ネットワーク障害伝播モデルの自動学習機能

1K－2

堀川 健一　桑原 教彰　夏目 晃宏　吉江 信夫
住友電気工業（株）システムエレクトロニクス研究開発センター

1　はじめに

筆者らは，イベント相関[1]の技術を用いて，ネットワーク管理ソフトウェアに報告される障害イベントの集合から，障害の原因を自動的に特定することができるシステムを開発している[2]．イベント相関技術では，ネットワークで観測される障害イベントの集合とその障害原因の対応表（表１）を用いて，実際に観測された障害イベントのパターンから障害の原因を推論する．対応表は，障害の因果関係（伝播モデル[3]）を記述すれば自動的に生成されるが，最初から完全な伝播モデルを用意することは難しい．そこで，障害イベントのログから自動的に因果関係を検出し，それを伝播モデルに反映させるメカニズムを開発した．

2　イベント相関技術を用いたネットワーク障害管理

ここでイベント相関について説明する．イベント相関とは，例えば 障害原因，P1，P2，P3と，障害イベント，S1，S2，S3，S4に対して，(1) P1が発生するとS1が発生する，(2) P2が発生するとS3が発生する，(3) P3が発生するとS2，S3，S4が発生するというルールが存在する場合，表１に示すような対応表を生成する．そして障害イベントの集合として入力されるS1,S2,S3,S4と対応表の各列との距離を計算する．通常は，原因候補とするか否かを決める閾値を設けておき，この閾値以内の距離にある障害原因を近い順に，原因候補として挙げる．距離としてはハミング距離が用いられる．

表１．対応表の例

	P1	P2	P3
S1	1	0	0
S2	0	0	1
S3	0	1	1
S4	0	0	1

本システムでは，イベント相関器として SMARTS 社製の InCharge を用いた．筆者らは，NetMate[4]の基底クラスを継承したクラスを用いて，InCharge 上にネットワークの管理モデルの定義を行った．この管理モデル上では，管理対象を因果関係の記述しやすいようにいくつかのインスタンスで表現する．このインスタンスのクラスには，TCP の接続（TcpTransportLink クラス），TCP の接続の末端（TcpTransportNode クラス），IP アドレスを持っているもの（IpNetworkNode），Ethernet（EtherLanLink），Ethernet の口（EtherLanNode），ソフトウェア（Server, Client, Manager），筐体（PhysicalService）などが定義されている．これらのクラスのインスタンスを生成し，インスタンス間に接続関係，レイヤの上下関係，筐体などへの包含関係などを設定することで，実際の構成情報が表現される．

次に，各クラスでの障害原因と障害イベントとの間の因果関係のルールを記述する．また，あるクラスの障害イベントは別のクラスの障害原因として伝播することがある．このような障害の伝播の記述も行う．これらを**伝播モデル**と呼ぶ．InCharge は，実際のトポロジが与えられると，各クラスのインスタンスを生成し，その際に伝播モデルをインスタンスレベルに展開して，表１のような対応表を生成する．

しかし，ネットワークの具体的な構成によって障害の発生の傾向などが異なることから，最初から完全な伝播モデルを用意することは難しいため，そのようなネットワークの特性に適応して，伝播モデルを自動的に検出し，修正するメカニズムが必要である．以下ではその手法について説明する．

3　因果関係の検出と伝播モデルへの反映

因果関係の検出は，一定のタイムウインドウ T 毎に自動的に行われる．また，利用者が指定した範囲の時間に対して検出することも可能である．

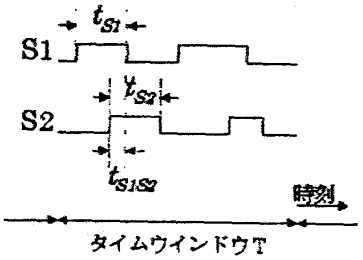

図１．障害イベントの発生

２つ障害イベントの S1, S2（図１）について
t_{S1}：S1 が発生していた時間
t_{S2}：S2 が発生していた時間

Autonomous Learning of Propagation Model for Network Fault Isolation
Ken'ichi Horikawa, Noriaki Kuwahara,
Akihiro Natsume and Nobuo Yoshie
Systems and Electronics R & D Center,
Sumitomo Electric Industries, Ltd.
1-1-3 Shimaya, Konohana, Osaka City, 554, Japan
E-mail: horikawa@rcom.sei.co.jp

t_{S1S2}：S1,S2 が共に発生していた時間

とすると，タイムウインドウ T 内での S1, S2 の間の相関の度合い $C(S1,S2)$ は以下の式(1)で計算される．$\sum$ は，T 内での和を表す．

$$C(S1,S2)=\frac{\sum t_{S1S2}}{\sqrt{\sum t_{S1}}\sqrt{\sum t_{S2}}} \quad (1)$$

式(1) は，2 つの時系列データの間の相互相関を表す量であり，0 から 1 までの値を取り，値が大きいほど相関が強いことを示す[5]．この値を T 内に発生した障害イベント間で計算し，値が閾値を超えるものが検出された因果関係とする．

検出された因果関係の内，既に伝播モデルに含まれていないものを伝播モデルに追加して対応表に反映する．現時点では，直接関係を持たない 2 クラス間の伝播は記述の作成が困難なため反映しない実装となっている．

4 実施例

図 2 は，シミュレーションにより TCP 接続の末端と Ethernet を表現するクラスのインスタンスに障害イベントを発生させた上で，因果関係の検出機能を実行した画面である．画面上側のリスト（Targets）で利用者が着目する障害イベント（ELLManyFragments, ELLManayJabbers）を 2 つ指定して，因果関係の検出を実行すると，画面下側のリスト（Detected Causalities）にこれらのイベントに関係する 3 対の因果関係が表示される．現時点では，伝播の向きの判定が困難なため，因果関係のある障害イベントのペアが見つかる毎に 2 つの因果関係が生成される．

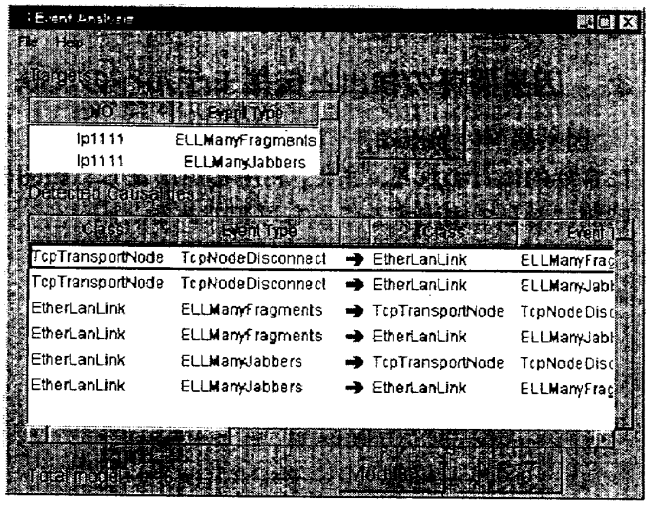

図2. 因果関係の検出の結果

利用者は，図 2 の画面で候補の中から追加したい因果関係として，TcpNodeDisconnet → ELLManyFragments を選択すると，1) 伝播モデルの追加分を既存のクラスのサブクラスとして定義した記述ファイル（リスト 1）が生成され，2) このファイルをコンパイルして作成されるライブラリを動作中の InCharge のエンジンにロードすることで反映が完了する．

```
// automatically generated by SituationManager
#include <limemodel.mdl>
#pragma include_h <limemodel.h>
#pragma include_c <limemodel2.h>

traced interface SUB_TcpTransportNode
        : TcpTransportNode {
  // 記述追加のためサブクラスを定義
  refine problem P_TcpNodeDisconnect =
    TcpNodeDisconnect, // 既存の記述
    PS_EtherLanLink_P_ELLManyFragments; // 追加分
  propagate symptom // 他クラスへの伝播を追加
    PS_EtherLanLink_P_ELLManyFragments =
    SUB_EtherLanLink,InLayer, P_ELLManyFragments;
};
traced interface SUB_EtherLanLink : EtherLanLink {
  // 記述追加のためサブクラスを定義
  problem P_ELLManyFragments = ELLManyFragments;
};
```

リスト 1. 生成された追加すべき伝播モデルの記述

5 まとめ

本稿では，発生した障害イベントの集合を分析して，新規の因果関係を検出することで，対応表を用いてネットワークの障害原因を特定するシステムが伝播モデルを強化する機能について述べた．今後は本機能の評価を行う予定である．

現時点では，直接関係を持たない 2 クラス間に検出された因果関係は反映できないが，間接的に関係を有する場合には，因果関係の連鎖をたどって，2 つのイベントの因果を説明できる場合に欠けている因果関係を発見することが必要である．また伝播の向きを判定する方法についても今後検討する．

謝辞

本研究は情報処理振興事業協会の「創造的ソフトウェア育成事業」の一貫として行っている．

参考文献

[1] S. Kliger, et al.: *"A Coding Approach to Event Correlation"*, Integrated Network Management IV, p266-p277, 1995

[2] 桑原, 堀川, 吉江, 夏目: "コンピュータネットワーク自立分散管理システムの開発", インターネット研究会第 1 回ワークショップ予稿集, p124-p131, 1997

[3] D. Ohsie, et al.: *"Event Modeling with the MODEL Language"*, Integrated Network Management V, p625-p636, 1997

[4] A. Dupuy, et al.: *"NetMate: A Network Management Environment"*, IEEE Network Magazine, 1991

[5] 長尾 真 監訳.: "デジタル画像処理", 近代科学社, 1978