LinuC

LinuC レベル 1 Version10.0
技術解説無料セミナー
2020/9/26 開催

主題	1.07 ネットワークの基礎
副題	1.07.1　インターネットプロトコルの基礎 1.07.2　基本的なネットワーク構成 1.07.3　基本的なネットワークの問題解決 1.07.4　クライアント側のDNS設定

INTERNOUS インターノウス株式会社
（LPI-Japanアカデミック認定校）

竹本 季史

LPI-JAPAN

会社紹介、自己紹介

#LinuC学習中

■会社紹介：インターノウス株式会社

- ●人材紹介サービス、人材派遣/SESサービス、IT未経験者の教育及び就職支援サービス、法人研修サービス
- ●未経験からインフラエンジニアやプログラマーになりたい方へ、無料で研修と就職支援サービスを行っています。
  https://engineercollege.jp/lp/

■自己紹介：竹本 季史(たけもと ときふみ)

- ●IT業界で約10年間勤務後、インターノウス株式会社エンジニアカレッジ講師。
- ●これまで約700人を未経験者からエンジニアに養成。Linuxサーバー(メール、OpenSSH、シェルスクリプト、DB、監視、演習)を担当。
- ●LinuCレベル1バージョン10.0の差分教材で「仮想マシン・コンテナの概念と利用」を執筆。

LinuCレベル１／レベル２ Version10.0とは

#LinuC学習中

■LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定資格

✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化を問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した、今、身につけておくべき技術範囲を網羅

今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

✓Linuxの範疇だけにとどまらない領域までカバー

セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

#LinuC学習中

■Version10.0と従来の出題範囲の比較

	テーマ	Version 10.0	従来
LinuC-1	仮想技術	・仮想マシン／コンテナの概念 ・クラウドセキュリティの基礎	← （Version10.0で新設）
	オープンソースの文化	・オープンソースの定義や特徴 ・コミュニティやエコシステムへの貢献	← （Version10.0で新設）
	その他	→ （Version10.0で削除）	アクセシビリティ、ディスククォータ、プリンタの管理、SQLデータ管理、他
LinuC-2	仮想化技術	・仮想マシンの実行と管理（KVM） ・コンテナの仕組みとDockerの導入	← （Version10.0で新設）
	システムアーキテクチャ	・クラウドサービス上のシステム構成 ・高可用システム、スケーラビリティ、他	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	← （Version10.0で出題範囲に含む）
		→ （Version10.0で削除）	RAID、記憶装置へのアクセス方、FTPサーバーの保護、他

#LinuC学習中

■本セミナーについて

- ●本セミナーは試験範囲で問われる内容の理解を深めるために
  ポイントを解説いたします。
- ●このセミナーの内容はCentOS7を前提とした内容となっています。

■受講者の想定スキルレベル

- ●LinuCレベル1の取得を目指している方

■本セミナーのゴール

- ●IPアドレスの仕組みが分かる。
- ●TCP/IPの代表的なプロトコルが分かる。
- ●ネットワークコマンドを使用してネットワークの調査ができる。

主題1.07：ネットワークの基礎

#LinuC学習中

■1.07.1 インターネットプロトコルの基礎(抜粋)

- ネットワークマスクとCIDR表記法。
  - サブネット化
- プライベートとパブリックのドット区切り形式のIPアドレスの違い。
- UDP、TCP、およびICMPの違いや主な機能

■1.07.2 基本的なネットワーク構成(抜粋)

- ネットワークインターフェイスの設定を手作業および自動で行う。
  - /etc/hostname
  - nmcli, ip addr
- ホストの基本的なTCP/IP設定
  - /etc/hosts, /etc/nsswitch.conf, ping
- デフォルトルートの設定
  - ip route, route

主題1.07：ネットワークの基礎

#LinuC学習中

■1.07.3 基本的なネットワークの問題解決(抜粋)

- ネットワークに関する問題の原因を調査する。
  - host, dig, ping
  - traceroute, tracepath
- 必要に応じてネットワークインターフェイスの追加、起動、停止、再起動、削除、および再設定する。
  - hostname
- ルーティングテーブルを変更、参照、設定し、不適切なデフォルトルート設定を手作業で訂正する。
  - ip route, route

#LinuC学習中

1.07.4 クライアント側のDNS設定

- リモートDNSサーバーに問い合わせる。
  - host, dig
- ローカルの名前解決の設定によりリモートDNSサーバーを使用する。
  - /etc/resolv.conf
- 名前解決の実行順序を変更する。
  - getent, /etc/nsswitch.conf, /etc/hosts

目次

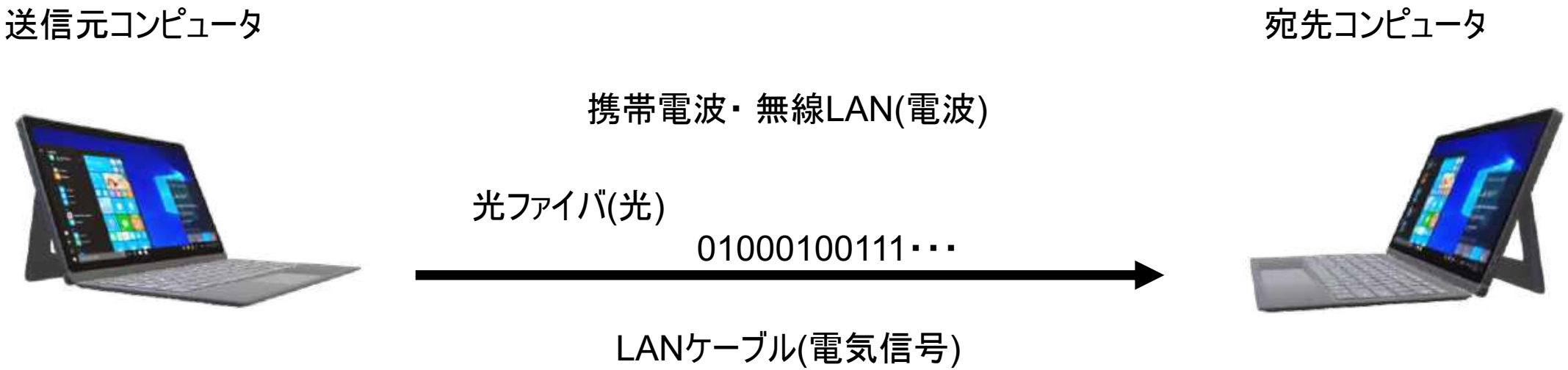
#LinuC学習中

- プロトコルとは？
- TCP/IPとは？
- IPアドレス
- ネットワークのコマンドやファイル

プロトコルとは？　①コンピューター間の通信

#LinuC学習中

コンピュータは0と1で情報を処理します。
コンピュータ間で通信をする際には、電気信号、光、電波などの伝送媒体で0と1を表現して宛先コンピュータまで情報を届けます。



プロトコルとは？　②通信ルールがない状態

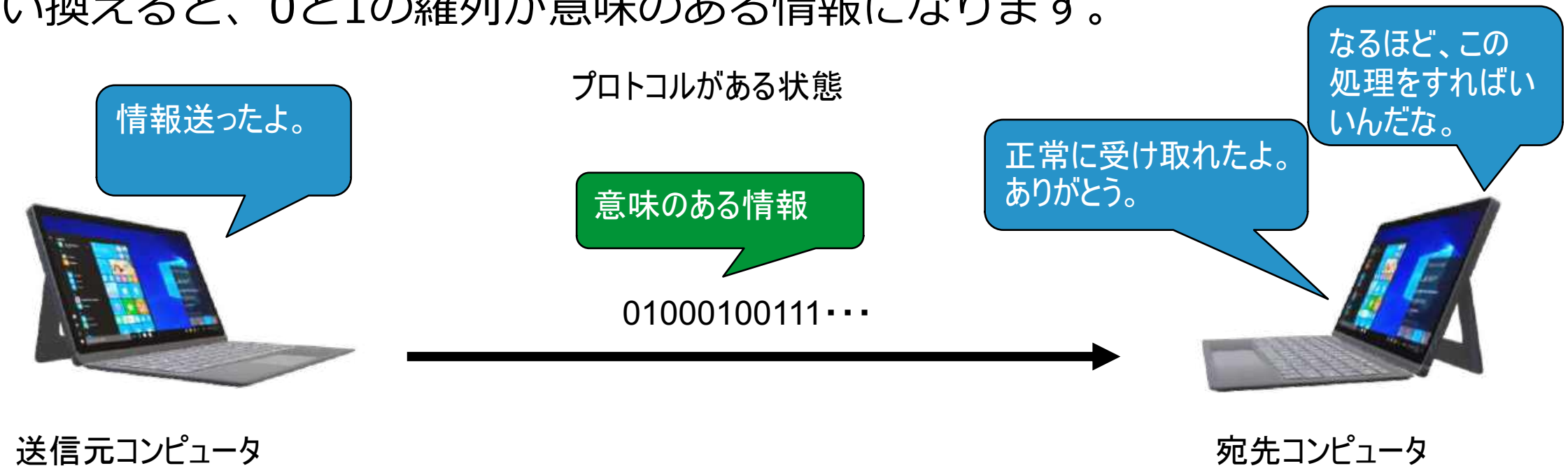
#LinuC学習中

コンピュータ同士の通信ではあらかじめ、通信に関するルールを決めておく必要があります。ルールがなければ宛先コンピュータが受け取るものは0と1の羅列に過ぎず、意味のある情報として受け取れません。

プロトコルとは？　③プロトコルが決めていること

#LinuC学習中

コンピュータ同士の通信のルールを**プロトコル**と呼びます。プロトコルは「通信手順」 と「データの形式」を決定しています。
「通信手順」が決められていることによって宛先コンピュータに正常に情報の受け渡しができます。そして、「データの形式」が決められていることによって宛先コンピュータは受け取った情報が何の情報かを認識できます。
言い換えると、0と1の羅列が意味のある情報になります。

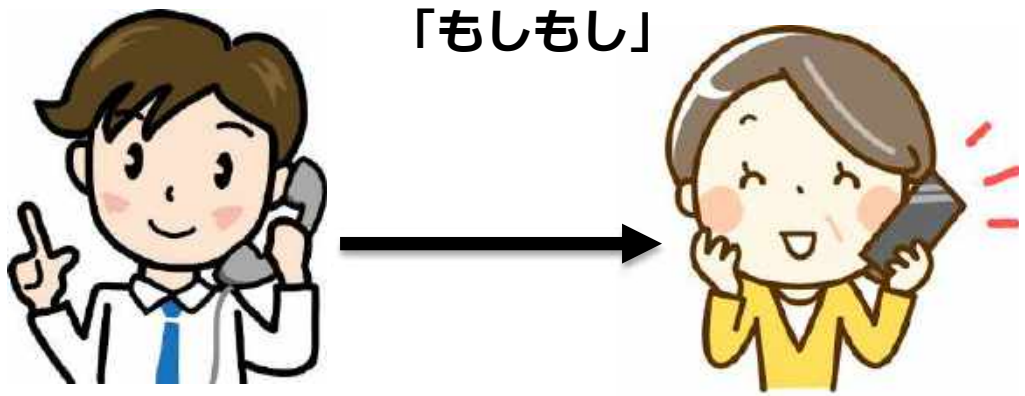


プロトコルとは？　④人同士のコミュニケーションの例

#LinuC学習中

電話に出た時に、急に「あの件だけどさぁ」と早口で要件をまくし立てられたらどう感じるでしょうか？また、話を理解できるでしょうか？

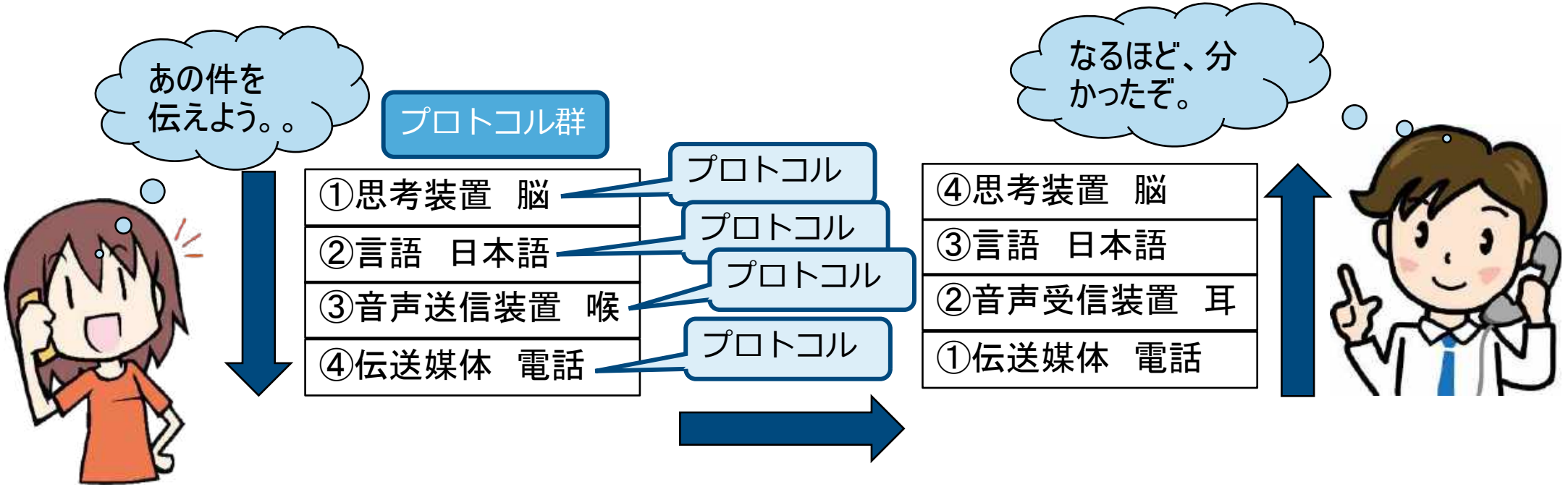
電話では「もしもし」と言ってから会話を始めて相手の応答を確認してから、要件を話すという暗黙の了解があります。
これはプロトコルの通信手順にあたります。また、対面よりも相手の様子が分かりずらいことから、伝わっていることを確認しながら話すのではないでしょうか？



プロトコルとは？　⑤プロトコルはシンプル+階層化

#LinuC学習中

ひとつのプロトコルで通信に関する全てを実現しようとすると、
プログラムが大きく、複雑になり修正することが難しくなります。
現在のコンピュータ通信はシンプルな機能のプロトコルを階層的に組み合わせて実現しています。これによって、プログラムの修正を容易にしたり、特定の層のプロトコルの入れ替えができます。このような階層化されたプロトコルをまとめて**プロトコル群**と言います。



TCP/IPとは？　①TCP/IPプロトコル群について

#LinuC学習中

現在インターネットを中心に広く使用されているプロトコル群として、
TCP/IPプロトコル群があります。中でも代表的なプロトコルのTCPとIPから名前がつけられています。
コンピュータ同士がプロトコル群としてTCP/IPを使用していれば、お互いにデータの送受信ができます。片方が他のプロトコル群を使用していると正常にデータの送受信ができません。
過去には様々なプロトコル群がありましたが、現在ではTCP/IPが標準的となっており、宛先のプロトコル群を気にすることはなくなっています。

TCP/IPとは？　②TCP/IPプロトコル群の4階層

#LinuC学習中

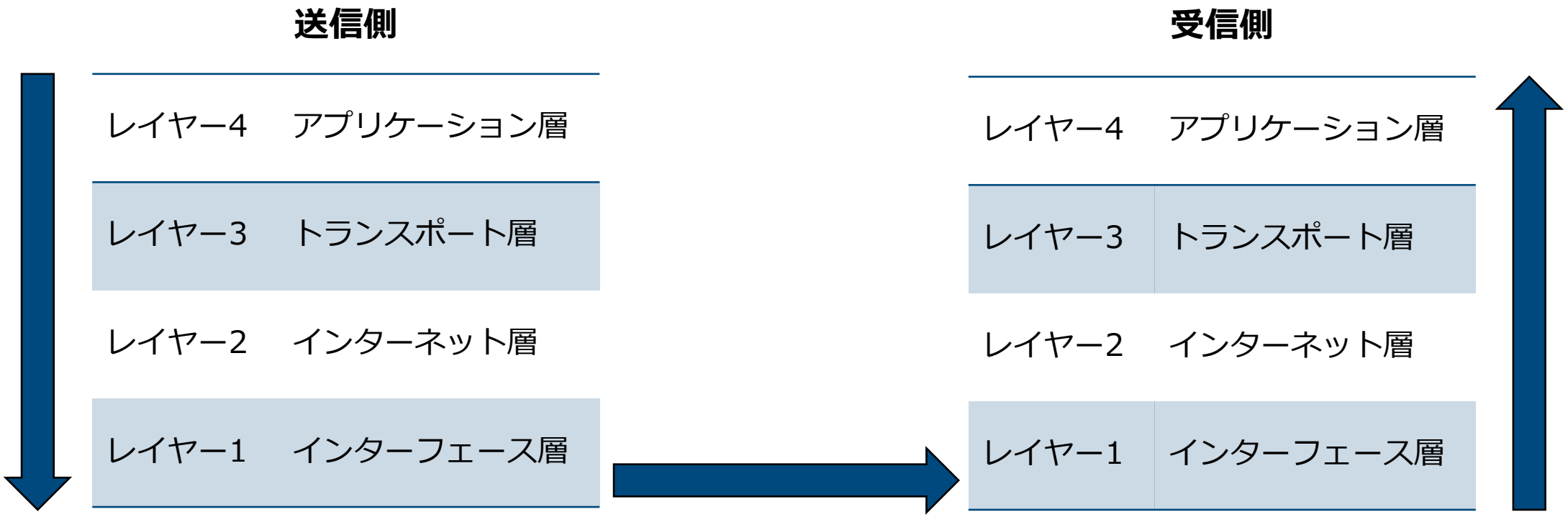
TCP/IP プロトコル群は表のように4つの階層に分けられます。

階層	階層の名前	階層の役割	階層の代表的なプロトコル
レイヤー4	アプリケーション層	アプリケーション同士でデータの形式や手順を定める役割	HTTP(Hyper Text Transfer Protocol) : Webページの閲覧 SMTP(Simple Mail Transfer Protocol) : メールの送信 など
レイヤー3	トランスポート層	エラー訂正や再送などの通信の管理やデータを宛先アプリケーションまで届ける役割	**TCP**(Transmission Control Protocol) **UDP**(User Datagram Protocol)
レイヤー2	インターネット層	複数のネットワークを相互接続した環境で宛先まで通信をする役割	**IP**(Internet Protocol) **ICMP**(Internet Control Message Protocol) ARP(Address Resolution Protocol)
レイヤー1	インターフェース層	伝送媒体を通じて直接接続した宛先と通信をする役割	イーサネット(Ethernet) フレームリレー(Frame-Relay) PPP(Point-to-Point Protocol)など

TCP/IPとは？　③TCP/IPの4階層の通信

#LinuC学習中

TCP/IP プロトコル群を使うコンピュータは下図のように通信します。
送信側のアプリケーション層から下位の層に向かって各層のプロトコルが処理をします。受信側はデータを受け取ったら、上位の層に向かって各層のプロトコルが処理してアプリケーション層が意味のある情報として受け取ります。



TCP/IPとは? ④トランスポート層のプロトコル

#LinuC学習中

トランスポート層はインターネット層とアプリケーション層の仲立ちをする層です。送信時はアプリケーションから受け取ったデータを宛先まで届ける通信方式の決定や管理。受信時は受け取ったデータをアプリケーションまで届ける役割となります。

TCP (Transmission Control Protocol)
信頼性の高い通信を実現するためのコネクション型のプロトコルです。消失やエラーが発生したパケットを再送したり、パケットを順番どおりに並び替えたり、到達したことを確認します。

UDP (User Datagram Protocol)
データの転送速度が速いコネクションレス型のプロトコルです。相手に届いているかの保証はしません。途中でデータが欠けても支障が少ない音声や映像のストリーミング配信で利用されます。また、DNSの名前解決やNTPでの時刻情報の配信でも使用されます。

TCP/IPとは? ⑤インターネット層のプロトコル

#LinuC学習中

インターネット層はトランスポート層のひとつ下に位置する層で、宛先までデータを送り届ける役割です。

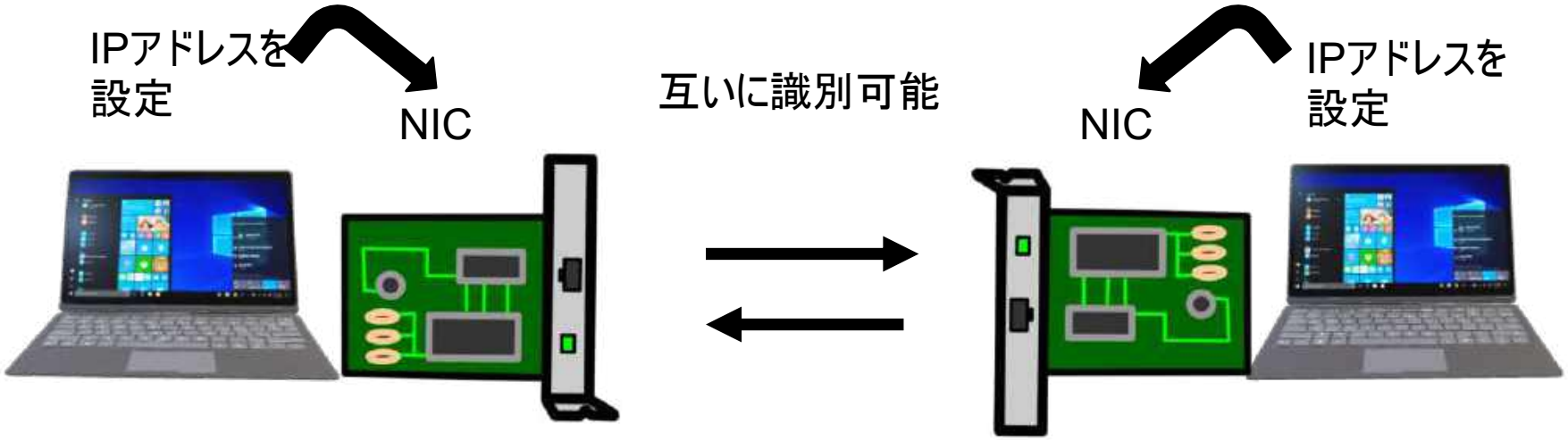
IP (Internet Protocol)
機器がどのネットワークの、どの機器かを識別するための**IPアドレス**を付与して、目的の機器までデータ(パケット)を届けるプロトコルです。
送信する機器では、扱いやすいようにパケットを分割し、到着した機器ではパケットを再構成します。

ICMP (Internet Control Message Protocol)
パケットのエラーメッセージや制御メッセージを通知するプロトコルです。
pingやtracerouteコマンドで使用されます。

IPアドレス　①IPアドレスについて

#LinuC学習中

IPはTCP/IPの中核をなすプロトコルです。コンピュータ間で通信をするために自身と宛先の識別をするために必要となるのが**IPアドレス**です。
IPアドレスはPCやサーバーの**ネットワークインターフェースカード(NIC)**にソフトウェア的に設定します。相手を識別する必要があるため、同一ネットワーク内に同じIPアドレスを複数の機器に割り当てることはできません。
IPアドレスにはv4とv6がありますが、ここでは現在主に使われるIPv4のみ説明します。



IPアドレス　②2進数の数え方

#LinuC学習中

IPアドレスについて解説する前に、IPアドレスは2進数で書き表すことがあるので2進数について理解しておきましょう。

2進数では0と1の数字のみ扱います。

1桁で表現できないときには左隣に桁を増やして0に戻ります。これを繰り返します。

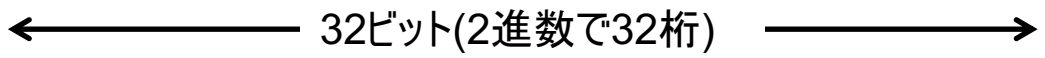
例えば10進数の2を2進数で表す場合は10(イチゼロ)となります。

10進数	2進数
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

IPアドレス　③IPアドレスの表記

#LinuC学習中

IPアドレス(**IPv4**)は**32ビット**で構成されます。8ビットごとのまとまりをオクテットと言い、「.」ドットで区切ります。4つのオクテットから構成されます。コンピュータ内部では2進数で処理されますが、通常は人が分かりやすいように10進数で表記されます。



2進数　11000000101010000000101000000001

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

オクテットごとに区切る　11000000.10101000.00001010.00000001

10進数　192 . 168 . 10 . 1

IPアドレス　④IPアドレスの10進数→2進数変換

#LinuC学習中

通常IPアドレスは10進数で表記されますが、2進数に変換することがあります。
オクテットごとに2進数に変換します。
まずオクテットの値から、2進数で1番左の桁の128(2の7乗)を引きます。引けたら1、引けなかったら0を書きます。残りの値から右隣に1つずらした桁の64(2の6乗)を引きます。以降、残り0になるまで繰り返し引いていきます。
以下では、第2オクテットの「168」を2進数に変換しています。

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
192 .	168 .	10 .	1

	8bit目	7bit目	6bit目	5bit目	4bit目	3bit目	2bit目	1bit目
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
10進数	128	64	32	16	8	4	2	1
引ける=>1 引けない=>0	168-128 =40	40-64	40-32 =8	8-16	8-8 =0			
2進数	1	0	1	0	1	0	0	0

残りが0になったので0

IPアドレス　⑤IPアドレスの2進数→10進数変換

#LinuC学習中

2進数のIPアドレスを10進数に変換することもあります。
オクテットのビットが「1」の桁の2の累乗を値を出して、全て足します。
以下では、第2オクテットの「10101000」を10進数に変換しています。

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

11000000.10101000.00001010.00000001

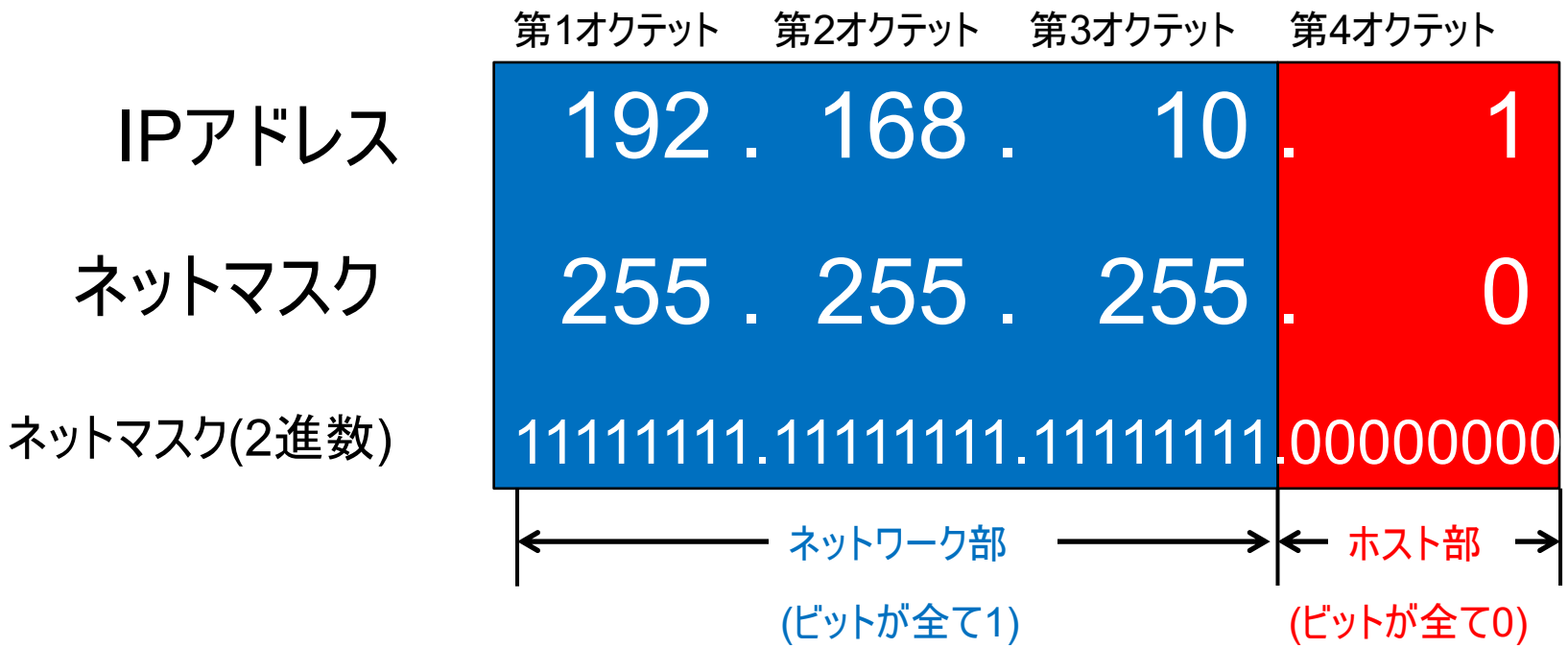
	8bit目	7bit目	6bit目	5bit目	4bit目	3bit目	2bit目	1bit目
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2進数	1	0	1	0	1	0	0	0
10進数	128	64	32	16	8	4	2	1
bitが1の桁を数える	128		32		8			

以上から、128+32+8=168となります。
※オクテットのビットが全て1(11111111)は255です。よく使うので覚えましょう。

IPアドレス　⑥ネットマスク

#LinuC学習中

IPアドレスはどのネットワークに所属するかを表す**ネットワーク部**とネットワーク内にある機器の番号である**ホスト部**から構成されます。ネットワーク部とホスト部の境界は**ネットマスク**で分かります。ネットマスクが2進数で1のビットはネットワーク部、0のビットはホスト部となります。



※ネットマスクは後で出てくる**サブネットマスク**と同じ意味で使われることがあります。

IPアドレス　⑦IPアドレスの使用可能範囲

#LinuC学習中

IPアドレスのホスト部のビットが全て0の場合を**ネットワークアドレス**と言い、ネットワーク自身を表します。ビットが全て1の場合を**ブロードキャストアドレス**と言い、ネットワーク内の全ての機器を表します。
この2つのアドレスは機器に割り当てができません。使用可能なIPアドレス数はホスト部の総数から2を引いた数となります。

	10進数	2進数	備考
IPアドレス	192.168.10.1	11000000.10101000.00001010.00000001	
ネットマスク	255.255.255.0	11111111.11111111.11111111.00000000	第4オクテットの8bitがホスト部
ネットワークアドレス	192.168.10.0	11000000.10101000.00001010.00000000	ホスト部のビットが全て0 IPアドレスとしては使用不可
IPアドレス 使用可能範囲	192.168.10.1 ～ 192.168.10.254	11000000.10101000.00001010.00000001～ 11000000.10101000.00001010.11111110	ネットワークアドレスの次からブロードキャストアドレスの前まで。 ホスト部の総数8bit=> 256-2=254個が使用可能
ブロードキャストアドレス	192.168.10.255	11000000.10101000.00001010.11111111	ホスト部のビットが全て1 IPアドレスとしては使用不可

IPアドレス　⑧クラス

#LinuC学習中

IPアドレスはICANNという組織が管理しており、勝手に使用することはできません。
ICANNはIPアドレスをA～Eまでの**クラス**を元に割り当てました。D,Eは特殊な例で使用されるので、ここではA～Cについて説明します。

クラス	IPアドレスの範囲(上:10進数　下:2進数)	ネットマスク(上:10進数　下:2進数)	備考
A	0.0.0.0～127.255.255.255	255.0.0.0	先頭が0なので0から始まる。 24bit使用可能。大規模ネットワークで使用。
	0xxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	11111111.00000000.00000000.00000000	
B	128.0.0.0～191.255.255.255	255.255.0.0	先頭が10なので128から始まる。 16bit使用可能。中規模ネットワークで使用。
	10xxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	11111111.11111111.00000000.00000000	
C	192.0.0.0～223.255.255.255	255.255.255.0	先頭が110なので192から始まる。 8bit使用可能。小規模ネットワークで使用。
	110xxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	11111111.11111111.11111111.00000000	

※「x」には0か1が入ります

IPアドレス　⑨プライベートIPアドレス

#LinuC学習中

IPアドレスは世界中で重複しないように割り当てる必要がありますが、IPアドレスの不足が見込まれたため、組織や家庭などの閉じられたネットワーク(LAN)内のみ使用可能な**プライベートIPアドレス**が定められました。
下表の範囲のIPアドレスは自由に使用することができます。プライベートIPアドレスはLAN内のルータで**グローバルIPアドレス**と相互変換されます。

クラス	IPアドレスの範囲(上:10進数　下:2進数)	ネットマスク(上:10進数　下:2進数)	備考
A	**10.0.0.0～10.255.255.255**	255.0.0.0	24bit使用可能
	00000110.xxxxxxxx.xxxxxxxx.xxxxxxxx	11111111.00000000.00000000.00000000	
B	**172.16.0.0～172.31.255.255**	255.240.0.0	20bit使用可能
	10101100.0001xxxx.xxxxxxxx.xxxxxxxx	11111111.1111**0000**.00000000.00000000	
C	**192.168.0.0～192.168.255.255**	255.255.0.0	16bit使用可能
	11000000.10101000.xxxxxxxx.xxxxxxxx	11111111.11111111.00000000.00000000	

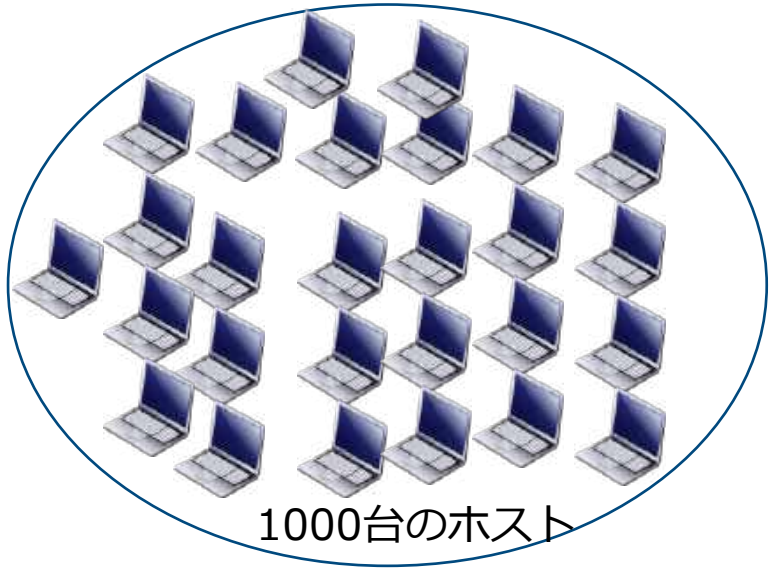
※「x」には0か1が入ります

IPアドレス　⑩クラスの問題点

#LinuC学習中

クラスによる割り当て方ではA～Cの3種類しかないため柔軟性に欠けます。組織内でIPアドレスが余っていても有効に活用することができず、逆に必要な組織にIPアドレスを回すことができません。IPアドレスが枯渇する原因でした。

クラスBの組織　65534台分のIPが使用可能



残り64534台分のIPが余る

クラスCの組織　254台分のIPが使用可能

あと46台分のIPが足りない

IPアドレス　⑪サブネット化、サブネットマスク

#LinuC学習中

IPアドレスの柔軟な運用のために、管理者がネットワークを分割する**サブネット化**が可能となりました。ホスト部のビットを切り出して、ネットワーク部として使用します。これを**サブネットマスク**と言います。
下記はホスト部の上位2ビットを使用して4つに分割した例です。

	IPアドレスの範囲(上:10進数　下:2進数)	サブネットマスク(上:10進数　下:2進数)	備考
クラスCのネットワーク	192.168.10.0～192.168.10.255	255.255.255.0	最大ホスト数 256-2=254
	11000000.10101000.00001010.**xxxxxxxx**	11111111.11111111.11111111.**00000000**	
割 4つのネットワークに分	192.168.10.**0**～192.168.10.63	255.255.255.192	最大ホスト数 64-2=62
	11000000.10101000.00001010.**00xxxxxx**	11111111.11111111.11111111.**11000000**	
	192.168.10.**64**～192.168.10.127	255.255.255.192	最大ホスト数 64-2=62
	11000000.10101000.00001010.**01xxxxxx**	11111111.11111111.11111111.**11000000**	
	192.168.10.**128**～192.168.10.191	255.255.255.192	最大ホスト数 64-2=62
	11000000.10101000.00001010.**10xxxxxx**	11111111.11111111.11111111.**11000000**	
	192.168.10.**192**～192.168.10.255	255.255.255.192	最大ホスト数 64-2=62
	11000000.10101000.00001010.**11xxxxxx**	11111111.11111111.11111111.**11000000**	

※「x」には0か1が入ります

IPアドレス　⑫CIDR表記

#LinuC学習中

IPアドレスの不足の対応するため、クラス分けを廃止して任意のビットでIPアドレスを配布する、**CIDR**(Classless Inter-Domain Routing)という仕組みが定められました。
サブネットマスクはIPアドレスと並記しますが、 **CIDR表記**ではIPアドレスの後に/(スラッシュ)とプレフィックス値で表現ができます。プレフィックス値はサブネットマスクを2進数にしたときに先頭から「1」が続く個数を示しています。どちらの表現も使われるので慣れておきましょう。

IPアドレス	192 . 168 . 10 . 1
サブネットマスク	255 . 255 . 255 . 192
CIDR表記	192 . 168 . 10 . 1/26

プレフィックス値

ネットワークのコマンドやファイル

#LinuC学習中

1．IPアドレス
　　ip, nmcli
2．疎通確認
　　ping
3．経路確認
　　traceroute, tracepath
4．ルーティング
　　ip route, nmcli
5．ホスト名
　　/etc/hostname, hostname, hostnamectl, nmcli
6．名前解決
　　/etc/hosts, /etc/resolv.conf, nmcli, /etc/nsswitch.conf, getent,
　　host, dig

1. IPアドレス　ip①　コマンド書式

#LinuC学習中

ipコマンドは、オブジェクトに対してコマンドを指定する形式で実行します。
各オブジェクトごとに指定可能なコマンドは異なります。
また、オブジェクトやコマンドは短縮して指定することも可能です。
設定は即時に反映されますが、再起動すると消えてしまいます。

ip オブジェクト コマンド [値]

オブジェクト ※()は省略形	説明
addr(a)	IPアドレスの確認や設定を行う
link(l)	ネットワークデバイスの確認や設定を行う
route(r)	ルーティングテーブルの確認や設定を行う
neighbor(n)	ARPキャッシュの表示

addrの コマンド ※()は省略形	説明
show(s)	全てのデバイスのIPアドレスの表示(省略可能) **ip addr show dev デバイス名** で指定したデバイスのみのIPアドレスを表示
add(a)	IPアドレスの追加
del(d)	IPアドレスの削除

1. IPアドレス　ip②　ip addr

ip addrで全てのデバイスのIPアドレスを表示した例です。

#LinuC学習中

```
[root@localhost ~]# ip addr
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 pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:bc:e5:37 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic enp0s3
       valid_lft 86142sec preferred_lft 86142sec
    inet6 fe80::a62c:98f1:95fb:cb9f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b8:fa:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.255.3/24 brd 192.168.255.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::8511:1e6d:8e4c:17ca/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:04:f1:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:04:f1:02 brd ff:ff:ff:ff:ff:ff
```

1. IPアドレス　ip③　ip addr show

#LinuC学習中

指定のデバイスのIPアドレスを表示した例です。
IPアドレスは**ネットワークインタフェースカード(NIC)**に紐づけられます。
enp0s8はLinuxが認識したネットワークインタフェースのデバイス名です。

ip addr show デバイス名

```
[root@localhost ~]# ip addr show enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b8:fa:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.255.3/24 brd 192.168.255.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::8511:1e6d:8e4c:17ca/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

1. IPアドレス　ip④　ループバックアドレス

#LinuC学習中

特別なIPアドレスとして**ループバックアドレス**があります。
論理的なネットワークインタフェースにループバックアドレスが割り当てられます。自身宛ての通信をするために使用され、ネットワーク機能のテストやサーバの動作確認などに使用されます。
デバイス名は**lo**です。IPアドレスは**127.0.0.1/8**が割り当てられます。

```
[root@localhost ~]# ip a s lo
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
```

1. IPアドレス　ip⑤　IPアドレスの追加と削除

#LinuC学習中

指定のデバイスにIPアドレスを追加・削除した例です。

<追加> ip addr add IPアドレス/プレフィックス値 dev デバイス名

<削除> ip addr del IPアドレス/プレフィックス値 dev デバイス名

```
[root@localhost ~]# ip addr add 192.168.255.10/24 dev enp0s8
[root@localhost ~]#
[root@localhost ~]# ip addr show enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b8:fa:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.255.3/24 brd 192.168.255.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet 192.168.255.10/24 scope global secondary enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::8511:1e6d:8e4c:17ca/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[root@localhost ~]#
[root@localhost ~]# ip addr del 192.168.255.10/24 dev enp0s8
[root@localhost ~]#
[root@localhost ~]# ip addr show enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b8:fa:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.255.3/24 brd 192.168.255.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::8511:1e6d:8e4c:17ca/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

1. IPアドレス　nmcli①　コマンド書式

#LinuC学習中

最近のLinuxではNetworkManagerサービスによってネットワークに関する様々な確認や設定が行えます。オブジェクトやコマンドは短縮して指定することも可能です。設定は永続的に維持されますが、即時に設定は反映されず、コネクション起動時に反映されます。

nmcli オブジェクト コマンド

オブジェクト ※()は省略形	コマンド ※()は省略形	説明
なし	-	デバイスの情報を一覧表示する
device(d)	show(s) デバイス名	指定したデバイス情報を表示
connection(c)	show(s) コネクション名	接続情報を表示
	modify(m) コネクション名 パラメータ	指定した接続を設定する
	up(u) コネクション名	接続を有効にする
	down(d) コネクション名	接続を無効にする

1. IPアドレス　nmcli②　デバイス情報の一覧表示

nmcliコマンドのみで各デバイスの情報を簡易的に一覧表示します。#LinuC学習中
IPアドレスも含まれます。

nmcli

```
[root@localhost ~]# nmcli
enp0s3: 接続済み to enp0s3
        "Intel 82540EM"
        ethernet (e1000), 08:00:27:BC:E5:37, hw, mtu 1500
        ip4 デフォルト
        inet4 10.0.2.15/24
        route4 0.0.0.0/0
        route4 10.0.2.0/24
        inet6 fe80::a62c:98f1:95fb:cb9f/64
        route6 fe80::/64
        route6 ff00::/8

enp0s8: 接続済み to enp0s8
        "Intel 82540EM"
        ethernet (e1000), 08:00:27:B8:FA:1A, hw, mtu 1500
        inet4 192.168.255.3/24
        inet6 fe80::8511:1e6d:8e4c:17ca/64
        route6 fe80::/64
        route6 ff00::/8
```

1. IPアドレス　nmcli③　各デバイス情報の表示

#LinuC学習中

nmcli device showでデバイスの情報を表示します。

nmcli device show デバイス名

```
[root@localhost ~]# nmcli device show enp0s8
GENERAL.DEVICE:                         enp0s8
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         08:00:27:B8:FA:1A
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (接続済み)
GENERAL.CONNECTION:                     enp0s8
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/2
WIRED-PROPERTIES.CARRIER:               オン
IP4.ADDRESS[1]:                         192.168.255.3/24
IP4.GATEWAY:                            --
IP4.DNS[1]:                             192.168.255.3
IP6.ADDRESS[1]:                         fe80::8511:1e6d:8e4c:17ca/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 101
IP6.ROUTE[2]:                           dst = ff00::/8, nh = ::, mt = 256, table=255
```

1. IPアドレス　nmcli④　IPアドレスの変更

#LinuC学習中

nmcli connection modifyのipv4.methodでIPアドレスを変更します。

nmcli connection modify コネクション名 ipv4.method manual ipv4.addresses IPアドレス/プレフィックス値

設定を反映させるために該当デバイスのコネクションをupします。

nmcli connection up コネクション名

```
[root@localhost ~]# nmcli connection modify enp0s8 ipv4.method manual ipv4.addresses 192.168.255.3/24
[root@localhost ~]#
[root@localhost ~]# nmcli connection up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス: /org/freedesktop/NetworkManager/ActiveConnection/5)
[root@localhost ~]#
[root@localhost ~]# ip a s dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b8:fa:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.255.3/24 brd 192.168.255.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::8511:1e6d:8e4c:17ca/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

1. IPアドレス　nmcli⑤　IP＋DGW＋参照先DNSの設定

#LinuC学習中

nmcliコマンドでIPアドレスの他にも後のページで解説する**デフォルトゲートウェイ**と**参照先DNS**を一度に設定することが可能です。

nmcli connection modify コネクション名 ipv4.method manual ipv4.addresses IPアドレス/プレフィックス値 ipv4.gateway デフォルトゲートウェイ ipv4.dns 参照先DNSのIPアドレス

```
[root@level2 ~]# nmcli connection modify enp0s8 ipv4.method manual ipv4.addresses 192.168
.255.3/24 ipv4.gateway 192.168.255.1 ipv4.dns 192.168.255.2
[root@level2 ~]#
[root@level2 ~]# nmcli c up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス: /org/freedesktop/NetworkManag
er/ActiveConnection/5)
[root@level2 ~]#
[root@level2 ~]# nmcli c show enp0s8 | grep -e ipv4.address -e ipv4.gateway -e ipv4.dns
ipv4.dns:                               192.168.255.2
ipv4.dns-search:                        --
ipv4.dns-options:                       ""
ipv4.dns-priority:                      10
ipv4.addresses:                         192.168.255.3/24
ipv4.gateway:                           192.168.255.1
[root@level2 ~]#
```

2. 疎通確認　ping①

#LinuC学習中

pingコマンドは指定されたホストに**ICMP**パケットを送り結果を表示します。
これを疎通確認と言い、宛先ホストの状態確認によく使われます。
宛先にホスト名を指定した場合は名前解決をします。
-cオプションの指定がない時はCtrl+Cキーが押されるまで連続実行されます。

ping オプション ホスト名またはIPアドレス

オプション	説明
-c 回数	指定した回数ICMPパケットを送信する
-i 間隔	指定した間隔(秒)ごとにICMPパケットを送信する(デフォルトは1秒)

pingコマンドはIPv4環境で使用されますが、IPv6環境では**ping6**コマンドを使います。

2. 疎通確認　ping②

#LinuC学習中

下記の画面は**lpi.or.jp**にpingを6回打って6回応答があった状態です。
その後、Ctrl+Cを押して中断しています。

ping lpi.or.jp

名前解決(ホスト名→IPアドレス)

```
[root@localhost ~]# ping lpi.or.jp
PING lpi.or.jp (219.94.215.12) 56(84) bytes of data.
64 bytes from 12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=1 ttl=54 time=11.3 ms
64 bytes from 12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=2 ttl=54 time=11.5 ms
64 bytes from 12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=3 ttl=54 time=11.6 ms
64 bytes from 12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=4 ttl=54 time=11.0 ms
64 bytes from 12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=5 ttl=54 time=11.1 ms
64 byte                .219.static.www232b.sakura.ne.jp (219.94.215.12): icmp_seq=6 ttl=54 time=11.4 ms
^C
--- lpi.or.jp ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5016ms
rtt min/avg/max/mdev = 11.050/11.365/11.622/0.220 ms
```

Ctrl+Cで中断

6回送信して、6回受信しているのでパケットロスは0%　全体で5016msかかりました。
往復時間(rtt)は、最小(min)11.050ms/平均(avg)11.365ms/最大(max)11.622msです。
標準偏差(mdev)は0.220で、ばらつき具合(0に近いほどばらつき少ない)を示します。
往復時間の多くは平均の11.365±0.220に収まります。

3. 経路確認　traceroute①

#LinuC学習中

tracerouteコマンドは指定されたホストまでの経路のルータに順番にパケットを送り、宛先ホストまでの経路のどこに障害があるのかを確認するために使われます。これを経路確認と言います。ホスト名の場合は名前解決をします。

traceroute オプション ホスト名またはIPアドレス

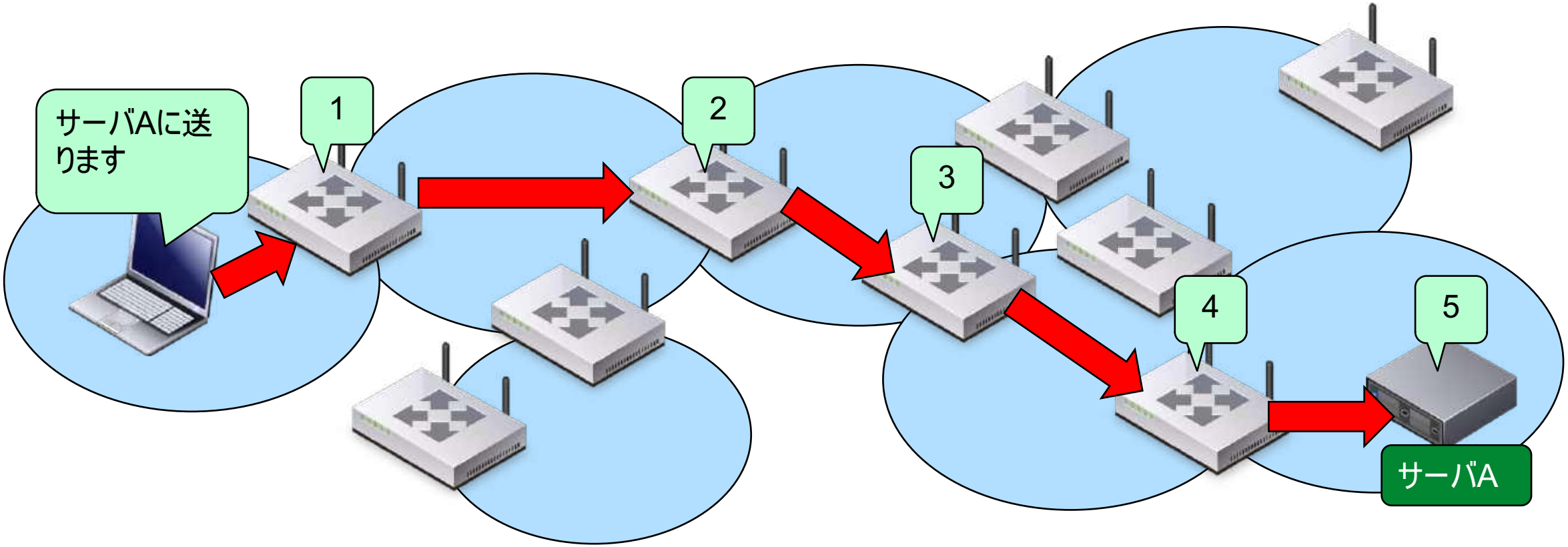
オプション	説明
-U	UDPパケットを送信する(デフォルト)
-I	ICMPパケットを送信する

tracerouteコマンドはIPv4環境で使用されますが、IPv6環境では**traceroute6**コマンドを使います。

3. 経路確認　traceroute②

#LinuC学習中

tracerouteは下図のように宛先ホストまでに経由するルータそれぞれにパケットを送って応答があることを確認します。途中で応答がなければそこで障害が発生していると予測します。



3. 経路確認　traceroute③

#LinuC学習中

traceroute lpi.or.jpの実行結果です。
lpi.or.jpまで11のルータを通って目的地である12番目のホストまで行っていることが分かります。それぞれのルータに3回パケットを送信して往復時間を測っています。ルータのセキュリティ対策により応答パケットを返さないルータは＊＊＊が表示されます。

```
[root@localhost ~]# traceroute lpi.or.jp
traceroute to lpi.or.jp (219.94.215.12), 30 hops max, 60 byte packets
 1  gateway (192.168.1.254)  2.070 ms  2.026 ms  2.003 ms
 2  fpa446da01.tkyc612.ap.nuro.jp (164.70.218.1)  10.964 ms  11.252 ms  11.446 ms
 3  118.238.197.104 (118.238.197.104)  5.135 ms  5.328 ms  5.325 ms
 4  39.110.252.105 (39.110.252.105)  5.592 ms  5.578 ms  6.511 ms
 5  202.213.193.67 (202.213.193.67)  5.832 ms  6.073 ms  6.640 ms
 6  tkort3-as2527.bb.sakura.ad.jp (157.17.131.17)  6.228 ms  4.044 ms  4.299 ms
 7  tkort4-ort3.bb.sakura.ad.jp (157.17.130.122)  4.497 ms  8.044 ms  8.080 ms
 8  oshrt1-tkort4.bb.sakura.ad.jp (157.17.131.102)  11.372 ms  11.611 ms  11.565 ms
 9  osnrt201s-hrt1-1.bb.sakura.ad.jp (157.17.146.82)  11.843 ms osnrt1s-hrt1.bb.sakura.ad.jp (
157.17.146.30)  12.078 ms osnrt201s-hrt1-3.bb.sakura.ad.jp (157.17.146.90)  11.955 ms
10  osnrt101b-nrt201s-2.bb.sakura.ad.jp (157.17.146.2)  11.930 ms osnrt101b-nrt1s.bb.sakura.ad
.jp (157.17.146.102)  11.915 ms osnrt101b-nrt201s.bb.sakura.ad.jp (157.17.146.214)  12.115 ms
11  osnrt123e-nrt102b.bb.sakura.ad.jp (157.17.150.78)  12.383 ms  12.593 ms osnrt123e-nrt101b.
bb.sakura.ad.jp (157.17.149.78)  11.336 ms
12  12.215.94.219.static.www232b.sakura.ne.jp (219.94.215.12)  11.243 ms  11.532 ms  11.512 ms
```

応答パケットを
返さないルータ

```
6   202.213
7   * * *
8   101.102
9   * * *
```

3. 経路確認　tracepath①

#LinuC学習中

tracepathコマンドはtracerouteと同じく指定されたホストまでの経路の機器にパケットを送ります。tracerouteと異なるのはPMTU値を表示することです。MTUとは機器が一度に送信できる最大のデータ量で、**PathMTU**値は経路全体で適用されるMTU値です。

tracepath オプション ホスト名またはIPアドレス

オプション	説明
-c 回数	指定した回数ICMPパケットを送信する
-i 間隔	指定した間隔(秒)ごとにICMPパケットを送信する(デフォルトは1秒)

tracepathコマンドはIPv4環境で使用されますが、IPv6環境では**tracepath6**コマンドを使います。

3. 経路確認　tracepath②

#LinuC学習中

tracepath lpi.or.jpの実行結果です。
lpi.or.jpまで11のルータを通って目的地である12番目のホストまで行っていることが分かります。それぞれのルータに1回パケットを送信して往復時間を測っています。ルータのセキュリティ対策により応答パケットを返さないルータはno replyが表示されます。

```
[root@localhost ~]# tracepath lpi.or.jp
 1?: [LOCALHOST]                                         pmtu 1500
 1:  gateway                                              1.165ms
 1:  gateway                                              1.160ms
 2:  fpa446da01.tkyc612.ap.nuro.jp                        4.416ms
 3:  118.238.197.104                                      3.758ms
 4:  39.110.252.105                                       4.646ms
 5:  202.213.193.35                                       4.387ms
 6:  tkort3-as2527.bb.sakura.ad.jp                        4.386ms
 7:  tkort4-ort3.bb.sakura.ad.jp                          6.839ms
 8:  oshrt1-tkort4.bb.sakura.ad.jp                       11.336ms asymm  7
 9:  osnrt1s-hrt1.bb.sakura.ad.jp                        11.265ms asymm  8
10:  osnrt102b-nrt1s-2.bb.sakura.ad.jp                   11.329ms asymm  9
11:  osnrt123e-nrt101b.bb.sakura.ad.jp                   11.724ms asymm 10
12:  12.215.94.219.static.www232b.sakura.ne.jp           11.439ms reached
     Resume: pmtu 1500 hops 12 back 11
```

PMTU値が1500

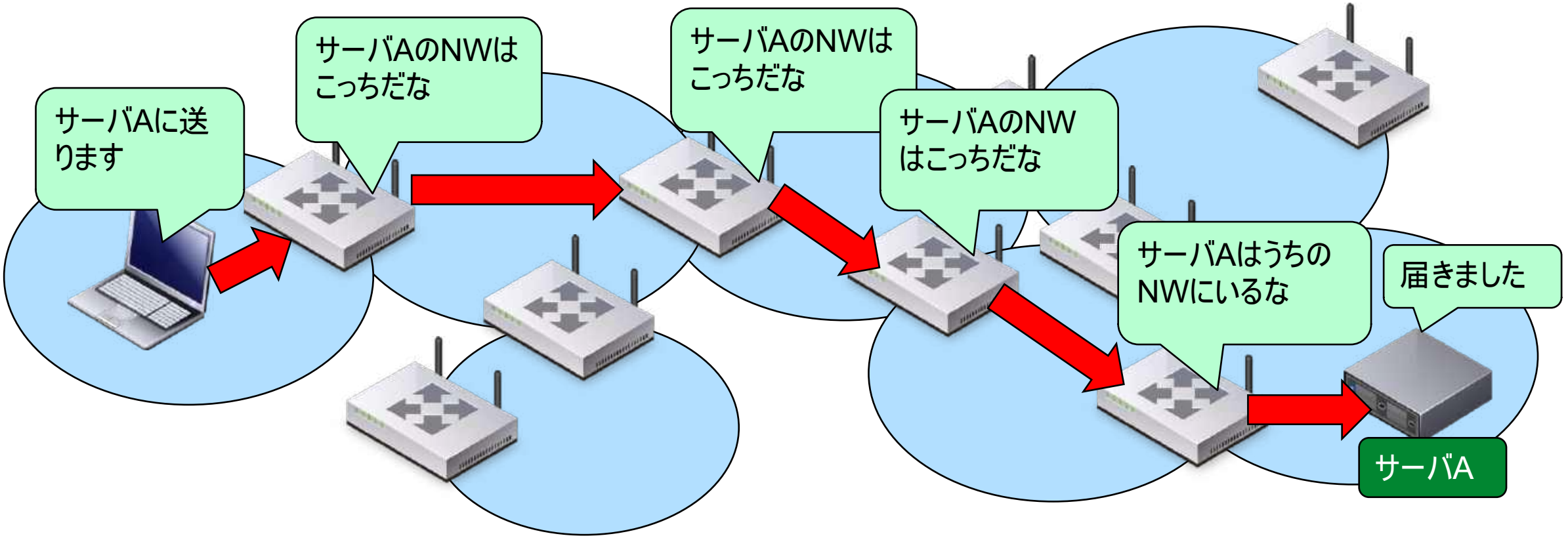
応答パケットを
返さないルータ

```
7:   no reply
8:   101.102.20
9:   no reply
```

4. ルーティング　ルーティングとは

#LinuC学習中

ルーティングとは宛先ホストまでの経路を制御することをいいます。
ホストやルーターにはルーティングテーブルというルーティングのための情報が格納されています。ルーティングテーブルはパケットを送信したり、受信したパケットを転送するときに参照されて送信先を決めます。



4. ルーティング　ip route①　ルーティングテーブルの表示

ルーティングテーブルを表示するコマンドにip routeがあります。 #LinuC学習中
他にも、route, netstat –rがありますが最近では非推奨となっています。

ip route

```
[root@localhost ~]# ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.2.0/24 via 192.168.255.1 dev enp0s8 proto static metric 101
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

route

```
[root@localhost ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
192.168.2.0     192.168.255.1   255.255.255.0   UG    101    0        0 enp0s8
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
192.168.255.0   0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
```

netstat -r

```
[root@localhost ~]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         gateway         0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
192.168.2.0     192.168.255.1   255.255.255.0   UG        0 0          0 enp0s8
192.168.122.0   0.0.0.0         255.255.255.0   U         0 0          0 virbr0
192.168.255.0   0.0.0.0         255.255.255.0   U         0 0          0 enp0s8
```

4. ルーティング　ip route②　ルーティングテーブルの見方

#LinuC学習中

ルーティングテーブルには**宛先ネットワーク**に行くための、
次に経由する機器である**ゲートウェイ (via)**や**送信デバイス(dev)**が記載されています。

192.168.2.0/24 via 192.168.255.1 dev enp0s8 proto static metric 101

宛先ネットワーク　ゲートウェイ　送信デバイス

```
[root@localhost ~]# ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.2.0/24 via 192.168.255.1 dev enp0s8 proto static metric 101
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

4. ルーティング　ip route③　ルーティングテーブルの見方

#LinuC学習中

宛先ネットワークに**default**と記載されているのは**デフォルトルート**と言い、**他に宛先がない場合の経路**となります。デフォルトルートのゲートウェイは**デフォルトゲートウェイ**と呼ばれます。

default via 10.0.2.2 dev enp0s3 proto dhcp metric 100

デフォルトルート　デフォルトゲートウェイ　送信デバイス

```
[root@localhost ~]# ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.2.0/24 via 192.168.255.1 dev enp0s8 proto static metric 101
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

4. ルーティング　ip route④　ルートの追加と削除

#LinuC学習中

ip routeでルーティングテーブルに経路を追加、削除できます。
なお、即時に反映されますが、再起動すると戻ります。

ip route add 宛先ネットワーク via ゲートウェイ dev デバイス

```
[root@localhost ~]# ip route add 192.168.2.0/24 via 192.168.255.1 dev enp0s8
[root@localhost ~]#
[root@localhost ~]# ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.2.0/24 via 192.168.255.1 dev enp0s8
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

オプション	説明
add	ルーティングテーブルに経路を追加する。
del	ルーティングテーブルから経路を削除する。via以降は省略可能。

4. ルーティング　ip route⑤　ルートの追加と削除

#LinuC学習中

ip route del 宛先ネットワーク (via ゲートウェイ dev デバイス)

省略可

```
[root@localhost ~]# ip route del 192.168.2.0/24
[root@localhost ~]#
[root@localhost ~]# ip r
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

宛先ネットワークを**default**とするとデフォルトルートを追加できます。

ip route add default via 192.168.1.254

```
[root@localhost ~]# ip route add default via 192.168.255.1 dev enp0s8
[root@localhost ~]#
[root@localhost ~]# ip r
default via 192.168.255.1 dev enp0s8
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
```

4. ルーティング　nmcli①　ルートの追加と削除

#LinuC学習中

nmcliでのルートの追加は+ipv4.routes、削除は-ipv4.routesを使用します。

nmcli connection modify コネクション名 +ipv4.routes “宛先ネットワーク ゲートウェイ”

nmcli connection modify コネクション名 -ipv4.routes “宛先ネットワーク ゲートウェイ”

追加したルートは/etc/sysconfig/network-scripts/route-コネクション名に永続的に保存されます。

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/route-enp0s8
ADDRESS0=192.168.2.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.255.1
```

設定を反映させるために該当デバイスのコネクションをupします。

nmcli connection up コネクション名

```
[root@localhost ~]# nmcli connection up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス:
/org/freedesktop/NetworkManager/ActiveConnection/8)
```

4. ルーティング　nmcli②　ルートの追加と削除

#LinuC学習中

下記は宛先ネットワークに192.168.2.0/24、
ゲートウェイに192.168.255.1のルートを追加、削除した例です。

```
[root@localhost ~]# nmcli connection modify enp0s8 +ipv4.routes "192.168.2.0/24 192.168.255.1"
[root@localhost ~]#
[root@localhost ~]# nmcli c up enp0s8
接続が正常にアクティベートされました（D-Bus アクティブパス: /org/freedesktop/NetworkManager/Ac
tiveConnection/6）
[root@localhost ~]# ip r
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
default via 192.168.255.1 dev enp0s8 proto static metric 101
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.2.0/24 via 192.168.255.1 dev enp0s8 proto static metric 101
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
[root@localhost ~]#
[root@localhost ~]# nmcli connection modify enp0s8 -ipv4.routes "192.168.2.0/24 192.168.255.1"

[root@localhost ~]# nmcli c up enp0s8
接続が正常にアクティベートされました（D-Bus アクティブパス: /org/freedesktop/NetworkManager/Ac
tiveConnection/7）
[root@localhost ~]# ip r
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
default via 192.168.255.1 dev enp0s8 proto static metric 101
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
[root@localhost ~]#
```

新しいルートが追加されている

192.168.2.0/24の宛先へのルートが削除された

4. ルーティング　nmcli③　デフォルトゲートウェイの設定

#LinuC学習中

nmcliでデフォルトゲートウェイの設定をするには、
ipv4.gatewayを使用します。

nmcli connection modify コネクション名 ipv4.gateway デフォルトゲートウェイ

設定したデフォルトゲートウェイは/etc/sysconfig/network-scripts/ifcfg-コネクション名に永続的に保存されます。

```
[root@localhost ~]# grep GATEWAY /etc/sysconfig/network-scripts/ifcfg-enp0s8
GATEWAY=192.168.255.2
```

設定を反映させるために該当デバイスのコネクションをupします。

nmcli connection up コネクション名

```
[root@localhost ~]# nmcli connection up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス:
/org/freedesktop/NetworkManager/ActiveConnection/8)
```

4. ルーティング　nmcli④　デフォルトゲートウェイの設定

#LinuC学習中

下記はデフォルトゲートウェイに192.168.255.1を設定した例です。

```
[root@localhost ~]# nmcli connection modify enp0s8 ipv4.gateway 192.168.255.1
[root@localhost ~]#
[root@localhost ~]# grep GATEWAY /etc/sysconfig/network-scripts/ifcfg-enp0s8
GATEWAY=192.168.255.1
[root@localhost ~]#
[root@localhost ~]# nmcli c up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス: /org/freedesktop/NetworkManager/Ac
tiveConnection/8)
[root@localhost ~]# ip r
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
default via 192.168.255.1 dev enp0s8 proto static metric 101
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
192.168.255.0/24 dev enp0s8 proto kernel scope link src 192.168.255.3 metric 101
[root@localhost ~]#
```

新しいデフォルトゲートウェイが追加されている

5. ホスト名　ホスト名とは？

#LinuC学習中

コンピュータの名前を**ホスト名**、コンピュータが所属する
組織の名前を**ドメイン名**と呼びます。
また、ホスト名＋ドメイン名全体を指して**FQDN（Fully Qualified Domain Name）**と呼びます。
※FQDNのことを指してホスト名ということもあります。

internous.co.jp.ドメインに所属するホスト(コンピュータ)の**www**という意味

5. ホスト名　ホスト名のファイルとコマンド①

#LinuC学習中

Linuxではホスト名が格納されているファイルやホスト名を表示、設定するコマンドがあります。

ホスト名が格納されているファイル

/etc/hostname

ホスト名を表示するコマンド

hostname
nmcli general hostname

```
[root@localhost ~]# cat /etc/hostname
localhost.localdomain
[root@localhost ~]#
[root@localhost ~]# hostname
localhost.localdomain
[root@localhost ~]#
[root@localhost ~]# nmcli general hostname
localhost.localdomain
```

#LinuC学習中

ホスト名を変更するコマンド

hostnamectl set-hostname ホスト名

```
[root@localhost ~]# hostnamectl set-hostname level1.linuc.or.jp
[root@localhost ~]#
[root@localhost ~]# hostname
level1.linuc.or.jp
[root@localhost ~]# cat /etc/hostname
level1.linuc.or.jp
[root@localhost ~]# su -
最終ログイン: 2020/09/19 (土) 18:17:59 JST日時 pts/0
[root@level1 ~]#
```

新しいシェルでホスト名が変更されている

nmcli general hostname ホスト名

```
[root@localhost ~]# nmcli general hostname level2.linuc.or.jp
[root@localhost ~]# hostname
level2.linuc.or.jp
[root@localhost ~]# cat /etc/hostname
level2.linuc.or.jp
[root@localhost ~]# su -
最終ログイン: 2020/09/20 (日) 08:57:59 JST日時 pts/0
[root@level2 ~]#
```

新しいシェルでホスト名が変更されている

6. 名前解決　名前解決の方法①　/etc/hostsを参照

#LinuC学習中

名前解決とはファイルまたはサーバーを参照して、ホスト名とIPアドレスを相互に変換することです。ホスト名からIPアドレスを求める**正引き**と、IPアドレスからホスト名を求める**逆引き**があります。ホスト内部でのファイルを参照する名前解決とDNSサーバーを参照する名前解決があります。

ホスト名とIPアドレスの紐づけが格納されているファイル

/etc/hosts

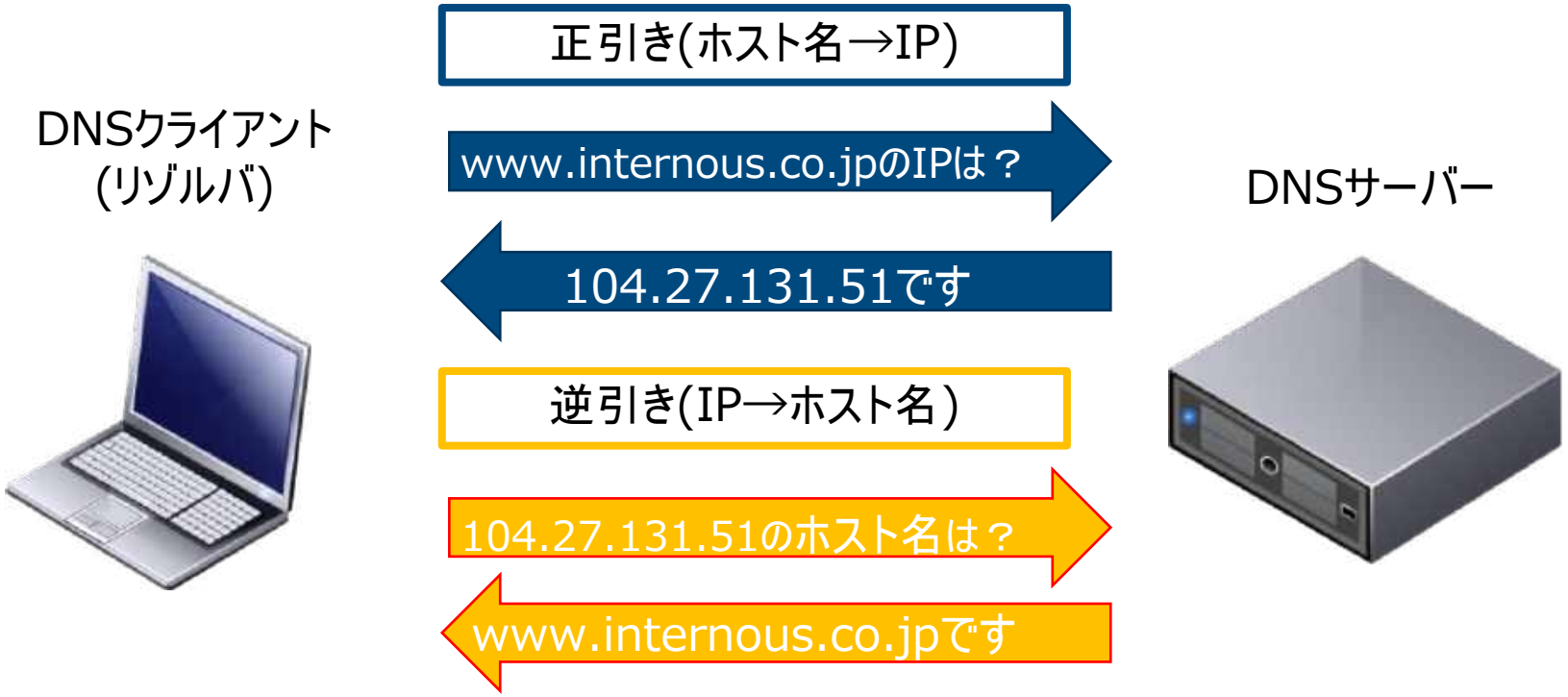
```
[root@localhost ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

/etc/hostsファイルの書式は、1行ごとにIPアドレスの後にスペースで区切って紐づけるホスト名を書きます。スペース区切りで複数のホスト名をつけることができます。
1行目は127.0.0.1のループバックアドレスにホスト名として、
localhost,localhost.localdomain,localhost4,localhost4.localdomain4の4つのホスト名が紐づけられていることが分かります。

6. 名前解決　名前解決の方法②　DNSサーバーを参照

#LinuC学習中

/etc/hostsは少数の名前解決するには便利ですが大量の名前解決をする仕組みとしてはファイルの更新が頻繁になりすぎて不便です。
/etc/hostsを参照する以外に、インターネット上のホストのような大量の名前解決が可能なDNSサーバーに名前解決を依頼する方法があります。名前解決のプログラムのことを**リゾルバ**といいます。



6. 名前解決　/etc/resolv.conf　DNS参照先のファイル

#LinuC学習中

/etc/resolv.confはDNSサーバーを利用した名前解決時にリゾルバに参照されるファイルです。

/etc/resolv.conf

```
[root@localhost ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search linuc.or.jp
nameserver 192.168.1.254
nameserver 192.168.255.3
```

searchはホスト名のみの名前解決時に自動で付加するドメイン名となり、通常自分の所属するドメイン名となります。
nameserverは問い合わせ先となるDNSサーバーのIPアドレスを記載します。複数行ある場合は1番上のDNSサーバーから順に問い合わせます。応答がなかった場合は次行のDNSサーバーに問い合わせます。

6. 名前解決　nmcli① 参照先DNSの設定

#LinuC学習中

nmcliでデバイスごとに参照先DNSの設定します。
設定した参照先DNSは**/etc/sysconfig/network-scripts/ifcfg-コネクション名**に永続的に保存されます。

nmcli connection modify コネクション名 ipv4.dns 参照先DNSのIPアドレス

```
[root@localhost ~]# nmcli connection modify enp0s8 ipv4.dns 192.168.255.5
[root@localhost ~]#
[root@localhost ~]# nmcli c up enp0s8
接続が正常にアクティベートされました（D-Bus アクティブパス: /org/freedeskto
[root@localhost ~]#
[root@localhost ~]# nmcli d show enp0s8 | grep IP4.DNS
IP4.DNS[1]:                             192.168.255.5
[root@localhost ~]#
[root@localhost ~]# grep DNS1 /etc/sysconfig/network-scripts/ifcfg-enp0s8
DNS1=192.168.255.5
[root@localhost ~]#
[root@localhost ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search linuc.or.jp
nameserver 192.168.1.254
nameserver 192.168.255.5
[root@localhost ~]#
```

6. 名前解決　nmcli② 参照先DNSの優先順位の変更

#LinuC学習中

複数のデバイスが存在する際に、DNSの名前解決に優先するデバイスをnmcliコマンドで決めます。優先する方に高いpriority値を設定します。

nmcli connection modify デバイス名 ipv4.dns-priority priority値

```
[root@localhost ~]# nmcli connection modify enp0s8 ipv4.dns-priority 10
[root@localhost ~]#
[root@localhost ~]# nmcli c show enp0s8 | grep ipv4.dns-priority
ipv4.dns-priority:                      10
[root@localhost ~]#
[root@localhost ~]# nmcli c show enp0s3 | grep ipv4.dns-priority
ipv4.dns-priority:                      0
[root@localhost ~]#
[root@localhost ~]# nmcli c up enp0s8
接続が正常にアクティベートされました (D-Bus アクティブパス: /org/freedesktop/
NetworkManager/ActiveConnection/10)
[root@localhost ~]#
[root@localhost ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search linuc.or.jp
nameserver 192.168.255.5
nameserver 192.168.1.254
```

priority値が高いenp0s8のIPアドレスが先

6. 名前解決　/etc/nsswitch.conf

#LinuC学習中

名前解決の順序を決めるファイルに**/etc/nsswitch.conf**があります。**hosts:**の後に名前解決をする順序を書きます。
デフォルトでは先に/etc/hostsを参照する**files**、次にDNSサーバーを参照する**dns**が記載されています。

/etc/nsswitch.conf

```
[root@localhost ~]# grep hosts /etc/nsswitch.conf
#hosts:      db files nisplus nis dns
hosts:       files dns
```

6. 名前解決　getent

#LinuC学習中

getentは/etc/nsswitch.confに記載されているライブラリから値を取得するコマンドです。データベースに**hosts**を指定することで名前解決の確認に使用できます。

getent データベース 値

getent hosts lpi.or.jp

```
[root@localhost ~]# getent hosts lpi.or.jp
219.94.215.12   lpi.or.jp
```

6. 名前解決　host① コマンド書式

#LinuC学習中

hostコマンドは名前解決の確認に使われます。シンプルに結果が表示されるので名前解決の可否が分かりやすいことが特徴です。参照先DNSを指定しないときは/etc/resolv.confのnameserverの先頭行から問い合わせます。引数にIPアドレスを指定した場合は逆引きを行います。

host [オプション] ホスト名 [参照先DNS]

オプション	説明
-t 検索タイプ	検索タイプを指定して問い合わせる
-v	詳細な出力をします

-t 検索タイプ	説明
a	IPアドレス(省略時のデフォルト)
any	すべての情報
mx	メールサーバ情報
ns	ネームサーバ情報
soa	ゾーン情報
txt	テキスト情報

6. 名前解決　host②　使用例

#LinuC学習中

host lpi.or.jp

```
[root@localhost ~]# host lpi.or.jp
lpi.or.jp has address 219.94.215.12
lpi.or.jp mail is handled by 10 lpi.sakura.ne.jp.
```

回答結果
lpi.or.jpのIPアドレス

host –t mx lpi.or.jp　(検索タイプにmxを指定)

```
[root@localhost ~]# host -t mx lpi.or.jp
lpi.or.jp mail is handled by 10 lpi.sakura.ne.jp.
```

回答結果
lpi.or.jpのメールサーバーのホスト名

host –v lpi.or.jp　(詳細確認)

```
[root@localhost ~]# host -v lpi.or.jp
Trying "lpi.or.jp"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46969
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;lpi.or.jp.                     IN      A

;; ANSWER SECTION:
lpi.or.jp.              141     IN      A       219.94.215.12

Received 43 bytes from 192.168.1.254#53 in 7 ms
Trying "lpi.or.jp"
```

Aレコード
ホストに紐づけられたIPアドレスを格納

回答結果
lpi.or.jpのIPアドレス

6. 名前解決　dig①　コマンド書式

#LinuC学習中

digコマンドは名前解決の確認に使われます。名前解決の結果を詳細に確認したり、名前解決ができない原因を調査するために使用されます。
参照先DNSを指定しないときは/etc/resolv.confのnameserverの先頭行から問い合わせます。

dig オプション [@参照先DNS] ホスト名 [検索タイプ]

オプション	説明
-t 検索タイプ	検索タイプを指定して問い合わせる
-x	逆引きを行う

検索タイプ	説明
a	IPアドレス(省略時のデフォルト)
any	すべての情報
mx	メールサーバ情報
ns	ネームサーバ情報
soa	ゾーン情報
txt	テキスト情報

6. 名前解決　dig②　使用例

#LinuC学習中

dig lpi.or.jp

```
[root@localhost ~]# dig lpi.or.jp

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> lpi.or.jp
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36674
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;lpi.or.jp.                     IN      A

;; ANSWER SECTION:
lpi.or.jp.              264     IN      A       219.94.215.12

;; Query time: 6 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: 日　9月 20 11:10:58 JST 2020
;; MSG SIZE  rcvd: 43
```

Aレコード
ホストに紐づけられたIPアドレスを格納

回答結果
lpi.or.jpのIPアドレス

参照先DNSサーバーのIPアドレス

6. 名前解決　dig③　使用例

#LinuC学習中

dig lpi.or.jp –t mx　(検索タイプにmxを指定)

```
[root@localhost ~]# dig lpi.or.jp -t mx

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> lpi.or.jp -t mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51668
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 8

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;lpi.or.jp.                     IN      MX

;; ANSWER SECTION:
lpi.or.jp.              300     IN      MX      10 lpi.sakura.ne.jp.
```

MXレコード
ドメインのメールサーバー
のホスト名を格納

回答結果
lpi.or.jpドメイン
のメールサー
バーのホスト名