分散システム／インターネット運用技術 16－7
（1999. 11. 26）

パケットフィルタリングの記述法について

鶴 正人[1*]，黒田 英夫[2†]
[1]長崎大学総合情報処理センター
[2]長崎大学工学部情報システム工学科

あらまし：外部ネットワークからの侵入／攻撃を防いだり，利用可能なサービスを制限したりする方法として重要なパケットフィルタリングの設計／記述／検証の問題の計算機支援のために，形式化したモデル／記法を与え，論理式としての宣言的仕様から規則の並びとしての実際のパケットフィルタ記述を計算によって導く目処を述べた．

General packet filtering descripition

Masato TSURU[1], Hideo KURODA[2]
[1] Science Information Center, Nagasaki university
[2] Dept. of Electrical Engineering & Computer Science, Nagasaki universit y

Abstract: Packet filtering is essential as a tool for keeping network security and controlling service accesses. In this paper, we try to apply formal methods to specification and verification of packet filtering designing aided by computers. we discuss how to describe it as a logical expression and derive a sequence of filtering rules from it.

1 はじめに

外部ネットワークからの侵入／攻撃を防いだり，利用可能なサービスを制限したりする方法として，ルータやファイアウォール専用機でのパケットフィルタリングは最も基本的であり，数多く使われている．しかし，現状の設計／記述／検証方法は，

- プログラムでいえばアセンブラレベルの低水準なもので，生産性／検証性が低い．
- ネットワーク (プロトコル等) に関する高度な知識が必要である．
- 機種毎に異なる文法と効果的な書き方の習得を必要としており，サポートツールも少ない．
- 複数インタフェースや複数ルータの連携を陽には記述できない．

など，非常に素朴な段階に止まっている．この中には，パケットフィルタリングの重要性が認識されはじめた初期の頃に既に指摘されていたもの [1] も含まれているが，7年経った今でもあまり改善されていない．

筆者は，ネットワーク全体の抽象度の高いサービス利用の許可／不許可の方針（仕様）から，多段階で具体化し，最終的に，各種ルータに固有のパケットフィルタ言語による実行効率のよい記述を生成できるような仕組みを検討している．すなわち，上位の仕様から下位に固有の知識を使って下位の仕様 (記述) を導出したり，与えられた下位の仕様 (記述) が上位の仕様を満しているかを検証したりすることを，できるだけ形式的／自動的に行えるようにするのが目標である．

1. 抽象サービスレベル — 各サービスを実現するネットワークプロトコルの詳細に独立な記述．許可 (サービスの利用／提供) と，不許可 (サービスの禁止／安全の確保) の両面から表現する．ただし，ネットワーク間にパケットフィルタ（フィルタノード）を置く，という構造は前提とするので，ネットワーク構成の抽象度に応じて変化し，このレベルの中での仕様の詳細化も重要である．

*tsuru@net.nagasaki-u.ac.jp
†kuroda@cis.nagasaki-u.ac.jp

抽象サービスレベル仕様 1
↓↑ ← ネットワーク構成／経路
抽象サービスレベル仕様 M
↓↑ ← ネットワークプロトコル
パケットフィルタ共通モデルでの宣言的記述
⇕ ← 基本対応
一般パケットフィルタ言語記述 1
↓↑ ← 等価（最適化）変換
一般パケットフィルタ言語記述 N
⇕ ← 単純翻訳
実パケットフィルタ記述

表 1: 全体の構想

2. パケットフィルタ共通モデルレベル — 通過しようとするパケット内の各フィールドの値によって通過／遮断を決めるというモデルの下で，ネットワークプロトコルの詳細に依存した記述．宣言的記述（1つの論理式）と，操作的記述（適用規則の列）とがあり，後者を GPFL（一般パケットフィルタ言語：General Packet Filter Language）と呼ぶことにする．GPFL の等価変換により，パケットフィルタとしての意味は同じだが実行効率などの低レベルの効果の異なる操作的記述を得ることができる．
3. 製品固有の実装レベル — 実際の各製品のパケットフィルタ記述．細かい点は各社違うが，どれも基本的には，2. のモデルと対応が取れ，その差異は，GPFL 上の制約として捉えられる．

本報告では，主に，2. と 3. について検討した．

なお，2. から 3. を生成する試みに近いものに，FLC(Filter language compiler)[†1] があるが，これは，2. を人手で直接記述することを想定して，IPF(4章参照）の機能をベースに，C 言語風の操作的記述を定義し，プリプロセッサを使って，ほぼ単純な対応で ipf.conf 記述や CISCO の access-list 記述に変換するだけである．

1. から 3. の生成を目指していると思われるものに，CISCO [†2] の ConfigMaker がある．これはパケットフィルタリングだけでなく，ネットワーク中の各デバイスのコンフィグレーション全体を，相互の整合もチェックしながら対話的に生成しようとするもので，枠組みとしては期待したいが，現時点での完成度は，少なくともパケットフィルタリングに関しては単なる GUI 化の域を出ていない．

[†1] http://coombs.anu.edu.au/~avalon/flc.html
[†2] http://www.cisco.com/

2 抽象サービスレベル

最も上位の仕様／ポリシーは，おそらく，モデルも手段も特定しないあいまいさのあるものであり，形式化は難しい．ここでは，既にパケットフィルタ (ネットワークの境界における通信の制限) を前提としたモデルにまでブレークダウンされた段階での仕様記述を考える．図 1 のようにネットワークの間にフィルタノード F が置かれ，各 F に，(部分) ネットワーク間でのサービスの利用／提供とその禁止の仕様を与える．

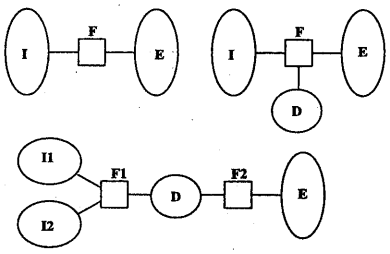

図 1: ネットワークとフィルタノード

このような人間寄りの記述形式は，“人間に取って直感的にわかりやすく，誤解しにくいこと” と “複雑な場合もそれなりに簡潔に記述できること” とが必要であり，この 2 つはしばしば両立しない．ここでは，仮に，{ サービス，利用ホスト集合，フィルタノード，提供ホスト集合，must/may，permit/deny } のような記述を 1 個の条件とし，その並びを仕様と考える．これは，

> “フィルタノード must/may permit/deny that a host in 利用ホスト集合 uses the サービス provided by a host in 提供ホスト集合”

という意味である．permit は許可，deny は不許可である．must と may はその条件が他の条件より優先するかどうかを示すもので，must-1 と must-0 は同じ強さ (よって衝突したら矛盾)，以下，may-0, may-1 の順に弱くなるものとする．一般には，多段階の may が必要になるかも知れない．

なお，実用化するには，パケットフィルタだけを独立して形式化しても効果は薄く，ネットワーク構成や経路なども含めて考える必要があり [2]，今後

の進展が要請されている．例えば，経路制御仕様に関しては，記述の標準化／形式化が始っている [3].

3 パケットフィルタ共通モデルレベル

ネットワークプロトコルは IP の世界とし，パケットフィルタの共通モデルとして以下のようなものを想定する．パケットフィルタは，各フィルタノードの各ネットワークインタフェースに置かれ，片方向 (IN または OUT) のパケットの通過に対して，そのパケットの，始点 IP アドレス，終点 IP アドレス，プロトコル (TCP/UDP/ICMP/IGMP など)，始点サービスポート，終点サービスポート，established かどうか (一般には TCP のフラグ) などを調べ，通過の許可/不許可を決定する．

実際のパケットフィルタは，通過の許可/不許可だけでなく，ログを取るかどうか，不許可の時の反応 (無視，ICMP xxx を返す，TCP の場合に RST を返す) などの動作も指定できるが，簡単のために，考えないこととする．

ここで，2章の抽象サービスレベルの仕様を，サービスを実現するネットワークプロトコルと，パケットフィルタの配置や機能に関する知識に基づいて，このモデルのレベルの記述にブレークダウンする必要がある．そのために，各仕様毎に，それを満すパケットフィルタに関する最も弱い十分条件を導出する．例えば，メールサービスの利用を必要とする場合，DNS 参照や IDENT の許可も合わせて導出するかも知れない．must permit 仕様から導出される条件（群）を must-1 条件と呼ぶ．must-0, may-1, may-0 も同様である．

このような導出は，その組織のネットワークの専門家が，サービスやトポロジ毎の知識 (導出規則) を用意しておくことで，ある程度自動化できると思われる．ただし，導出した条件の全体が矛盾する (充足可能でない) 場合，上位の抽象サービスレベルに戻って再検討が必要であり，この2つのレベルの間はいろいろな形式化できない条件も勘案しながら行き来することになる．

一方，IP spoofing(一般に偽装／矛盾アドレス) やソースルーティングを防いだりするのは高位レベルのポリシではなく，ネットワークの仕組に依存した技術的要件であるので，このレベルで初めて陽に現れ，自動的に付加されてよい．

例えば，図1の最も単純な左上の図において，サービス A，B とホスト h があるとして，

	始 H	始 P	終 H	終 P	フラグ	評価
OUT フィルタ						
1	I	any	E	any	any	may-1
2	I	any	E	A	any	must-1
3	h	A	E	any	estd	must-1
4	E	any	any	any	any	must-0
5	any	any	I	any	any	must-0
IN フィルタ						
6	E	any	I	any	estd	may-1
7	E	any	I	A	estd	must-1
8	E	any	h	A	any	must-1
9	E	any	h	B	not estd	must-0
10	E	any	I \ h	any	not estd	must-0
11	I	any	any	any	any	must-0
12	any	any	E	any	any	must-0

表 2: パケットフィルタの例

{any, I, F, E, may, permit}
{A, I, F, E, must, permit}
{A, E, F, h, must, permit}
{B, E, F, h, must, deny}
{any, E, F, I \ h, must, deny}

という仕様を，F の E 側の IN フィルタと，F の E 側の OUT フィルタで実現する場合，表2のようになる (ただし TCP 以外のことは省略).

以下に示すように各パケットフィルタの意味 (効果) は，通過許可の時は 1 を，不許可の時は 0 を出力するような論理関数で表現するので，始点アドレスから終点アドレスへの向きの通信経路中に，複数のパケットフィルタが存在する場合，その全体の意味は，論理関数の積になる．このようにすると，フィルタの合成／分解や，包含関係／等価性の検証などが，すべて論理関数の計算に帰着される．一般には論理関数の計算は変数の指数時間が必要であるが，BDD などの効率的な計算手法やそのソフトが実用化されており，成功している分野も多い [4].

なお，最近のパケットフィルタでは動的／ステートフルなフィルタも定義可能になっている．例えば，CISCO でいえば，IOS 11.1 で導入された dynamic access-list (ダイアルアップ接続などの動的に変わる IP アドレスからのアクセスを，ユーザ認証後，一定時間だけ許す) や，IOS 11.3 で導入された reflective access-list (UPD のようにコネクションのないやりとりも，始点終点の IP アドレス／ポート番号の組みと時間経過で関連を予想してアクセス制御する) などである．これらに対応するようなモデルの拡張も必要である．

3.1 宣言的記述

パケットフィルタを論理関数(あるいはそれを特性関数とする集合)で表現するために，判定に使うパケット内の各フィールドの値を2進コーディングする．例えば，IPアドレスは，32ビット整数として各々32個の論理変数で表現する．本来サービスポートはシンボリックなもので番号割当て自体に意味はないはずだが，実際は，1023以下の番号に意味を持たせたりするので，最も一般には16ビット自然数として各々16個の論理変数で表現する．

表2の例の“INフィルタ”で考えてみる．アドレスは2ビット ($I=\{0,1\}$, $E=\{2,3\}$，ホスト $h=1$)，ポートは1ビット ($A=1$, $B=0$) として，始点／終点アドレスを各々 (x,y), (p,q)，始点／終点ポートを各々s, d, establised かそうでないかを e という論理変数で表す．

must-1	7,9	$f=$	$x\overline{p}se+x\overline{p}qd$
may-1	6	$g=$	$x\overline{p}e$
must-0	8,10,11,12	$h=$	$x\overline{pqe}+x\overline{p}q\overline{de}$ $\overline{x}+p$

一般に，must-1，may-1，must-0，may-0条件を表す論理式を，各々，f,g,h,i とすると，そのパケットフィルタの全体仕様は，$f+\overline{hi}g$ と表現され，それが矛盾していないことのチェックは，$\overline{f}\supset h$ となる．

3.2 操作的記述 (GPFL)

一般パケットフィルタ言語(GPFL)の構文は以下のように定義する．eは積和論理式で，Pを Permit 規則，Dを Deny 規則と呼ぶ．

フィルタ定義 ::= 規則列; デフォルト動作
規則列 ::= 規則 | 規則; 規則列
デフォルト動作 ::= 1 | 0
規則 ::= $P(e)$ | $D(e)$

これの直感的な動作(非形式的な意味)は，“通過しようとするパケットに対して，規則列を前から順に見て，最初に当てはまった規則が P であれば通過許可，D であれば不許可にする”，というものである．

ここで，宣言的意味を与える．規則の意味(以下，⟨規則⟩と書く)を，論理式を部分的に変換して別の論理式を作る，論理式上の関数とし，規則列をその関数の合成だと考える．

$$\begin{aligned}\langle P(e)\rangle(f) &= f+e\\ \langle D(e)\rangle(f) &= f\overline{e}\\ \langle \text{規則;規則列}\rangle &= \langle \text{規則}\rangle\circ\langle \text{規則列}\rangle\end{aligned}$$

そして，X を P または D,e_iを論理式,e_0をデフォルト動作とすると，$X(e_1);X(e_2);....;X(e_n);e_0$ というフィルタ定義全体の意味は，$\langle X(e_1)\rangle\circ\langle X(e_2)\rangle....\circ\langle X(e_2)\rangle(e_0)$ を計算した結果の一つの論理式になる．

ここで，$\langle X(e_1+e_2)\rangle=\langle X(e_1);X(e_2)\rangle$ なので，隣合うP規則，D規則同志をまとめて，PとDが交互に現れる形にできる．また，隣合うPD/DPは，

$$\langle P(e_1);D(e_2)\rangle=\langle P(e_1);D(e_2\overline{e_1})\rangle=\langle D(e_2\overline{e_1});P(e_1)\rangle$$

という等価変形規則(定義に戻ればブール代数の分配法則)を使って，順序を入れ替えることができる．

3.3 宣言的記述から GPFL への変換

以下では，デフォルト動作を0としておく．宣言的記述は，一般に $f+\overline{hi}g$ という形であったので，そのまま GPFL に変換すると，

$$P(f);D(h);D(i);P(g);0$$

となる．

ここから出発して，等価変換を利用して，実際のパケットフィルタにマッピングした時の実行効率の向上や，それらのフィルタ毎の固有の制約の反映を行うことができる．これは宣言的な仕様から性質のよい操作的な命令列を導出することの一例と捉えることができる[†3]

表2の例の“INフィルタ”で考えてみる．出発は，

$$P(x\overline{p}se+x\overline{p}qd);D(x\overline{pqe}+x\overline{p}q\overline{de}\overline{x}+p);P(x\overline{p}e);0$$

ここから，例えば，論理式におけるリテラルの出現(参照)回数が少ない表現を求めて，必要ならば順番を入れ替えながら積項をまとめていくと，以下のようになった．

$$P(x\overline{p}e);D(\overline{x});D(\overline{p});P(qd);0$$

4 製品固有の実装レベル

3章の GPFL 記述から，実際の各種ルータのパケットフィルタ記述への変換を考える．指定可能な“条件”の細かさは，製品毎に多少異なる．

[†3] 何が“実行効率のよい規則列”が問題になるが,,,.

CISCO IOS

IOS の extended access-list は代表的なパケットフィルタ記述であり，GPFL のモデルもこれである．例えば，

permit	条件 e_1
permit	条件 e_2
deny	条件 e_3
deny	条件 e_4
permit	条件 e_5

の動作は，単純に以下のようになる．

$$P(e_1); P(e_2); D(e_3); D(e_4); P(e_5); 0$$

富士通 LR(Link Relay)

LR では，例えば，

deny リスト	: (条件 e_1, 条件 e_2)
permit リスト	: (条件 e_3, 条件 e_4, 条件 e_5)

のようにパケットフィルタを記述し (実際はメニュー形式で対話的に行う)，その直感的な動作は，“通過しようとするパケットに対して，まず deny 規則の中に当てはまるものがあるかを調べ，なければ通過許可，一つでも当てはまれば，次に permit 規則の中に当てはまるものがあるかを調べ，一つでもあれば通過許可，そうでなければ，不許可”，というものである．動作は以下のようになる．

$$P(e_3); P(e_4); P(e_5); D(e_1); D(e_2); 1$$

つまり，GPFL では，Permit 規則は全部左側にまとめ，Deny 規則は全部右側にまとめる必要がある．

IPF(IP Filter)

UNIX 上のフリーソフトであり，そのパケットフィルタ記述はここで上げた 3 つの中では最も細かく，記述の自由度が高いが，これは人間が直接記述するときの書きやすさや見やすさのための機能や，あるいは実行性能を意識して改善できるようにするための機能などがあるが，本質的には GPFL の範囲内である．例えば，

block quick	条件 e_1
block	条件 e_2
pass	条件 e_3
pass quick	条件 e_4
block	条件 e_5

の動作は，“通過しようとするパケットに対して，規則列を前から順に見て，ある規則に当てはまったら，それが block ならそこで block 印がつけられ，pass なら pass 印がつけられ，さらに次の規則へ進む．ただし，quick 付きの規則に当てはまったら，次の規則へは進まず，そこの規則が pass なら 通過許可，block なら不許可になる．途中で quick 付き規則にマッチせずに全規則のマッチングが終ったら，その時点で，pass 印または無印なら 通過許可，block 印なら不許可になる”，というものである．

最終的な許可／不許可だけを考えると，quick 付き規則は，すべて先頭に集めることができ，quick 付き規則の並びは後ろから，他の規則の並びは前から，関数として合成すればよい．すなわち，

$$D(e_1); P(e_4); D(e_5); P(e_3); D(e_2); 1$$

と対応する．

5 まとめ

パケットフィルタの設計／記述／検証の問題の計算機支援のために，形式化したモデル／記法を与え，論理式としての宣言的仕様から規則の並びとしての実際のパケットフィルタ記述を計算によって導く目処を述べた．

現時点では机上で行ったが，今後これをプログラムとして実装できるよう，効率のよいデータの形式やアルゴリズムを調査し，実用性を検証したい．また，細かいバリエーションや新しい動的／ステートフルなフィルタモデルへの対応も検討する必要がある．

一方，実用化のためには，上流工程の具体的な形式化の目処を立てることが必要である．

参考文献

[1] D. B. Chapman. Network (in)security through ip packet filtering. *the Third USENIX UNIX Security sympo.*, 1992.

[2] 鶴正人, 黒田英夫. Ip ネットワークの経路設計と検証の手法について. 情報処理学会研究報告 DSM 研究会, Vol. 8, pp. 19–24, 1997.

[3] C. Alaettinoglu, T. Bates, and et. al. Routing policy specification language. *RFC 2280*, 1998.

[4] 湊真一. 計算機上での bdd の処理技法. 情報処理, Vol. 34, No. 5, pp. 593–599, 1993.