2Y-1

サイクル構造に着目した障害復旧方式における複数 OpenFlow コントローラの階層化による実装

田島 伸一† 長野 純一† 篠宮 紀彦†

創価大学 工学研究科†

1. はじめに

情報通信ネットワークは大規模かつ複雑になっており，短時間の障害でも通信に大きな影響を与えるため，高速かつ効率の良い障害復旧の実現が求められている．高速な障害復旧を実現するため，リング型ネットワーク上で動作する障害復旧技術がある[1]．この技術は，障害箇所のスイッチングのみで障害復旧が可能であるため，高速で効率的な障害復旧が可能である．

この技術をメッシュ型のネットワークに適応するために，ネットワークに内在するリング構造（サイクル）の集合を取り出すことが課題となる．ネットワーク内から最適なサイクル集合を求めることは困難であることが指摘されている[2]．

そこで，本研究は，容易に算出可能なサイクル集合（基本タイセット系）を用いた障害復旧方式とOpenFlowを用いて実装する方法を提案し，その特性を評価してきた[3]．

しかし，先行研究の実装は，単一のコントローラを用いていたため，大規模なネットワークに適応することは難しい[4]．そこで，本稿では，複数のOpenFlowコントローラを用いた基本タイセット系を用いた障害復旧方式の実装方法について述べる．

2. 先行研究

2.1 サイクル構造に着目した障害復旧方式

ネットワーク機器をノードとし，その接続関係をリンクとすることで，情報ネットワークをグラフとして表現する．タイセットはリングを構成するリンクの集合であり，基本タイセット系とは，ひとつの木より導出されるタイセットの集合であり，これはすべてのリンクを被覆することが知られている．

この方式はグラフ理論における基本タイセット系の概念を用いる．単一のリンクに障害が発生した場合，障害リンクを含むタイセットを基本タイセット系から取り出し，障害復旧に用いる．タイセットはリングを構成するリンク集合であるため,タイセットを用いることで障害リンクを迂回する経路を構成することができる．基本タイセット系を用いることで，文献[2]の手法と比べサイクル算出に要する計算量を短縮できる．

2.2 障害復旧方式の OpenFlow への適用

これまで，独自のプロトコルを実際のネットワーク上で検証することは，ルータやスイッチなどのハードウェアに書き込まれた制御プログラムを変更する必要があり，困難であった．そこで，容易にスイッチの制御プログラムが変更可能な OpenFlow プロトコルが提案されている[5]． OpenFlow では，ネットワーク制御機能を提供するコントローラと，転送機能のみを提供する OpenFlow スイッチを定義している．

先行研究は，障害復旧方式をOpenFlowへ適用するため，障害復旧の高速性の観点からコントロールメッセージの往復回数を最小限にとどめ，スイッチのスケーラビリティを確保するという観点から，フローエントリ数を削減している．

3. OpenFlow の分散制御

OpenFlow ネットワークにおいて，コントローラのスケーラビリティが問題視されており，これに対応するための対策が提案されている [4]．これは，OpenFlow の利点であるプログラマビリティを生かしつつ，スケーラビリティを確保するために，複数コントローラによるコントロールプレーンの構築方法を提案している．

しかし，この方法を用いて，障害復旧を実装した場合，コントローラ間の協調動作に必要な通信遅延により障害復旧が遅れたり，コントローラ間のトポロジ情報の食い違いによる不正確な経路制御が発生したりする可能性がある．また，文献[4]の方法では，DHT（Distributed Hash Table）を用いてネットワーク全体のトポロジ情報やリンク断を表すイベント情報を共有しており，ネットワークが大規模になると，トポロジの情報量やイベント発生率は大きくなるため，各コントローラへの負担が大きくなる可能性がある．このため，共有する情報を少なくすることが重要となる．

そこで本稿では，コントロールプレーンを上位層と下位層に分け，可能な限り局所的に障害復旧を行う障害復旧方式を提案する．図１に示す

Implementation of a failure recovery method based on cycle structure with layered OpenFlow controllers

Shinichi Tajima†, Junichi Nagano† and Norihiko Shinomiya†

†Graduate School of Engineering, Soka University

ように，下位層には下位グラフを持つLower_Controller（LC）が存在し，上位層には上位グラフを持つUpper_Controller（UC）が存在する．LC はスイッチと接続されており，管理下にあるスイッチ間の接続状況を把握する．UC は，複数の LC と接続されており，LC 同士の接続関係を把握している．上位グラフは，下位グラフのうちの2連結成分を縮約したグラフである．それぞれのコントローラは作成したグラフを用いて基本タイセット系を作成する．

LC のスイッチがパケットを受け取った時，そのパケットがその LC の管理下にあるネットワークのみを用いて対応可能である場合は，その LC のみで経路制御を行う．対応不可能な場合は，UC と通信し，UC が対応する経路を決定し，LC に対して指示を出し経路制御を行う．障害復旧の場合も，LC のみで対処可能な場合，下位グラフから作った基本タイセット系を使い障害復旧する．対処不可能な場合，UC と通信し，上位グラフから作った基本タイセット系を使い復旧経路を決め LC に指示を出す．

4. 階層化コントローラを用いた実装

ここでは，LC の実装方法について述べる．クラス構成は図2のようなる．

図2の LC は管理下にあるスイッチから送られてくるイベントを受け付け，スイッチへ指示し，packet_in や port_status，feature などのイベントの発生を各コンポーネントへ通知する．

routing は packet_in イベントを受け取り，そのパケットを転送するための経路情報を LC に返す．LC は経路情報を path として保持する．

lower_graph は feature イベントを受け取り，ノードを把握する．リンクの把握は，さまざまな OpenFlow による実装で使用されている LLDP（Link Layer Discovery Protocol）による方法を用いる．lower_graph に変化があったとき，lower_graph の2連結成分を縮約した Contracted graph を作成する．

path は経路情報を定義しており，経路情報はインポートを含まない mach のルールとスイッチと出力ポートの列により表現される．

fundamental_tie-set は，lower_graph が変更されたときに基本タイセット系を作成し，保持する．また，リンク断を表す port_stats イベントが発生したとき，LC は，障害リンクとそのリンクを含む path の集合を fundamental tie-set に渡す．これらの情報を元に，予備経路を示す path が作成される．その後，LC はその復旧経路をスイッチに登録する．

LC が対応できない障害が発生した場合，UC

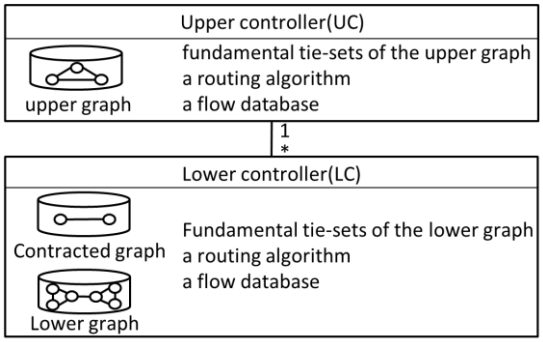

図1　Upper_Controller と Lower_Controller

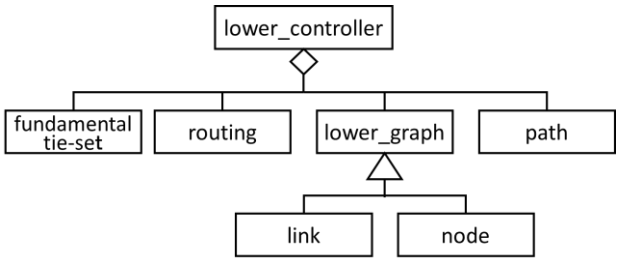

図2　Lower_Controller のクラス構成

にイベント情報を送る．UC は，障害リンクと上位グラフより，上位グラフ上の復旧経路を算出する．UC は，この経路を各 LC の Contracted graph 上の経路に変換し，各 LC に通知する．LC は受け取った経路を下位グラフ上の経路に変換し，その経路をスイッチに登録する．

5. まとめ

本稿では，階層化した複数の OpenFlow コントローラを用いた基本タイセット系に基づく障害復旧方式の実装について述べた．

今後は，コントローラのスケーラビリティ確保の観点からさまざまな評価実験を行う．

6. 参考文献

[1] ANSI T1. 105.01-1998 “Telecommunications Synchronous Optical Network (SONET) – Automatic Protection Switching,”

[2] M.S. Kiaei et al. “A Survey on the p-Cycle Protection Method," IEEE Communications Surveys & Tutorials, Vol.11, Issue 3, pp.53-70, 2009.

[3] 長野ほか, “OpenFlow によるサイクル構造に着目した障害復旧方式の実装と評価,” 電子情報通信学会論文誌 D, Vol.j96-D, No.10, pp.2340-2350, Oct. 2013.

[4] A. Tootoonchian et al. “HyperFlow: A Distributed Control Plane for OpenFlow,” Proc. of NSDI Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN), San Jose, CA, USA, April 2010.

[5] OpenFlow Consortium. “OpenFlow Switch specification version 1.0.0,” http://www.openflow.org/