広域ネットワークにおける自律的なグループ形成機構を用いた分散ネットワークモニタ方式の提案

内山 彰[†] 梅津高朗[†]
安本慶一[††] 東野輝夫[†]

本論文では，広域ネットワークにおいて複数領域にわたるウイルスや DoS 攻撃などの問題が発生している範囲を特定するため，自律的なグループ形成機構を用いた分散ネットワークモニタ方式 FLEXA を提案する．FLEXA では，問題検出時にあらかじめ指定された隣接エージェントが，同種の問題を検出している場合に自律的にグループを形成して，グループ内で監視情報を共有する．異なるネットワークに存在するセグメント間でも論理的に隣接していると定義できるため，ネットワーク環境に応じた柔軟な監視が可能となる．収集された情報は，一定のルールに基づきグループ内の複数ノードから管理者に送信されるため，そのうちの一部のメンバが管理者と通信困難な状況になっても，他のメンバが共有した情報を管理者に送信することで高確率で情報を取得できる．また，グループ情報を集約して送信することで，管理者付近の監視用トラフィックを削減できる．提案方式を実現するための枠組みとして，自律的なグループ形成，グループ内通信などの機能を提供するいくつかの API を実装し，シミュレーションによる評価を通して，提案方式の有効性を確認した．

A Proposal of Distributed Network Monitoring with Autonomous Group Formation

Akira Uchiyama,[†] Takaaki Umedu,[†] Keiichi Yasumoto[††] and Teruo Higashino[†]

In this paper, we propose a distributed network monitor with autonomous group formation to detect various problems such as viruses and DoS attacks in wide area networks. In this technique, multiple neighboring nodes where the same kinds of problems are occurring form groups autonomously and share the information by the communication in each group. Logical neighboring relations can be specified between network segments even if they are located in different networks for flexible monitoring. In order to increase the probability of success in gathering information, our network monitor allows any member of the group to send the shared information to the administrator node. Furthermore, by summarizing the information in each group, it can reduce the control traffic around the administrator node. We have designed and implemented several APIs as primitives to achieve the proposed technique. We have confirmed effectiveness of our technique through ns-2 simulation.

1. まえがき

近年，ネットワークの構成ノードにおける OS の脆弱性などをつくウイルスやサービス拒否（DoS）攻撃[1)~3)] が後を絶たず，その損害は莫大なものとなっている．このため，広域ネットワークにおいてネットワークトラフィックを様々な項目に関して監視することで，構成ノードにおける障害や異常なトラフィックを早期に検知し，問題が発生しているノード群（**問題範囲**）を迅速に把握して，ウイルス，DoS 攻撃などの影響をできるだけ小さくするためのネットワーク監視手法の確立が求められている．

これまでにいくつかのネットワーク監視手法が提案されている．文献 4) では，ノード間で論理的な木構造を構築することによって階層構造を持たせ，拡張性を高めているが，階層構造が固定されているため時々刻々と変化する問題範囲の特定に時間がかかってしまう場合がある．文献 5)〜7) で提案されている監視対象ネットワークを分割し，各領域を担当するモバイルエージェントに巡回させることによって情報収集を行

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

う手法や，文献 8) で提案されている情報収集頻度を下げる手法など，監視用トラフィックを削減する手法も提案されている．しかし，これらの方式では問題範囲の特定には収集した情報の解析が必要となり，管理者に大きな負担がかかることになる．また，これらの方式では，ネットワークセグメント間の隣接関係は基本的に物理的なリンク接続に基づいて決定されている．しかし，昨今のウイルスや DDoS 攻撃は異なるネットワークに存在するセグメントに設置された同じ企業のサーバなどの，物理的に隣接していない範囲に及ぶ場合がある．したがって，ネットワークセグメント間の論理的な接続関係（論理的な隣接関係）に基づいて監視することは，DDoS 攻撃などによる問題範囲を特定するために大変重要である．さらに，モバイルエージェントを利用した方法では，巡回経路によっては，リンク障害などの問題発生時にモバイルエージェントが管理者のネットワークへ戻ることができず，迅速な問題検出が困難な場合がある．

本論文では，これら既存手法の問題を解決するため，自律分散型のネットワークモニタ方式 FLEXA（FLEXible Autonomous network monitor）を提案する．FLEXA では，各セグメントに分散配置されたエージェントに対してあらかじめ監視項目や問題発生時の動作などを指定する**監視シナリオ**を与える．各エージェントはネットワークを監視してワームの感染，トラフィックの集中または（分散）DoS 攻撃などの問題を検出した際に，監視シナリオに基づいて自律的に問題範囲でグループを形成し，グループ内通信によってメンバ間で監視情報を共有し，必要に応じてパケットフィルタリングなどの処置を講じて問題解決を図る．収集した情報はグループに属するエージェント群のうち，管理者と通信可能なエージェントから送信されるため，高い確率で管理者に届けられる．

FLEXA における監視シナリオを容易に実現するための API を開発し，ネットワークシミュレータ ns-2 上に実装し，実験を行った．その結果，既存の階層型手法と比較して，FLEXA が十分に小さな制御トラフィックで，問題範囲を迅速に特定可能なことを確認した．

2. FLEXA を実現するためのミドルウェア

2.1 FLEXA の概要

FLEXA では，**センサエージェント**と**管理用エージェント**の 2 種類のエージェントを用いる．センサエージェントは監視対象とする各ネットワークセグメントに分散配置され，それぞれが監視項目に基づき，トラフィック情報やセグメントの状態などの情報を収集する．問題検出および情報収集は，SNMP や tcpdump などを用いて宛先アドレスやポート番号ごとのトラフィック量を取得したり，Snort [9] や他の侵入検知システム [10),11)] と連携したりして行う．各センサエージェントは，監視を行ううえで隣接していると見なすべき**隣接エージェント**のアドレスを保持し，それらが同じ問題を検出している場合にはグループを形成する．グループはあらかじめ指定されたサイズまで自律的に拡大し，それぞれが収集している情報をグループ内で互いに交換・集約する．集約された情報を基に，自律的に監視項目を変更することで，問題に対して原因を絞り込んだり，パケットフィルタリングなどを行ったりしながら解決を図る．センサエージェントが収集する情報の種類や，グループを形成する動作などは，管理者が監視シナリオを記述して与える．以降，特に明示しない限りエージェントという表記はセンサエージェントを指すものとする．

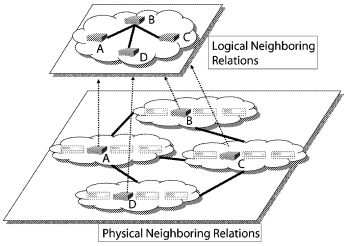


図 1 物理的/論理的な隣接関係
Fig. 1 Physical/logical neighboring relations.

管理用エージェントは，管理者が操作するコンピュータ上で実行され，センサエージェントへの監視シナリオの配布や，各ネットワークセグメントの監視情報の実時間表示などの機能を管理者に提供する．また，管理者は管理用エージェントを介して，指定したグループに対して直接，監視項目やグループ形成条件の変更などの指示を送ることができる．

2.2 隣接関係

各エージェントの隣接エージェントは，**隣接関係**として指定される．基本的に隣接関係は，既存のネットワーク監視手法と同様に物理的な隣接関係に基づいて定義される．しかし，異なる複数のネットワークに設置されているセグメントが同時に攻撃される場合があるため，論理的な隣接関係も定義できる．論理的な隣接関係の例を**図 1** に示す．

たとえば，同一企業の本社，支社間や，複数プロバイダに接続しているネットワークセグメントなどは複数箇所から同時に攻撃を受ける可能性が高いと考えられる．また，以前に攻撃を受けたことのあるネットワークや特にトラフィック量の多いリンクなどは，よ

表 1 提供する API
Table 1 Provided APIs.

API 名	説明
formGroup(*channel*, *cond*, *maxsize*)	条件 *cond* を満たす隣接エージェントに対してグループ形成を要求する．グループ形成時には指定されたチャネル *channel* がメンバ間で共有される．“g, h” のように複数のチャネルを *channel* として指定してもよい．この API は現在のグループメンバ数を返す．
numberOf(*cond*)	グループ内で条件 *cond* を満たすようなエージェントの数を返す．
setCommand(*channel*, *command*, *cond*)	*cond* で指定された監視項目に関するコマンド *command* をチャネル *channel* を共有しているエージェントに配布する．監視項目の変更やパケットフィルタリングの指示などに用いる．
info = **getGrpInfo**(*channel*)	チャネル *channel* を共有しているメンバから，現在の監視項目のデータを取得する．監視項目のデータの配列 *info* が返される．
sendInfo(*channel*, *info*)	チャネル *channel* を経由して情報 *info* を送信する．管理者への情報送信，および上位層に属するグループへの情報送信に用いる．
getInfo(*channel*)	チャネル *channel* を経由して情報を受信する．
disconnect(G, S)	所属しているグループ G から，集合 S で指定されたメンバが離脱する．

り厳密に監視される必要があるため，論理的な隣接関係を持たせることで監視を強化するなどといった利用法も考えられる．また，ネットワーク構成に階層構造がある場合には下位層の問題を迅速に上位層に伝える必要があるため，そういったセグメント間でも隣接関係を定義することは有効である．提案方式では必要に応じて動的に，これらの隣接関係を管理用エージェントを用いて追加・削除することができる．

2.3 ミドルウェアが提供する API

提案方式では監視シナリオの記述を容易に行えるようにするため，グループ形成やグループ通信を実現する API を提供している（**表 1**）．

formGroup(*channel*, *cond*, *maxsize*)☆は，エージェント間でグループを形成するための API である．**formGroup** は非同期的に **formGroup** 以降の動作記述と並行にグループ形成手続きを開始し，グループが形成されるまで処理を続ける．最初に **formGroup** を呼び出したときの返り値は 1 となる．2 回目以降はそのときのグループメンバの数を返し，グループ形成手続がすでに終了していれば，グループ形成手続が再び開始される．条件 *cond* を指定して **formGroup** が実行された際，隣接エージェントが **formGroup** を同様の条件で実行していれば，それらのエージェント間でグループが形成される．グループ内では，指定されたチャネル *channel* がメンバ間で共有され，グループ内での情報交換に使われる．チャネルを指定せず実行した場合には，新たなチャネルが生成される．また，*channel* として，“g, h” のように複数のチャネルを指定することもできる．**formGroup** を繰り返し実行することによって，隣接エージェント，あるいは隣接エージェントを含むグループと，グループを形成できる．グループが際限なく拡大して効率が低下することを抑制するために，形成されるグループのメンバ数の上限は **formGroup** の引数 *maxsize* で指定する．もとのグループで用いているものとは異なるチャネルを用いてグループ間で階層構造を構築することで，よりサイズの大きいグループを形成することもできる．たとえば**図 2** のように，全体で結合すると制限サイズを超えてしまうような 3 つのグループ G_1，G_2，G_3 が存在したとする．これらのグループがチャネル h を指定して，同じ条件 *cond* で **formGroup** を実行した場合，これらのグループを要素とする上位階層の新しいグループ H が形成される．

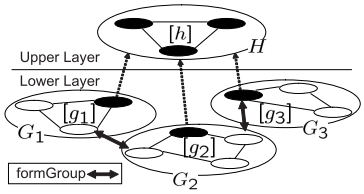


図 2 グループの階層構造
Fig. 2 Hierarchical structure of groups.

formGroup，**numberOf**，**setCommand** の引数 *cond* には，各セグメントの情報を取得する，監視プリミティブ（**表 2**）を用いて，*warning.snort* == MS_BLASTER（Snort が MS Blaster を検出）や *icmp.traffic* > 3,000,000（icmp トラフィックが 3 Mbps を超えている）のような条件式が指定可能である．表 2 以外のプリミティブを必要に応じて実装し，追加することもできる．

☆ 以降，*channel* は引数としてのチャネルを指すものとする．それ以外の場合はチャネルと表記する．

表 2 センサエージェントが利用するプリミティブ
Table 2 Primitives used by sensor agents.

プリミティブ	説明
tcp.traffic(*pn*, *src*, *dst*)	現在の TCP トラフィック量（bps）．*pn*，*src*，*dst* を用いてそれぞれ特定のポート番号，送信元 IP アドレス，および宛先アドレスが指定可能．これらの引数は省略できる．
udp.traffic(*pn*, *src*, *dst*)	現在の UDP トラフィック量（bps）．引数は *tcp.traffic* と同様．
icmp.traffic(*src*, *dst*)	現在の ICMP トラフィック量（bps）．*pn* が指定できないことを除けば，*tcp.traffic* と同様．
ip.traffic(*src*, *dst*)	現在の IP トラフィック量（bps）．引数は *icmp.traffic* と同様．
throughput.traffic(*pn*, *src*, *dst*)	現在のスループット（bps）．引数は *tcp.traffic* と同様．
anomaly.traffic	監視トラフィックで検出された異常．返り値は異常の種類．異常がない場合には 0 が返される．
warning.snort	Snort により出力される警告情報．返り値は警告の種類．警告がない場合には 0 が返される．
id	実行したエージェントの識別子が返される．
summary(*info*)	*info* で指定された情報の統計情報を計算して返す．

setCommand(*channel*, *command*, *cond*) はグループメンバに対してコマンドを送信するための API である．コマンド *command* には，監視を行う *Watch*，フィルタリングを行う *Filter* などが指定可能である．たとえば，それぞれ *command*，*cond* として *Watch* および *throughput.traffic*（スループット）が指定された場合，監視項目が *throughput.traffic* に設定され，全メンバが各セグメントにおいてこの情報を収集する．

info = **getGrpInfo**(*channel*) は現在の監視項目のデータをグループに属する全メンバから収集するための API である．この API は返り値として各メンバの監視項目のデータの配列 *info* を返す．

sendInfo および **getInfo** はグループメンバと管理者の間，もしくは図 2 のような階層が存在する場合に下位層のグループメンバと上位層のグループメンバの間で *channel* を介した 1 対 1 通信による情報交換を行うための API である．あるエージェントが全メンバのトラフィック情報 *info* を取得した後，**sendInfo**(*h*, *info*) を実行し，管理用エージェントが **getInfo**(*h*, *info*) を実行していると，管理者に対して *info* の内容（グループの全メンバが収集した監視データ）が送信される．その際，通信量を削減するため，全メンバの情報をそのまま送信する代わりに，プリミティブ *summary* を利用して，全メンバの監視データを集約して送信することもできる．たとえば，*summary*(*info*) を実行すると，グループメンバ全員のトラフィック情報 *info* から宛先別の平均値や最大値，最小値などを計算し，その結果を返す．**sendInfo** は対応する **getInfo** が実行されていない（通信できない）場合，データを転送せずに実行終了する．いくつかのグループメンバが同時に **sendInfo** を実行した場合，そのうちのいずれかが管理用エージェントに情報を送信できればその後実行された **sendInfo** はスキップ（データ転送せずに終了）されるため，そのような監視シナリオを記述することで通信困難時のデータ取得に成功する確率の向上，および管理用エージェントが受け取るデータ容量の削減を同時に達成できる．

disconnect(*G*, *S*) はグループ *G* から集合 *S* で指定されたメンバが離脱するための API である．この API は問題が解決した場合などに使う．

2.4 監視シナリオの記述例

前述の API やプリミティブを用いて，自律的な分散型ネットワークモニタの監視シナリオを簡潔に記述できる．この監視シナリオはスクリプトプログラムとして記述される．センサエージェントに与える監視シナリオの例を以下に示す（**図 3**）．

（1） 監視対象となるセグメントにおいて動作している侵入検知システム Snort [9] が MS Blaster を検出した場合，隣接エージェントとグループを形成する．グループ形成時に発生するトラフィック量，および，グループ形成に必要な時間を考慮して，各グループのメンバ数の上限は 10 とする．

（2） グループ形成後メンバ数が上限値の半数以上（5 以上）になったら，管理者ノードをグループに加え，各メンバが監視セグメントの現在のトラフィック量を集約して管理者に送信する．さらにグループメンバに対して **setCommand** によりコマンドを送信し，**getGrpInfo** で現在のトラフィック量を取得する．

（3） 現在のトラフィック量が閾値（5 Mbps）を超えているメンバ数がグループメンバの半数を超えていた場合，全メンバに対して TCP パケットの 135 番

```
while(TRUE){
  warning_type = warning.snort;
  switch (warning_type){
  case MS_BLASTER:
    while(TRUE){
      num_of_members = formGroup(‘‘g, h’’,
                ‘‘warning.snort == MS_BLASTER’’, 10);
      if (num_of_members >= 5){
        setCommand(‘‘g’’, Watch,
                ‘‘throughput.traffic’’);
        info = getGrpInfo(‘‘g’’);
        formGroup(‘‘h’’, ‘‘id == Manager’’, 2);
        sendInfo(‘‘h’’, summary(‘‘info’’));
        if(numberOf(‘‘info >= 5Mbps’’)
                                  >= num_of_members/2){
          setCommand(‘‘g’’, Watch,
                      ‘‘throughput.traffic(135)’’);
          setCommand(‘‘g’’, Filter,
                  ‘‘throughput.traffic(135) >= 50%’’);
          info = getGrpInfo(‘‘g’’);
          sendInfo(‘‘h’’, summary(‘‘info’’));
        }
      }
    }
    break;
  case BAGLE:
   ...
  case MYDOOM:
   ...
  }
}
```

図 3 監視シナリオ例
Fig. 3 Example scenario for sensor agents.

ポート（すなわち，MS Blaster が使用するポート）に関するトラフィック量を収集するよう，コマンドを送信する．それと同時に，135 番ポートに対するトラフィックが，全トラフィック量の 50%以上を占めている場合には，フィルタリングを行うよう，コマンドを送信する．グループ内の（チャネル g を共有している）各エージェントはこれらのパケットに関する情報を取得し，集約して管理用エージェントへ送信する．

3. ミドルウェアの実装

3.1 自律的なグループ形成機構の実装

文献 12) で提案されているグループ通信ミドルウェアを基に拡張を行い，2 章で述べた API を実装した．

formGroup を呼び出したエージェントは，同じグループに属していないすべての隣接エージェントに対して，グループ形成条件，共有チャネルとともにグループ形成要求メッセージ $GROUP$ を送信する（**図 4** (a)）．$GROUP$ を受信したエージェントは，形成条件を満たすかどうかを判定し，ACK もしくは $NACK$ を返信する（図 4 (b)）．ACK には，各エージェントがすでに形成しているグループのメンバリストが含まれ，ACK を受け取ったエージェントは，$CONF$ メッセージをメンバリストとともに返信し（図 4 (c)），ACK に含まれているメンバリストと，現在保持しているメンバリストを統合する．$CONF$ を受け取ったエージェントも同様にメンバリストを統合する．エージェント間でやりとりされるこれらのメッセージの送受信には，TCP を用いる．このようにして，互いにメンバリストを交換することで，1 回のグループ形成が完了する．

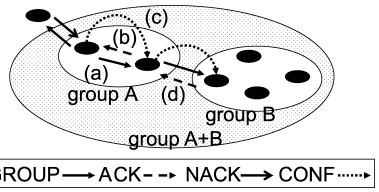


図 4 グループ通信ミドルウェアによるグループ形成
Fig. 4 Formation of a group with our middleware.

グループどうしが結合する場合（図 4 (d)），それぞれのメンバリストが統合され，以下で述べる **getGrpInfo** の実装と同様にしてマルチキャストによってメンバ間で共有される．同時に，1 つのノードがマスタとして ID を基に選出される．グループ間で階層構造を作る場合には，各グループのマスタのみからなる上位グループが構築される（図 2）．

3.2 グループ内通信の実装

setCommnad はグループ内のエージェント間でマスタを介して同期実行される．各メンバはマスタに対して，**setCommand** の実行準備ができたことを知らせるために，指定された *command* のハッシュ値を含む，*ready* メッセージを直接送信する．マスタは同じハッシュ値が付与された *ready* をすべてのメンバから受信すると，*permission* メッセージをグループ内にブロードキャストし，各メンバにコマンドを実行させる．ネットワークが不安定で一部のエージェントが *ready* を送信できない場合には，マスタはタイムアウトが発生するまで待ち，*ready* を受信することができたメンバに対してのみ，*permission* をブロードキャストする．マスタ m が通信困難な状況になった場合，隣接エージェント n がそれを検出し，グループ内にマスタ交替メッセージをブロードキャストして，すべてのメンバから *Ack* を受け取った後，n はマスタとして動作する．隣接エージェントが複数ある場合には，bully アルゴリズム[13] を用いて，マスタを決める．

getGrpInfo はフラッディングベースで実装した（**図 5**）．**getGrpInfo** を実行したメンバは，監視して

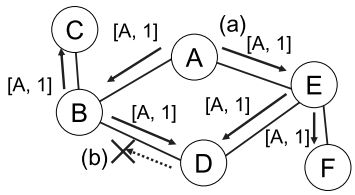


図 5 グループ内フラッディング
Fig. 5 Flooding in a group.

いるセグメントで収集された情報をグループに属するすべての隣接エージェントに送信し（図 5 (a)），それぞれのエージェントは自身の収集した情報を追加しながらフラッディングを行う．ただしメッセージにはセグメント ID と，セグメントごとのシーケンス番号が付与されており，それを比較することで重複して送信することを回避する（図 5 (b)）．全メンバからそれぞれが監視しているセグメントの情報を受信するまで，このエージェントの動作はブロックされる．ただし，**setCommand** と同様，タイムアウトが発生するまで待っても受信できなかったセグメントの情報は諦め，ブロックを解除する．

sendInfo とそれに対応する **getInfo** の組は，同期して実行するためマスタを利用して実装する．あるエージェントが **getInfo** を実行すると，マスタに対して *ready_info* メッセージが送信される．マスタは受信した *ready_info* をキューに追加する．また，他のエージェントが **sendInfo** を実行すると，引数として与えられた情報を含む *write_info* メッセージがマスタに対して送信される．マスタが *write_info* を受信すると，キューの先頭にある *ready_info* を取り出し，その *ready_info* を送信したエージェントに対して *write_info* を転送し，*write_info* の送信者に対しては *ack* メッセージを返信する．キューが空の場合は，一定の時間 *ready_info* が到着するのを待ち，到着しなければ *skipped* メッセージが返信される．上記の機構により，通信困難な状況でもグループ内の複数メンバに **sendInfo** を実行させることで，管理者ノードは高い確率で情報を受け取ることができるとともに同じ情報を重複して受け取らないようにできる．

4. シミュレータによる提案方式の性能評価

提案方式をネットワークシミュレータ ns-2 上に実装し，グループ形成に要する時間，グループ形成時に発生する制御トラフィック量，およびグループ内通信に要するトラフィック量の 3 つの基本性能に関する評価を行った．基本性能の評価により，適切なメンバ数の上限値を定めるための指標を提供し，さらに，提案方式が現実的なコストで実現可能であることを示す．また，問題特定性能として，問題を検出したノード群（すなわち，グループ形成条件を満たしたノード群）の情報を管理者が取得するまでの時間，および管理者が情報を取得できた問題範囲の割合を調べるため，文献 4) のような，ネットワークを物理的な隣接関係を基に複数領域に分割した階層型手法と提案方式で比較を行った．実験は，耐障害性の検証のため，攻撃を受けるノード（攻撃対象ノード）が 1 つで，かつ障害がある場合とない場合，さらに，攻撃対象ノードが 4 つで，かつ障害がない場合の 3 つで行った．攻撃対象ノードが 4 つの場合の実験では，論理的な隣接関係を柔軟に定義できることによる，提案方式の有効性を示す．

表 3 シミュレーションパラメータ
Table 3 Simulation parameters.

ケース I：基本性能評価	
ノード数	50
制御パケットサイズ	1,024 バイト
共有情報パケットサイズ	1,024 バイト/ノード
リンク遅延	1.0–10.0 ミリ秒
帯域幅	100 Mbps
GROUP メッセージ送信間隔	0.1 秒，1.0 秒，10.0 秒
ケース II：問題特定性能評価	
ノード数	2,791
制御パケットサイズ	1,024 バイト
共有情報パケットサイズ	1,024 バイト/ノード
リンク遅延	10.0–20.0 ミリ秒
帯域幅	0.1–10.0 Gbps
GROUP メッセージ送信間隔	0.1 秒
グループサイズ制限値	10
通知グループサイズ	5
攻撃トラフィック	1.6 Mbps/ノード
グループ形成閾値	5.0 Mbps

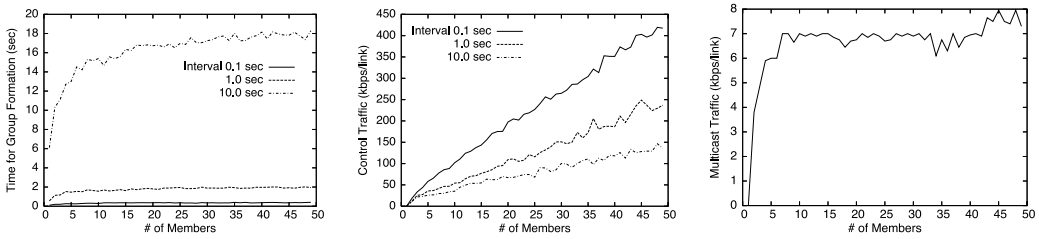
4.1 グループ形成と通信コスト

4.1.1 グループ形成と通信コスト評価の設定

基本性能を評価するため，トポロジ生成ツール tiers[14] を用いてネットワークトポロジを生成した（**表 3**）．センサエージェントが問題を検出した際に *GROUP* を送信する間隔は 0.1 秒，1.0 秒，10.0 秒の 3 つとした．また，提案方式の制御パケットサイズは 1,024 バイトとした．ただし，制御パケットのうちメンバリストを含むものは，そのメンバリストのサイズに比例した大きさのパケットサイズとした．**formGroup** の引数 *cond* は *true* に設定し，いっせいにグループ形成を開始させた．以上のような設定で，グループのメンバ数を 1 から 50 まで変化させ，それぞれの値を計測した．

4.1.2 グループ形成とグループ通信コスト

図 6 (a) は形成されるグループサイズと，グループ



(a) グループ形成時間 (b) 制御トラフィック (c) グループ通信トラフィック

図 6 基本性能のシミュレーション結果（20 回シミュレーションを行った平均値）
Fig. 6 Simulation results of basic performance (average of 20 simulations).

形成にかかる時間を示している．*GROUP* の送信間隔が長くなるにつれて，グループ形成にかかる時間が増加している．グループは分散並列的に形成されていくため，このグループ形成にかかる時間はグループサイズを N として，$\log N$ に比例する．また，この時間は *GROUP* の送信間隔に大きく依存している．この結果から，迅速な問題範囲の特定を行うために，*GROUP* の送信間隔は，0.1 秒〜1.0 秒が適切であることが分かる．

図 6 (b) はグループ形成時における 1 メンバあたりの最大制御トラフィックがグループメンバ数に対してどのように変化するかを評価した結果である．このグラフから，*GROUP* の送信間隔が短いほど，グループ形成時の制御トラフィックは増大することが分かる．メンバ数に比例して制御トラフィックも増加するが，*GROUP* の送信間隔が 0.1 秒と非常に短い場合でさえ，グループメンバ数が 10 程度までは，十分に小さい（100 kbps 以下）制御トラフィックといえる．

図 6 (c) は，1 つのマスタからグループ内の全メンバに対してデータがマルチキャストされるときの，1 リンクあたりのトラフィックを表している．このグラフから，1 リンクあたりのグループ通信トラフィックはグループメンバ数に関係なくほぼ同じであることが分かる．マルチキャストされるデータは各リンクにおいてたかだか 2 回しか送信されないため，グループ通信トラフィックは，グループメンバ数に依存しない．

4.2 問題特定性能評価

4.2.1 問題特定性能評価の設定

問題特定性能を評価するため，学術情報ネットワーク SINET [15] のトポロジを再現し，SINET 上の各ノードに 40 ノードが接続されているものとして，約 2,800 ノードでシミュレーションを行った．詳細な設定は表 3 のとおりである．このような設定の下で，UDP パケットを約 500 ノードからランダムに選択された一定数の攻撃対象ノードに対して送りつけることで，DDoS 攻撃により輻輳が頻繁に発生する状況を再現した．攻撃を行うノード（攻撃ノード）は葉となるノードからランダムに定め，攻撃を開始する時刻は 1.0 秒から 30.0 秒の間でランダムに定めた．センサエージェントを SINET 上の全ノードに相当する 71 ノードに分散配置し，それぞれの隣接エージェントは SINET 上で物理的に 1 ホップで到達可能なエージェントとした．

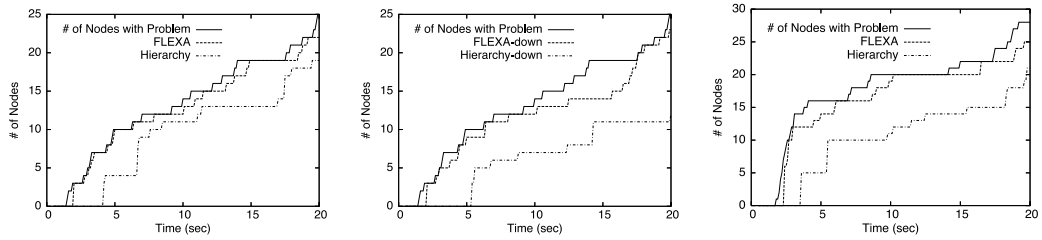
エージェントに与えた監視シナリオは，特定の宛先へのトラフィック量が指定した閾値（5.0 Mbps）を超えた場合に，隣接エージェントと自律的にグループを形成し，グループサイズが指定したサイズ（通知サイズ）を超えた場合には，そのグループメンバのアドレスリストを管理用エージェントへ送信する，という動作を行うよう記述されている．*GROUP* の送信間隔は予備実験により，0.1 秒，グループサイズの制限値は 10，通知サイズは 5 とした．

また，この測定結果と階層型手法を比較した．階層型手法では，静的な木が物理的な隣接関係に基づき構築され，各サブツリーの親ノード（中間管理ノード）がその子ノード（センサノード）から受信した問題範囲の情報が，指定された通知サイズを超えた場合に，その問題範囲の情報は木に沿って即座に管理者へ送信される（通知サイズは提案方式と同様に 5 とした）．提案方式との公平性を保つため，各センサノードは 0.1 秒ごとに特定の宛先へのトラフィック量が指定した閾値（5.0 Mbps）を超えているかどうか確認するものとした．

DDoS 攻撃発生時には，しばしばノード障害が発生する．そこで，いくつかのノードに障害が発生し，通信不能となる状況を再現した．具体的には，71 個のセンサエージェントからランダムに 4 つのノードを選び，それらはパケットの送受信を行うことができないようにして，ノード障害を発生させている．

4.2.2 障害発生時における性能評価

図 7 (a)，(b) はそれぞれ，ノード障害がない場合



(a) ノード障害なし (b) ノード障害あり (c) 論理的な隣接関係あり

図 7 問題特定性能のシミュレーション結果
Fig. 7 Simulation results of detecting problematic areas.

とある場合の，センサエージェント/センサノードからの情報を基に管理者が問題範囲を特定するまでの時間を示している．ノード障害がない場合は，提案手法（FLEXA）と階層型手法（Hierarchy）のどちらも問題範囲を特定することができている．しかしグラフから分かるように，階層型手法で問題範囲を特定するには，FLEXA よりもずっと多くの時間がかかる．たとえば 5 秒の時点で，10 ノードで問題が発生しており，FLEXA はこれらの問題範囲をほぼ実時間で特定できているが，階層型手法では約 7.5 秒の時点まで特定できていない．これは，階層型手法は物理的な隣接関係の一部を木として使うが，提案方式はすべての物理的な隣接関係を使って柔軟にグループ形成を行うためである．

ノード障害がある場合，いくつかの中間管理ノードに障害が発生することがあるために，階層型手法（Hierarchy-down）は管理者がすべての問題範囲を特定できない．一方，FLEXA（FLEXA-down）では通信可能な任意のエージェントが情報を送信するため，ノード障害がない場合と同程度の時間で管理者がすべての問題範囲を特定できている．階層型手法でも，バックアップサーバなどを導入することで，中間管理ノードの障害にも対応することができるが，提案方式ではそのようなサーバを設置する必要がなく，効果的に問題範囲を特定可能である．

4.2.3 論理的な隣接関係による性能評価

FLEXA では物理的な隣接関係によらず，論理的な隣接関係を定義することもできる．ここでは SINET 上のいくつかのノードが，巨大なデータベースを公開しているといった状況を想定し，それらを論理的な隣接関係にあると定義して実験を行った（図 7 (c)）．提案方式では，管理者が問題範囲のほぼすべてを迅速に特定できており，攻撃の初期段階でさえ，迅速な問題範囲の特定ができている．一方，階層型手法では数秒の後れがある．階層型手法では，物理的に隣接していない複数のネットワークセグメントに対して，同時に攻撃が行われた場合はシステム全体としては迅速に問題範囲を特定できず，管理者は通常時に定期的に行われる情報送信を待たなければならない．提案方式は物理的な隣接関係と論理的な隣接関係の両方を使って，同種の問題が発生しているノードどうしで自律的にグループを形成する．このため，問題範囲を特定するための時間が短く，迅速な対応が可能である．

5. おわりに

本研究では，自律的なグループ形成機構を用いた分散ネットワークモニタの実現方式を提案した．提案方式では，各ネットワークセグメントを監視するためのエージェントを分散配置し，監視シナリオを与えることで自律的に監視を行うことができる．

実験から，提案方式がノード数 3,000 程度のネットワークに設置された 71 個のエージェントにより，十分小さい制御トラフィックで，問題範囲を数秒以内で特定可能であり，階層型手法よりも迅速な問題範囲の特定ができることを確認した．

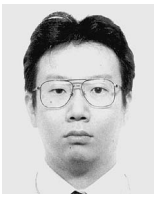
参考文献

1) Garber, L.: Denial-of-Service Attacks Rip the Internet, *IEEE Computer*, Vol.33, No.4, pp.12–17 (2000).
2) Schuba, C., Krsul, I., Kuhn, M., Spafford, E., Sundaram, A. and Zamboni, D.: Analysis of a Denial of Service Attack on TCP, *Proc. IEEE Symposium on Security and Privacy*, pp.208–223 (1997).
3) Moore, D., Voelker, G.M. and Savage, S.: Inferring Internet Denial-of-Service Activity, *Proc. USENIX Security Symposium*, pp.9–22 (2001).
4) Su, M., Thulasiraman, K. and Das, A.: A scalable on-line multilevel distributed network fault detection/monitoring system based on the

SNMP protocol, *Proc. IEEE Global Telecommunications Conference*, pp.1971–1975 (2002).
5) Gavalas, D., Greenwood, D., Ghanbari, M. and O'Mahony, O.: Hierarchical network management: A scalable and dynamic mobile agent-based approach, *Computer Networks*, Vol.38, No.6, pp.693–711 (2002).
6) Liotta, A., Pavlou, G. and Knight, G.: Exploiting Agent Mobility for Large Scale Network Monitoring, *IEEE Network*, Vol.16, No.3, pp.7–15 (2002).
7) Zapf, M., Herrmann, K. and Geihs, K.: Decentralized SNMP management with mobile agents, *Proc. 6th IFIP/IEEE International Symposium on Distributed Management for the Networked Millennium*, pp.623–635 (1999).
8) Dilman, M. and Raz, D.: Efficient Reactive Monitoring, *IEEE Journal on Selected Areas in Communications*, Vol.20, No.4, pp.668–676 (2002).
9) Snort.org. http://www.snort.org/
10) Cabrera, J.B.D., Lewis, L., Qin, X., Lee, W., Prasanth, R.K., Ravichandran, B. and Mehra, R.K.: Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables — A Feasibility Study, *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, pp.609–622 (2001).
11) Hajji, H.: Baselining Network Traffic and Online Faults Detection, *Proc. IEEE International Conference on Communications*, pp.301–308 (2003).
12) 梅津高朗，安本慶一，中田明夫，東野輝夫：マルチランデブに基づくグループ通信機能を提供する Java ミドルウェアの提案，情報処理学会論文誌，Vol.45, No.11, pp.2519–2527 (2004).
13) Tanenbaum, A.S. and van Steen, M.: *Distributed Systems: Principles and Paradigms*, Prentice Hall (2002).
14) Tiers Topology Generator. http://www.isi.edu/nsnam/ns/ns-topogen.html#tiers
15) SINET. http://www.sinet.ad.jp/english/

(平成 17 年 5 月 23 日受付)
(平成 17 年 11 月 1 日採録)

内山　彰（学生会員）
平成 17 年大阪大学大学院情報科学研究科情報ネットワーク学系専攻博士前期課程修了．同年同大学院博士後期課程進学．ネットワークセキュリティやアドホックネットワークの研究に従事．IEEE 会員．



梅津　高朗（正会員）
平成 13 年大阪大学大学院基礎工学研究科情報数理系専攻博士前期課程修了．同年同大学院博士後期課程進学．平成 14 年同大学院博士後期課程退学後，同大学院情報科学研究科助手．博士（情報科学）．アドホックネットワーク用ミドルウェアや開発環境の研究に従事．

安本　慶一（正会員）
平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院基礎工学研究科博士後期課程退学後，滋賀大学経済学部助手．平成 14 年より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS, ACM 各会員．

東野　輝夫（正会員）
昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院基礎工学研究科博士課程修了．同年同大学助手．現在，同大学大学院情報科学研究科教授，工学博士．分散システム，通信プロトコル，モバイルコンピューティング等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．