ネットワークデバッガの提案

廣中 颯[1]　篠田 陽一[1]　知念 賢一[1]　宇多 仁[1]

概要：ネットワークの複雑化と規模の拡大により、ネットワークにおける障害解決は難しくなってきている。本研究はネットワークの障害解決における作業記録の統合的な管理・提供および問題把握の支援により、ネットワーク障害解決の効率化を支援するシステムを提案する。現在はオペレータが ping, traceroute, tcpdump などの複数のツールを用いて情報収集を行っている。これらのツールは優れた機能を持つが、多数のツールを使いこなすことは難しく、習得にもコストがかかる。ネットワークは分散システムの性質上、個々の機器が独立して動作しているため、個々の機器に対して行われた操作の作業記録が断片化しやすい。それぞれの機器に加えられた操作や設定記述を時間軸に沿って統合的かつ横断的に管理しなければ、ネットワーク障害解決における作業工程の全体像を把握できない。本稿は、ネットワーク障害解決を対象に、障害解決における作業工程を整理し、提案システムに求められる機能およびシステム構成を論じる。

Proposal of Network Debugger

HAYATE HIRONAKA[1]　YOICHI SHINODA[1]　KEN-ICHI CHINEN[1]　SATOSHI UDA[1]

1. はじめに

ネットワークの利用形態の複雑化と多様化によって、ネットワークに求められる要件が増え、ネットワークが複雑化している。また、ネットワークの規模もネットワークに繋がる端末や機器の増加に伴って拡大している。従って、複雑性を増し、規模が拡大したネットワークにおける障害解決 (トラブルシューティング) はより難しくなってきている。

障害解決において、情報収集は最も重要な要素である。複雑化し、規模が拡大したネットワークでは、情報量が増え、情報収集が難しくなっている。現在はオペレータが ping[1], traceroute[2], tcpdump[3] をはじめとする複数のツールを用いることでネットワークの情報収集を行っている。これらのツールは洗練されており、優れた機能を提供してくれるが、多数のツールを使いこなすことは難しく、習得にコストがかかる。そのため、オペレータにこれらのツールが持つ機能をより使いやすく提供し、ツールの出力をわかりやすく伝えることが必要である。

ネットワーク機器は分散システムの性質上、個々の機器が独立して動作している。そのため、個々のネットワーク機器に対して、設定記述の変更やログの閲覧などの操作を行うと、個々の機器上に作業記録が残る。従って、ネットワーク内に存在する機器に断片的に作業記録が保存される。その場合には、それぞれの機器に加えられた操作や設定記述を時間軸に沿って統合的かつ横断的に管理しなければ、作業工程の全体像を把握できない。また、組織内でネットワークの管理運用を行う場合には、ネットワーク構成図、作業手順書、報告書などの書類を作成し、情報共有を行う。しかし、ネットワークの障害対応は時間に制約がある中で行われることも多く、最終的に判明した障害原因を記録できても、どのようなプロセスをたどって問題の特定に至ったかを記録することは容易ではない。加えて、これらの問題に対して、ネットワークの障害解決においては複数のツールを用いて、独立した複数の機器に対する情報収集および操作を行う必要があり、このこともネットワークの障害解決の難しさの一因である。

Software Defined Network (SDN) を対象とした障害解決を支援するシステムにネットワークデバッガ ndb[4] がある。本研究では、仮想化されたネットワークだけでなく、実在の機器を構成要素としてもつネットワークも対象とする。

[1] 北陸先端科学技術大学院大学
JAIST

ネットワークにおける障害解決について述べる。ネットワークの障害対応では、まず、問題の切り分け、障害箇所の絞り込みを行う。そして、障害原因を推定し、障害解消のための変更を行う。さらに、問題が解決するまで一連の作業を繰り返す。この一連の作業を一つの障害解決プロセスとして捉える。この障害解決プロセスにおける作業の記録や問題把握の支援などを行うことにより、作業の効率化や手戻りの削減など、障害解決に対して一定の効果が見込める。

本研究では、障害解決における断片的な作業の記録の統合的な管理・提供及び問題把握の支援により、ネットワーク障害解決の効率化を支援するシステム (ネットワークデバッガ) を新たに提案する。提案システムを用いることでネットワークに対する情報収集を容易にし、オペレータの現状把握を助けることで作業中の手戻りを減らし、作業効率を改善できる。

2. 背景

2.1 デバッガの概念

デバッガは主にソフトウェアに対する障害解決を行うためのツールである。デバッガが持つ一般的な機能には、ステップ実行、ブレークポイント、実行中の変数書き換えなどがある [5]。

デバッガは、プログラムのバグの追跡、特定および排除を助け、プログラムの動的な性質を明らかにするだけでなく、プログラムを理解するためにも役立つ。

デバッガはステップ実行やブレークポイントの機能によって、時間軸に沿って、任意の時点の変数の中身やプログラムの動的な流れをプログラマに提供し、プログラム実行時における情報を可視化することでプログラマの認識を助ける。また、任意の時点での変数書き換えなどの機能によって、プログラマが行う問題の切り分けや原因の推定を容易にし、効率化する。

2.2 ネットワークにおける障害解決

ネットワークは独立したネットワーク機器の集合によって構成されており、ネットワークの各構成機器は設定記述に従って、独立に動作している。従って、ネットワークは構成機器の設定記述の集合によって記述されているとみることができる。ソフトウェアを構成するプログラムのソースコードも多くの場合分割されて記述されている。つまり、ネットワーク内の各機器の中に存在する設定記述はプログラムの分割されたソースコードと似ている。

しかし、ネットワークにおける障害解決は、ソフトウェアに対する場合と様々な面で異なる。ネットワークは多くのプログラムのように手続き的に実行されるわけではなく、複数の独立した機器の相互作用によって構成されている。そのため、プログラムに対してデバッガを用いるように、ネットワークに流れるパケットをステップ実行のようにノード単位で停止させながら転送したり、ブレークポイントを設置して任意の時点でネットワークに流れるパケットを静止させたりすることは、物理的な機器を含むネットワーク上で実現することは難しい。

また、ネットワークはレイヤ構造をなしており、あるレイヤに対して障害解決を行う際に、その下位レイヤの設計に根本的な原因が含まれていたとしても、上位レイヤの設計によってその原因を補修しなければならない場合がある。

ネットワークにおける障害解決に関する知見の共有は障害解決の効率化に役立つ。実運用中のネットワークに対する障害解決は時間に制約がある中で行われることが多く、障害解決プロセスを逐一記録していくことは難しい。時間に追われて作業する中では、一つ一つの作業に対して記録をとる余裕がない。そのため、オペレータが意識をしなくとも自動的に操作が記録として保存される仕組みが必要である。

その他にも、ネットワーク機器へ操作を行う場合は、直接機器に結線するか、もしくはリモートでログインして作業を行うが、実際の操作履歴 (ヒストリ) は各機器に記録される。後から全体の操作履歴の時間軸に沿った流れを確認するためには、複数の機器に記録が断片化して保存されており、まとめる手間がかかる。また、例えば、断片化した履歴にタイムスタンプなどの付加情報を加えて記録したい場合に、ネットワーク機器には様々なベンダの製品があるため、個々の機器に対して改修を行うのは現実的ではない。

2.3 障害解決プロセス

Cisco[6] はネットワーク環境の障害解決について、8 段階からなるプロセスで一般的な問題解決モデルを定義している。

本研究では次のように 4 段階で障害解決プロセスを定義する。ネットワークにおける障害解決では、まず、問題の切り分けおよび障害箇所の絞り込みを行う。そして、障害原因を推定し、障害解消のための変更を行う。変更を行っても問題が解決しない場合は、問題が解決するまで一連の作業を繰り返す。この一連の作業を障害解決プロセスと定義する。そして、提案システムはこの障害解決プロセスを支援し、記録することを目的とする。

2.4 ネットワーク特有の難しさ

前述したように、ネットワークの障害解決には特有の難しさがある。まず、複数のツールを用いて独立した複数の機器に対する情報収集および操作を行う必要がある。加えて、ネットワークはレイヤ構造をなしており、インターネットにつながる個別のネットワークにはそれぞれオペレータがいるため、対象のネットワークやそれにつながるネットワークに操作できないレイヤや機器が存在する。

次にネットワーク障害解決における現状の課題を挙げる。

- 独立して動作する複数のネットワーク機器に対して情報収集や矛盾のない設定を行うことが難しい。
- 作業記録が個々の機器上に断片化して保存される。
- 障害解決は時間に制約がある中で行われることが多く、障害解決プロセスを記録することが難しい。

2.5 関連研究

ネットワークの障害解決を目的とした研究には色々な手法がある。SDN を対象としたネットワークデバッガ ndb の研究として Heller ら [4] の研究がある。Heller らの作成した ndb は、ブレークポイントを設定し、条件に合致したネットワーク内のパケットに対してバックトレースによる解析を行うことができる。

高山ら [7] の障害解決における情報収集プロセスを対象とした研究がある。ネットワーク障害解決を得意とする熟練者の知識構造を論じ、座学による知識と実習による知識、現場経験による知識の 3 つの知識を組み合わせて故障の切り分けを行なっていると結論づけている。

糸井ら [8] は障害対応業務におけるルール判定を用いた自動障害箇所推定の研究を行なっている。機械的な判定を基とした手法はオペレータの熟練度に依存しない利点があるが、初めて生じた障害や発生頻度の少ない障害は検知が難しい。

その他にも、ネットワークの障害解決に対するひとつの手法として、監視・運用ソフトウェアをあらかじめ入れておき、ネットワークの状態を観測・監視することが挙げられる。zabbix[9] は SNMP, IPMI を用いて様々なメトリックを収集し、異常を検知してオペレータに通知したり、収集したデータをグラフで可視化したりすることができる。

3. 提案と実装

3.1 ネットワークデバッガ

本研究では、ネットワークにおける障害解決の支援を目的としたシステムを提案する。提案システムはオペレータがネットワークの問題点を発見するのを助け、オペレータは発見した問題点に対して、ネットワーク構成機器の設定記述の修正やネットワークの構成変更により、ネットワークの完成度を高める。そして、ネットワークの問題点を取り除くだけでなく、ネットワークの振る舞いを解析し、そのネットワークの状態や動作や構成への理解を深めることにも役立つシステムの構築を目指す。

3.2 ネットワークの障害解決における情報収集

ネットワーク障害解決において、最も重要なことは情報収集である。ネットワークの構成、各構成機器の設定記述および状態、それらに加えて、オペレータの行う操作に対するネットワークの変化をオペレータに正確にわかりやすく伝えることにより、障害解決を助けることができる。

ネットワーク障害解決の際には、ネットワークの現状把握を行うことで発生している問題を定義し、問題の切り分けによって、問題の発生箇所を特定し、さらに情報収集を重ねて行うことで、より詳細な原因の特定を行う。そのため、ネットワークに関する情報収集を支援することは障害解決の効率化につながる。

3.3 機能

前の章で整理した内容をもとに、本研究で提案するシステムに必要となる機能を検討する。

3.3.1 操作

ネットワークの構成機器に対する操作機能を提供する。操作機能は、オペレータの行った操作を提案システムが仲介し、記録するためにも役立つ。

また、グラフィカルユーザインタフェース (GUI) を通じて対象を選択して、機器に対する操作を行えるようにすることで、操作対象の取り違えから生じる操作ミスを減らし、オペレータが自身の行なっている作業を正確に把握することにも役立つ。

ネットワークを構成する機器はすべて独立に動作するため、矛盾する設定記述を受け入れる。例えば、直接結線された対向のインタフェースにおけるポート VLAN 設定の不一致といった、機械的に容易に判定できる問題に関しては、ネットワークの構成機器の設定記述を横断的に管理し、設定検証を行うことで問題の発見を支援できる。

その他には、ダミーパケットの生成を伴う設定の自動検証といったネットワークに対する能動的な操作を行う機能の提供が情報収集における問題の切り分けを助けるために役立つと考えられる。

3.3.2 記録

オペレータは提案システムが提供する操作機能からネットワークの構成機器に対して操作を行う。そのため、提案システムは独立して動作するネットワークの構成機器に対してオペレータが行う操作を統合的に記録することができる。そして、提案システムは操作機能を通じて各機器の設定記述や経路情報を収集し、横断的に管理を行うことができる。

OSPF のような動的なプロトコルを含むネットワークは経路情報が動的に書き換わるため、ネットワークの完全なスナップショットをとることは難しい。しかし、それぞれのネットワーク機器は設定記述に基づいて動作するため、保存しておいた設定記述をネットワークの構成機器に投入することでネットワークの擬似的なロールバックを行うことができる。

また、提案システムは操作機能を通じて、多様な操作を記録する。例えば、ネットワーク機器に入力されるコマンドを記録していくことで多様な操作をコマンド列として記

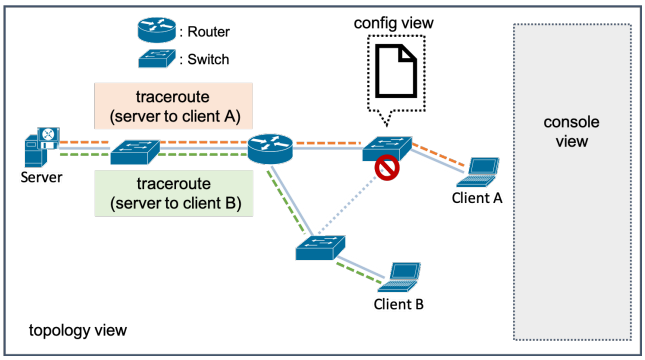

図 1　画面イメージ

録できる。仲介した操作を記録するだけでなく、特定の時点に各機器から設定記述を収集することでネットワークの設定情報の記録を行うことができる。そして、その反対に、設定記述の集合として記録した設定情報をまとめて各機器に投入することで設定情報の再生を行うことも考えられる。

ただし、本来は逐次的に各機器に入力される設定記述をまとめて複数の機器に投入すると、ネットワークの最終的な状態が意図する状態に収束するかどうかは調査する必要がある。

3.3.3　可視化

可視化機能の画面イメージを図 1 に示す。可視化を行う対象はネットワークの構成図である。その上で、オペレータが疎通確認や経路の確認を行うための ICMP パケットの経路を構成図上に重ねて表示する。

3.4　想定する使用事例

3.4.1　到達性

まず、ネットワークの動作検証において最も基本となるのは到達性の確認である。ネットワーク全体の構成やパケットの通る経路を可視化することにより、障害が生じている箇所の発見を助けることができる。

3.4.2　経路確認

柔軟な経路制御によって、ネットワークをより効率的に使用することができる。しかし、ネットワークを流れるパケットは直接目に見えないため、動作検証が難しい。もし見えていても運用中のネットワークには大量のパケットが高速で流れているため、単に見えるだけでは必要な情報を正しく把握できない。

ネットワーク上のパケットが通る経路の把握も前項と同様に可視化によって容易になる。

3.4.3　リンクアグリゲーション (LAG) の検証

リンクアグリゲーション (LAG) は RFC7424[10] で定義されている。LAG の動作確認は障害解決において手間がかかる。LAG によって論理的にまとめられたリンクがネットワーク内に無数に存在する場合、その全てを試験することが困難である。一見到達性があるように見えても、論理的にまとめられたリンクが全て正常に動作し、負荷が生じた際に想定するスループットが出せるか、もしくは、片方のリンクが故障した際にも通信が行えるかなど正しく確認するには非常に手間がかかる。

LAG による負荷分散におけるパケットの振り分けルールには、送信元・宛先の IP アドレスや MAC アドレスによるものなどいくつかある。もし、送信元 IP アドレスによってパケットのロードバランシングを行なっている場合、送信元 IP アドレスを変更したパケットをいくつか自動的に生成し、通るリンクを観測することで LAG の動作確認を行うことができる。

表 1　構成モデルの分類

構成モデルの名称	配置	対象装置への追加方法	対象装置への影響
エージェント型	分散	内蔵	大
エージェントレス型	集中	外付け	少
ハードウェア型	分散	外付け	大

3.5　システムの構成モデル

ネットワークデバッガを実現するための手法はいくつか考えられる。本章ではシステムの構成モデルを検討する。システムの構成モデルとは、ネットワークデバッガの実体がどこに配置され、動作するかを表現するモデルである。3 つの構成モデルを考え、表 1 に分類した。

3.5.1　エージェント型

まず、エージェント型の構成モデルを考える。エージェント型の構成モデルを図 2 に示す。エージェントとは、ある機器やシステムに常駐して情報収集や通信の仲介などを行うソフトウェアの実体のことを指す。エージェント型では、ネットワーク内のすべての構成機器にエージェントを常駐させる。

すべての機器にエージェントを常駐させるため、後述するエージェントレス型と比較し、より高度な機能をユーザに提供できる。しかし、実際には、ベンダの提供するネットワーク機器への外部ソフトウェアの導入は許されていない場合が多い。そのため、動作環境となるネットワーク機器が限定される。また、障害解決のための外部ソフトウェアをネットワーク機器上で動作させることは、調査対象のネットワークに少なからず影響を与えることとなり、障害解決の対象となる要素が増えることに繋がる。

3.5.2　エージェントレス型

次に、エージェントレス型の構成モデルを考える。エージェントレス型の構成モデルを図 3 に示す。エージェントレスとは、ネットワークの各構成機器にはソフトウェアを常駐させないこととする。つまり、エージェントレス型では、オペレーション用のサーバから遠隔で各構成機器に対する情報収集や操作を行う。

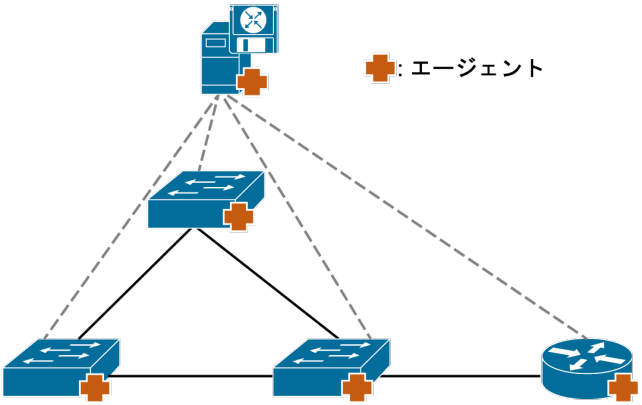

図 2　エージェント型

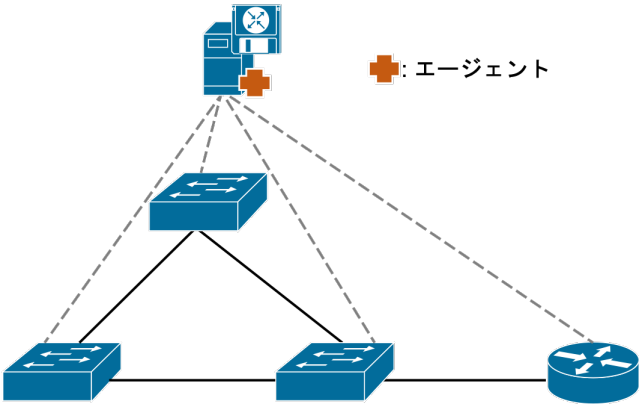

図 3　エージェントレス型

エージェントレス型の利点は導入が容易であること、対象となるネットワークの構成機器に構成外のソフトウェアおよびハードウェアを導入する必要がないため、 ネットワークに不意の影響を与える恐れが少ないことである。その反面、遠隔で情報収集を行うより、機器上で動作するエージェントや機器に直結したハードウェアで情報収集を行った方がより詳細な情報が得られる。例えば、前章で述べたダミーパケットの挿入を用いた LAG の検証のような機能は実現が難しくなる。

3.5.3　ハードウェア型

ハードウェア型の構成モデルでは、ネットワークの監視やパケットに対する操作を行うための専用ハードウェアをネットワーク内に設置することでネットワークデバッガの機能を実現する。ハードウェア型の構成モデルを図 4 に示す。具体的には、ネットワーク機器同士を接続するリンクの間に専用ハードウェアを設置し、パケットの観測やパケットの停止、書き換えなどの操作を行う。この構成モデルを用いることにより、ネットワークの構成機器のソフトウェアおよび設定記述を変更することなく、専用ハードウェアによって、パケットの観測や書き換え、挿入などの機能を実現できる可能性がある。しかし、専用ハードウェアの設置により、ネットワークの動作に影響を与えうる点はエージェント型と同様である。

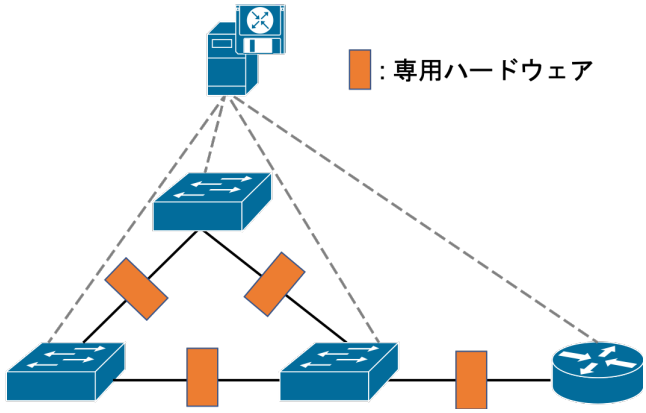

図 4　ハードウェア型

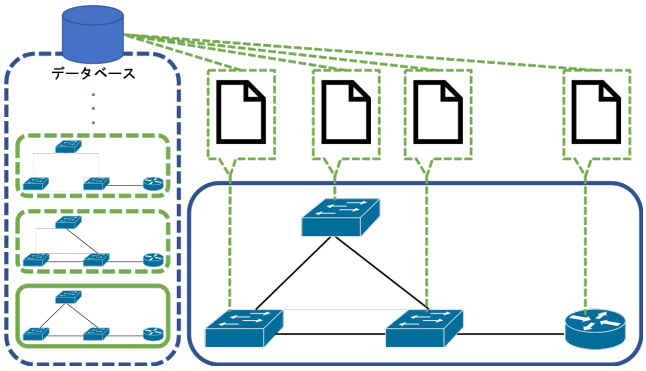

図 5　システム構成：記録の収集

3.6　実装計画

前章で議論した構成モデルの中で、実運用環境への導入の容易さ、対象ネットワークに対する影響の少なさを考慮し、エージェントレス型を採用する。

概念実証として、図 1 のように可視化機能およびネットワークの各構成機器に対する操作機能、記録機能を実装する。システム構成を図 5, 6 に示す。実装したシステムを試験環境において動作させ、いくつかの適用例において有効であることを確認する。

実装の流れについて述べる。まずネットワーク構成および ping, traceroute をはじめとする操作の可視化機能の実装から行う。その次に、可視化機能およびコマンドラインインタフェース (CLI) を通じてネットワークの構成機器に対して操作を行うための操作機能を実装する。最後に、操作機能を通じて行われた機器への操作を統合的に記録し、各機器の設定記述を管理する記録機能の実装を行う。

これら主な 3 つの機能が揃うことで、ネットワーク全体に対する設定記述のロールバックの適用や障害解決プロセスの記録が可能になる。

4.　おわりに

ネットワークの複雑化と多様化、規模の拡大によってネットワークにおける障害解決が難しくなってきている。ネットワークの障害対応は時間に制約がある中で行われ

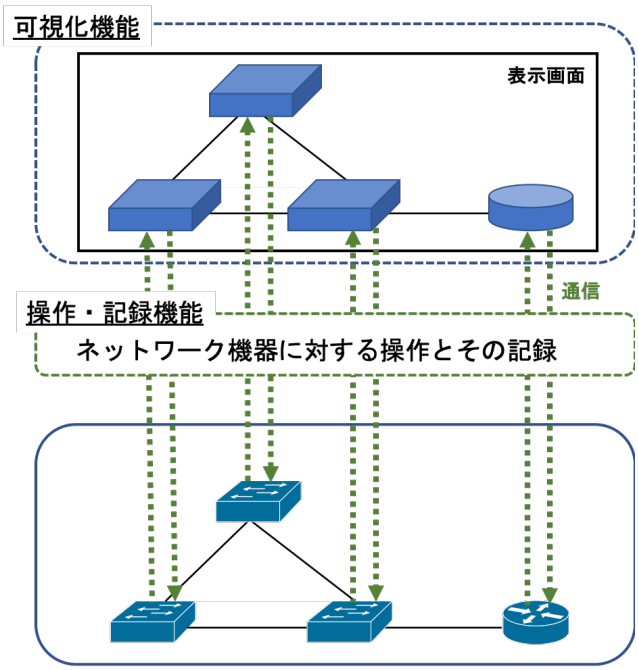

図 6　システム構成：機能の関係

ることも多く、障害解決までに辿ったプロセスを正確に記録することは容易ではない。また、ネットワーク機器はそれぞれが独立した設定記述を持っており、独立して動作する。そのために、各機器に対して行った操作の記録もネットワークの各構成機器に断片的に記録される。そのため、障害解決プロセスの全体像を把握するためには、それぞれの機器に加えられた変更や設定記述を時間軸に沿って統合的に記録かつ横断的に管理する必要がある。

本稿では、ネットワークの障害解決における作業記録の統合的な管理・提供および問題把握の支援により、ネットワーク障害解決の効率化を支援するシステムを提案した。そして、ネットワークにおける障害解決プロセスを整理し、提案システムに必要となる機能を検討した上で、想定する使用事例を挙げ、システムの構成を議論した。

最後に展望を述べる。障害解決プロセスを定式化し、記録できるようにすることで将来的に記録した障害解決プロセスを機械学習の教師データに適用できる可能性がある。

参考文献

[1] Foundation, F.: ping, `https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&manpath=FreeBSD+12.0-RELEASE+and+Ports`. (参照 2019-05-13).

[2] Foundation, F.: traceroute, `https://www.freebsd.org/cgi/man.cgi?query=traceroute&manpath=FreeBSD+12.0-RELEASE+and+Ports`. (参照 2019-05-13).

[3] Group, T. T.: tcpdump, `https://www.tcpdump.org/manpages/tcpdump.1.html`. (参照 2019-05-13).

[4] Handigol, N., Heller, B., Jeyakumar, V., Maziéres, D. and McKeown, N.: Where is the Debugger for My Software-defined Network?, *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pp. 55–60 (2012).

[5] Jonathan, R.: デバッガの理論と実装，株式会社アスキー (1998).

[6] Cisco: トラブルシューティングの概要，`https://www.cisco.com/c/ja_jp/support/local-guide/tr1901.html`. (参照 2019-04-26).

[7] 高山千尋，大野健彦：トラブルシューティングにおける情報収集プロセス：熟練者はどのように手がかりを得ているか，技術報告 12，NTT サイバーソリューション研究所 (2011).

[8] 糸井謙史，矢川太祐，大石晴夫，岡崎勝彦：障害対応業務のナレッジ化・迅速化をめざした自動障害箇所推定技術，NTT 技術ジャーナル，Vol. 29, No. 5, pp. 60–64 (2017).

[9] Zabbix 社：Zabbix The Enterprise-Class Open Source Network Monitoring Solution, `https://www.zabbix.com/jp/network_monitoring`. (参照 2019-05-01).

[10] Krishnan, R., Yong, L., Ghanwani, A., So, N. and Khasnabish, B.: Mechanisms for Optimizing Link Aggregation Group (LAG) and Equal-Cost Multipath (ECMP) Component Link Utilization in Networks, RFC7424 (2015).