

ネットワークの仕組みを実習形式で学べる決定版

Linux

ネットワーク標準教科書

Ver. **1.0.0**



open your NEXT future

LPI-JAPAN

Linux Professional Certification

<https://lpi.or.jp>

Linux ネットワーク標準教科書

2026 年 10 月 1 日 Ver.1.0.0

LPI-Japan 発行

まえがき

このたび、特定非営利活動法人エルピーアイジャパンは、Linux 技術者教育に利用していただくことを目的とした教材「Linux ネットワーク標準教科書」（以下、本教科書）を開発し、インターネット上にて公開し、提供することとなりました。

本教科書は、多くの教育機関から Linux を「基礎」から学習するための教材や学習環境の整備に対するご要望があり、開発したものです。

公開にあたっては、本教科書に添付されたライセンス（クリエイティブ・コモンズ・ライセンス）の下に公開されています。

本教科書は、最新の技術動向に対応するため、随時アップデートを行っていきます。

本教科書の最新情報は以下の Web ページをご参照ください。

<https://linuc.org/textbooks/network/>

本教科書の目的

本教科書の目的は、「Linux サーバー構築標準教科書」をはじめとした「標準教科書シリーズ」で解説しきれなかったネットワークの知識を、実習を通して学習することにあります。

実際に Linux を操作して、ネットワークの仕組みを理解することができます。

全体の流れ

本教科書では、以下の通りに実習を進めます。

1 章 ネットワーク学習環境の構築

実習環境の構築と、VirtualBox のネットワークについて解説します。また、パケットキャプチャを行うために Wireshark をインストールします。

2 章 IP アドレス

インターネットの通信の基盤である IP の仕組みと、IP アドレスについて解説します。

3 章 TCP/UDP/ICMP

IP 通信の上で使用される、TCP や UDP、ICMP について解説します。

4 章 パケットフィルタリング

IP アドレスやポート番号などを元にパケット通信の許可や拒否を行うパケットフィルタリングについて解説します。

5 章 ルーティング/IP マスカレード/プロキシ

インターネットの基本的な通信の仕組みであるネットワーク間を接続する仕組みであるルーティングについて解説します。また、Linux カーネルのアドレス変換の仕組みである IP マスカレードや特定のプロトコルをクライアントの代理として接続するプロキシについて解説します。

6 章 ネットワークインターフェースの管理

ネットワークの通信を担うハードウェアであるネットワークインターフェースについて解説します。複数のネットワークインターフェースを束ねて冗長化する方法も解説します。

7 章 Ubuntu のネットワーク管理

Ubuntu のネットワーク管理に使われるツール (Netplan・UFW) について解説します。

利用上の注意

本教科書は、Markdown 形式で執筆を行い、Pandoc を使用して PDF 形式および EPUB 形式に変換しています。変換ツールの使用上、以下のような問題が判明しています。可能な限り対応していますが、技術的な制約のため問題が残ってしまっている点に注意してください。

- (ハイフンが) 2 つ続く場合の問題

本文中で-が2つ続くと、長い-になってしまう問題が確認されています。Web ブラウザ上での取り扱いでは回避方法が確認されていますが、Pandoc による変換では正常に動作しないため、コマンドのオプションやパーミッション表記などで-を2つ記述できません。この問題を回避するため、-と-の間にスペースを入力しています。見た目が少しおかしくなっていますが、この回避方法によるものです。

コマンド実行例などで空白が入ってしまう問題

コマンド実行例や URL など空白が入ってしまう問題が確認されています。-や/、:などの記号が含まれているとその前に空白が入るため、コピー&ペーストで実行しようとする正常に実行できない、URL をクリックして Web ブラウザにアクセスできないなどの問題が発生します。発生する理由が不明のため、問題を解決、回避できていません。このような問題が起きた場合には、コピー&ペーストする際に手動で空白を取り除いてください。

執筆者・制作者紹介

本教科書は、オープンなプロジェクト形式で開発を行っています。企画段階から意見交換を行い、事前の技術的な調査、執筆、レビューなどをプロジェクトのメンバーで分担して行っています。

宮原 徹 (1~6 章執筆担当・株式会社びぎねっと)

本教科書は、Linux/オープンソースソフトウェアをこれから勉強する皆さんと、熱心に指導に当たられている先生方の一助になればと思い、作成いたしました。「標準教科書」シリーズ3部作の改訂を完了させましたが、ネットワークの解説をかなり省略してしまったので、別途補完する形で本教科書を企画、執筆しました。

Linux はネットワークに接続することでその真価を発揮するので、ネットワークについてもしっかりと理解する一助になれば幸いです。

鯨井 貴博 (7 章執筆担当・株式会社ゼウス・エンタープライズ)

「Linux サーバー構築標準教科書 Ubuntu 版」に引き続き、Ubuntu のネットワーク管理について執筆しました。読者の方が様々な Linux ディストリビューションを使いこなせる技術者となるために役立てれば、幸いです。

開発にご協力をいただいた方々 (50 音順)

- ・秋庭 雅明 (データプロセス株式会社)
- ・大澤 文孝 (モウフカブール)
- ・小笠原 種高 (モウフカブール)
- ・桂 崇徳
- ・河原木 忠司
- ・中 翔 (土業 AI / FDE Consulting)
- ・西 侑一郎 (株式会社アーベルソフト)
- ・樋口 侑磨 (大学生)
- ・福永 昭臣 (株式会社ボード)

著作権

本教科書の著作権は特定非営利活動法人エルピーアイジャパンに帰属します。

Copyright© LPI-Japan. All Rights Reserved.

使用に関する権利

本教科書は、クリエイティブ・コモンズ・ライセンスの「表示 - 非営利 - 改変禁止 4.0 国際 (CC BY-NC-ND 4.0)」によってライセンスされています。

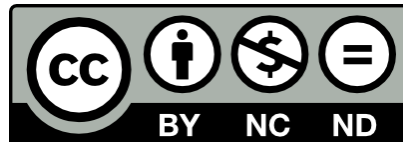


図 1: CC BY-NC-ND 4.0

表示

本教科書は、特定非営利活動法人エルピーアイジャパンに著作権が帰属するものであることを表示してください。

非営利

本教科書は、非営利目的で教材として自由に利用することができます。

商業上の利得や金銭的報酬を主な目的とした営利目的での利用は、特定非営利活動法人エルピーアイジャパンによる許諾が必要です。ただし、本教科書を利用した教育において、本教科書自体の対価を請求しない場合は、営利目的の教育であっても基本的に使用できます。その場合も含め、LPI-Japan 事務局までお気軽にお問い合わせください。

*営利目的の利用とは以下のとおり規定しております。営利企業または非営利団体において、商業上の利得や金銭的報酬を目的に本教科書の印刷実費以上の対価を受講生に請求して本教科書の複製を用いた研修や講義を行うこと。

改変禁止

本教科書は、改変せず使用してください。本教科書に対する改変は、特定非営利活動法人エルピーアイジャパンまたは特定非営利活動法人エルピーアイジャパンが認める団体により行われています。

フィードバック

フィードバックは誰でも参加できる **Slack** で受け付けていますので、積極的にご参加ください。Slack 参加の詳細は以下の本教科書の **Web** ページを参照してください。

<https://linuc.org/textbooks/network/>

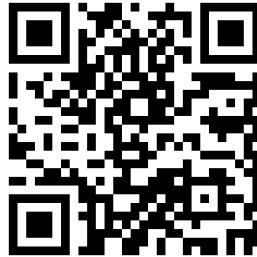


図 2: <https://linuc.org/textbooks/network/>

本教科書の使用に関するお問合せ先

特定非営利活動法人エルピーアイジャパン（LPI-Japan）事務局

お問い合わせ：<https://lpij.tayori.com/f/textbookinfo/>



図 3: <https://lpij.tayori.com/f/textbookinfo/>

Linux 技術者認定「LinuC（リナック）」のご紹介

Linux 技術者認定「LinuC（リナック）」とは、クラウド／DX 時代の IT エンジニアに求められるシステム構築から運用管理に必要なスキルを証明できる技術者認定です。アーキテクチャ設計からシステム構築、運用管理までの技術領域を広くカバーしており、4つのレベルの認定取得を通じて一歩ずつ確実に求められるスキルを習得し、それを証明することができます。

LinuC の出題範囲策定や試験開発は、実際に現場で活躍しているハイレベルな IT エンジニアが参加するコミュニティによって行われています。そのため、グローバルで業界標準として利用されている技術領域をカバーし、システム開発や運用管理の現場で本当に必要とされる知識や実践的なスキルを問う内容になっています。その結果として従来型の Linux 領域にとどまった技術認定とは異なり、国内・海外を問わず活躍を目指す IT エンジニアにとって、実践的かつ有用な技術者認定となっています。

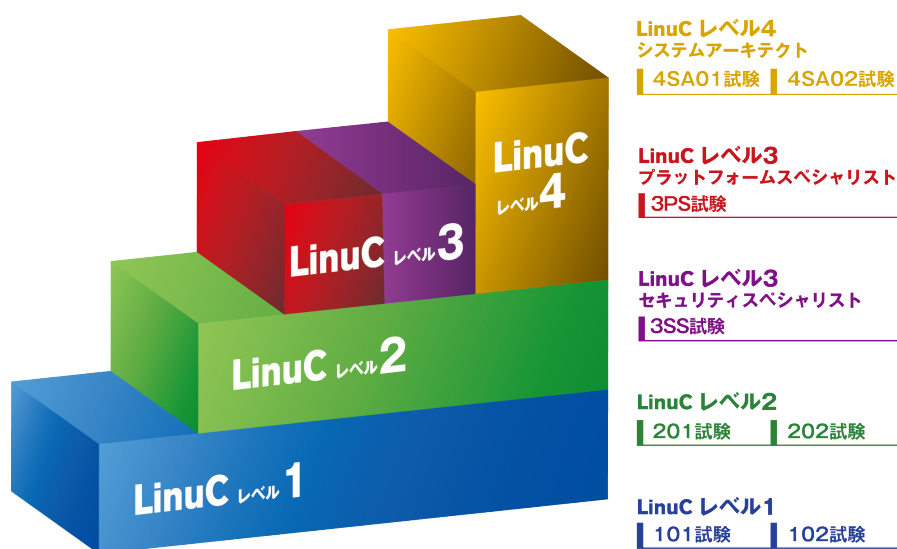


図 4: LinuC の体系図

LinuC レベル 1

コンピュータシステムを理解し、仮想環境を含む Linux システムの基本操作とシステム管理が行える即戦力エンジニアの証明（ITSS レベル 1）

LinuC レベル 2

仮想環境を含む Linux のシステム設計・ネットワーク構築において、アーキテクチャに基づいた設計・導入・保守・問題解決ができるエンジニアの証明（ITSS レベル 2）

LinuC レベル 3 プラットフォームスペシャリスト・セキュリティスペシャリスト

仮想化・自動化などにより柔軟性と可用性を備えた Linux プラットフォームの構築運用や、セキュア Linux システムのための OS からミドルウェアまでの各層堅牢化や認証認可基盤の構築および攻撃対策を実現できるスペシャリストの証明（ITSS レベル 3）

LinuC レベル 4 システムアーキテクト

オンプレ／クラウド、物理／仮想化を含むシステムのライフサイクル全体を俯瞰して最適なアーキテクチャを設計・構築ができる上級エンジニアの証明（ITSS レベル 4）

LinuC の詳細については、以下の Web サイトを参照してください。

<https://linuc.org/about/01.html>



図 5: <https://linuc.org/about/01.html>

LinuC の認定取得に向けた学習方法

LinuC の特徴の 1 つは学習環境が充実していることです。LinuC の認定取得に必要なスキルや知識を身につけるには体系的な学習として技術解説書を読むだけではなく、実際に手を動かして確認し問題集を繰り返し解いて理解を定着させる方法を推奨しています。

体系的にしっかりとした教育を実施している認定校や認定教材のほか、様々な学習コンテンツや無料で参加できるセミナーなどが豊富に用意されています。

詳しくは以下のリンク先「学習のすすめ方」をご覧ください。

<https://linuc.org/measures/>



図 6: <https://linuc.org/measures/>

LinuC の出題範囲を網羅した LPI-Japan 認定教材については以下のリンク先「認定教材」をご覧ください。

<https://linuc.org/measures/textbook/>



図 7: <https://linuc.org/measures/textbook/>

目次

1	ネットワーク学習環境の構築	1
1.1	実習用環境について	1
1.1.1	ホスト OS として Windows 11 が動作する PC	1
1.1.2	有線 LAN、あるいは Wi-Fi でインターネットに接続可能	1
1.1.3	Linux が動作する仮想マシン	1
1.2	インターネット接続用ルーター	1
1.2.1	Wi-Fi 接続と有線 LAN 接続	1
1.2.2	DHCP サーバー機能	1
1.2.3	企業内ネットワークとの違い	2
1.3	スイッチングハブ	2
1.3.1	スイッチは Ethernet 接続	2
1.3.2	OSI 7 階層	2
1.3.3	L3 スイッチ	3
1.3.4	カスケード接続	3
1.4	Wi-Fi 接続	3
1.4.1	Ethernet と Wi-Fi の関係	3
1.4.2	Wi-Fi のセキュリティ機能	3
1.5	VirtualBox で使用できるネットワーク	4
1.5.1	NAT 接続	4
1.5.2	ブリッジ接続	5
1.5.3	仮想マシンのみ接続	6
1.5.4	ホストオンリー接続	7
1.6	実習用仮想マシンの準備	8
1.6.1	仮想マシンのマウスがうまく動作しない	9
1.7	仮想マシンのネットワークのテスト	9
1.7.1	インターネットに接続できる	9
1.7.2	ホスト OS からゲスト OS に接続できる	10
1.8	ブリッジ接続を試してみる	10
1.8.1	実習のシナリオ	10
1.8.2	ブリッジ接続の前提条件	10
1.8.3	NAT 接続時の IP アドレスの確認	11
1.8.4	ブリッジ接続への変更	11
1.8.5	ブリッジ接続時の IP アドレスの確認	11
1.8.6	ホストから接続できることを確認	12
1.8.7	ホスト PC 以外からの接続	13
1.8.8	NAT 接続に戻す	13
1.9	VirtualBox のネットワーク設定	13
1.9.1	VirtualBox の高度な設定の有効化	13
1.9.2	ネットワーク設定の確認	13
1.10	ホストオンリーネットワークの確認	14
1.10.1	ホストオンリーアダプターの IP アドレス	14
1.10.2	ホストオンリーネットワークの DHCP サーバー	14
1.10.3	固定 IP アドレスの範囲を設定	14
1.10.4	ホストオンリーネットワークの DHCP の動作確認	14
1.11	Wireshark によるパケットキャプチャ	15
1.11.1	Wireshark のインストール	15
1.11.2	wireshark グループへの所属設定	15
1.11.3	グループ所属設定の反映	16
1.11.4	Wireshark の実行	16
1.11.5	インターフェースの指定	16
1.11.6	通信を行ってパケットをキャプチャする	16
1.11.7	IP アドレスを指定したフィルタリング	17
1.11.8	ポート番号を指定したフィルタリング	17
1.11.9	複数の条件を指定したフィルタリング	18
2	IP アドレス	19
2.1	用語について	19
2.2	IPv4 と IPv6	19
2.3	IP アドレスの表記	20

2.4	ネットワークアドレスとネットマスク	21
2.4.1	ネットマスクと AND 演算	22
2.4.2	ネットマスクの表記	23
2.4.3	ネットワークアドレスの表記	23
2.4.4	ネットワークアドレスの使い方	23
2.5	ルーティングとは	24
2.5.1	インターネットはルーターの集合体	24
2.5.2	デフォルトゲートウェイ	24
2.5.3	デフォルトゲートウェイの確認	24
2.6	ブロードキャストアドレス	25
2.7	アドレスクラス	26
2.7.1	CIDR	27
2.8	グローバル IP アドレスとプライベート IP アドレス	27
2.8.1	NAT	27
2.9	IP アドレスの変更	28
2.10	GUI による IP アドレスの変更	28
2.11	TUI による IP アドレスの変更	31
2.11.1	操作例	32
2.12	コマンド実行による IP アドレスの変更	34
2.12.1	デバイスの確認	34
2.12.2	デバイスの設定の確認	34
2.12.3	コネクションの確認	34
2.12.4	コネクションの詳細を確認	34
2.12.5	IP アドレスの変更	35
2.13	DHCP とは	35
2.13.1	VirtualBox の DHCP サーバー機能	36
2.13.2	DHCP で IP アドレスを取得する	36
2.13.3	CLI で DHCP で自動的に IP アドレスを取得する設定を行う	36
3	TCP/UDP/ICMP	38
3.1	TCP とは	38
3.1.1	3way ハンドシェイク	38
3.1.2	シーケンス番号と通信の保証	38
3.2	UDP とは	39
3.2.1	UDP はセッションレスプロトコル	39
3.3	ポート番号とは	39
3.3.1	Well-Known ポート	39
3.4	Registered ポート	40
3.4.1	ハイポート	40
3.4.2	/etc/services を参照する	40
3.5	ICMP	41
3.5.1	ICMP は TCP/UDP とは別のプロトコル	41
3.6	TCP の通信を Wireshark で見てみよう	41
3.6.1	パケットキャプチャを開始する	41
3.6.2	Web サイトにアクセスする	41
3.6.3	パケットキャプチャを停止する	41
3.6.4	条件でフィルタリングを行う	41
3.6.5	3way ハンドシェイクを確認する	42
3.6.6	その後のやり取りを観察する	42
3.7	UDP の通信を Wireshark で見てみよう	42
3.7.1	条件でフィルタリングを行う	42
3.7.2	UDP のやり取りを確認する	44
3.8	ICMP の通信を Wireshark で見てみよう	44
3.8.1	パケットキャプチャを開始する	44
3.8.2	ping コマンドを実行する	44
3.8.3	パケットキャプチャを停止する	44
3.8.4	条件でフィルタリングを行う	44
3.8.5	ICMP のやり取りを観察する	44
4	パケットフィルタリング	45
4.1	Linux のパケットフィルタリング機能 nftables	45

4.1.1	カーネルモジュールの確認	45
4.2	firewalld	46
4.2.1	firewalld サービスの確認	46
4.2.2	firewall-cmd	46
4.3	パケットフィルタリングの設定を確認する	46
4.3.1	ゾーン	47
4.3.2	ターゲット	47
4.4	ターゲットの変更	47
4.4.1	ターゲット REJECT に設定	47
4.4.2	ターゲット DROP に設定	48
4.5	ICMP の扱い	48
4.5.1	ターゲットを default に設定	49
4.5.2	icmp-block-inversion の設定	49
4.5.3	icmp-block-inversion の設定を元に戻す	50
5	ルーティング/IP マスカレード/プロキシ	51
5.1	ルーティング	51
5.1.1	ルーター	51
5.1.2	ルーターと L3 スイッチの違い	51
5.2	アドレス変換	51
5.2.1	NAT	51
5.2.2	NAPT	51
5.2.3	ポートフォワーディング	52
5.3	IP マスカレードによるアドレス変換	52
5.3.1	システムの構成	52
5.3.2	ネットワークインターフェースを確認する	52
5.3.3	ゾーンの確認	53
5.3.4	現在のゾーンの状態を確認	53
5.3.5	ネットワークインターフェースの接続ゾーンの変更	53
5.3.6	IP マスカレードの設定を確認	53
5.3.7	internal ゾーンのターゲットを ACCEPT に変更する	54
5.4	IP マスカレードの動作の確認	55
5.4.1	クライアント用仮想マシンを準備する	55
5.4.2	クライアント仮想マシンの参照 DNS とデフォルトゲートウェイを変更する	55
5.4.3	設定の確認	55
5.5	名前解決できることを確認	57
5.5.1	Web サーバーに接続できることを確認	57
5.6	Wireshark で IP マスカレードをしてみる	58
5.7	IP マスカレードを解除する	58
5.8	プロキシの設定	59
5.8.1	Squid のインストール	59
5.8.2	Squid の起動	59
5.8.3	ポートの待ち受け状況の確認	59
5.8.4	firewalld の設定変更	59
5.9	クライアント用仮想マシンのプロキシ設定	60
5.9.1	リモートログインではプロキシは設定されない	60
5.10	プロキシ経由でアクセスする	61
5.10.1	Squid のアクセスログ確認	61
5.10.2	プロキシ経由で Web サーバーにアクセス	61
6	ネットワークインターフェースの管理	62
6.1	ネットワークインターフェースとは	62
6.1.1	オンボード	62
6.1.2	ネットワークカード (NIC)	62
6.1.3	USB 接続	62
6.1.4	無線 LAN	62
6.2	実習環境のネットワークインターフェース	62
6.3	インターフェース名の命名規則	62
6.3.1	インターフェース名の意味	63
6.3.2	インターフェース種別とインターフェース名	63
6.4	インターフェースの詳細情報	63

6.5	ボンディングによるネットワーク冗長化	63
6.5.1	ボンディングの種類	64
6.5.2	負荷分散	64
6.5.3	スイッチ対応	64
6.6	ボンディングの準備	64
6.7	ボンディングの設定	65
6.7.1	ボンディングデバイスの作成	65
6.7.2	ネットワークインターフェースの確認	65
6.7.3	デバイスの追加	66
6.7.4	ボンディングデバイスの有効化	66
6.7.5	ボンディングデバイスの確認	66
6.8	ボンディングデバイスの自動構成	66
6.9	ボンディングの動作確認	67
6.10	ボンディングデバイスの冗長化テストを行う	68
7	Ubuntu のネットワーク管理	69
7.1	実習で使用する Ubuntu のバージョン	69
7.1.1	実習環境の構築	69
7.2	Ubuntu のネットワーク設定は Netplan を使用	69
7.3	Ubuntu Desktop でのネットワーク設定	70
7.3.1	固定 IP アドレスを設定する	71
7.3.2	Wi-Fi を設定する	72
7.4	Ubuntu Server でのネットワーク設定	72
7.4.1	IP アドレスとデフォルトゲートウェイを確認する	72
7.4.2	Netplan の設定ファイル	72
7.4.3	固定 IP アドレスへの変更	73
7.4.4	変更の適用	73
7.4.5	DNS 設定の確認	74
7.4.6	Netplan から生成されたコンフィグの確認	74
7.5	ボンディングについて	74
7.6	パケットフィルタリング	74
7.6.1	UFW の状態を確認	75
7.6.2	UFW の制御ルールを追加	75
7.6.3	UFW をアクティブ化	75
7.6.4	UFW の制御ルールの確認	76
7.6.5	UFW の動作確認	76
7.6.6	Web サーバーへのアクセスを拒否してみる	76

1 ネットワーク学習環境の構築

Linux のネットワークについて学習をするための環境を構築します。

1.1 実習用環境について

本教科書の実習では、以下のような環境を想定しています。

1.1.1 ホスト OS として Windows 11 が動作する PC

x86_64 プロセッサを搭載した一般的な Windows PC を想定していますが、VirtualBox が動作する macOS、Linux でも可能です。

1.1.2 有線 LAN、あるいは Wi-Fi でインターネットに接続可能

インターネットに接続するためのルーターがあり、実習用の PC が有線 LAN、または Wi-Fi でネットワークに接続されていて、インターネットに接続できることを想定しています。

実習用の PC は DHCP を使って、IP アドレスが自動的に設定されます。

1.1.3 Linux が動作する仮想マシン

VirtualBox で仮想マシンを動作させ、AlmaLinux 9.8 をゲスト OS として動作させます。実習用仮想マシンの準備は後ほど解説します。

仮想マシン実行環境は VirtualBox 以外のものでも構いませんが、ネットワーク設定はそれぞれの仮想マシンソフトウェアによって異なるので、機能や設定方法は個別に確認してください。

1.2 インターネット接続用ルーター

実習環境は、インターネット接続用のルーターが存在し、そのルーターを通じてインターネットに接続できることを想定しています。

以後、単に「ルーター」と記述した場合、このインターネット接続用のルーターを指します。

1.2.1 Wi-Fi 接続と有線 LAN 接続

想定しているルーターは、一般の家庭で使用する Wi-Fi のアクセスポイントを兼ねたものです。Wi-Fi の接続の他、スイッチングハブ機能を内蔵し、有線 LAN でも接続が可能です。

1.2.2 DHCP サーバー機能

ルーターは DHCP サーバー機能も備えており、接続した PC やスマートフォンなどに IP アドレスやデフォルトゲートウェイ、名前解決のための DNS サーバーの情報などを提供します。

想定ネットワーク構成図

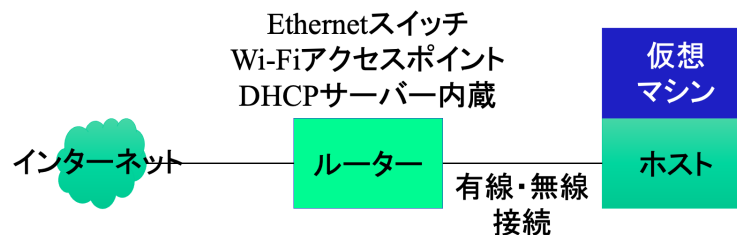


図 8: 想定ネットワーク構成

1.2.3 企業内ネットワークとの違い

企業など数多くのクライアントがネットワークに接続する環境の場合、インターネット接続用のルーター、Wi-Fiのアクセスポイント、DHCPサーバーなどの機能はそれぞれ別々の機器を使用して構成することが多くなります。

家庭用ルーターではすべての機能がオールインワンで備わっていますが、ネットワークにどのような機能が必要なのか理解する必要があります。

1.3 スイッチングハブ

有線 LAN で PC を相互に接続する場合、スイッチングハブが必要です。

接続速度は 1Gbps が主流ですが、最近では 2.5Gbps や 10Gbps といった高速なネットワークを構築できるものも増えています。

以後、単に「スイッチ」と記述した場合、スイッチングハブのことを指します。

1.3.1 スwitchは Ethernet 接続

スイッチは Ethernet と呼ばれるプロトコルで相互に接続します。

Ethernet は「OSI 7 階層」で下から 2 番目の層に当たるため、「L2 スwitch」と呼ばれることがあります。

1.3.2 OSI 7 階層

OSI 7 階層は ISO が定めた「OSI 参照モデル」の通称で、通信機能を機能別に 7 つの階層に分けて定義したものです。

階層数	名称	例
第 7 層	アプリケーション層	HTTP など
第 6 層	プレゼンテーション層	文字コードなど
第 5 層	セッション層	ソケットなど
第 4 層	トランスポート層	TCP など
第 3 層	ネットワーク層	IP など
第 2 層	データリンク層	Ethernet など
第 1 層	物理層	ケーブルなど

OSI参照モデルとTCP/IP

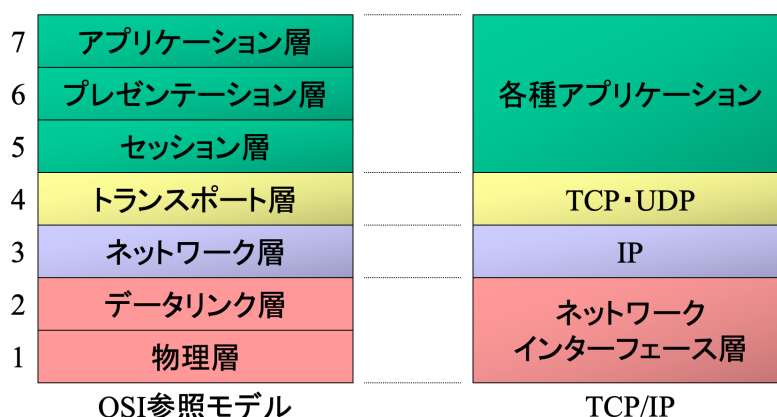


図 9: OSI7 階層

1.3.3 L3 スイッチ

スイッチの中には、IP によるルーティングの機能を内蔵したものもあり、OSI7 階層の第 3 階層に当たることから「L3 スイッチ」と呼ばれます。

1.3.4 カスケード接続

ルーターにはスイッチ機能が備えられていますが、ポート数が限られています。より沢山の PC を有線 LAN 接続する場合には、別途スイッチを用意する必要があります。

スイッチとスイッチを相互に接続することを「カスケード接続」と呼びます。カスケード接続すると、1 つのポート数の多いスイッチとして機能するようになります。

1.4 Wi-Fi 接続

Wi-Fi 接続は、有線 LAN のようにケーブルの接続が不要なため、PC やスマートフォンをネットワークに接続するために多く使われるようになりました。

1.4.1 Ethernet と Wi-Fi の関係

Wi-Fi は、「IEEE 802.11」という国際標準規格に則った無線 LAN のことを指します。

有線 LAN で使っている Ethernet は「IEEE 802.3」として規格化されており、どちらも同じ L2 ネットワークを担っています。有線 LAN と無線 LAN は相互に接続が可能です。

1.4.2 Wi-Fi のセキュリティ機能

実習では、同じネットワークに接続されている別の PC から仮想マシン上のゲスト OS にアクセスすることがあります。

一部の Wi-Fi ルーターには、セキュリティを高めるため Wi-Fi 接続している PC の相互接続を許さない設定がされている場合があります。この機能は「プライバシーセパレータ」や「AP アイソレーション」と呼ばれます。

実習で別の PC から仮想マシンにアクセスできない場合、この機能が有効になっていないか、ルーターの管理画面で確認してみてください。

1.5 VirtualBox で使用できるネットワーク

VirtualBox では、様々な接続の仮想ネットワークが利用できます。

大きく分けて、ホスト以外の外部のコンピューターと接続する「外部接続」ネットワークと、「仮想マシンのみ」、「仮想マシンとホスト」の 3 種類に分類できます。

外部接続は、さらに IP による「NAT」接続と、イーサネットによる「ブリッジ」接続に分けられます。

	名称	接続方法
外部	NAT	外部に NAT 接続 (L3)
外部	ブリッジ	外部にブリッジ (L2)
内部	内部ネットワーク	仮想マシンのみ
内部	ホストオンリー	仮想マシンとホスト

以下、主に使用するネットワーク接続について解説します。これら以外のネットワークについては説明は割愛します。

1.5.1 NAT 接続

仮想マシンの設定では「NAT」という選択肢になっています。

ゲスト OS からの外部への接続をホスト OS からの接続に変換します。変換は IP アドレスと TCP/UDP のポート番号を使用しているので、厳密に言えば「NAPT」(Network Address Port Translation)を行っています。

NAT接続

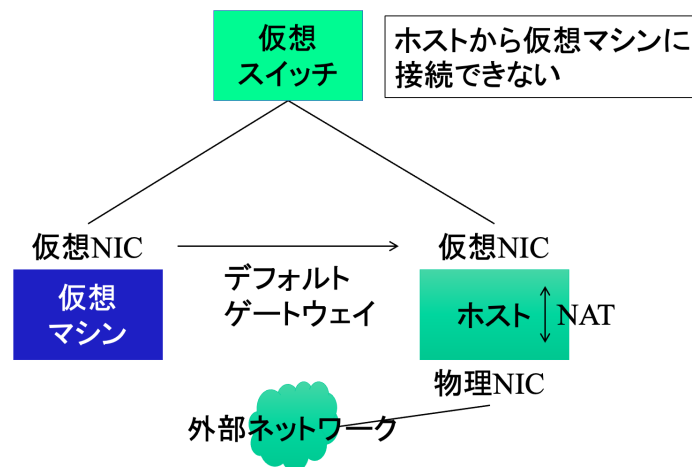


図 10: NAT 接続

1.5.2 ブリッジ接続

仮想マシンの設定では「ブリッジアダプター」という選択肢になっています。

仮想マシンからの外部への接続をホストマシンのネットワークインターフェースに Ethernet で接続します。Ethernet のレベルで相互に接続することを「ブリッジ」と呼びます。

ブリッジ接続

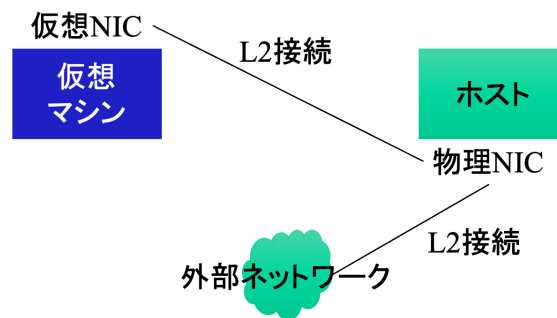


図 11: ブリッジ接続

1.5.3 仮想マシンのみ接続

仮想マシンの設定では「内部ネットワーク」という選択肢になっています。

仮想マシンのみを相互に接続する仮想スイッチに接続します。外部に一切接続できないため、動作が他の PC に影響したり、セキュリティ的に問題があったりする検証を行う際に使用します。

内部ネットワーク接続

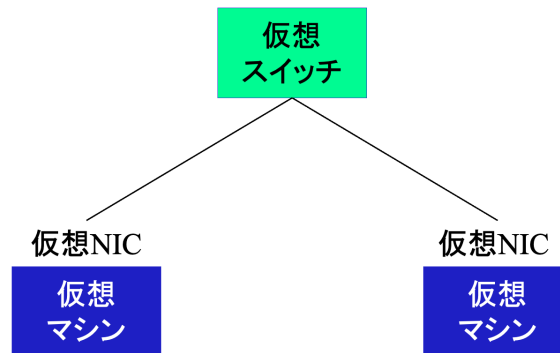


図 12: 内部ネットワーク接続

1.5.4 ホストオンリー接続

仮想マシンの設定では「ホストオンリーアダプター」という選択肢になっています。

仮想マシンとホスト OS に用意されたホストオンリーアダプターを相互に接続する仮想スイッチに接続します。

ホストオンリー接続

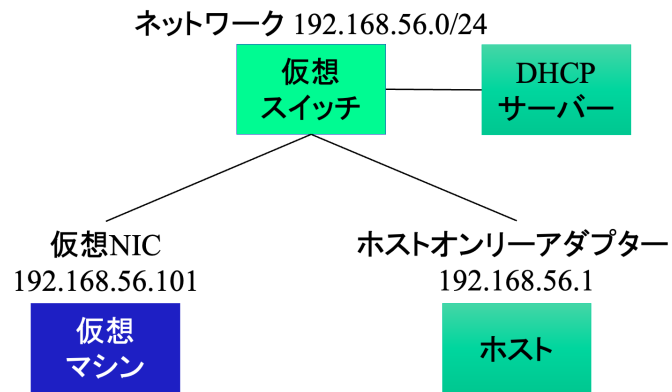


図 13: ホストオンリー接続

「192.168.56.1」はホストに設定されるデフォルトの IP アドレスです。

1.6 実習用仮想マシンの準備

この後、実習で使用する仮想マシンを準備します。『Linux 標準教科書』や『Linux サーバー構築標準教科書』などを参考に、仮想マシンを作成し、ゲスト OS をインストールしてください。

仮想マシンの作成手順はまったく同じですが、本教科書でのポイントは以下の通りです。

- ゲスト OS は AlmaLinux 9.8
- 仮想ネットワークはネットワークアダプター 1 を「NAT」、ネットワークアダプター 2 を「ホストオンリーアダプター」に設定

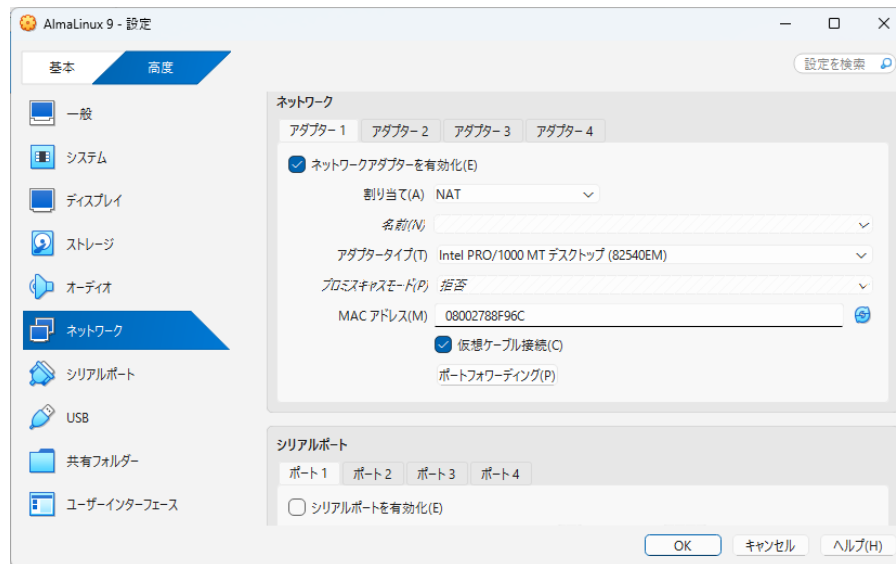


図 14: ネットワークアダプター 1 を「NAT」に設定

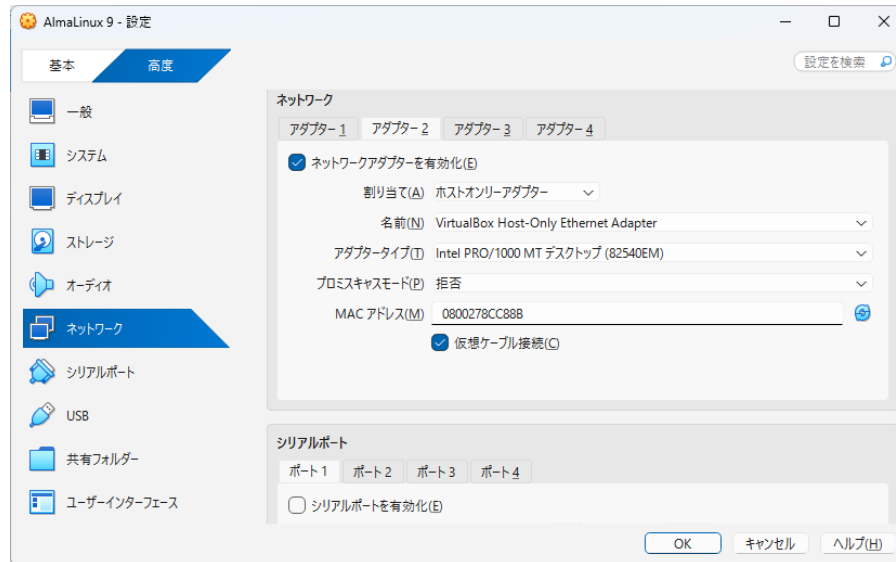


図 15: ネットワークアダプター 2 を「ホストオンリーアダプター」に設定

1.6.1 仮想マシンのマウスがうまく動作しない

仮想マシンにゲスト OS をインストールする際に、インストーラーを操作するためにマウス動作をゲスト OS 側にする必要があります。これをマウスキャプチャと呼びます。

一部の環境では、マウスキャプチャをするとマウスカーソルがホスト OS 側の左上に移動してしまうようです。

この問題を解決するには、仮想マシンの設定でポインティングデバイスを「PS/2 マウス」から「USB タブレット」に変更します。設定の変更は仮想マシンが停止している状態で行う必要がありますので、インストーラーが起動している場合には仮想マシンをシャットダウンしてください。

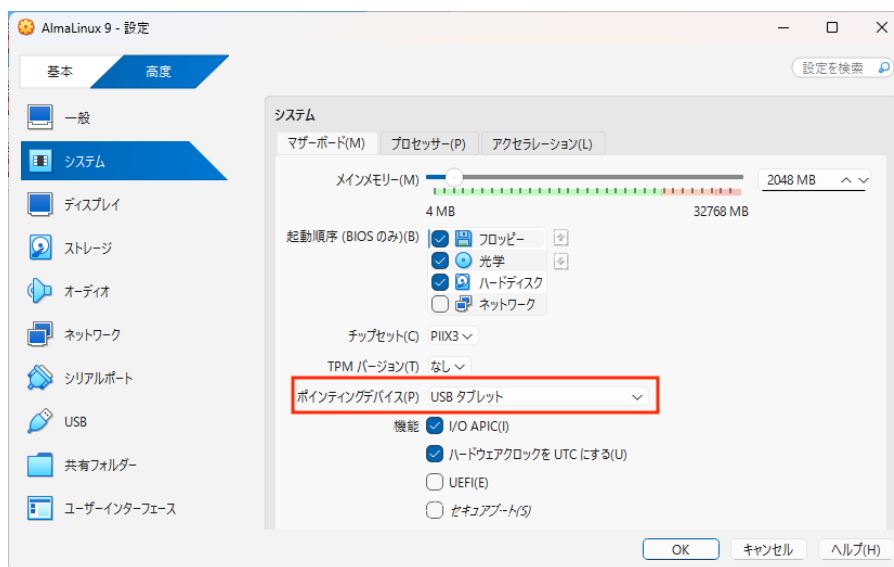


図 16: 仮想マシンのポインティングデバイスを「USB タブレット」に変更

1.7 仮想マシンのネットワークのテスト

仮想マシンを作成したら、以下の操作が行える事を確認しておいてください。

1.7.1 インターネットに接続できる

ゲスト OS で Web ブラウザを起動し、インターネット上の Web サイトにアクセスできることを確認してください。

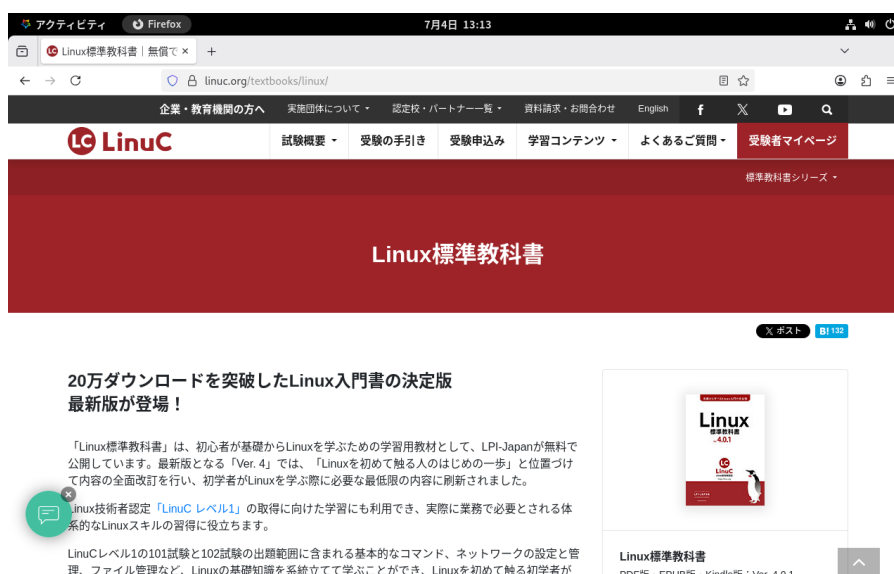


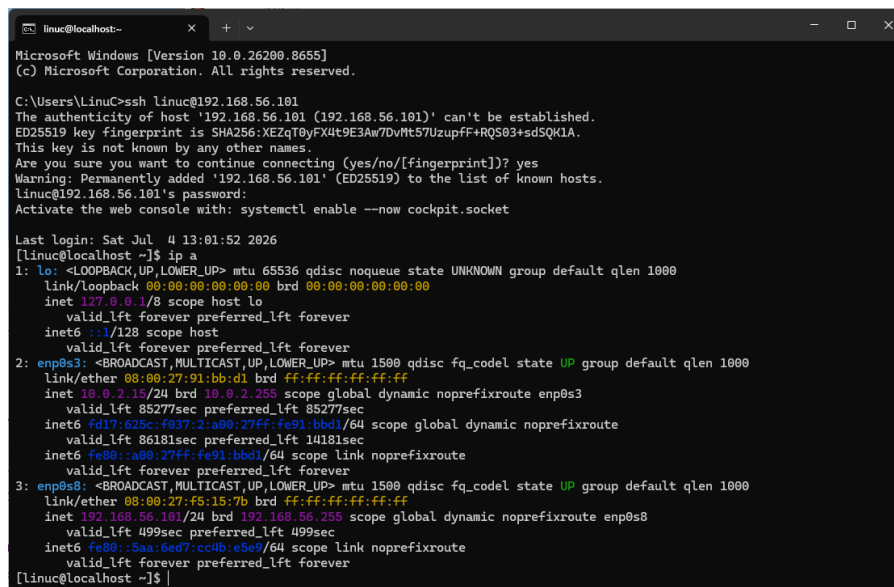
図 17: インターネット上の Web サイトにアクセス

この操作が行えると、以下の事項が行えていることが確認できます。

- ゲスト OS に NAT 用の IP アドレスが割り当てられている
- 名前解決が行えている
- VirtualBox の NAT が機能している
- ルーターからインターネットに接続できる

1.7.2 ホスト OS からゲスト OS に接続できる

ホスト OS でターミナルを起動し、SSH でゲスト OS に接続できることを確認してください。



```

Microsoft Windows [Version 10.0.26280.8655]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Linuc>ssh linuc@192.168.56.101
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:XEZqT0yFX4t9E3Aw7DvMt57UzupFF+RQ503+sdSQK1A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.101' (ED25519) to the list of known hosts.
linuc@192.168.56.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Jul 4 13:01:52 2026
[linuc@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:91:bb:d1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 85277sec preferred_lft 85277sec
    inet6 fd17:625c:f037:2:a00:27ff:fe91:bbd1/64 scope global dynamic noprefixroute
        valid_lft 86181sec preferred_lft 14181sec
    inet6 fe80::a00:27ff:fe91:bbd1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 499sec preferred_lft 499sec
    inet6 fe80::8a4:6a07:cc4b:e8a9/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[linuc@localhost ~]$
  
```

図 18: SSH でゲスト OS に接続

この操作が行えると、以下の事項が行えていることが確認できます。

- ゲスト OS にホストオンリーネットワーク用の IP アドレスが割り当てられている
- ホスト OS にホストオンリーアダプターが設定されている
- ホストオンリーネットワークを経由してホスト OS とゲスト OS が相互に接続できている

1.8 ブリッジ接続を試してみる

VirtualBox の接続のうち、これまで使用していないブリッジ接続を試してみましょう。

1.8.1 実習のシナリオ

この状態で、仮想マシンのネットワークアダプター 1 を「NAT」から「ブリッジアダプター」に変更してみます。

ゲスト OS のネットワークインターフェースに、外部で動作している DHCP サーバーから IP アドレスが設定されること、インターネットに接続できることも確認します。

また、可能であればホスト PC 以外の PC からの接続も確認してみます。

1.8.2 ブリッジ接続の前提条件

ブリッジ接続を試すには、以下の前提条件を想定しています。

- ホスト PC が接続しているネットワークで DHCP サーバーが動作している
- ホスト OS も DHCP クライアントとして IP アドレスを取得している
- 仮想マシンの NAT 接続でゲスト OS がインターネットに接続できている

1.8.3 NAT 接続時の IP アドレスの確認

NAT 接続を行っている時の IP アドレスを確認します。ゲスト OS で `ip a` コマンドを使って確認します。

```
$ ip a
(略)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:91:bb:d1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
(略)
```

VirtualBox の NAT 接続は、デフォルトで VirtualBox の用意した DHCP サーバー機能が動作しており、ゲスト OS に対して「10.0.2.15」の IP アドレスを設定するようになっています。

1.8.4 ブリッジ接続への変更

仮想マシンのネットワークアダプターの設定を「NAT」から「ブリッジアダプター」に変更します。手順は以下の通りです。

1. Oracle VirtualBox マネージャーで「仮想マシン」を選択します。
2. 使用している仮想マシンを選択します。
3. 「設定」ボタンをクリックします。
4. 「ネットワーク」を選択します。
5. 「アダプター 1」タブの割り当てを「NAT」から「ブリッジアダプター」に変更します。
6. 「OK」をクリックします。

ネットワークアダプターの設定変更は即時有効になります。

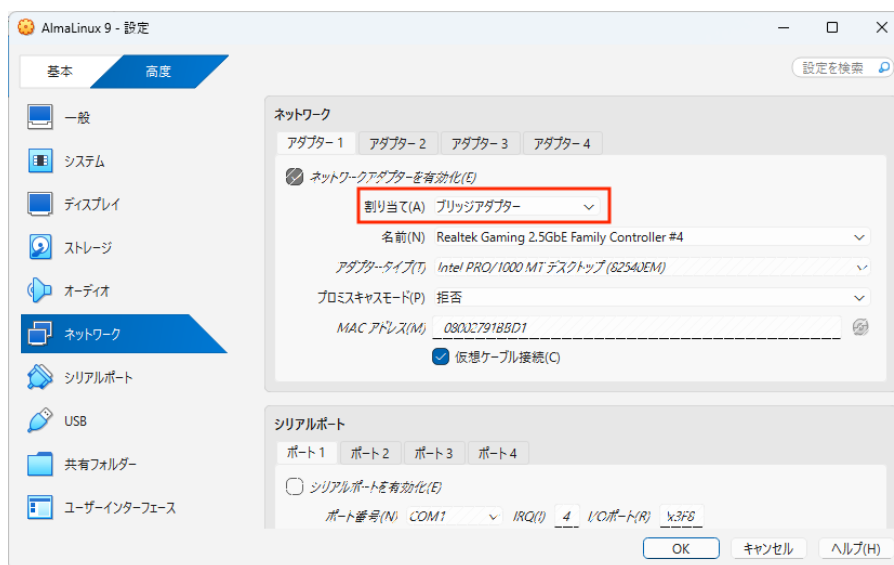


図 19: 仮想マシンのネットワークアダプター 1 を「ブリッジアダプター」に変更

1.8.5 ブリッジ接続時の IP アドレスの確認

再度、ゲスト OS で IP アドレスを確認します。

```
$ ip a
(略)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:91:bb:d1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.140/24 brd 192.168.11.255 scope global dynamic
        noprefixroute enp0s3
(略)
```

外部の DHCP サーバーから IP アドレスが設定されていることが確認できます。

```

[linuc@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp8s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:91:bb:d1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.140/24 brd 192.168.11.255 scope global dynamic noprefixroute enp8s3
        valid_lft 172784sec preferred_lft 172784sec
    inet6 fd17:625c:f937:2:a08:27ff:fe91:bbd1/64 scope global dynamic noprefixroute
        valid_lft 86257sec preferred_lft 14257sec
    inet6 2409:11:1b40:db00:a00:27ff:fe91:bbd1/64 scope global dynamic noprefixroute
        valid_lft 86398sec preferred_lft 14398sec
    inet6 fe80::a08:27ff:fe91:bbd1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp8s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp8s8
        valid_lft 539sec preferred_lft 539sec
    inet6 fe80::5aa:6ed7:cc4b:e5e9/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[linuc@localhost ~]$

```

図 20: ブリッジ接続の IP アドレスを確認

Web ブラウザを起動して、インターネットに接続できることも確認してみてください。

1.8.6 ホストから接続できることを確認

ブリッジ接続は、ホスト OS からゲスト OS への接続も可能です。ゲスト OS に割り当てられた IP アドレスに対して、ping コマンドを実行したり、SSH でリモートログインしたりしてみます。

ホスト OS から、ゲスト OS に対して PING を実行してみます。

```

C:\Users\LinuC>ping 192.168.11.140

192.168.11.140 に ping を送信しています 32 バイトのデータ:
192.168.11.140 からの応答: バイト数 =32 時間 =11ms TTL=64
192.168.11.140 からの応答: バイト数 =32 時間 <1ms TTL=64
192.168.11.140 からの応答: バイト数 =32 時間 =1ms TTL=64

192.168.11.140 の ping 統計:
    パケット数: 送信 = 3、受信 = 3、損失 = 0 (0% の損失)、
    ラウンドトリップの概算時間 (ミリ秒):
        最小 = 0ms、最大 = 11ms、平均 = 4ms
Ctrl+C
^C

```

ホスト OS から、ゲスト OS に対して SSH 接続してみます。

```

C:\Users\LinuC>ssh linuc@192.168.11.140
The authenticity of host '192.168.11.140 (192.168.11.140)' can't be established.
ED25519 key fingerprint is SHA256:XEZqT0yFX4t9E3Aw7DvMt57UzupfF+RQS03+sdSQK1A.
This host key is known by the following other names/addresses:
    C:\Users\LinuC/.ssh/known_hosts:1: 192.168.56.101
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.11.140' (ED25519) to the list of known
hosts.
linuc@192.168.11.140's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Jul  6 14:17:36 2026 from 192.168.11.123
[linuc@localhost ~]$

```

1.8.7 ホスト PC 以外からの接続

ネットワークに別の PC がある場合には、ホスト PC 以外から接続できることを確認してみてください。

両方の PC が Wi-Fi で接続している場合、Wi-Fi のアクセスポイントの設定によっては相互に接続ができない場合がありますので、その時はアクセスポイントのセキュリティの設定を確認してみてください。

ホスト PC 以外の PC で作業を続けたい場合には、次の NAT 接続に戻す作業は行わなくても構いません。

1.8.8 NAT 接続に戻す

ブリッジ接続の確認が一通り終わったら、ネットワークアダプター 1 の接続を「NAT」に戻しておきます。

ゲスト OS は、ネットワークの接続が変更されたことを自動的に認識して、再度 IP アドレスを設定します。

1.9 VirtualBox のネットワーク設定

VirtualBox は、デフォルトでホスト OS 側に「ホストオンリーアダプター」という仮想ネットワークインターフェースを作成して「192.168.56.1」の IP アドレスを設定します。そしてホストオンリーネットワークに接続した仮想マシンに対して DHCP サーバーで IP アドレスを自動設定できるようにします。

この設定を確認したり、IP アドレスを変更したりするには、「Oracle VirtualBox マネージャー」のネットワーク設定ツールを使います。

VirtualBox はデフォルトでネットワーク設定ツールが使用できるようになっていますが、設定が変更されていてネットワーク設定ツールが使用できない場合には、VirtualBox の環境設定で高度な設定が行えるように変更する必要があります。

1.9.1 VirtualBox の高度な設定の有効化

VirtualBox の高度な設定を有効化するには、「Oracle VirtualBox マネージャー」のウインドウで「ファイル」→「環境設定」をクリックして VirtualBox の環境設定画面を呼び出します。

ウインドウ左上の「高度」をクリックし、「OK」ボタンをクリックします。

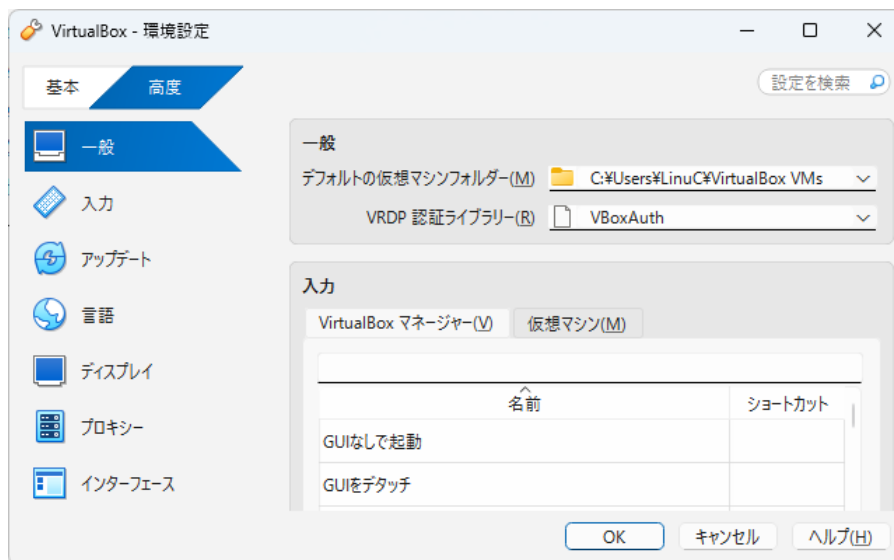


図 21: 設定を高度に変更

Oracle VirtualBox マネージャーの左側の選択肢が増えており、ネットワーク設定ツールが使用可能になっていることが確認できます。

1.9.2 ネットワーク設定の確認

ネットワークの設定画面では、「ホストオンリーネットワーク」、「NAT ネットワーク」、「クラウドネットワーク」の設定が行えます。

ここでは、ホストオンリーネットワークの設定を確認してみます。

1.10 ホストオンリーネットワークの確認

「ホストオンリーネットワーク」タブを選択します。リストに現在設定されているホストオンリー接続のためのネットワークが表示されます。

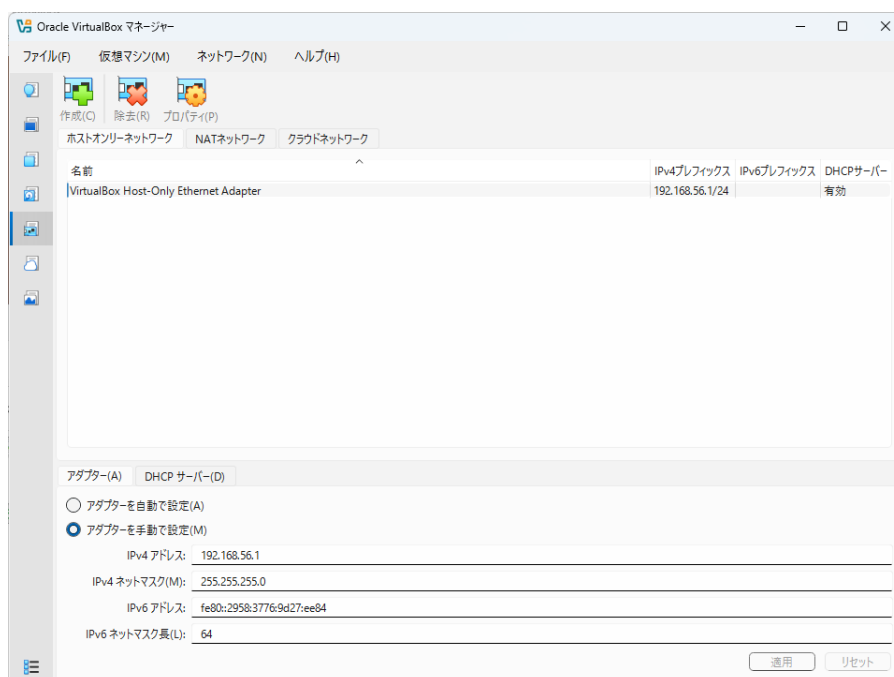


図 22: ネットワークの設定

1.10.1 ホストオンリーアダプターの IP アドレス

リストからネットワークを選択すると、画面下の「アダプター」タブに、そのホストオンリーネットワークに接続されているホスト OS 側のホストオンリーアダプターの IP アドレス設定が表示されます。

デフォルトで「192.168.56.1」に設定されています。

ホスト OS 側の IP アドレスを変更したい場合には、ここを変更を行います。

1.10.2 ホストオンリーネットワークの DHCP サーバー

「DHCP サーバー」タブでは、そのホストオンリーネットワークに接続された仮想マシンに対して IP アドレスの自動設定を行うための DHCP サーバーのパラメータが設定できます。

ホストオンリーアダプターの IP アドレスを変更した場合には、それに合わせてこちらを変更する必要があります。

サーバーアドレスは DHCP サーバー自体の IP アドレスです。デフォルトで「192.168.56.100」に設定されています。

アドレス下限からアドレス上限の範囲が、仮想マシンに対して自動設定される IP アドレスの範囲です。

1.10.3 固定 IP アドレスの範囲を設定

仮想マシンに固定 IP アドレスを割り当てたい場合には、DHCP サーバーのアドレス下限または上限を変更して、固定 IP アドレスとして手動で割り当て可能な IP アドレスの範囲を設定する必要があります。

たとえば、デフォルトのようにアドレス下限を「192.168.56.101」に設定すると、「192.168.56.2～192.168.56.99」は DHCP で自動的に設定されません。

1.10.4 ホストオンリーネットワークの DHCP の動作確認

起動している実習用仮想マシンで、ホストオンリーネットワークに接続されたインターフェースの IP アドレスを確認してみます。

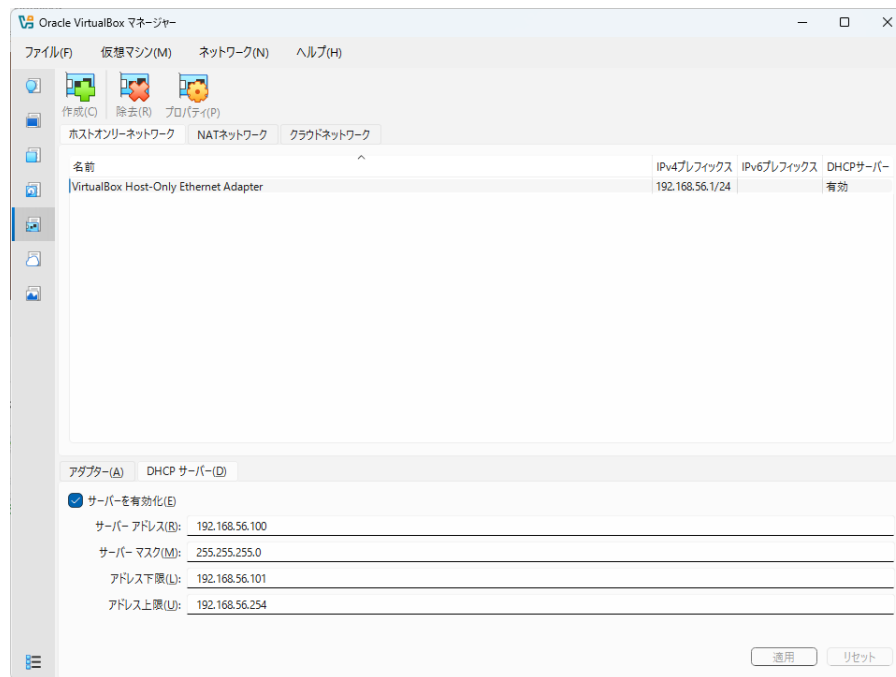


図 23: DHCP サーバーの設定

```
$ ip a
(略)
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   group default qlen 1000
   link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
   inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic
       noprefixroute enp0s8
(略)
```

「192.168.56.101」が設定されているのがわかります。

1.11 Wireshark によるパケットキャプチャ

ネットワークを流れている通信は、パケットという単位でやり取りが行われています。通常はパケットを見ることはできませんが、パケットキャプチャを行うことで見るができます。

パケットキャプチャツールの **Wireshark** をインストールして、パケットキャプチャを行ってみましょう。

1.11.1 Wireshark のインストール

dnf コマンドを実行して、**Wireshark** をインストールします。

```
$ sudo dnf install wireshark
```

1.11.2 wireshark グループへの所属設定

Wireshark を実行するユーザーは **wireshark** グループに所属している必要があります。

usermod コマンドを実行して、ユーザーのサブグループに **wireshark** グループを追加します。サブグループの追加なので、**usermod** コマンドに **-a** オプションをつけて実行してください。

```
$ sudo usermod -a -G wireshark linuc
```

1.11.3 グループ所属設定の反映

グループ所属の設定を GUI 環境に反映させるために、システム再起動を行います。

再起動後、ログインして `id` コマンドを実行してユーザーのサブグループに `wireshark` グループが追加されたことを確認します。

```
$ id
uid=1000(linuc) gid=1000(linuc) groups=1000(linuc),10(wheel),984(wireshark)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

1.11.4 Wireshark の実行

Wireshark を実行します。

GUI から実行する場合には、画面左上の「アクティビティ」をクリックし、画面下の Dock の一番右の「アプリケーションを表示する」をクリックして、Wireshark のアイコンをクリックします。

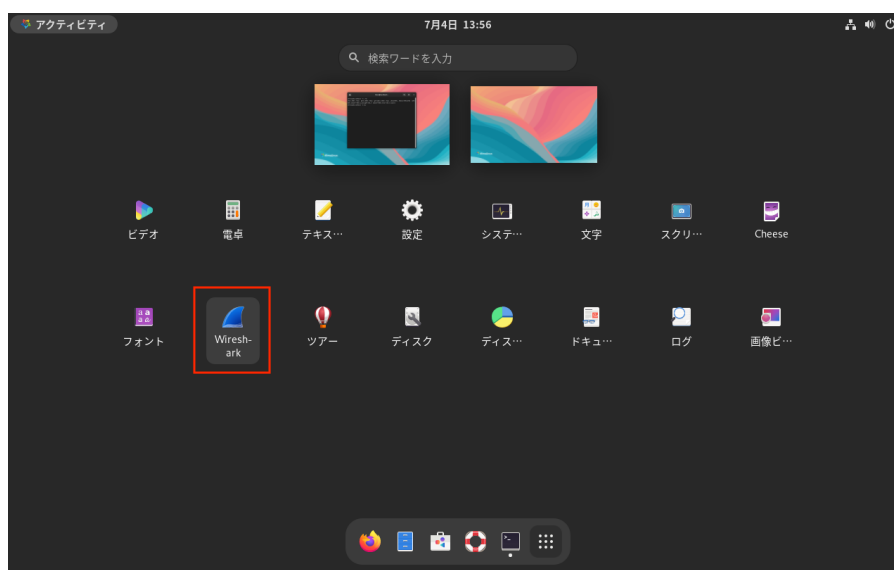


図 24: Wireshark をアイコンから起動

コマンドラインから実行する場合には、`wireshark` コマンドを実行します。

```
$ wireshark &
```

1.11.5 インターフェースの指定

パケットキャプチャを行うネットワークインターフェースを指定します。

ここでは、NAT 接続でインターネットと通信ができる `enp0s3` を選択します。

画面左上の「パケットキャプチャを開始します」ボタンをクリックします。

1.11.6 通信を行ってパケットをキャプチャする

指定したインターフェースを使ってネットワーク通信を行い、パケットをキャプチャします。

特別何もしないでも、バックグラウンドで様々なプロセスが通信を行っていますが、Web ブラウザを起動してインターネット上のサイトにアクセスしてみます。

ある程度パケットキャプチャが行えたら、画面左上の「パケットキャプチャを停止します」ボタンをクリックします。

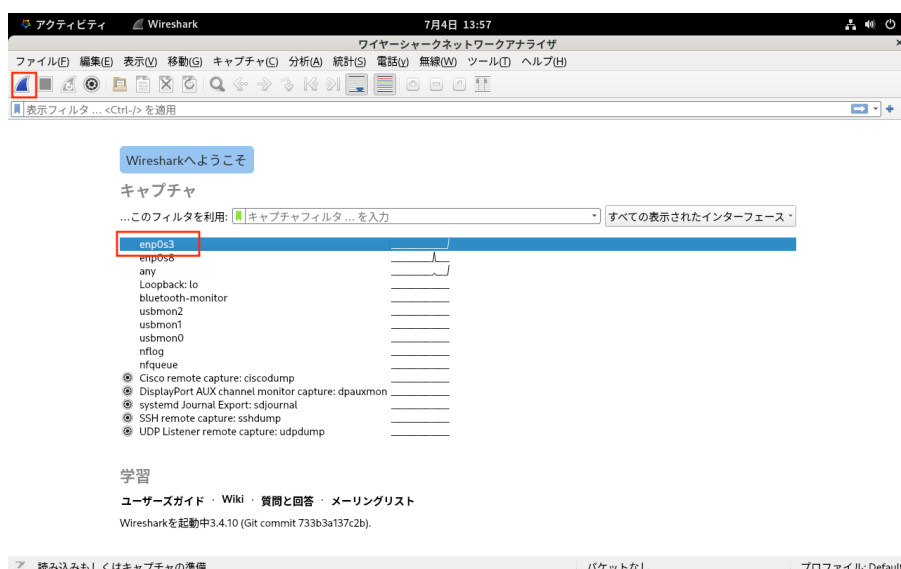


図 25: パケットキャプチャを行うインターフェースを選択

1.11.7 IP アドレスを指定したフィルタリング

パケットキャプチャの結果は非常に多くのパケットをキャプチャしているので、必要となるパケットのみに絞り込むためフィルタを設定します。

「表示フィルタ」と書かれたテキスト入力欄に以下の通りに入力すると、送信元が NAT 接続している IP アドレスのパケットのみに絞り込まれます。

```
ip.src==10.0.2.15
```

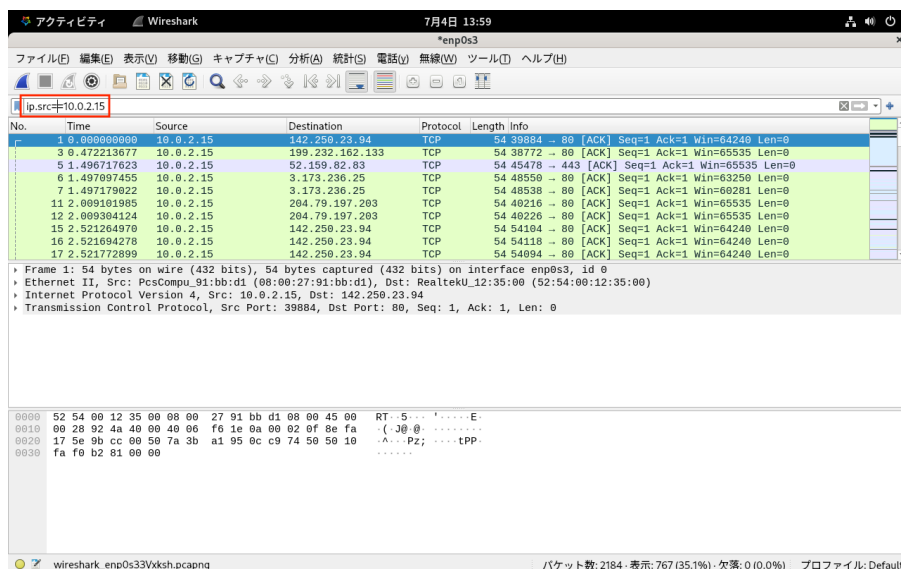


図 26: 送信元 IP アドレスで絞り込み

1.11.8 ポート番号を指定したフィルタリング

フィルタに以下の通りに入力すると、宛先の TCP ポートが 443 番、HTTPS プロトコルの通信パケットのみに絞り込まれます。

```
tcp.dstport==443
```

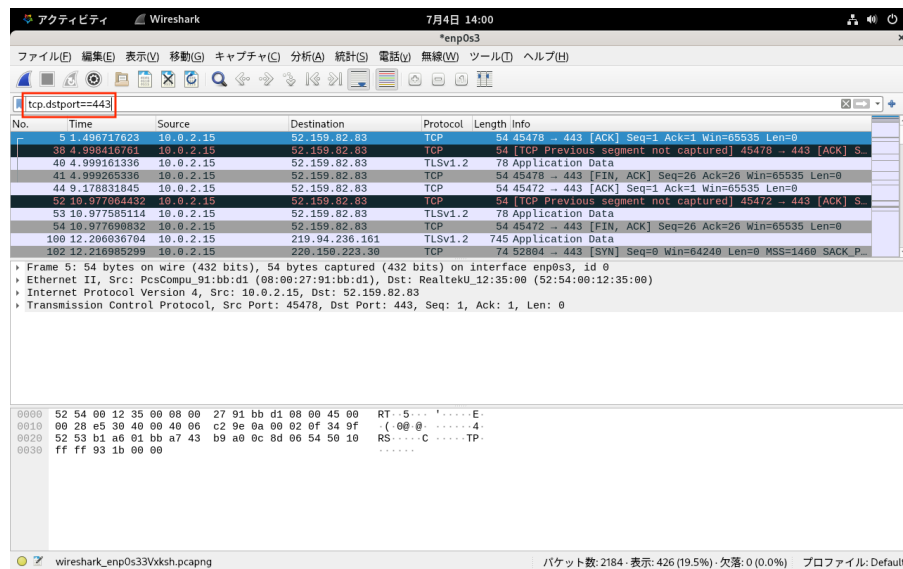


図 27: 宛先 TCP ポートで絞り込み

1.11.9 複数の条件を指定したフィルタリング

上記 2 つの条件を同時に指定したい場合には、2 つの条件を「&&」でつなげて指定します。

```
ip.src=10.0.2.15 && tcp.dstport==443
```

2 IP アドレス

IP アドレスは、インターネットの通信の基盤である IP (Internet Protocol) において、通信を行う各端末 (ノード) に割り当てられるアドレスです。適切な IP アドレスが割り当てられることで、ネットワークを経由してお互いに通信が行えます。

IP アドレスには様々な要素が関係してくるので、それらについて解説していきます。

2.1 用語について

ネットワークの解説では、ほぼ同様のものが文脈や役割によって呼び方が異なる場合があります。

たとえば、以下の言葉はすべて概ね同じく 1 台のコンピューターのことを指しています。

- ホスト
- マシン
- 端末
- ノード
- サーバー
- クライアント

2.2 IPv4 と IPv6

本教科書においては、IP といった場合には IPv4 を指します。

IPv4 の IP アドレスは 32 ビットで表されます。32 ビットで表せる IP アドレスの個数は約 43 億個となるので、世界中に張り巡らされたインターネット上で割り当てるのは数が不足しています。

そこで作られたのが IPv6 です。IPv6 は IP アドレスを 128 ビットで表すため、IP アドレスは約 340 澗 (かん) 個 (43 億の 4 乗) と事実上無制限の個数となります。

本教科書では IPv6 については解説しません。IPv4 について理解したら、IPv6 についても調べてみてください。

2.3 IP アドレスの表記

IP アドレスは 32 ビットのデータですが、0 と 1 の 2 進数表記が 32 個並ぶのでは可読性がないので、8 ビット（1 オクテット）ずつに区切って 10 進数で表記します。8 ビットは 10 進数では 0 から 255 までの値を取ります。これが 4 つ。（ドット）で区切られるので、それぞれを第 1 オクテット、第 2 オクテット、第 3 オクテット、第 4 オクテットと呼びます。

たとえば、「192.168.56.101 の第 3 オクテットは 56」というような言い方をします。

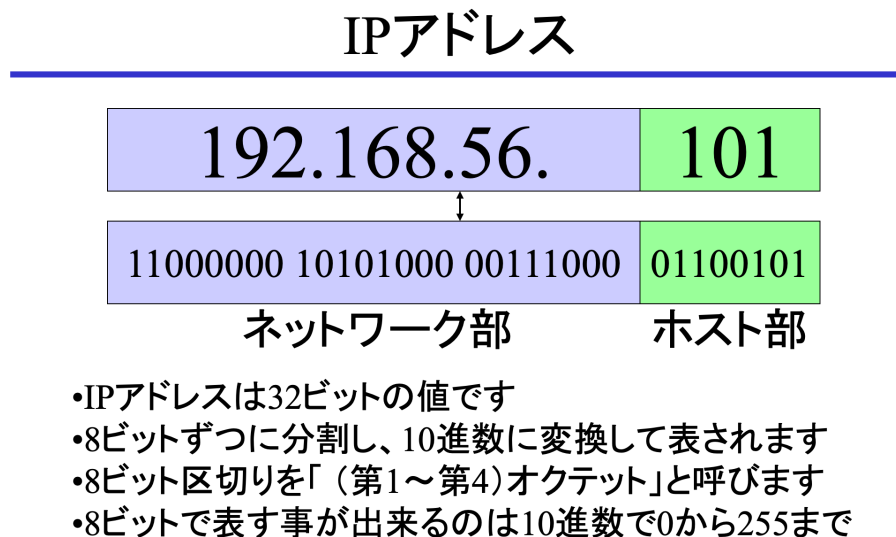


図 28: IP アドレス

2.4 ネットワークアドレスとネットマスク

ネットワークアドレスは、ノードが接続されているネットワーク全体を指し示すアドレスです。

インターネットを支える IP というプロトコルは、複数のノードが集まったネットワーク同士を相互に接続して通信を行う、という考え方に基いています。外部にある別のネットワークを指し示すのがネットワークアドレスです。

同じネットワークに接続されている各ノードの IP アドレスは、同じ「ネットワーク部」の値と、ノード毎に異なる「ホスト部」の値から成ります。どこまでがネットワーク部かを表すのが「ネットマスク」です。IP アドレスの 2 進数表記 32 ビットのうち、ネットマスクで決められた部分まではすべて同じアドレス、そして残りの部分は各ノード毎に異なるアドレス値を取ります。

ネットワークアドレス

同じネットワークアドレスを持つホスト同士は直接通信できる

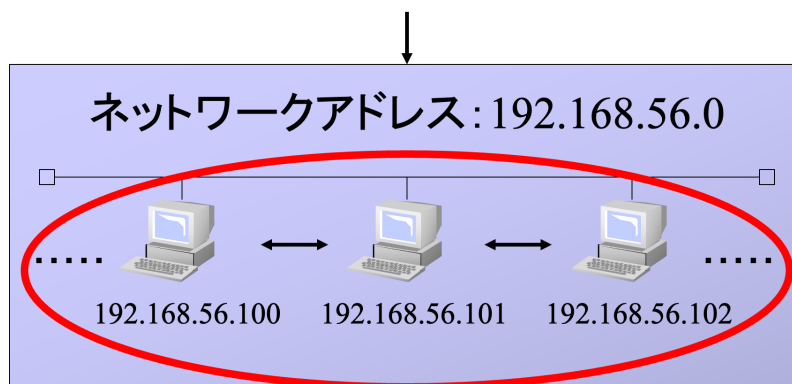


図 29: ネットワークアドレス

たとえば、「192.168.56.101」という IP アドレスのうち、第 3 オクテットまでの 24 ビットがネットワーク部とした時、「192.168.56.101/24」と IP アドレスの後ろにネットマスクの長さを記述します。この表記を「プレフィックス表記」と呼ぶことがあります。

2.4.1 ネットマスクと AND 演算

IP アドレスは内部的には 32 ビットの 2 進数となります。ネットマスクと AND 演算することで、ネットワークアドレスを取り出すことができます。

ネットマスクとネットワークアドレス

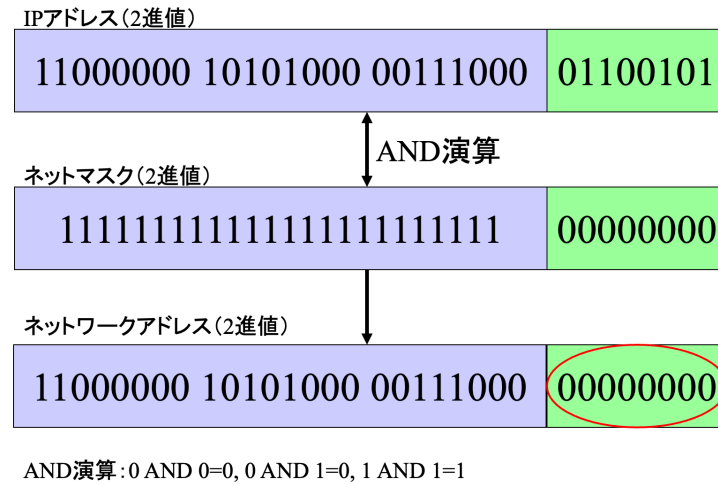


図 30: ネットマスクとネットワークアドレス

2.4.2 ネットマスクの表記

たとえば「192.168.56.101」という IP アドレスのうち、第 3 オクテットまでの 24 ビットがネットワーク部とした時、「192.168.56.101/24」と IP アドレスの後ろにネットマスクの長さを記述します。

ネットマスクだけを表記する場合、10 進数に変換して「255.255.255.0」と表記します。

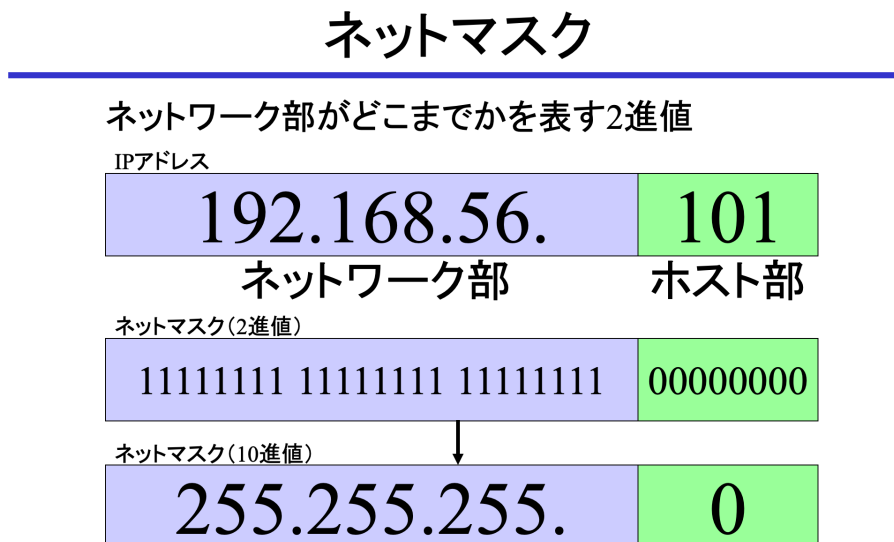


図 31: ネットマスク

2.4.3 ネットワークアドレスの表記

ネットワークアドレスを表記する場合、ネットマスクを利用してネットワーク部を計算し、アドレス部はすべて 0 として扱います。

「192.168.56.101/24」の場合、「192.168.56.0/24」がネットワークアドレスとなります。

2.4.4 ネットワークアドレスの使い方

ネットマスクは、IP アドレスからネットワークアドレスを計算するために使われます。ネットワークアドレスを計算すると、自分と通信相手が同じネットワークに接続されているノード同士かどうかわかります。

計算するためには、相手が同じネットワークに接続されているノードの IP アドレスと仮定します。その場合、相手も同じネットマスクの値であると仮定されます。このルールを利用して計算します。

たとえば、以下の IP アドレスが「192.168.56.101/24」と同じネットワークに接続されているノードのものかどうかを計算してみましょう。

192.168.56.102 相手も 24 ビットのネットマスクであれば、第 3 オクテットまでがネットワークアドレスです。つまり、「192.168.56.0/24」になります。これは「192.168.56.101/24」のネットワークアドレスと同じなので、同一のネットワークに接続されていると分かります。

同一のネットワークに接続されている場合、スイッチを経由して直接通信を行います。

192.168.57.1 ネットワークアドレスは「192.168.57.0/24」になります。これは「192.168.56.101/24」のネットワークアドレスと異なるので、別のネットワークに接続されたノードの IP アドレスと分かります。

異なるネットワークに接続されている場合、ルーターを経由して通信を行います。ルーターは相手のネットワークを探して通信を行ってくれます。これを「ルーティング」と呼びます。

2.5 ルーティングとは

前述したとおり、異なるネットワークに接続されているノードと IP で通信するには、ルーターによるルーティングが必要です。

2.5.1 インターネットはルーターの集合体

巨大なネットワークの集合体であるインターネットは、沢山のルーターで構成されています。そのため、最適な通信経路を決定する必要があります。ルーティング経路を決定するための様々なプロトコルが存在していますが、本教科書ではルーティングプロトコルの説明は省略します。興味がある人は別途調べてみてください。

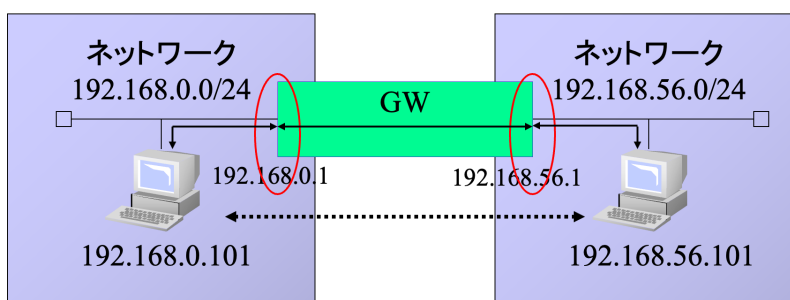
2.5.2 デフォルトゲートウェイ

最低限知っておくべきことは、異なるネットワークと通信する際にデフォルトでルーティングを依頼するデフォルトゲートウェイです。閉鎖されていて外部と通信を行えないネットワークでない限り、最低 1 つはルーターが必要です。このルーターの IP アドレスがデフォルトゲートウェイとなります。

Linux では、ゲートウェイアドレスとして設定します。

ゲートウェイアドレス

通信したい相手が同一ネットワークに無い時、ゲートウェイを経由して別のネットワークへ通信パケットを送る。



※IPアドレス/数値はネットマスクのビット数を表す

図 32: ゲートウェイアドレス

一般的なインターネット接続環境であれば、インターネットプロバイダーと接続しているルーターがデフォルトゲートウェイとして設定されます。社内ネットワークなど複数のネットワークで構成されている場合には、各ネットワークにルーターや L3 スイッチが接続されており、これらの機器の IP アドレスがデフォルトゲートウェイとなります。

2.5.3 デフォルトゲートウェイの確認

実習環境でデフォルトゲートウェイを確認してみます。確認するには `ip route` コマンドを実行します (`ip r` と省略可能)。

```
$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.56.0/24 dev enp0s8 proto kernel scope link src 192.168.56.101 metric 101
```

実習環境は、外部との接続を行う NAT 接続 (10.0.2.0/24) と、ホストと接続するホストオンリーネットワーク (192.168.56.0/24) の 2 つのネットワークと接続しています。そして、外部との接続のために 10.0.2.2 という IP アドレスがデフォルトゲートウェイとして設定されていることが分かります。

2.6 ブロードキャストアドレス

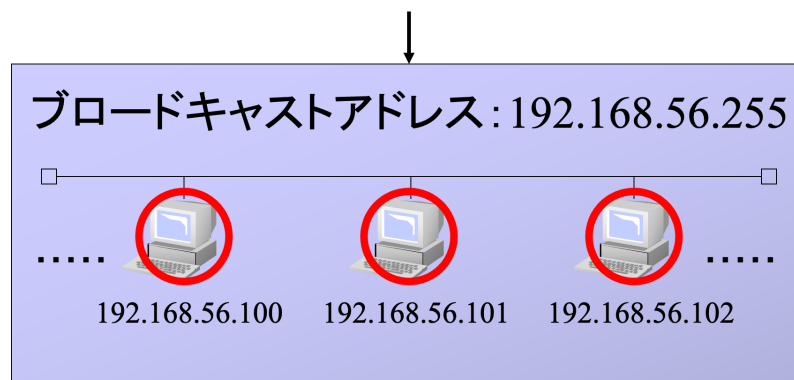
ブロードキャストアドレスは、同一のネットワークに接続されているすべてのノードに対して一斉に通信を行うための IP アドレスです。アドレス部のビットがすべて 1、オクテット区切り 10 進数表記であれば 255 になっている IP アドレスです。いくつかの通信プロトコルが、ネットワーク内に接続されているすべてのノードに対して呼びかけをするような場合に使われます。

「192.168.56.0/24」のネットワークであれば、「192.168.56.255/24」がブロードキャストアドレスになります。

ブロードキャストアドレスは自動的に計算され、必要とするソフトウェアが自動的に使用するもので、設定の必要はありません。

ブロードキャストアドレス

同一ネットワーク内に接続された全てのホストを指し示すアドレス



ホスト部のビットが全て1となるアドレス

図 33: ブロードキャストアドレス

2.7 アドレスクラス

約 43 億個ある IP アドレスを扱いやすく分類したのがアドレスクラスです。32 ビット表記の先頭の値を使って 5 つのアドレス範囲に分割し、それぞれをクラス A からクラス E としています。また、クラス A は 8 ビット、クラス B は 16 ビット、クラス C は 24 ビットのネットマスクを使用してネットワークを分割します。

なお、クラス D はマルチキャストアドレス用、クラス E は将来用に予約された範囲となります。

クラス	IP アドレスの範囲	ネットマスク	ネットワーク数	IP アドレス数
クラス A	0.0.0.0～ 127.255.255.255	8 ビット	126 個※	約 1600 万個
クラス B	128.0.0.0～ 191.255.255.255	16 ビット	16,384 個	65536 個
クラス C	192.0.0.0～ 223.255.255.255	24 ビット	約 209 万個	256 個
クラス D	224.0.0.0～ 239.255.255.255			
クラス E	240.0.0.0～ 255.255.255.255			

※ 0.0.0.0 はすべてを表す IP アドレス、127 で始まるネットワークはループバックで使用するため、2 つのネットワークを除外しています。

アドレスクラス

クラス A (IP アドレスは 16,777,216 個)

10.	0.0.1
-----	-------

クラス B (IP アドレスは 65,536 個)

172.20.	0.1
---------	-----

クラス C (IP アドレス 256 個)

192.168.0.	1
------------	---

図 34: アドレスクラス

2.7.1 CIDR

CIDR (Classless Inter-Domain Routing) は、8 ビット単位のネットマスクでネットワークを分割するのではなく、任意の長さのネットマスクでネットワークを分割する方法です。

IPv4 の IP アドレスの個数が限られているため、固定長のネットマスクで IP アドレスの割り当てを行うと、使用されない無駄な IP アドレスが発生してしまいます。そこで、任意の長さのネットマスクでネットワークを小さく分割するのが CIDR です。

たとえば、クラス C のアドレスの場合、ネットマスクを 1 ビット増やす毎に以下のようにネットワーク数と IP アドレス数が変化します。

ネットマスク	ネットワーク数	IP アドレス数
24	1 個	256 個
25	2 個	128 個
26	4 個	64 個
27	8 個	32 個
28	16 個	16 個
29	32 個	8 個

注意点として、各ネットワークにはネットワークアドレス、ブロードキャストアドレスが必要なので IP アドレスは 2 個減ります。また、ルーティングのためのデフォルトゲートウェイが必要になるので、合計で 3 個の IP アドレスが必要となるので、ノードに割り振れるのはネットワークあたりの IP アドレス数から 3 個を引いた数となります。

たとえば、29 ビットのネットマスクの場合、IP アドレス数は 8 個ですが、ノードに割り当てられる IP アドレスは 5 個になります。

インターネットに直接アクセスできるグローバル IP アドレスをプロバイダーから割り当ててもらふ際に、28 ビット (16 個) や 29 ビット (8 個) で割り当てを受ける場合があります。これらの割り当てでは、実際に使える IP アドレスの数は減ってしまうことになります。

2.8 グローバル IP アドレスとプライベート IP アドレス

初期のインターネットでは、すべてのノードはインターネットに直接接続されていましたが、インターネットに接続する組織が増えたため、IP アドレスが枯渇する恐れが出てきました。

そこでインターネットに直接接続できる IP アドレスをグローバル IP アドレス、逆に直接接続せず誰もが自由に利用できる IP アドレスをプライベート IP アドレスとして分けるようにしました。

プライベート IP アドレスとして定義されているのは、以下の IP アドレスです。

クラス	IP アドレスの範囲	ネットワーク数
クラス A	10.0.0.0~10.255.255.255	1 個
クラス B	172.16.0.0~172.31.255.255	16 個
クラス C	192.168.0.0~192.168.255.255	256 個

2.8.1 NAT

NAT (Network Address Translation) は、ルーティングの際に IP アドレスを変換する技術です。

プライベート IP アドレスを割り当てられたノードは、インターネット上のグローバル IP アドレスに直接アクセスできません。そこで NAT を使用して、プライベート IP アドレスをグローバル IP アドレスに変換して通信を行います。一般的なインターネットに接続するためのルーターは、NAT を行っています。

NAT については、後の章で Linux カーネルが備えているアドレス変換の仕組みである IP マスカレードで動作を確認します。

2.9 IP アドレスの変更

Linux における IP アドレスの設定は、ディストリビューションによって様々な仕組みで行われていますが、実習環境の AlmaLinux では **NetworkManager** を使って管理を行っています。

IP アドレスの変更は、設定ファイルを直接変更するのではなく、**NetworkManager** に対応したツールを使って行います。

2.10 GUI による IP アドレスの変更

GUI による IP アドレスの変更は、**NetworkManager** に対応した「設定」アプリから行えます。

画面右上の「ネットワーク、音量、電源の設定」をクリックし、「設定」を選択すると「設定」アプリケーションを呼び出せます。起動時から「ネットワーク」設定が選択されています。

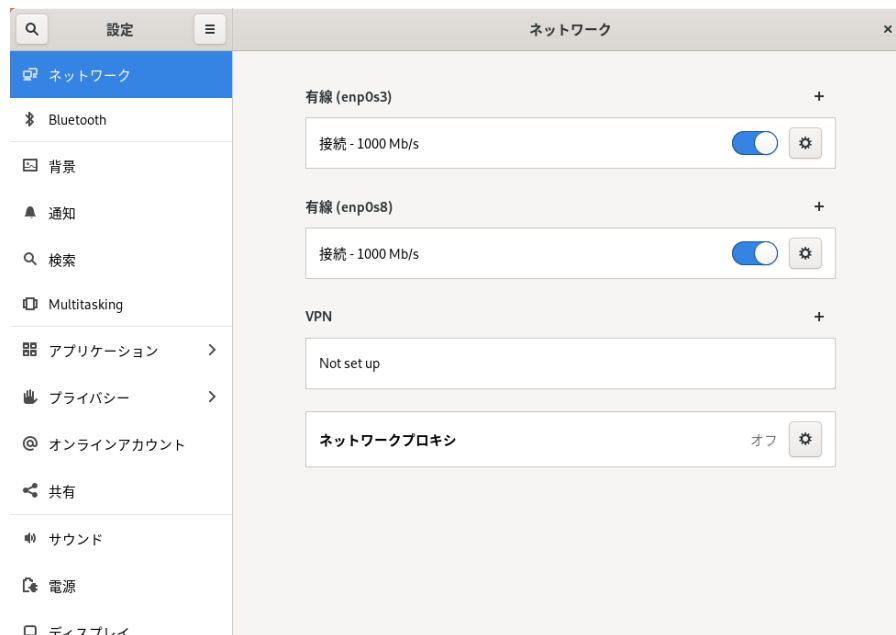


図 35: 「設定」アプリケーションの画面

仮想マシンにはネットワークアダプターが2つ接続されているので、設定はそれぞれに対応した2つのインターフェースが存在しています。スイッチで有効無効が選択できます。歯車ボタンをクリックすると、設定画面が表示されます。

「詳細」では、現在の設定状態の確認や、システム起動時の自動接続などが設定できます。

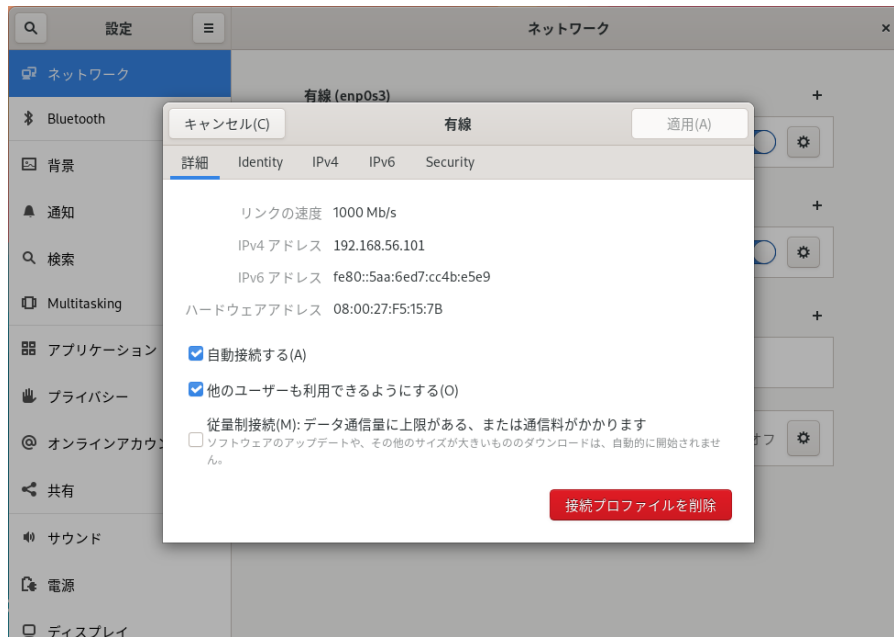


図 36: ネットワークアダプターの「詳細」の画面

IP アドレスを変更するには、「IPv4」を選択します。デフォルトでは「自動 (DHCP)」が選択されていますが、「手動」を選択すると IP アドレスやネットマスク (ビット数またはオクテット表記)、デフォルトゲートウェイ、参照する DNS サーバーなどを手動で設定できます。

ここでは IP アドレスを「192.168.56.50/24」に変更します。

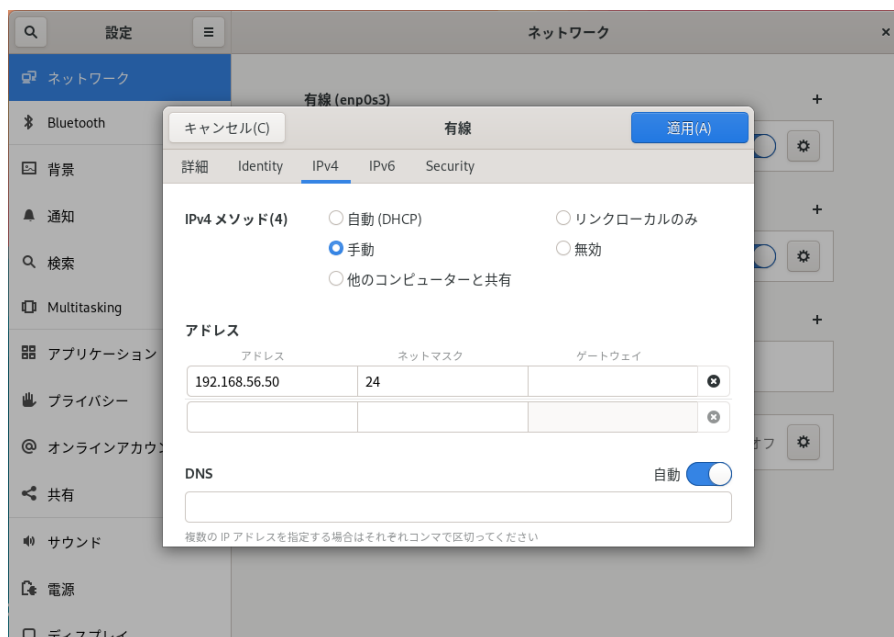


図 37: IP アドレスを変更

設定を行った後、「適用」をクリックすると設定が保存されます。

設定をネットワークインターフェースに適用するためには、歯車ボタンの左にあるスイッチをクリックして一旦ネットワークインターフェースを無効にし、再度スイッチをクリックして有効にする必要があります。

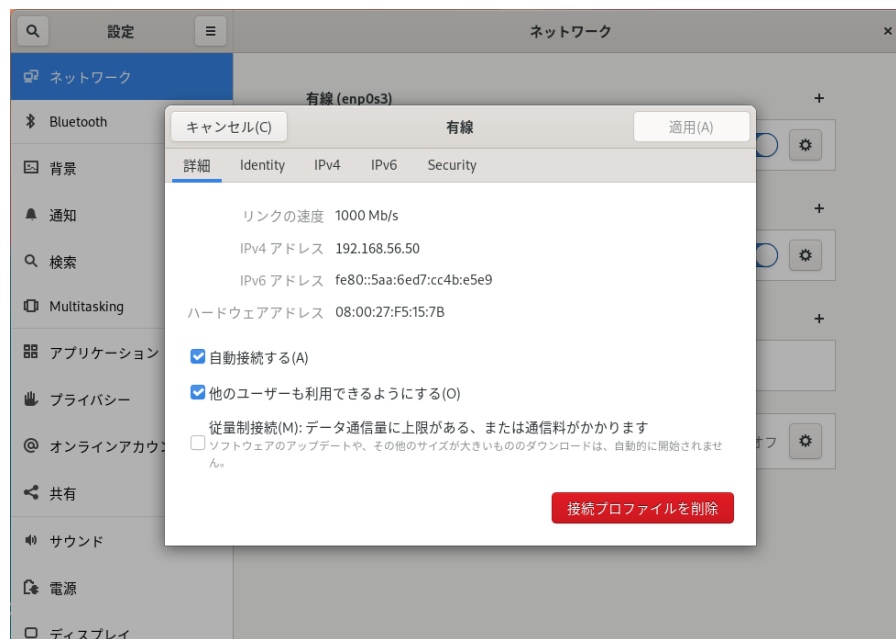


図 38: IP アドレス変更を確認

IP アドレスが指定した IP アドレスに変更されたのが確認できます。

2.11 TUI による IP アドレスの変更

GUI が使用できない SSH 接続などの場合、TUI（Text User Interface）に対応した `nmtui` を使用して対話形式で IP アドレスの変更が行えます。

コマンド実行可能な端末で `nmtui` コマンドを実行すると、「NetworkManager TUI」が起動します。

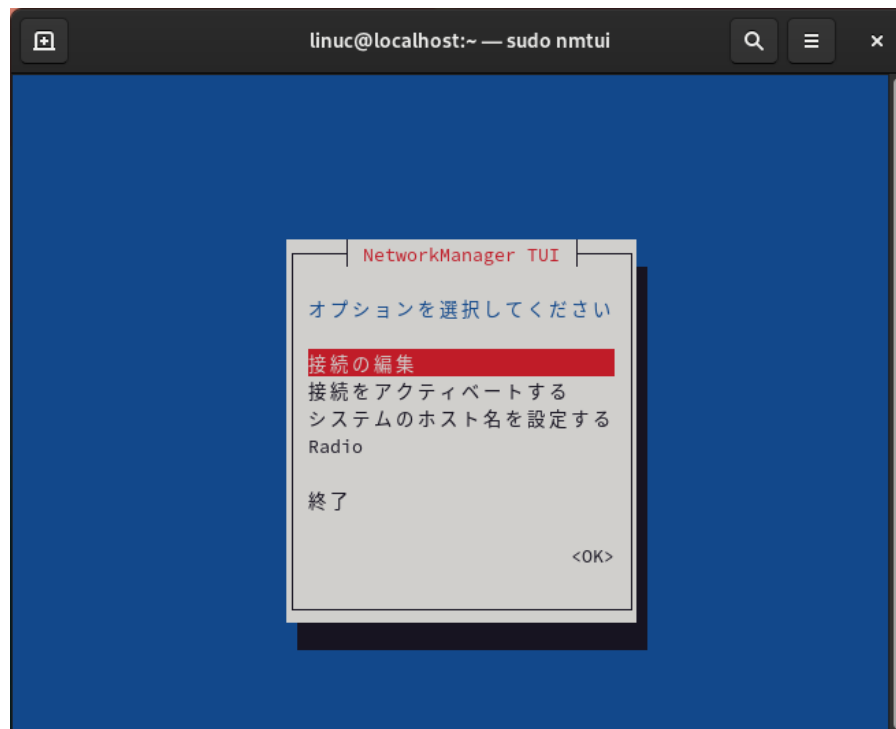


図 39: 「NetworkManager TUI」起動時の画面

TUI は以下のキーで操作できます。

- TAB キーで選択項目の移動
- カーソルキーで選択項目、あるいはリストなどの選択項目内での選択肢の移動
- Enter キーで選択
- ESC キーで戻る

TUI 操作は慣れが必要なため、まずは TAB キーで選択項目を移動させて、Enter キーを押してどのように画面が遷移するのかなどを確認してみてください。

2.11.1 操作例

たとえば IP アドレスを変更したい場合、以下のように操作します。リモートから SSH で接続している場合、SSH 接続で使用しているインターフェースの IP アドレスを変更しようとする、作業の途中で接続できなくなり、ローカルコンソールでの作業が必要になるので注意してください。

nmtui コマンドを実行して「NetworkManager TUI」を起動します。管理者権限が必要なので、**sudo** コマンドを付けて実行します。

```
$ sudo nmtui
```

1. 「接続の編集」を選択して、Enter キーを押します。
2. IP アドレスを変更したいインターフェースをカーソルキーで選択し、Enter キーを押します。
3. TAB キーを 4 回押して「IPv4 設定」の「自動」を選択し、Enter キーを押します。
4. カーソルキーで「手作業」を選択し、Enter キーを押します。
5. TAB キーを押して「表示する」を選択し、Enter キーを押します。
6. TAB キーを押してアドレスの「追加…」を選択し、Enter キーを押します。
7. 表示されたテキストボックスに IP アドレスを入力します。記述は「IP アドレス/ネットマスクのビット数」とします。
8. (必要に応じて) TAB キーを 3 回押してゲートウェイのテキストボックスを選択し、デフォルトゲートウェイの IP アドレスを入力します。
9. (必要に応じて) TAB キーを押して DNS サーバーの「追加…」を選択し、Enter キーを押します。
10. 表示されたテキストボックスに参照する DNS サーバーの IP アドレスを入力します。
11. TAB キーを 14 回押して「OK」を選択し、Enter キーを押します。
12. インターフェース選択画面に戻るので、ESC キーを押します。
13. 最初の画面に戻るので、カーソルキーで「接続をアクティベートする」を選択し、Enter キーを押します。
14. カーソルキーで設定を有効にしたいインターフェースを選択し、Enter キーを押します。この時点でそのインターフェースは無効になります。
15. 再度 Enter キーを押します。新しい設定が適用されてインターフェースが有効になります。
16. ESC キーを押します。
17. 最初の画面に戻るので、ESC キーを押して nmtui コマンドを終了します。

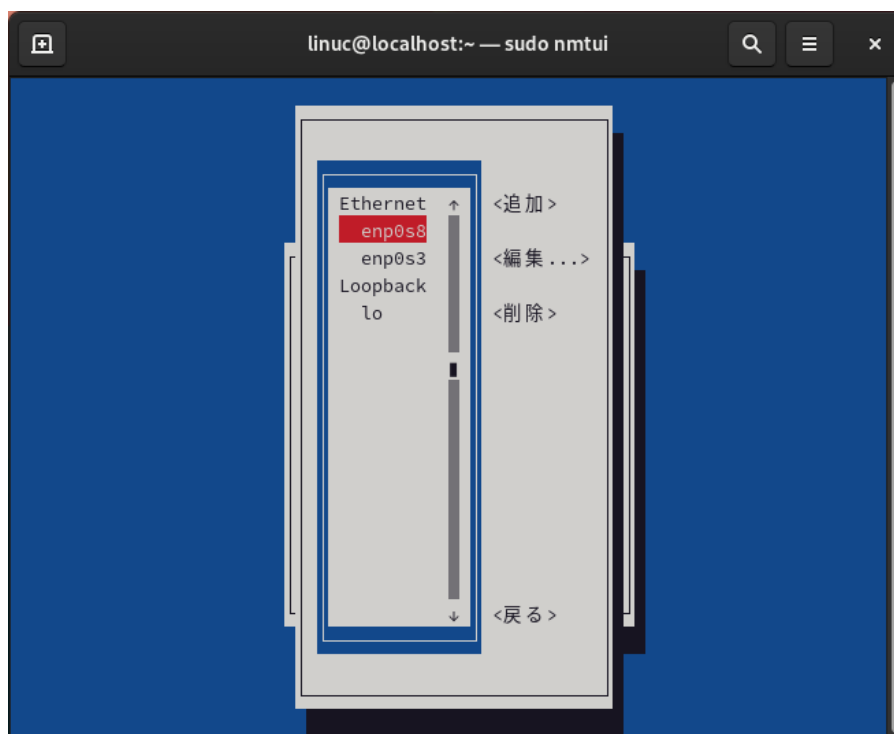


図 40: 「NetworkManager TUI」 インターフェース選択画面

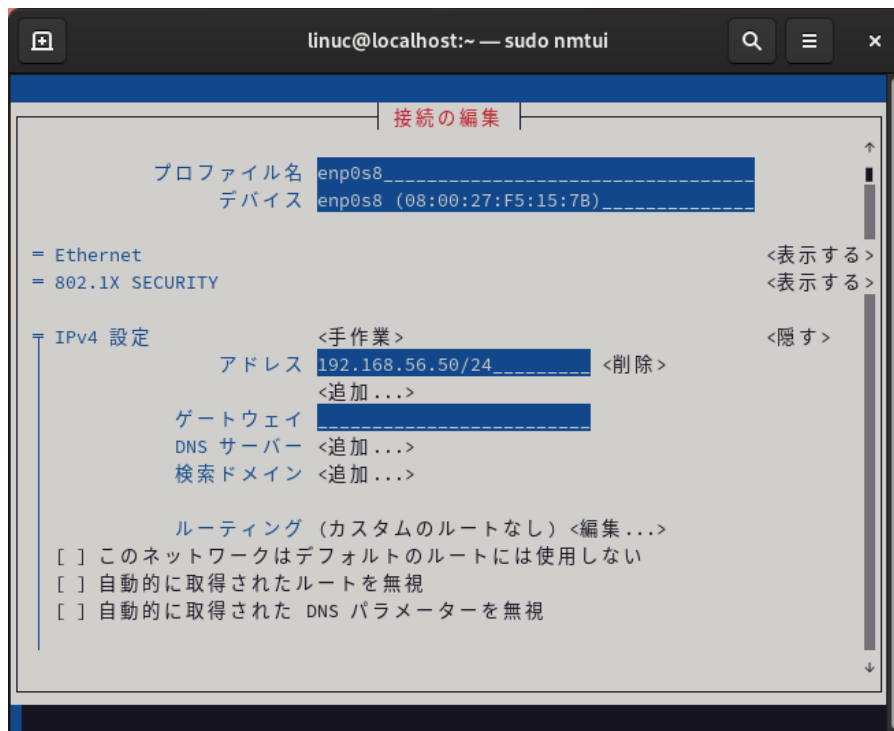


図 41: 「NetworkManager TUI」 IP アドレス設定画面

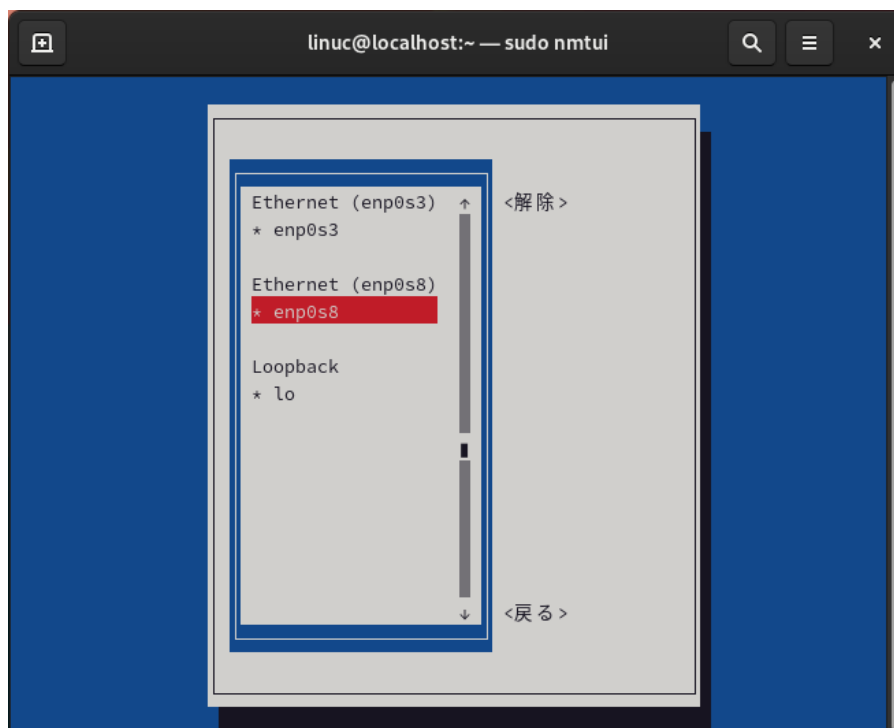


図 42: 「NetworkManager TUI」 インターフェースアクティベート画面

2.12 コマンド実行による IP アドレスの変更

nmcli コマンドを実行すると、コマンド実行で IP アドレスを変更できます。1 回のコマンド実行、あるいはシェルスクリプト内で実行します。

2.12.1 デバイスの確認

まず、nmcli コマンドを使って NetworkManager の設定状態を確認します。nmcli dev コマンドでデバイスの状態を確認します。

```
$ nmcli dev
DEVICE   TYPE        STATE          CONNECTION
enp0s3   ethernet    接続済み      enp0s3
enp0s8   ethernet    接続済み      enp0s8
lo       loopback    接続済み (外部) lo
```

enp0s8 デバイスには enp0s8 というコネクションが適用されているのがわかります。コネクションがネットワークの各種情報を保持しています。

2.12.2 デバイスの設定の確認

各デバイスがどのような設定になっているのかを確認します。nmcli dev show コマンドを使います。

```
$ nmcli dev show enp0s8
GENERAL.DEVICE:           enp0s8
GENERAL.TYPE:             ethernet
GENERAL.HWADDR:          08:00:27:F5:15:7B
GENERAL.MTU:              1500
GENERAL.STATE:            100 (接続済み)
GENERAL.CONNECTION:       enp0s8
GENERAL.CON-PATH:         /org/freedesktop/NetworkManager/ActiveC>
WIRED-PROPERTIES.CARRIER: オン
IP4.ADDRESS [1]:          192.168.56.50/24
IP4.GATEWAY:              --
IP4.ROUTE [1]:            dst = 192.168.56.0/24, nh = 0.0.0.0, mt>
IP6.ADDRESS [1]:          fe80::5aa:6ed7:cc4b:e5e9/64
IP6.GATEWAY:              --
IP6.ROUTE [1]:            dst = fe80::/64, nh = ::, mt = 1024
lines 1-14/14 (END)
```

設定されている IP アドレス、ルーティングの情報などが確認できます。ページングで表示されているので、q キーを入力して終了します。

2.12.3 コネクションの確認

どのようなコネクションがあるのかを確認します。nmcli con コマンドを使います。

```
$ nmcli con
NAME      UUID                                  TYPE      DEVICE
enp0s3    ae880da5-eb22-3b2f-8a72-3924ffd95eaf ethernet  enp0s3
enp0s8    f1870675-0703-4d88-8e9b-cbfcf9c22038 ethernet  enp0s8
lo        cd417d66-c5c7-4107-bc9b-d47195390c47 loopback   lo
```

コネクションは UUID で管理されているのがわかります。

2.12.4 コネクションの詳細を確認

コネクションがどのような設定なのかを確認します。nmcli con show コマンドを使います。

```
$ nmcli con show enp0s8
connection.id:                enp0s8
connection.uuid:              f1870675-0703-4d88-8e9b-cbfcf9c22038
connection.stable-id:        --
connection.type:              802-3-ethernet
connection.interface-name:    enp0s8
connection.autoconnect:      はい
(略)
```

コネクションは非常に多くの設定項目を保持しているのがわかります。ページングで表示されているので、q キーを入力して終了します。

2.12.5 IP アドレスの変更

IP アドレスを DHCP ではなく指定した固定アドレスを割り当てるように変更します。コネクションの変更なので `nmcli connection modify` コマンドを使います。

リモートログインしている場合、IP アドレスが変更になると接続は切断されます。正しく IP アドレスが設定されなかった場合、リモートログインが行えなくなりますので、ローカルコンソールで作業できる状態で変更を行ってください。

まず、`ipv4.method` というパラメータを `manual` に変更し、`ipv4.addresses` に IP アドレスとネットマスクを指定します。

```
$ sudo nmcli connection modify enp0s8 ipv4.method manual ipv4.addresses
192.168.56.51/24
```

コネクションが保持している設定は修正されましたが、デバイスにはまだ修正は適用されていません。`nmcli con up` コマンドを実行してコネクションを有効にすることで新しい設定が適用され、IP アドレスが変更されます。

```
$ sudo nmcli con up enp0s8
```

リモートログインしている場合、この段階で IP アドレスが変更されるので接続は切断されます。再度、新しい IP アドレスに宛ててリモートログインしてみてください。

IP アドレスが変更されているかを確認します。

```
$ ip a
(略)
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.51/24 brd 192.168.56.255 scope global noprefixroute enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::5aa:6ed7:cc4b:e5e9/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

IP アドレスが変更されているのが確認できます。

2.13 DHCP とは

DHCP (Dynamic Host Configuration Protocol) は、IP アドレスなどを自動的に設定するためのプロトコルです。ネットワーク機器などに DHCP サーバー機能が組み込まれていて、DHCP クライアントがネットワークに接続するだけで IP アドレスやデフォルトゲートウェイなどが設定され、インターネットに接続できるようになります。

実習環境では VirtualBox に DHCP サーバーの機能が備えられているので、設定を確認してみましょう。

2.13.1 VirtualBox の DHCP サーバー機能

VirtualBox の DHCP サーバー機能は、NAT およびホストオンリーネットワークで利用できます。NAT は設定値が固定されているので、ホストオンリーネットワークの DHCP サーバー機能で設定を見てみましょう。

ホストオンリーネットワークの DHCP サーバーの設定は、1 章で確認した VirtualBox のネットワーク設定ツールで確認、変更が行えます。

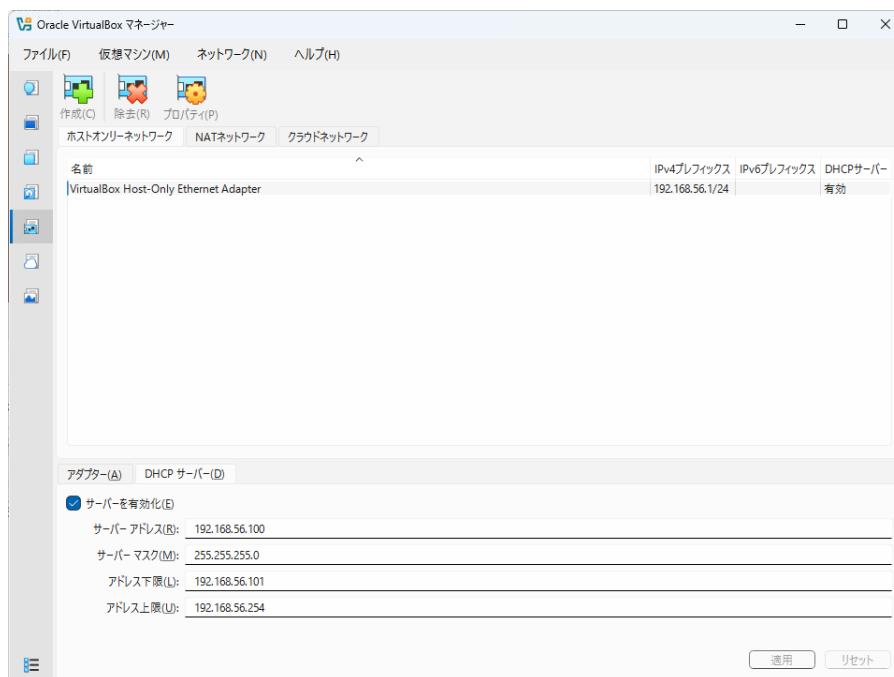


図 43: DHCP サーバーの設定

デフォルトの設定で、DHCP サーバーは DHCP クライアントに対して 192.168.56.101 から 192.168.56.254 までの IP アドレスを割り当てるのがわかります。

2.13.2 DHCP で IP アドレスを取得する

実際に DHCP サーバーが IP アドレスを割り当てる様子を見てみましょう。

現在、ホストオンリーネットワークに接続されているネットワークアダプター 2 は、ゲスト OS では `enp0s8` という名前のインターフェース、コネクションとして管理されており、前の実習で IP アドレスを手動で設定しました。この設定を DHCP で自動的に IP アドレスを設定するように変更します。GUI や TUI では自動設定に変更して、設定を適用します。

2.13.3 CLI で DHCP で自動的に IP アドレスを取得する設定を行う

CLI の `nmcli` コマンドを実行して、IP アドレスを手動設定から DHCP による自動設定に変更します。

`nmcli connection modify` コマンドを実行して、`ipv4.method` パラメータを `auto` に設定し、手動設定した IP アドレスを `-ipv4.addresses` パラメータ（マイナス付）で削除します。

```
$ sudo nmcli connection modify enp0s8 ipv4.method auto -ipv4.addresses 192.168.56.51/24
```

`nmcli con up` コマンドでコネクションを有効にします。

```
$ sudo nmcli con up enp0s8
```

IP アドレスを確認して、DHCP サーバーから IP アドレスが割り当てられていることを確認します。

```
$ ip a
(略)
```

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic
        noprefixroute enp0s8
        valid_lft 597sec preferred_lft 597sec
    inet6 fe80::5aa:6ed7:cc4b:e5e9/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

DHCP サーバーの下限に設定された 192.168.56.101 が IP アドレスとして自動設定されていることが確認できます。

3 TCP/UDP/ICMP

TCP や UDP、ICMP は IP 通信の上で使用されるプロトコルです。インターネットの様々なサービスは、これらのプロトコルを使って動作しています。

プロトコルの特徴が異なるので、それぞれについて解説します。

3.1 TCP とは

TCP (Transmission Control Protocol) は、様々な通信制御を行うプロトコルです。

IP と組み合わせてインターネットの様々な場面で使用されるため、TCP/IP と呼ばれることもあります。

TCP の特徴は、3way ハンドシェイクによるセッションの確立と、シーケンス番号を使った通信の保証にあります。

3.1.1 3way ハンドシェイク

TCP は、通信を開始する際に 3way ハンドシェイクと呼ばれる方法でセッションを確立します。

1. 送信側は、通信を開始するために SYN (Synchronize・同期) フラグを設定したパケットを受信側に送信します。
2. 受信側は、送られてきた SYN フラグに対して ACK (Acknowledgement・了承) フラグと、自分の側からの SYN フラグを設定したパケット、すなわち SYN + ACK フラグのパケットを送信側に送信します。
3. 送信側は、受信側からの SYN フラグに応答する ACK フラグを設定したパケットを受信側に送信します。

TCPの接続方法

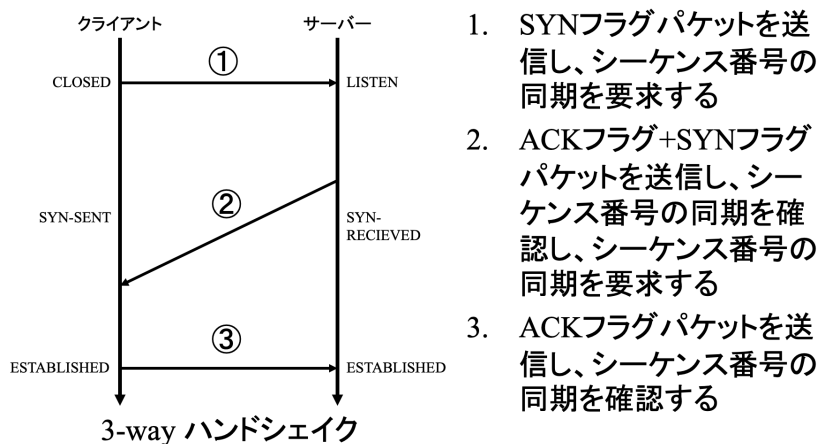


図 44: 3way ハンドシェイク

この 3 つのやり取りで 3way ハンドシェイクは完了し、通信のためのセッションが確立します。

3.1.2 シーケンス番号と通信の保証

3way ハンドシェイクのやり取りでは、TCP の特徴である通信の保証を行うため、お互いのシーケンス番号を交換しています。

送信側は、データを送信する際にパケットに自分のシーケンス番号を設定し、その値をパケットにつけて送信します。

受信側は、受け取ったパケットのシーケンス番号にデータサイズを足した値を確認応答番号として送信側に返信します。この確認応答番号が次に送受信すべきパケットのシーケンス番号となります。

もしパケットが途中の通信経路で失われてしまうと、送信したはずのパケットの確認応答番号が戻って来ない事になります。一定時間待っても確認応答番号が戻らない場合、再度パケットを送信します。

シーケンス番号と確認応答番号

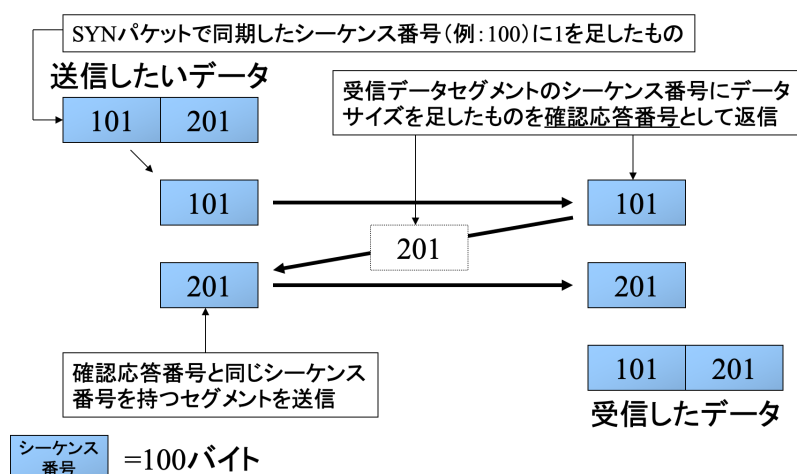


図 45: シーケンス番号

このようにして、TCP は通信パケットが失われないようにすることを保証しています。

3.2 UDP とは

UDP (User Datagram Protocol) は、TCP 同様に IP ネットワークでデータ通信を行うためのプロトコルです。TCP と異なり、3way ハンドシェイクやシーケンス番号による通信の保証を行わない、軽量な通信プロトコルです。

UDP が使われている代表的な通信プロトコルとして、DNS に対する名前解決の問い合わせがあります。

3.2.1 UDP はセッションレスプロトコル

TCP が通信のためのセッションを確立してから通信を行うのに対して、UDP はセッションの無いセッションレスのプロトコルです。そのため、通信パケットが通信経路の途中で失われても分からないため、再送要求は行われません。

UDP を利用するアプリケーションは、自分の送信した UDP パケットに対する返信が無い場合には、一定時間待った後にタイムアウトさせて、必要に応じて再度 UDP パケットを送信するなどの処理が必要です。

3.3 ポート番号とは

ポート番号は、TCP や UDP が使う送信側や受信側を識別するための値です。IP アドレスがネットワークに接続されたノードに紐付けられるのに対して、ポート番号は通信を行うプロセスに紐付けられます。

たとえば、HTTPS で Web サーバーに対して通信を行う場合、ポート番号 443 番に対して通信を行います。Web サーバー側では、Web サーバーのプロセスがポート番号 443 番に対する通信を待ち受け (LISTEN)、リクエストが送られてくると処理を行い、返信します。

3.3.1 Well-Known ポート

ポート番号のうち、主要なプロトコルに対応付けられているポート番号 0 番から 1023 番までを Well-Known ポートと呼びます。

Linux では、Well-Known ポートを使用するには root 権限が必要です。

ポート番号の役割

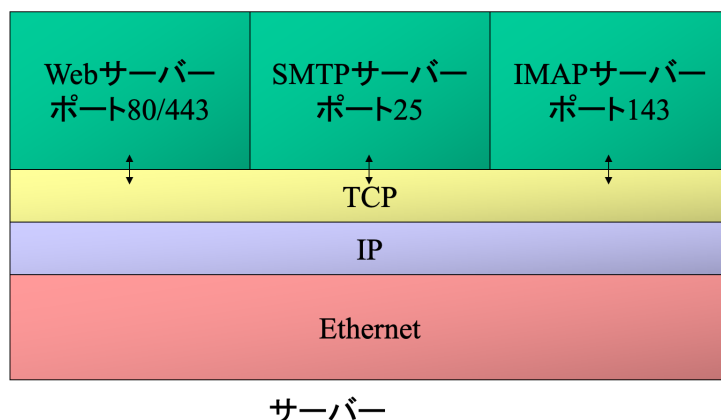


図 46: ポート番号の役割

3.4 Registered ポート

ポート番号のうち、1024 番から 49151 番までを Registered ポートと呼びます。インターネットに関わる様々な事柄を管理する組織である IANA (Internet Assigned Numbers Authority) に申請を行い、登録される (Registered) ことで他のプロトコルとポート番号が重複しないようにしています。

3.4.1 ハイポート

プロセスが自由に使えるポート番号です。IANA が管理している Well-Known ポートと Registered ポートよりも上の番号なのでハイ (High) ポートと呼ばれます。

ハイポートは、通信を行うクライアントが使用してサーバーからの返信を受け取るために使用したり、一部は特定のソフトウェアが IANA に未登録で使用しています。前者は、使用していたプロセスが終了すると解放されるため、エフェメラルポート (一時的なポート) と呼ばれます。

IANA の定義では 49152 番から 65535 番ですが、Linux では異なっています。実際にエフェメラルポートとして使われるポート番号は `/proc/sys/net/ipv4/ip_local_port_range` を参照すると確認できます。

```
$ cat /proc/sys/net/ipv4/ip_local_port_range
32768    60999
```

エフェメラルポートとして、32768 番から 60999 番が使用されることが分かります。

3.4.2 `/etc/services` を参照する

ポート番号は、`/etc/services` に名称との対応が記述されているので確認してみましょう。

```
$ tail /etc/services
aigairserver    21221/tcp      # Services for Air Server
ka-kdp          31016/udp      # Kollektive Agent Kollektive Delivery
ka-sddp         31016/tcp      # Kollektive Agent Secure Distributed
Delivery
edi_service     34567/udp      # dhanalakshmi.org EDI Service
axio-disc       35100/tcp      # Axiomatic discovery protocol
axio-disc       35100/udp      # Axiomatic discovery protocol
pmwebapi        44323/tcp      # Performance Co-Pilot client HTTP API
```

cloudcheck-ping	45514/udp	# ASSIA CloudCheck WiFi Management
keepalive		
cloudcheck	45514/tcp	# ASSIA CloudCheck WiFi Management
System		
spremotetablet	46998/tcp	# Capture handwritten signatures

Linux のハイポートは 32768 番からなので、34567 番の `edi_service` 以降は重複しています。もちろん、先にプロセスが起動してそのポート番号を使用し始めた場合、同じポート番号は使用されません。

3.5 ICMP

ICMP (Internet Control Message Protocol) は、ネットワークの通信を制御するためのプロトコルです。

ICMP を使った通信の代表例として、`ping` コマンドで IP 通信が正常に行えるかを確認する方法があります。`ping` コマンドは、内部的に ICMP の Echo と、応答に Echo Reply を使っています。

3.5.1 ICMP は TCP/UDP とは別のプロトコル

ICMP は、TCP や UDP 同様に IP 上で使用されるプロトコルですが、TCP や UDP とは異なるプロトコルです。

たとえば、`ping` コマンドには応答するが、Web ブラウザでのアクセスが通らないといった場合、パケットフィルタリングなどで TCP のパケットをフィルタされている可能性があります。逆に、ICMP を通さない設定にしているルーターやファイアーウォールなどもあります。

`ping` コマンドに応答が無い場合でも IP 通信は正常にできていることもあるので、`ping` コマンドの実行結果はあくまで 1 つの手掛かりであって、その他の通信の結果もふまえて通信の状態を判断する必要があります。

3.6 TCP の通信を Wireshark で見てみよう

Wireshark を使って、TCP 通信の特徴である 3way ハンドシェイクのパケットのやり取りを見てみましょう。

3.6.1 パケットキャプチャを開始する

Wireshark の起動方法は 1 章で解説しました。起動後、NAT 接続でインターネットと通信ができる `enp0s3` を選択します。

画面左上の「パケットキャプチャを開始します」ボタンをクリックします。

3.6.2 Web サイトにアクセスする

Web ブラウザを起動し、インターネット上の Web サーバーにアクセスします。

実行例では「<https://linuc.org/>」にアクセスしています。

3.6.3 パケットキャプチャを停止する

Web サイトにアクセスが完了したら、パケットキャプチャを停止します。

3.6.4 条件でフィルタリングを行う

TCP のやり取りを確認するために、条件を指定してフィルタリングを行います。

今回は HTTPS 通信をやり取りするパケットを絞り込みたいので、宛先ポートあるいは発信ポートのいずれかが TCP の 443 番となっているパケットを指定します。いずれか (or) 条件を指定するには 2 つの条件を「||」でつなげて指定します。

```
tcp.dstport==443 || tcp.srcport==443
```

あるいは、送受信双方のポート番号が 443 番であることを指定しても同じ結果が得られます。

```
tcp.port==443
```

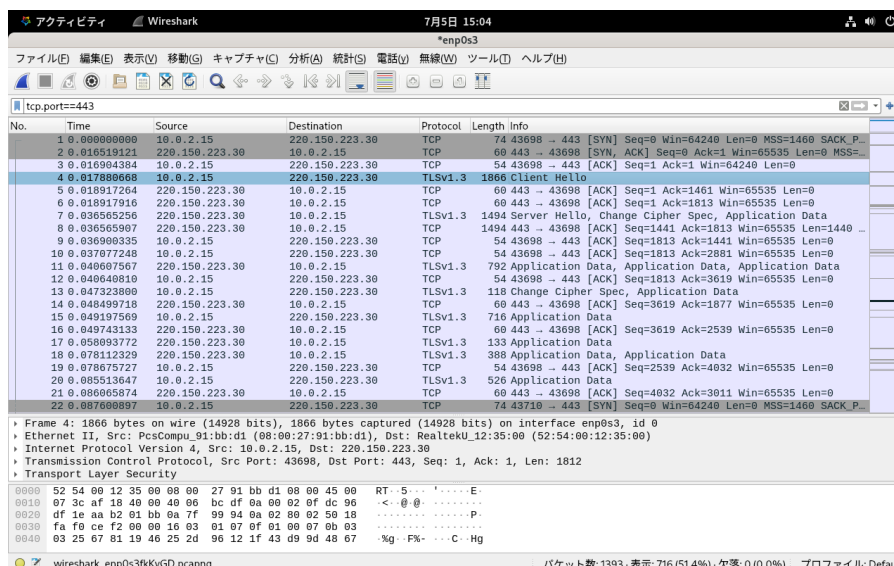


図 47: Wireshark パケットキャプチャ画面

3.6.5 3way ハンドシェイクを確認する

表示されたパケットの最初の 3 つが通信を開始する 3way ハンドシェイクのパケットです。

1. SYN フラグが設定されたパケットが送信される
2. SYN フラグおよび ACK フラグが設定されたパケットを受信する
3. ACK フラグが設定されたパケットが送信される

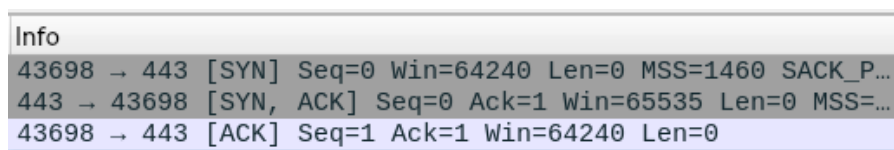


図 48: 3way ハンドシェイク

3.6.6 その後のやり取りを観察する

3way ハンドシェイクは、通信を行うノード間でシーケンス番号を交換するのを目的としています。

3way ハンドシェイク後の通信を観察すると、シーケンス番号 (Seq) と確認応答番号 (Ack) が送信され、それに対して応答番号をシーケンス番号に置き換えたパケットが返信される通信が繰り返されていることが分かります。

例では、4 番のパケットはシーケンス番号 1 番、データサイズは 1812 バイトのパケットです。これに対して、6 番のパケットは確認応答番号 1813 番 (シーケンス番号+データサイズの値) を返しています。

このようにして、パケットが到達していることを知らせることで、TCP 通信のパケットが失われないようにすることを保証しています。

3.7 UDP の通信を Wireshark で見てみよう

Wireshark を使って、UDP 通信の特徴であるセッションレスな通信を見てみましょう。

先ほどの Web サイトへのアクセスの際に行われた名前解決のパケットを表示します。

3.7.1 条件でフィルタリングを行う

DNS による名前解決は、暗号化されていないので問い合わせ内容を直接条件として設定できます。linuc.org の名前解決に関するパケットを絞り込みます。

```
dns.qry.name==linuc.org
```

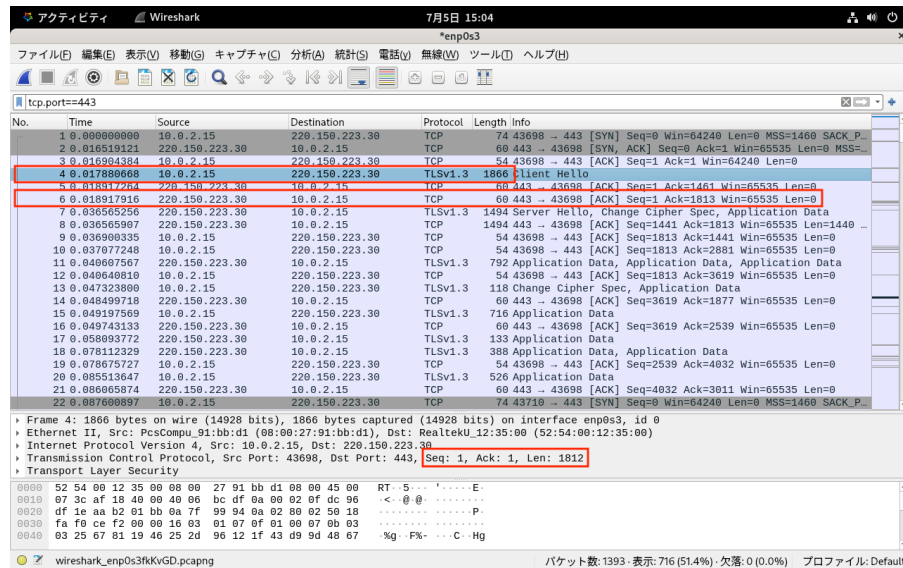


図 49: パケットのやり取りとシーケンス番号

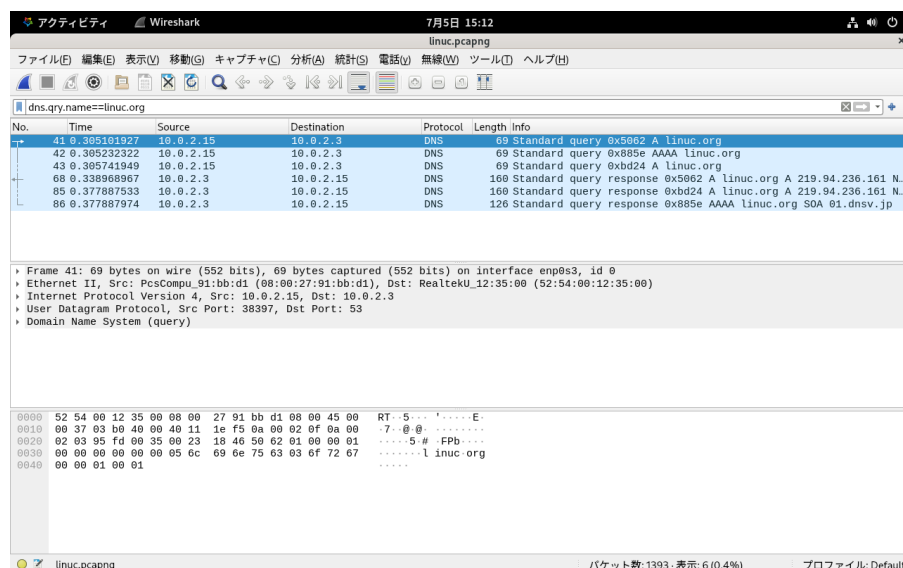


図 50: パケットのやり取りとシーケンス番号

3.7.2 UDP のやり取りを確認する

TCP の場合と異なり、UDP ではすぐに名前解決のパケットを送信しています。それに対して、DNS サーバーも名前解決の結果をすぐに返してきています。

UDP はセッションを確立せずにやり取りを行うセッションレスな通信プロトコルであることが分かります。

3.8 ICMP の通信を Wireshark で見てみよう

Wireshark を使って、ICMP 通信の代表例である ping コマンドの通信を見てみましょう。

3.8.1 パケットキャプチャを開始する

Web サイトへのアクセスの例と同様に、Wireshark でパケットキャプチャを開始します。

キャプチャしたパケットを保存するかを確認するダイアログが表示されますが、「保存せずに続ける」ボタンをクリックします。

3.8.2 ping コマンドを実行する

ターミナルで ping コマンドを実行して、ICMP による通信を行います。

```
$ ping linuc.org
```

応答が得られたら、Ctrl+C キーを押して停止します。

3.8.3 パケットキャプチャを停止する

パケットキャプチャを停止します。

3.8.4 条件でフィルタリングを行う

ICMP のやり取りを確認するために、条件を指定してフィルタリングを行います。単純に ICMP のパケットだけを絞り込みます。

```
icmp
```

3.8.5 ICMP のやり取りを観察する

パケットは、自分から送られた Echo request と、それに対する Echo reply が ping コマンドが行った回数分だけ表示されます。

Wireshark のウィンドウ真ん中のペインで、「Internet Control Message Protocol」をクリックすると、ICMP パケットの中身を見ることができます。

Echo request は、ICMP の Type が 8、Echo reply は Type が 0 になっており、非常にシンプルな通信を行っていることが分かります。

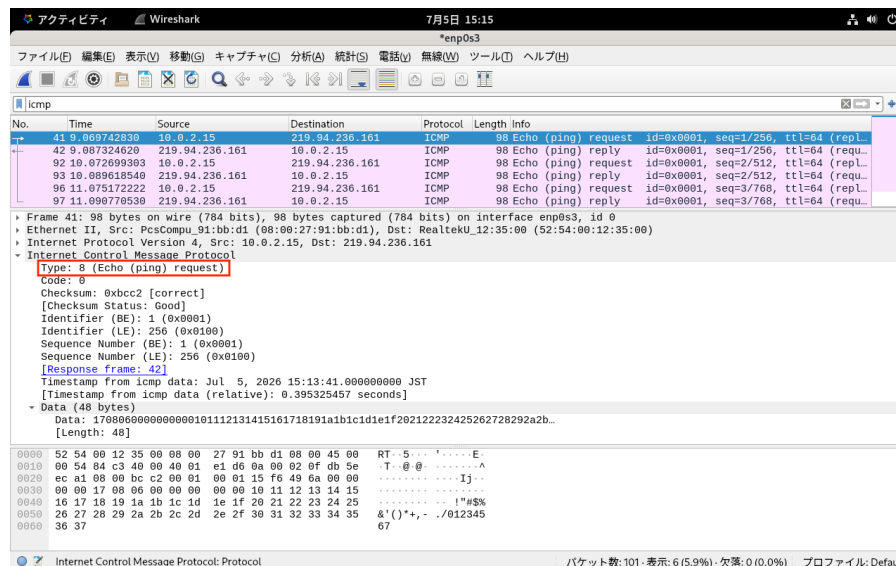


図 51: ICMP パケットの中身を参照

4 パケットフィルタリング

パケットフィルタリングは、IP アドレスやポート番号などを元にパケット通信の許可や拒否を行う仕組みです。

ネットワークのセキュリティを高める目的で、ファイアーウォールとして機能させることが多いので、しっかりとその原理を理解しておきましょう。

4.1 Linux のパケットフィルタリング機能 nftables

Linux のパケットフィルタリングは、nftables として実装されています。nftables に関連するカーネルモジュールを読み込ませることで利用できます。

nftables の設定を行う基本的なツールとして nft コマンドが用意されていますが、実習環境である AlmaLinux では firewalld がフロントエンドとして採用されています。

4.1.1 カーネルモジュールの確認

nftables 関連のカーネルモジュールが読み込まれているかどうか確認してみましょう。

```
$ lsmod | grep nf
nft_fib_inet          12288    1
nft_fib_ipv4          12288    1 nft_fib_inet
nft_fib_ipv6          12288    1 nft_fib_inet
nft_fib               12288    3 nft_fib_ipv6,nft_fib_ipv4,nft_fib_inet
nft_reject_inet       12288    6
nf_reject_ipv4        16384    1 nft_reject_inet
nf_reject_ipv6        20480    1 nft_reject_inet
nft_reject            12288    1 nft_reject_inet
nft_ct                24576    7
nft_chain_nat         12288    3
nf_nat                65536    1 nft_chain_nat
nf_conntrack          225280   2 nf_nat,nft_ct
nf_defrag_ipv6        24576    1 nf_conntrack
nf_defrag_ipv4        12288    1 nf_conntrack
nf_tables             356352   235
  nft_ct,nft_reject_inet,nft_fib_ipv6,nft_fib_ipv4,nft_chain_nat,nft_reject,nft_fib,nft_
nfnetlink             20480    2 nf_tables
libcrc32c             12288    4 nf_conntrack,nf_nat,nf_tables,xfs
```

nf_tables が nftables の本体となるカーネルモジュールです。その他に様々なカーネルモジュールが読み込まれているのがわかります。

4.2 firewalld

nftables を操作するためのフロントエンドが firewalld です。firewalld は、デーモンとして動作するサービスです。

4.2.1 firewalld サービスの確認

firewalld が動作していることを確認しましょう。systemctl コマンドで確認できます。

```
$ sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset>
   Active: active (running) since Mon 2026-07-06 14:15:01 JST; 6min ago
     Docs: man:firewalld(1)
   Main PID: 812 (firewalld)
    Tasks: 2 (limit: 10514)
   Memory: 44.3M (peak: 46.3M)
      CPU: 1.158s
   CGroup: /system.slice/firewalld.service
           └─812 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

7月 06 14:14:57 localhost systemd[1]: Starting firewalld - dynamic firewall da>
7月 06 14:15:01 localhost systemd[1]: Started firewalld - dynamic firewall dae>
```

表示から、firewalld の実体は Python で書かれたプログラムが動作していることが分かります。

4.2.2 firewall-cmd

デーモンとして動作している firewalld に対して、様々な指示を出すのが firewall-cmd コマンドです。

パケットフィルタリングの設定を変更して、外部からのパケットを受け入れる設定方法は『Linux サーバー構築標準教科書』で解説していますが、設定の確認について補足として解説しておきます。

4.3 パケットフィルタリングの設定を確認する

現在の firewalld によるパケットフィルタリングの設定を確認するには、firewall-cmd - list-all コマンドを実行します。

```
$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

services で受け入れるパケットをサービス名で定義しています。たとえば、ssh が定義されているので、外部からの SSH 接続が受け入れられています。

4.3.1 ゾーン

firewalld はゾーンという考え方でパケットフィルタリングを管理しています。特別な構成と設定が必要でない限り、表示されている **public** ゾーンに対して設定を行えば、外部とのパケットのやり取りに対してフィルタが設定されます。

firewall-cmd - get-zones コマンドで、定義されているゾーンの一覧が確認できます。

```
$ firewall-cmd --get-zones
block dmz drop external home internal nm-shared public trusted work
```

public の他、様々なゾーンが定義されているのが分かります。

firewall-cmd - get-default-zone コマンドで、デフォルトに設定されているゾーンが確認できます。

```
$ sudo firewall-cmd --get-default-zone
public
```

public がデフォルトのゾーンとして定義されています。外部とのパケットのやり取りは、デフォルトで public ゾーンに入ってきて、定義されているルールによって処理されることになります。

4.3.2 ターゲット

ターゲットは、そのゾーンに入ったパケットが設定されているルールにマッチしなかった場合、どのような処理を行うのかを定義しています。

ターゲットは以下の種類があります。

- **ACCEPT** ルールで拒否されたパケット以外をすべて受け入れます。
- **REJECT** ルールで許可されたパケット以外をすべて拒否します。拒否したことを送信元に通知します。
- **DROP** ルールで許可されたパケット以外をすべて破棄します。破棄したことを送信元に通知しません。
- **default REJECT** 同様の振るまいをしますが、いくつかの違いがあります。ping コマンドなどで使われる ICMP は通します。

確認した設定では public ゾーンの target は default は設定されているので、ICMP 以外のパケットは services などのルールを設定して許可しない限り拒否されます。

```
$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client ssh
(略)
```

4.4 ターゲットの変更

ターゲットを変更してみましょう。現在の default から REJECT、DROP に変更して動作を確認してみます。

4.4.1 ターゲット REJECT に設定

ターゲットを REJECT に設定してみます。

```
$ sudo firewall-cmd --zone=public --set-target=REJECT --permanent
```

設定を有効にするには、firewall-cmd - reload コマンドを実行します。

```
$ sudo firewall-cmd --reload
success
```

設定を確認します。

```
$ sudo firewall-cmd --zone=public --get-target --permanent
REJECT
```

動作を確認するため、ホスト OS から ping コマンドを実行します。

```
C:\Users\LinuC>ping 192.168.56.101

192.168.56.101 に ping を送信しています 32 バイトのデータ:
192.168.56.101 からの応答: 宛先ネットワークに到達できません。
192.168.56.101 からの応答: 宛先ネットワークに到達できません。
192.168.56.101 からの応答: 宛先ネットワークに到達できません。

192.168.56.101 の ping 統計:
    パケット数: 送信 = 3、受信 = 3、損失 = 0 (0% の損失)、
Ctrl+C
^C
```

ターゲットを REJECT にすると、nftables はパケットを拒否したことを通知します。そのため、ping コマンドは拒否されたことを表示します。パケットの拒否の通知も ICMP によって行われます。

4.4.2 ターゲット DROP に設定

ターゲットを DROP に設定してみます。

```
$ sudo firewall-cmd --zone=public --set-target=DROP --permanent
```

設定を有効にするには、firewall-cmd - reload コマンドを実行します。

```
$ sudo firewall-cmd --reload
success
```

設定を確認します。

```
$ sudo firewall-cmd --zone=public --get-target --permanent
DROP
```

動作を確認するため、ホスト OS から ping コマンドを実行します。

```
C:\Users\LinuC>ping 192.168.56.101

192.168.56.101 に ping を送信しています 32 バイトのデータ:
要求がタイムアウトしました。
要求がタイムアウトしました。
要求がタイムアウトしました。
要求がタイムアウトしました。

192.168.56.101 の ping 統計:
    パケット数: 送信 = 4、受信 = 0、損失 = 4 (100% の損失)、
```

ターゲットを DROP にすると、nftables はパケットを拒否したことを通知しません。そのため、ping コマンドは Echo request に対する Echo reply が返って来ず、タイムアウトしたことを表示します。

4.5 ICMP の扱い

ターゲットが default の時、ICMP は受け入れますが、細かい扱いは別途 icmp-blocks と icmp-block-inversion で設定されています。inversion は「反転する」という意味で、icmp-blocks で設定した種類の ICMP を拒否するのか、受け入れるのかを icmp-block-inversion で設定します。

icmp-block-inversion を no に設定すると、icmp-blocks で設定した種類の ICMP を拒否します。yes に設定すると、icmp-blocks で設定した種類の ICMP を受け入れます。

4.5.1 ターゲットを default に設定

ターゲットを default に設定します。

```
$ sudo firewall-cmd --zone=public --set-target=default --permanent
success
$ sudo firewall-cmd --reload
```

```
$ sudo firewall-cmd --zone=public --query-icmp-block-inversion
no
$ sudo firewall-cmd --list-icmp-blocks
```

icmp-block-inversion は no なので icmp-blocks で設定した ICMP を拒否しますが、icmp-blocks には何も設定されていないので、何も拒否せず、すべての種類の ICMP を受け入れます。

ホスト OS から ping コマンドを実行すると、応答が返ってきます。

```
C:\Users\LinuC>ping 192.168.56.101

192.168.56.101 に ping を送信しています 32 バイトのデータ:
192.168.56.101 からの応答: バイト数 =32 時間 =1ms TTL=64
192.168.56.101 からの応答: バイト数 =32 時間 =1ms TTL=64
192.168.56.101 からの応答: バイト数 =32 時間 =1ms TTL=64

192.168.56.101 の ping 統計:
    パケット数: 送信 = 3、受信 = 3、損失 = 0 (0% の損失)、
    ラウンドトリップの概算時間 (ミリ秒):
        最小 = 1ms、最大 = 1ms、平均 = 1ms
Ctrl+C
^C
```

4.5.2 icmp-block-inversion の設定

icmp-block-inversion を yes に設定するためには、firewall-cmd - add-icmp-block-inversion コマンドを実行します。この設定で、icmp-blocks には何も設定されていないので、すべての種類の ICMP を拒否することになります。

```
$ sudo firewall-cmd --zone=public --add-icmp-block-inversion
success
$ sudo firewall-cmd --zone=public --query-icmp-block-inversion
yes
```

ホスト OS から ping コマンドを実行すると、エラー応答が返ってきます。

```
C:\Users\LinuC>ping 192.168.56.101

192.168.56.101 に ping を送信しています 32 バイトのデータ:
192.168.56.101 からの応答: 宛先ネットワークに到達できません。
192.168.56.101 からの応答: 宛先ネットワークに到達できません。
192.168.56.101 からの応答: 宛先ネットワークに到達できません。
192.168.56.101 からの応答: 宛先ネットワークに到達できません。

192.168.56.101 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
```

4.5.3 icmp-block-inversion の設定を元に戻す

icmp-block-inversion を no に戻すには、firewall-cmd - remove-icmp-block-inversion コマンドを実行します。

```
$ sudo firewall-cmd --remove-icmp-block-inversion
success
$ sudo firewall-cmd --zone=public --query-icmp-block-inversion
no
```

5 ルーティング/IP マスカレード/プロキシ

ルーティングは、インターネットの基本的な仕組みであるネットワーク間を接続するための仕組みです。遠く離れたネットワーク同士が通信できるのは、ルーティングによってパケットがルーターを経由して適切に転送されるためです。

IP マスカレードは、ローカルネットワークからグローバルネットワークにパケットを送り出す際にアドレスを変換する Linux の仕組みです。

別々のネットワークを接続する実習として、IP マスカレードの設定を解説します。

また、特定の通信プロトコルを中継する仕組みであるプロキシの設定も解説します。

5.1 ルーティング

ルーティングは、送信側が接続されているネットワークと異なるネットワークに IP 通信を行う仕組みです。

ルーターがルーティングを行います。受信側のネットワークが直接接続されていない場合、複数のルーターを介してルーティングが行われる場合もあります。

5.1.1 ルーター

ルーターは、第 2 章で解説したように異なるネットワークとの間で IP 通信を行うための機器です。

Linux を使ってルーターを構成することもできます。販売されている家庭用のインターネット接続用のルーターには、内部的に Linux を組み込んでいるものも多く存在しています。

ルーターの設定や、複数のルーターを介したルーティングなどは本教科書の想定範囲を越えるため説明は省略しますが、ネットワークについてより深く理解したい場合には学習してみてください。

5.1.2 ルーターと L3 スイッチの違い

ネットワーク機器に L3 スイッチと呼ばれるものがあります。一般的なネットワークスイッチは L2 スイッチ、すなわちイーサネットのレイヤーでスイッチングを行います。L3 スイッチは IP のレイヤーでスイッチングを行います。L3 スイッチに接続されているネットワーク同士を IP で接続するので、ルーターの一種といえます。オフィスなどで複数の IP ネットワークに分割されているネットワークを相互に接続する場合などに L3 スイッチは利用されます。

ルーターは主に外部と接続する機器、L3 スイッチは LAN の内部で接続する機器と考えておけばよいでしょう。

5.2 アドレス変換

ルーティングはパケットを書き換えずにそのまま送信していくのに対して、アドレス変換はパケットの書き換えを伴う通信方法です。アドレス変換にもいくつかの方式があるので、それぞれの特長を解説します。

5.2.1 NAT

NAT (Network Address Translation) は、IP アドレスのみを書き換えるアドレス変換方式です。送信元の IP アドレスと書き換え後の IP アドレスが 1 対 1 で対応するので、複数のノードからの送信書き換えには対応できません。

5.2.2 NAPT

NAPT (Network Address Port Translation) は、IP アドレスとポート番号の両方を書き換えるアドレス変換方式です。ポート番号を含めることによって、複数のノードからの送信パケットを書き換えることができます。一般的なインターネットに接続するためのルーターは、インターネットプロバイダーから与えられた 1 個のグローバル IP アドレスを複数のノードで共用するために NAPT によるアドレス変換を行っています。

NAPT はアドレス変換した IP アドレスとポート番号の組み合わせをすべて記録しておき、受信側からの応答パケットを逆方向に書き戻す必要があるため、沢山の通信が行われる場合には変換テーブルが溢れてしまって処理可能な量を超えてしまい、通信が行えなくなる場合があります。

5.2.3 ポートフォワーディング

NAT や NAPT は、LAN からインターネットに出て行く場合のアドレス変換ですが、外部のインターネット側から内側の LAN に入ってくる場合のアドレス変換がポートフォワーディングです。

ルーターに与えられたグローバル IP アドレスの特定のポートに外部からアクセスがあった場合、内部の特定の IP アドレスとポート番号に対するパケットとして書き換えを行い、転送する仕組みです。

VirtualBox では、「NAT ネットワーク」を構成することでポートフォワーディングを設定し、外部からのパケットを仮想マシンに転送することができます。

また、コンテナを外部ネットワークからアクセスできるようにする仕組みとしてもポートフォワーディングは利用されています。

5.3 IP マスカレードによるアドレス変換

IP マスカレードは、Linux で NAPT を行う `nftables` の機能です。

NAPT の動作を確認するために、IP マスカレードを設定し、動作させてみましょう。

5.3.1 システムの構成

クライアント用仮想マシンを別途作成し、実習用の仮想マシンの IP マスカレードを経由してインターネットに接続させます。

クライアント用仮想マシンと実習用の仮想マシンは、VirtualBox のホストオンリーネットワークで接続します。実習用の仮想マシンは VirtualBox の NAT を経由して、さらにルーターの NAPT を経由してインターネットに接続しているので、多段のアドレス変換を行っていることになります。

IP マスカレード実習構成図

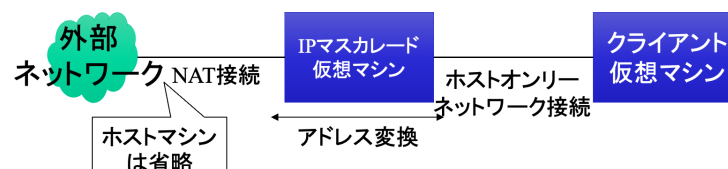


図 52: IP マスカレード構成図

5.3.2 ネットワークインターフェースを確認する

2 つあるネットワークインターフェースがそれぞれネットワークの外側、内側のいずれに接続されているかを確認します。

```
$ ip a
(略)
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   group default qlen 1000
   link/ether 08:00:27:91:bb:d1 brd ff:ff:ff:ff:ff:ff
```

```

    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
(略)
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    group default qlen 1000
    link/ether 08:00:27:f5:15:7b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic
        noprefixroute enp0s8 (略)

```

IP アドレスから、**enp0s3** がネットワークの外側 (NAT)、**enp0s8** が内側 (ホストオンリーネットワーク) に接続されていることが確認できます。

5.3.3 ゾーンの確認

firewalld のゾーンとしてあらかじめ設定されているものを確認します。

```

$ firewall-cmd --get-zones
block dmz drop external home internal nm-shared public trusted work

```

ネットワークの外側として **external** ゾーン、内側として **internal** ゾーンを使用することにします。後ほど確認しますが、**external** ゾーンにはあらかじめ IP マスカレードマスカレードが設定されているので、細かい設定をしないでも IP マスカレードを動作させることができます。

5.3.4 現在のゾーンの状態を確認

現在のゾーンの状態を確認します。

```

$ firewall-cmd --get-active-zone
public
    interfaces: enp0s3 enp0s8

```

デフォルトゾーンである **public** ゾーンのみが使用されており、2 つのネットワークインターフェースが **public** ゾーンに接続されています。

5.3.5 ネットワークインターフェースの接続ゾーンの変更

2 つのネットワークインターフェースをそれぞれ **external** ゾーンと **internal** ゾーンに接続します。設定の変更には **nmcli** コマンドを使用します。

```

$ sudo nmcli connection modify enp0s3 connection.zone external
$ sudo nmcli connection modify enp0s8 connection.zone internal

```

ゾーンの状態を確認します。

```

$ firewall-cmd --get-active-zone
external
    interfaces: enp0s3
internal
    interfaces: enp0s8

```

各ネットワークインターフェースの接続ゾーンが変更されたことが確認できます。

5.3.6 IP マスカレードの設定を確認

IP マスカレードが設定されていることを確認します。

まず **external** ゾーンの設定を確認します。

```
$ sudo firewall-cmd --list-all --zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

external ゾーンは、forward が yes になっているので、他のインターフェースとの間でのパケット転送が行われます。そして、masquerade が yes になっているので、IP マスカレードでパケットの書き換えが行われます。

internal ゾーンの設定も確認します。

```
$ sudo firewall-cmd --list-all --zone=internal
internal (active)
  target: ACCEPT
  icmp-block-inversion: no
  interfaces: enp0s8
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

internal ゾーンも、forward が yes になっているので、他のインターフェースとの間でのパケット転送が行われます。しかし、masquerade が no になっているので、IP マスカレードでパケットの書き換えは行われません。

5.3.7 internal ザーンのターゲットを ACCEPT に変更する

デフォルトでは internal ザーンのターゲットが default になっているため、クライアント用仮想マシンから送られてきた転送すべきパケットをすべて落としてしまいます。internal ザーンのターゲットを ACCEPT に変更します。

```
$ sudo firewall-cmd --zone=internal --set-target=ACCEPT --permanent
```

設定後- reload オプションで読み込み直します。

```
$ sudo firewall-cmd --reload
```

target の設定が変更されたことを確認します。

```
$ sudo firewall-cmd --zone=internal --list-all
internal (active)
  target: ACCEPT
(略)
```

internal ザーンの target が ACCEPT になっていることが確認できます。

5.4 IP マスカレードの動作の確認

クライアント用仮想マシンを用意して、IP マスカレードの動作を確認します。

5.4.1 クライアント用仮想マシンを準備する

VirtualBox でクライアント用の仮想マシンを作成し、AlmaLinux をインストールします。

実習用仮想マシンとクライアント用仮想マシンは、ホストオンリーネットワークで接続します。クライアント用仮想マシンには NAT によるインターネット接続は必要ありませんので、デフォルトで設定されたネットワークアダプター 1 を「NAT」から「ホストオンリーネットワーク」に変更します。

5.4.2 クライアント仮想マシンの参照 DNS とデフォルトゲートウェイを変更する

クライアント用の仮想マシンに、参照 DNS とデフォルトゲートウェイを設定します。

IP アドレスは、VirtualBox のホストオンリーネットワークに接続されている DHCP サーバーが自動的に設定してくれていますが、参照 DNS とデフォルトゲートウェイは設定されていません。

参照 DNS は、Google 社がインターネット上に用意している Public DNS の「8.8.8.8」を指定します。

デフォルトゲートウェイは、実習用仮想マシンのホストオンリーネットワークに接続されているネットワークインターフェースの IP アドレスを調べて設定します。ここでは「192.168.56.101」を設定します。

GUI のネットワーク設定から、ネットワークインターフェースの IPv4 設定を呼び出します。

「DNS」の「自動」のチェックを外し、テキストボックスに参照する DNS の IP アドレスを入力します。

「ルート」の「自動」のチェックを外し、テキストボックスにゲートウェイの IP アドレスを入力します。

設定項目	意味	設定値
アドレス	ルーティング先のネットワークアドレス	0.0.0.0
ネットマスク	ルーティング先のネットマスク（省略可能）	省略
ゲートウェイ	ルーターの IP アドレス	192.168.56.101
メトリック	優先順位（省略可能）	省略

アドレスはすべてを表す「0.0.0.0」、ゲートウェイは IP マスカレードを設定した仮想マシンのホストオンリーネットワークに接続されている IP アドレス「192.168.56.101」を指定します。

設定を行ったら、「適用」ボタンをクリックします。

設定をネットワークインターフェースに適用するためには、歯車ボタンの左にあるスイッチをクリックして一旦ネットワークインターフェースを無効にし、再度スイッチをクリックして有効にする必要があります。

5.4.3 設定の確認

正しく設定が行われたか、確認します。

GUI でも確認できますが、コマンドでも確認してみます。

名前解決の設定は、`/etc/resolv.conf` に記述されます。

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 8.8.8.8
```

デフォルトゲートウェイの設定は、`ip route` コマンドで確認します。

```
$ ip route
default via 192.168.56.101 dev enp0s3 proto static metric 100
192.168.56.0/24 dev enp0s3 proto kernel scope link src 192.168.56.102 metric 100
```

設定したアドレス「0.0.0.0」は、ルーティングテーブルでは `default` として扱われます。

キャンセル(C)

有線

適用(A)

詳細

Identity

IPv4

IPv6

セキュリティ

IPv4 メソッド(4)

☒ 自動 (DHCP)

☐ リンクローカルのみ

☐ 手動

☐ 無効

☐ 他のコンピューターと共有

DNS

自動 ☐

8.8.8.8

複数の IP アドレスを指定する場合はそれぞれコンマで区切ってください

ルート

自動 ☐

アドレス	ネットマスク	ゲートウェイ	メトリック
0.0.0.0	0.0.0.0	192.168.56.101	✕
			✕

☐ この接続はネットワーク上のリソースのためだけに使用する(O)

図 53: 参照 DNS とゲートウェイの設定

キャンセル(C)

有線

適用(A)

詳細

Identity

IPv4

IPv6

セキュリティ

リンクの速度 1000 Mb/s

IPv4 アドレス 192.168.56.102

IPv6 アドレス fe80::a00:27ff:fe3d:4e59

ハードウェアアドレス 08:00:27:3D:4E:59

デフォルトルート 192.168.56.101

DNS 8.8.8.8

☒ 自動接続する(A)

☒ 他のユーザーも利用できるようにする(O)

従量制接続(M): データ通信量に上限がある、または通信料がかかります

☐ ソフトウェアのアップデートや、その他のサイズが大きいもののダウンロードは、自動的に開始されませ
ん。

接続プロファイルを削除

図 54: 参照 DNS とゲートウェイの設定を確認

5.5 名前解決できることを確認

クライアント用仮想マシンで、名前解決や Web サーバーへのアクセスができることを確認します。

```
$ dig linuc.org

; <<>> DiG 9.16.23-RH <<>> linuc.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34053
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;linuc.org.                IN      A

;; ANSWER SECTION:
linuc.org.                 3600    IN      A      219.94.236.161

;; AUTHORITY SECTION:
linuc.org.                 73654   IN      NS      02.dnsv.jp.
linuc.org.                 73654   IN      NS      03.dnsv.jp.
linuc.org.                 73654   IN      NS      04.dnsv.jp.
linuc.org.                 73654   IN      NS      01.dnsv.jp.

;; Query time: 15 msec
;; SERVER: 192.168.11.1#53(192.168.11.1)
;; WHEN: Mon Jul 06 18:36:02 JST 2026
;; MSG SIZE rcvd: 129
```

5.5.1 Web サーバーに接続できることを確認

HTTP でもアクセスしてみます。

```
$ curl linuc.org
<html>
<head><title>302 Found</title></head>
<body>
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

HTTP のため、正しくページが表示されていない旨のメッセージが返って来ていますが、きちんとインターネット上の Web サーバーと通信が行えていることがわかります。

クライアント用仮想マシンで Web ブラウザを起動して、いろいろなサイトにアクセスしてみてください。

5.6 Wireshark で IP マスカレードを見てみる

IP マスカレードでパケットの書き換えが起きていることを確認します。

Wireshark を起動し、enp0s3 と enp0s8 の両方のネットワークインターフェースのパケットをキャプチャしてみます。複数のネットワークインターフェースを選択するには、Ctrl キーを押しながらネットワークインターフェースを選択します。

キャプチャした状態で dig コマンドで名前解決を行い、パケットが書き換えられる様子を確認してみます。絞り込み条件で「dns」を設定すると、名前解決のパケットのみが表示されます。

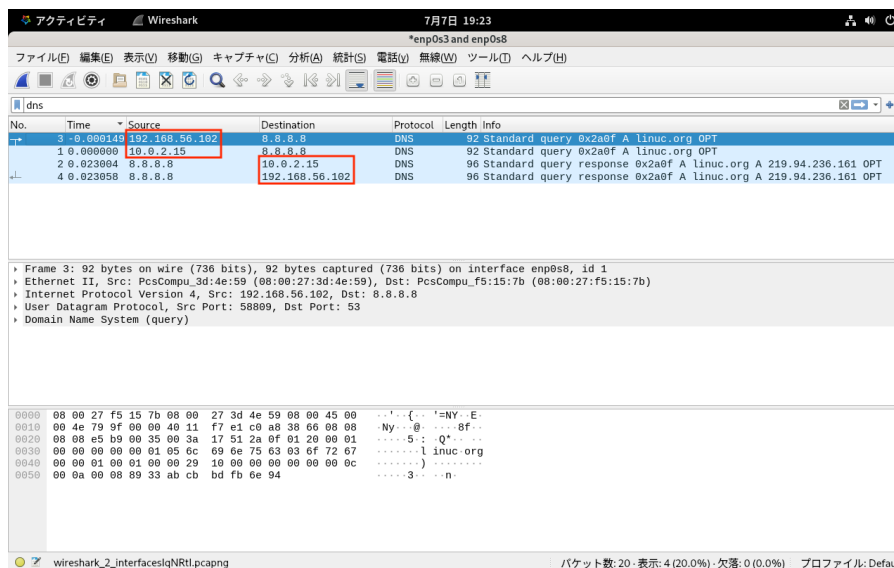


図 55: IP マスカレードによる送信元 IP アドレス書き換えの確認

実行例では、4つのパケットがキャプチャされています。非常に高速な処理な為、Time の表示が丸められてしまっていますが、リスト表示の Time をクリックすると時間順にソート表示されます。問い合わせのパケットの送信元が「192.168.56.102」から「10.0.2.15」に書き換えられ、戻って来たパケットの返信先が「10.0.2.15」から「192.168.56.102」に書き換えられているのがわかります。

5.7 IP マスカレードを解除する

IP マスカレードを解除するには、ネットワークインターフェースを public ゾーンに接続します。

```
$ sudo nmcli connection modify enp0s3 connection.zone public
$ sudo nmcli connection modify enp0s8 connection.zone public
```

インターフェースの接続が public ゾーンに戻ったことを確認します。

```
$ firewall-cmd --get-active-zone
public
interfaces: enp0s3 enp0s8
```

internal ゾーンの target も default に戻しておきます。

```
$ sudo firewall-cmd --zone=internal --set-target=default --permanent
```

設定後、firewall-cmd - reload コマンドで読み込み直します。

```
$ sudo firewall-cmd --reload
```

internal ゾーンの設定を確認します。

```
$ sudo firewall-cmd --list-all --zone=internal
internal
  target: default
  icmp-block-inversion: no
  interfaces:
  (略)
```

target が default に戻っていることが確認できます。

5.8 プロキシの設定

プロキシは、特定の通信プロトコルを中継する仕組みです。主に Web アクセスで利用されており、アクセスログを取得したり、許可されてない Web サイトへのアクセスを禁止したりすることに使われています。

AlmaLinux では Squid を使うことでプロキシを実行できます。

5.8.1 Squid のインストール

dnf コマンドで Squid をインストールします。

```
$ sudo dnf install squid
```

5.8.2 Squid の起動

Squid を起動します。また、システム起動時に自動的に起動するように設定します。

```
$ sudo systemctl start squid
$ sudo systemctl enable squid
Created symlink /etc/systemd/system/multi-user.target.wants/squid.service →
/usr/lib/systemd/system/squid.service.
```

5.8.3 ポートの待ち受け状況の確認

ポートの待ち受け状況を確認します。

```
$ sudo lsof -i -P | grep squid
squid      25084  squid    7u  IPv6  62718      0t0  UDP  *:45065
squid      25084  squid    8u  IPv4  62719      0t0  UDP  *:33449
squid      25084  squid   11u  IPv6  63515      0t0  TCP  *:3128 (LISTEN)
```

TCP のポート番号 3128 で待ち受けています。UDP のポートがあるのは、プロキシが名前解決を行うための処理をポート固定で行っているためです。

5.8.4 firewalld の設定変更

クライアント用仮想マシンを Squid に接続させるためには、ポート番号 3128 への接続を許可する必要があります。firewall-cmd コマンドで firewalld の設定を変更します。

```
$ sudo firewall-cmd --add-port=3128/tcp --zone=public --permanent
$ sudo firewall-cmd --reload
```

5.9 クライアント用仮想マシンのプロキシ設定

クライアント用仮想マシンのプロキシ設定は、GUI の設定アプリで設定可能です。

クライアント用仮想マシンで「設定」アプリを起動して、「ネットワーク」を選択します。「ネットワーク」プロキシの歯車ボタンをクリックします。

「ネットワークプロキシ」ダイアログが表示されるので「手動」を選択し、「HTTP プロキシ」「HTTPS プロキシ」を設定します。IP アドレスは **Squid** をインストールした実習用仮想マシンのホストオンリーネットワークに接続された IP アドレス「192.168.56.101」、ポート番号は 3128 番をそれぞれ設定します。ダイアログ右上の「X」をクリックして、ダイアログを閉じます。



図 56: プロキシの設定

プロキシの設定は GUI 環境の環境変数として設定されるので、「端末」を起動して `env` コマンドで確認します。環境変数「`HTTPS_PROXY`」、「`HTTP_PROXY`」、「`NO_PROXY`」が設定されていることを確認します。環境変数が設定されていない場合、一度「端末」を終了して、再度「端末」を起動してみてください。

```
$ env | sort
(略)
HTTPS_PROXY=http://192.168.56.101:3128/
HTTP_PROXY=http://192.168.56.101:3128/
(略)
NO_PROXY=localhost,127.0.0.0/8,::1
(略)
http_proxy=http://192.168.56.101:3128/
https_proxy=http://192.168.56.101:3128/
no_proxy=localhost,127.0.0.0/8,::1
(略)
```

5.9.1 リモートログインではプロキシは設定されない

GUI で設定したプロキシ設定は、GUI 環境でのみ有効です。そのため、SSH でリモートログインしても環境変数は設定されません。

リモートログインしたシェルでプロキシを設定したい場合には、別途 `.bashrc` などに同様の環境変数を設定してください。

5.10 プロキシ経由でアクセスする

準備ができたなら、クライアント用仮想マシンからプロキシ経由でインターネットの Web サーバーにアクセスします。

5.10.1 Squid のアクセスログ確認

Squid がプロキシとして動作していることを確認するために、Squid のアクセスログを表示します。

```
$ sudo tail -f /var/log/squid/access.log
```

5.10.2 プロキシ経由で Web サーバーにアクセス

クライアント用仮想マシンで Web ブラウザからインターネット上のサイトにアクセスします。

以下のようなログが表示されたら、プロキシが機能していることが確認できます。

```
1783427785.546      6790 192.168.56.102 TCP_TUNNEL/200 15772 CONNECT linuc.org:443  
- HIER_DIRECT/219.94.236.161 -
```

6 ネットワークインターフェースの管理

ネットワークインターフェースは、Linux が動作するハードウェアに備えられているネットワーク機能を指します。

Linux では、ドライバとしてカーネルモジュールをロードして使うことができますようになります。通常は各インターフェースに IP アドレスを割り当てて通信を行いますが、複数のインターフェースを束ねて冗長化することで性能や耐障害性を高めることができます。

ネットワークインターフェースの基本的な管理方法と、冗長化の方法について解説します。

6.1 ネットワークインターフェースとは

ネットワークインターフェースの接続方式は、有線 LAN や無線 LAN、物理的な形状としてオンボードにネットワークカード、USB 接続など、様々な種類があります。

6.1.1 オンボード

ハードウェアの中心基板であるマザーボード上に最初から備えられているネットワークインターフェース。

6.1.2 ネットワークカード (NIC)

ハードウェアの PCI (Peripheral Component Interconnect) 拡張スロットに挿すボード形状のネットワークインターフェース。この形式が最も一般的なため、形式に関係なくネットワークインターフェース全般を NIC (Network Interface Card) と呼ぶことがある。

6.1.3 USB 接続

オンボードも PCI 拡張スロットも無い場合、USB 接続のネットワークインターフェースが使用できます。

6.1.4 無線 LAN

Linux では無線 LAN (Wi-Fi) も使用できます。有線 LAN 同様、オンボードや USB 接続など、様々な種類があります。

6.2 実習環境のネットワークインターフェース

実習環境は仮想マシンで用意された仮想ネットワークインターフェースです。ゲスト OS の Linux では、PCI に接続された有線 LAN のネットワークカードとして扱われています。

6.3 インターフェース名の命名規則

ネットワークインターフェースは、命名規則に従ってインターフェース名がつけられています。

実習環境で使用しているネットワークインターフェースを例に、命名規則を確認してみましょう。

まず、Linux 起動時のログを `dmesg` コマンドで確認し、ネットワークインターフェースの情報を確認します。

```
$ sudo dmesg | grep enp
[  3.360306] e1000 0000:00:08.0 enp0s8: renamed from eth1
[  3.381082] e1000 0000:00:03.0 enp0s3: renamed from eth0
[ 13.890225] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: RX
[ 13.950392] e1000: enp0s8 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: RX
```

6.3.1 インターフェース名の意味

e1000 はカーネルモジュールとして読み込まれたネットワークインターフェース用のデバイスドライバです。その後ろが PCI デバイスとしての番号です。頭についている 0000: は無視して、00:08.0 の部分だけを参照します。それぞれの意味は以下の通りです。

値	意味
00	PCI バス番号
08	デバイス番号 (スロット番号)
0	ファンクション番号

これらのうち、バス番号 (PCI の p) とスロット番号 (Slot の s) がインターフェース名に使用されます。ファンクション番号は使用されません。

従来、一番最後に書かれている `eth0` や `eth1` のように認識された順番にインターフェース名がつけられていましたが、新たにネットワークインターフェースが追加されると順番が変わってしまいました。そこでハードウェア的な接続位置によってインターフェース名が設定されるようになりました。ただし、PCI スロットの位置を変えるとインターフェース名も変わり、それまでの設定との整合性が取れなくなるので注意が必要です。

6.3.2 インターフェース種別とインターフェース名

また、先頭についている `en` はイーサネット (Ethernet) から取られています。インターフェースの種別によって、以下のような名称が使われます。

記号	種類
en	有線イーサネット
wl	無線 LAN (Wi-Fi)
ww	WWAN (LTE/5G など)

6.4 インターフェースの詳細情報

使用できる状態のネットワークインターフェースは、`ethtool` コマンドを実行することでカーネルモジュールなどの詳細情報を確認できます。

```
$ ethtool -i enp0s8
driver: e1000
version: 5.14.0-687.36.1.el9_8.x86_64
firmware-version:
expansion-rom-version:
bus-info: 0000:00:08.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

6.5 ボンディングによるネットワーク冗長化

Linux でサーバーを動作させていると、ネットワーク障害が発生するとサーバーに接続できなくなります。このような状態を「ネットワークが単一障害点」と言います。単一障害点は SPOF (Single Point of Failure) とも呼びます。

ネットワークを SPOF にしないために、ネットワークインターフェースを複数用意し、ボンディングすることでネットワークを冗長化して SPOF を解消できます。

6.5.1 ボンディングの種類

ネットワークのボンディングには、その動作によっていくつかの種類があります。

	名称	負荷分散	スイッチ対応
0	balance-rr	○	要
1	active-backup	×	不要
2	balance-xor	○	要
3	broadcast	×	要
4	802.3ad	○	要
5	balance-tlb	○	不要
6	balance-alb	○	不要

6.5.2 負荷分散

冗長化した複数のネットワークインターフェースを使って、ネットワークの帯域を増やして、より沢山の通信が行えるようにします。

6.5.3 スイッチ対応

ネットワークインターフェースが接続されているスイッチ側に設定が必要です。ボンディングの種類によって、LAG や LACP など設定が異なります。

6.6 ボンディングの準備

ボンディングを行うためのネットワークインターフェースを仮想マシンに追加します。

1. 仮想マシンをシャットダウンします。
2. Oracle VirtualBox マネージャーで仮想マシンの設定を呼び出します。
3. ネットワークアダプター 3 を有効化します。
4. 「ホストオンリーネットワーク」に割り当てます。
5. ネットワークアダプター 4 を有効化します。
6. 「ホストオンリーネットワーク」に割り当てます。
7. 仮想マシンを起動します。
8. 追加した 2 つのネットワークインターフェースが認識されていることを確認します。

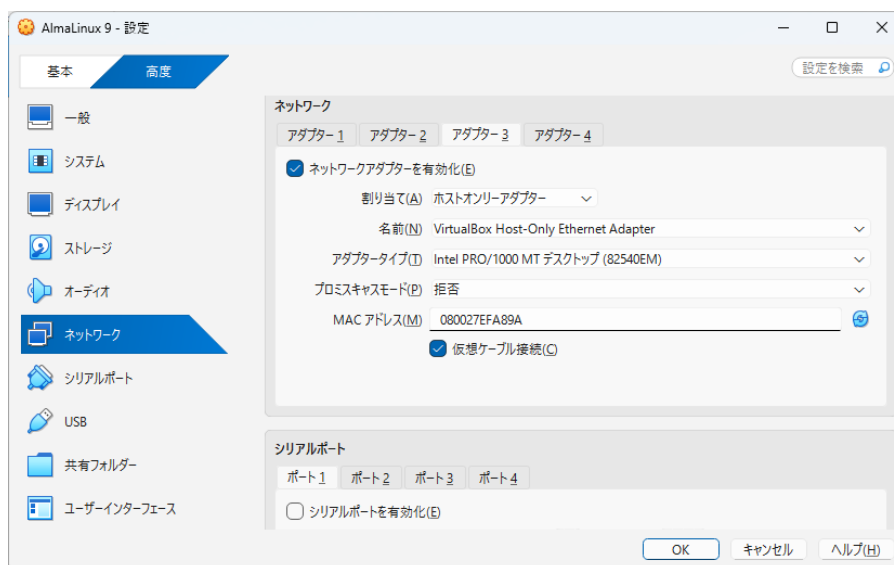


図 57: 仮想マシンにネットワークアダプター 3 を追加

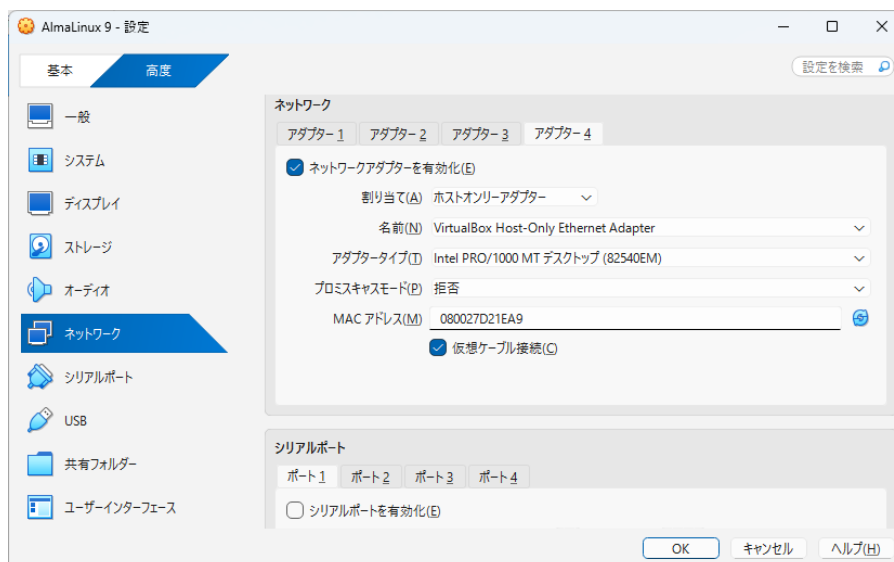


図 58: 仮想マシンにネットワークアダプター 4 を追加

```
$ ip a
(略)
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   group default qlen 1000
   link/ether 08:00:27:ef:a8:9a brd ff:ff:ff:ff:ff:ff
5: enp0s10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   group default qlen 1000
   link/ether 08:00:27:d2:1e:a9 brd ff:ff:ff:ff:ff:ff
```

6.7 ボンディングの設定

仮想マシンに追加した 2 つのネットワークアダプターを使って、ボンディングを設定します。

6.7.1 ボンディングデバイスの作成

まず最初にボンディングデバイス `bond0` を作成します。ボンディングモードはスイッチ側の設定が不要な `balance-tlb` を指定します。

```
$ sudo nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=balance-tlb"
接続 'bond0' (5e62b315-cbdf-4d92-81db-b5f6b5e29acb) が正常に追加されました。
```

6.7.2 ネットワークインターフェースの確認

作成したボンディングデバイスおよび、仮想マシンに追加したネットワークインターフェースを確認します。確認には、`nmcli device status` コマンド、および `nmcli conn show` コマンドを実行します。

DEVICE	TYPE	STATE	CONNECTION
enp0s3	ethernet	接続済み	enp0s3
enp0s8	ethernet	接続済み	enp0s8
bond0	bond	接続中 (IP 設定を取得中)	bond0
lo	loopback	接続済み (外部)	lo
enp0s10	ethernet	切断済み	--
enp0s9	ethernet	切断済み	--

ボンディングデバイスと、追加した 2 つのネットワークインターフェースが確認できます。

```
$ nmcli conn show
NAME      UUID                                  TYPE      DEVICE
enp0s3    7e141e48-25d0-336e-85dd-dbda3bc81918 ethernet  enp0s3
enp0s8    e4458f90-f841-37b2-a918-38446edc1d46 ethernet  enp0s8
bond0     18ac4a63-a638-4945-ac39-aee11552e9b4 bond       bond0
lo        43b40be6-f1bb-460b-a707-5225b18b77c6 loopback   lo
```

追加したネットワークインターフェースは、まだ接続されていない状態です。

6.7.3 デバイスの追加

2つのネットワークインターフェースをボンディングデバイスに接続します。

```
$ sudo nmcli conn add type ethernet slave-type bond con-name bond0-1 ifname
enp0s9 master bond0
接続 'bond0-1' (d9f3f510-3e87-4e68-ad91-771431602fad) が正常に追加されました。
$ sudo nmcli conn add type ethernet slave-type bond con-name bond0-2 ifname
enp0s10 master bond0
接続 'bond0-2' (82f17a99-344b-4cae-b936-6359c1563e0b) が正常に追加されました。
```

6.7.4 ボンディングデバイスの有効化

ボンディングデバイスを有効にします。

```
$ sudo nmcli conn up bond0
接続が正常にアクティベートされました (controller waiting for ports) (D-Bus
アクティブパス: /org/freedesktop/NetworkManager/ActiveConnection/9)
```

6.7.5 ボンディングデバイスの確認

ボンディングデバイスの状態を確認します。

```
$ nmcli dev status
DEVICE    TYPE      STATE      CONNECTION
enp0s3    ethernet  接続済み   enp0s3
enp0s8    ethernet  接続済み   enp0s8
bond0     bond      接続済み   bond0
enp0s10   ethernet  接続済み   bond0-2
enp0s9    ethernet  接続済み   bond0-1
lo        loopback  接続済み (外部) lo
```

コネクションも確認します。

```
$ nmcli conn show
NAME      UUID                                  TYPE      DEVICE
enp0s3    7e141e48-25d0-336e-85dd-dbda3bc81918 ethernet  enp0s3
enp0s8    e4458f90-f841-37b2-a918-38446edc1d46 ethernet  enp0s8
bond0     5e62b315-cbdf-4d92-81db-b5f6b5e29acb bond       bond0
bond0-1   d9f3f510-3e87-4e68-ad91-771431602fad ethernet  enp0s9
bond0-2   82f17a99-344b-4cae-b936-6359c1563e0b ethernet  enp0s10
lo        43b40be6-f1bb-460b-a707-5225b18b77c6 loopback   lo
```

6.8 ボンディングデバイスの自動構成

ここまでの作業では手動でネットワークインターフェースをボンディングデバイスに接続しましたが、再起動をすると接続は行われません。自動的に接続が行われるようにボンディングデバイスのパラメータ `connection.autoconnect-slaves` の設定値を 1 (有効) に変更します。

```
$ sudo nmcli conn modify bond0 connection.autoconnect-slaves 1
```

一度ボンディングデバイスを無効化し、再度有効化してネットワークインターフェースがボンディングデバイスに自動接続することを確認します。

```
$ sudo nmcli conn down bond0
接続 'bond0' が正常に非アクティブ化されました (D-Bus アクティブパス:
/org/freedesktop/NetworkManager/ActiveConnection/9)
$ sudo nmcli conn up bond0
接続が正常にアクティベートされました (controller waiting for ports) (D-Bus
アクティブパス: /org/freedesktop/NetworkManager/ActiveConnection/12)
```

ボンディングデバイスの状態は `proc` ファイルで確認することもできます。

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v5.14.0-687.36.1.el9_8.x86_64

Bonding Mode: transmit load balancing
Primary Slave: None
Currently Active Slave: enp0s10
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: enp0s10
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:d2:1e:a9
Slave queue ID: 0

Slave Interface: enp0s9
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:ef:a8:9a
Slave queue ID: 0
```

6.9 ボンディングの動作確認

作成したボンディングデバイスの動作を確認します。

まず、ボンディングデバイスに割り当てられた IP アドレスを確認します。

```
$ ip a
(略)
8: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 08:00:27:d2:1e:a9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.106/24 brd 192.168.56.255 scope global dynamic
        noprefixroute bond0
        valid_lft 462sec preferred_lft 462sec
    inet6 fe80::5fe6:1060:df5:6267/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

ホスト OS からボンディングデバイスの IP アドレスに対して `ssh` コマンドでリモートログインを実行してみます。

```
>ssh linuc@192.168.56.106
```

リモートログインできたら、接続したままにしておきます。

6.10 ボンディングデバイスの冗長化テストを行う

ボンディングデバイスに接続された 2 つのネットワークインターフェースのうち、片方を無効化します。疑似的にネットワークインターフェース、あるいは接続されているスイッチのポートに障害が発生した状態にします。

VirtualBox でネットワークアダプター 4 を無効化します。仮想マシンのウインドウの右下にあるネットワークアイコン（左から 4 番目）をクリックして「C ネットワークアダプターを接続 4」のチェックを外します。仮想マシンの設定画面から、ネットワークアダプター 4 の「仮想ケーブル接続」のチェックを外しても同様です。

ボンディングデバイスの状態を確認します。

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v5.14.0-687.36.1.el9_8.x86_64

Bonding Mode: transmit load balancing
Primary Slave: None
Currently Active Slave: enp0s9
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: enp0s10
MII Status: down
Speed: Unknown
Duplex: Unknown
Link Failure Count: 1
Permanent HW addr: 08:00:27:d2:1e:a9
Slave queue ID: 0

Slave Interface: enp0s9
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:ef:a8:9a
Slave queue ID: 0
```

片方のネットワークインターフェース `enp0s10` の `MII Status` が `down` になっているのがわかります。

この状態で、先ほど実行した `SSH` のリモートログインの接続が切断されず継続されていることを確認します。

7 Ubuntu のネットワーク管理

この章では、Ubuntu のネットワーク管理について解説します。

7.1 実習で使用する Ubuntu のバージョン

Ubuntu は 2 年おきに LTS (Long Term Support) と呼ばれる長期サポート版のバージョンがリリースされています。本教科書の実習環境では、Ubuntu 26.04 LTS/24.04 LTS を使っています。

7.1.1 実習環境の構築

実習環境の構築は、VirtualBox の仮想マシンについては第 1 章を、OS のインストールは「Linux サーバー構築標準教科書 Ubuntu 版」を参考に行ってください。

「Linux サーバー構築標準教科書」ダウンロードページ <https://linuc.org/textbooks/server/>

7.2 Ubuntu のネットワーク設定は Netplan を使用

Ubuntu のネットワーク管理では Netplan が設定の管理を行い、OS のバックエンド (レンダラー) がネットワークインターフェイスに IP アドレスなどを付与します。Netplan は、`/etc/netplan/` 配下の `*.yaml` ファイルに設定を記載し、それぞれのレンダラーに合わせてコンフィグを生成します。

レンダラーは、Ubuntu Desktop では NetworkManager、Ubuntu Server では systemd-networkd が担います。

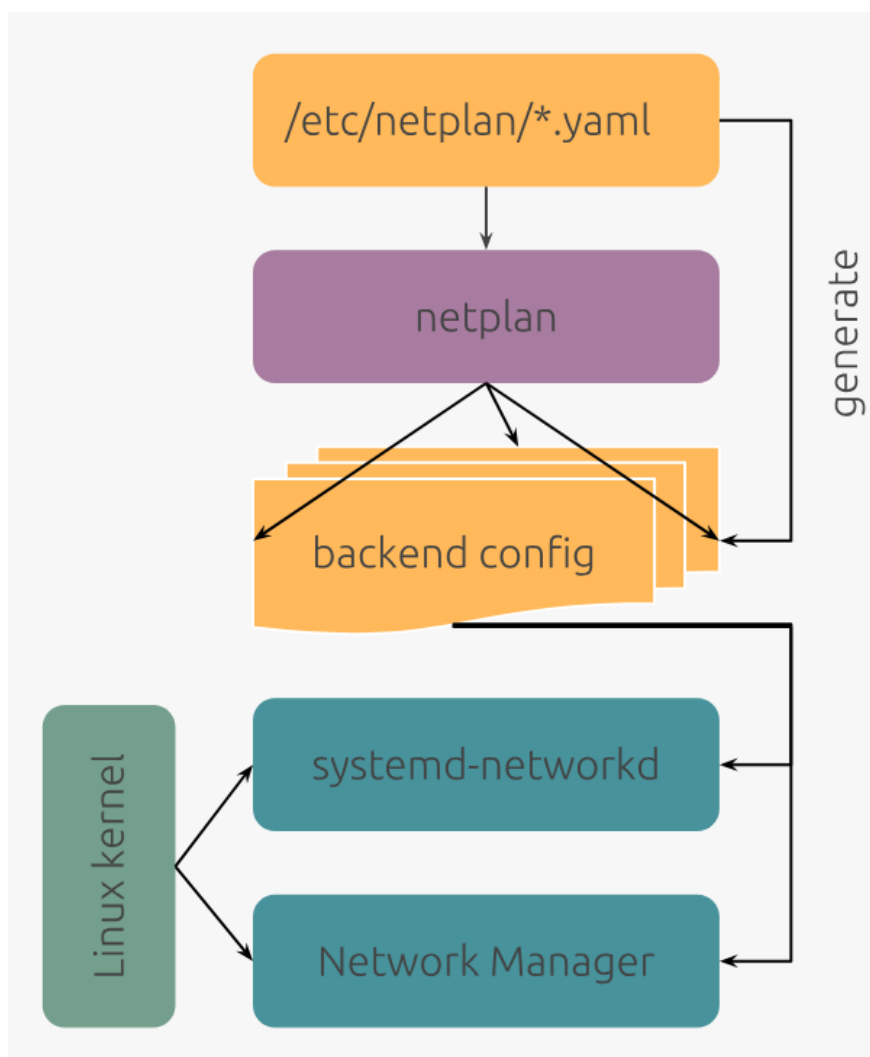


図 59: Ubuntu のネットワーク管理

基本的な使用方法では Netplan だけを見ておけばよいので、管理者がレンダラーの違いを意識する必要はありません。

7.3 Ubuntu Desktop でのネットワーク設定

Ubuntu Desktop では、AlmaLinux 同様に GUI 設定ツールや `nmtui` コマンド、`nmcli` コマンドが使用できます。ここでは GUI 操作でネットワーク設定を行います。

タスクバー (画面の例だと右上) から設定アイコンをクリックし、ネットワークメニューから設定を行います。

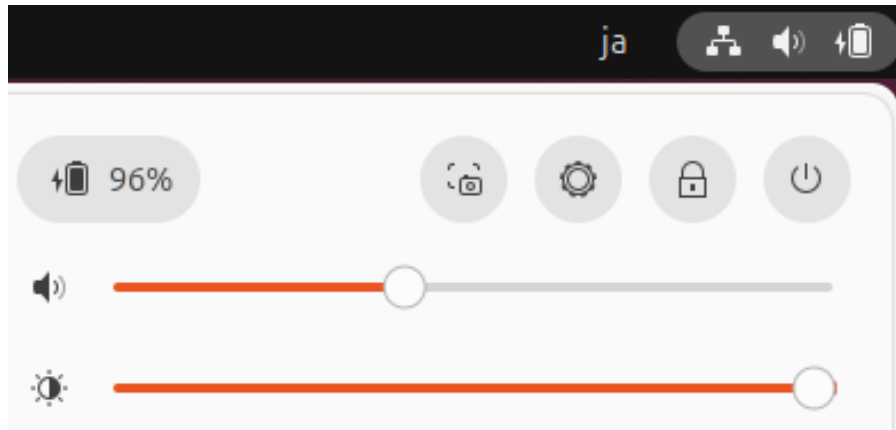


図 60: 画面右上の設定アイコン

以下の例では、DHCP でネットワーク設定を行っています。

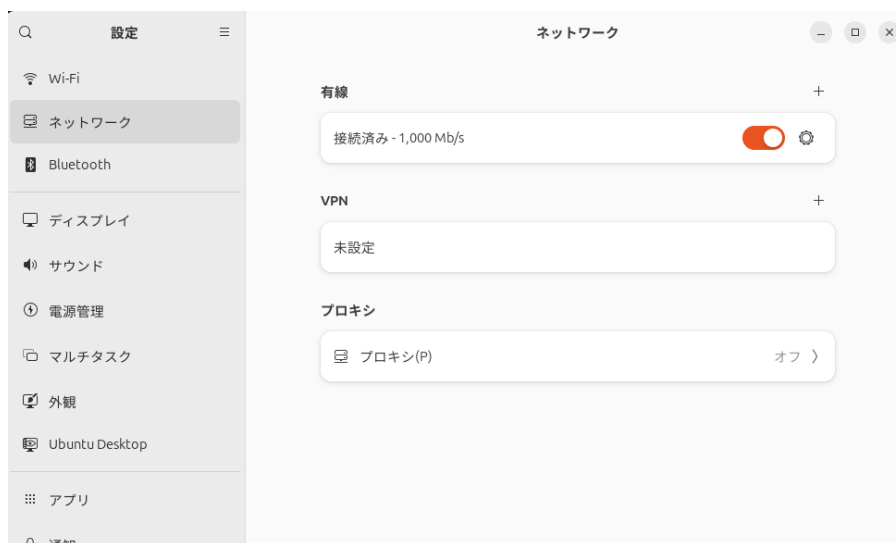


図 61: ネットワーク設定画面

The screenshot shows the 'Network Settings' window for a wired connection. The 'IPv4' tab is selected. Under 'IPv4 Method', 'Automatic (DHCP)' is selected. The 'DNS' and 'Routes' sections are both set to 'Automatic' and are currently empty.

キャンセル(C) 有線 適用(A)

詳細 識別子 IPv4 IPv6 セキュリティ

IPv4 メソッド(4)

☒ 自動 (DHCP) ☐ リンクローカルのみ

☐ 手動 ☐ 無効

☐ 他のコンピューターと共有

DNS 自動 ☒

複数の IP アドレスを指定する場合はそれぞれコンマで区切ってください

ルート 自動 ☒

アドレス	ネットマスク	ゲートウェイ	メトリック

☐ この接続はネットワーク上のリソースのためだけに使用する(O)

図 62: DHCP でネットワーク設定

7.3.1 固定 IP アドレスを設定する

固定 IP アドレスなどを設定する場合、手動を選択します。

The screenshot shows the 'Network Settings' window for a wired connection. The 'IPv4' tab is selected. Under 'IPv4 Method', 'Manual' is selected. In the 'Addresses' section, the IP address '192.168.1.143' is entered with a netmask of '255.255.255.0' and a gateway of '192.168.1.1'. The 'DNS' section is set to 'Automatic' and contains the list '8.8.8.8, 8.8.4.4, 1.1.1.1'. The 'Routes' section is also set to 'Automatic' and is empty.

キャンセル(C) 有線 適用(A)

詳細 識別子 IPv4 IPv6 セキュリティ

IPv4 メソッド(4)

☐ 自動 (DHCP) ☐ リンクローカルのみ

☒ 手動 ☐ 無効

☐ 他のコンピューターと共有

アドレス

アドレス	ネットマスク	ゲートウェイ
192.168.1.143	255.255.255.0	192.168.1.1

DNS 自動 ☒

8.8.8.8, 8.8.4.4, 1.1.1.1

複数の IP アドレスを指定する場合はそれぞれコンマで区切ってください

ルート 自動 ☒

アドレス	ネットマスク	ゲートウェイ	メトリック

図 63: 手動でネットワーク設定

7.3.2 Wi-Fi を設定する

また、無線 NW(Wi-Fi) は、Wi-Fi メニューから設定を行います。



図 64: Wi-Fi 設定画面

7.4 Ubuntu Server でのネットワーク設定

Ubuntu Server では CLI(コマンド) 操作でネットワーク設定を行います。

7.4.1 IP アドレスとデフォルトゲートウェイを確認する

設定を確認すると、IP アドレスとして 192.168.1.143、デフォルトゲートウェイは 192.168.1.1 が設定されていることが分かります。

```
ubuntu@ubuntu2604:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 08:00:27:f4:77:82 brd ff:ff:ff:ff:ff:ff
    altname enx080027f47782
    inet 192.168.1.143/24 metric 100 brd 192.168.1.255 scope global dynamic
        enp0s3
        valid_lft 7157sec preferred_lft 7157sec
    inet6 240f:32:57b8:1:a00:27ff:fef4:7782/64 scope global dynamic mngtmpaddr
        noprefixroute
        valid_lft 295sec preferred_lft 295sec
    inet6 fe80::a00:27ff:fef4:7782/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever

ubuntu@ubuntu2604:~$ ip route show
default via 192.168.1.1 dev enp0s3 proto dhcp src 192.168.1.143 metric 100
```

7.4.2 Netplan の設定ファイル

IP アドレスとデフォルトゲートウェイの設定は、以下の Netplan 設定ファイル (YAML ファイル) で行われています。

```
ubuntu@ubuntu2604:~$ sudo cat /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
```

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
      dhcp6: true
      match:
        macaddress: 08:00:27:f4:77:82
      set-name: enp0s3
  version: 2
```

7.4.3 固定 IP アドレスへの変更

DHCP から固定 IP に設定を変更する場合は、上記 YAML ファイルを変更します。

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
        - 192.168.1.142/24
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4, 1.1.1.1]
```

7.4.4 変更の適用

変更の適用は、`netplan try` コマンドを実行します。

`netplan try` コマンドでは設定適用の即時の確定を行わず、猶予時間（デフォルトでは 120 秒、`- timeout` オプションで指定可能）が設定されます。その猶予期間中にリモートアクセス (SSH) など必要な通信が問題なければ、**Enter** キーを押して設定の適用を確定します。猶予期間中に **Enter** キーが押されなかった場合には、ネットワークの設定はコマンド実行前の設定にロールバックされます。

なお、設定の確定を即座に行いたい場合には、`netplan apply` コマンドを実行します。

```
ubuntu@ubuntu2604:/etc/netplan$ sudo netplan try
Do you want to keep these settings?
Press ENTER before the timeout to accept the new configuration
Changes will revert in 111 seconds
Configuration accepted

ubuntu@ubuntu2604:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 08:00:27:f4:77:82 brd ff:ff:ff:ff:ff:ff
    altname enx080027f47782
    inet 192.168.1.142/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
```

```
inet6 240f:32:57b8:1:a00:27ff:fe4:7782/64 scope global dynamic mngtmpaddr
    noprefixroute
    valid_lft 299sec preferred_lft 299sec
inet6 fe80::a00:27ff:fe4:7782/64 scope link proto kernel_ll
    valid_lft forever preferred_lft forever
```

7.4.5 DNS 設定の確認

DNS 設定は `resolvectl status` コマンドで確認できます。

```
ubuntu@ubuntu2604:~$ resolvectl status enp0s3
Link 2 (enp0s3)
    Current Scopes: DNS
        Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 8.8.8.8
    DNS Servers: 8.8.8.8 8.8.4.4 1.1.1.1
    Default Route: yes
```

7.4.6 Netplan から生成されたコンフィグの確認

Netplan の YAML ファイルから生成されたコンフィグは、`/run/systemd/network/10-netplan-enp0s3.network` というファイルに反映されています。

Ubuntu Server のレンダラーである `systemd-networkd` はこのファイルを読み込んでネットワークの設定を行っています。

```
ubuntu@ubuntu2604:~$ sudo cat /run/systemd/network/10-netplan-enp0s3.network
[Match]
Name=enp0s3

[Network]
LinkLocalAddressing=ipv6
Address=192.168.1.142/24
DNS=8.8.8.8
DNS=8.8.4.4
DNS=1.1.1.1

[Route]
Destination=0.0.0.0/0
Gateway=192.168.1.1
```

7.5 ボンディングについて

詳細な設定方法については割愛しますが、`netplan` ではボンディングにも対応しています。<https://netplan.readthedocs.io/en/stable/multi-nic-vm-host-with-bonds-and-vlans/>

7.6 パケットフィルタリング

Ubuntu のパケットフィルタリングでは、`UFW(Uncomplicated Firewall)` を使用します。

`UFW` の動作を確認するため、Web サーバである `apache2` をインストールします。

```
ubuntu@ubuntu2604:~$ sudo apt update

ubuntu@ubuntu2604:~$ sudo apt install apache2
```

インストール後、Ubuntu の IP 宛てに他 PC のブラウザよりアクセスすると、「It works!」と書かれたデフォルトページが表示されます。初期状態では `UFW` は非アクティブとなっており、Ubuntu へアクセスしてくる通信は制御していません。



図 65: Web サーバのデフォルトページ

では、UFW の設定を行います。

7.6.1 UFW の状態を確認

まず、UFW の状態を確認します。

```
ubuntu@ubuntu2604:~$ sudo ufw status
Status: inactive
```

7.6.2 UFW の制御ルールを追加

この状態で UFW をアクティブ (有効化) すると、全ての通信をブロックしてしまうので、サーバの管理に必要な SSH(22/tcp) や、Web サーバーへのアクセスで必要になる HTTP/HTTPS(80/tcp・443/tcp) への通信を許可します。

```
ubuntu@ubuntu2604:~$ sudo ufw allow proto tcp from 192.168.1.0/24 to any port 22
Rules updated

ubuntu@ubuntu2604:~$ sudo ufw allow proto tcp from 192.168.1.0/24 to any port 80
Rules updated

ubuntu@ubuntu2604:~$ sudo ufw allow proto tcp from 192.168.1.0/24 to any port
443
Rules updated
```

7.6.3 UFW をアクティブ化

UFW をアクティブにします。

```
ubuntu@ubuntu2604:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

7.6.4 UFW の制御ルールの確認

UFW の制御ルールを確認します。

```
ubuntu@ubuntu2604:~$ sudo ufw status
Status: active

To Action From
--
22/tcp ALLOW 192.168.1.0/24
80/tcp ALLOW 192.168.1.0/24
443/tcp ALLOW 192.168.1.0/24
```

7.6.5 UFW の動作確認

ブラウザから Web サーバーにアクセスできることを確認してみてください。

ルールで許可しているので、変わらずアクセスできることが確認できます。

7.6.6 Web サーバーへのアクセスを拒否してみる

試しに HTTP/HTTPS(80/tcp・443/tcp) の通信を拒否してみます。

```
ubuntu@ubuntu2604:~$ sudo ufw deny proto tcp from 192.168.1.0/24 to any port 443
Rule updated
ubuntu@ubuntu2604:~$ sudo ufw deny proto tcp from 192.168.1.0/24 to any port 80
Rule updated
ubuntu@ubuntu2604:~$ sudo ufw status
Status: active

To Action From
--
22/tcp ALLOW 192.168.1.0/24
80/tcp DENY 192.168.1.0/24
443/tcp DENY 192.168.1.0/24
```

設定後、再度ブラウザからアクセスすると、通信が拒否され表示できないメッセージが返されます。

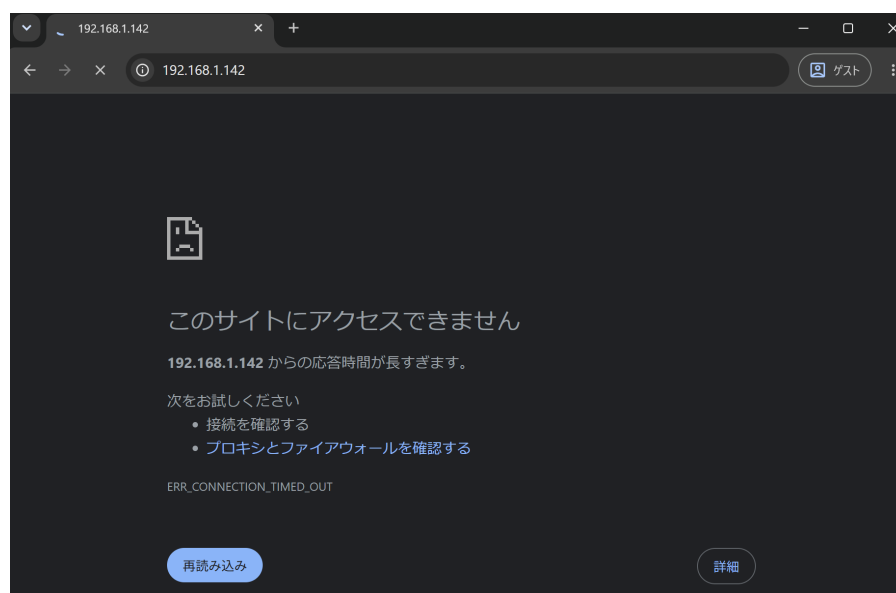


図 66: Web ブラウザのアクセスエラー画面

Linux ネットワーク標準教科書

2026 年 10 月 1 日 Ver.1.0.0

発行元 LPI-Japan

Copyright © LPI-Japan. All Rights Reserved.