システムソフトウェアと
オペレーティング・システム 77－28
マルチメディア通信と分散処理 87－28
（1998. 2. 27）

リフレクタエージェントを用いた自律組織間診断システム

明石 修　菅原 俊治　村上 健一郎　丸山 充　高橋 直久
{akashi,sugawara,murakami,mitsuru,naohisa} @core.ntt.co.jp
NTT 光ネットワークシステム研究所

概要

自律システム (AS) がインターネットにアドバタイズした経路情報の振舞を理解し，自組織の意図が正しく反映されて伝わっていることを検証することは重要である．この機能を実現するため，他の AS からの視点で情報を収集するリフレクタモデルに基づいて，協調して動作するマルチエージェントによる AS 間障害診断システムを提案する．本システムでは，各 AS に観測・診断機能を持ったエージェントを配置し，他の AS 中のエージェントと協調してデータを交換することにより，ネットワークの障害を検出し，その原因を解析する．

An Inter-AS Diagnostic System by Cooperative Reflector Agents

Osamu Akashi, Toshiharu Sugawara, Ken-ichiro Murakami,
Mitsuru Maruyama, and Naohisa Takahashi
{akashi,sugawara,murakami,mitsuru,naohisa}@core.ntt.co.jp
NTT Optical Network Systems Laboratories

abstract

This paper proposes a network diagnostic system using the *reflector model* to infer the dynamics of the Internet. It provides essential functions that an autonomous system (AS) can view how the routing information about the AS spread in the Internet and can diagnose anomalies. To achieve these functions, the system is constructed using agents located in multiple ASs. Each agent monitors the routing information and reports anomalies based on diagnostic knowledge. These agents can cooperate autonomously to identify problems through observation from multiple viewpoints.

1 はじめに

多くの自律システム (Autonomous System,AS) を相互接続するインターネットにおいて，自組織がアドバタイズした経路情報が正しく意図を反映して伝わっているかどうかを確認することは重要である．しかしながら，今日のインターネットでは，BGP[1] による AS 間経路情報は不安定であり [2, 3]，障害の早期発見や原因の特定は困難である．それは，インターネットが多くの AS から構成されることによる複雑さに加えて，以下のような経路情報制御自体が持つ性質が，その観測を困難にしている．

- ある AS がアドバタイズした経路情報は，各 AS において各々のポリシーに基づき経路選択などの情報の加工が行なわれ伝搬される．そのため，同じオリジンでアドバタイズした

情報が，AS毎に(空間的に)変化する．

- 経路の選択とアドバタイズには，静的な選択規則のみでなく，動的なパラメータである物理的な回線やルータの障害，他のプロトコルとの相互作用が影響するため，経路情報は時間的に変動する．

本研究の目的は，インターネットの経路情報の動的な変化を観測することによりその振舞を理解し，自ASの経路が正しく意図を反映して伝搬していることを検証し，障害があれば，それを検知，解析することが可能なAS間診断システムを提案することである．

経路情報の振舞を理解するには，前述した性質のため，他のASにおいて経路情報を定常観測し，その解析を行なう必要がある．また障害原因解析のためには，エキスパートの診断知識をシステムが持つ必要がある．

従来WEBで提供されていた“looking glass”[4]や“traceroute server”は，経路情報を自AS以外の地点で観測可能とするが，特定の情報をオペレータが取得するのを前提としたツールであり，トラフィックや負荷集中の観点からも定常観測への適用は困難である．GDT[5]のように，リソースの依存関係から順次上流のエージェントに調査依頼をして，自動診断を行なうアプローチもあるが，経路情報自体の障害を診断対象とする場合は，依存関係が構築できないという問題がある．

本稿では，上記問題を解決するため，次のような方式を実現したマルチエージェントによるAS間診断システム **ENCORE**(Inter-AS Diagnostic System by Cooperative Reflector Agents) を提案する．

1. 診断知識と観測機能を持った知的エージェントを各ASに配置し，観測と解析はローカルに行なう．他のASの情報の取得方法は，エージェントが自分自身の状態を映す鏡として動作するリフレクタモデル[6]に基づく．本システムでは，必要に応じて協調して観測結果を交換し，障害の通知，解析を行なうことにより，システム内のトラフィックを押さえる．
2. エージェントの持つ診断知識を，ENCOREの提供する仮説検証フレームワークの中で記述することができるので，システムの内部処理とは独立に，エキスパートの知識を容易に追加，変更することが可能である．
3. エージェントは，AS毎のルータの違いなどを吸収し，統一的なインタフェースを与える．
4. 各エージェントに，観測戦略に従って観測の頻度，密度を切り換える機能を導入して，効率化をする．例えば，観測負荷の軽い大域的なデータから障害の発生を予測する検証仮説に従い，次の観測や診断をプランニングする．
5. タビュレーション技法[8]を用いた実行機構を提供し，適応型の実行動作制御を行なうことにより，効率化を行なう．

本稿では，最初にリフレクタモデルに関して説明する．次に，マルチエージェントを用いた診断システムを構築する際の要求条件をまとめ，ENCOREのシステム構成と，知識処理アーキテクチャに関して述べる．最後に，具体的な事例を用いて診断動作を説明する．

2 リフレクタモデル

インターネットの経路情報の動的な変化を観測し，自分のアドバタイズした経路が意図を正しく反映していることを検証するため，他のASから経路情報を取得するリフレクタモデル[6]がある．このモデルでは，各AS毎にエージェントが存在し，他のエージェントからの依頼で，自ASから観測した依頼元のASに関するさまざまな情報を，依頼元に送り返す．

図1に，リフレクタモデルの例を示す．この例では，AS_{self} が，AS_1 と AS_3 を通じて，経路情報をアドバタイズしている．AS_{self} のリフレクタエージェント R_{self} は，自AS中に存在する経路への経路情報が，インターネット中で正しく自らの意図を反映していることを検証するため，各ASに存在するリフレクタに依頼し，その経路に関する情報を観測し送り返してもらう．このとき，他のASのエージェントは自分自身の状態を映す鏡(リフレクタ)として動作する．

AS_1 と AS_2 の間に回線障害があり直接該当ASのリフレクタ R_2 にアクセスできない場合，AS_4 のリフレクタ R_4 がメッセージを中継することにより，AS_4 がIPレベルではトランジットしないASである場合でも，情報が得られる．

また問題の切分けのみでなく，障害の早期発見，通知のフレームワークでもある．これは，経路情報の障害は，アドバタイズしたASよりも，それを受けとる他のASがその問題に気付きやすく，

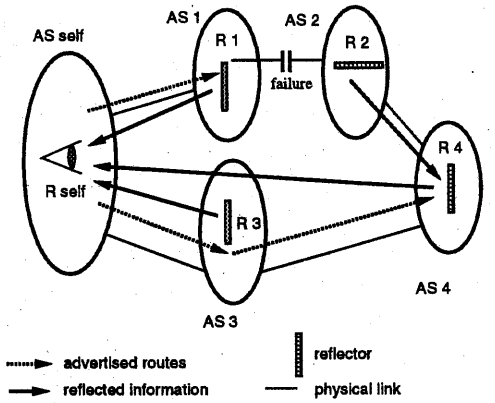

図 1: リフレクタモデルの例

障害 (またはその傾向) を検知した場合，該当する AS，あるいは他の AS のエージェントに通知する．

3 AS 間診断システム ENCORE

3.1 要求条件

ENCORE における知識を持ったリフレクタエージェントは，知的な鏡として動作する．すなわち，他のエージェントからのデータの要求に対しては，独自の観測手段を用いて必要な情報を収集し，必要に応じて，統計処理や他のエージェントとの更なる協調動作をし，求められた結果を返す．これらの動作を実現するためには，ネットワークに関するエキスパートの知識が必要である．またこの知識は，柔軟に追加・変更が可能であるように，システム実行部分組込みではなく，別の構成要素として記述可能である必要がある．

また動的に変化するネットワークに関する診断知識を表現するために，その AS 固有のローカルな事象や，経路数などの動的に変化する値，またそのパラメータの変動の傾向を表す値を表記可能とし，システムが値の処理をサポートする必要がある．これらの値は，診断知識にパラメータとしとして与え，診断知識に反映する．

診断のために必要な，ICMP を使った診断ツールは，代替の手段を柔軟に用いられるように，実装に依存しないようにする．すなわち，実際のツールを呼出し，その結果を予め定めた共通フォーマットに変換する関数と，ツールの実行結果を判断する関数に分ける必要がある．

観測負荷の観点からは，比較的負荷の軽い大域的なデータや，統計データから，障害の予想をし，

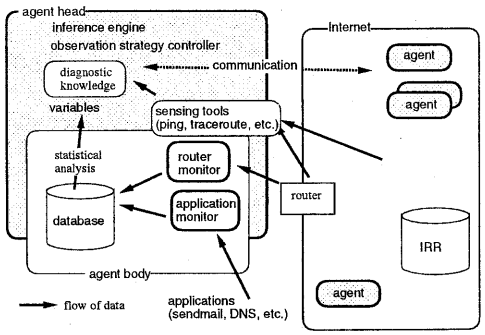

図 2: システムの構造

仮説を絞り込みながら，観測負荷の高い観測や診断は効率的に実行することが必要である．逆に，ある診断結果から，早期に警報を出し，一時的にある程度負荷のかかる観測を新たに実行したり，他のエージェントに観測依頼を出したりする．このように，常に詳細な観測をすることは困難であるので，観測戦略の切換機能と，その切換戦略が柔軟に変更可能である必要がある．

また，知識に基づいたさまざまな診断ルールをプランニングする場合，そのコストやあらかじめ与えた可能性のみでなく，過去の実行履歴に基づきローカルな環境に適応するようにフィードバックする機能も重要である．

3.2 システム構成

ENCORE のシステム構成を図 2に示す．ENCORE のエージェントは，診断のための知識処理を行なうヘッダ部と，経路情報の観測や診断のためのツール，データベース等からなるボディ部で構成する．ヘッダ部とボディ部を分割するのは，特定のツールや観測手段の実装，得られる統計データといった環境独自の値と，共通な知識を分離することにより，ヘッダ部をさまざまな環境で共通に用いることができるようにするためである．ボディ部は，共通インターフェース層以下では，その AS 毎に独自のツール，実装を用いることができる．

ヘッダ部は，以下の部分から成る．

- 推論エンジン・・・観測された事象から仮説を取りだし，順次その検証のためのルールをプランニングし，実行する
- 観測制御モジュール・・・観測動作の実行と，その観測戦略の選択・切換えを指示する

- 診断知識，戦略の記述
  - 診断知識は，特定の事象に対する複数の仮説を記述し，それぞれ検証するためのルールを含む．仮説は，更に詳細なレベルの複数の仮説を含むことが可能．
  - 戦略は，実行する複数の観測ルールとその時間間隔，結果の値を解釈して動作する関数から成る．
  - 環境毎に異なる情報や，観測結果を統計処理した結果得られる，時間的に変動する変数を持つ．

変数は，知識の記述に使われ，値の更新方法に関してはヘッダ部で記述し，ボディ部で値が生成される．観測するデータの例として，そのASで観測される全経路数がある．変数の値は動的に変わるパラメータであり，AS毎の環境により，また時間の経過と共に変化するため，定常的に観測しないと正確に求められない．また，単なる経路数の差分のみでなく，経路数の変動の周期も考慮することにより，異常状態の切分けの精度を上げることが可能であるが，この周期を表す値を求めるためには，継続した経路数の観測と統計処理が必要である．

ヘッダ部は，あらかじめ定めたインタフェースに従って，ボディ部のツールを使用し，データを参照する．また必要に応じて他のエージェントと会話，協調を行なう機能を持つ．

ボディ部は，以下の要素から構成する．

- 診断ツール・・・ICMPを使ったツール，IRR[7]データベースへの問い合わせ機能
- 監視プロセス
  - ルータモニタ・・・BGPの経路情報の変動を監視し，統計処理したデータを提供
  - アプリケーションモニタ・・・DNSやsendmailの出力するログデータを監視

監視プロセスの機能は，観測戦略によって記述することも可能である．しかし典型的な監視作業を独立したプロセスとすることにより，リフレクタエージェントの観測戦略記述，プランニング，タイムアウト処理の記述を簡潔にすることができる．これらのプロセスは，観測制御モジュールから，データ取得，観測等の処理要求を受け，統計処理をして観測結果を返す．観測結果に対する判断，すなわち特定の事象に関する警報イベントの生成等は，観測制御モジュールで判断する．

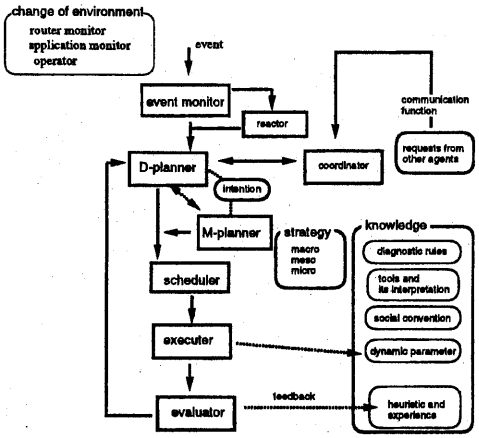

図3: エージェントアーキテクチャ

4 ENCOREにおける診断処理

4.1 アーキテクチャの概要

エージェントの知識処理アーキテクチャを図3に示す．エージェントの知識処理部は，診断知識，観測戦略を記述した静的データ部と，動的に環境から取得した動的データを知識として持つ．エージェントはこの知識に基づいて，自律的に行動する．外部からエージェントへのメッセージは，イベントモニタが受付ける．

プランナは，実行可能な複数のルールの中から，意図(intention)に基づき実行するルールを選択する．スケジューラにより，最終的に直列化されたルールは，executerにより順次実行され，evaluatorによりその結果が履歴データとして知識に反映される．他のエージェントとの協調は，coordinatorが仲介する．

4.2 プランニング

エージェントは，診断知識と観測戦略という性質の異なる2つの知識を用いた動作を明確に記述できるように，プランナを診断プランナ(D-planner)と観測戦略プランナ(M-planner)と呼ぶ2つの協調型プランナに分割して構成している．また定常的な観測を効率よく実現する戦略を与えられるように，負荷に従って観測戦略を3レベルに分類し，観測戦略に従って観測の頻度，密度を切替える機能を機能を与えている．またM-plannerが扱う観測戦略はシステム組込みではなく，診断知識記述と同様のフレームワークを提供することにより，容易に新たな戦略の追加，変更が可能である．こ

の機能はAS毎のローカルな環境を管理者が反映させるために，重要である．

診断のための推論エンジンは，オペレータ，あるいは内部の観測制御モジュールからの通知イベントにより起動される．診断規則には，想定される複数の仮説とそれを検証するためのルールを記述する．ルールは主に，外部から必要なデータを取得するための値取得関数とその仮引数(値は実行時に束縛される)，取得したデータを引数にその中味を解釈し，最終的にルール全体の真偽を決定する評価関数からなる．診断順序は，D-plannerが，半順序で記述された仮説，ルールのツリーを取り出し，実行コストや過去の履歴を参照して実行ルールを選択する．ルールが実行されることにより真偽が確定すると，D-plannerは可能性を絞り込み，次の仮説の検証のためのルールをプランニングする．

他のエージェントとの協調が必要なルールが存在する場合は，coordinatorモジュールが仲介する．協調動作を実現するためには検証ルールの中で，ローカルに実行可能か，他に依頼すべきかを記述して，実行時に他のリフレクタに要求メッセージを送信する．メッセージの種類は，相手に要求する`request`と，その返事の`reply`，そして異常を検知した場合に通知する`inform`から成る．現実装では，協調の相手は知識として静的に記述しているが，動的に相手を捜すプロトコルと組合わせることによりrobustに拡張可能である．

一方，M-plannerは，タイマーにより起動される動作を扱う．これは，記述された観測戦略，あるいはD-plannerからの要求に基づき，記述された時間間隔に従って，観測するためのルールをプランニングする．

観測戦略の代表例として，変数として表されるパラメータの処理がある．この処理の記述では，パラメータ値を表すデータベースオブジェクトを定義し，初期化手段，値取得関数，データ更新頻度と統計処理関数を指定する．各々の戦略には，更新した値が，閾値を超えた場合に，実行する関数が定義可能であり，この関数内で，M-plannerに診断要求を送り，より詳細なレベルの観測戦略を実行する．戦略は，以下の3段階を切り換える．

- 比較的観測負荷の軽い大域的なデータの観測 ··· マクロ戦略．例) 全経路数とその変動周期
- ある特定の性質を持つデータの観測 ··· メソ戦略．例) 特定ASをオリジンとする経路数
- 各経路毎の詳細な観測 ··· ミクロ戦略

2つのプランナは，その実行結果により，相互に影響を及ぼしあう．すなわち，定期観測によりある(異常と思われる)事象を観測した場合は，M-plannerはD-plannerに観測状況をパラメータとして渡し，診断を依頼する．逆に，ある別のイベントで実行された診断の結果，D-plannerがより細かいレベルでの統計観測データを取ることを必要と判断した場合，M-plannerに依頼する．

4.3 ルールの実行

D-planner/M-plannerからの出力であるルールの集合の実行に際して，タビュレーション技法[8]を用いた実行機構を提供し，適応型の実行動作制御を行なう．

スケジューラは，2つのプランナからのルールをマージして実行順序を定め，executerに指示する．executerは，値取得関数による値の生成をタビュレーション技法を用いて，効率化を行なう．これは，仮説検証処理の中で実行時に値取得関数の引数に値を束縛し実行するが，値取得関数は外部のルータやモニタへの問合せを行なうコストの高い関数であること，その結果得られる値が仮説検証における多くのルールに共通に含まれることから，特にENCOREのような診断処理システムには有効である．例えば，自ASのルータへの問合せによりBGPデータのサマリーを取得し，その結果得られるBGP peerのアドレス，状態，全BGPエントリ数などは多くのルールにおいて参照される値であり，その有効利用は重要である．タビュレーションで用いる表の値は、有効時間を設定し，executerが値の消去処理を行う．

evaluatorは，ルールの実行時間や，タイムアウト事象，最終的な仮説の検証結果を，ローカルな知識として蓄え，これをプランナが参照することによりフィードバック機能を実現する．変数パラメータとフィードバック機能により，エージェントは環境に適応するように振舞を変える．

エージェントの知識処理部分は主にCommon Lisp，コミュニケーションとツールのテキスト処理部分は主にCとPerlを用いて記述し，UNIX上で開発を行なっている．現在，システム基本部分の動作は確認し，実際の診断知識/観測戦略の記述を通してその検証を行なっている．

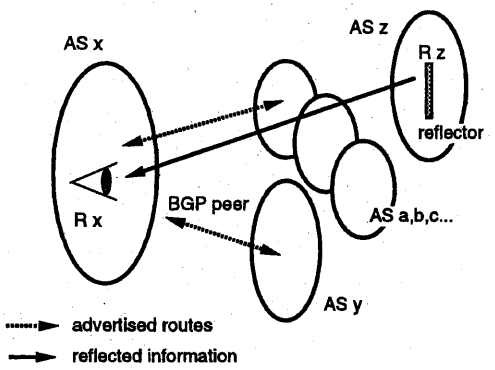

図 4: 診断動作例

4.4 診断例

本節では，図4ような障害を例にし，システムの動作を説明する．この例では，AS_x と AS_y は BGP peer であり，AS_y からアドバタイズされた経路は AS_z に伝わっている．この障害例では，AS_y は AS_x に通知せず，突然自 AS 番号を変えてしまった．この場合の，動作は以下の通りである．

1. AS_x のルートモニタは，経路エントリの突然の減少を観測する (= マクロレベル)．過去の AS 毎の経路数の記録と比較することにより，減少した経路はある特定の AS(=AS_y) からの経路であることが，判明する (= メソレベル)．モニタは，警報メッセージを自 AS のリフレクタ R_x に送る．
2. R_x は，仮説リストから，対応する複数の仮説を取りだし，順次検証するためのルールをプランニング，実行する．この場合，最初に AS_y のボーダルータへの到着性を検証するが，AS_x と AS_y の間の TCP/IP レベルの障害は検出されない．また，AS_x において，最近ルータの設定が変えられていないことも確認する．

この例では，説明のため，AS_y にはリフレクタエージェントは存在しない，として説明を行なう．もし存在した場合は，このエージェントとの協調により，仮説の検証は容易である．

3. 自 AS 内の情報のみでは，もはやこれ以上の解析は不可能であるため，R_x は，他の AS 中のリフレクタ R_z と協調して，診断を行なう．具体的には，AS_x 中の経路 E_x，AS_y 中の経路 E_y に関する BGP 情報を調査するよう依頼する．(ミクロレベルの観測)
4. R_z は，依頼された調査を行ない，R_x に以下の結論を送る．
   - AS_x をオリジンとし，AS_y を経由して，アドバタイズされた経路 E_x は AS_z の経路表にはない．
   - 経路 E_y は，AS_z の経路表に存在する．しかし，オリジン AS 番号は，$AS_{y'}$ に変わっている．
5. R_x は R_z からの報告を受けとり，R_x は AS_y の AS 番号が $AS_{y'}$ に変わったのではないかと，推論する．

5 おわりに

自 AS の経路が正しく意図を反映していることを検証し，障害があれば，それを検知，解析することが可能な AS 間診断システムとして，エキスパートの知識を組込んだマルチエージェントシステム ENCORE を提案した．今後は，診断知識の検証を進め，実環境での実験を行っていく予定である．

参考文献

[1] Y. Rekhter and T. Li. "A Border Gateway Protocol 4 (BGP-4)", 1995. RFC1771.

[2] C. Labovitz, G.R. Malan, and F. Jahanian. "Internet Routing Instability". In *Proc. of ACM SIGCOMM*, 1997.

[3] Vern Paxson. "End-to-End Behavior in the Internet". In *Proc. of ACM SIGCOMM*, 1996.

[4] Ed Kern. http://nitrous.digex.net.

[5] G. Thaler D and C.V. Ravishankar. "An Architecture for Inter-Domain Troubleshooting". In *Proc. of IEEE ICCCN*, 1997.

[6] 村上健一郎. "リフレクタモデルに基づくインターネット管理アーキテクチャ". 第 52 回情報処理学会全国大会, volume 1, pp. 121–122, 1997.

[7] Routing Arbiter Project. http://www.ra.net/-RADB.tools.docs/overview.html.

[8] R.S. Bird. Tabulation Techniques for Recursive Programs. In *ACM Computing Surveys*, pp. 403–417, 1980.

[9] G.M.P. O'hare and N.R. Jennings. *"Foundations of Distributed Artificial Intelligence"*. Wiley-Interscience, 1996.