ネットワークとサーバによる ネットワークサービス提供

名古屋大学 情報基盤センター
情報基盤ネットワーク研究部門
基盤ネットワーク研究グループ
嶋田 創

なぜネットワークとサービスの話をここで?

- Local/Wide Areaなネットワーク(LAN/WAN)が関連しないセキュリティ関連の話が少ないため

  →一旦ざっくりとした説明をしておいた方が説明が楽

  →すでに皆さんもスマホを介して多数恩恵を浴びていると思うので、ざっくりとした説明でも知っておいた方が良い(自分で考えるための土台)
- もちろん、ネットワークを介さないセキュリティ話もある(主にソーシャルエンジニアリングの方面)
  - 例: 関係者のフリをして電話でパスワードや暗証番号(金融関連)を聞き出そうとした
  - 例: パスワードを物理的な方向で入手や推測
    - 入力の様子をのぞき見
    - キーの入力音や入力部を触った跡(赤外線等も含む)から推測

概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

最近のサイバー攻撃の動向

- R7年の警察庁のレポート[1]
  - ランサムウェア被害が高い水準で続く
    - 2019年から2021年で5-6倍に増えたまま減らない
  - フィッシング報告とクレジットカード被害も激増中
    - 2018年までは200億円前後だった番号不正利用が2023年は500億超に
  - もちろん、警察に上がってこない被害は計上されていない
- ニュースサイト[2]上も毎日のように新たな被害話
  - トップページの大項目に「個人情報漏洩」「不正アクセス」「ランサムウェア」のカテゴリができるぐらい
  - IPAの情報セキュリティ10大脅威2026[3]もランサムウェア、(内部不正による)情報漏洩、標的型攻撃(不正アクセス)が連続ラインクイン
    - 2026年にはAI利用(都度利用、エージェント利用)のリスクが新規追加

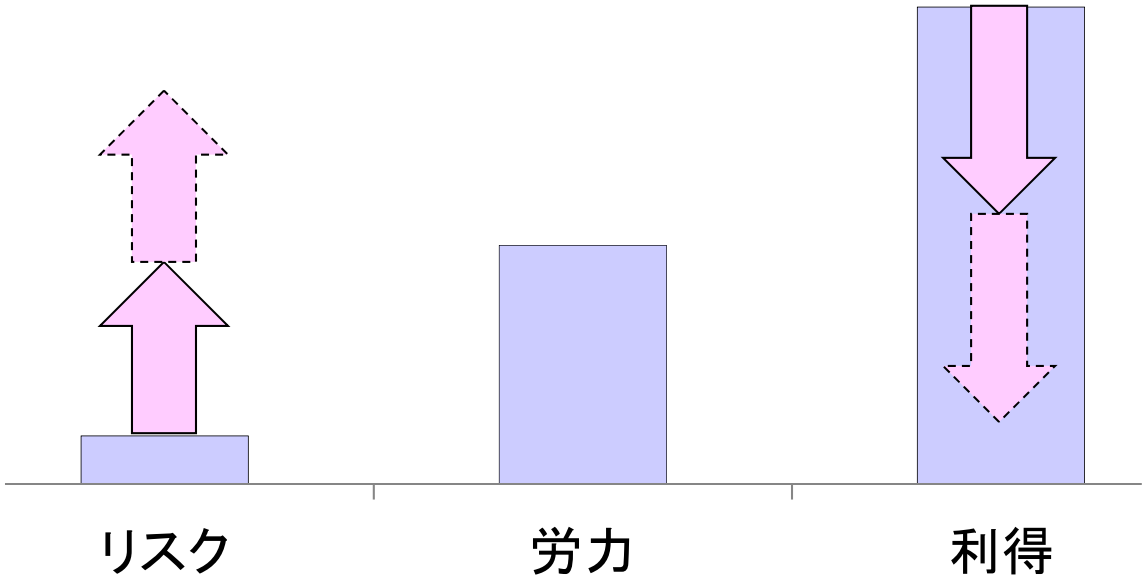
[1] https://www.npa.go.jp/publications/statistics/cybersecurity/
[2] https://scan.netsecurity.ne.jp/
[3] https://www.ipa.go.jp/security/10threats/10threats2026.html

なんで改善されていかないの？

- 攻撃者側の近況
  - だいぶ前からサイバー攻撃がビスネスとなっており、引き続きビジネス拡大している
  - ならずもの国家がサイバー攻撃が有用であることを強く認識した
- 被害者側の近況
  - 攻撃対象となる情報システムや情報サービス利用が増えた
    - DXで利便性や効率は上がったが、攻撃面が増えてしまった
    - さらに言うと、システムが連携して攻撃面や被害範囲が増える
  - 一方で、人や組織の意識の改善はゆっくりとしか進まない
    - 地震/火災/労災などの対策と比べてまだ意識は低い
    - が、労災なども意識を上げるのに時間をかけて実施してきたので、すぐに上がらないのはしかたない所はある
    - 地道に意識を改善していくしかない
    - 個人的には、災害大国日本は過去からの災害対策の意識改善をどうやってきたかを参考にすればサイバー攻撃対策の意識改革も進みそう

相変わらず攻撃者側に取ってリスク利得比が大きい(1/2)

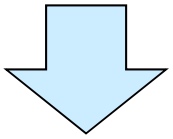
- 現在のサイバー犯罪はリスクに対して利得が大きすぎ
- リスクを上げて利得を減らす必要がある
- 現実は、サイバー犯罪は世界規模なので、リスクを上げれない国家において野放しの状態が続く
  - というか、サイバー犯罪に国が関わっている事例も増加
    - 「地政学的リスクに起因するサイバー攻撃」な話



この図はいつか
変えたい(変わって
欲しい)のだが、
未だに変える状況に
達していないのが残念

相変わらず攻撃者側に取ってリスク利得比が大きい(2/2)

- サイバー攻撃を行う犯罪者は随時新たな攻撃を開発していて頭良い
- 疑問: その攻撃手段への発想力を活かせば高収入で社会的地位の高い職業につけるのでは?



- A: 金になるからサイバー犯罪を行う
  - 普通の職業では一生かかっても稼げない金が手にはいるなら?
    - 所属する国によっては犯罪で得た金の価値が相対的に大きくなる
    - (サイバー)犯罪に対する追跡が緩い国ではなおさら
  - 金になる情報の代表: クレジットカード情報、オンラインバンキング決済情報、企業秘密、DDoS攻撃(脅迫用)、乗っ取ったコンピュータ(踏み台、仮想通貨発掘)、サーバやPCのデータ(脅迫用)、など
  - というか、サイバー犯罪組織が企業化(国営化)している

最近では、どう見ても国家がバックについている攻撃グループも多い

- こんな多い[1][2]

Iran [edit]

- Charming Kitten (also known as APT35)
- Elfin Team (also known as APT33)
- Helix Kitten (also known as APT34)
- Pioneer Kitten[69]
- Remix Kitten (also known as APT39, ITG07, or Chafer)[70][71]

North Korea [edit]

- Kimsuky
- Lazarus Group (also known as APT38)
- Ricochet Chollima (also known as APT37)

Russia [edit]

- Berserk Bear
- Cozy Bear (also known as APT29)
- Fancy Bear (also known as APT28)

China [edit]

See also: Cyberwarfare by China, Chinese information op abroad

- PLA Unit 61398 (also known as APT1)
- PLA Unit 61486 (also known as APT2)
- Buckeye (also known as APT3)[39]
- Red Apollo (also known as APT10)
- Numbered Panda (also known as APT12)
- DeputyDog (also known as APT17)[40]
- Dynamite Panda or Scandium (also known as APT18, a
- Codoso Team (also known as APT19)
- Wocao (also known as APT20)[42][43]
- APT22 (aka Suckfly)[44]
- APT26 (aka Turbine Panda)[45]
- APT 27[46]
- PLA Unit 78020 (also known as APT30 and Naikon)
- Zirconium[47] (also known as APT31 and Violet Typhoon)
- APT40

[1] https://attack.mitre.org/groups/
[2] https://en.wikipedia.org/wiki/Advanced_persistent_threat#APT_groups

近年の話で個人的にきついと思っているもの(1/2)

IPAも毎年セキュリティ脅威Top 10[1](ついに"〇年連続"も追加)を出していますが、個人的に印象が残っているものを

- ランサムウェアを他と組み合わせて効率的に武器化
  - ランサムウェア: データを暗号化して読めなくして身代金を請求
  - 脆弱性と組み合わせて送り込んだり、感染後に人手で操作したり
  - 窃取したデータを公開することも新たな脅迫の種に
- 個人や企業のつながりを狙った攻撃の増加(復権?)
  - 内部業務フローを把握して詐欺請求を送るビジネスメール詐欺[2]
  - 受信メールボックスにあるメールに対して返信をするマルウェアEmotetの流行(たまに復活する)
- 地政学リスクがもたらすサイバー攻撃の可能性
  - インフラ系企業は特にこれにピリピリしている

[1] https://www.ipa.go.jp/security/10threats/10threats2026.html
[2] https://forbesjapan.com/articles/detail/29551

近年の話で個人的にきついと思っているもの(2/2)

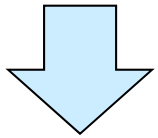
- 多要素認証の突破を狙った攻撃
  - 2019年の時点でネットバンキング不正送金(全1852件)の56%はワンタイムパスワード(OTP)を突破している話がある[1]
  - 中間者攻撃をやりやすいプロキシ型フィッシング(詐欺)サイトが増加
  - OTP生成用シードを狙うマルウェアやOTP窃取目的に携帯電話会社側のSMSサーバを狙ったマルウェアも[2]
- 携帯電話のSIM(回線情報の入ったカード)を狙った攻撃
  - 携帯電話会社のSIM再発行を騙して標的のSIMを再発行させる
    - 2022年あたりから国内でも増えてきた[3]
    - この先のeSIMの普及でもっと容易になりそう
  - SIM自体のセキュリティ破りを狙うSimjacker攻撃

→「携帯電話番号(SMS含む)」に依存したセキュリティはダメに

[1] https://www.nikkei.com/article/DGXMZO56407040V00C20A3MM0000/
[2] https://blog.kaspersky.co.jp/simjacker-sim-espionage/24282/
[3] https://www.yomiuri.co.jp/national/20230414-OYT1T50157/

防御側から見えている希望

- 防御側からも希望はいっぱいある
  - クラウドコンピューティングを利用した集中防御
  - SDN(Software Defined Network)による柔軟なネットワーク制御
  - 機械/深層学習の応用による新攻撃の発見および対応の自動化
  - 演習を含めた一般の人への情報セキュリティ教育の推進
    - もはや、サイバー犯罪対策は地震対策や火災対策と同レベルの認識になるべきだが、現状はそこまでの認識になっていない
- まだまだ色々新しいやり方はあると思う



- とりあえず、新しいやり方を考えるには現状の把握が重要

→本講義でいろいろと概論の形で話します

  - 背景技術は上の学年で学ぶ物も多いので、ざっくりと概論で

概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

ネットワークを介した情報ネットワークの基礎

ネットワークを介した情報サービスの基礎を軽く説明

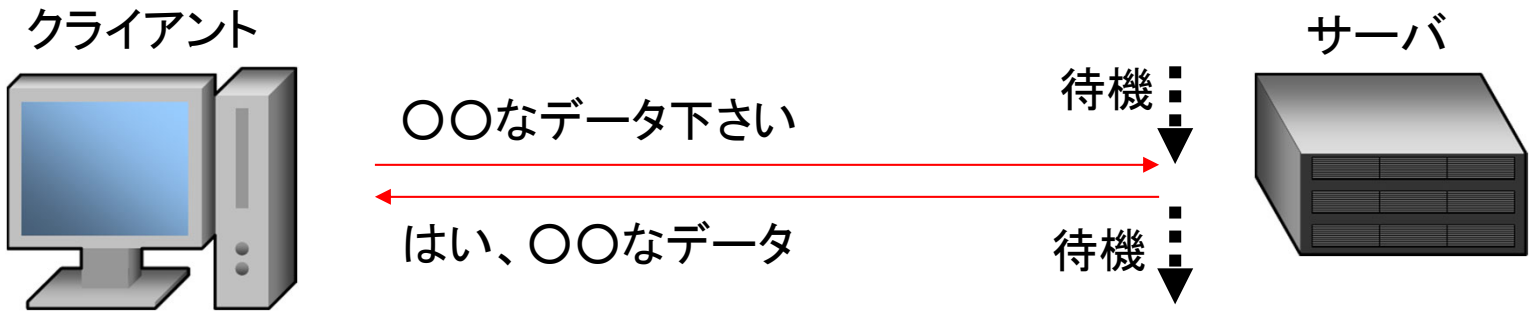
- サーバとクライアント
- インターネットプロトコル周りの処理
- 代表的なネットワークサービスとその動作を考える

→ウェブブラウザからGoogle検索→Google Mapの地図へ

- Googleの検索ページ(http://www.google.com/)を開く
  - DNSによるIPアドレスへの変換、HTTPによる通信手続き(HTTP GET)
- Googleの検索フォームに入力して検索ボタンを押す
  - HTTPのPOSTやURLクエリ
- Google Mapに移動
  - Ajaxによる地図タイル読み込み

ネットワーク上のサービスにおけるサーバとクライアント

- サーバ(server: 給仕する人): サービスを提供する側
- クライアント(client: 顧客、依頼主): サービスを受ける側
- 基本的に、通信開始依頼を待ち受けている側がサーバ
  - 基本的にネットワークサービスは待受側がいることを前提としている
  - 状況によっては、一時的にサーバとクライアントが入れ替わることも
- 余談: コンピュータ科学科では学生実験で自分でサーバ-クライアントのシステムをプログラミングしてもらいます



概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

インターネット上でのサーバの指定のしかた

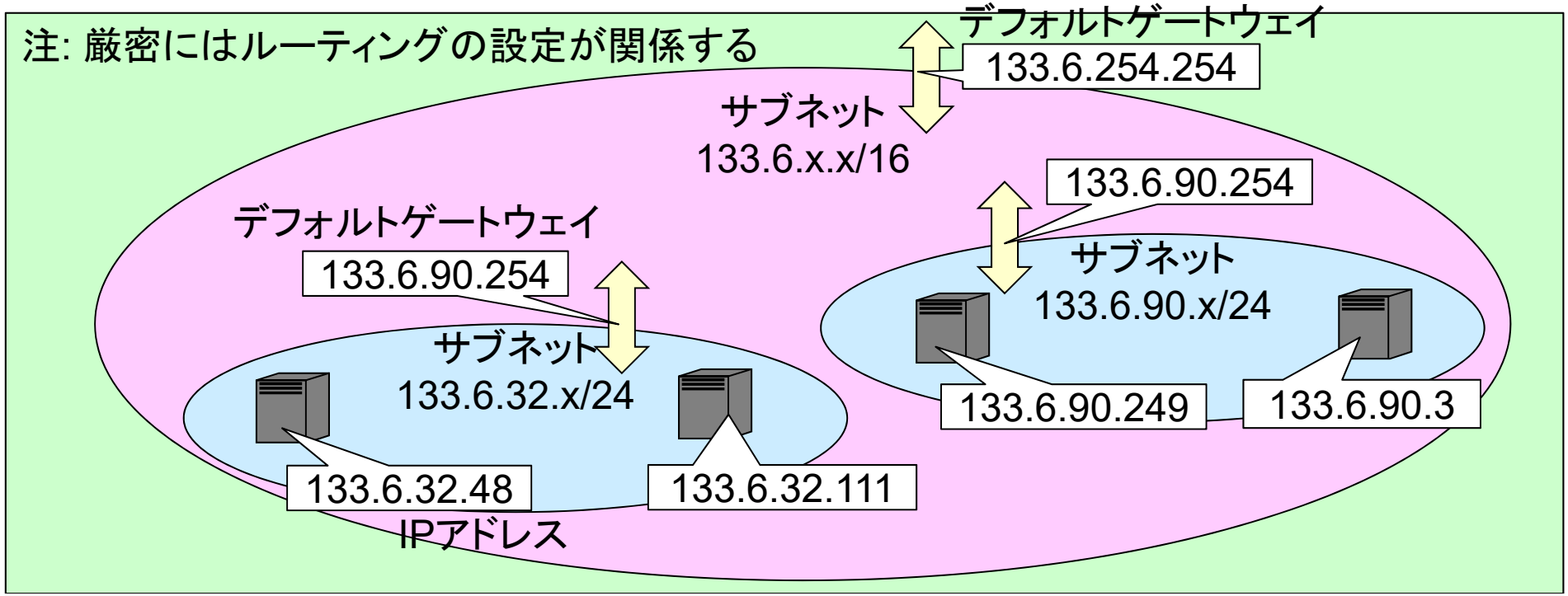
- IPアドレスを利用
  - インターネットプロトコル(IP)における、インターネット上の通信元/通信先の識別子
  - IPv4(version 4 = 第4版)とIPv6(version 6)のアドレスがある
- IPv4アドレス
  - 4個の0-255の数字(2進数で8桁 = 8bit)をピリオドで区切った表記
  - 例: 133.6.90.249
  - 総数4,294,967,296個(2の32乗)だが、とても足りない
    - 実際は特殊用途IPアドレスが指定されているので、さらに不足
    - DHCPで必要な時に一時的に割り当てたり、NATで1つのグローバルIPアドレスを複数プライベートIPアドレスの出口にしたり
    - 根本的な解決のためにIPv6アドレス(総数は2の128乗個)へ移行中
- 約42億個とは言え、ネットワーク上で通信先を探すのは工夫が必要

IPにおける通信の概要(1/2)

IP上の通信はサブネット内と外への通信に分けて考える

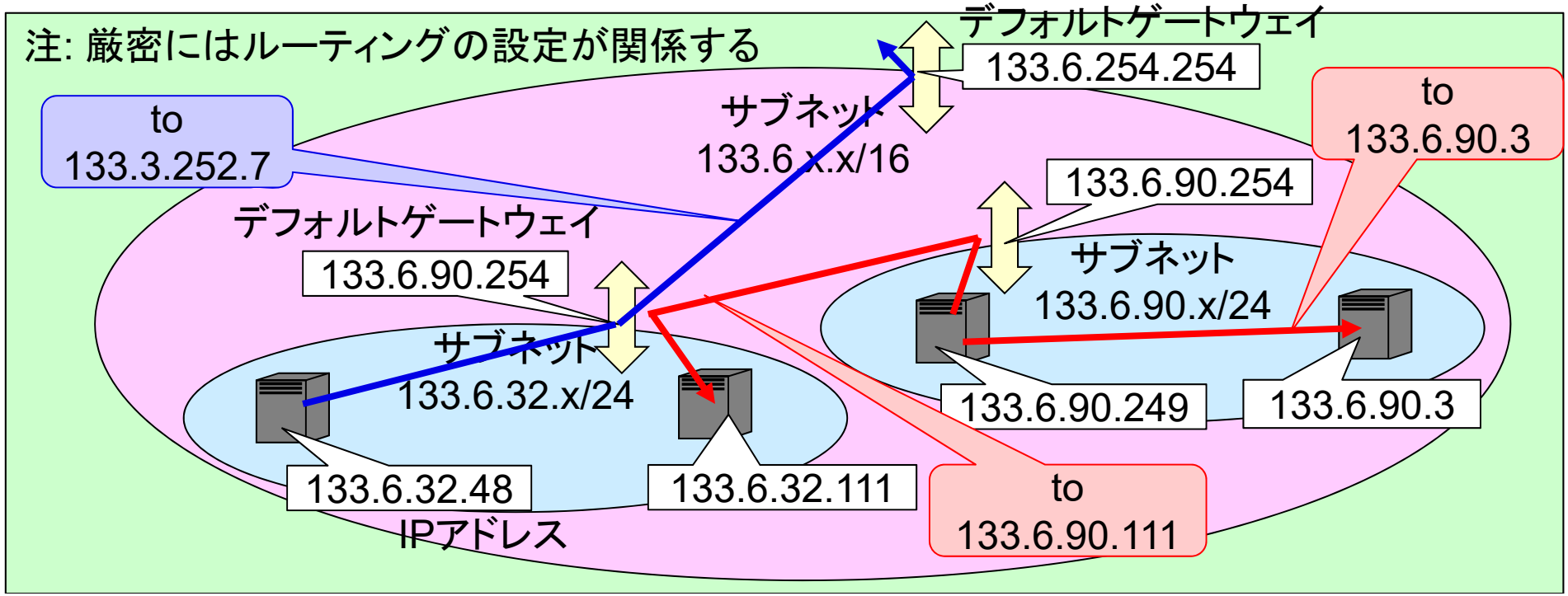
- 同一サブネット内の宛先は直接送る
- 別サブネットの宛先はとりあえず出口(デフォルトゲートウェイ)に送り転送してもらう
  - 下位のサブネットに宛先がある場合、そちらに転送する

経路ハイジャックなるサイバー攻撃もある(例: 2018/4 Amazon Route53)



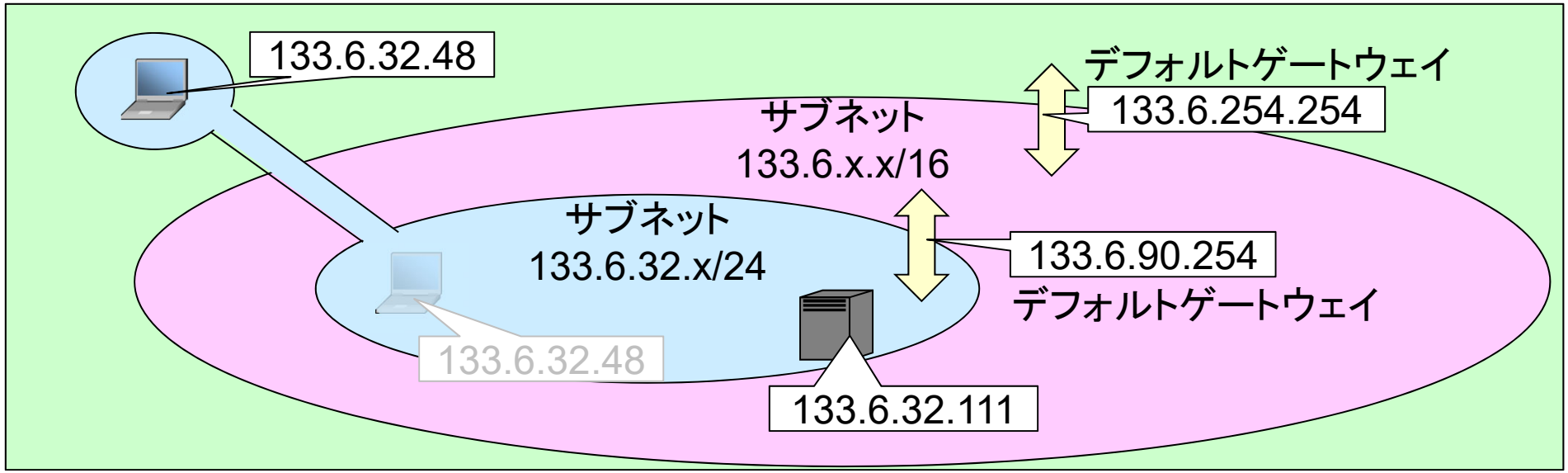
IPにおける通信の概要(2/2)

- 133.6.90.249→133.6.90.3: 同一サブネット内だから直接送る
- 133.6.90.249→133.6.32.111: デフォルトゲートウェイ経由で上位サブネットに送り、上位サブネットが包含するサブネットに転送
- 133.6.32.48→133.3.252.7: 上位サブネットにも存在しないため、上位サブネットのデフォルトゲートウェイ経由でさらに外へ



余談: テレワークで出てくるVPN(Virtual Private Network)

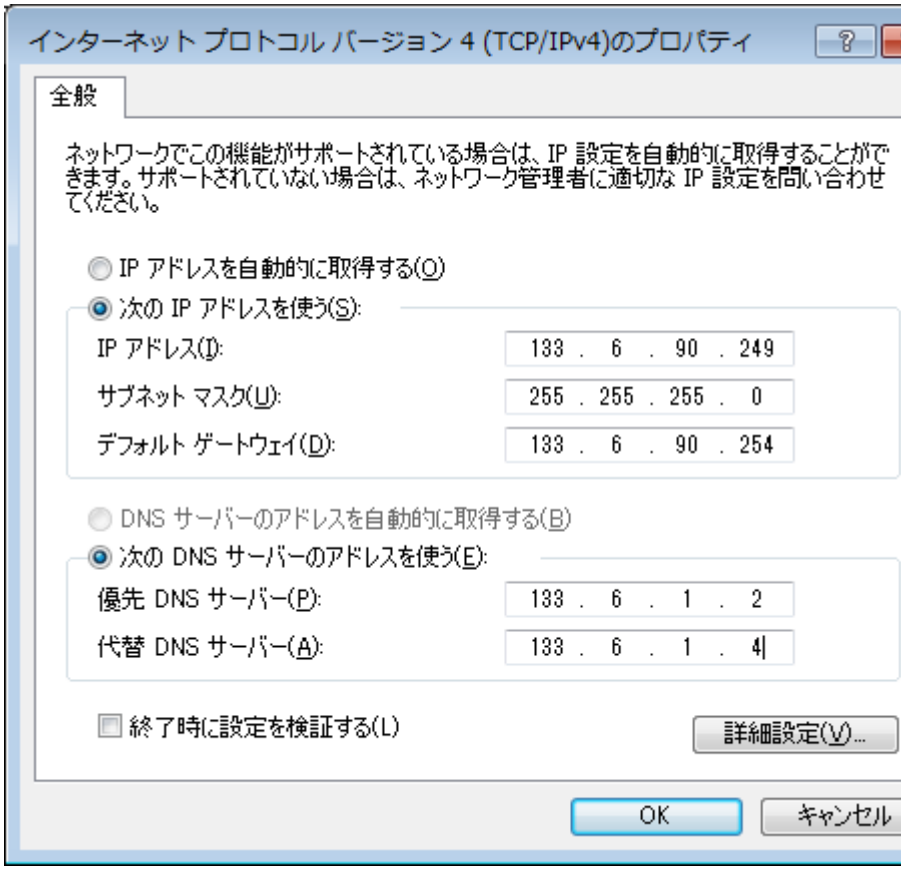
- 仮想的にネットワークを構成する手法(多数の手法がある)
  - 遠隔で特定サブネットに入るためによく多用される
- 全通信をVPNに通す運用と、特定通信のみをVPNに通す運用(スプリットトンネリング)がある
  - 組織のセキュリティ的には、全通信を通した方が監視が容易で安全
- 将来的には、ゼロトラスト前提で不要になる?



WindowsにおけるIPv4の設定

- 一般的な設定項目
  - IPアドレス
  - サブネットマスク:
    サブネットの範囲を規定
  - デフォルトゲートウェイ:
    サブネットの出口
  - DNSサーバ:
    ホスト名表記からIPアドレスへの変換を行うサーバの指定(後述)
- DHCPという、自動で設定する手続きもあり
  - 「…自動的に…」の項目
  - IPv6だと自動設定が大幅に増えて悪用されそうだという話が…

コントロールパネルからのIPv4設定
(コンパネからの設定は一覧性が良い)



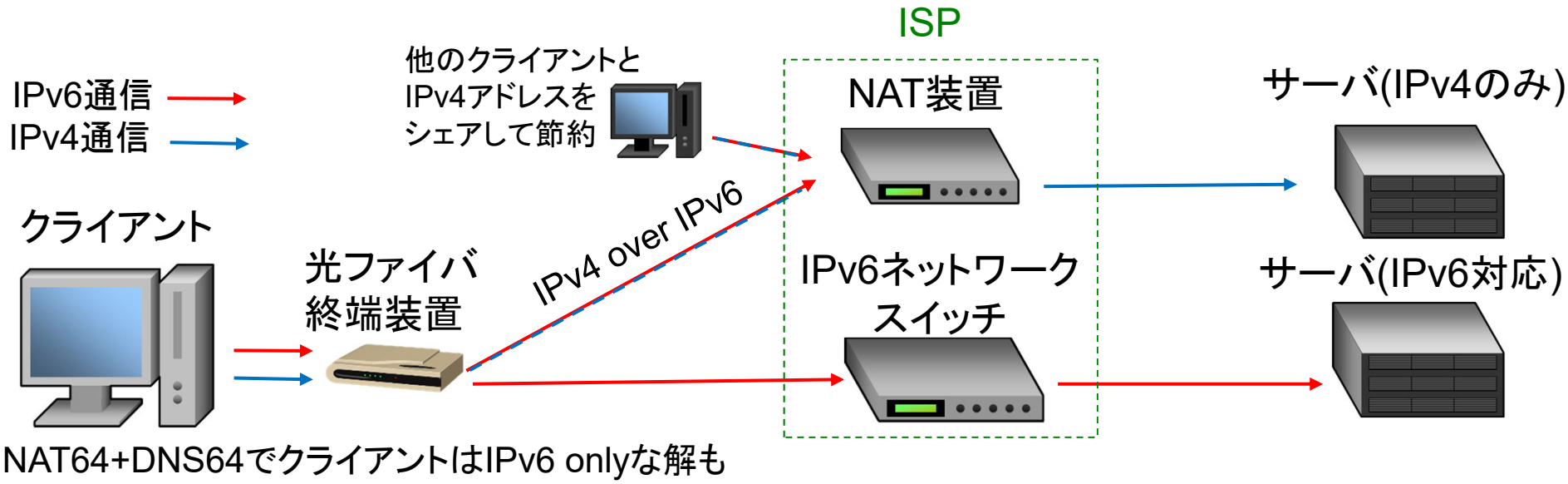
IPv6についてちょこっと

- 利用者的には、「IPv4のアドレスを拡張した上で、DHCPの機能を取り込んだもの」と考えればOK
- IPv4と同じく、サブネット識別部(プレフィックス)がある
- 16進数で表記するが、それでも32文字になるので略記
  - 基本は4文字ごとに:で区切る
    (2001:02f8:002f:0100:0000:0000:0000:0102)
  - 「4文字とも0」の連続を::で省略可(→2001:02f8:002f:0100::0102)
  - 「4文字の冒頭の0」を省略可(→2001:2f8:2f:100::102)
- 名大内はすでにIPv6 readyだが利用は縮小中
  - セキュリティ面から「全世界の端末が自由に1対1接続」の利点が減
  - 2026/4にIPv8なるものが改めて提案されたニュースが
- IPv4/IPv6のデュアルスタックするとルータが扱うIPアドレスが3倍になるのでやっかい(2022/4と2024/4のNUWNET障害原因)

IPv4アドレス不足によるISPのIPv6デフォルト化(IPoE利用)

IPv4アドレス不足もあり、光ファイバ網からISP(インターネットサービスプロバイダ)への通信に、IPv6を主体するIPoE利用へ

- IPv4通信はIPv4 over IPv6とNAT(アドレス変換)で対応
- ただし、IPv4で固定されたポートが欲しい自宅サーバ勢や(古い)オンラインゲーム利用者はさらなる工夫が必要
- 昔から使われていたIPv4なPPPoEの設備投資は削減



概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

DNS(Domain Name System)

- 通信の宛先のIPアドレスを覚えるのは面倒
  →意味のある識別子からIPアドレスに変換させよう
  - ホスト名とかFQDN(Fully Qualified Domain Name)と呼ばれる
  - 複数のIPアドレスに変換して負荷分散など副次効果も
- 例: www.itc.nagoya-u.ac.jp→133.6.91.82
  - www: World Wide Web
  - itc: Information Technology Center
  - ac: ACademic
  - jp: JaPan
- DNSサーバが変換を司る
  - DNSサーバもネットワーク上のサーバなのでIPアドレスで指定
- 本物に似せたFQDNで偽サーバを準備する攻撃は多い
  - サブドメインを使う事例とか(例: nagoya-u.ac.jp.example.com)

DNSに関する情報セキュリティ

- DNSによる変換過程に変な情報を入れて変な所に誘導するサイバー攻撃(DNSハイジャック)もある
  - 偽ウェブサーバに接続させて認証情報を入力させる攻撃と連携させるとか
- 逆に、DNSで変換をわざと失敗させて悪性サーバに接続させないセキュリティ対策もある
- 普通は組織(大学やISP)が提供するDNSを使うが、他のDNSを指定してもOK
  - Googleの提供するDNS(8.8.8.8や8.8.4.4)は代替DNSとしてよく使う
  - デバイスによってはDNSが固定されることもあるらしい
    - DNSへのリクエストも通信の秘密で守るものの対象のため、勝手にDNSリクエストを収集されて解析されるのは問題
    - DNS over HTTPSなどで特定DNSにリクエストを投げたりする物も

概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

どんな通信規則(プロトコル)で通信するかの指定

- URL(http://www.google.com/)のhttp://の部分は、利用する通信規則(プロトコル)を表す
  - HTTP(Hyper Text Transfer Protocol)での通信を指定
  - https://www.google.com/ ならば通信内容を暗号化するHTTPS(Hyper Text Transfer Protocol Secure)で通信
    - 基本HTTPSという流れにある(man-in-the-middle対策)
    - 多くのWebサーバでHTTPで接続されるとHTTPSに投げる設定に
    - なお、HTTPSは悪性ウェブサイトも普通に使っているため、HTTPSか否かは悪性ウェブサイト判別には全く役に立たない
  - 昔は「ftp://」なFTP(File Transfer Protocol)なども使われた
- 余談: 最近のブラウザでは「http://...」は表示されないことも
  - 「http://」な部分を消している(Firefox)
  - ウェブページによってはアドレスバー自体を消す工夫(うさんくさい)も

→ネットワークセキュリティ上、個人的には表示を推奨する

プロトコルとTCP/UDPのポート(1/2)

- 実はプロトコルも番号に変換されている(DNSによるIPアドレスへの変換みたいに)
  - 基本的に「XXプロトコルならば〇〇番」というルール
    - 厳密にはポート番号の割当は自由だが、自動変換されない
- 通信のやり方でTCPとUDPの2種類がある(後述)

→最終的には、通信はIPアドレス、TCP/UDP、ポートで指定

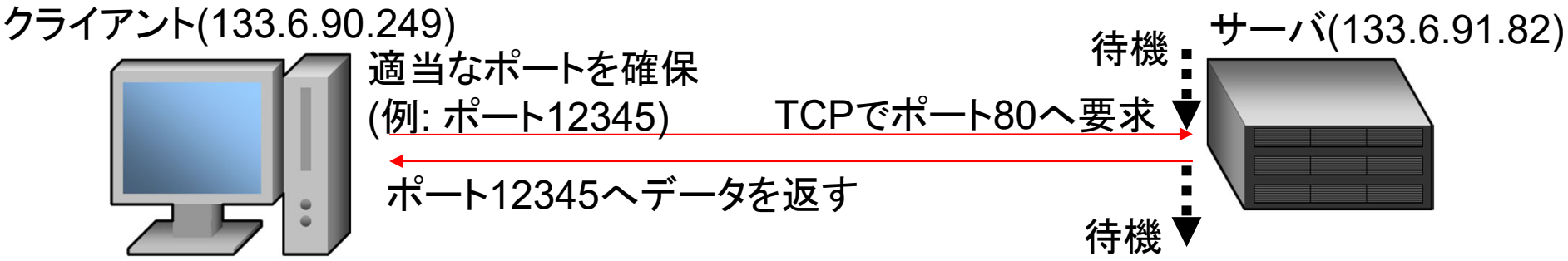
  - HTTPならばTCPの80番ポート

- サーバ側は特定のポート番号でクライアントからの要求を待ち受ける
- クライアント側も、要求を出す時に戻り先のポート番号を確保してサーバ側に提示

プロトコルとTCP/UDPのポート

一般的なTCPによる通信

1. サーバは(基本的に)サービスに対応したポートで待ち受け
   - 例: HTTPならば80番ポート
2. クライアント適当なポート番号を確保して、そこからサーバに接続要求を出す
   - TCP/UDPのポート番号は0-65535の範囲
   - 確保するポートは通常は1024以上のポート番号
3. サーバ側は提示されたポートに、要求されたデータを返す
   - 最初の接続要求がうまくいかなければ、通信は成立しない(混雑とか)



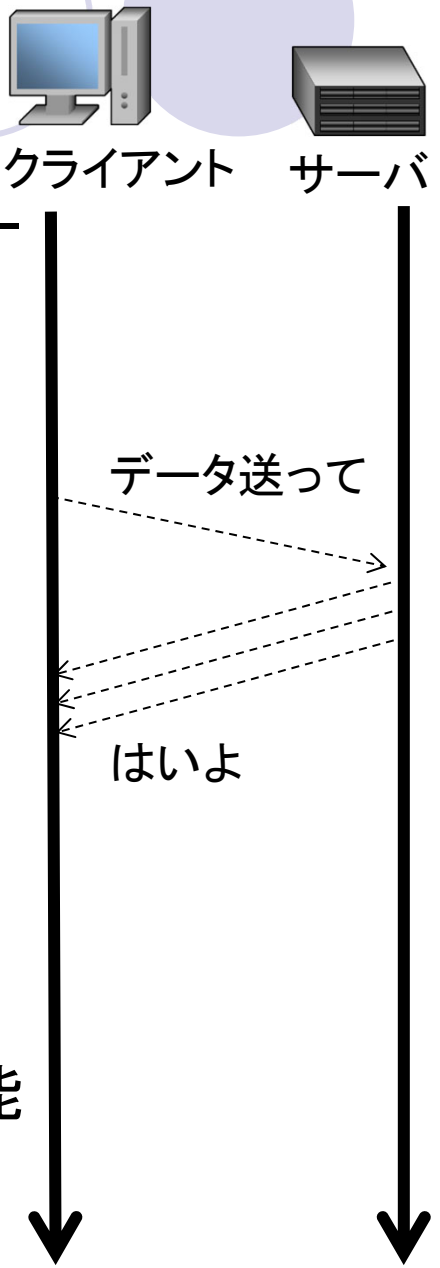
TCPとUDPの違い(1/2)

- TCP(Transmission Control Protocol)は確実なデータのやりとりを保証する
- 3ウェイハンドシェイクで通信の開始/終了
  - SYN or FIN: 通信OK? or 終了OK?
  - SYN+ACK or FIN+ACK: OK!
  - ACK: ほな行くでー or ほなさいならー
- 通信はパケット単位で番号づけして管理
  - 抜けたパケットは再送依頼を出す
  - 「何個まで確認なしにパケットを送るか」の上限がある(回線の状態によって決める)

    →効率が悪い
- 手続きをわざと間違えたりするサイバー攻撃が多い

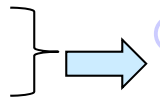
TCPとUDPの違い(2/2)

- UDP(User Datagram Protocol)は基本的にデータを投げるだけ
- パケットが抜け落ちても気にしない
  - 映像ストリーミングなどの「少し画像乱れても気にしないで」なサービスでの利用
  - 後からデータを検証することを前提の高速データ転送
  - HTTP/3にも利用されるQUICはUDPを利用
- サイバー攻撃上ではけっこうやっかい
  - ひたすらデータを送り込む(DoS攻撃: Denial of Service攻撃)
  - 返信先を偽る(リフレクション攻撃)
- TCP/UDPとも途中で返信先ポートの変更は可能
  - 特に、TCPでは次の待受のために通信やり取りは別ポートにする



各種サービスとポート

- 一般的に、各種サービスとTCP/UDPの待ち受けポートは標準的な番号が決まっている
- 例
  - HTTP: 80/TCP
  - QUIC: 80/UDP, 443/UDP
  - HTTPS: 443/TCP
  - DNS: 53/UDP
  - (FTP: 21/TCP)
  - (telnet: 23/TCP)
  - SMTP: 25/TCP
  - NTP: 123/UDP

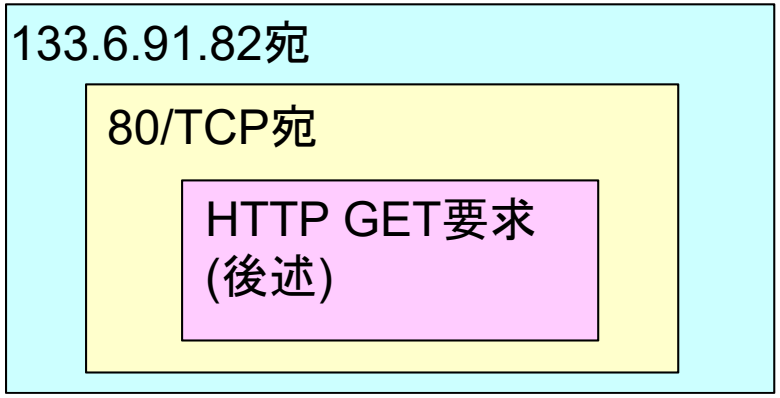
  - SSH/SCP: 22/TCP
- ただし、標準以外のポートで待ち受けも可能
  - 例: HTTPを8080/TCPとか10080/TCPで待ち受け
  - 「http://xxx.xxx.xxx:8080/」のように「ホスト名:ポート番号」で指定
  - 例: マルウェアMiraiはtelnetを23/TCPの他に2323/TCPで利用
- 待ち受けポートを開けるプログラムの管理&ファイアウォール等でブロックするのは(サーバ)セキュリティの基本

概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

前述ネットワーク通信に関する手続きはどこが行なう?

- 基本的にはOS(Operating System)に実装されたプロトコルスタックが行なう
  - OSの例: Linux, Android, iOS, MacOS, Windows
  - 各種プロトコルスタックの例: TCP, UDP, IP, イーサネット
  - なぜスタック? →ネットワークは階層モデルを構築して動いている
- ネットワークの階層モデル
  - 下位の階層は上位階層のデータをカプセル化して、その階層のルールで送る
  - 送る方は順番にカプセル化し、受ける方は順番にカプセルをほどく
  - カプセル化の例:



TCP/IPから見たネットワーク階層(OSI参照モデル)

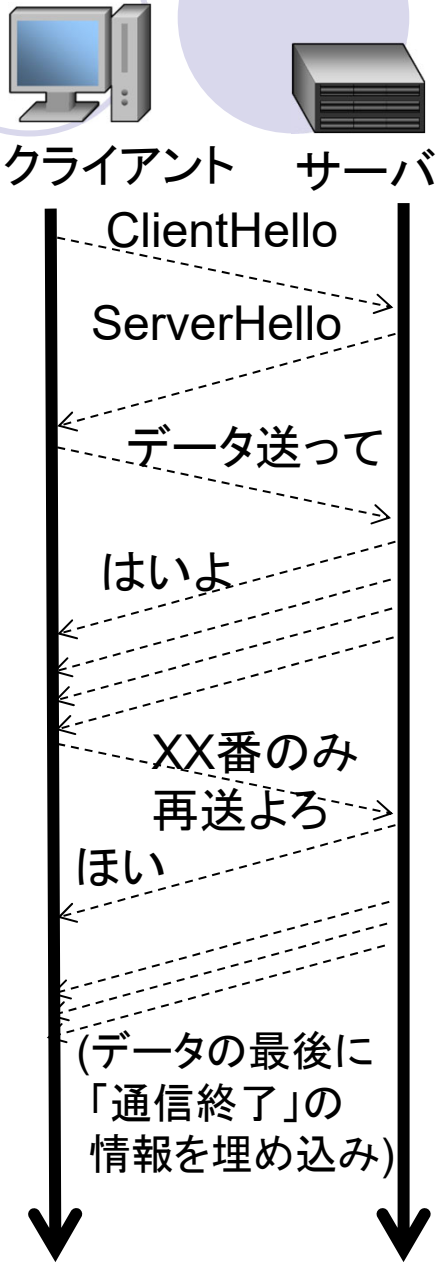
1. 物理層: 光ファイバ、カテゴリ6 UTPケーブル、5GHz帯無線
2. データリンク層: 物理層などに応じたフレームでIPパケットをカプセル化して送る
3. ネットワーク層: IP
4. トランスポート層: TCP/UDP
5. セッション層: TCPのセッションの制御、UDPのデータ送出
   - 個人的には、現在では5,6層もアプリケーションがカバーする事例も多いので、まとめてアプリケーション層で考えた方が実践的
6. プレゼンテーション層: データ表現の統一(影薄い)
7. アプリケーション層: ウェブサーバソフトウェアやブラウザ
   - 近年の実装では、この層でさらに細かく分けたくなる

最近ではさらに階層分けの意義が薄くなっているが、システム構成の考え方としては参考になる

QUICプロトコル

- TCPの再送処理はネットワークが遅い時代をベースに設計していて現在では効率悪い
- 現在は暗号化は必須だが、TCPハンドシェイク後に暗号化ハンドシェイク実施は効率悪い

- QUIC(Quic UDP Internet Connections)
  - 実際のデータのやりとりはUDPで行う
  - アプリケーション層でTCPの再送処理を実装
  - 通信と暗号化のハンドシェイクを一括実施
  - 2回目以降は前回のハンドシェイク情報を使ってさらに高速にハンドシェイク
- Googleが提案し(2012年)、IETF(Internet Engineering Task Force)が標準化(2021年)



概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

クライアント側でのウェブのデータ操作

- ウェブブラウザを通して操作
  - ウェブブラウザ(ブラウザ)の例: Edge, Firefox, Chrome, Safari
- URLを入れるとHTTP GET要求でデータをもらって表示
- ウェブページのフォームに入力したデータをHTTP POSTやURLクエリでウェブサーバに提供

サーバ側でのウェブのデータ操作(1/2)

ウェブサーバソフトウェアを利用

- 代表例: Internet Information Service, Apache, nginx
- 事前に登録しておいたデータを表示
- クライアントからのアクセス方法により動作を変更可能
- クライアントから受けたデータをサーバ内に保存
  - 必要に応じて、保存したデータを加工して表示
- ウェブサーバ側でクライアントのリクエストに応じてプログラムで処理した結果を提示可能
  - CGI方式による外部プログラム呼び出しやPHP言語(Hypertext Processor)の利用
  - クライアント側でJavaScriptプログラムを動作させて、ユーザの操作に応じたリクエストを出す(ウェブブラウザ側から)物もある →Ajax

サーバ側でのウェブのデータ操作(2/2)

- URLの最初の/より後でウェブサーバ内のデータの位置を表す

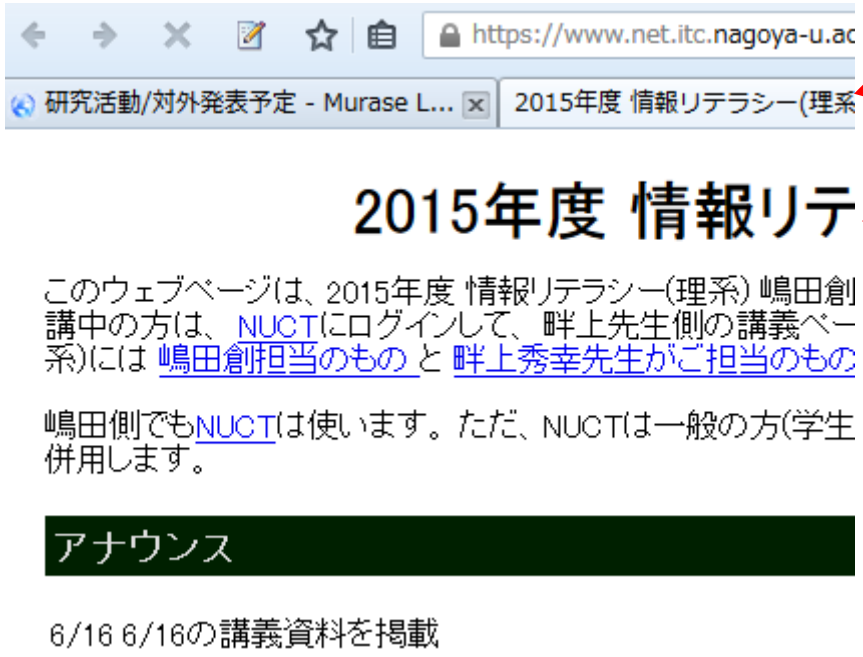
  例: https://www.net.itc.nagoya-u.ac.jp/member/shimada/index.html

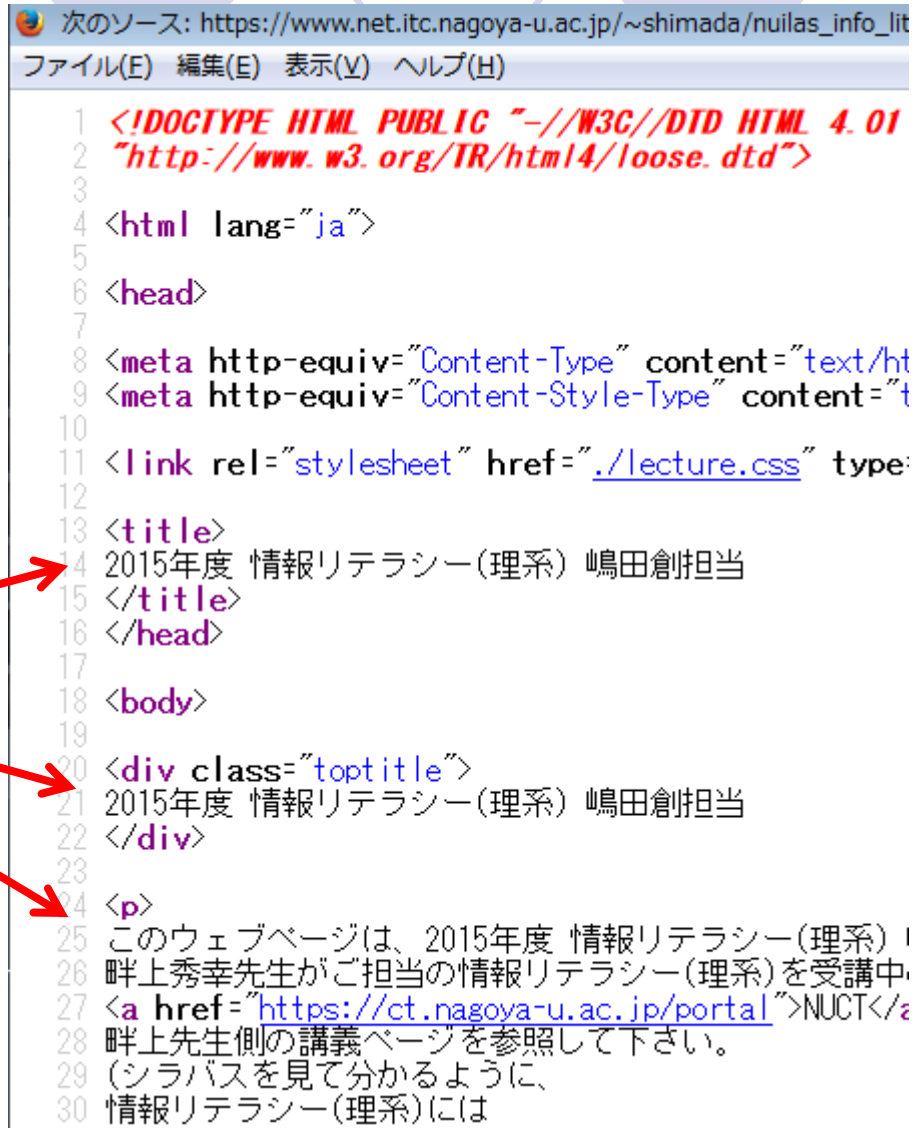
  - membersディレクトリ(=フォルダ)のshimadaディレクトリのindex.htmlというファイルを指定
  - 最初の/の位置はウェブサーバソフトウェアの設定で決定
    - 例: /var/www, /usr/local/www
- このあたりでやらかす不注意な情報公開
  - 「2016/kessan.pdf」をもとに「2017/kessan.pdf(未公開)」を発見される
  - 「2016/」とディレクトリ名でアクセスすると一覧が見えてしまい(ディレクトリインデクス設定)、その中に編集途中の物などの都合の悪い物が…

  個人的には、誤って公開するぐらいならば、誤って公開されていない方が安全な気がする(よくやる講義資料公開ミスに対する開き直り)

HTML(Hyper Text Markup Language)

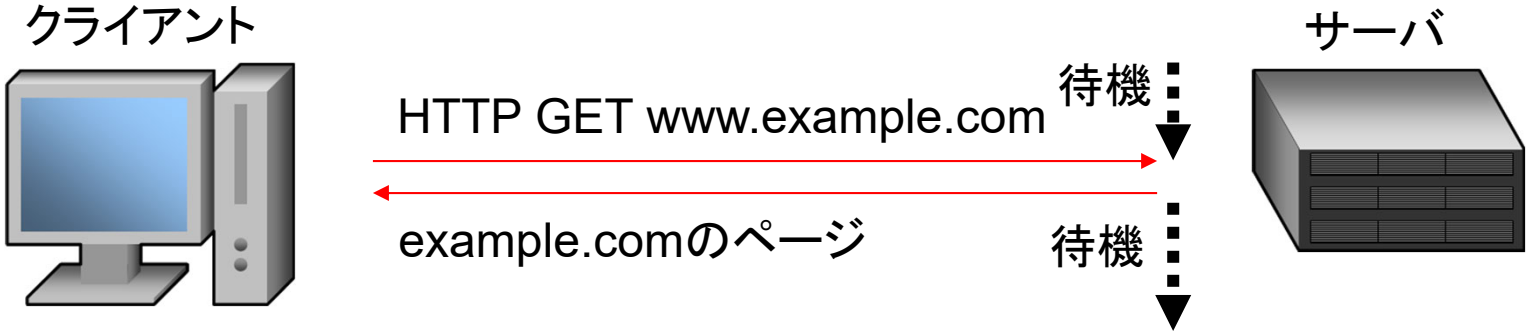
- ウェブ上の文書の多くはHTML
- 文章に<>で表されるタグをつけて文章構造を定義
- 修飾はCSS (Cascading Style Sheet)で実施





HTTP GET

- HTTP GET: ウェブサーバへのコンテンツ要求
  - 言語、ブラウザ、受付可能なメディアタイプ、なども送信
  - 必要に応じて後述するCookieやAuthenticationなどの情報も送信
- HTTP GETへのウェブサーバ側の応答
  - コンテンツ(HTMLファイル、mp4動画ファイル、PDFファイル、など)
  - それ以外の情報: 最終更新日時、コンテンツのタイプ、など
  - 必要に応じて後述するSet-Cookieなどの情報も送信
- スマホアプリも裏ではHTTPで通信している物が非常に多い
  - 独自規格を0から作るよりは、広く使われている規格を使う方がいい



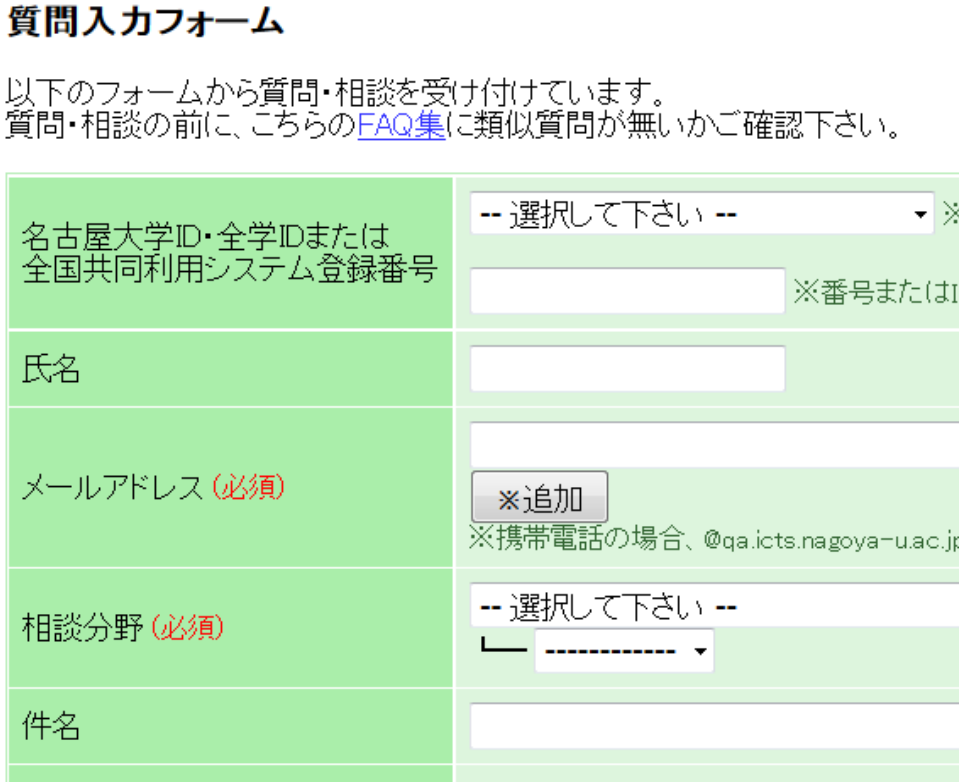
HTTP POST

- 質問入力フォームやウェブ掲示板書き込みなど、クライアント側からサーバにデータの塊を送る時に利用
  - 一旦、ウェブサーバ側から質問入力フォームをHTTP GETしてから
- HTTP GET要求の末尾に「入力フォームの内容」の羅列を追加する感じ
  - 各記入項目にはサーバ側で名前が付けられている

HTTP POSTの例

名大ITヘルプデスクの質問入力フォーム

- HTMLのformタグでpostが指定されている(上の赤線)
- 入力項目にはnameが指定されている(下の赤線)

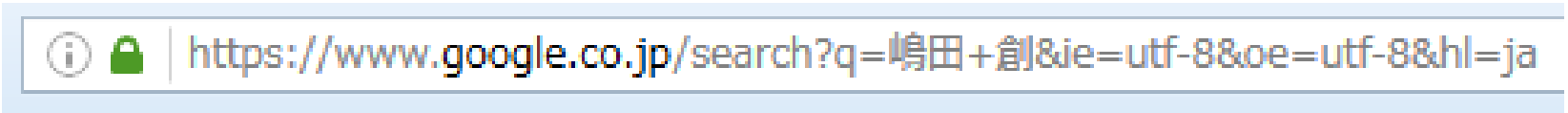



```
24 <form method="post" action="/index.php" enctype="mult
25 <input type="hidden" name="action" value="post">
26 <input type="hidden" name="input_mail_addr" value="">
27 <table class="common_table" cellspacing="1">
28 <tr>
29 <th>名古屋大学ID・全学IDまたは<br>
30 全国共同利用システム登録番号</th>
31 <td>
32 <select name="id_type_code">
33 <option value="0">-- 選択して下さい --
34 <option value="nagoya_id">名古屋大学ID・全学ID
35 <option value="sys_regist_num">全国共同利用システム登録
36 </select>
37  <span class="comment">※種別</span>
38 <br>
39 <br>
40 <input type="text" name="id_value" value="" size="20">
41 </td>
42 </tr>
43 <tr>
44 <th>氏名</th>
45 <td><input type="text" name="name" value="" size="20">
```

URLクエリ

- HTTP POSTと同様にウェブブラウザからウェブサーバ側へ情報を渡す方法
- URLの末尾に「?」をつけ、それ以降がURLクエリとなる
- 個々のパラメータと値は「"パラメータ名"="値"」で表現
- 複数のパラメータがある場合は「&」で区切る

例:



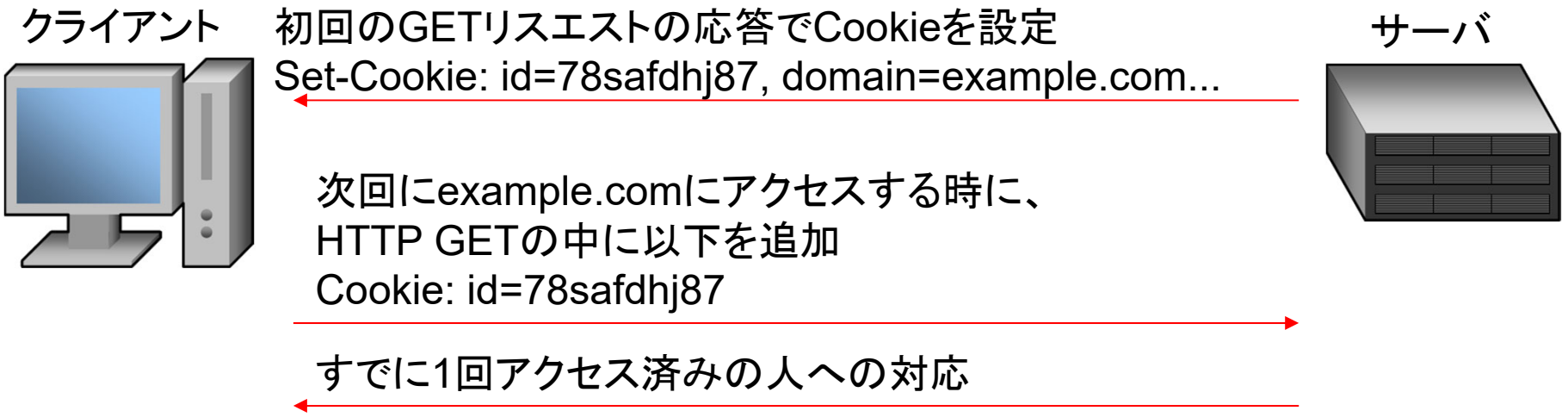
  - 実際のURLでは全角文字を扱えないので、パーセントエンコーディングされる(「=&?」などの記号も同じ)

https://www.google.co.jp/search?q=%E5%B6%8B%E7%94%B0+%E5%89%B5&ie=utf-8&oe=utf-8&hl=ja

- URLクエリはブラウザのアドレスバーに表示されるため、おおっぴらに見えて欲しくない情報が露わになることも

Cookie

- GETに対しての応答だけだと、連続したやりとりに対応できない →HTTP Cookieを利用
  - サーバ側からクライアント(のウェブブラウザ)に情報を記憶させることができる、広く使われている仕様
  - ウェブブラウザはCookieのドメインに対応するウェブサーバへの再アクセス時にはCookieの情報を送信
    - 乱数ベースの推測不能なIDを記憶させておいて、再アクセスを判別



Cookie関連の情報セキュリティ

- Tracking Cookie: 利用者の追跡を目的としたCookie
  - 主に広告コンテンツが実施
    - 例: サイトAに出した広告でCookieを設定 → サイトBに出した広告で設定したCookieを受け取る
    - 通常は3rd party Cookie(閲覧中サイトとは違うドメインのCookie)となる
  - 気になるならばブラウザで「Do not track」の設定と「3rd party Cookieの禁止」を実施しておく
    - それでも、IPアドレスやUser agent情報などからある程度は追跡できる
- その他のトラッキング手法
  - HTTP Cookieと同等の物をHTTPとは別の処理の中で実装
    - 例: HTML5のLocal storageを使ったデータ共有で追跡
  - 他にも、Canvas fingerprintingなど、利用者追跡の技術は積極的に開発されている(有効性の高低はあるが)

余談: コンテンツマネージメントシステム(CMS)と情報セキュリティ

- コンテンツを登録するだけできれいなウェブページを作成可能なため、最近ではWebサーバ側で多用される
  - データベースに登録したコンテンツを指定フォーマットに沿って表示
  - Electronic Commerce(電子商取引)用のCMSもある
- 代表的なCMSとシェア[1]
  - WordPress、Joomla、Drupalの利用が多い
- 最近、脆弱性を突かれてトラブルを起こすこと多い
  - バックエンドで利用中のデータベースシステムの脆弱性を突かれる
  - プラグインの脆弱性を突かれる
- 一番重要なこと: 最新のCMSを使いましょう
- WikiエンジンとかBlogエンジンもCMSのうち

[1] https://w3techs.com/technologies/history_overview/content_management

概要

- (情報セキュリティの現状)
- クライアントとサーバ
- ウェブサーバ利用時のネットワーク上の動作
  - IPアドレス
  - DNS
  - TCP/UDP
  - OSのプロトコルスタック
  - クライアント/サーバソフトウェア
  - HTTPによるやりとり
- その他のサーバ

ウェブ以外の種々のサーバによる情報サービスの提供(1/2)

TCP/IP, UDP/IPを利用したサービスはウェブ以外にも多数

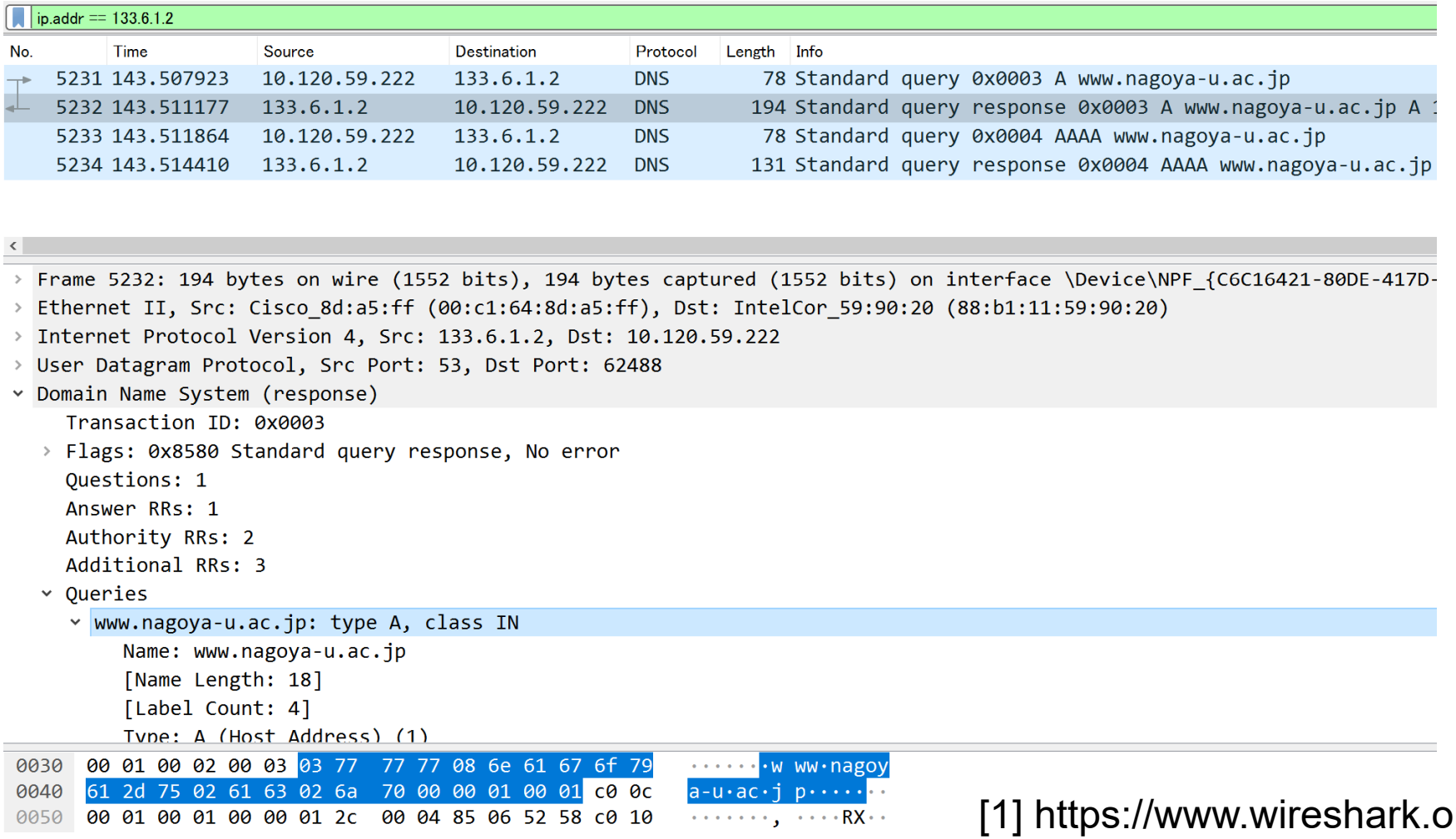
- 電子メール
  - SMTP(Simple Mail Transfer Protocol): メールサーバ間のメール送信
  - POP3(Post Office Protocol): メールサーバからのメール読み出し
  - IMAP(Internet Massage Access Protocol): メールサーバ上のメール閲覧
  - Submission: メールサーバへのメール提出
- FTP, SMB, NFSによるファイルの転送やフォルダ共有
- リモートデスクトップ, telnet, SSHなどのコンピュータの遠隔操作
- NTPによる自動時刻合わせ

種々のサーバによる情報サービスの提供(2/2)

- 昔に設計されたサービスはセキュリティ上の問題があって後継サービスに移っているもの多い(古いのは利用禁止)
  - 例: 暗号化されていないパスワードがネット上を流れる物を改良
    - telnet, rsh → SSH
    - FTP → FTPS, SSH(SSHサブセットのSCP, SFTP)
    - POP3/IMAP → POP3S/IMAPS
  - 例: そもそも認証が緩すぎる物を置き換え
    - SMTP(クライアントとサーバの間において) →Submission
- とは言え、個々のプロトコルをセキュアにするのも大変なので、最近ではHTTPS(+QUIC)を下層に用いる事例も
  - 例: DNS over HTTPS, SMB over QUIC
  - というか、最近はあらゆるサービスがHTTPS化されていく傾向

ネットワーク上のやりとりを可視化して見てみたい人へ

- Wireshark[1]をインストールして自分の通信を見てみよう



[1] https://www.wireshark.org/