存関係を解析し並列化するACP並列化コンパイラと、その出力を解釈しフィルタ処理を行なう実行機構を提案する。アクセス制御プログラムのi行目のアクセス制御規則を、経路条件[4]$PC[i]$を用いて次のように記述すると、全てのアクセス制御規則を並列に実行させることが可能となる。

```
if PC[i] then A[i]
```

ただし、$PC[i] = \overline{P[1]} \cap \overline{P[2]} \cap \cdots \cap \overline{P[i-1]} \cap P[i]$

ACP並列化コンパイラと実行機構による並列パケットフィルタシステムの構成を図2に示す。コンパイラは、アクセス制御プログラムを読み込み、並列に解釈可能な中間コードを生成する。実行機構は、コンパイラが生成した中間コードを解釈し、受信パケットと比較を行なって、パケットの転送または廃棄を判断する。この時、実行機構は使用可能な計算資源の数により3種類の形態を採る。

3.1　ACP並列化コンパイラ

ルータ起動時にACP並列化コンパイラは、先に述べた経路条件による記述に従って、アクセス制御プログラムから中間コードを生成する。中間コードは、互いに依存関係を持たない複数行のアクセス制御規則からなり、各アクセス制御規則はパケット条件とルータ動作からなる。この時、単純に上記経路条件を求めて中間コードを生成すると、次のような問題が生じる。

- パケット条件が長くなる。また、パケット条件を短くするために、アクセス制御規則を複数行に分割すると中間コードの行数が大きくなる。
- 前章で述べた従来フィルタ機構の問題2、問題3を解決していない。

このためコンパイラは次のように中間コードを生成し、中間コードを最適化する。

A Design of Parallel Packet Filter Mechanism for *CORErouter*.
Takeshi MIEI, Tsuyoshi OGURA, Mitsuru MARUYAMA, Naohisa TAKAHASHI
NTT Software Laboratories
9-11, Midori-Cho 3-Chome Musashino-Shi, Tokyo 180 Japan.

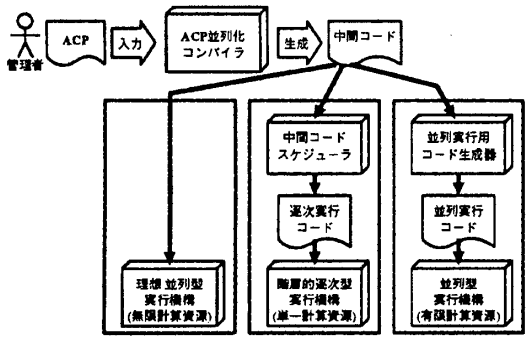

図 2: 並列パケットフィルタシステムの構成

- 中間コードのアクセス制御規則を、比較パラメータの数に従って分割する。アクセス制御プログラムの i 行目のアクセス制御規則を $P[i]$、$P[i]$ の比較パラメータを $p[i,j]$、$P[i]$ から生成する中間コードのアクセス制御規則の集合を $\{PC[i]\}$ とすると、コンパイラは次のように中間コードを生成する。

  if $\{PC[i]\}$ then $A[i]$
  ただし、
  $\{PC[i]\} = \{Q[i-1]\} \cap P[i]$,
  $\{Q[0]\} = \{\mathrm{TRUE}\}$
  $\{Q[k]\} = \{\{Q[k-1]\} \cap \overline{p[k,1]}$,
  $\{Q[k-1]\} \cap p[k,1] \cap \overline{p[k,2]}$,
  $\{Q[k-1]\} \cap p[k,1] \cap p[k,2] \cap \overline{p[k,3]}$,
  $\vdots$
  $\{Q[k-1]\} \cap p[k,1] \cap p[k,2] \cap \cdots \cap \overline{p[k,6]}\}$

- 中間コードのパケット条件に複数個の同一な比較パラメータあるいは包含関係を持つ比較パラメータを生成した場合、すなわち $\alpha = \beta$ あるいは $\alpha \subset \beta$ が成立する比較パラメータ $\alpha \cap \beta$ を生成した場合には、これを簡単化する。
  例 1: $tcp \cap tcp \Rightarrow tcp$,
  例 2: $11.22.*.* \cap 11.22.33.44 \Rightarrow 11.22.33.44$

- $\alpha \cap \beta \equiv \emptyset$ が成立する比較パラメータを生成した場合には、その比較パラメータを持つアクセス制御規則を中間コードから削除する。

図 1で例示したアクセス制御プログラムを入力した場合に、コンパイラが生成する中間コードを図 3に示す。

3.2 実行機構

実行機構は、ルータがパケット転送を行なう際に、中間コードを解釈してパケットの転送または廃棄を判断する。図 2に示したように、使用可能な CPU 資源の数によって、理想並列型実行機構、階層的逐次型実行機構、並列型実行機構の 3 種類を考える。ここでは、これら実行機構の概要を述べる。詳細は別の機会にて議論する。

理想並列型実行機構 無限個の CPU 資源が使用可能な場合の実行機構であり、ACP 並列化コンパイラから出力された中間言語をそのまま解釈する。中間コードの全てのアクセス制御規則に、それぞれ CPU を割り付け、並列に受信パケットと比較し、並列なフィルタ処理を行なう。

階層的逐次型実行機構 単一の CPU のみ使用可能な場合の実行機構である。中間コードは任意の順序で実行可能であるので、中間コードスケジューラは中間コードを解釈して、木構造となるように比較パラメータと受信パケットの比較順序を決定する。木構造において、ノードは一つの比較パラメータを表し、枝は比較パラメータと受信パケットとの比較結果 (等しい、大きい、小さい) に従って、次に比較すべきノードへのポインタである。葉はルータ動作を表す。この時、中間コードスケジューラは、各アクセス制御規則で共通する比較パラメータを抽出し、これらが木の同一パスとなるように構築することで、木の高さを小さく抑える。実行機構は、根のノードから順に、木構造に沿って受信パケットとの比較を行ない、到達した葉の値に従ってパケットの転送 / 廃棄を決定する。

パケット条件						ルータ動作
プロトコル種類	始点アドレス	始点ポート	終点アドレス	終点ポート	ACK ビット	
tcp	*	*	= 11.22.*.*	*	established	permit
tcp	*	*	= 11.22.*.*	= 53	NOT established	permit
tcp	*	*	= 11.22.33.44	= 25	NOT established	permit
tcp	*	*	= 11.22.55.66	= 119	NOT established	permit
tcp	*	*	≠ 11.22.*.*	*	established	deny
tcp	*	*	= 11.22.33.44	≠ 25 ≠ 53	NOT established	deny
tcp	*	*	= 11.22.55.66	≠ 53 ≠ 119	NOT established	deny
tcp	*	*	= 11.22.*.* ≠ 11.22.33.44 ≠ 11.22.55.66	≠ 53	NOT established	deny
tcp	*	*	≠ 11.22.*.*	*	NOT established	deny
≠ tcp	*	*	*	*	*	deny

図 3: 中間コードの例

並列型実行機構 有限個の CPU 資源が使用可能な場合の実行機構である。実行に先立ち、並列実行用コード生成器で、中間コードの比較パラメータの共通部分を抽出し、受信パケットとの比較結果を各 CPU が共有できるように並列実行コードを生成する。実行機構は、各 CPU が未実行の並列実行コードを動的に選択し、そのパケット条件を受信パケットと比較することにより、並列にフィルタ処理を行なう。

4 おわりに

アクセス制御プログラムの並列化コンパイラと、その実行機構を提案した。現在、コンパイラの作成が終了し、実行機構シミュレータの実現を進めている。今後は、シミュレータを用いて、従来のフィルタ機構と並列フィルタ機構の定量的評価を行なう予定である。

謝辞 本研究を御支援下さる梅本栄治 広域コンピューティング研究部長、ならびに日頃御討論いただく超並列プログラミング研究グループの皆様に感謝いたします。

参 考 文 献

1) W. R. Cheswick and S.M.Bellovin, "Firewalls and Internet Security", ADDISON-WESLEY PUBLISHING COMPANY.
2) 高橋, 丸山, 三栄, 小倉, "柔軟でスケーラブルな高性能ルータ CORErouter の基本構想", 第 52 回情処学会全国大会, pp. 1-207, 1996.
3) D. E. Comer, "Internetworking With TCP/IP, Vol I", 1991, Prentice Hall.
4) 直井, 高橋, "経路依存フローフラグを用いた Infeasible Path 検出法", 電子情報通信学会論文誌, D-I, Vol. J76-D-I, No. 8, pp. 429-439, 1993.