修士論文

ネットワークの分断を想定したデータとサービスの配置戦略

中村 一貴

主指導教員 宇多 仁

北陸先端科学技術大学院大学
先端科学技術専攻
（情報科学）

令和８年３月

Abstract

Information communication via the Internet has evolved into critical social infrastructure underpinning daily life and economic activities. Under such an environment, physical network partitioning caused by large-scale natural disasters signifies a complete separation from information sources. This results in significant disadvantages for users, such as delays in ensuring safety during emergencies and the inability to access vital evacuation instructions. The "Guidebook for Introducing Disaster-Resilient Information and Communication Networks 2024," published by the Ministry of Internal Affairs and Communications, explicitly points out that the loss of communication infrastructure hinders the initial transmission of crisis information, safety confirmation, and the collection of evacuation center data. A critical issue often overlooked in disaster countermeasures is the logical dependency of services. Even if data physically exists within a partitioned local network, acquiring such content becomes difficult due to "Hidden Dependency Services." These are protocols and their interrelations required for information communication that users are typically unaware of, such as the Domain Name System (DNS). When these underlying services become unreachable due to backbone fragmentation, the entire service layer fails, rendering local connectivity useless. This study defines this phenomenon as the "Hidden Dependency" problem and positions it as a core challenge for realizing true disaster resilience. Various studies have proposed methods to secure communication during disasters. Mobile Ad-Hoc Networks (MANET) and Delay Tolerant Networks (DTN) have been widely discussed to secure communication paths without fixed infrastructure. Additionally, practical initiatives by three major mobile carriers in Japan (KDDI, Softbank, and NTT Docomo) include the rapid deployment of vehicle-mounted base stations and satellite communication services like Starlink. These efforts primarily focus on securing physical communication paths to improve accessibility. However, they do not sufficiently consider the impact of Hidden Dependency Services, often assuming that once connectivity is restored, services will function normally. Regarding Edge Computing, commercial services such as Cloudflare Workers and AWS Wavelength leverage edge locations to reduce network latency and improve efficiency under normal operating conditions. However, these architectures generally assume a healthy backbone network. Their behavior and autonomous operation during disasters, particularly when the connection to the core cloud is severed, have not been sufficiently examined. Consequently, there is a lack of established strategies for maintaining service continuity in a completely isolated local environment. This study proposes a disaster-resilient network architecture by utilizing Edge Computing and applying a novel data and service placement strategy. The primary objective is to construct a system immune to the influence of hidden depen-

dency services, ensuring that essential services and data remain accessible within a local network even when physically isolated from the global Internet. Hidden dependency services and associated challenges were analyzed by examining past large-scale communication failures and protocol stack behavior. Furthermore, dependencies were analyzed from the perspective of network topology. Consequently, three major challenges were defined: (1) Complexity of Communication Establishment via Protocol Stack: If protocols used for communication establishment, centering on DNS, stop functioning, content acquisition becomes impossible even if the server holding the content exists. (2) Congestion of Communication Bandwidth: Even if a communication path is established, traffic surges during disasters can cause severe congestion, compressing bandwidth and causing timeouts that make data acquisition impossible. (3) Centralized Topology: The current Internet arranges servers hierarchically. If upper-tier servers cannot be reached due to network partitioning, resolving domain names becomes impossible. To validate these analyses, a confirmation experiment simulated a network partition. Results confirmed that when separated from the backbone, communication to the global network fails, but local network communication is maintained. Therefore, service availability is significantly enhanced by placing content locally, provided logical dependencies are resolved locally. In the proposed placement strategy, data required during disasters was classified into three types: Type A (Data necessary for immediate evacuation, sent from Municipality to General Public), Type B (Information shared between municipalities), and Type C (Safety confirmation among citizens). Priorities were defined for these data types across three disaster phases: Initial, Emergency, and Recovery. Services were classified into "Push-type" and "Pull-type." While Pull-type services are prioritized in normal times, Push-type services should take precedence immediately after a disaster to forcibly deliver critical alerts. Based on these classifications, four system requirements were defined: (R1) Autonomous Operation, the ability to function independently without external upstream systems; (R2) Physical Proximity, placing resources within the physical reach of users; (R3) Dynamic Resource Optimization, dynamically prioritizing essential traffic (Type A) and blocking non-essential traffic to conserve bandwidth; and (R4) Information Reliability, ensuring disseminated information is trustworthy. To satisfy these requirements, this study proposes an autonomous decentralized architecture using an Edge Gateway. The core mechanism automatically switches between "Normal Mode" and "Disaster Mode." The Edge Gateway constantly monitors external network liveness. Upon detecting a disconnection from the backbone, the system autonomously transitions to Disaster Mode. In this mode, the gateway executes DNS Hijacking via Destination NAT. It intercepts all DNS lookup queries from client devices and forcibly forwards them to

the local Edge Server. This mechanism allows the user's device to resolve domain names to the local server's IP address without manual configuration changes, such as setting static IP addresses or proxies. Consequently, users can acquire disaster information replicated in advance on the local server, transparently and immediately. The effectiveness of the proposed method was evaluated through a Proof of Concept (PoC) implementation. The experimental environment used VyOS as the gateway router and Ubuntu as the edge server and client within a virtual environment. The experiment simulated a backbone network failure using Blackhole Routing. The monitoring script successfully detected the communication loss and triggered the transition to Disaster Mode. Client terminals, which lost access to external websites, were automatically redirected to the local edge server and successfully downloaded map data. This demonstrated that the architecture satisfies the requirement for autonomous operation and transparent user guidance. Based on the findings, several future challenges have been identified. First, the evaluation revealed security challenges related to HTTPS and HSTS (HTTP Strict Transport Security). Since the proposed redirection mechanism behaves similarly to a Man-in-the-Middle approach, browsers with HSTS enabled block connections due to certificate mismatches. Addressing this requires utilizing Captive Portal functions to legitimately display disaster information pages or establishing a local trust model. Second, since the current evaluation was conducted in a virtual environment, implementing the system on power-saving hardware such as Raspberry Pi is necessary to ensure operation during power outages. Finally, performance verification in actual wireless LAN environments is required, considering potential radio interference and bandwidth saturation when a large number of evacuees connect simultaneously. In conclusion, this research demonstrates that by strategically placing data and services at the edge and employing an autonomous switching mechanism, it is possible to build a disaster-resilient network that overcomes the vulnerabilities of hidden dependency services. This approach offers a practical solution for maintaining essential information flow during the critical initial phase of a disaster.

概要

現代社会において、インターネットによる情報通信は生活や経済活動を支える上で必要不可欠な社会基盤となっている。このような環境において、自然災害などによるネットワークの分断は情報からの分断を意味し、災害時における安全確保の遅れなどの利用者に対して多大な不利益をもたらす。総務省の『災害に強い情報通信ネットワーク 導入ガイドブック 2024』においても、通信インフラの消失によって、初動時の住民への危機情報の伝達や安否確認、避難所の情報収集などが困難になると指摘されている。一方で、分断されたネットワーク内にデータなどのコンテンツが存在していたとしても、DNS や証明書検証といった、利用者が普段認識しない情報通信の際に必要となるプロトコルやプロトコルの関連性といった「隠れた依存サービス」によってコンテンツの取得が困難になる可能性が高い。

災害対策に関する先行研究として、Mobile Ad-Hoc Network(MANET) と Delay Tolerant Network(DTN) を挙げた。また、先行取組として、日本国内の携帯キャリア 3 社 (KDDI, Softbank, docomo) での対策を挙げた。これらの研究や取組は、通信経路の確保に焦点を当てており、コンテンツなどの情報へのアクセス性を高めている。しかし、隠れた依存サービスの影響については充分に検討されていない。エッジコンピューティングに関する先行研究や取組では、Cloudflare や AWS Wavelength といったサービスの事例を挙げた。エッジコンピューティングの研究やサービスでは、エッジを活用したネットワークの低遅延化や効率化に焦点を当てており、災害時における動作については充分に検討されていない。本研究では、エッジコンピューティングを用いてネットワークアーキテクチャを構築した上で、隠れた依存サービスの影響を受けないデータとサービスの配置戦略を適用させることで、耐災害性のあるネットワークを提案する。

本研究では、隠れた依存サービスとそれに伴う課題の分析として、過去の通信障害の事例やプロトコルスタックによる通信の一例を紹介した。また、ネットワーク構成からも隠れた依存サービスを分析した。分析の結果として、次のような課題があると定義した。(1) プロトコルスタックによる通信確立の複雑性：DNS(Domain Name System) を中心とした通信確立に用いられるプロトコルが停止した場合、コンテンツを保持しているサーバが存在していたとしても、コンテンツ取得が不可能になる。(2) 通信帯域の圧迫：通信経路が確立できたとしても、輻輳による通信帯域の圧迫によって、コンテンツ取得までの所要時間が増加し、最悪の場合はデータ取得が困難になる。(3) 中央集権的なトポロジ：現在のインターネットは、階層的にサーバを配置している。上位のサーバに到達できなければ、最新のデータを取得することが不可能である。また、確認実験によりバックボーンネットワークと分断されると、オリジンサーバなどが存在しているグローバルなネットワークへの通信を行うことが不可能になることを確認した。また、バックボーンとの分断では、ローカルネットワークとの通信は維持できているため、コンテンツをローカルに配置することで可用性が高くなると結論づけた。

配置戦略では、はじめに本研究におけるデータとサービスの定義を述べた。災

害時に必要となるデータの分類として、ハザードマップなどの避難行動に必要な Type A(自治体発一般市民宛)、自治体間で共有される Type B、一般市民間で共有される Type C と分類した。これらのデータを初動段階、応急段階、復旧段階の 3 フェーズに対して優先度を定義した。サービスは Push 型と Pull 型に分類され、基本は Pull 型が優先されるが、災害発生直後においては Push 型が優先されるとした。配置戦略を検討するための考慮事項として、データとサービスの配置方針、コストバランス、導入期間の 3 つを挙げた。本研究では、配置戦略を実現するためのシステム要求要件として、自律運用性、物理的近接性、動的資源制御、情報信頼性が必要と定義した。

定義した要求要件を満たす配置戦略として、エッジゲートウェイによる自律的なモード切替による戦略を提案した。本アーキテクチャでは、平常時を Normal Mode、災害時を Disaster Mode としており、エッジゲートウェイにて外のネットワークとの死活監視を常時行う。バックボーン回線との断絶を検知すると、Disaster Mode へと移行する。このモードでは、Destination NAT による DNS ハイジャックを行い、ユーザの DNS 検索を強制的にローカルのエッジサーバへ転送することで、事前にレプリケーションしてある災害情報の取得を可能にする。提案手法の評価を行った結果、有効性が確認できたが、提案した手法は中間者攻撃と同様の挙動をしているため、よりセキュアな通信を行うようにすることが課題である。

今後の展望として、省電力デバイスでの実装や無線環境での評価、キャプティブポータル機能を活用した提案手法の改善が挙げられる。

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究背景

今日、インターネットによる情報通信は、生活や経済活動を支える上で必要不可欠な社会基盤となっている。そのような環境において、自然災害などによるネットワークの分断は情報からの分断を意味し、災害時における安全確保の遅れなどの利用者に対して多大な不利益をもたらす。総務省が公開している『災害に強い情報通信ネットワーク 導入ガイドブック 2024』[1] においても、通信インフラの消失によって、初動時の住民への危機情報の伝達や安否確認、避難所の情報収集などが困難になると指摘されている。また、令和 6 年 1 月に発生した能登半島地震においては、通信インフラの損傷によって孤立地域との連絡が取れない事態に陥ったと報告されている。

一方で、分断されたネットワーク内にデータなどのコンテンツと利用者が存在していたとしても、DNS や証明書検証といった、利用者が普段認識しない情報通信を行う際のプロトコルやプロトコルの関連性によってコンテンツの取得が困難になる可能性がある。本研究では、プロトコルやそれらの関連性のことを「隠れた依存サービス」と定義する。この隠れた依存サービスは災害などの非常時に顕在化することが予想されるが、平常時においても通信障害という形で表面化することが課題となる。

1.2　研究目的

本研究の目的は、現在のインターネットを基盤とした上で、災害などによるネットワークの分断や孤立が生じた場合でも最低限必要なデータの送受信やサービスの維持を可能とする配置戦略の確立および提案である。災害発生時において初動の情報伝達が最も重要である。そのため、災害発生からの経過時間にも焦点を当てて、エッジコンピューティングを活用した配置戦略を提案する。他にも、災害などの非常時において求められる要件を満たす配置戦略の考察も行う。

また、通信障害の事例や OSI 参照モデルにおける各層のプロトコルの関連性から隠れた依存サービスを明確にし、非常時における情報伝達の課題解決を行う。また、潜在的なネットワークの課題についても本研究内で指摘する。

1.3　論文の構成

本論文は以下のように構成されている。

第 2 章では、はじめにネットワークの耐災害性・耐障害性に関する先行研究や企業の行っている取り組みについて提示する。その後、本研究で用いる手法であるエッジコンピューティングに関する先行研究とエッジコンピューティングを活用した事業について提示する。それらと本研究を比較して、本研究の位置付けを明確にする。

第 3 章では、隠れた依存サービスの課題分析を行う。本章では、はじめに過去の大規模な通信障害の事例や OSI 参照モデルを参考にした分析から隠れた依存サービスおよびそれらに関する課題を明確にする。

第 4 章では、本研究の核である配置戦略について議論を行う。はじめに本研究におけるデータとサービスの定義を明確にし、データとサービスをそれぞれ分類する。その後、配置戦略を検討する上で考慮すべき点を明示し、それらを満たす要求要件を定義する。

第 5 章では、エッジコンピューティングを活用した戦略を提案し、その戦略の評価を行う。本章では、配置戦略のアーキテクチャなどの概要を提示し、配置戦略を実現させるために必要な技術と動作シナリオについて述べる。その後、PoC を用いて配置戦略どおりに動作することを確認し、提案した配置戦略について評価を行う。

第 6 章では、配置戦略について考察を行う。本章では、第 3 章で議論した要求要件を元に本研究で提案する配置戦略の他にも、要求要件を満たすような配置戦略について過去の災害に適用した場合に得られる効果などを思考実験を通して比較し考察を行う。

第 7 章では、本研究のまとめを行う。本研究の成果についてまとめた後、今後の展望について述べる。

第2章　先行研究と先行取組

本章では、本研究の主要な構成要素である「災害対策」および「エッジコンピューティング」について、それぞれの先行研究と先行取組を提示する。

2.1　災害対策に関する先行研究と取組

2.1.1　先行研究

災害による通信の途絶は大きな課題である。本節では、はじめに通信の堅牢性 (レジリエンス) を担保するための課題の分類を示す。その後、災害時に焦点を当てたネットワーク技術を紹介する。

1. レジリエンスへの課題分類

   Mauthe ら [2] は、通信ネットワークのレジリエンスは、大規模な自然災害とそれに伴う障害、天候による混乱、技術関連の災害、悪意のある人間の活動の結果としての内部の変化と外部の混乱に直面しても、同じレベルの機能を維持する能力と定義している。現在のインターネット基盤において、障害やエラーを引き起こし、最終的にシステム障害を引き起こす可能性のある事象を課題としており、その課題の分類を行っている。

   主要な課題の分類として、大規模災害、社会政治的および経済的な課題、依存関係によるエラー、人的ミス、悪意のある攻撃、異常なトラフィック、動作環境の課題が定義されている。

2. Mobile Ad-Hoc Network(MANET) を用いたネットワーク復旧

   間瀬 [3] によると、Mobile Ad-Hoc Network(MANET) とは、基地局や固定網に依存せず、移動端末を構成要素とする自律分散形のネットワークと定義されている。端末がバケツリレー的にマルチホップでデータを転送を行う。無線通信を実現する方法として、パケット交換形の無線 LAN や回線交換形の PHS、その両方をサポートする Bluetooth などが挙げられる。

   Tarasov ら [4] は、MANET を基盤とした環境において、分断されたネットワークの復旧について述べている。ネットワーク分断の検出と回復のためのアプローチとして、追加ノードを用いることを提案している。リンクの切断などの通信上の問題が発生した場合にのみアクティブにあるリアクティブ

型のシステムを採用している。問題検出のためにネットワーク内の各ノードは、IP(unicast) 通信の宛先ノードを追跡・監視している。各ノードが自身の位置情報や IP アドレスを隣接するノードに定期的にブロードキャストすることで、ブロードキャストされていないノードが分断されたと判断している。

3. Delay Tolerant Network(DTN)
   内田ら [5] によると、災害によるネットワークが機能不全となった場合の有効な手法として Delay Tolerant Network(DTN) が挙げられている。DTN によるデータ伝達は劣悪なネットワーク環境下でデータパケットを伝送するため、”store-carry-forward”プロトコルを採用している。この”store-carry-forward”とは、各ノードが近くに利用可能なノードがない場合に送信データを保存し、送信先のノードが近づくとデータを複製する方式である。この方式を用いることで、スマートフォンなどの移動する端末のノードにより、災害情報を被災地内外に伝達することが可能になる。
   この DTN を地方地域に適用する場合に、都市部に比べ車や歩行者が少ないため、メッセージの配信速度や遅延などが課題として挙げられている。その課題を解消するためにこの研究では、データトリアージや地域に基づくノード選択などを実現する拡張メディア座標系のアーキテクチャを提案している。

2.1.2 先行取組

ここでは、日本国内の携帯キャリア 3 社の対策について以下に示す。

1. KDDI の取組
   KDDI では災害への備えとして次のようなものを用意している [6]。

   - 災害時に稼働する設備
     災害時に稼働する設備として陸・海・空それぞれの移動基地局が存在する。陸では車載型や可搬型の装備が用いられており、現在は Starlink をバックホール回線として利用し復旧を迅速にしている。海では船舶を活用し、陸路からの復旧が困難な地域においては海側から沿岸地域の復旧を行なっている。空ではヘリコプターを活用しており、基地局としての役割のほか、携帯電話電波捕捉システムによる要救助者の捜索も行うことが可能である。
   - Starlink
     Starlink とは衛星通信方法の一つである。開けた場所に車載型や可搬型基地局を設置することで、au の回線を利用することが可能になる.
   - 運用システム
     東京と大阪にある拠点で運用自動化機能を活用したサービスを導入し

ている。また、通信経路の多ルート化やそれに伴う迂回、自動化によって通信経路の救済を行なっている。さらに au 災害復旧システムによる被災地の状況の把握・可視化を行い、現場作業員との円滑な連携を可能にしている。

2. Softbank の取組
   Softbank では次のような取組を行なっている [7]。

   - 輻輳回避のための通信規制
     輻輳と呼ばれる携帯電話の集中アクセスによって 119 番や 110 番といった重要回線の利用が困難になる事態を回避するために、輻輳の規模に応じて通信サービスの一時制限を行なっている。
   - 無線中継システム
     有線給電ドローンを活用することで無線中継システムを構築することが可能である。このシステムは現地到着から 30 分以内に構築することが可能であるため、迅速な復旧が可能である。
   - 移動基地局車や可搬型基地局
     移動基地局車は小型・中型・大型の 3 種類あり、現場に適した配置を行うことが可能である。可搬型基地局では、Starlink をはじめとする衛星通信が可能なアンテナを配置可能である。また、マイクロエントランスと呼ばれる、電波を遮る障害物などのない双方の基地局にパラボラアンテナを取り付け、エントランス無線を使用して電波の送受信を行うことも可能である。
   - 運用システム
     伝送路の二重化や多ルート化、主要都市を中心とするネットワークセンタの分散配置により災害に強い通信確保を行なっている。

3. docomo の取組
   docomo では次のような取組を行なっている [8]。

   - 大ゾーン基地局の設置
     通常の基地局とは別に、大ゾーン基地局と呼ばれる半径約 7km をカバー可能な基地局の設置を行なっている。
   - アンテナ角度の遠隔操作
     災害によるサービス中断時に、隣接局のアンテナ角度を遠隔から操作することでエリア救済を行うことが可能である。
   - 異周波オーバーレイ
     異なる周波数の基地局同士や無線制御装置同士をオーバーレイ (重ね合わせ) することで、1 つの施設が故障したとしても残りの施設で通信継続を可能にする。

- ノード系装置の複数帰属
  グループ化された複数ノード系装置に無線系制御装置を複数帰属させることで、一部装置で故障が生じても別の正常な装置で通信を継続することが可能である。
- 災害時データ無制限モード
  利用可能データ量の上限に到達しても速度制限がかからなくなる。

2.2 エッジコンピューティングに関する先行研究と取組

2.2.1 先行研究

本節では、はじめにエッジコンピューティングの定義とユースケースを示す。その後、エッジコンピューティングを活用した技術について紹介する。

1. エッジコンピューティングについて
   エッジコンピューティングとは、クラウドとスマートフォンや IoT デバイスなどの終端端末 (エッジデバイス) の間、特にエッジデバイス近傍にエッジサーバと呼ばれるサーバを設置し、クラウドで処理する計算などの一部をエッジサーバで処理することを可能にするアーキテクチャである。以降は、エッジサーバとエッジデバイスを合わせてエッジと述べる。
   Liu ら [9] によると、エッジコンピューティングシステムは大きく 3 つに分類できる。その分類を以下に示す。

   - クラウドからの Push
     この分野では、クラウドプロバイダーがサービスやコンピューティング (計算能力) をエッジにプッシュすることで、局所性を活用し、応答時間が短縮するため、ユーザ体験を向上させる。
   - IoT デバイスからの Pull
     IoT アプリケーションでは、IoT デバイスが生成する膨大な量のデータを処理するために、遠く離れているクラウドから近くのエッジにサービスやコンピューティングを配置する。
   - ハイブリッド
     クラウドとエッジの利点を統合することで、現代の高度なサービスやアプリケーションにおいて、グローバルに最適な結果と最小の応答時間を両立する。

   エッジコンピューティングは、一般的にデータ処理と転送のオーバーヘッドを削減し、モバイルデータ分析の効率と有効性を向上させている、と結論づけている。

2. Mobile Edge Computing

   Abbas ら [10] によれば、Mobile Edge Computing はモバイル基地局を活用してクラウドコンピューティングサービスをエッジに拡張するアーキテクチャのことである。以降は、Mobile EC と呼称する。Mobile EC は、リソース制約のあるスマートフォンなどのモバイルデバイスに、帯域幅やバッテリーの寿命、ストレージを提供する将来のエッジ技術として可能性があると述べられている。特に 5G ネットワークにおいて、高帯域幅と超低遅延を伴う計算集約型タスクを要求するアプリケーションに対し、ネットワークの末端で柔軟にリソースを提供することを目指している。

3. Multi-access Edge Computing

   Taleb ら [11] によると、Multi-access Edge Computing は通信サービスと IT サービスを統合し、無線アクセスネットワークのエッジにクラウドのプラットフォームを提供するシステムのことである。エッジコンピューティングの研究において当初はモバイルネットワークでのリソースとしてエッジを活用することが検討されていたが、Wifi などの多様なアクセス網を包含した概念として Multi-access Edge Computing が提唱されている。以降は MEC と呼称する。MEC では、エッジにストレージと計算資源を配置することで、エンドユーザのレイテンシーを短縮し、モバイルバックホールとコアネットワークをより効率的に活用することが可能である。MEC のユースケースとして、計算オフロードやキャッシュを用いた分散コンテンツ配信などが挙げられている。

   分散コンテンツ配信では CDN(Contents Delivery Network) を活用している。CDN とは、エンドユーザの近くで地理的に分散したサーバ群にコンテンツをキャッシュすることを指す [12]。これにより、ユーザの QoE を向上させることが可能である。

2.2.2 先行取組

現在商用化されているエッジコンピューティングサービスの代表例を以下に示す。

1. Cloudflare

   Cloudflare は、125 カ国以上の都市に展開されているグローバルネットワークを有し、エッジコンピューティングプラットフォームを提供している [13]。ユーザに近いデータセンター、エッジでプログラムを実行することでより高速なネットワーク通信を実現している。

2. AWS Wavelength

   AWS Wavelength は、通信事業者のデータセンターで AWS のインフラとサービスを使用してアプリケーションを実行することで、超低遅延なアプリ

ケーション配信を実現するサービスである [14]。ユースケースでも、リアルタイムアプリケーション向けの低遅延の実現例やエッジでの機械学習推論を効率的に行う事例が挙げられている。

2.3 本研究の位置付け

災害対策の先行研究や先行取組では、通信経路の確保を主眼としており、情報へのアクセス性を高めている。しかし、様々なプロトコルからなる隠れた依存サービスの影響について充分に検討されていない。

エッジコンピューティングの先行研究や先行取組では、エッジを活用した低遅延化やネットワークの効率化を主眼としている。耐障害性への対策も行われているが [13]、災害時における動作については充分に検討されていない。

本研究は、エッジコンピューティングを用いてアーキテクチャを構築した上で、データの配置方法などによる隠れた依存サービスの影響を受けない戦略を適用することで、耐災害性のあるネットワークを提案する。

第3章　隠れた依存サービスの課題分析

本章では、前述した「隠れた依存サービス」について分析する、既存のネットワーク環境において、通信を行う上で必要となるプロトコルなどの関連性などに注目し分析する。分析を行うにあたって、ケーブル切断やハードウェア故障といった物理的な要因は除外する。

はじめに、情報通信の全体像を捉えるため、OSI 参照モデルに基づいた通信フローについて整理する。図 3.1 を以下に示す。

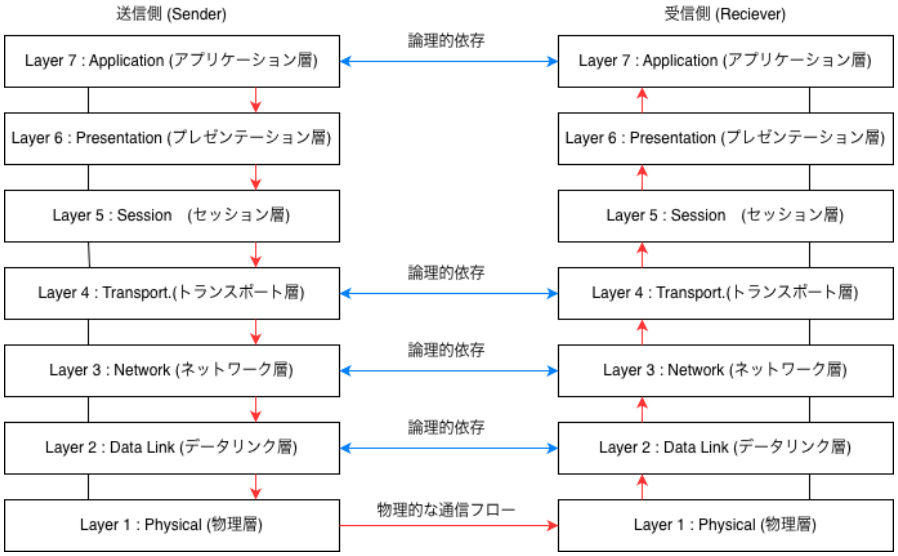


図 3.1: OSI 参照モデルと通信フロー

図 3.1 より、現在のインターネットを用いた情報通信において、OSI 参照モデルの下層から上層へ、あるいは上層から下層へと、情報通信に必要なプロトコル同士が密接に連動していることがわかる。このフローにおいて、いずれかの階層のプロトコルや処理が機能不全に陥ることで、他の階層にも影響を及ぼし、最終的に情報通信のフロー自体が途絶してしまう。

3.1 通信障害事例からの分析

前述した通り、隠れた依存サービスは、平常時に通信障害となって表面化する。本節では、過去の通信障害の事例を挙げ、それらの原因を分析することで、隠れた依存サービスの特徴を捉える。

1. 2021 年 10 月 05 日の Meta(Facebook) の全サービス停止 [15]

   - 概要

     定期メンテナンス中にグローバルバックボーンの可用性を評価するために使用したコマンドが、意図せずバックボーンネットワーク内の接続を全て切断し、Meta(Facebook) のデータセンターへの接続が全世界的に遮断された。これによって BGP(Border Gateway Protocol) による経路が全て消失し、全サービスが停止した。このようなミスを防ぐための監視ツールがバグによりコマンドを停止させることができなかったことも原因である。

   - 分析

     この障害には BGP の他にも DNS(Domain Name System) にも原因があったとされている。DNS はドメイン名 (例：`google.com`) を特定の IP アドレスに変換する。通常は DNS クエリが権威 DNS サーバによって応答される。その応答によって得られた IP アドレスは BGP によってインターネット全体に通知される。信頼性の高い運用を担保するために、DNS サーバはデータセンターと通信できない場合に、ネットワーク接続に異常があると判断し、BGP advertisement を無効化する。この障害においては、バックボーン全体が停止してしまったため、DNS サーバがネットワークに異常があると判断し、BGP による経路を全て削除した。これにより、DNS サーバは稼働していたにもかかわらず、アクセスすることが不可能になった。

     この障害から DNS と BGP の相互依存関係によって機能しなくなったことがわかる。BGP による経路制御が行えなくなったことから DNS サーバに到達できず、DNS による名前解決が不可能になった。人間が IP アドレスを複数覚えることは困難であり、また、現在のクラウドなどは IP アドレスでの直接的な通信が行えない。この BGP から DNS が機能しないことは隠れた依存サービスであるといえる。

2. 2022 年 07 月 02 日の KDDI の通信障害 [16]

- 概要
  メンテナンス作業においてルータの経路の誤設定による通信断から、位置登録要求信号の再送が大量に発生した。これにより、VoLTE 交換機や加入者 DB(Data Base) の輻輳、および加入者 DB のデータ不一致が連鎖的に発生し、通信しづらい状態に陥った。
- 分析
  この障害の原因は、VoLTE 交換機にある。メンテナンス作業中のルータの経路誤設定によって位置登録要求が破棄されてしまい、それに伴った同要求の再送が急増し輻輳が発生した。また、他拠点の VoLTE 交換機も全国中継網を介して分散処理を行なっているため、輻輳が発生した。この位置登録要求の再送ごとに加入者 DB への認証を行うため、連鎖的に加入者 DB も輻輳が生じた。
  この障害から、通信経路が正常であっても輻輳によって帯域が圧迫されてしまい、通信することが困難になることがわかる。

3. 2021 年 06 月 08 日の Fastly の障害 [17]

- 概要
  未確認のソフトウェアバグが特定の顧客のサービス設定変更で引き起こされてしまい、世界規模の障害が発生した。
- 分析
  Fastly は Managed CDN(mCDN) を用いたエッジクラウドプラットフォームを提供している [18]。Fastly のサービスは論理的には顧客それぞれに隔離して提供されているが、ある顧客の設定変更によってネットワークの大部分に障害が発生した。このことからリソースは共有されていることがわかる。リソースが共有されていることが課題ではなく、特定のリソースに不具合があった場合にその不具合も共有されてしまうことが課題である。

4. 2020 年 12 月 14 日の Google Cloud の障害 [19]

- 概要
  Google OAuth アクセスを必要とする Google サービスが利用できなくなる障害が発生した。Google User ID Service は、全てのアカウントに固有の識別子を保持し、OAuth トークンと Cookie の認証情報を処理している。このサービスを新しい割り当てシステムに移行する作業の一環で当該サービスを登録するように変更を行なったところ、過去の割り当てシステムの一部が残っており、それによって容量不足と誤認し障害が発生した。

- 分析

  この障害から、現在のインターネットサービスではユーザを特定するために認証を行なっていることがわかる。認証によってセキュリティを担保しているほか、Cookie を利用したユーザのログを活用したサービスの提供が行われている。しかし、この事例では、認証が不可能な状況に陥った場合にコンテンツの取得などが行えなくなるという課題が表面化している。コンテンツサーバが正常に稼働していても認証基盤がダウンしていればその認証を必要とするサービスに影響を及ぼすという、認証基盤の中央集権的な依存関係がある。

これらの通信障害の事例から、隠れた依存サービスについて次のような課題があると考えられる。

1. 通信経路の消失

   BGP などによる通信経路が消失した場合、インターネットを利用する上で必要となる DNS などのサーバにアクセスすることができず、サービスを利用することが困難になる。

2. 通信帯域の圧迫

   通信経路が正常だったとしても、輻輳などによって通信帯域が圧迫されてしまうと、ユーザの通信に対して遅延や途絶といった悪影響を及ぼす。

3. 共有リソース管理

   現在のクラウドサービスの中には、共有リソースを論理的に分割してユーザに提供しているものがある。論理的に分割していたとしても、物理的に共有されていればエラーも共有されてしまう恐れがある。

4. 中央集権的な認証基盤

   認証によるセキュリティの担保やユーザに適したサービスの提供を行うことが可能になっている。しかし、認証が行われなければそれに付随するサービスに対してアクセスすることが不可能になってしまう。

3.2 OSI参照モデルやネットワークトポロジからの分析

前節で述べた隠れた依存サービスの課題について、OSI 参照モデルを考慮した上で分析する。また、現在のインターネット基盤での階層構造などといったトポロジの観点からも分析する。

3.2.1 プロトコルスタックによる隠れた依存サービスの分析

図 3.1 に示したように、インターネットでの情報通信を行う際には各層のプロトコルが上から下、下から上へと関連している。また、アプリケーション層やネットワーク層のように、送信側と受信側とが同じ階層のプロトコルを用いているものもある。一般的なブラウザの動作フローを図 3.2 に示す [20]。ブラウザの動作フローは次のように段階を踏んで行われている [20]。

1. DNS 検索

   DNS 検索では、Web ページがどのサーバに存在するかを検索することを行なっている。例えば、`https://www.example.com` へアクセスする場合、IP アドレスが `123.123.123.123` のサーバにページが存在しているとする。この時、ブラウザがサイトに一度も訪れたことがなかった場合に DNS 検索が必要となる。ブラウザが DNS 検索をリクエストし、DNS サーバが IP アドレスを返す。このリクエスト後は、IP アドレスはキャッシュされる。

2. 通信の確立

   DNS 検索によって IP アドレスが判明した後は、TCP three-way handshake によってブラウザとサーバとの通信を確立する。TCP three-way handshake では、ブラウザとサーバとの間で 3 回のメッセージ送受信を行うことで通信が確立したことを判断している。

3. TLS ネゴシエーション

   HTTPS を用いた通信では TLS ネゴシエーションが行われている。TLS ネゴシエーションでは、通信の暗号化に使用する暗号の種類を決定し、サーバを認証し、実際のデータ送信の開始前に安全な通信の準備を行なっている。

4. 応答

   Web サーバとの通信が確立されると、ブラウザはユーザの代わりに最初の HTTP `GET` リクエストを送信する。リクエストを受信したサーバは、適当なレスポンスヘッダーと HTML のコンテンツを送信する。

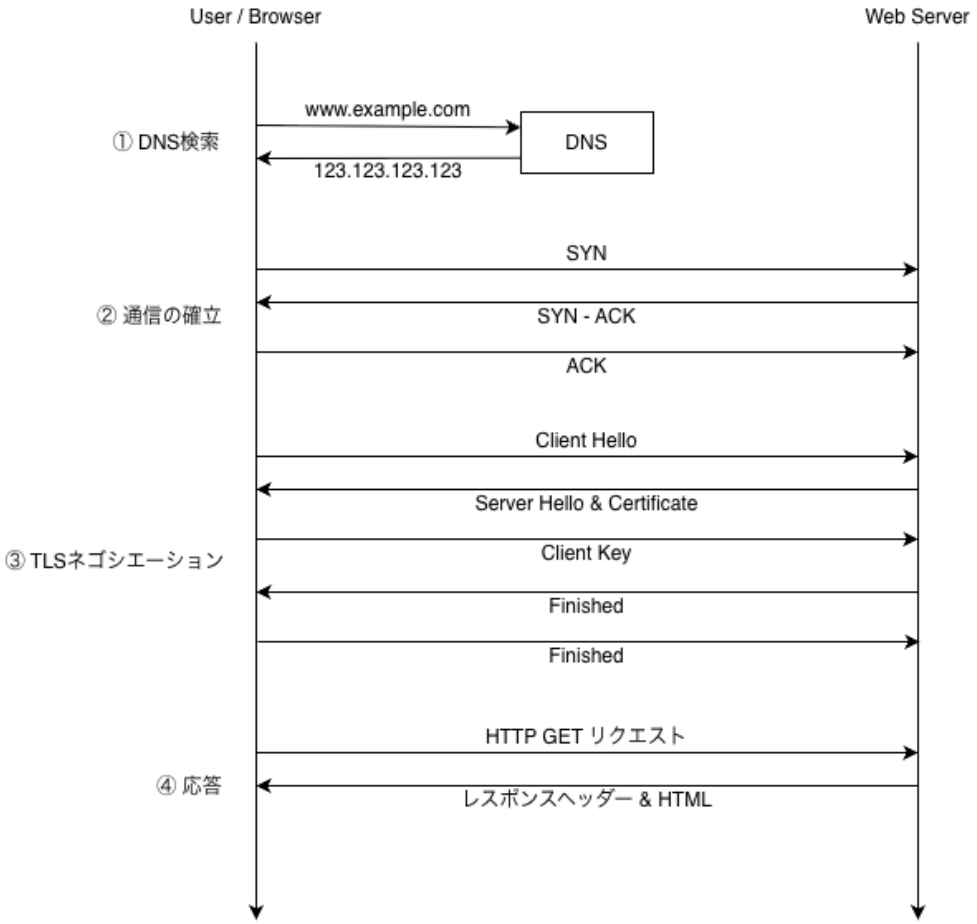


図 3.2: ブラウザの動作フロー

このようにユーザがブラウザを利用する際には、複数のプロトコルが用いられていることがわかる。

今回はブラウザの事例を示したが、メールを利用するならばアプリケーション層で利用されるプロトコルが変わるだけで、DNS 検索や通信の確立はブラウザと同様に行われている。したがって、隠れた依存サービスの課題は、DNS 検索や通信の確立に用いられているプロトコルの一つが機能しなくなることで他のプロトコルにも連鎖的に影響を与え、情報通信を行うことが困難になることである。特に DNS の機能不全が大きな課題になると考えられる。ブラウザの動作フローで示したように、特定のサーバへアクセスするためには IP アドレスを知る必要がある。前節でも述べたように、IP アドレスを人間が覚えることは困難であるため、人間にとってわかりやすいドメイン名が用いられている。DNS が機能しなければ、そのドメイン名を利用してサーバへアクセスできずコンテンツの取得が行えない。このことから、DNS を中心としたプロトコルスタックが隠れた依存サービスの課題となる。

3.2.2　トポロジの隠れた依存性の分析

現在のインターネットはサーバを階層的に配置している。前述したDNS検索についても、DNS権威サーバと呼ばれる上位のサーバへDNS検索のリクエストを送信し、その結果をユーザ近傍のDNSキャッシュサーバにキャッシュしておくことでDNS検索の時間短縮を図っている。また、CDNも元々コンテンツのあるオリジンサーバからユーザ近傍のサーバへコンテンツをキャッシュしている[12]。

これらのことから、トポロジからわかる隠れた依存サービスの課題は2つある。

1. 上位サーバとの分断
   前述した通り、コンテンツを取得する際のリクエストは、上位のサーバへと転送され、応答結果を下位のサーバへとキャッシュを行う。そのため、新しいリクエストを送る際に上位サーバと分断されていれば、新しいコンテンツを取得することが不可能になる。

2. キャッシュの有効期限
   上位サーバと分断されていてもキャッシュがあれば、そのキャッシュに対応したコンテンツは取得することが可能である。しかし、キャッシュには有効期限(TTL(Time to Live))が存在しており、TTLが切れればキャッシュを利用することが不可能になる。したがって、キャッシュにおいても時間切れにより可用性が低下すると言える。

3.3　隠れた依存サービスの課題についてのまとめ

隠れた依存サービスの大きな課題について、本節でまとめを行う。

1. プロトコルスタックによる通信確立の複雑性
   DNSを中心とした通信確立のためのプロトコルが機能不全に陥った場合、データなどを含むコンテンツのあるサーバが存在していたとしてもコンテンツ取得が不可能になる。

2. 通信帯域の圧迫
   通信経路が確立したとしても、輻輳による通信帯域の圧迫によってデータ取得までの所要時間が増加、最悪の場合にデータ取得が不可能になる。

3. 中央集権的なトポロジ
   現在のインターネットにおけるサーバの配置は階層的である。上位のサーバへと到達できなければ新しいデータを取得することが不可能である。
   また、認証を要求されるサービスに対して、認証が行えなければサービスを享受することが不可能である。

3.4　ネットワークを分断する確認実験

前述した通り、隠れた依存サービスはプロトコルスタックやトポロジに起因するものが多い。しかし、災害時においてバックボーンネットワークとの分断による上位ネットワークや上位サーバへの到達が困難になることが最も大きな課題になると予想する。

本研究では、物理的なネットワーク切断を行わずに、パケットフィルタリングを用いて論理的にバックボーン回線の断絶を再現する実験環境を構築した。作成した実験環境を図 3.3 に示す。

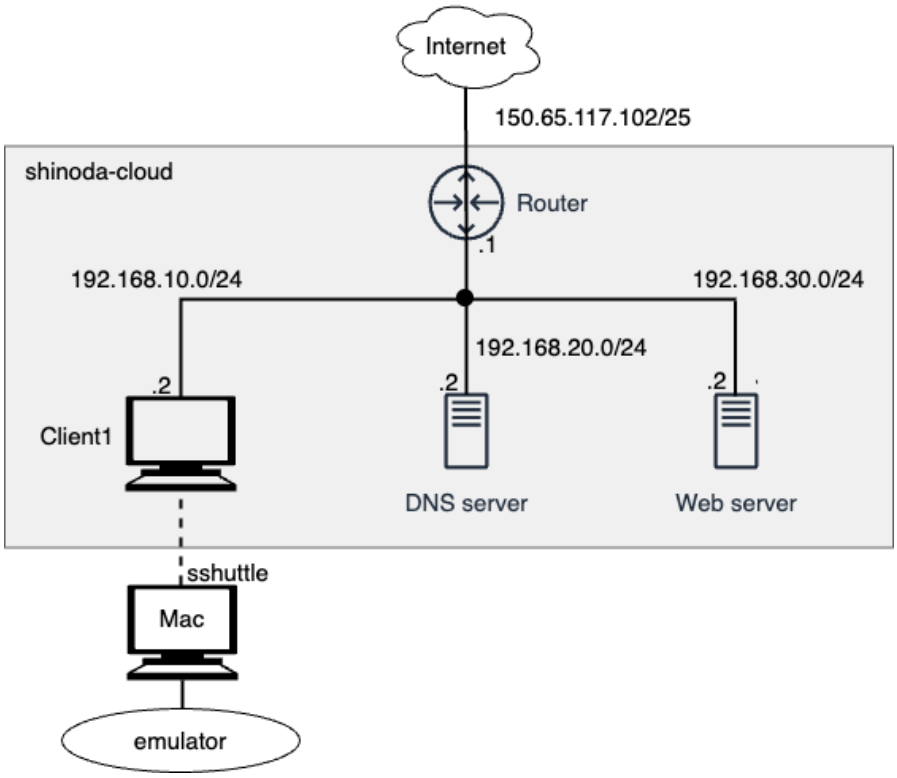


図 3.3: 作成した実験環境

3.4.1 ネットワーク構成

実験ネットワークの構成を表 3.1 に示す。

表 3.1: 確認実験における実験環境の構成

デバイス名	OS / ソフトウェア	IP アドレス
Router	VyOS	外側：150.65.117.102 Client1 側：192.168.10.1 DNS 側：192.168.20.1 Web 側：192.168.30.1
Client1	Ubuntu 24.04 server, UFW	192.168.10.2
DNS server	Ubuntu, BIND9	192.168.20.2
Web server	Ubuntu, Nginx	192.168.30.2

3.4.2 バックボーン断絶の再現手法

バックボーンネットワークとの断絶を再現するため、Client1 にて UFW を用いたパケットフィルタリングを行なった。具体的には、ローカルネットワーク（192.168.0.0/16）宛ての通信のみを許可し、それ以外の全アドレス宛ての通信を破棄するホワイトリスト方式を採用した。適用したフィルタリングルールを以下に示す。

```
# Client1 での UFW 設定

# 1. デフォルトですべての外部への通信を拒否
sudo ufw default deny outgoing

# 2. ローカルネットワーク（192.168.0.0/16）への通信だけは許可
sudo ufw allow out to 192.168.0.0/16

# 3. 設定の有効化
sudo ufw enable
```

この設定により、Client1 は、物理的な回線が接続されていても、論理的にインターネットから隔離された状態になる。

3.4.3 クライアント接続とプロトコル制御

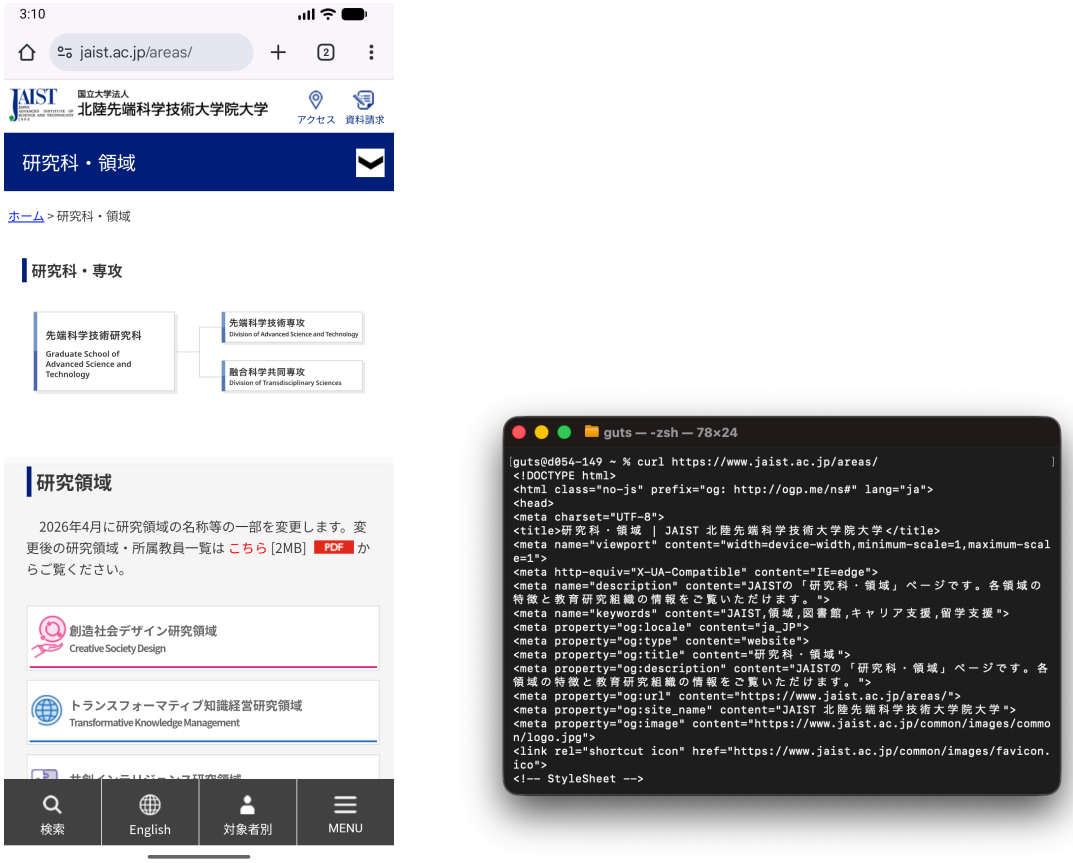
ホストマシン (Mac) の Android Studio 上で動作するエミュレーターを Client1 に対して sshuttle で接続することで、論理的に実験環境内にスマートフォンを配置し

た。なお、実験においてブラウザである Google Chrome が使用する QUIC が TCP トンネリングをバイパスして直接外部ネットワークへ流出する現象が確認されたため、ブラウザ設定にて QUIC を無効化し、全ての HTTP/HTTPS 通信を TCP 経由で行うよう制限した。また、IPv6 による通信漏洩を防ぐため、Mac のネットワークスタックを IPv4 のみに制限した状態で検証を行った。

3.4.4 確認実験の結果

構築した環境において、エミュレーター上の Web ブラウザおよび `curl` コマンドを用いて分断前後の確認実験を行った。

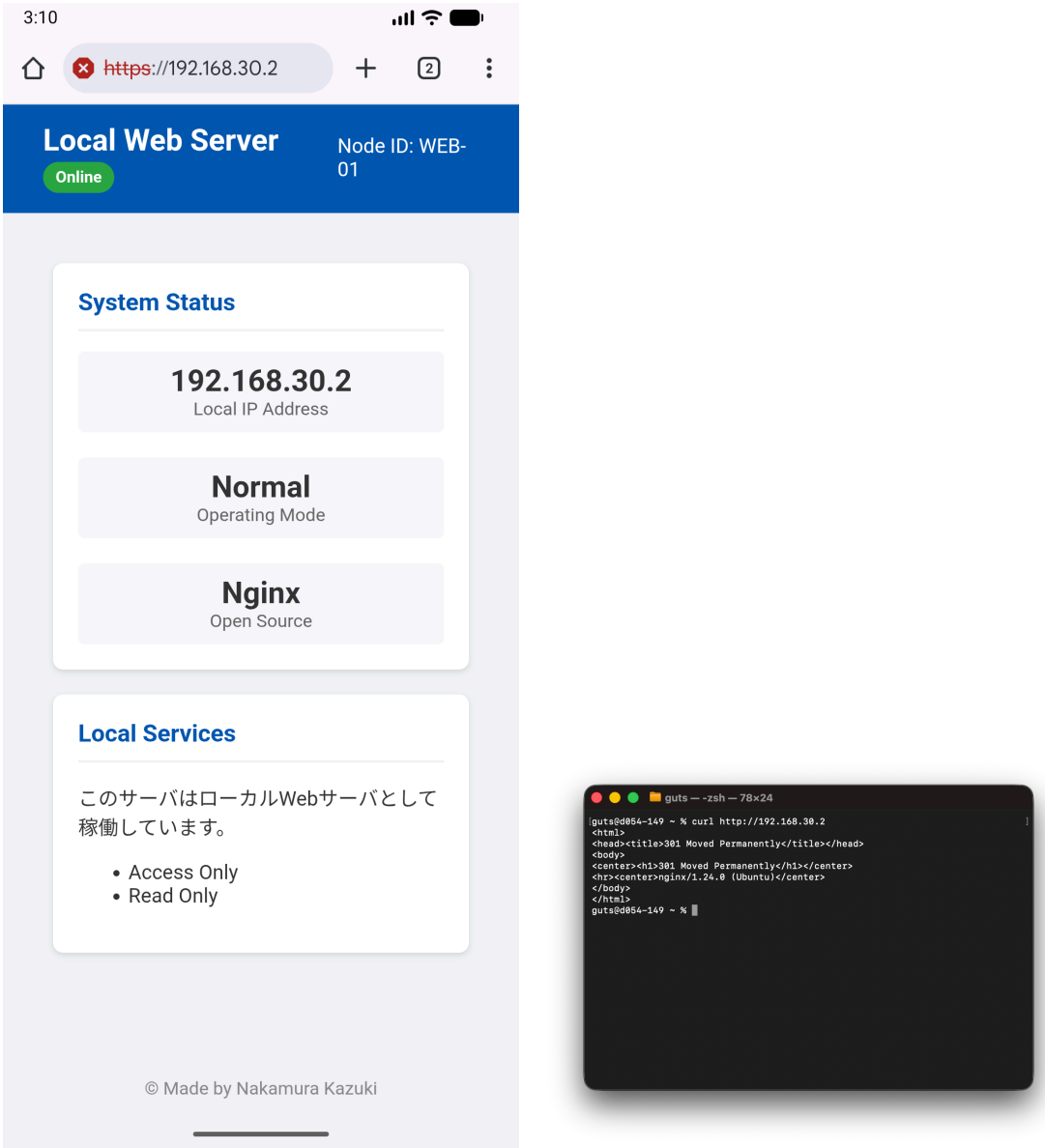
はじめに分断前の挙動の確認を行なった。図 3.4 は、外部ネットワーク (`https://www.jaist.ac.jp/areas/`) へのアクセスを行なった際の挙動である。エミュレータや `curl` コマンドの出力結果から、正常に通信が行われていることが確認できる。



(a) エミュレーター　　(b) curl コマンド

図 3.4: 外部ネットワークと通信する平常時の挙動

図 3.5 は、作成した環境内 (ローカルネットワーク) へのアクセスを行なった際の挙動である。外部ネットワークと同様に正常に通信が行われていると言える。図 3.5(a) において、`https` が赤い横線で消されている理由は、Web server は自己署名証明書を用いているためである。また、図 3.5 において、`301 Moved Permanently` と出力されているのは、HTTP でアクセスしているためである。HTTP から HTTPS にリダイレクトされているため出力された。

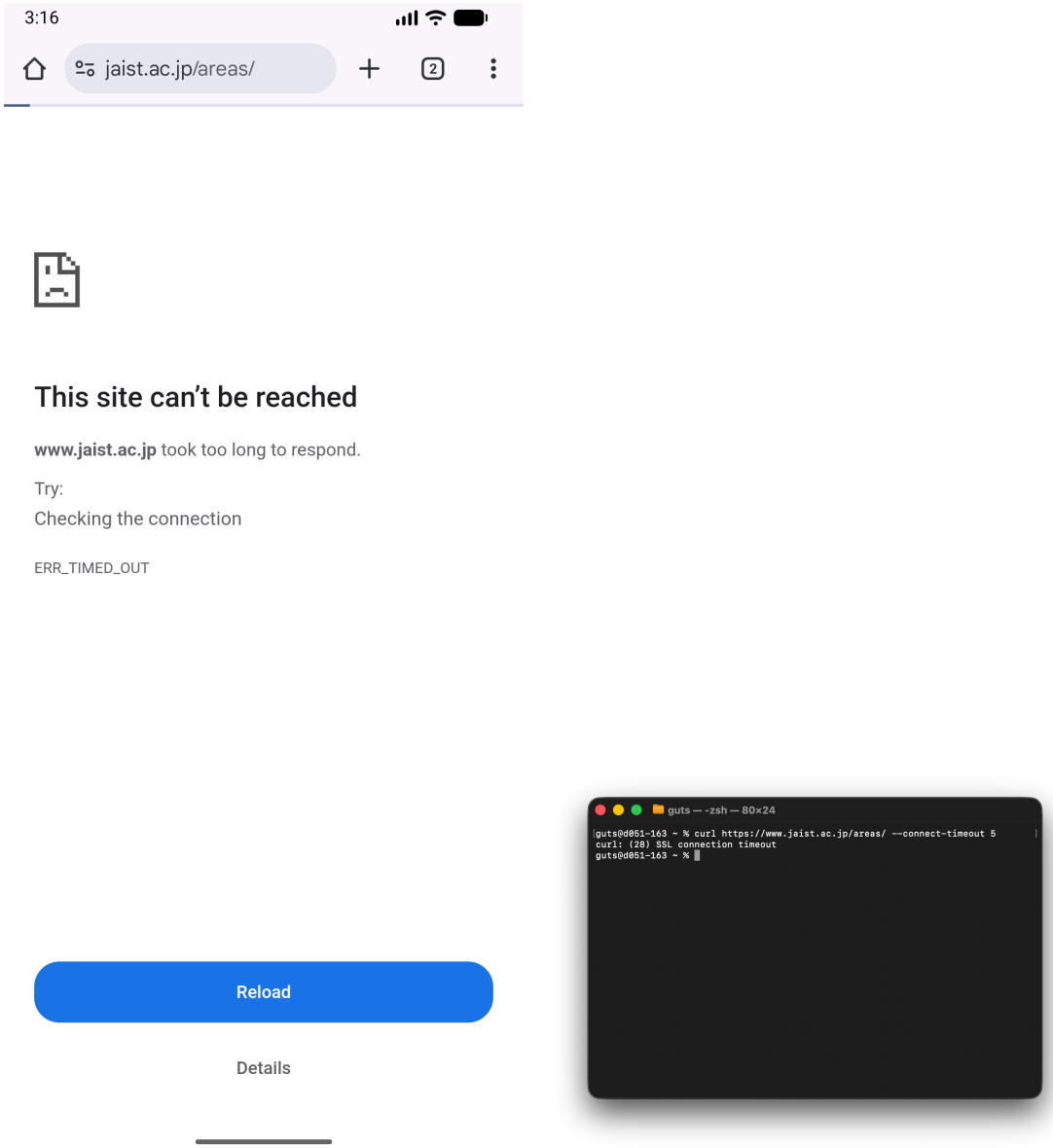


(a) エミュレーター　　(b) curl コマンド

図 3.5: ローカルネットワークと通信する平常時の挙動

次に前述したパケットフィルタリング適用下での挙動を確認した。図 3.6 にバックボーンネットワークと論理的に分断した結果を示す。分断前と同じ外部ネットワー

クへのアクセスを行ったが分断されているため、図 3.6(a) では ERR_TIMED_OUT、図 3.6(b) では (28) SSL connection timeout と出力されている。このことから、外部ネットワークと分断されておりリクエストの有効時間切れが確認できた。



(a) エミュレーター　　(b) curl コマンド

図 3.6: 分断時の外部へのアクセス挙動

同様に分断下でのローカルネットワーク内通信の挙動を図 3.7 に示す。バックボーンネットワーク分断前後で出力結果に影響がないことが確認できた。また、図 3.7 内で Normal Operating Mode や Online になっているのは Web ページ作成時の仕様であり、ネットワークの監視を行っているわけではない。

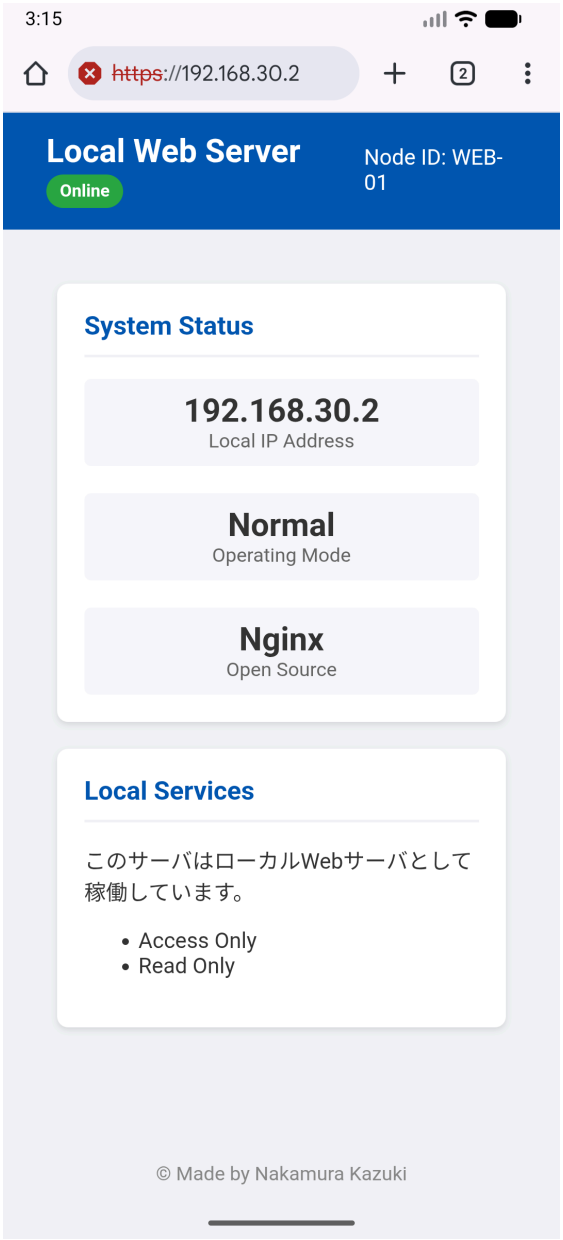


図 3.7: 外部との分断した時のローカルネットワーク内通信の挙動

この確認実験の結果から、バックボーンネットワークと分断されてしまうと外部へのアクセスが困難になり、データの取得や様々なサービスの享受することが不可能になることが明らかになった。しかし、確認実験の結果からローカルネットワークはバックボーンネットワークとの分断下においても利用可能な通信資源となることを示唆している。この結果を元に、次章以降ではローカルネットワークという資源を活用することも考慮すべきと結論づけた。

第4章　配置戦略に関する議論

4.1　データとサービスの定義

本節では、本研究におけるデータとサービスの定義について述べる。また、各定義内にて本論文で用いている語句の分類も明示する。

データ

サービスを通じてユーザが消費する情報そのもののことを指す。本研究にて用いているコンテンツという用語もユーザが利用する情報のことであるため、データという定義に含まれる。具体的には、文字データや画像データなどが挙げられる。

サービス

情報を送受信し、それらの情報を処理するための機能や仕組みのことを指す。本研究で定義している、隠れた依存サービスはサービスに該当しており、情報通信を行う上でのプロトコルなども含まれている。具体的には、サーバや配信プラットフォームのアプリケーションなどが挙げられる。

4.2 データの分類

本研究における非常時のデータの分類について、図 4.1 に示すような判断基準によって分別を行う。

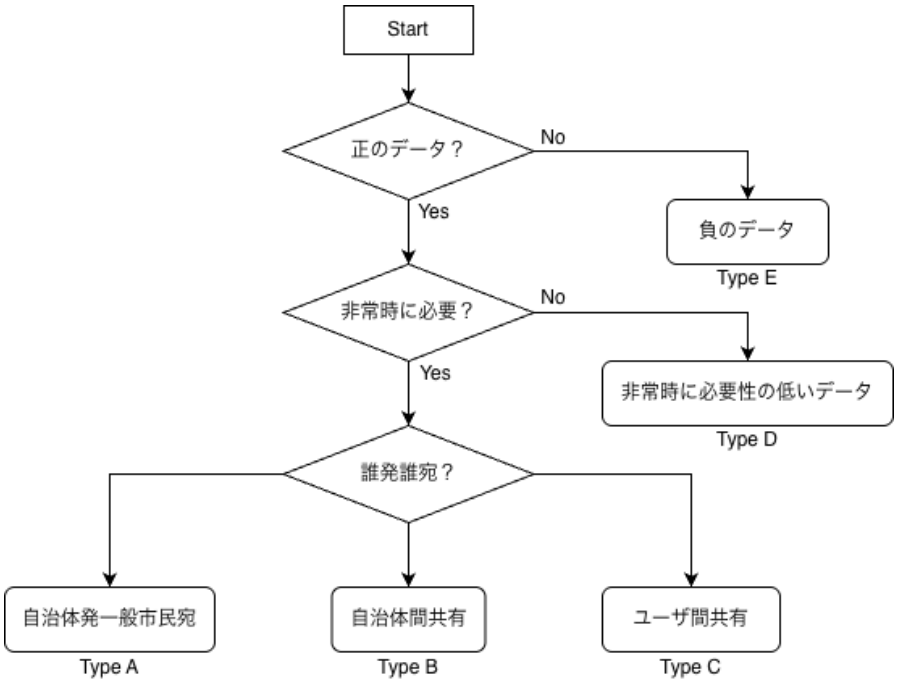


図 4.1: データの分類

1. データの正負
   データは正のデータと負のデータとの二つに大別することが可能である。

   正のデータとは、大多数のユーザにとって有効に利用可能なデータと定義する。また、ユーザに対して悪影響を及ぼす可能性が低いデータも正のデータとする。例えば、一方のユーザが利用している画像データは他方のユーザは関心がないとしても、他方のユーザに対してそのデータは悪影響になる可能性が低い。そのようなデータを正のデータとして枠組みに含めている。

   反対に負のデータはユーザにとって有害なデータを指す。ユーザ間の衝突や混乱の原因となる可能性のあるもの、悪意を持った人間による改竄されたデータが負のデータに該当する。例えば、事実無根のデマや流布された誤情報などが挙げられる。流布された誤情報について、ユーザの善意によって伝搬することがある。誤情報については、情報源が信頼できる機関や組織から発信されているかどうかを判断基準とする。つまり、災害発生時などの非常時において、一次情報源は自治体や気象庁になるため、各機関から発表された情報と照会し異なる情報があれば誤情報として扱う。

データの正負について、中間に位置する「どちらでもない」情報という枠組みを作る必要性についても検討した。ここでの「どちらでもない」情報とは、データの取得や利用前において、危険性を判断するのが困難なものを指す。負のデータの中には、デマのようにユーザが受信・発信しなければ有害性は認められない可能性が高いデータが存在している。どちらでもないという枠組みを設けることで負のデータが潜伏し流通する可能性があると考え、今回は不採用とした。

2. 非常時における必要性の有無

正のデータの中にも、災害などの非常時において必要性が高いデータとそうでないデータに二分される。

非常時において必要性の高いデータとは、避難情報が記載されている防災マップやハザードマップ、避難所の情報のことを指す。防災マップとハザードマップはどちらも避難情報が記載されているが、ハザードマップは災害による被害の程度もマップ内に含まれているため、災害別の避難行動が要求される場合はハザードマップの方がより有効となる可能性が高い。

反対の非常時に必要性の薄いデータは、生命に直結しない情報のことを指す。非常時において優先されるべき情報とは、生命に直結する情報である。非常時の娯楽目的の情報について必要性が皆無なわけではなく、相対的に必要性が薄くなるということである。そのようなデータをこちらに区分する。

3. 送信元と宛先

非常時において必要なデータについて、データの送信元と宛先が、自治体などの運営者か、ユーザのような一般市民かによって大きく三つに分類できる。

自治体発一般市民宛データとは、災害発生直後の初動段階において、一般市民の避難行動や生命の安全確保に直結するデータのことを指す。本研究では、このような非常時にこそ提供されるべき、避難行動に不可欠なデータを本区分として定義する。

自治体間共有データとは、避難所内の避難状況などのリアルタイム性の高いデータのことを指す。このようなデータは、近隣の避難所が互いの避難所の状況の共有や、自治体が発足する災害対策本部への情報共有を目的としている。

ユーザ間共有データとは、安否確認のようなデータのことを指す。このようなデータは、避難完了した一般市民間でのみ利用されるものとしている。

これらの分類法により、本研究でのデータは5つに分類される。

1. Type A
   Type A データとは、自治体発一般市民宛データのことである。災害発生直後に行われる避難行動に直結する優先度の高いデータをここに区分した。

2. Type B
   Type B データは、自治体間共有データである。近隣の避難所同士や災害対策本部の情報共有といった、運営者にとって重要なデータをここに区分した。

3. Type C
   Type C データは、ユーザ間共有データである。避難行動を完了した一般市民間で行われると予想される安否確認に関するデータをここに区分した。

4. Type D
   Type D データは、平常時に利用されるデータ群のことである。

5. Type E
   Type E データは、ユーザに対して悪影響を及ぼす可能性のある負のデータ群のことである。

分類したデータと時間による関係性を表した表を以下に示す。

表 4.1: 各フェーズにおけるデータタイプの優先度と利用可否

データタイプ	平常時	初動段階	応急段階	復旧段階
Type A (自治体 → 市民)	○	◎	○	△
Type B (自治体間共有)	×	○	◎	○
Type C (市民間安否)	×	△	◎	○
Type D (平常時データ)	◎	×	△	◎
Type E (負のデータ)	×	××	×	×

記号の意味: ◎=最優先/必須, ○=利用可, △=帯域制限/低優先, ×=利用なし, ××=厳格な遮断

ここでの時間区分は 4 つに分けられる。

1. 平常時
   平常時とはその名の通り、災害などの非常時ではない状態のことである。平常時下において、利用されるデータは Type D が最も多く、次点で Type A だと定義する。Type D が最も多くなる理由として、日常生活を送る上で避難行動のことを想定して行動する一般市民は多くないためである。次点で Type A データが利用されるのは、避難訓練などの防災・減災を意識した活動が各自治体によって行われているためである。Type B および Type C データは災害発生後にのみ使われるため、ここでは利用されない。

2. 初動段階
   初動段階とは、災害発生直後から 72 時間以内の状況のことである。初動段階において、Type A データが最も多く利用され、Type B も利用される。Type A が最も多く利用される理由は、災害発生直後に一般市民は避難行動を行うためである。これにより、避難行動に関係するハザードマップなどの情報や危険なエリアの情報を入手することが多くなる。災害発生直後から自治体は災害対策本部を発足させるため、逐一状況の共有や把握が必要となるため、Type B も利用される。

3. 応急段階
   応急段階とは、災害発生後の時間経過で 3 日から 1 週間以内の状況のことである。この状況下では多くの一般市民が避難行動を完了していると予想される。そのため、避難行動に直結する Type A データの利用量が減り、Type B データによる運営者の情報共有が主になる。また、一般市民間の安否確認も行われ始めるため、Type C データも利用されると予想できる。

4. 復旧段階
   復旧段階とは、災害発生から 1 週間以上経過した状況のことである。この状況下では、被災状況の取りまとめも目処がつき、一般市民も日常生活への復帰を始める。そのため、Type B データや Type C データの利用はあるが、前段階に比べて利用量が小さくなる。また、日常生活への復帰が行われるため、Type D データの利用が増える。

Type E データについては、どの時間・段階においてもユーザが取得できないようにすることが重要である。特に、初動段階におけるデマの流布は厳重に対策する必要がある。災害発生直後に冷静な判断を下すことが可能な一般市民は多くはない。そのよう状況下において、デマのような事実と異なる情報が共有されると混乱を招き、正常に避難行動を行うことが不可能にある。その結果、助かる命を救うことができなくなる可能性が高くなる。この課題を解決するために、特に初動段階は SNS などの情報源が曖昧なサービスの利用を制限すべきである。

4.3 サービスの分類

サービスは 2 つに分類できる。

1. Push 型サービス
   Push 型サービスとは、ユーザが受動的にデータを取得するサービスのことである。Push 通知と呼ばれるような通知方法は push 型サービスの代表例である。

2. Pull 型サービス
   Pull 型サービスとは、ユーザが能動的にデータを取得するサービスのことである。多くのサービスがこの Pull 型に属している。

この 2 つに分類したサービスについて、平常時と非常時においてどちらが優先されるかを次のように定義する。

1. 平常時
   平常時において、push 型より pull 型サービスの方が優先される。しかし、push 型が必要ないわけではなく、利用される頻度が低いということである。また、近年では災害発生直前にアラートが出ることがある。このアラートに関してはどの pull 型サービスよりも最優先で扱われるべきである。

2. 初動段階
   初動段階において、push 型サービスが優先される。災害アラートが災害発生直前に行われることがあるが、全てがそうであるとは限らない。日本では、海外からの津波の影響を受けることであるため、このような場合で push 型が優先される。push 型での通知が完了した後に、ユーザがデータを取得するという流れを想定しているため、pull 型に比べ push 型が優先されている。

3. 応急段階および復旧段階
   応急段階および復旧段階においては、平常時と同様に pull 型サービスの方が優先される。ただし、地震では余震の影響もあるため、災害アラートのような push 型サービスが最優先である。

4.4　配置戦略における考慮事項

配置戦略の確立や策定を行う上で、考慮すべき事柄が大きく分けて 3 つ存在していると定義する。

4.4.1　データとサービスの配置方針

平常時と非常時のフェーズ変化に伴い、求められるサービス形態（Push 型/Pull 型）とデータの可用性は変動する。

1. **平常時およびスタンバイ状態**
   平常時において、ユーザは娯楽や日常業務のために Type D データ（平常時データ）を能動的に取得するため、通信需要は Pull 型サービスが主体となる。しかし、災害は突発的に発生するため、緊急地震速報や J アラートのような「最優先の Push 型サービス」は、平常時であっても即座に稼働できるスタンバイ状態を維持しなければならない。

2. **非常時（発災後）の需要変化**
   災害発生直後の通知（Push 型）が完了した後、ユーザは避難行動や安否確認のために自ら情報を求め始める。したがって、非常時の主たる通信フローは、再び Pull 型（Type A データの取得等）へと移行する。ただし、地震災害における余震や、気象災害における状況急変の可能性があるため、最優先の Push 型サービスは決して停止させてはならず、Pull 型サービスと並行して常に割り込み可能な状態を維持する必要がある。

3. **時間的制約**
   発災後 72 時間は、いかなるネットワーク分断状況下であっても、避難に直結する Type A データへのアクセスが保証されなければならない。そのため、配置戦略においては、バックボーン復旧を待つのではなく、ローカル環境内でこれらを自律的に提供し続ける持続性が求められる。

4.4.2　コストバランス

エッジサーバの配置には、金銭、土地、人材という 3 つのコスト要因が存在し、これらは機能要件とトレードオフの関係にある。

- **金銭的コスト**
  高性能なサーバを導入すれば、多くの機能や大量のデータを配置可能だが、導入・維持コストは増大する。災害対策は利益を生みにくい公共的な側面が強いため、これらのコストを自治体が負担するのか、通信事業者が負担する

のかという主体の問題も発生する。したがって、必要最低限の機能に絞ったスリムな構成が求められる。

- **土地的コスト**
  物理的近接性を満たすためにユーザ近傍（基地局や避難所）にサーバを設置する場合、設置スペースの確保が課題となる。既存の施設に併設するのか、あるいは専用のハウジングを新設するのかによって、金銭的コストも変動する。特に都市部においては物理的なスペース確保自体が困難な場合があり、省スペース性が要求される。

- **人材的コスト**
  エッジサーバの運用・保守には専門的な知識を持つ技術者が必要となる。自組織で技術者を育成・確保するのか、外部へ委託するのかによってコスト構造と運用体制が変化する。特に、高度な設定が必要なシステムでは特定の技術者に依存する「属人化」のリスクが高まるため、可能な限り自律制御を行い、人的介入を減らす設計が望ましい。

4.4.3 導入期間とフェーズ戦略

配置戦略の社会実装には、技術成熟度とインフラ整備状況に応じた 3 つの期間区分が存在する。

1. **短期的戦略**
   既存の汎用サーバや仮想化技術、および現在の通信規格の範囲内で実現可能な戦略である。大規模なインフラ刷新（サーバの新設等）を行わず、ソフトウェア制御のみで対策を行うため、即効性が高い。

2. **中期的戦略**
   既存技術と将来技術の橋渡しとなる期間である。このフェーズでは、物理的なエッジサーバの増設や、自治体・事業者間での運用ルールの策定など、設備投資と制度設計が行われる。前述のコスト問題の解決が必須となるため、3 つの期間の中で最も長期化し、かつ停滞するリスクが高いフェーズであると予想される。この期間を乗り越えなければ、完全な配置戦略の実現は不可能である。

3. **長期的戦略**
   次世代通信規格（6G 等）の標準化や技術革新を見据えた戦略である。災害時の自律分散制御がプロトコルレベルで標準化されることで、個別の対策が不要となる可能性がある。しかし、標準化の成否は不確実であり、それを見越して先行投資を行う組織がどれだけ現れるかは未知数である。このフェーズで理想的な実装がなされれば、隠れた依存サービスの問題は根源的に解決される可能性がある。

4.5 要求要件

前節までの議論および配置戦略における考慮事項を踏まえ、隠れた依存サービスを排除し、災害時の通信可用性を担保するためのシステム要求要件を定義する。本研究では、要件を「R1: 自律運用性」「R2: 物理的近接性」「R3: 動的資源最適化」「R4: 情報信頼性・安全性」の 4 つのカテゴリに分類し、それぞれについて満たすべき詳細要件を表 4.2 にまとめる。

表 4.2: 配置戦略を実現するためのシステム要求要件

ID	大項目	詳細要件
R1	**自律運用性**	• **R1-1 依存の排除:** 上位 NW やクラウドが途絶しても、サービスとデータが単独で動作可能であること。 • **R1-2 永続性の担保:** 一時的なキャッシュ(TTL 依存) ではなく、レプリケーションによりデータを永続保持すること。 • **R1-3 トポロジからの脱却:** DNS 等の既存インフラに依存せず、ローカル内で名前解決等が完結すること。
R2	**物理的近接性**	• **R2-1 バックボーン非依存:** 物理回線切断時でも、既存端末 (スマホ等) から L1 レベルで接続維持できる場所に配置すること。 • **R2-2 ローカル配置:** 避難所や基地局近傍など、ユーザの生活圏内に配置すること。
R3	**動的資源制御**	• **R3-1 フェーズ適応:** 平常時 (Pull 優先) と非常時 (Push 優先) で、サービスの優先度を動的に変更できること。 • **R3-2 帯域保護:** Type D を遮断し、有限な帯域を Type A/B のために確保すること。
R4	**情報信頼性**	• **R4-1 負のデータの排除:** Type E(デマ・攻撃) の流入を入口で阻止すること。 • **R4-2 信頼の閉域化:** アクセス先を信頼できるローカルリソースのみに限定し、攻撃の余地を排除すること。

要求要件と提案手法の対応関係

次章で提案するシステムは、これらの要件を以下の技術的アプローチによって解決するものである。

R1 自律運用性への対応

エッジサーバに配置するデータ形式を、有効期限のあるキャッシュではなく、オリジンサーバにあるデータの完全な複製とする。これにより、外部との同期が取れない孤立状態であっても、TTL 切れによるデータ消失を防ぎ、サービスを継続提供する。

R2 物理的近接性への対応

エッジサーバの設置場所を、ユーザ端末が直接電波を送受信可能な携帯電話基地局（eNodeB/gNodeB）の近傍あるいは避難所の Wi-Fi アクセスポイント直下とする。これにより、バックボーン回線が物理的に破断しても、無線区間のみで通信が成立する環境を構築する。

R3 動的資源最適化への対応

ゲートウェイルータにおける DNS 誘導を活用する災害発生時には外部への不要なトラフィック (Type D データ等) をネットワークの入口で物理的に遮断することで、有限な無線 LAN 帯域を Type A/Type B/Type C(避難情報) の配信のみに占有させる資源制御を行う。

R4 情報信頼性への対応

DNS ハイジャックを用いることで、ユーザのアクセス先を OS レベルで強制的にローカルのレプリケーションサーバへと誘導する。これにより、デマが蔓延する外部 SNS 等への接続を物理的に遮断し、信頼できる自治体情報のみを提供する閉域網を実現する。

第5章　エッジコンピューティングを活用した戦略

5.1　アーキテクチャの概要

本研究で提案するエッジコンピューティングを活用したアーキテクチャを図5.1に示す。

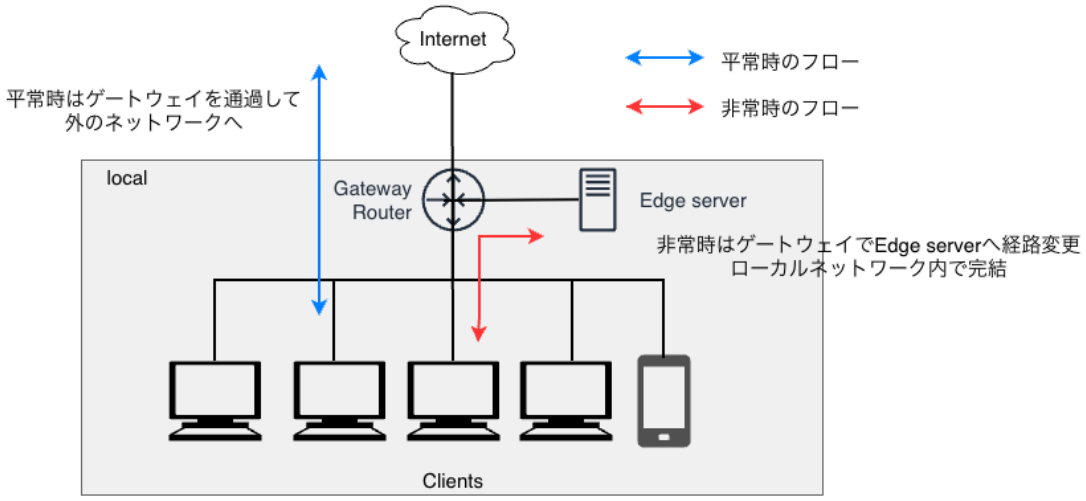


図 5.1: 提案アーキテクチャの概要

このアーキテクチャの根幹となる設計思想は、「緊急時におけるエッジサーバの自律的な権限昇格」である。平常時のインターネットにおける、上位の権威サーバやオリジンサーバなどの階層構造や隠れた依存サービスを、災害時において意図的に無視・遮断すべきという考えに基づいている。具体的には、次のようなアプローチを採用する。

1. **依存関係の剪定**：ネットワークの分断によるサービス継続を阻害しうる隠れた依存サービス（外部DNSやPKIによる証明書検証など）を用いないように、災害時はプロトコルを制限・排除する。

2. **絶対的な情報源への自律的昇格**：平常時においてエッジサーバはキャッシュサーバやプロキシとして振る舞うが、分断時においては事前にレプリケー

ションされたデータを基にオリジンサーバとして振る舞い、ユーザ端末に対して強制的にローカルサービスを提供する。

3. **ユーザ負担の最小化**：災害発生直後の混乱下において、一般利用者に複雑な設定変更（IP アドレスの手動設定やプロキシ設定など）を強いることは現実的ではない。そのため、端末側の設定変更を一切必要とせず、透過的にサービスへと誘導を行う。

5.2 技術的要素

4.1 節で述べた設計思想を実現するために、本システムでは以下の技術要素を組み合わせる。

- **DNS ハイジャックによる透過的誘導**：IP Anycast などの高コストなルーティング技術ではなく、エッジゲートウェイ（ルータ）における DNS スプーフィング（Destination NAT）を採用し、ユーザ端末からのクエリを強制的にローカルエッジへ転送する。これにより、クライアントの設定変更を不要にする。
- **自律監視スクリプト（Watchdog）**：ゲートウェイ上で動作する軽量スクリプトが、バックボーン回線の到達性を常時監視し、接続断絶を検知した瞬間にネットワーク設定（NAT ルール等）を動的に書き換える。
- **キャプティブポータル技術の応用**：OS やブラウザが持つ接続性確認機能に対する応答を模倣、あるいは DNS 誘導を利用することで、Web ブラウザ経由での情報提供インターフェースを即座に表示させる。
- **静的コンテンツの事前配置（Type A データ）**：避難所地図や防災マニュアルなど、更新頻度は低いが即時性の高いデータを、平常時にエッジサーバへ同期させておく。

5.3 動作シナリオ

本節では、提案手法における自律的なモード切替の動作フローを説明する。図 5.2 に平常時と災害発生時（分断時）のステートマシンを示す。

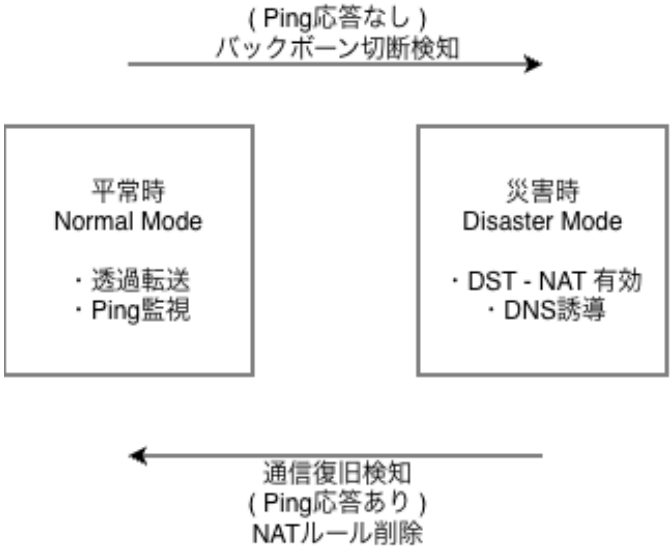


図 5.2: ステートマシン

5.3.1 平常時 (Normal Mode)

- ゲートウェイルータ上の監視プロセスが、外部ネットワーク（バックボーン）への死活監視を行う。監視間隔は 5 秒ごとの ICMP Echo Request（Ping）を用いる。
- 並行して、1 日 1 回の頻度で Type A データ（避難所マスタデータ等）の更新チェックを行い、エッジサーバ内のデータを最新化する。
- ハートビート（Ping）に対し、連続して所定回数（本研究では 3 回）以上の応答がない場合、ネットワーク分断と判断し、災害モードへ移行する。

5.3.2 災害発生時 (Disaster Mode)

1. **Disaster Mode の有効化**：監視プロセスが通信断絶を検知し、Disaster Mode を有効化する。
2. **経路変更**:ルータの NAT テーブルを書き換え、DNS クエリ（UDP/53, TCP/53）の宛先をエッジサーバへ強制転送（DNAT）する。

3. **誘導**：ユーザ端末が任意のドメイン（例: `google.com`）へアクセスしようとすると、エッジサーバが自身の IP アドレスを応答する。

4. **情報提供**：ユーザの HTTP リクエストに対し、エッジサーバ内の Web サーバが災害用ポータルや避難地図画像をレスポンスとして返す。

5. **復旧検知**：バックボーンへの Ping 応答が回復（連続 2 回以上成功）した場合、復旧と判断し、NAT ルールを削除して平常モードへ復帰する。

5.4 配置戦略の評価（PoC）

提案する自律分散型エッジシステムの有効性を検証するため、プロトタイプの実装および Proof of Concept (PoC) を実施した。

5.4.1 実験環境

VM で作成した実験トポロジを図 5.3 に、実験環境の仕様を表 5.1 に示す。ゲートウェイルータとして VyOS、エッジサーバおよびクライアントとして Ubuntu を用いた。

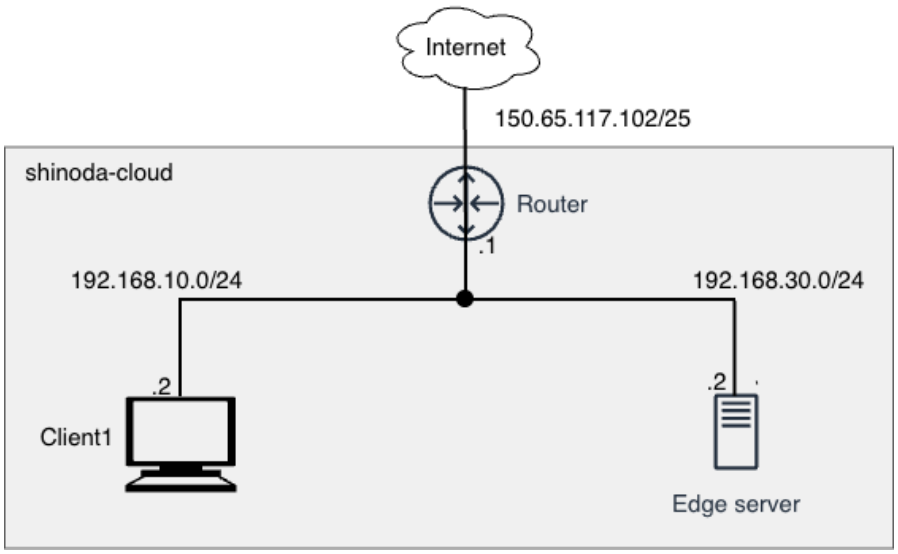


図 5.3: PoC 実験環境のトポロジ

表 5.1: 実験環境の仕様

Role	Specification
Gateway Router	VyOS 1.5-circinus (Watchdog Script 実装)
Edge Server	Ubuntu 22.04 LTS, Nginx, DNSmasq
Client	Ubuntu 22.04 LTS, curl
Backbone (Pseudo)	外部 IP (1.1.1.1) への Blackhole Routing による擬似切断

5.4.2 実験結果

本実験では、以下の 3 フェーズにおけるシステムの挙動と可用性を検証した。

1. 平常時の動作

図 5.4 に示す通り、クライアントからのリクエストは正規のインターネット（Google サーバ, 142.250.x.x）へとルーティングされ、通常の Web 閲覧が可能であることを確認した。

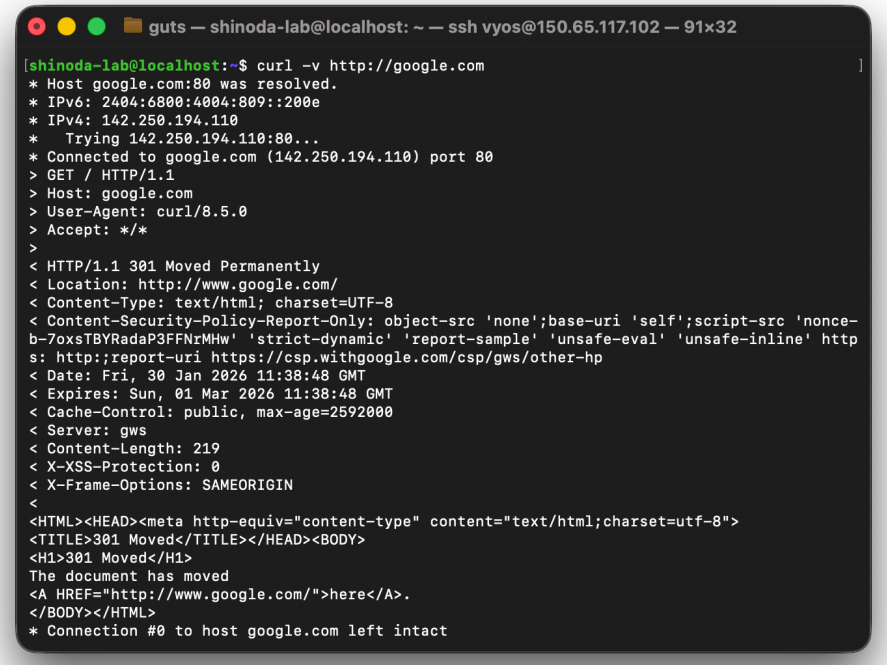


図 5.4: 平常時の curl コマンドの挙動

2. 災害発生時の自律切替

VyOS 上にて疑似的な回線切断（Blackhole route の適用）を行った直後、監視スクリプトが通信断を検知し、自律的に Disaster Mode へと移行した（図 5.5）。本システムにおける通信断絶の検知条件は 5 秒間隔の Ping 応答が連続 3 回欠損することであるため、検知に要する時間は約 15 秒となる。また、検知後の Disaster Mode 移行処理（NAT ルールの動的適用等）にかかる時間を 3 回計測した結果、8.909 秒、9.056 秒、8.738 秒となり、平均して約 8.90 秒であった。したがって、通信途絶の発生からエッジサーバへの経路誘導が完了するまでの総所要時間は約 23.9 秒となり、災害時の初期対応基盤として十分な即応性を有していることが確認できた。

その後、クライアントから再度同一の URL へアクセスを行った結果、IP アドレスがエッジサーバ（192.168.30.2）へと解決され、エッジサーバ内に保存された災害用ポータル画面および避難地図データ（map.png）が正常にダウンロードされたことを確認した（図 5.6 および図 5.7）。これにより、ユーザ端末の設定変更を一切行うことなく、ネットワーク分断下において迅速に Type A データの情報提供が可能であることが実証された。

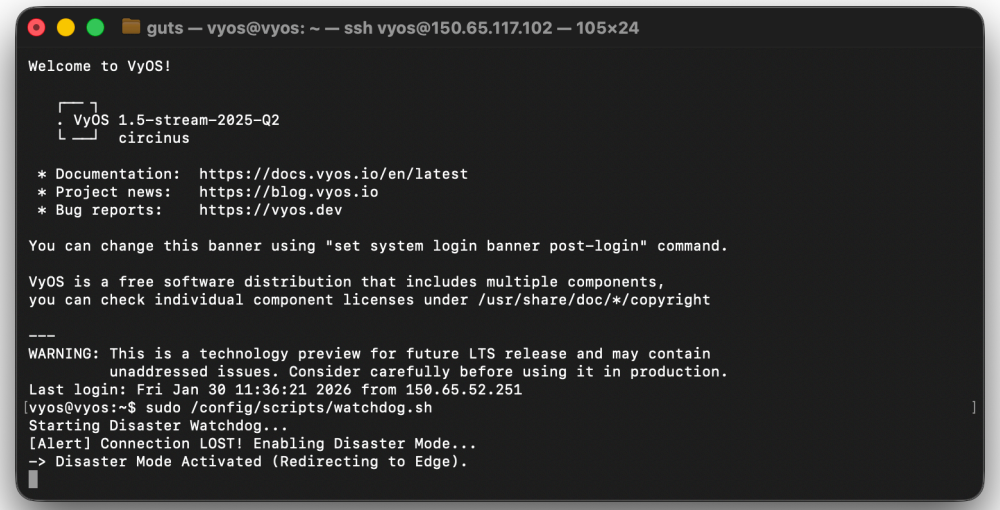


図 5.5: Disaster Mode に移行した様子

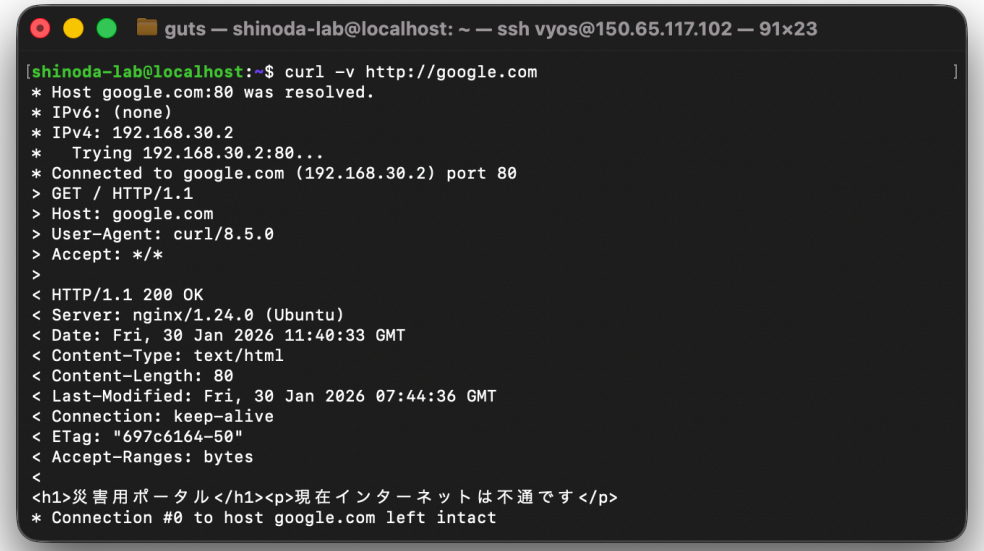


図 5.6: 災害時の curl 結果

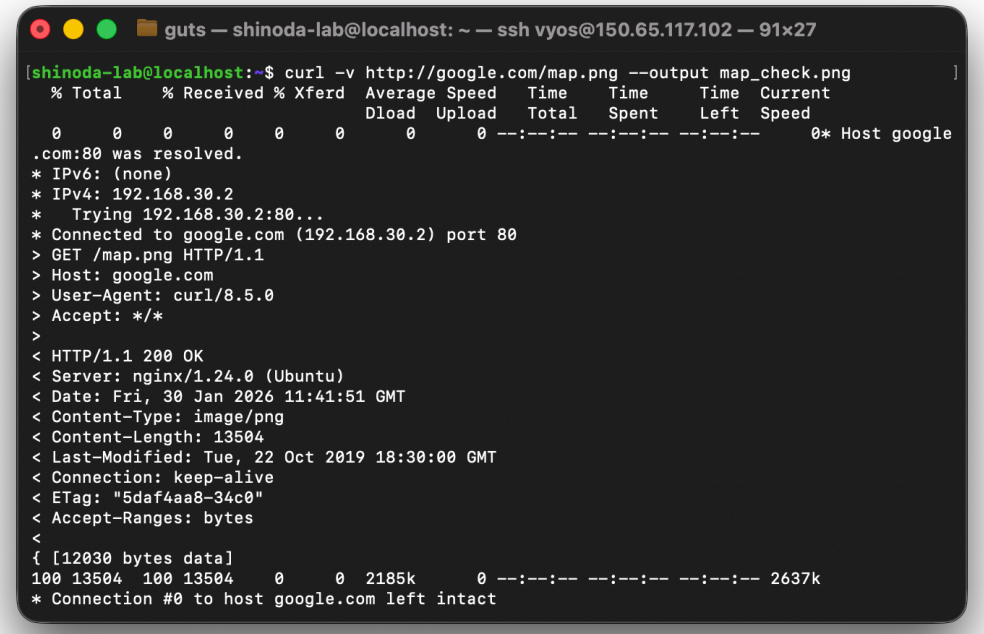


図 5.7: 災害時の地図データ取得

3. 復旧時の挙動とエラーの観測

回線復旧操作を行った際、監視スクリプトは通信の回復（Ping 応答の連続 2 回成功、すなわち約 10 秒での検知）を正常に認識し、復旧処理を開始した（図 5.8）。しかし、ルータの NAT ルールを平常時の状態へ動的に削除する過程において、ルータ OS（VyOS）のコンフィグ管理プロセス内部で C++の例外エラー（`std::out_of_range` 等）が発生し、コマンドがクラッシュする事象が観測された（図 5.9）。この内部エラーにより、ルータ内には災害モード用の NAT ルールが残留し続け、ネットワーク層が復旧した後もエッジサーバへ誘導される状態となった。結果として、本 PoC 環境においては正常な設定のロールバックが完了せず、非常時から平常時への完全なモード切り替え時間の有効な測定は困難であるという結果に至った。

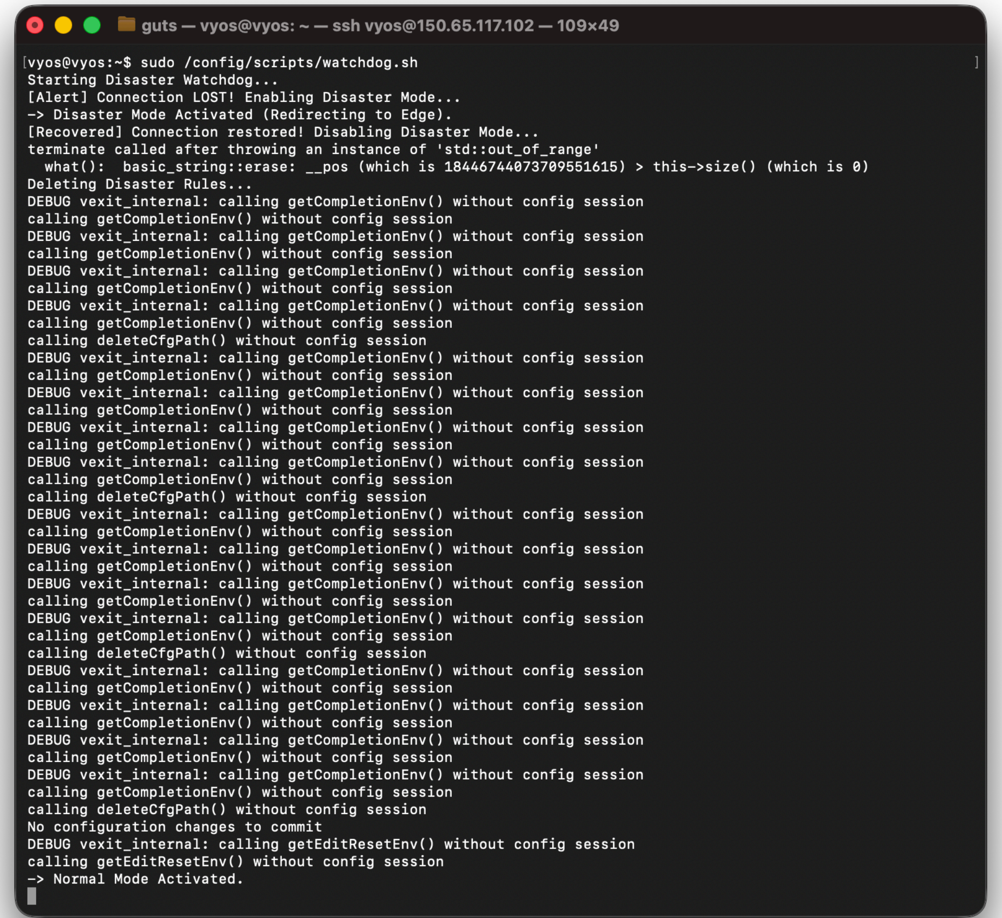


図 5.8: 監視スクリプトにおける復旧

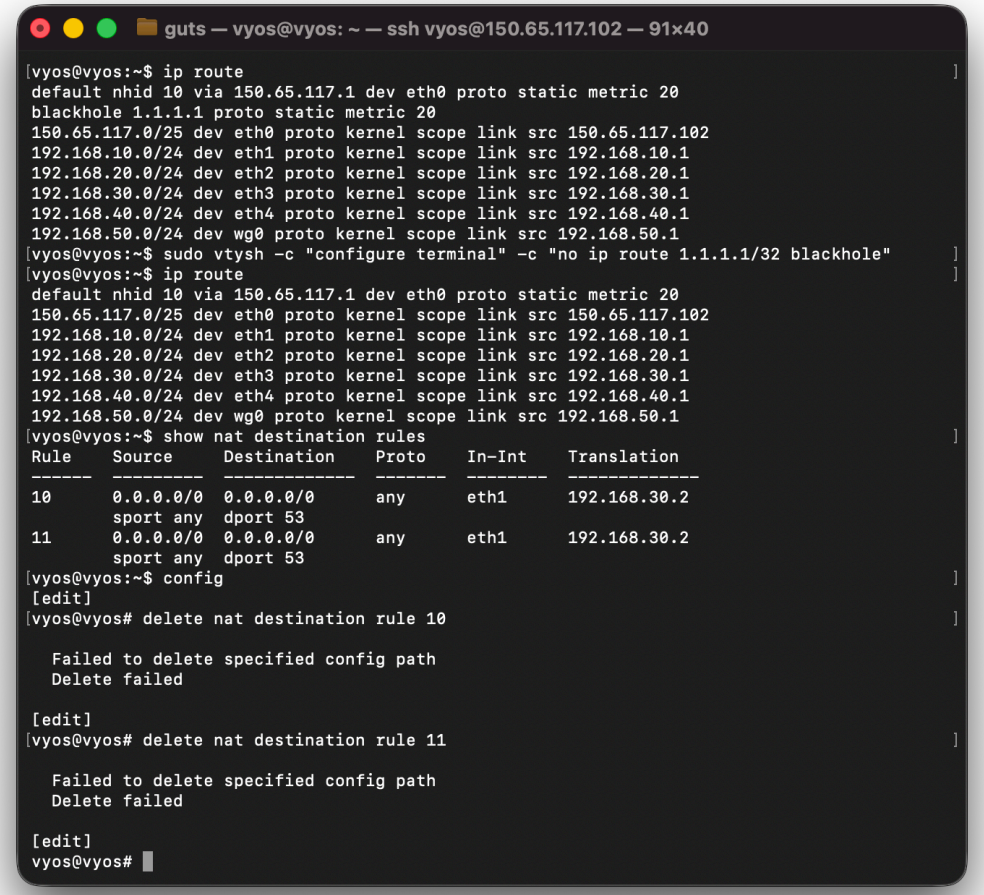


図 5.9: NAT ルール削除に失敗する現象

5.4.3 評価

本 PoC の結果より、第 3 章で示した隠れた依存サービスの課題のうち、特に中央集権的なトポロジに対する本アーキテクチャの有効性が実証された。

現在のインターネットは階層的なトポロジを前提としており、分断時には上位の権威 DNS サーバやオリジンサーバへの到達不能が致命的な障害となる。本提案手法では、事前にエッジサーバへコンテンツ（Type A データ）を配置した上で、エッジゲートウェイの Destination NAT による DNS ハイジャックを用いて、クライアントの要求を強制的にローカル内へ誘導した。これは、DNS や NAT といった機能を上位ネットワークへの依存を断ち切り、末端のローカルネットワーク内だけで通信を完結させるための手段として用いたものである。これにより、上位レイヤーのインフラが機能不全に陥った状況下においても、その影響を受けること

なく既存のアプリケーションを用いて情報提供を維持できることを確認した。
また、要求要件に対する定性・定量評価を以下にまとめる。

- **自律性・迅速性**：管理者の介入なく、分断検知（約 15 秒）から Disaster Mode への NAT ルール切り替え（平均約 8.90 秒）まで、通信途絶発生から合計約 23.9 秒で自律的な経路変更を完了した。既存のアプリケーションを非常時環境へシームレスにフェイルオーバーさせる基盤として十分な迅速性を備えている。
- **透過性**：クライアント端末のネットワーク設定（DNS, IP）を変更することなく、DNS ハイジャッキングによりスムーズにエッジサーバへ誘導できた。
- **可用性**：バックボーン回線が完全に遮断された状態においても、ローカルに配置した Type A データ（地図画像等）の配信に成功した。

以上の結果より、提案する配置戦略およびアーキテクチャは、災害時における孤立ネットワーク内の情報共有基盤として、上位トポロジへの依存という隠れた依存サービスを排除し、データを取得可能にするという目的を達成できていると結論付ける。

5.4.4 考察

本 PoC を通じて、提案システムの基本動作（自律検知、透過的誘導、コンテンツ配信）が正常に機能することを確認した。その一方で、実運用に向けて解決すべき技術的課題および運用上の課題も明らかになった。以下にその詳細を述べる。

HTTPS 通信と HSTS における制約

本実験では HTTP 通信を用いた検証を行ったが、現在のインターネットトラフィックの大半は HTTPS（SSL/TLS）によって暗号化されている。提案手法のような DNS ハイジャッキングを用いた場合、クライアント端末が要求する正規のドメイン（例: `google.com`）のデジタル証明書と、エッジサーバが提示する証明書との間に不整合が生じる。特に、HSTS (HTTP Strict Transport Security) が有効化されている主要なドメインにおいては、ブラウザがセキュリティ警告を表示するだけでなく、接続そのものを強制的に遮断する挙動をとる。これは実験環境特有の事象ではなく、中間者的な介入を行う本アーキテクチャと現代の Web セキュリティ仕様が競合することによって原理的かつ必然的に発生する制約である。この問題への実践的な対策として、OS が提供する Captive Network Assistant (CNA) の検知用 URL（HTTP で通信される特定の接続確認用エンドポイント）に対する誘導を優先的に処理する機構の実装が、今後の課題として挙げられる。

セキュリティリスクと可用性のトレードオフ

本システムが採用した通信を強制的に傍受し、意図しないサーバへ誘導するというアーキテクチャは、技術的には中間者攻撃（Man-in-the-Middle Attack）と同一の挙動である。平時においてこの手法を用いることは重大なセキュリティリスクとなるが、大規模災害時においては情報の機密性・完全性よりも情報の可用性が人命救助の観点で優先されるケースが存在する。本研究では、このトレードオフを許容する立場をとるが、悪意ある第三者による偽ポータルの設置等を防ぐため、災害時のみ信頼できる「Root of Trust」を確立する認証技術の検討が必要である。

実装環境に依存する課題と復旧時間の推算

復旧処理時に観測された NAT ルール削除の失敗は、本研究で提案する配置戦略や論理アーキテクチャの欠陥ではなく、PoC 実装に用いたソフトウェアルータ（VyOS 1.5 開発版）のコンフィグ操作 API における非同期処理の不具合、またはメモリ参照エラーといった実装依存の問題に起因するものである。ルータの構成管理機能が正常に動作する安定した環境を想定した場合、復旧処理にかかる理論上の時間は、通信回復の検知時間（Ping 5 秒間隔 $\times$ 2 回 $=$ 10 秒）に、災害モード移行時と同等の設定書き換え時間（約 8.90 秒）を加えた、約 18.9 秒程度になると推算される。実運用環境に向けては、長期サポート（LTS）版の OS 選定や、CLI に依存しない API 経由での確実なステート管理（トランザクション処理）が必要であると考えられる。

誤検知の防止と安定性

監視スクリプトの実装において、単発の Ping 応答失敗ですぐに災害モードへ移行するのではなく、連続 3 回の失敗をトリガーとする閾値を設けた。無線区間のゆらぎや一時的な輻輳によるパケットロスは日常的に発生しうるため、過敏な反応によるモードの頻繁な切り替えを防ぐ上で、このタイムウィンドウの設定はシステムの安定稼働に不可欠な要素であると確認できた。

アプリケーション層におけるキャッシュクリアランスの課題

本アーキテクチャにおいてシステム全体の完全な平常化を考慮する場合、ネットワーク層（L3）のルーティング復旧時間に加えて、アプリケーション層（L7）における DNS キャッシュの生存期間（TTL: Time To Live）の影響を考慮する必要があることが明らかになった。仮にルータ側で NAT による強制転送が約 18.9 秒で正常に解除されたとしても、クライアント端末や内部 DNS サーバには、エッジサーバの IP アドレスが一定時間キャッシュとして保持され続ける。そのため、実

質的にクライアントが外部の正規 Web サイトへアクセス可能になるまでには、さらに数秒から数十秒の遅延が見込まれる。したがって、実運用に向けた今後の課題として、災害モード時にエッジサーバが応答する DNS レコードの TTL を意図的に極小値（0〜1 秒）に設定するなど、アプリケーション層の仕様と連携した復旧設計の最適化が求められる。

第6章　配置戦略に関する考察

本章では、要求要件を満たしうる技術とそれを活用した配置戦略について考察する。

6.1　Mobile Ad-Hoc Network

先行研究で紹介したように、Mobile Ad-Hoc Network(MANET) とは、基地局や固定網に依存せず、移動端末を構成要素とする自律分散形のネットワークと定義されている [3]。本考察は参考文献 [3] で述べられている技術に基づいて行う。

アドホックネットワークは、無線通信によるマルチホップでのデータ転送が特徴である。アドホックネットワークを構成する要素について参考文献 [3] では次のように挙げられている。

1. 構成形態

   アドホックネットワークを構成する形態として、以下の 4 つがあると述べられている。

   (a) 通常のモバイルノードのみの形態

   (b) ノードの一部が固定的に配置される形態

   (c) 情報の受付・中継・蓄積・分配を専門的に行うノード (コミュニケーション・ポート) を使用する形態

   (d) 従来の移動通信ネットワークと併用される形態

   MANET においては (a) の形態が検討対象となっている。

2. マルチホップ通信の実現手段

   すべての構成形態に共通するのは、前述した通り、無線通信によるマルチホップを行うことである。これを実現するためには以下の 2 方式があると定義されている。

   - パケット無線方式

     MANET では、パケット無線方式を前提としている。アドホック・モードでノード間を接続し、ノードがルータの役割を果たすことでマル

チホップを実現する。通信範囲内に複数のノードがある場合、1 回のパケット送信でそれら全てのノードにブロードキャスト的に情報転送を行うことが可能である。また、宛先を指定することでユニキャストでの通信も可能である。

- 無線回線接続方式
  隣接ノード間で回線確立、情報伝送、回線切断を行う。これによって情報が 1 ホップ移動する。この方式では、1 回のパケット送信で一つのノードに情報が転送されている。一つのノードに無線回線接続方式の通信装置を複数装備させれば、複数のノードとの通信が可能になる。

これらを踏まえて、災害時の配置戦略やデータの可用性について考察する。

MANET での情報伝達は、避難行動している一般市民間での通信には適している。しかし、課題として、避難している一般市民が複数人近くにいなくてはならない点が挙げられる。都市部なら避難している一般市民の数も多いため、通信可能なケースが多いと予想されるが、地方地域では通信することが不可能になる可能性が高い。

また、初動段階 (災害発生から 72 時間以内) では、定義した通りハザードマップや避難情報といった Type A データ (自治体発一般市民宛のデータ) が重要になる。アドホックネットワークでは、バケツリレー方式で情報を転送しているため、情報発信源が不明になる。そのため、デマの伝播が起こりやすくデータの信頼性が低下する。もし Type A データを転送できたとしても、最初に Type A データを取得する一般市民がいなければ、可用性が低下する。

さらに、通常のモバイルノードのみでアドホックネットワークを構成した場合、スマートフォンなどの携帯端末において、バッテリーに依存している [3]。バッテリーの消費を抑制する研究もあるが、常時アドホックによる通信を可能にするならば、バッテリーによる制限が大きい。バッテリーを完全に消費した場合、他のノードへの情報転送だけでなく、その携帯端末の所持者自身はデータを閲覧することが不可能になる。

したがって、情報通信頻度が災害発生地域に依存している点、データの信頼性が低下する点、バッテリーによるデバイスの制約がある点により、解決手段とするのは困難だと判断した。

しかし、これはスマートフォンなどのモバイルノードのみでアドホックネットワークを構成した場合の課題である。例えば、構成形態で紹介した (b) と (c) を組み合わせると災害時の情報伝達を行える可能性が高い。自治体の災害対策本部や避難所に固定設置されているコミュニケーションノードを分散配置し、避難経路がすべてコミュニケーションノードの通信範囲内に含まれるようにすれば、データの信頼性や可用性が向上する可能性がある。しかし、避難行動をとっている一般市民のモバイルノードのバッテリーに依存している点が依然として課題になる。

6.2 DTN (Delay Tolerant Network)

先行研究で挙げたように、DTN(Delay Tolerant Network) は劣悪なネットワーク環境下での活用するネットワーク技術である [5]。DTN では”store-carry-forward” と呼ばれる方式を用いていることが大きな特徴である。この方式は、近くに利用可能なノードがない場合に送信データを保存し、宛先と近づくことでデータを複製し送信する。

DTN では、情報通信の確実性に焦点を当てているため、情報通信の成功率が高いと考えられる。しかし、先行研究内の課題として挙げられていた通り、地方地域では避難する一般市民が少ない可能性が高く、情報通信の頻度が低下する恐れがある [5]。また、”store-carry-forward”によって、素早い情報伝達が不可能な点も課題として挙げられる。

初動段階における Type A データの伝達を焦点に当てている本研究では、この課題による可用性の低下が大きなボトルネックとなったため、解決手段とするのは困難だと判断した。

しかし、先行研究 [5] のような課題を解決する手法が提案されている。日本国内の携帯キャリア 3 社 (KDDI, SoftBank, NTT docomo) が保有する車載型基地局を DTN のノードとして稼働させることで、データの伝達頻度の向上などが見込める。実現するためには災害発生地域までの陸路が利用不可能な場合の対応や分散配置方法など検討事項が多い。

6.3 衛星通信

先行取組として携帯キャリア 3 社が行なっているものとして、Starlink などの衛星通信技術の活用がある。バックボーンネットワークが途絶しても衛星通信経路を用いることで通信の維持を図るものである。

本研究では、ハードウェアなどの故障といった物理な課題について焦点を当てていない。衛星通信は基地局の代替手法といった物理的な部分を担っているため、本研究の焦点とは異なる。そのため、配置戦略を満足する手法の一つとしての議論を行わなかった。また、一般家庭への普及が進んでいない点や衛星電波を送受信するアンテナの設置などの課題があるため、現段階では災害時におけるネットワークの課題を抜本的に解決できないと考えた。

第7章　まとめ

7.1　研究成果のまとめ

本研究では次のことを行った。

1. 隠れた依存サービスの課題分析

   隠れた依存サービスの課題を特定するために、過去の通信障害の事例を参考にした。また、プロトコルスタックの分析を行い、多様な種類の隠れた依存サービスがあることを明確にした。以下に、3.3 節で定義した隠れた依存サービスの課題について示す。

   (a) プロトコルスタックによる通信確立の複雑性

   DNS を中心とした通信確立のためのプロトコルが機能不全に陥った場合、データなどを含むコンテンツのあるサーバが存在していたとしてもコンテンツ取得が不可能になる。

   (b) 通信帯域の圧迫

   通信経路が確立したとしても、輻輳による通信帯域の圧迫によってデータ取得までの所要時間が増加、最悪の場合にデータ取得が不可能になる。

   (c) 中央集権的なトポロジ

   現在のインターネットにおけるサーバの配置は階層的である。上位のサーバへと到達できなければ新しいデータを取得することが不可能である。

   また、認証を要求されるサービスに対して、認証が行えなければサービスを享受することが不可能である。

2. 配置戦略

   配置戦略では、はじめにデータとサービスの定義をし、データ・サービスの分類を行った。その後、平常時と緊急時のフェーズ移行という時間的な観点から、各フェーズにおいてどのようなデータ・サービスが必要になるか検討した。特に初動段階においては、EEI（災害対応基本共有情報）に準拠した動的情報を Type A（自治体発一般市民宛データ）として再定義し、優先度を明確にした。最後に要求要件としてまとめ、それらの大項目を全て満足しなければ災害対策ネットワークとしては不十分であると定義した。

3. エッジコンピューティングを用いた配置戦略と PoC による実証
   配置戦略や要求要件をもとに、エッジゲートウェイの自律的なモード切替と DNS ハイジャックを活用したアーキテクチャを提案・実装した。PoC を通じた評価の結果、通信途絶の検知から合計約 24 秒という迅速さでローカルへの経路切り替えを自律的に完了し、上位トポロジへの依存（課題 3）を排除した情報提供が可能であることを実証した。一方で、実運用に向けては、Web セキュリティ仕様（HSTS 等）との競合や、ルータ OS の実装依存に起因する課題があることも明らかになった。

7.2 今後の展望

本研究で得られた知見に基づき、実用化および次世代アーキテクチャの確立に向けて解決すべきいくつかの課題が明らかになった。

第一に、HTTPS および HSTS 通信におけるセキュリティ上の制約である。本提案手法におけるリダイレクト機構は、中間者的な介入を行う性質上、HSTS が有効化されているブラウザでは証明書の不一致により接続が原理的に遮断されてしまう。この問題に対処するため、OS のキャプティブポータル機能（CNA）を活用して正規の手順で災害情報ページを表示させる手法や、ローカル環境における信頼モデルの確立に向けた検討が必要である。

第二に、インフラストラクチャの確実なステート管理とアプリケーション層の連携である。PoC における復旧検証から、ルータ OS の実装依存による設定ロールバックの失敗や、DNS キャッシュの生存期間（TTL）に起因するアプリケーション層の復旧遅延が確認された。実運用環境においては、API を経由したトランザクション制御の導入や、DNS の TTL を極小値に設定するなどのシステム全体を見据えた最適化が求められる。

第三に、ハードウェアへの実装と実環境における性能評価である。現在の評価は仮想環境上で行われたものであるが、実際の災害時における停電環境下での動作を担保するためには、Raspberry Pi などの省電力ハードウェアへのシステム実装が不可欠である。さらに、避難所などで多数の避難者が一斉に接続する状況を想定し、電波干渉や帯域幅の飽和が通信品質に与える影響を考慮した、実際の無線 LAN 環境における検証が必要である。

第四に、長期的戦略としての ICN (Information-Centric Networking) の導入である。本研究の PoC では中央集権的なトポロジ（課題 3）への依存を排除することに成功したが、既存の IP ネットワークを利用する以上、DNS や NAT といったプロトコルスタックへの依存（課題 1）はローカル環境内に形を変えて残存している。朝枝ら [21] が述べるように、ICN は従来の IP アドレスではなく「コンテンツ名」を識別子として通信を行うアーキテクチャであり、ネットワーク内キャッシュを活用することで、DNS や BGP といった隠れた依存サービスの影響を原理的に受けない利点がある。現在のところ、グローバル・ユニークな命名規則の定義や

セキュリティ要件など実用化に向けた課題が多く、本研究での採用は見送ったが、10 年単位の長期的な視点で見れば通信確立の複雑性という根本課題を完全に解決する世界標準技術となる可能性がある。今後の研究動向を注視しつつ、エッジコンピューティングと ICN を統合した次世代の災害対策アーキテクチャの検討が期待される。

謝辞

本研究を行うにあたり, 多くの方から多大なご助言やご助力いただきました。その方々のご協力がなければ、本研究は成り立ちませんでした。ここに深く感謝し、心から厚く御礼申し上げます。

本研究を進めるにあたり、主指導教員である宇多 仁 准教授には適切な助言とご指導を賜りました。また、研究に対する指導だけでなく、物事の捉え方や、人に物事を説明する方法など多くのことを学ばせていただきました。WIDE Project での研究発表、学外の研究者との関わりなど、多くの機会を提供してくださいました。深く感謝いたします。

副指導教員である丹 康雄 教授をはじめ、リム 勇仁 教授、島田 淳一 教授には、中間審査会や修士論文審査会にて客観的かつ的確なご意見を賜りました。厚く謝意を表します。

篠田 陽一 特任教授には、研究に対する多角的視点やプレゼンテーションに対する姿勢など、熱心なご指導を賜りました。心より御礼申し上げます。

また、Project ARIA の皆様には、災害対策の考え方などについて有益なご助言を賜りました。ここに深く感謝いたします。

本研究室修了生の中川 颯馬 氏、本研究室の西村 優典 氏、高田 敦生 氏、金田 昂大 氏、Lu Xingchen 氏、斎藤 翼 氏、岡坂 直徒 氏には研究に関する様々な議論や研究生活を送る上での多大なご助力をいただきました。深く感謝いたします。

最後に、学生生活を支えていただいた家族へ心から感謝いたします。

修士論文の提出に至るまで、多くの方々より多大なるご支援をいただきました。心より感謝申し上げます。

参考文献

[1] 総務省．災害に強い情報通信ネットワーク 導入ガイドブック 2024，耐災害 ICT 研究協議会 (2024)

[2] A. Mauthe, D. Hutchison, E. K. Çetinkaya, *et al.* Disaster-resilient communication networks: Principles and best practices, 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), pp.1–8 (2016), doi: 10.1109/RNDM.2016.7608262

[3] 間瀬憲一. モバイル・アドホックネットワーク, 日本オペレーションズ・リサーチ学会 第 47 回シンポジウム資料, pp.13–26 (2002). Available: https://orsj.org/wp-content/or-archives50/pdf/sym/S47_013.pdf

[4] M. Tarasov, J. Seitz, and O. Artemenko. A network partitioning recovery process in Mobile Ad-Hoc Networks, 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp.32–36 (2011), doi: 10.1109/WiMOB.2011.6085393

[5] N. Uchida, N. Kawamura, K. Takahata, *et al.* Proposal of Data Triage Methods for Disaster Information Network System Based on Delay Tolerant Networking, 2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications, pp.15–21 (2013), doi: 10.1109/BWCCA.2013.12

[6] KDDI 株式会社. KDDI の備え/災害発生時の対応, https://www.kddi.com/anti-disaster/safety-measures/, (参照 2026/01/31)

[7] ソフトバンク株式会社. 復旧への取り組み, https://www.softbank.jp/corp/sustainability/esg/social/disaster/restoration/, (参照 2026/01/31)

[8] 株式会社 NTT ドコモ. 災害対策, https://www.docomo.ne.jp/corporate/csr/disaster/, (参照 2026/01/31)

[9] F. Liu, G. Tang, Y. Li, *et al.* A Survey on Edge Computing Systems and Tools, Proceedings of the IEEE, vol.107, no.8, pp.1537–1562 (2019), doi: 10.1109/JPROC.2019.2920341

[10] N. Abbas, Y. Zhang, A. Taherkordi, *et al.* Mobile Edge Computing: A Survey, IEEE Internet of Things Journal, vol.5, no.1, pp.450–465 (2018), doi: 10.1109/JIOT.2017.2750180

[11] T. Taleb, K. Samdanis, B. Mada, *et al.* On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration, IEEE Communications Surveys & Tutorials, vol.19, no.3, pp.1657–1681 (2017), doi: 10.1109/COMST.2017.2705720

[12] Cloudflare. CDN とは, https://www.cloudflare.com/ja-jp/learning/cdn/what-is-a-cdn/, (参照 2026/02/01)

[13] Cloudflare. Cloudflare グローバルネットワーク, Cloudflare, https://www.cloudflare.com/ja-jp/network/, (参照 2026/02/02)

[14] Amazon Web Services. AWS Wavelength, Amazon Web Services, https://aws.amazon.com/jp/wavelength/, (参照 2026/02/02)

[15] Meta. More details about the October 4 outage, https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/, (参照 2026/02/01)

[16] KDDI 株式会社. 2022 年 7 月 2 日に発生した通信障害について, https://news.kddi.com/kddi/corporate/newsrelease/2022/07/29/6183.html, (参照 2026/02/01)

[17] Fastly. Summary of June 8 outage, https://www.fastly.com/blog/summary-of-june-8-outage, (参照 2026/02/01)

[18] Fastly. Fastly ' s powerful edge cloud platform delivered across your private network, https://www.fastly.com/jp/services/managed-cdn, (参照 2026/02/01)

[19] Google Cloud. Google Cloud Infrastructure Components Incident, https://status.cloud.google.com/incident/zall/20013, (参照 2026/02/01)

[20] Mozilla. ページの生成: ブラウザーの動作の仕組み, MDN Web Docs, https://developer.mozilla.org/ja/docs/Web/Performance/Guides/How_browsers_work, (参照 2026/02/02)

[21] 朝枝 仁, 松園 和久, 大岡 睦. 情報指向型ネットワーク技術, 情報通信研究機構研究報告, vol.64, no.2, pp.93–102 (2018), https://www.nict.go.jp/publication/shuppan/kihou-journal/houkoku-vol64-2/K2018N-06-02.pdf

付録

ここでは、修士研究に用いたVMの設定や作成したソースコードを示す。なお、Client1、DNS server、Web serverの設定については、標準的なIPアドレスの割り当ておよび静的ルーティング設定のみであるため、本付録では割愛する。

Router（VyOS）の全体設定

```
firewall {
    ipv4 {
        forward {
            filter {
                default-action drop
                rule 10 {
                    action accept
                    state established
                    state related
                }
                rule 20 {
                    action accept
                    description "Allow Client to Internet"
                    source {
                        address 192.168.10.0/24
                    }
                }
                rule 21 {
                    action accept
                    description "Allow DNSServer to Internet"
                    source {
                        address 192.168.20.0/24
                    }
                }
                rule 22 {
                    action accept
                    description "Allow WebServer to Internet"
                    source {
                        address 192.168.30.0/24
                    }
                }
                rule 30 {
                    action accept
                    source {
                        address 192.168.40.0/24
```

```
            }
        }
        rule 40 {
            action accept
            description "Allow Web server Access"
            destination {
                address 192.168.30.2
                port 80,443
            }
            protocol tcp
        }
    }
}
input {
    filter {
        default-action drop
        rule 10 {
            action accept
            state established
        }
        rule 11 {
            action accept
            icmp {
            }
        }
        rule 15 {
            action accept
            destination {
                address 192.168.0.0/16
            }
            source {
                address 192.168.0.0/16
            }
        }
    }
}
output {
    filter {
        default-action drop
        rule 10 {
            action accept
            icmp {
            }
        }
        rule 15 {
            action accept
            state established
            state related
        }
        rule 20 {
            action accept
            destination {
                address 192.168.0.0/16
```

```
                }
            }
            rule 30 {
                action accept
                description "Allow Router to DNS and NTP"
                destination {
                    port 53,123
                }
                protocol udp
            }
        }
    }
}
interfaces {
    ethernet eth0 {
        address dhcp
        description "To external network"
        hw-id 52:54:00:26:6f:45
        offload {
            gro
            gso
            sg
            tso
        }
    }
    ethernet eth1 {
        address 192.168.10.1/24
        description "To VM client"
        hw-id 52:54:00:2a:09:1f
        offload {
            gro
            gso
            sg
            tso
        }
    }
    ethernet eth2 {
        address 192.168.20.1/24
        description "To DNS server"
        hw-id 52:54:00:50:ac:6a
        offload {
            gro
            gso
            sg
            tso
        }
    }
    ethernet eth3 {
        address 192.168.30.1/24
        description "To Web server"
        hw-id 52:54:00:a0:a0:c4
        offload {
```

```
            gro
            gso
            sg
            tso
        }
    }
    ethernet eth4 {
        address 192.168.40.1/24
        description "Mobile clients network"
        hw-id 52:54:00:84:b5:5a
    }
    loopback lo {
    }
}
nat {
    destination {
    }
    source {
        rule 100 {
            outbound-interface {
                name eth0
            }
            source {
                address 192.168.0.0/16
            }
            translation {
                address masquerade
            }
        }
    }
}
policy {
}
protocols {
    static {
        route 0.0.0.0/0 {
            next-hop 150.65.117.1 {
            }
        }
    }
}
service {
    dhcp-server {
        shared-network-name MOBILE {
            authoritative
            subnet 192.168.40.0/24 {
                option {
                    default-router 192.168.40.1
                    domain-name local.net
                    name-server 192.168.20.2
                }
                range 0 {
                    start 192.168.40.10
```

```
                    stop 192.168.40.100
                }
                subnet-id 1
            }
        }
    }
    ntp {
        allow-client {
            address 127.0.0.0/8
            address 169.254.0.0/16
            address 10.0.0.0/8
            address 172.16.0.0/12
            address 192.168.0.0/16
            address ::1/128
            address fe80::/10
            address fc00::/7
        }
        server time1.vyos.net {
        }
        server time2.vyos.net {
        }
        server time3.vyos.net {
        }
    }
    ssh {
    }
}
system {
    config-management {
        commit-revisions 100
    }
    console {
        device ttyS0 {
            speed 115200
        }
    }
    host-name vyos
    login {
        user vyos {
            authentication {
                encrypted-password ****************
                plaintext-password ****************
            }
        }
    }
    option {
        reboot-on-upgrade-failure 5
    }
    syslog {
        local {
            facility all {
                level info
            }
```

```
            facility local7 {
                level debug
            }
        }
    }
}
```

確認実験でのローカル Web ページのソースコード

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Local Web Server - Status</title>
    <style>
        body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 0;
        padding: 0; background-color: #f0f2f5; color: #333; }
        header { background-color: #0056b3; color: white; padding: 1rem 2rem; display: flex;
        justify-content: space-between; align-items: center; }
        h1 { margin: 0; font-size: 1.5rem; }
        .badge { background-color: #28a745; color: white; padding: 5px 10px;
        border-radius: 15px; font-size: 0.8rem; vertical-align: middle; }
        .container { padding: 40px; max-width: 800px; margin: 0 auto; }
        .card { background-color: white; border-radius: 8px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1); padding: 20px; margin-bottom: 20px; }
        .card h2 { margin-top: 0; border-bottom: 2px solid #f0f2f5; padding-bottom: 10px;
        font-size: 1.2rem; color: #0056b3; }
        .status-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))
        gap: 20px; }
        .status-item { text-align: center; padding: 10px;
        background: #f8f9fa; border-radius: 5px; }
        .status-value { font-size: 1.5rem; font-weight: bold; color: #333; }
        .status-label { font-size: 0.9rem; color: #666; }
        .footer { text-align: center; margin-top: 40px; color: #888; font-size: 0.9rem; }
    </style>
</head>
<body>
    <header>
        <h1>Local Web Server <span class="badge">Online</span></h1>
        <span>Node ID: WEB-01</span>
    </header>

    <div class="container">
        <div class="card">
            <h2>System Status</h2>
            <div class="status-grid">
                <div class="status-item">
                    <div class="status-value">192.168.30.2</div>
                    <div class="status-label">Local IP Address</div>
                </div>
                <div class="status-item">
```

```
                    <div class="status-value">Normal</div>
                    <div class="status-label">Operating Mode</div>
                </div>
                <div class="status-item">
                    <div class="status-value">Nginx</div>
      <div class="status-label">Open Source</div>
  </div>
            </div>
        </div>

        <div class="card">
            <h2>Local Services</h2>
            <p>このサーバはローカル Web サーバとして稼働しています。</p>
            <ul>
                <li>Access Only</li>
  <li>Read Only</li>
            </ul>
        </div>
    </div>

    <div class="footer">
        &copy; Made by Nakamura Kazuki
    </div>
</body>
</html>
```

watchdog.sh のソースコード

```
#!/bin/vbash

# VyOS のコマンドをスクリプト内で使うための環境読み込み
source /opt/vyatta/etc/functions/script-template

# ========================================
# 監視設定パラメータ
# ========================================
TARGET="1.1.1.1"          # 監視先のインターネット IP（バックボーン）
EDGE_IP="192.168.30.2"    # 誘導先のエッジサーバ IP
INTERVAL=5                # Ping を打つ間隔（秒）
FAIL_THRESHOLD=3          # 何回連続で失敗したら災害モードにするか
RECOVER_THRESHOLD=2       # 何回連続で成功したら平常モードに戻すか

# ========================================
# 状態管理変数（初期化）
# ========================================
fail_count=0
recover_count=0
current_mode="NORMAL"

echo "Starting Disaster Watchdog..."
echo "Target: $TARGET, Interval: ${INTERVAL}s"
```

```
# 無限ループで常時監視
while true; do
    # Ping を 1 回だけ送り、タイムアウトを 1 秒に設定
    ping -c 1 -W 1 $TARGET > /dev/null 2>&1
    PING_RESULT=$?

    if [ $PING_RESULT -ne 0 ]; then
        # ----------------------------------
        # Ping 失敗時 (パケットロス)
        # ----------------------------------
        recover_count=0
        ((fail_count++))

        # 閾値に達し、かつ現在が NORMAL モードなら切り替え
        if [ $fail_count -ge $FAIL_THRESHOLD ] && [ "$current_mode" = "NORMAL" ]; then
            echo "[Alert] Connection LOST! Enabling Disaster Mode..."

            # --- 計測開始 ---
            START_TIME=$(date +%s.%N)

            # VyOS 設定モード移行
            configure

            # 1. DNS リクエスト (UDP/TCP 53) をエッジサーバへ強制転送 (DNAT)
            set nat destination rule 10 description 'DNS Hijack UDP'
            set nat destination rule 10 destination port '53'
            set nat destination rule 10 inbound-interface name 'eth1'
            set nat destination rule 10 protocol 'udp'
            set nat destination rule 10 translation address $EDGE_IP

            set nat destination rule 11 description 'DNS Hijack TCP'
            set nat destination rule 11 destination port '53'
            set nat destination rule 11 inbound-interface name 'eth1'
            set nat destination rule 11 protocol 'tcp'
            set nat destination rule 11 translation address $EDGE_IP

            # 2. エッジからの戻りパケットを正しくクライアントに返すためのマスカレー
ド (SNAT)
            set nat source rule 105 description 'Masquerade for Edge UDP'
            set nat source rule 105 destination address $EDGE_IP
            set nat source rule 105 destination port '53'
            set nat source rule 105 outbound-interface name 'eth3'
            set nat source rule 105 protocol 'udp'
            set nat source rule 105 translation address 'masquerade'

            set nat source rule 106 description 'Masquerade for Edge TCP'
            set nat source rule 106 destination address $EDGE_IP
            set nat source rule 106 destination port '53'
            set nat source rule 106 outbound-interface name 'eth3'
            set nat source rule 106 protocol 'tcp'
            set nat source rule 106 translation address 'masquerade'
```

```
        # 設定の適用と保存
        commit
        save
        exit

        # --- 計測終了 ---
        END_TIME=$(date +%s.%N)
        ELAPSED_TIME=$(awk "BEGIN {printf \"%.3f\", $END_TIME - $START_TIME}")

        current_mode="DISASTER"
        echo "-> Disaster Mode Activated (Redirecting to Edge)."
        echo "[Metrics] Disaster Mode 移行処理時間: ${ELAPSED_TIME} 秒"

        # ログファイルに記録
        echo "$(date '+%Y-%m-%d %H:%M:%S') -Disaster Mode Activated- Time: ${ELAPSED_TIME}s"
        >> /var/log/watchdog_metrics.log
    fi

else
    # ----------------------------------
    # Ping 成功時（通信確保）
    # ----------------------------------
    fail_count=0
    ((recover_count++))

    # 閾値に達し、かつ現在が DISASTER モードなら復旧
    if [ $recover_count -ge $RECOVER_THRESHOLD ] && [ "$current_mode" = "DISASTER" ]; then
        echo "[Recovered] Connection restored! Disabling Disaster Mode..."

        # --- 計測開始 ---
        START_TIME=$(date +%s.%N)

        # VyOS 設定モード移行
        configure

        # 平常時のコンフィグをロード
        echo "Loading Normal Configuration Snapshot..."
        load /config/normal.boot

        # 設定の適用と保存
        commit
        save
        exit

        # --- 計測終了 ---
        END_TIME=$(date +%s.%N)
        ELAPSED_TIME=$(awk "BEGIN {printf \"%.3f\", $END_TIME - $START_TIME}")

        current_mode="NORMAL"
        echo "-> Normal Mode Activated."
        echo "[Metrics] Normal Mode 復旧処理時間: ${ELAPSED_TIME} 秒"

        # ログファイルに記録
```

```
            echo "$(date '+%Y-%m-%d %H:%M:%S') - Normal Mode Activated - Time: ${ELAPSED_TIME}s"
            >> /var/log/watchdog_metrics.log
        fi
    fi

    # 次の Ping までの待機時間
    sleep $INTERVAL
done
```