G-17

ネットワーク障害が通信品質に与える影響に基づく品質劣化パターン分析システムの提案

Proposal of a Quality Degradation Pattern Analysis System Based on the Effect of Network Failures on Communication Quality

上野 颯大†　　井口 信和†‡
Sota Ueno　　Nobukazu Iguchi

1. 序論

令和 6 年度の総務省の情報通信白書によると，2011 年に発生した東日本大震災以来，迂回経路や冗長化といった伝送路断線対策を強化したネットワークが重要視されている[1]．しかし，人的なミスや自然災害などによる要因から，ネットワーク障害の発生を完全になくすことは極めて困難である．また，迂回経路の高速な計算手法を提案している鈴木氏は，障害発生後の迂回路におけるトラフィック変化の影響を明らかにする必要性について述べている[2]．そのため，実際に障害を発生させて，通信への影響を検証するといったアプローチが求められる．

擬似的に障害を発生させる検証手法として，カオスエンジニアリングという方法が存在する．この手法では，稼働中の本番環境において，検査対象となるサーバ機器やシステムの正常な状態をはじめに定義し，その後に意図的に障害を発生させることで，正常な状態との違いや負荷状況を比較する．その結果から，対象となるサービスの信頼性を調査することがこの手法の主な目的となる．しかし，本番環境に障害を発生させることはユーザへの直接的な影響が懸念されるため，検証時にはサービスが稼働していない保守日を設ける必要があるといったデメリットが存在する．特に検証範囲をネットワークへと拡張する場合，障害生成によるネットワーク利用者への影響はさらに拡大する可能性もある．

また，障害が発生した時に備えて，対策を練ることが重要である．対策方法の一例として，事業継続計画（BCP）が存在する．福田氏は，大学での BCP の策定は，他の自治体や企業と比較して極めて遅れていると述べており，図 1 に示すように，策定済みでない大学はおよそ 6 割にも及ぶ[3]．さらに，大学図書館の蔵書やネットワーク，サービスへのアクセスを維持することが重要であると述べている．一般企業においても，ネットワーク構成の変更があった時には，動作検証および有事への対策を練り直す必要があると考えられる．

そこで本研究では，ネットワーク障害時における通信品質の劣化の把握と，障害対応策の検討の補助を目的に，仮想ネットワーク環境で擬似障害を発生させ，通信品質の測定データに基づいて品質の劣化パターンを分析するシステムを提案する．本システムを用いることで，ネットワーク運用者は品質劣化パターンをもとに，障害発生時にネットワークの性能がどのように低下するかを事前に予測することが可能となる．これは，障害対策を論理的に検討するための，客観的な判断材料になると期待される．

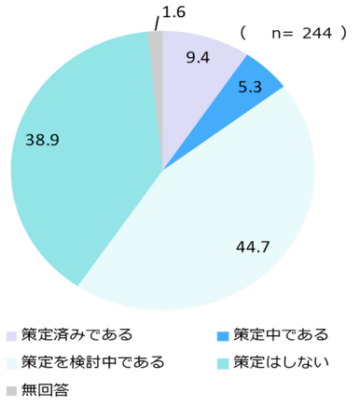

図 1 業務継続計画（BCP）策定の状況[3]

2. 関連研究

菊田らの研究では，LLM を用いたカオスエンジニアリング自動化手法として，Kubernetes 上で動作するサービスに対して障害を自動生成し，その時の負荷状況やサービスの脆弱性を確認するシステムを開発している[4]．菊田らは，カオスエンジニアリングの自動化を目的としているが，本研究では，擬似障害を生成して検証するカオスエンジニアリングの考え方をもとに，通信品質の測定と劣化パターンを分析する．

また，仮想ネットワークを構築する手段として，コンテナ・ベース・エミュレータ（CBE）がある．Schalk Peach らの研究では，CBE を用いることで低オーバーヘッドの仮想ネットワークの構築が可能であり，ネットワーク管理の専門家にとって，費用対効果の高いネットワーク実験プラットフォームであると結論づけている[5]．CBE の 1 つである containerlab[1]は，Linux ベースの Docker コンテナを構築することで仮想機器をエミュレートする OSS である．containerlab では，ネットワーク OS をベースとするコンテナを作成することもできるため，OS に対応した実機に近い挙動が再現可能である．本研究では，仮想ネットワークの基盤として containerlab を採用する．

† 近畿大学 情報学部 情報学科
Department of Informatics, Faculty of Informatics, Kindai University
‡ 近畿大学情報学研究所
Cyber Informatics Research Institute, Kindai University

[1] https://containerlab.dev/

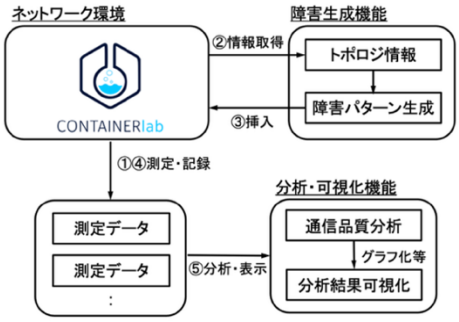

図 2 システム構成図

図 3 測定用パラメータの設定(抜粋)

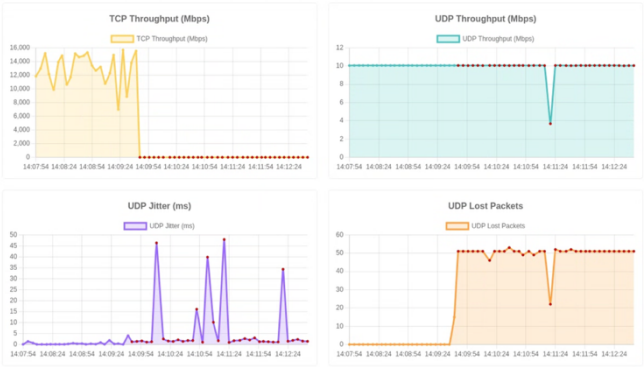

図 4 測定状況のリアルタイム表示(抜粋)

3. 研究内容

本システムの構成を図 2 に示す. 本システムは Linux 環境上で動作するアプリケーションであり, 一定間隔で通信品質を取得する測定機能, ネットワークトポロジ情報に基づいて擬似障害を生成する障害生成機能, 記録した測定データの分析・可視化機能から構成される. 次節以降に, 本システムの機能について述べる.

3.1. 測定機能

本機能では, 任意の仮想機器間の通信品質を測定し, その障害生成前後の測定データをログとして記録する. 通信品質を測定するための指標として, 遅延, パケットロス率, TCP および UDP スループット, ジッターを使用しており, 測定手法として ping および iperf3 を使用する. ここで, 記録される各レコードが障害生成前のデータであるか後のデータであるかを判断するために, boolean 型の障害生成フラグをフィールドとして設定する. 測定機能と障害生成機能は基本的に独立して動作するが, 障害生成時のみ, 障害生成機能から測定機能へリクエストを送信し, 測定機能が

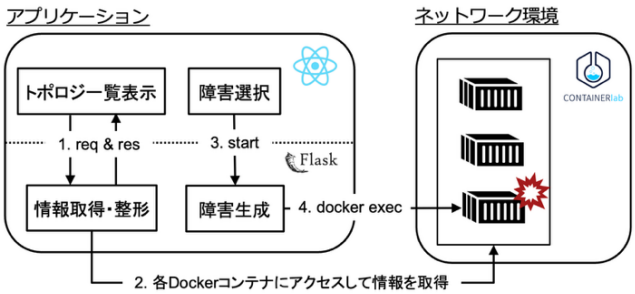

図 5 障害生成機能の流れ

管理する障害生成フラグを立てる. この仕組みにより, 分析・可視化機能において, 障害生成前後の比較分析が可能となる. 測定用パラメータの設定に関する GUI の例を図 3 に示す. 送信元コンテナ名には ping, iperf3 のクライアントとなる Docker コンテナ名を入力し, 宛先コンテナ名および宛先 IP には, サーバとなる Docker コンテナ名およびインターフェースの IP アドレスを入力する. 宛先となるコンテナでは, 測定開始ボタンの押下と同時に iperf3 -s コマンドがバックグラウンドで実行され, iperf3 クライアントとの間で測定用のコネクションが結ばれる. 測定間隔(sec)は, 測定処理が終了してから次の測定処理が開始するまでのタイムアウトを表すものであり, ping カウント数, iPerf 間隔(sec)に入力した数値は, それぞれ測定ツールのコマンドオプションとして指定される.

また, アプリケーション上で確認可能な測定状況のグラフの例を図 4 に示す. 各グラフの横軸は測定時のタイムスタンプを表し, 縦軸は通信品質の指標を表す. 測定記録は csv ファイルとしてローカルに保存されるため, 該当する csv ファイルからリアルタイムで測定データを読み取り, グラフとしてそれぞれ描画する. 障害生成後の測定データの判別として, 前述した障害生成フラグが立っている場合, グラフの節となる部分を赤く強調表示する.

3.2. 障害生成機能

本機能では, 仮想ネットワーク環境に対して擬似障害を発生させる. 障害生成画面では, 仮想機器として動作している Docker コンテナにアクセスし, プロセス情報やインターフェース情報からネットワークトポロジを取得する. 障害生成機能の流れを図 5 に示す. containerlab で動作する Docker コンテナ名は, デフォルトでは接頭辞に”clab-”という文字列が付くため, docker ps コマンドの filter オプションで指定して識別することが可能である. 対象となるコンテナを識別した後に, それぞれのコンテナに対して ip addr コマンドを順次実行し, インターフェースごとのアドレス情報やサブネット情報を取得する. 最終的にこれらの情報をマッピングし, ネットワークトポロジとして管理する. トポロジ情報の取得は障害生成画面上から手動で行い, 任意のタイミングでトポロジ情報の更新が可能である. 取得したトポロジ情報をもとに, 障害を生成するノードやリンクを選択し, 障害を生成する. 本機能により, 擬

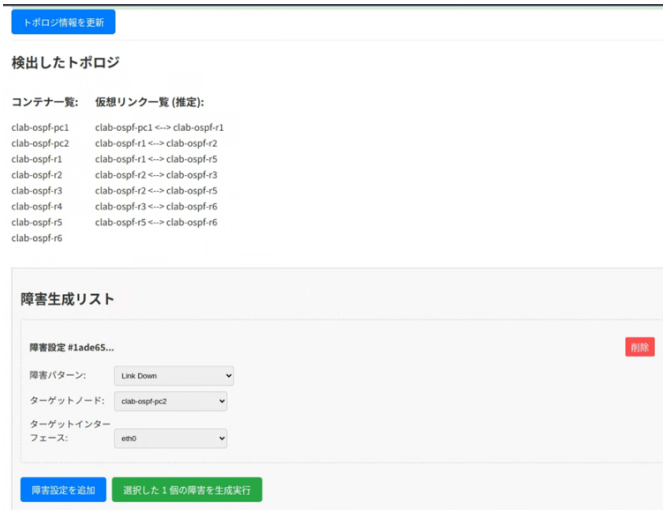

図 6 障害生成画面(抜粋)

表 1 障害パターン一覧

レイヤー	障害パターン
物理層(L1)	ノード・リンクダウン
データリンク層(L2)	リンクダウン
ネットワーク層(L3)	ルーティングループ 帯域制限・遅延付与

似障害の設定と生成の一連の動作が円滑に実施可能となる.

本システムにおける障害生成画面の例を図 6 に示す. システム利用者は, プルダウンメニューから障害パターン, ターゲットとなる仮想機器, インターフェースを選択する. その後, 生成実行ボタンを押下することで, バックエンドへリクエストが送信され, ターゲットとなるコンテナに docker exec コマンドを介して擬似的な障害が生成される. また, “障害設定を追加”というボタンを押下することで, 障害生成リストの末尾に新しく障害設定フォームが追加される. この仕組みにより, 複数箇所で任意の障害パターンを生成することや, 2 種類以上の障害パターンを組み合わせ, 同じタイミングで障害を生成することが可能となる. 今後, 追加で実装する内容として, 障害の同時生成に加えて, 連鎖的に生成するようにスケジュール可能な仕組みについても実装する予定である. このスケジュール機能を実装することにより, 障害事例の再現の幅がさらに広がることが期待される. スケジュール機能の実装方法として, 1 つ目に生成される障害から, 対象の障害が生成されるまでの秒数を各障害設定フォームに設定し, 1 つ目の障害が生成されてからの経過時間を, タイマーによって管理する手法を思案している.

本システムに実装されている障害パターンについて, レイヤーごとにまとめた表を表 1 に示す. 物理的な断線や, 電源断を再現するためのリンクダウンやノードダウンといった L1・L2 周りの障害をはじめ, 機器に大きな負荷を与える要因となるルーティングループ障害, 災害時の逼迫した

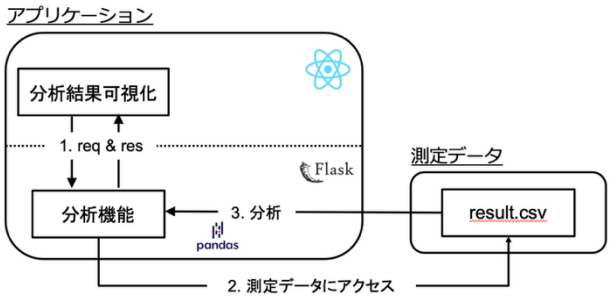

図 7 分析・可視化機能の流れ

通信状況を再現するための帯域制限, 遅延付与といった L3 を対象とした障害の生成に対応している. ここで, ルーティングループ障害について, 障害生成の設定次第では測定用パケットが宛先に届かず, 測定結果が正しく返ってこない場合がある. また, ループにより数十秒ほどで PC の動作が重くなるという問題が生じた. この問題を解決するため, 障害生成後から指定した秒数が経過すると, ループが解除されるようにする. 測定パケットが宛先に到達しない場合は, 通信品質の値をそれぞれ None に設定し, パケットロス率は 100%と記録する. これらの変更により, 上記の問題は概ね解決し, ルーティングループ解消後の通信品質にも着目することが可能となる. 他にも, さまざまなネットワーク障害のパターンが考えられるが, その中でも今後は, 主に L2 を対象としたループ障害や, ブロードキャストストームのような, 通信品質に大きな影響を与えることが予測される障害パターンを取り扱い, 実装する予定である. ここで, 新しい障害パターンの実装において, 各障害パターンが containerlab で構築した仮想環境上で再現可能であるかどうかを検証する必要がある. そのため, 上記の障害パターンの再現に必要となる技術やツールの調査を進める.

3.3. 分析・可視化機能

本機能では, 障害生成前後の通信品質を測定した結果を分析し, 可視化する. 分析手法として, 障害生成前と障害生成後の 2 種類に分け, 通信品質ごとの平均値, 標準偏差, 最小値, 最大値を計算し, 表示する. また, 障害生成前後の平均値を比較した場合の影響分析や, 異なる通信品質同士の相関についても同様に計算し, 表示する. 本機能は現在, プロトタイプとして実装している.

分析・可視化機能の流れを図 7 に示す. 分析・可視化画面に遷移すると, フロントエンドからバックエンドに向けて分析機能へのリクエストが送信される. バックエンド側では, ローカルに保存されている csv ファイルにアクセスし, 測定データを JSON 形式で読み取る. その後, 読み取ったデータをもとに, Python のデータ分析ライブラリである Pandas を用いて, 2 次元のテーブル形式のデータ構造をもつデータフレームを作成する. 現在は, このデータフレームを用いて, それぞれの通信品質を分析するという仕組み

通信品質の要約統計 (障害後)

指標	平均	標準偏差	最小	最大
パケットロス率 (%)	0.000	0.000	0.000	0.000
平均RTT (ms)	0.096	0.051	0.061	0.291
TCP スループット (Mbps)	4.766	0.527	2.700	5.160
UDP ジッター (ms)	6.534	12.780	0.945	47.879
UDP ロストパケット数	50.105	4.792	22.000	53.000
UDP ロストパケット率 (%)	38.000	1.691	28.947	39.850
UDP スループット (Mbps)	9.878	1.033	3.680	10.050

障害による影響分析 (平均値の変化)

指標	変化率 (%)	絶対的な変化
パケットロス率 (%)	0.00%	0.000
平均RTT (ms)	13.70%	0.012
TCP スループット (Mbps)	-99.96%	-12832.110
UDP ジッター (ms)	1207.27%	6.035
UDP ロストパケット数	8584.91%	49.528
UDP ロストパケット率 (%)	8660.18%	37.566
UDP スループット (Mbps)	-1.70%	-0.171

図 8 分析・可視化画面(抜粋)

となる.

また，実装した分析・可視化画面の例を図 8 に示す． しかし， 分析結果をテーブル形式で表示するだけでは直感的に得られる効果として不十分であると考える． そのため，各項目をどのように表示すべきであるか検討し， 今後実装する必要がある．加えて，前述した分析手法が妥当であるかどうか精査する必要がある．現在， 新たな分析手法として， 時系列データに潜む傾向や特徴を解析する時系列分析の導入を予定している． この分析手法は， 時間ごとに測定データを集計する特徴をもつ本システムに有効であると考えており， 障害発生後の回復時間の推定や， 将来的な品質低下の予測に役立つことが期待される．今後は， どの分析手法が品質劣化パターンの分析に適しているか改めて検討する． また， 現状では分析・可視化画面に遷移することで， 本システムで測定した csv 形式のデータに直接アクセスして分析し， 表示する仕組みとなっている． しかし，いくつかの測定データを順番に分析する場合， または保管していた別の測定データを分析する場合， ファイルのインポートをするための機能が必要となる． そこで利便性の向上のため，外部から測定データをインポートし， そのデータの分析が可能となるように拡張する予定である． ここで，ローカルの測定データに直接アクセスする場合と， 外部からファイルをインポートする場合で， 測定データの読み取りフローが異なることによる数値の揺れの発生を防止するために， データの加工は最低限に留めて実装する必要がある．

4. 初期検証

本システムの動作を確認するために， 初期検証として使用したトポロジを図 9 に示す． 8 台の仮想機器で構成され， 6 つの仮想ルータのイメージには， 省リソースで検証するために FRRouting を採用する．また，各仮想機器では，OSPF による動的ルーティングが動作するように設定した．

検証の流れとして，測定機能の動作中に各障害パターン

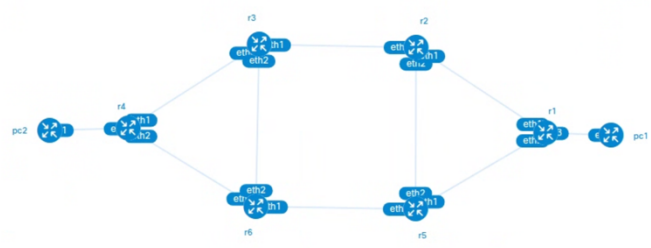

図 9 初期検証に使用したトポロジ

を単体で生成し，インターフェースの状態を確認する．複数障害生成についても同様に， 障害が生成されているかどうか確認する．また，分析・可視化画面に遷移した場合に，分析結果がシステムの不具合なく計算， 表示されることが，想定通りの検証結果となる． 初期検証の結果， 本システムを用いることで， 想定通り任意のネットワーク障害の生成と， 障害生成前後の通信品質が測定， 分析されることを確認した．その他ベンダの OS イメージを使用した場合の動作検証， およびシステムの対応については， 今後の課題として実装を進める予定である．

5. 結論

本研究では， ネットワーク障害時における通信品質の劣化の把握と， 障害対応策の検討の補助を目的に， 仮想ネットワーク環境で擬似障害を発生させ， 通信品質の測定データに基づいて品質の劣化パターンを分析するシステムを提案した．

今後は， 必要に応じて障害パターンの追加と， それぞれの生成メカニズムの改善を行い， 各障害パターンがどの通信品質の指標に影響を与えるかを検証する． また， ネットワーク構成と障害パターンの組み合わせを設定し， 複数回測定， 分析した結果の差異を比較することによる再現性実験を実施する予定である．

参考文献

[1] 総務省：“令和 6 年版情報通信白書 総務省編”，第 I 部 第 1 章 第 2 節， pp. 10， 入手先 <https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r06/pdf/00zentai.pdf>(参照 2025-06-17)

[2] 鈴木 一哉：TI-LFA を用いた IP 高速迂回路計算手法の提案，秋田県立大学ウェブジャーナル B， vol. 6，pp. 73-80(2019).

[3] 福田 充：災害時における大学の業務継続計画（BCP），大学図書館研究 117 号（2021. 3）

[4] 菊田 大輔，池内 光希，田尻 兼悟，中野 雄介：大規模言語モデルを用いたカオスエンジニアリングの自動化，日本電信電話株式会社，The 38th Annual Conference of the Japanese Society for Artificial Intelligence，2024

[5] Schalk Peach， Barry Irwin and Renier van Heerden：An Overview of Linux Container Based Network Emulation， 15th European Conference on Cyber Warfare and Security(2016)