ハードウェア処理されるAccess Control Listの短縮化

石川拓道[†1] 吉浦紀晃[†1]

ネットワークにおけるアクセスコントロールリスト (Access Control List,ACL) はファイアウォールの機能の1つでネットワークセキュリティにおいて重要な役割を果たすものである．ACLの処理には一律でかつ短い時間で処理が可能なハードウェアが使われることが多いが，ACLは物理的制限のあるハードウェアに適したサイズでなければならない．そこで本論文では与えられたACLの記述を短縮し，ハードウェアのサイズに当てはまるようにするプログラムを開発する．そして開発したプログラムを実際のACLに使用することでそのプログラムの性能を見る．プログラムを実装して処理を行った結果，ACLを短縮することができた．

Reduction of Access Control List for Hardware Processing

TAKUMICHI ISHIKAWA[†1] and NORIAKI YOSHIURA[†1]

Access control list (ACL) is one of the most important things in computer network security. While an inexpensive router or PC processes ACL by software, network equipment such as Layer 2 or 3 switch processes ACL by hardware because there is a speed limit in software process ability. The hardware process of ACL can handle high speed network packet, however, this capability limits ACL configuration such as the limit of the number of rules in ACL. This paper proposes the software that decreases the number of rules in ACL to satisfy the limit of hardware. This paper also evaluates this software by experiment in which this software is applied to practical ACL.

1. はじめに

コンピュータネットワーク間の通信制御を行うシステムであるファイアウォールはネットワークセキュリティにおいて重要な位置を占める．そのファイアウォールの通信制御は**アクセスコントロールリスト (ACL)** を設定することで行われる．

ネットワークにおけるACLとは，発信源から目的地への通信に対し許可属性を付加した記述であり，通信時にそれを参照することで許可するか拒否するかの制御が行われる．ACLをソフトウェアで処理する場合は，参照されるべき記述をリストの上から順番に探す直列処理で行われる[1),2)]．基本的に読み込むことのできるACLのサイズに制限はないが，処理に要する時間はACLのサイズ等により変化し，その処理に最適なACLを構築することはNP完全であることが知られている[3)]．一方，ACLをハードウェアで処理する場合は，連想メモリ (Content Addressable Memory,CAM) などを用いるために記述を参照する際，並列処理が可能で一律の時間により実行できる[4),5)]．ただし，ハードウェアのサイズに収まるような大きさのACLでなければ実行できない．また，大きいACLでは，CAMでの消費電力が電力コストが大きくなってしまう．よって，ACLはより小さいサイズのものが好ましい．

本論文で扱うLayer 3スイッチであるアルカテル社のOmniSwitchでは，利用できるACLのサイズがマニュアルに記載されており，ACL関連以外の利用できる限界値，例えば，VLAN数，dynamic VLANやユーザ認証によるVLANの利用者数の上限なども記載されている．しかしながら，これらの機能を同時に上限まで利用できるとは限らず，ACL関連以外の機能を最大限利用したときに，ACLの機能が上限まで利用できない可能性がある[*1]．よって，ACLのサイズを出来るだけ小さくした方が管理運用上も好ましい．

一方，ネットワークの管理運用の立場から考えると，何らかのトラブルやセキュリティ上の問題への対応，ネットワークの新設や廃止など様々なオペレーションがACLの更新を必要とすることもある．そのときに，

†1 埼玉大学大学院理工学研究科数理電子情報部門情報領域
Department of Information and Computer Science, Saitam University

*1 OmniSwitchのログやマニュアル等を見るとOmniSwitchでのACLの処理は，CAMで行われているわけではなく，pseudo CAM (PCAM)，つまり擬似的なCAMが利用されているようである．よって，他の機能にPCAMが多く利用されるとACLの処理に必要なPCAMが足りなくなる可能性があるのではないかと思われる．

ネットワーク管理者がサイズを考慮して ACL の更新を行うことは難しい．よって，管理者は自身が記述したい ACL をそのまま記述し，ACL のサイズを自動的に短縮することができれば，ネットワークの管理コストを下げることができる．

以上のことから，本論文では ACL に記述された許可属性の意味は損なわずに ACL を書き換え，そのサイズを特定のハードウェアのサイズまで短縮するようなプログラムの開発を行うことを目的とする．ただし，ACL の最適化が NP 完全であるため，実際に最適な大きさまでプログラムやアルゴリズムを考案しても，それが実用的かどうかは分からない．しかし、ACL がハードウェア処理される場合には，制限内に ACL を収めることで一定時間での処理が保証される．また，最適化よりも制限内に収めることは問題として簡単になるのではないかと思われる．そこで，ACL を局所的に見て，短縮する規則を導入し，それを実装したプログラムを作成する．このようなプログラムがあれば，ACL はハードウェアのサイズにとらわれず構築することが可能で，ハードウェア処理におけるデメリットを克服することができる．

この目的を達成するため以下 3 つのことを研究目標として定める．

(1) ACL を短縮するようなアルゴリズムの考案
(2) 考案した ACL の短縮アルゴリズムの実装
(3) 実装プログラムのテストとその評価

次章では本論文で扱う ACL の詳細な説明を行う．第 3 章では ACL を短縮するアルゴリズムを考案し，第 4 章では開発したプログラムによる実験について述べる．それを受けて第 5 章では実験結果に対する考察を，第 6 章では本論文のまとめと今後の課題について述べる．

2. Access Control List (ACL)

2.1 ACL とは

ACL には，基本的に発信源 (source) から目的地 (destination) までの通信を許可するか否かが記されている．通信は IP アドレスや MAC アドレス，使用するポート番号やプロトコルなどの特徴で表現されている．また，許可属性と通信の性質以外にはその記述の優先度が設定される．類似する性質を持つ複数の通信にはグループを作成し，そのグループを用いて通信路を表現することができる．例えば IP アドレスならネットワークグループ，プロトコルならサービスグループと呼ばれるグループを作成し，そのグループから他のまたは自身への通信を許可または拒否する，といった記述が可能である．なお本論文で取り扱うアルカテル社の OmniSwitch では，source/destination は同じ性質でなければいけない．すなわち 1 つの記述には 1 種類の性質しか許されないことになっており，グループを扱った場合でも同様である．例えば source に IP アドレスが設定されている場合，destination に設定できるのは IP アドレスのみで MAC アドレスなどは設定してはいけない．source にネットワークグループが設定されている場合，destination に設定できるのはネットワークグループのみでサービスグループや IP アドレスなどは設定してはいけない．

2.2 ACL のハードウェア処理

ACL をハードウェアで処理するには，ハードウェアである連想メモリ (CAM) が用いられる．複数の連想メモリのキーと入力パケットの情報との一致を並列に検索し，連想メモリの値にはその入力に対する出力が与えられる．なお，source/destination の情報を格納する部分やグループを設定する部分などは CAM の中で分れている．例えば，入力のパケットが IP アドレス A から IP アドレス B へ送信された場合，IP アドレス A や IP アドレス B がどのネットワークグループに属するかを調べる部分、あるネットワークグループに属していた場合は，今度はそのグループに対する条件やその処理方法を調べる部分はそれぞれハードウェア上で独立しており、並列に実行される．

2.3 OmniSwitch における ACL の特徴

本論文は実際に埼玉大学で用いられている ACL を元に行うため，その文法はアルカテル-ルーセント社の Layer3 スイッチである OmniSwitch のものに則る．OmniSwitch の ACL の例を**表 1** に示す．

表 1 のように，OmniSwitch では，アクセス制御を行いたいネットワーク，アクセス制御の適用条件，通過や遮断などのアクセス制御の実際の動作をそれぞれ，network group，condition，action としてそれぞれ設定し，これらを利用することでアクセス制御の規則を定義する．OmniSwitch の場合，rule と設定することにより実施でき，これらの設定には，policy と呼ばれる命令が利用される．policy によりつけた名称は他の規則により参照されるが，その際，policy の種類と名称を明示しなければならない．例えば表 1 では IPaddressA/mask，IPaddressB/mask を含むネットワークグループ N1 から IPaddressC/mask，IPaddressD/mask を含むネットワークグループ N2 への通信する状態 C1 に対し，優先度 100 で制御 A1 を実行することを R1 で設定している．policy による各設定についての詳細を以下に解説する．

- **service**
  service ではプロトコルの設定を行う．TCP や UDP などプロトコルの種類と用いるポート番号を記述する．service は condition で主に参照される．
- **group**
  group では各グループを構成する．ここの policy の種類は以下の通り．
  - policy network group

表 1 OmniSwitch の Access Control List の例
Table 1 An example of Access Control List of OmniSwitch

```
policy network group N1 IPaddressA/mask IPaddressB/mask
policy network group N2 IPaddressC/mask IPaddressD/mask
policy network group N3 IPaddressE/mask IPaddressF/mask
policy network group N4 IPaddressG/mask IPaddressH/mask IPaddressI/mask
policy condition C1 source network group N1 destination network group N2
policy condition C2 source network group N3 destination network group N4
policy action A1 disposition accept
policy action A2 disposition drop
policy rule R1 precedence 100 condition C1 action A1
policy rule R2 precedence 10 condition C2 action A2
```

- policy mac group
- policy service group
- policy port group

network group では IP アドレス，mac group では MAC アドレス，service group では service，port group では使用するポートをグループ化する．service group を構成するときには policy service で設定した service の名称で参照する．group は condition で主に参照される．

- **condition**
  condition では source/destination の設定を行う．また service もここで設定される．ここで扱われるキーワードの種類は数多くあるが，本論文では，network group を短縮化の対象とする．condition では，source/destination，またはいずれか一方を設定する．このとき上記のキーワード，設定内容の順に policy を構成する．group を用いる場合にはその group を作成したときに設定した group の名称で参照する．condition は rule で主に参照される．
- **action**
  action では通信に対する処理の設定を行う．通信を許可するのであれば accept，拒否するのであれば deny を設定する．action は rule で主に参照される．また，action は，他の設定にも用いるため，通信の可否の設定の場合には，"disposition" も必要となる．
- **rule**
  rule では各 condition に action と優先度を設定する．rule にも各 policy 同様，名称が付けられる．設定された複数の rule 間での優先度を"precedence"を用いて設定することができる．

以上のような方法で ACL を設定する．また，各 service，group，condition，action，rule はハードウェア上に設定されるため，設定可能な数には制限がある．OmniSwitch の ACL に関連する各設定項目の上限は**表 2** のようになっている[6]．ACL 全体はこの上限に収まるサイズでなくてはならない．多くの情報を ACL に格納したい場合や，簡潔な ACL を構成したい場合，冗長な設定を削除することが必須である．本論文では，network group に着目することで，ACL の短縮化を目指す．

表 2 ACL の仕様
Table 2 ACL Specifications

Maximum number of policy rules	2048
Maximum number of policy conditions	2048
Maximum number of policy actions	2048
Maximum number of policy services	256
Maximum number of groups	1024
Maximum number of group entries	512 per group

本論文では，OmniSwitch の書式に則った ACL を利用するが，他のハードウェアにより ACL の処理を行うネットワーク機器でも類似の書式のものが多く*1，つまり，アクセス制御したいネットワーク，アクセス制御の適用条件，通過や遮断などをそれぞれ設定して，最終的にこれらを利用してアクセス制御規則を定義している．よって，本論文で提案する方法は OmniSwitch に特化しているが，他のハードウェア処理される ACL の短縮にも利用できると考えている．

3. Access Control List 短縮アルゴリズムの開発

3.1 課題設定

ACL のハードウェア処理を行う場合，ACL のサイズはハードウェアにより制限されてしまう．ハードウェアのサイズよりも大きい ACL がある場合，それをハードのサイズまで短縮できれば処理が可能となる．本論文では入力としてハードウェアのサイズよりも大きいサイズの ACL を与える．ACL を短縮していきハードウェアと ACL のサイズが等しくなった時点で処理を終了し，短縮した ACL を出力するようなプログラムを作成する．

3.2 開発方針

定めた課題を解決するため，本論文では network group に着目することで ACL の短縮を試みる．ACL を構成するにあたり，複雑なマスク処理が必要な IP アドレスを扱う network group では他の記述と内容が重複してしまう事態がおきやすい．なお mac group

*1 Enterasys 社のものについても類似の書式であった．

も同様だが，今回扱ったサンプルのACLはnetwork groupを多く扱っていたため，mac groupの処理は本論文では扱わない．今回のACLの短縮はnetwork groupの無駄を排除することで実現を図る．

ACLの短縮にあたりもっとも重要なことは，そのACLが持つ意味を変化させないことである．そのため短縮処理を行うのは，等しい優先度，等しい許可属性を持つ2つのruleを見ることから始める．さらには選んだ2つのruleが扱う2つのconditionが等しい種類の要素を扱っているかも確認しなければならない．加えて今回はsource/destinationにnetwork groupを持つconditionに限る．そうしたチェックを行った上で，ACLの短縮処理を行える可能性がある状況を以下に挙げる．

- **sourceが等しくdestinationが異なる**
  表3に例を挙げる．この例ではnetwork groupN1からnetwork groupN2,N3への通信を許可している．ここでN2とN3を1つのnetwork groupにするとACLを短縮できる．例えばN2にN3を含めると3,5,8行目を削除できる．
- **destinationが等しくsourceが異なる**
  表4に例を挙げる．この例ではnetwork groupN1,N3からnetwork groupN2への通信を許可している．ここでN1とN3を1つのnetwork groupにするとACLを短縮できる．例えばN1にN3を含めると3,5,8行目を削除できる．
- **一方のgroupがもう一方のgroupを含む**
  2つのconditionを見て処理を行う場合，source1がsource2を含む，またはその逆，destination1がdestination2を含む，またはその逆の4種が考えられる．**表5**に例を挙げる．この例ではN1で指定しているIPアドレスはN2で指定しているIPアドレスを含んでいる．そのため2,5,8行目は削除できる．

以上のような状態を発見した場合でもすぐにnetwork groupを削除できるとは限らない．なぜならそのnetwork groupは別のconditionで用いられている可能性があるためである．groupを削除することで他のconditionが意味をなさなくなってしまう危険性がある．従ってnetwork groupがconditionに現れる回数を考慮しなければgroupの削除を行うことはできない．**表6**に例を示す．この例ではconditionC1,C2においてnetwork groupN1,N2は同じdestinationに用いられているため1つのgroupにできそうだが，network gorupN2はconditionC3でも用いられているため，安易に1つのgroupにはできない．

また，削除対象となったnetwork groupで設定されたIPアドレスの範囲が全く別のnetwork groupで設定されたIPアドレスの範囲と重複している場合には，削除対象となったnetwork groupを削除できるとは限らない．よって，削除対象となったnetwork groupが他のnetwork groupとアドレスの重複が起きていないかどうかをチェックする必要がある．

本論文では2つのrule,2つのconditionを見るため最大4つのnetwork groupを扱う．従ってnetwork groupのcondition出現回数が最大4までなら短縮処理が可能と考えられる．**表7**に例を示す．この例でrule R1とR2を見た場合，利用されているcondition C1とC2のsourceはN1とN2であるが，destinationは同じなので短縮できる可能性がある．N1,N2のcondition出現回数はともに2で，その2回とも用いられているnetwork groupはN1またはN2である．そのためこの例では2,5,7,10,12行目が削除できる．また，network groupのcondition出現回数により削除できるgroup,condition,ruleの数は**表8**のように変わってくる．例えば，ある2つのnetwork groupがN1，N2がconditionやruleで同じように利用されているとすると，N1とN2を1つのnetwork groupにまとめて，N2に関するconditionやruleをすべて削除することができる．この時，network groupはN2に関するものを1つ，conditionやruleはN2が利用されている分だけで削除され，conditionとruleはほとんどの場合同数であるので，表8のような削減数となる．

これまで示した方法では，2つのrule，2つのconditionに着目して短縮化を行っているが，より多くのruleやconditionに着目して短縮化を行うこともできる．しかし，多くのruleやconditionに着目すると処理時間が大きくなってしまう．そのため，本論文でのACLの短縮化は2つのrule，2つのconditionまでに着目することとした．また，着目するruleやconditionの選択についても様々な方法が考えられるが，本論文では，ACLに記載されている順番で選択することとし，選択方法に工夫を行うことはしない．

3.3 実装内容

プログラム言語はPerl(ver.3.05)を用いて開発を行った．コマンドラインから入力のACL，処理後に出力するファイル名を指定できるようにしておく．まずはじめにハードウェアのサイズを指定する．後に変更しやすいよう，表2で示されたサイズを変数で割った数がハードウェアのサイズとなるようにした．入力として与えられたACLをpolicyの種別に配列へ格納し，そのサイズを計測する．この時点でハードウェアのサイズを下回っていれば短縮処理は必要ないためプログラムは終了する．このアルゴリズムで参照し，かつ実際に短縮するrule,condition,network groupは，CAMを模倣するため連想配列へと格納する．さらにnetwork groupを見ることで短縮可能かどうかを判別し，可能だった場合はそのgroupに関係するconditionを参照しなければいけないため，condition,source group,destination groupのそれぞれの名称を格納した二次元配列を用意して参照を簡単に行えるようにし

表 3　source が等しく destination が異なる例
Table 3　An example that source addresses are the same and destination addresses are not

```
1 policy network group N1 IPaddressA/mask IPaddressB/mask
2 policy network group N2 IPaddressC/mask IPaddressD/mask
3 policy network group N3 IPaddressE/mask IPaddressF/mask
4 policy condition C1 source network group N1 destination network group N2
5 policy condition C2 source network group N1 destination network group N3
6 policy action A1 disposition accept
7 policy rule R1 precedence 10 condition C1 action A1
8 policy rule R2 precedence 10 condition C2 action A1
```

表 4　destination が等しく source が異なる例
Table 4　An example that destination addresses are the same and source addresses are not

```
1 policy network group N1 IPaddressA/mask IPaddressB/mask
2 policy network group N2 IPaddressC/mask IPaddressD/mask
3 policy network group N3 IPaddressE/mask IPaddressF/mask
4 policy condition C1 source network group N1 destination network group N2
5 policy condition C2 source network group N3 destination network group N2
6 policy action A1 disposition accept
7 policy rule R1 precedence 10 condition C1 action A1
8 policy rule R2 precedence 10 condition C2 action A1
```

表 5　一方の group がもう一方の group を含む例
Table 5　An example that one group contains the other group

```
1 policy network group N1 33.38.0.0 mask 255.255.0.0
2 policy network group N2 33.38.1.0 mask 255.255.255.0 33.38.2.0 mask 255.255.255.0
3 policy network group N3 33.37.0.0 mask 255.255.0.0
4 policy condition C1 source network group N1 destination network group N3
5 policy condition C2 source network group N2 destination network group N3
6 policy action A1 disposition accept
7 policy rule R1 precedence 10 condition C1 action A1
8 policy rule R2 precedence 10 condition C2 action A1
```

た．以上のようなデータ構造で ACL の情報を格納したうえで，次に前述の開発方針から，ACL の短縮に必要な条件は以下のようなものが挙げられる．その条件に沿って機能の手順を構成し実装した内容の概要を記述する．

短縮方法の概要

手順 1　優先度，許可属性が等しい

policy rule で設定されている rule を順番に見ていき優先度と許可属性が等しいものを見つけ出す．見つかった場合，その 2 つの rule が扱う 2 つの condition の名前を保存して手順 2 へ．見つからなかった場合はこのプログラムでは入力の ACL を短縮できないためプログラムを終了する．

手順 2　condition で扱う種類が等しい

手順 1 で保存した 2 つの condition の設定内容を見て共に network group を扱うものならば，そこで扱われる 4 つの network group の名前を保存して手順 3 へ．network group を扱っていなければ手順 1 へ戻る．

手順 3　2 つの source/destination において，group が等しい，もしくは group の一方が他方を含む

手順 2 で保存した 4 つの network group に設定された IP アドレスから 2 つの source または 2 つの destination それぞれにおいて等しいもしくは一方が他方を含んでいて短縮可能性があるかどうかを判別する．短縮ができる可能性があるならば手順 4 へ．なければ手順 1 へ戻る．

手順 4　等しい，もしくは含まれている group を削除しても condition に影響がない

手順 3 で短縮できる可能性があると判別された network group が，policy condition において手順 2 で求めた 4 つの network group としか関わりがなく，他の network group と IP アドレスに重なりがなければ，その network group を削除しても condition に影響がないことがわかる．このチェックを行い，可能であればその network group を削除する．加えてその network group を持つ condition，その condiotion を扱う rule も削除する．network group を削除することで condition に影響が出るのであれば手順 1 へ戻る．

以上の手順の中で ACL を短縮できた場合，その都

度ハードウェアのサイズとACLのサイズを比較する．ハードウェアのサイズと等しいもしくはそれ以下となっていたら，その時点で短縮したACLをファイルに出力してプログラムを終了する．ハードウェアのサイズよりも大きかったら処理を手順1から繰り返す．手順1をすべてのruleに対して行ったらそのときのACLのサイズを記録しておく．再び手順1から処理を行いruleを見終わったらそのサイズと比較し，変化がなかったらこのプログラムではACLのサイズをハードウェアのサイズに納めることができないということがことがわかるので，エラーメッセージと共にプログラムを終了する．

4. 実　　験

4.1 方　　法

この実験では，OSはLinux，CPUはPentium(R) 4 CPU 1.80GHz，メモリは512MBのPCを用いた．埼玉大学で実際に使われているACLのテキストファイルを入力として使う．ただし，本論文に実際のACLを掲載することはセキュリティ上問題があるので，ここで示しているのは実際のACLにあるアドレスや名称を全く別のものに置き換えたものである．

実際に運用されているACLは表2に示されているサイズのハードウェアに格納済みのものである．そこでハードウェアのサイズを表2の1/8スケールと仮定し，サンプルのACLを実装したプログラムがこのサイズに収まるACLへ直すことが出来るかを確認する．テストに使用したサンプルのACLのサイズとハードウェアのサイズを**表9**に示す．この表のようにサンプルのACLでは，network groupがハードウェアサイズを超えている．また，プログラムの能力を見るため，ACLを可能な限り短縮するテストも行う．このとき指定したハードウェアのサイズは1/64スケールである．

4.2 結　　果

プログラムをハードウェアの1/8スケールまでACLを短縮した結果と，ハードウェアのサイズを1/64スケールに指定しこれ以上短縮処理ができなくなったときの結果を**表10**に示す．なお，本論文のアルゴリズムでは処理をnetwork group,condition,ruleにしか行っていないためそのほかのpolicyは処理前と処理後に変化はない．そのため以下の結果には短縮処理により変化したnetwork group,condition,ruleの部分のみを示す．

なお，処理に要した時間は1/8スケールのときで11.45秒，1/64スケールのときで19.16秒であった．また，プログラム実行時のメモリ使用量はともに約3.5Mbyteであった．

実際にACLの中で短縮を行った箇所を，短縮処理前のACLの一部である**表11**と短縮処理後のACLの一部である**表12**との比較により示す．表11の中の太字で書かれている部分が処理後に削除され，表12の中の太字で書かれている部分が処理後に追加される．

5. 考　　察

実験結果より，本論文で開発したプログラムはACLを短縮化することに成功した．また，ハードウェアのサイズを1/64スケールに設定して可能な限り短縮処理を施したときには，1/8スケールのときよりもrule,conditionの数を削減したACLを構成できた．しかしながら，そのACLがもっとも短いACLであるとは限らない．なぜならこのプログラムはpolicy ruleを上から順番に処理対象としているためで，表7に示したようにnetwork group同士の関係性によりACLのサイズを削減する能力が変化するこのアルゴリズムではruleを参照する順番でACLを短縮する量も変化してしまうからである．この問題を解消するには，優先度と許可属性の等しいruleをランダムに選んで処理を行うプログラムに改良することが必要となる．ACLを可能な限り短縮してもハードウェアのサイズに収まらなかった場合にこの手法が求められる．

メモリの使用量に関してはハードウェアサイズが1/8スケールのときと1/64スケールのときで変化は生じていない．これは入力のACLを配列に格納して処理を行っているためで，入力として与えるACLが大きくなればなるほどメモリ使用量は大きくなるだろうと考えられる．

また，ACLの各policyには固有の名称が付けられているが，このプログラムではpolicyの性質のみを維持し，名称の情報は削除される．ハードウェアのサイズに格納することを大前提としたため，出力されたACLを見てもそのpolicyの名称が示す情報と性質の情報には差異が生まれてしまう可能性があると考えられる．例えば，network group N1,N2を一つにしたgroupの名称をN1&N2にするなど，policyの名称も維持したほうが処理後のACLを見る人にとってわかりやすいものが構成できたのではないかと思われる．

短縮後のACLと短縮前のACLの意味が変わっていないことは当然要求されることである．ここでは，その証明の概要を示す．本論文のACLの短縮化は非常の簡単な方法であり，つまり，出現回数が多いnetwork groupは対象にはならず，また，network groupの削減を行う場合に，削減による副作用が起きないように，削減対象となるnetwork groupが他で利用されていないかそして，IPアドレスが他のnetwork groupで利用されていないかをチェックしている．よって，短縮前後でACLの意味に変化はない．

本論文での実験では，アルゴリズムが簡単であるため，当初にも関わらず，ACLを短縮できた．この理由を考えると，ネットワーク管理者，つまり，人間が

表 6　1 つの network group が複数回用いられる ACL の例
Table 6　An example that one network group is used several times

```
policy network group N1 IPaddressA/mask IPaddressB/mask
policy network group N2 IPaddressC/mask IPaddressD/mask
policy network group N3 IPaddressE/mask IPaddressF/mask
policy condition C1 source network group N1 destination network group N3
policy condition C2 source network group N2 destination network group N3
policy condition C3 source network group N2 destination network group N4
policy action A1 disposition accept
policy rule R1 precedence 100 condition C1 action A1
policy rule R2 precedence 100 condition C2 action A1
policy rule R3 precedence 10 condition C3 action A1
```

表 7　condition に複数回出現する network group を削除できる ACL の例
Table 7　An ACL example that the network group which appears on several conditions can be removed

```
1 policy network group N1 IPaddressA/mask IPaddressB/mask
2 policy network group N2 IPaddressC/mask IPaddressD/mask
3 policy network group N3 IPaddressE/mask IPaddressF/mask
4 policy condition C1 source network group N1 destination network group N3
5 policy condition C2 source network group N2 destination network group N3
6 policy condition C3 source network group N3 destination network group N1
7 policy condition C4 source network group N3 destination network group N2
8 policy action A1 disposition accept
9 policy rule R1 precedence 10 condition C1 action A1
10 policy rule R2 precedence 10 condition C2 action A1
11 policy rule R3 precedence 10 condition C3 action A1
12 policy rule R4 precedence 10 condition C4 action A1
```

表 8　condition 出現回数に応じた削除可能な要素の数
Table 8　The number of elements which can be removed accoding to the number of occurrences on condition

	Delete groups	Delete conditions	Delete rules
1 network group in condition	1	1	1
2 network groups in condition	1	2	2
3 network groups in condition	1	2	2
4 network groups in condition	1	3	3

表 9　テストに用いるハードウェアと ACL のサイズ
Table 9　The sizes of hardware and ACL in the experiment

	1/8 Scale Size	Sample ACL Size
Number of policy rules	256	176
Number of policy conditions	256	176
Number of policy actions	256	2
Number of policy services	32	9
Number of network groups	128	**153**

表 10　テスト結果
Table 10　The test result

	1/8 hardware scale	1/64 hardware scale
Number of policy rules	**140**	**103**
Number of policy conditions	**140**	**103**
Number of network groups	**128**	**108**

表 11　処理前の ACL の一部
Table 11　A part of ACL before processing

policy network group G1 44.8.53.192 mask 255.255.255.224
policy network group G2 44.8.7.0 mask 255.255.255.192
policy network group G3 44.8.22.192 mask 255.255.255.192
policy condition c1 source network group G1 destination network group G3
policy condition c2 source network group G2 destination network group G3
policy action OK disposition accept
policy rule r1 precedence 10 condition c1 action OK reflexive
policy rule r2 precedence 10 condition c2 action OK reflexive

表 12　処理後の ACL の一部
Table 12　A part of ACL after processing

policy network group G1 44.8.53.192 mask 255.255.255.224 **44.8.7.0 mask 255.255.255.192**
policy network group G3 44.8.22.192 mask 255.255.255.192
policy condition c1 source network group G1 destination network group G3
policy action OK disposition accept
policy rule r1 precedence 10 condition c1 action OK reflexive

書く ACL はそれほど複雑ではなく，また，同一管理者または管理組織によって管理されている ACL の書き方はネットワーク毎にことなっているわけではなく，類似している部分が多いと考えられる．そのため，本論文で示したような簡単は短縮化でも，ACL のサイズを小さくできたのではないかと考えられる．

また，ネットワーク機器メーカは ACL の作成ツールを提供している場合が多いが，埼玉大学では利用してはいない．これは，ネットワーク管理者がツールの使い方を理解するよりも，ACL を直接読むことのほうになれているからであると思われる．このようなネットワーク管理者が少なからずいることを考えると，本論文で提案している方法は有効ではないかと思われる．

6. おわりに

本論文ではハードウェアのサイズに収まらない ACL に対し，ハードウェア処理が可能となるようなサイズまで ACL を短縮するプログラムの開発を行った．ACL の持つ機能が変化しないようなアルゴリズムを考案してプログラムを実装し，テストを行った．その結果，実際に利用されている ACL を短縮させることができた．確認すると ACL の condition や group の記述内容が持つ意味に変化はみられなかったが，いくつかの改善すべき点も見つかった．

考察でも触れたように，policy の名称を維持したまま短縮処理がなされるような改善が求められる．他には，各 rule には優先度が設定されているが，同じ優先度で同じ規則にマッチする通信があった場合，policy rule でより上に記述されているものが採用される．本論文で開発したプログラムは ACL の短縮処理によりその順番が元の ACL と変化しているため，rule の位置に関して改善が必要である．

本論文では 2 つの rule,2 つの condition を見て短縮処理を行っているため，最大で 4 つの group が関わる状況でしか処理を行えない．しかし実際の ACL にはより複雑にかかわり合う group も存在するため，そういった group の処理も考案しなければならない．また，本論文のプログラムでは mac group,port group を見た処理は行っていない．より汎用性の高いプログラムとするには，それらを用いた短縮処理の手法も必要となる．また，考察で挙げたように，短縮後の ACL の健全性をテストするプログラムの作成が求められる．

参考文献

1) Hazem Hamed,Ehab Al-Shaer:"Dynamic Rule-ordering Optimization for High-speed Firewall Filtering",ASIACCS,2006.
2) Subrata Acharya, Mehmud Abliz, Bryan Mills, Taieb F.Znati, Jia Wang, Zihui Ge,Albert Greenberg:"OPTWALL : A Hierarchical Traffic-Aware Firewall",NDSS Symposium,2007.
3) Vic Grout,John McGinn,John Davies:"Real-time optimisation of access control lists for efficient Internet packet filtering",J Heuristics,2007.
4) Dominique Alessandri:"Access Control List Processing in Hardware",Diploma Thesis, Zurich, Switzerland: ETH,Electrical Engineering Department,1997.
5) Shingo Ata, Haesung Hwang, Koji Yamamoto, Kazunari Inoue, Masayuki Murata:"Managment of Routing Table in TCAM for Reducing Cost and Power Consumption",IEICE technical report.information networks,2007.
6) Alcatel:"OmniSwitch 7700/7800 OmniSwitch 8800 Network Configuration Guide",2005.