コンピュータ・サイエンス2

第8回
情報ネットワーク

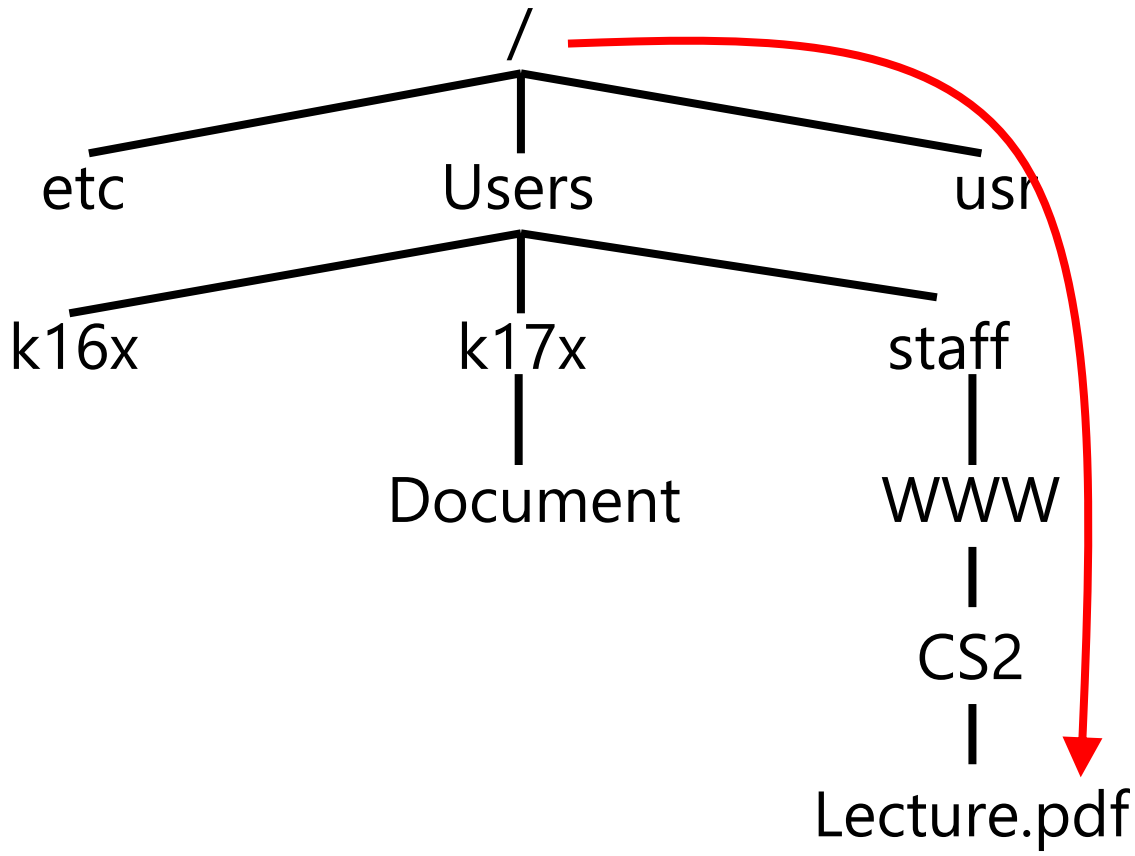
人間科学科コミュニケーション専攻
白銀 純子

今回の内容

- アプリケーションの種類
- 情報ネットワーク

設問1

- Lecture.pdfの絶対パス



解答: /usr/staff/WWW/CS1/Lecture.pdf

設問2

- コンパイル型(ソースコードを翻訳するタイプ) と、インタプリタ型(ソースコードを通訳するタイプ)のプログラムでは、どちらの実行速度が速いか考えなさい
  - ヒント: 手紙を自分で読む場合と、誰かに読み聞かせてもらう場合は、どちらが速い?

解答: コンパイル型(コンピュータが自分で読むタイプ)の方が速い

設問3

- オープンソースのアプリケーションについて、どんなメリット・デメリットがありそうか、考えなさい
  - 不具合の修正の早さ以外で

解答例(メリット):

- 無料で使えるものが多い
- プログラミングができる人であれば、自分でカスタマイズができる
- 様々な人が見るので、比較的品質が良い

解答例(デメリット):

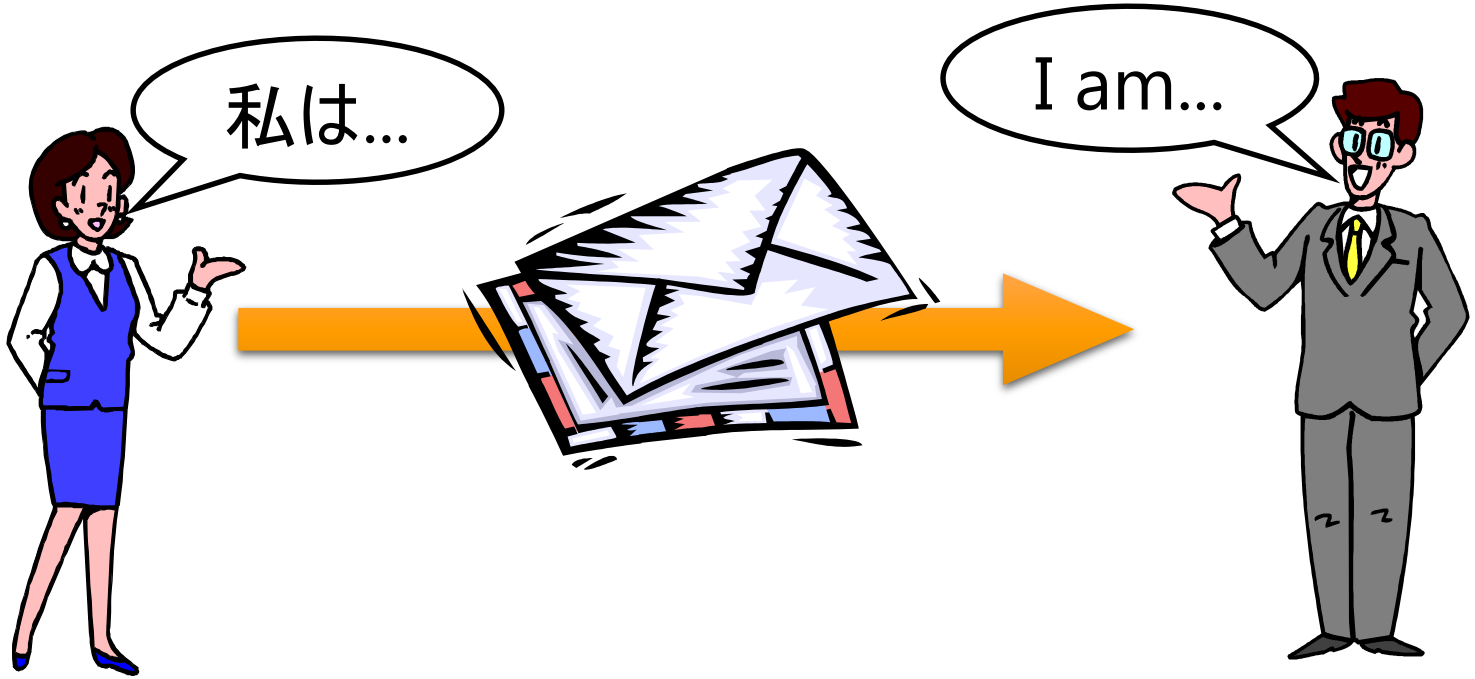
- トラブルが起こったときに責任を取ってもらえない
- マニュアルなどが整備されていないことが多い
- メンテナンスをきちんとしてもらえないことがある(特にマイナーなソフトウェアの場合)

前回の質問の回答

前回の復習

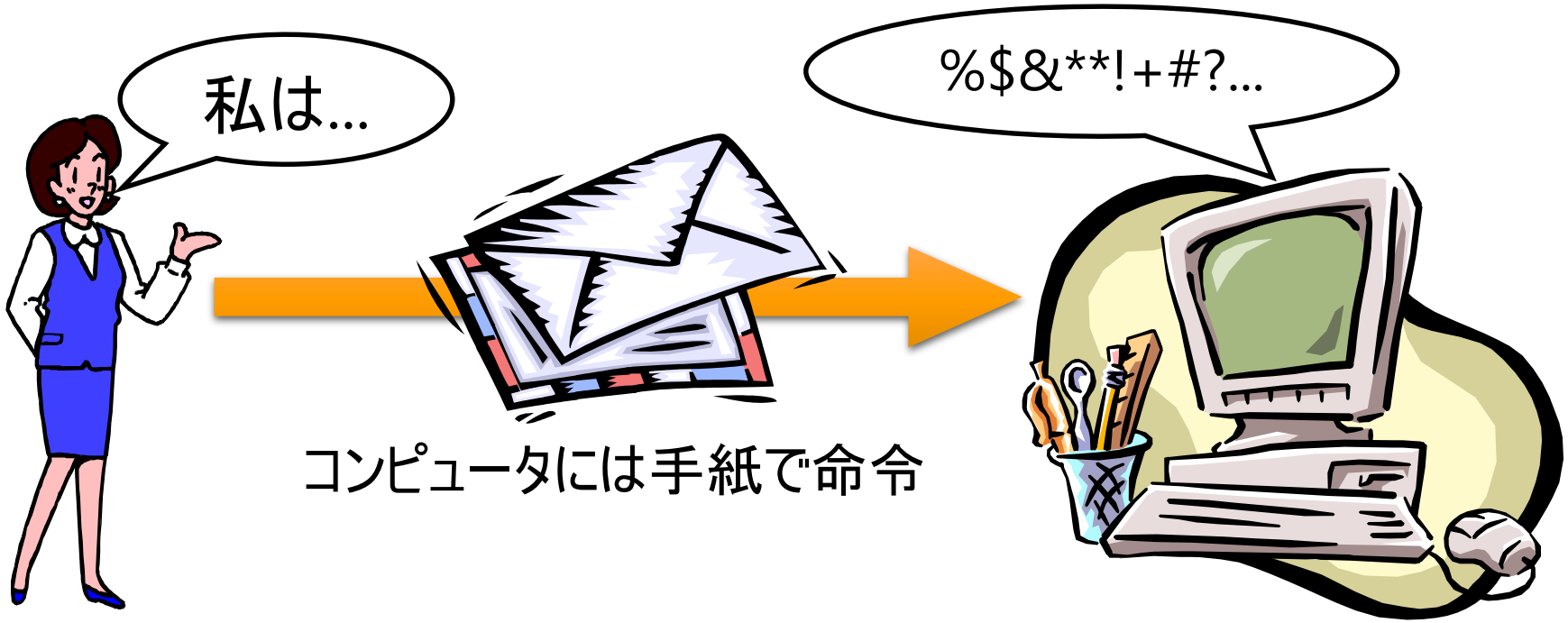
外国人に手紙を書く場合どうする??

- 相手がわかる言葉で手紙を書く
  - 相手が理解できる言葉を覚えるのは大変!!
- コンピュータには、手紙(命令書)で命令
  - コンピュータが理解できる言葉で手紙(命令書)を書く



外国人に手紙を書く場合どうする??

- 相手がわかる言葉で手紙を書く
  - 相手が理解できる言葉を覚えるのは大変!!
- コンピュータには、手紙(命令書)で命令
  - コンピュータが理解できる言葉で手紙(命令書)を書く



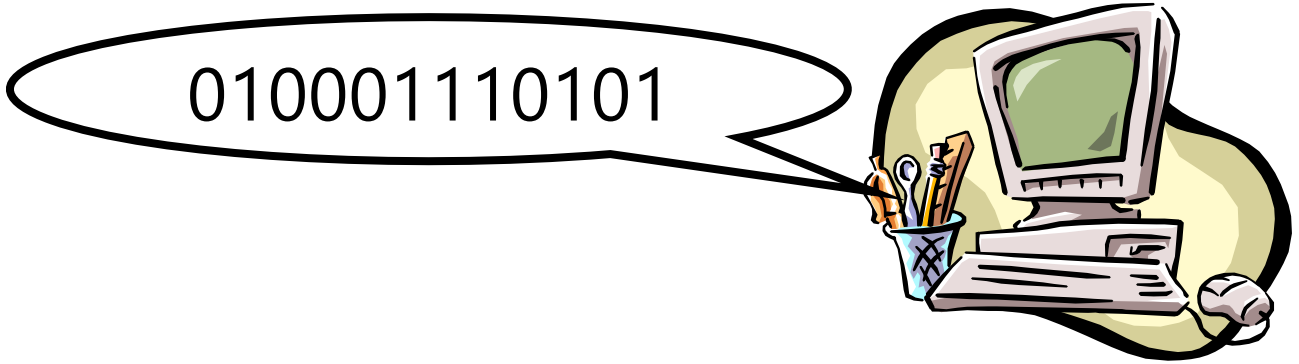
コンピュータが理解できる言葉は?

- コンピュータが理解できる言葉: 機械語
  - コンピュータは、2進数しか理解できない

人間が理解するのは難しい

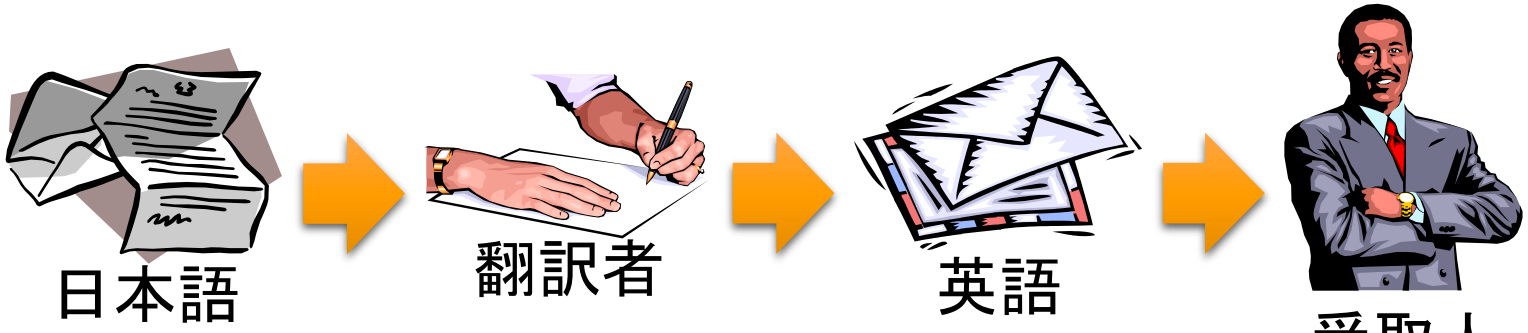
命令書を人間が理解できる言葉で書き、それを訳したものをコンピュータに渡す

プログラミング言語



手紙を訳すには?

- 手紙を翻訳する



- 手紙を通訳する

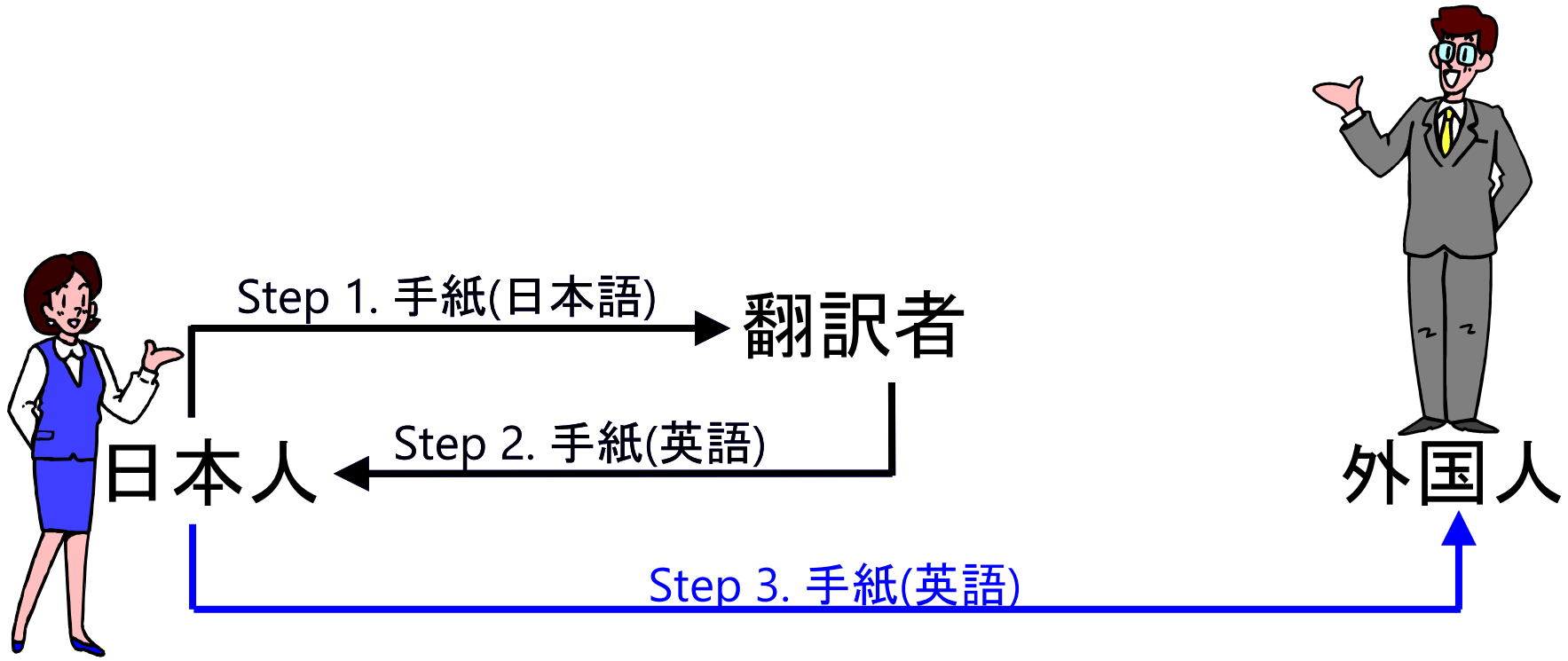
プログラムも、同じように機械語に訳す

変換方式

- プログラミング言語で書かれた命令書: 機械語に変換しなければ、コンピュータは実行不可能
- 変換方式は大きく分けて2種類
  - コンパイラ型: 翻訳
  - インタプリタ型: 通訳

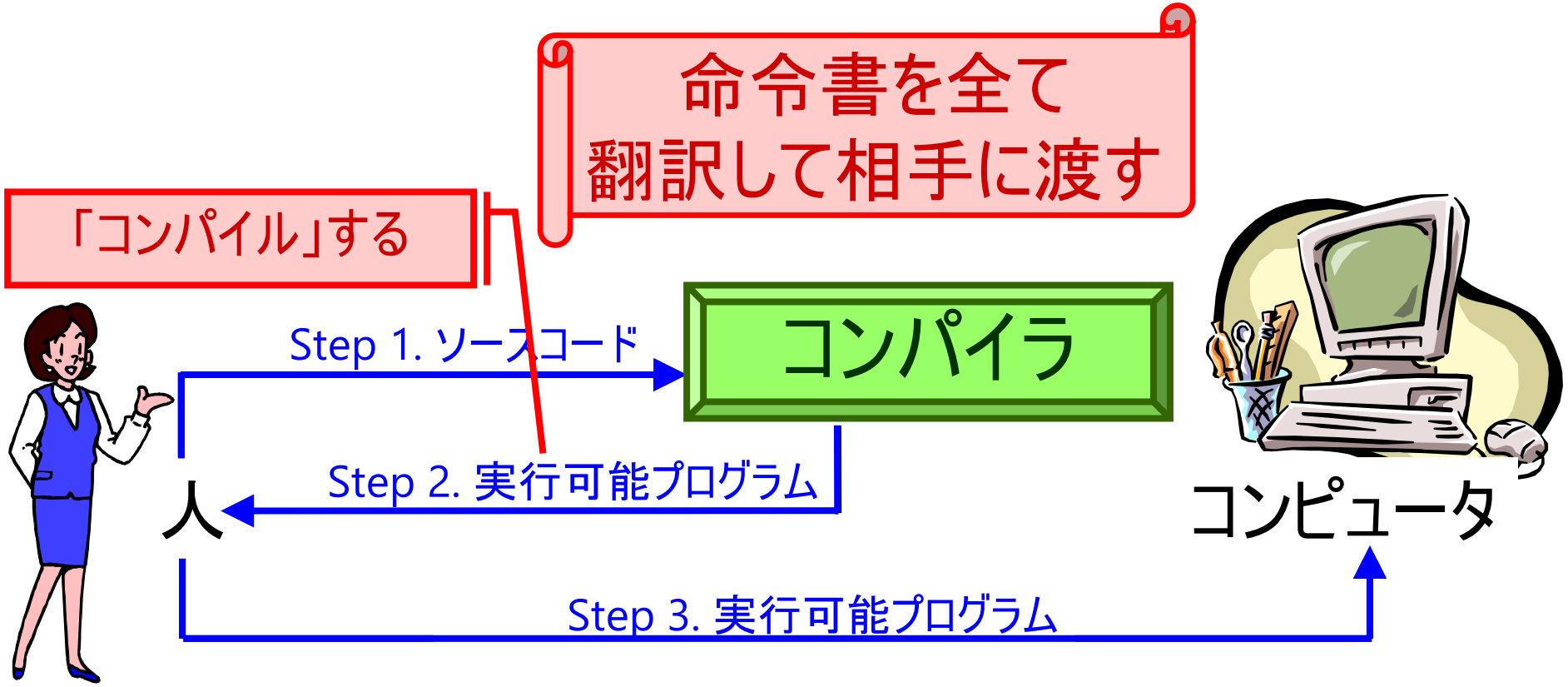
コンパイラ[概要](p. 78)

- コンパイラ: 命令書を機械語に翻訳し、コンピュータで実行可能にするためのソフトウェア

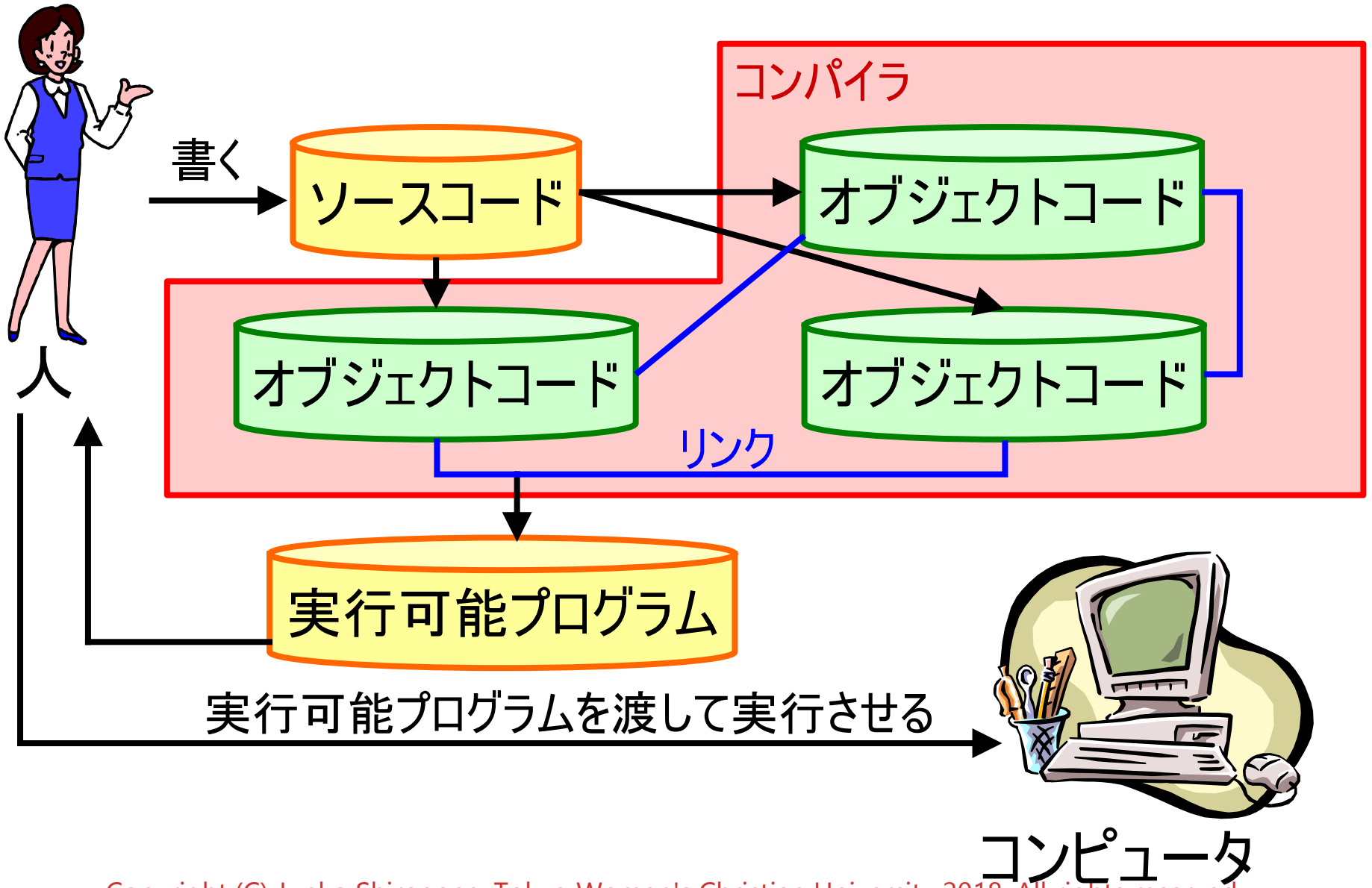


コンパイラ[概要](p. 78)

- コンパイラ: 命令書を機械語に翻訳し、コンピュータで実行可能にするためのソフトウェア

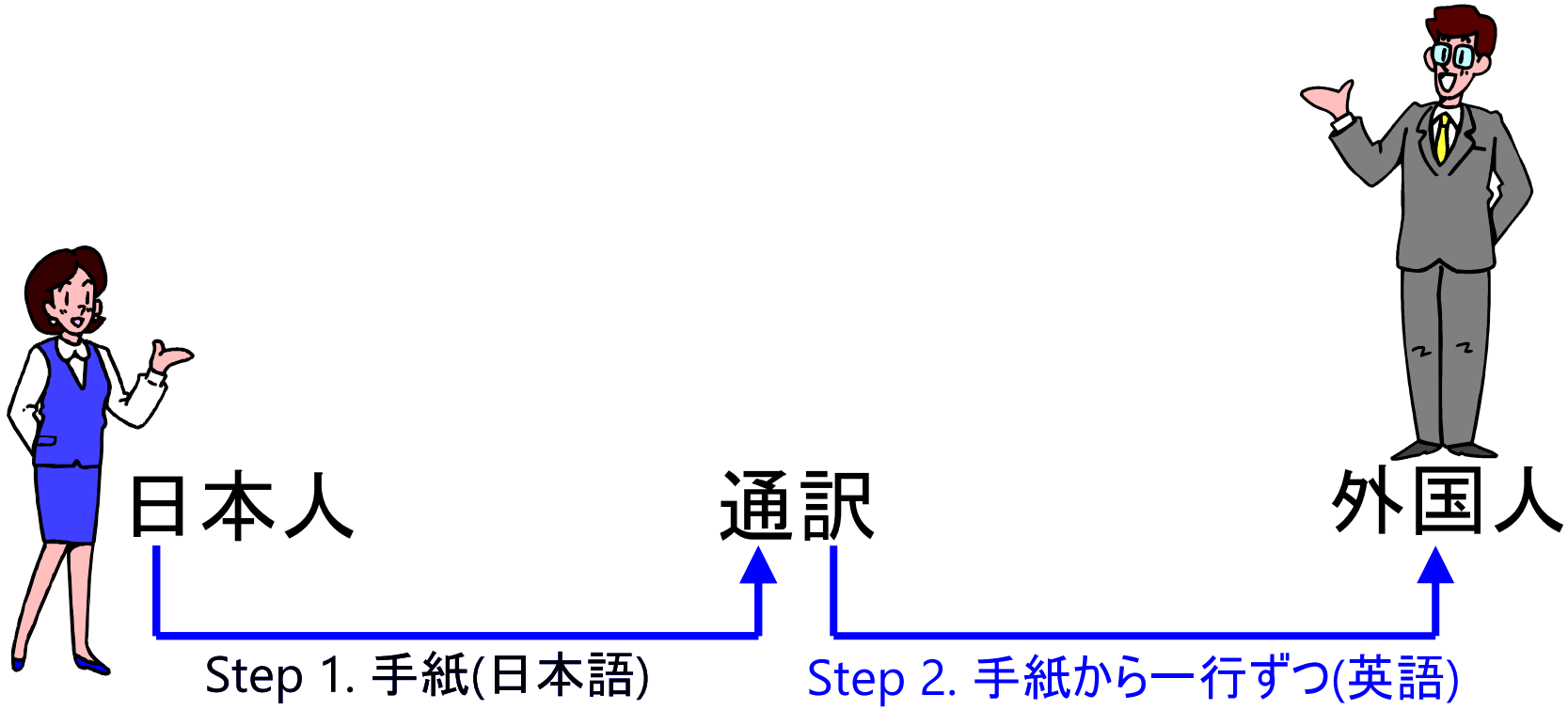


コンパイラ[詳しく](p. 78)



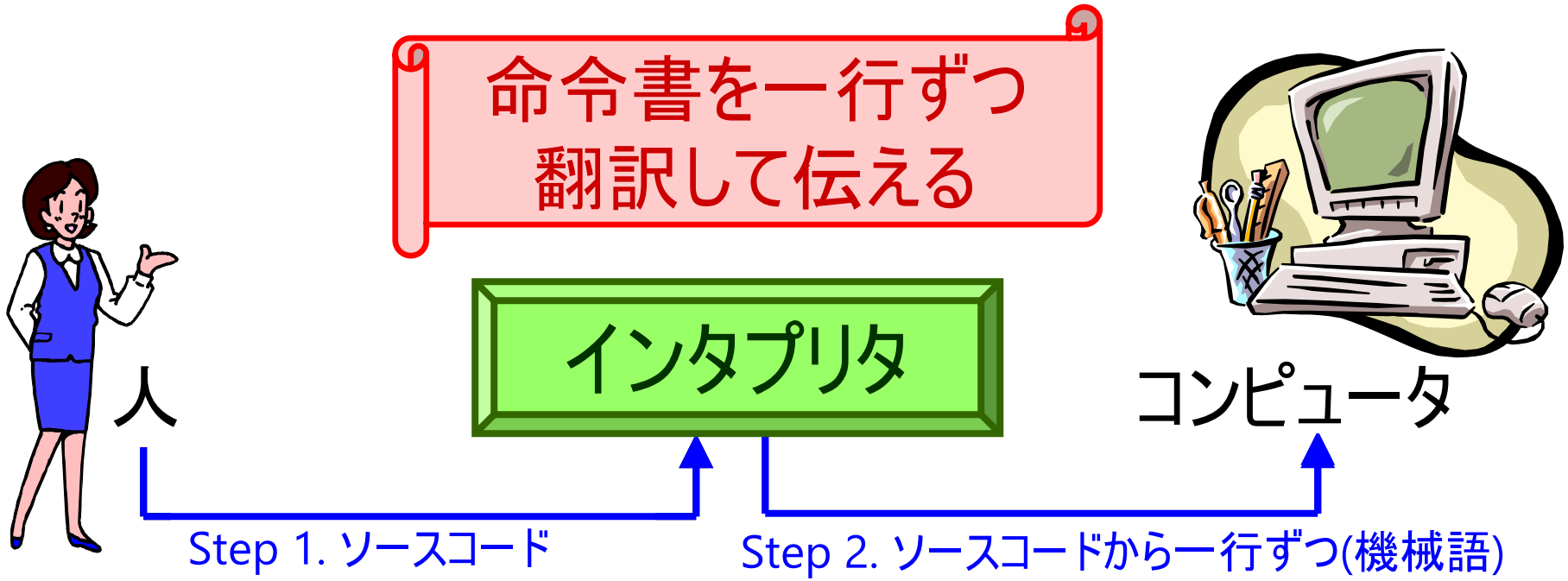
インタプリタ

- インタプリタ: 命令書を最初から1行ずつ読んで機械語に通訳するためのソフトウェア



インタプリタ

- インタプリタ: 命令書を最初から1行ずつ読んで機械語に通訳するためのソフトウェア



オープンソース(p. 84)

- オープンソース: ソースコードをインターネットで公開して、誰でも編集できるようにしたもの
  - 様々な人の知恵を集めて良いものを作る、という考えのもとにできた仕組み
    - オープンソースの場合、様々な人がソースコードを見るので、不具合の修正が早い(自分で修正することも可能)
    - オープンソースでない場合、ソフトウェアの作成者が不具合を修正するのを待つ
  - ソフトウェアの著作権は保護
  - 多くの場合、複数のプラットフォームに対応
    - プラットフォーム: OSやハードウェアなどのコンピュータの環境

オープンソースのOS(p. 84)

- Linux
  - 1991年に開発されたOSのカーネル
    - カーネル: OSの中でも核となる機能をあつめたもの
  - UNIXに似たOS(UNIX like OS)
  - インターネット上でのサービスを提供するためのコンピュータで多く利用
  - 様々な人が手を加えて、現在では様々な形のもの(ディストリビューション)が存在
    - 操作性や管理方法などがそれぞれ異なるものが存在
    - この授業のOSインストール実習でも利用(Vine Linux)

オープンソースのアプリケーション(p. 85)

- ブラウザ上で実現されているもの
  - SNS(Social Networking Service)
    - Mixiのようなコミュニケーションツール
    - オープンソースソフトウェアを組み合わせて開発
  - Wiki
    - ブラウザを利用して文書を書き込めるシステムで、意見交換やデータの共有などに利用
    - Wikipediaが代表的な例
  - etc.
- ブラウザ上以外で実現されているもの
  - gimp(グラフィックスソフト), OpenOffice.org, Mozilla Firefox(ブラウザ), etc.

Question!

情報ネットワーク

現代の情報ネットワーク[1](p. 89)

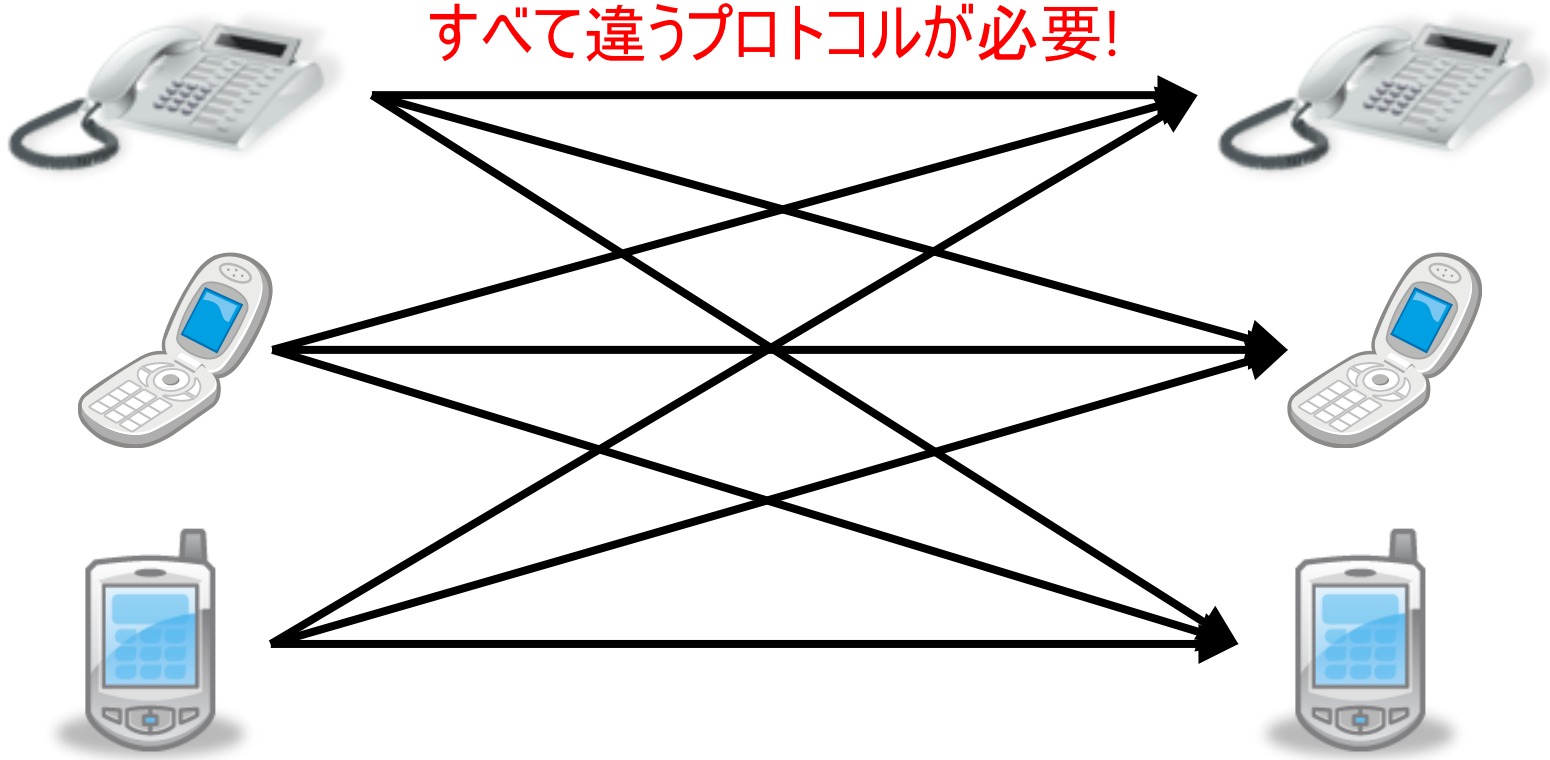
- 電気通信技術とデジタル化技術の組み合わせ
  - 文字・音声・画像などの情報をデジタル情報として扱い
  - 高速性・信頼性・経済性・利便性など、多くの利点
- 通信路(伝送路)によって、情報を伝達
  - 伝送媒体や通信機器などで構成
  - 多数の約束事(通信プロトコル)が必要
    - 通信路に正しく情報をのせ、届いた情報の意味を正確に理解するため
  - 多種多様な情報を、情報ごとに目的の場所に届けるための交換機能も必要
    - どこに届けるかを識別し、選択するための仕組み

現代の情報ネットワーク[2](p. 90)

- 通信路
  - 銅線・光ファイバ・電波などの伝送媒体を適材適所で組み合わせ
  - 伝送媒体同士をつなぐ通信機器
- 交換機能
  - 複数の利用者が共通の通信路を共有し、伝送先に対応した通信路を選び、信号を通過させる仕組み
    - あらゆる通信相手との間に専用の通信路を設けるのは不可能なため
- 通信プロトコル(通信をするためのルール)
  - 電気信号を伝えるためのケーブルの材質やコネクタの形状
  - 信号の種類や大きさ、意味, etc.

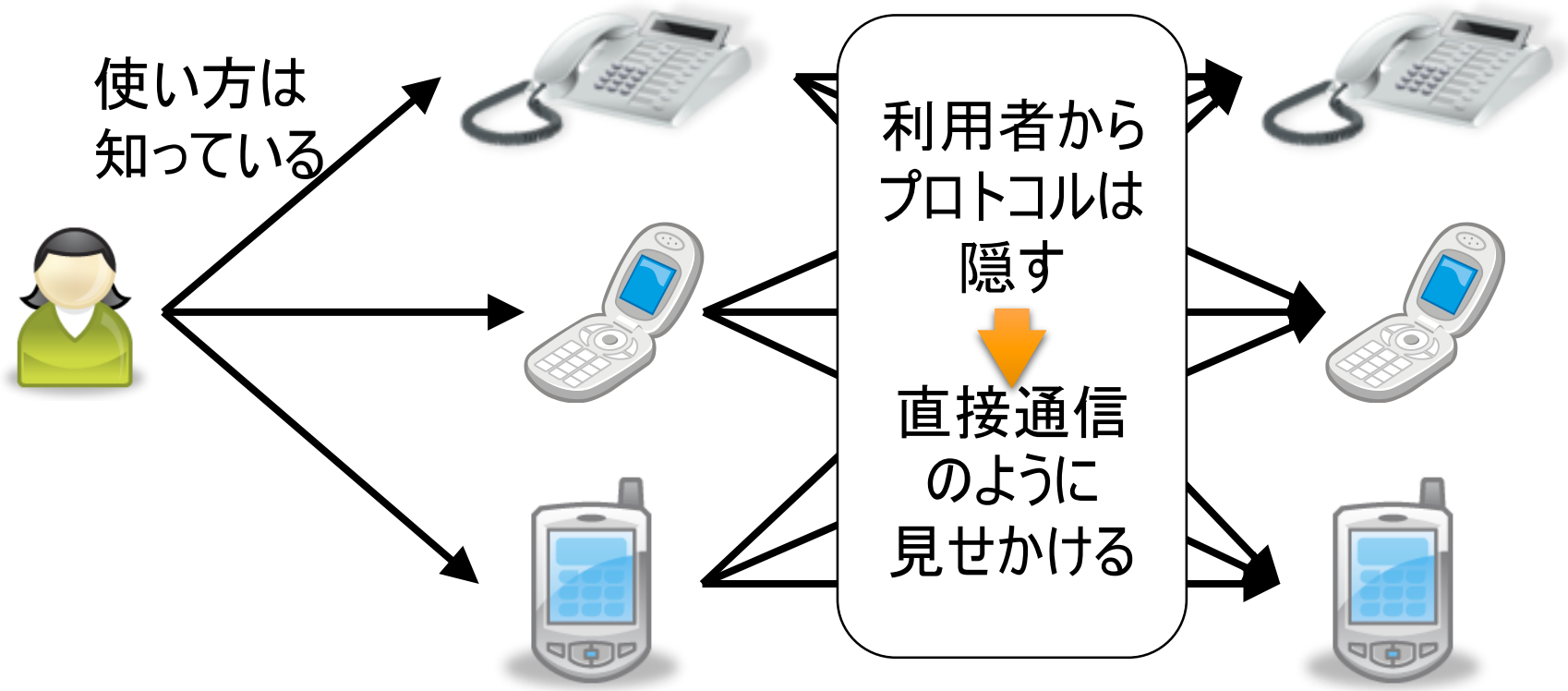
仮想化[1](p. 91)

- 利用者側として、利用する通信に必要なプロトコルをすべて使うのは面倒
  - Ex. 固定電話から固定電話へのプロトコル, 携帯電話から固定電話へのプロトコル, etc.



仮想化[2](p. 91)

- 自分の身近な部分だけ扱い方を知っていればOK
  - Ex. 電話のかけかた(電話番号のボタンを押す)
- それ以外の部分は、利用者からは隠して、統一化しているように見せかけ(仮想化)



階層化(p. 91)

- 正常に利用できているときには仮想化は便利
- 問題が起こった時や別の使い方を考えるときなどにはプロトコルの深い理解が必要

階層化

- ネットワークを機能別に分割し、それぞれを独立して扱えるようにする
  - ✓ プロトコルも機能ごとに分類する
- 各機能を順番に実行すれば、通信できるようにする

階層化により…

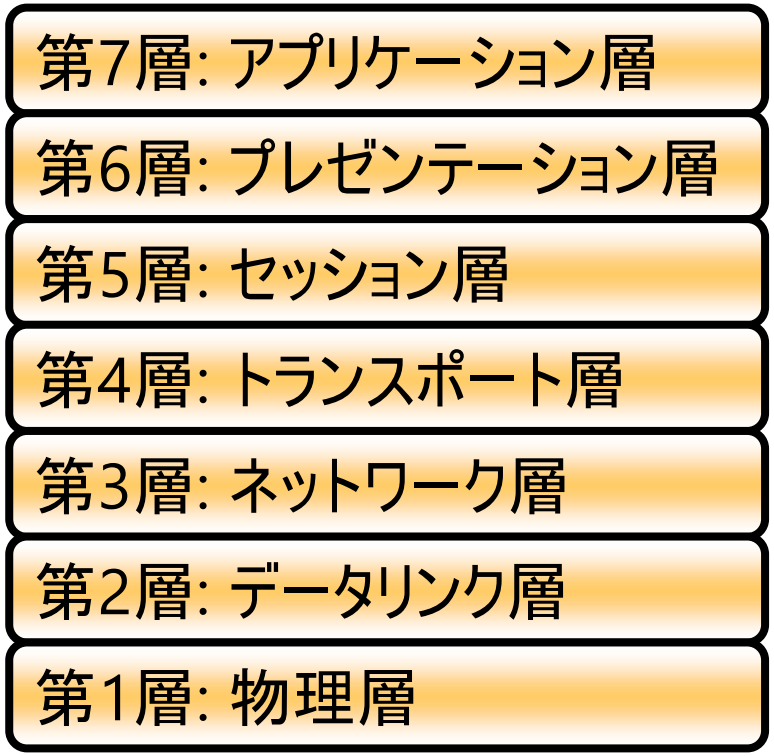
- 故障した時の機器の入れ替えなどがしやすくなる
- 機能の変更をする時に、その機能の前後の機能のみ注意すれば良い
  - ✓ 変更の影響を少なくできる

Question!

OSI参照モデル

OSI参照モデル(p. 94)

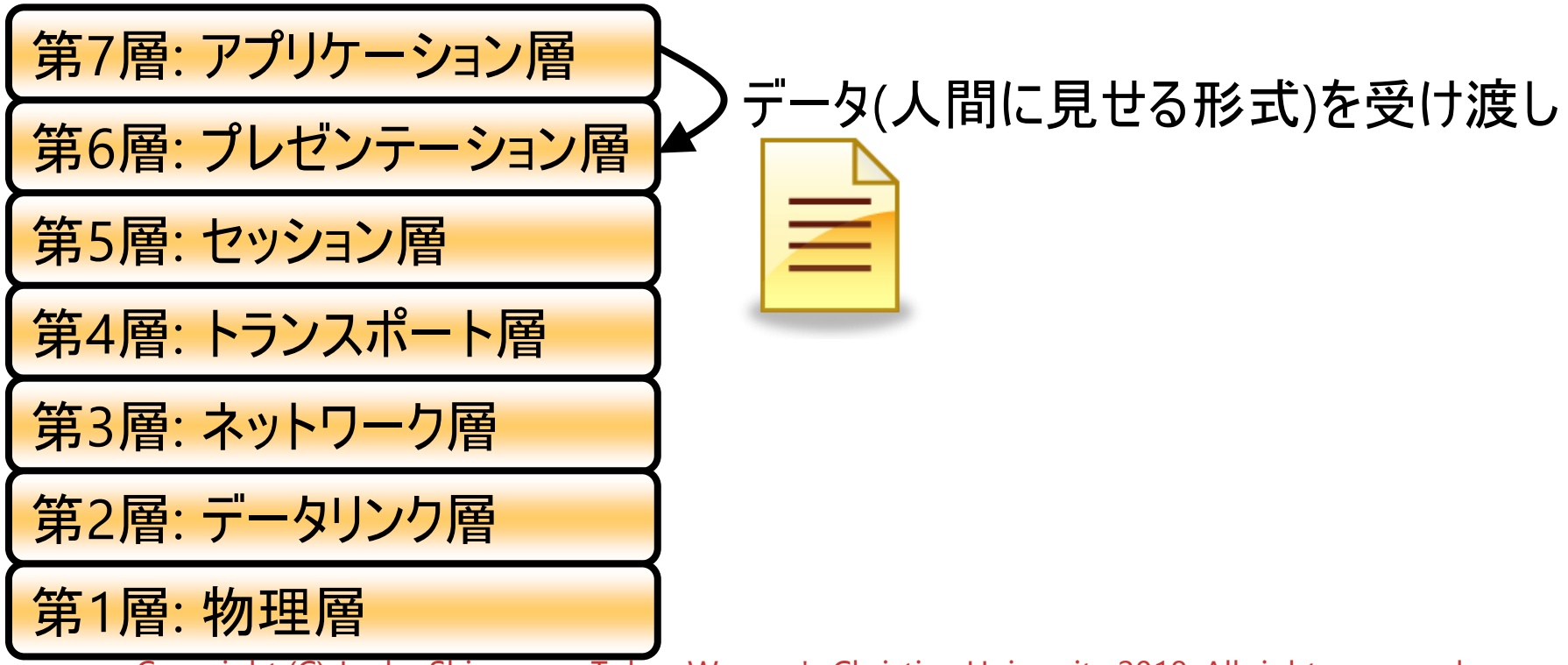
- コンピュータの通信機能を7つの階層に分割したモデル
  - 各階層ごとに必要なプロトコルを定義
  - 実際に使われるモデルは、OSI参照モデルをもとに規定



データの送信(p. 94)

データ送信[1](p. 94)

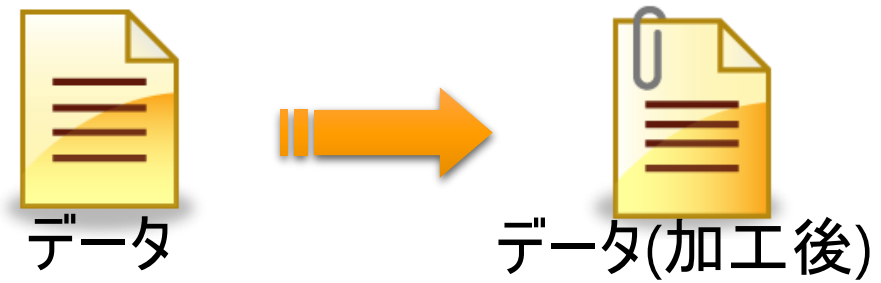
- アプリケーション層の役割: 人間が直接接するアプリケーション部分の規定
  - 具体的な通信サービスの提供(メールやWebなど)



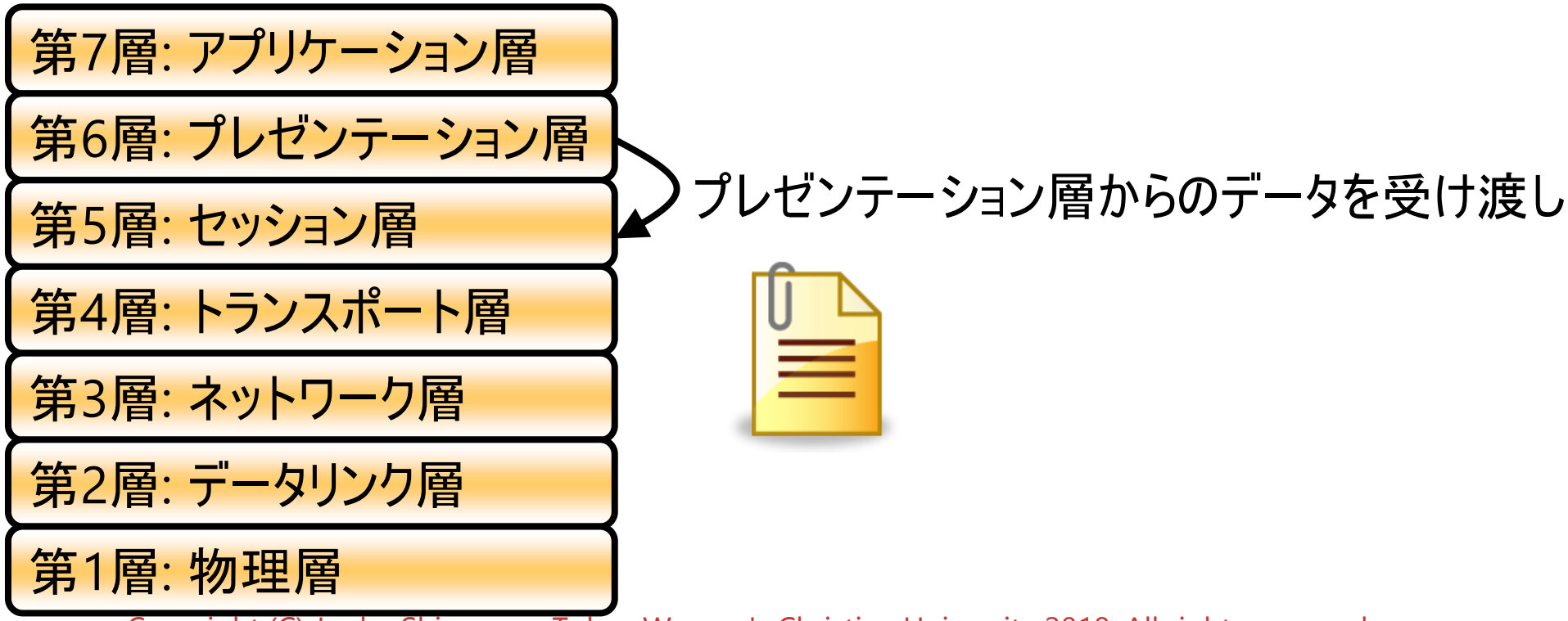
データ送信[2](p. 94)

- プレゼンテーション層の役割: データの表現形式の規定
  - データの圧縮・暗号形式や画像の形式、文字コード等に関する規定
  - データをネットワークで送信できる形式に変換
  - ネットワークから受信したデータをソフトウェアが理解できる形式に変換

第7層: アプリケーション層
第6層: プレゼンテーション層 → 暗号化や圧縮など、データを必要な形に加工
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

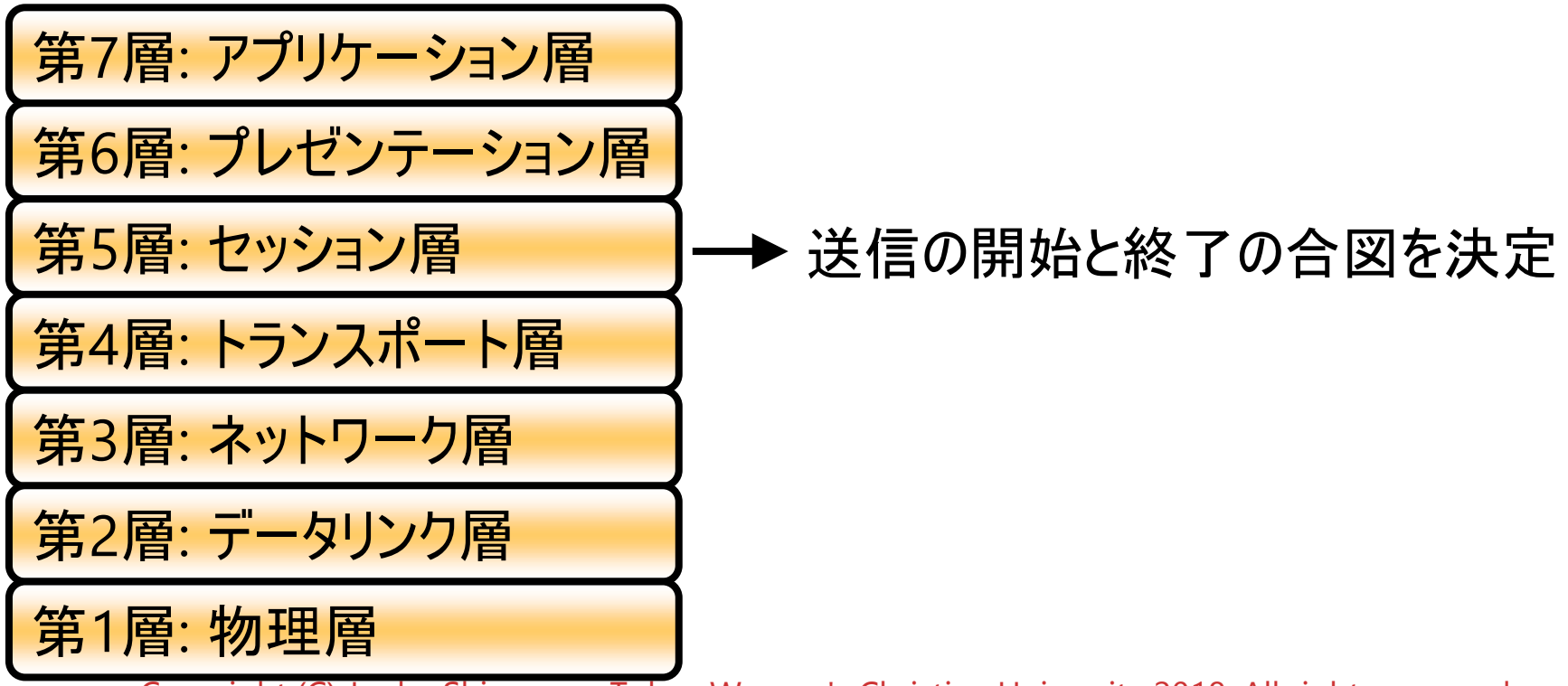


データ送信[3](p. 94)

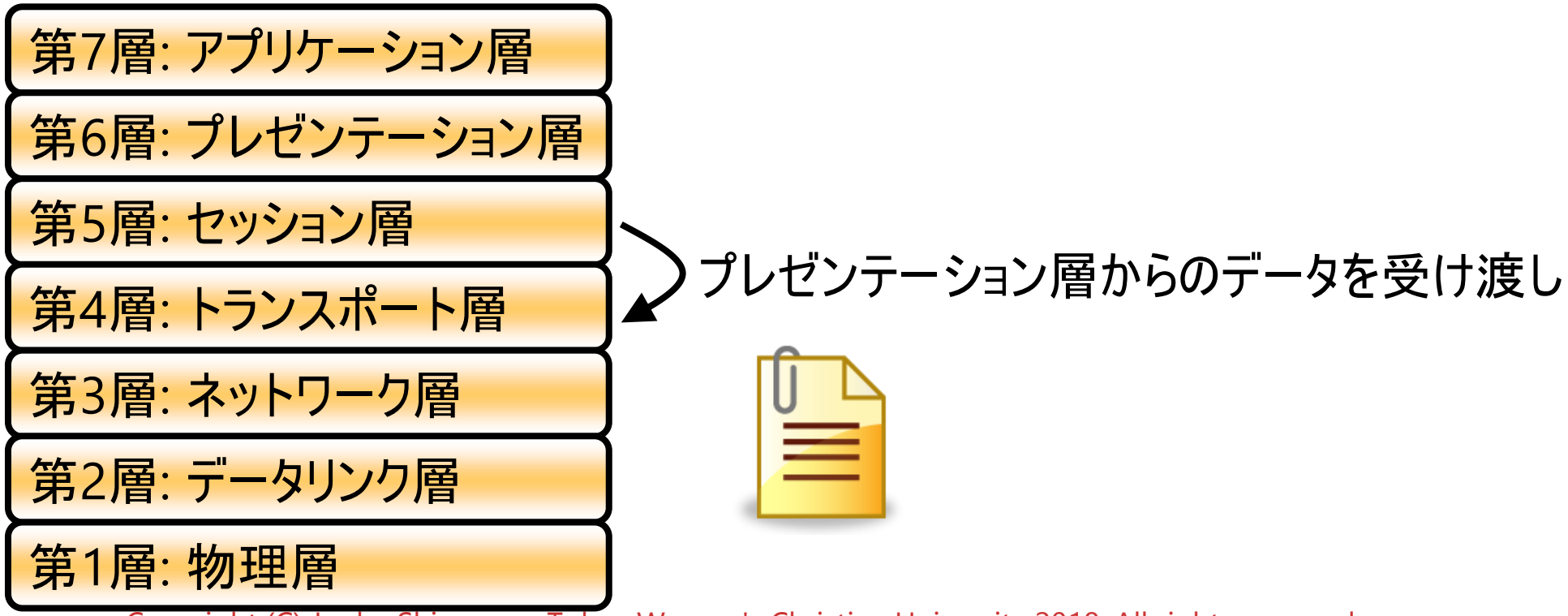


データ送信[4](p. 94)

- セッション層の役割: 通信の開始時・終了時の合図を規定
  - 通信の開始から終了までの手順
  - データ送受信のための経路の確保

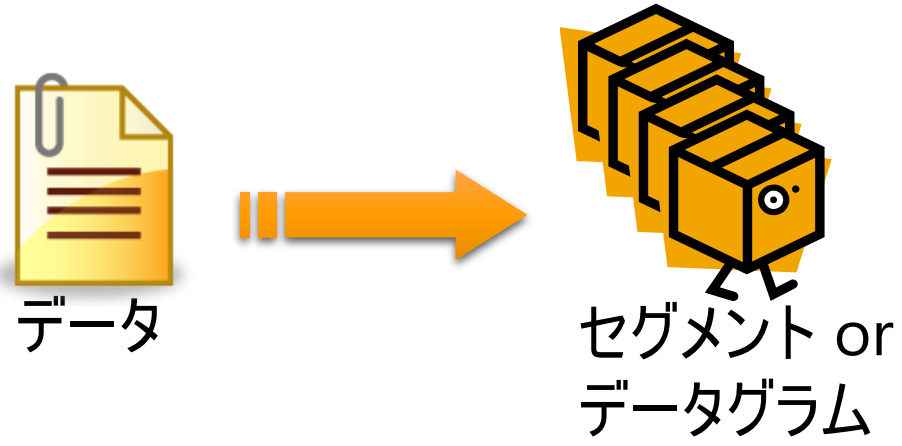


データ送信[5](p. 94)



データ送信[6](p. 94)

- トランスポート層の役割
  - データ送受信の信頼性を確保
    - データ内容にエラーがないかをチェックし、あれば是正や再送
  - データをネットワークで送信できる大きさに分割

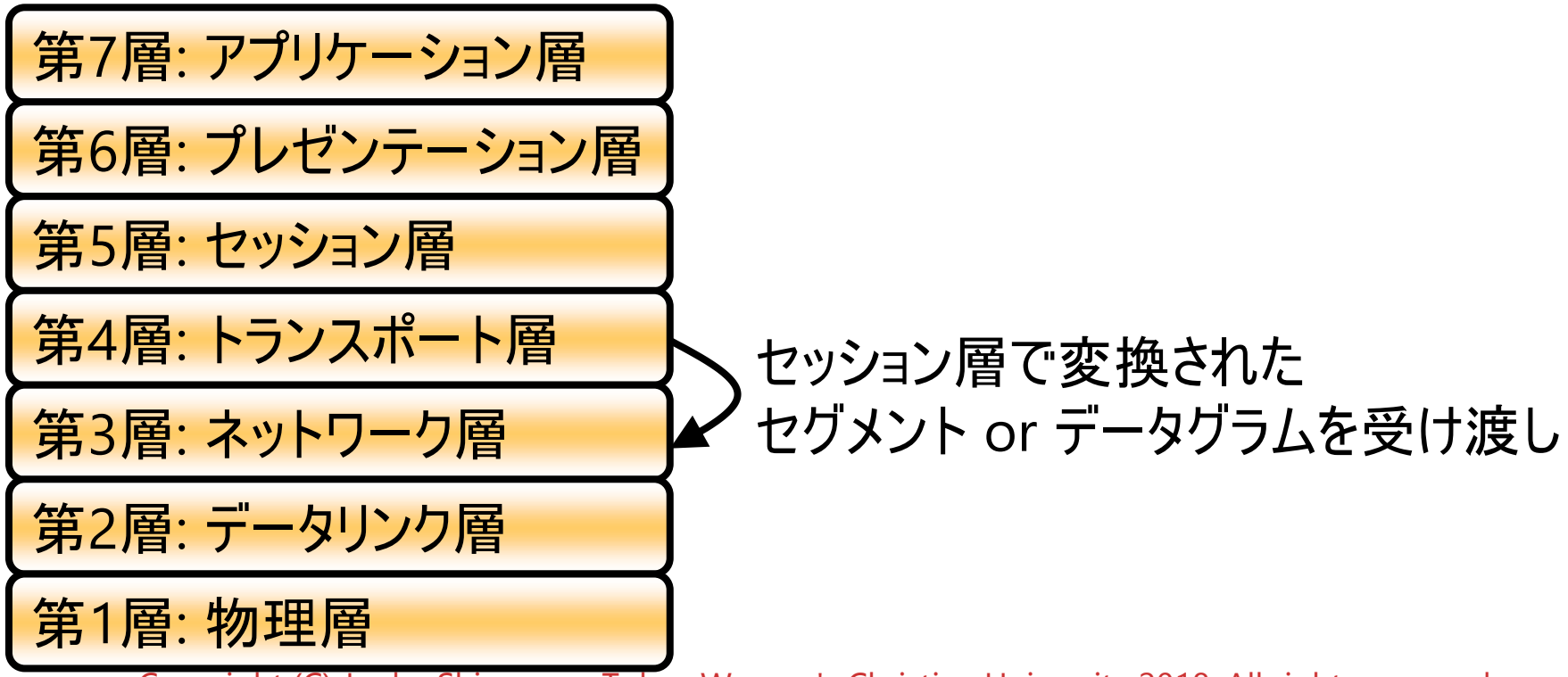


第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

→ セッション層からのデータをネットワークで送信できる大きさに分割

分割された1つ1つのデータ: セグメント or データグラム
- セグメント: 高信頼の通信の時
- データグラム: 低信頼の通信の時

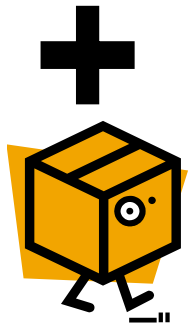
データ送信[7](p. 94)

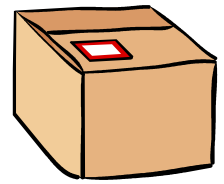


データ送信[8](p. 94)

- ネットワーク層の役割
  - データの宛先を特定して送受信
    - データに宛先の情報を付加し、送る経路の選択
  - ルータ(経路選択のための機器)などの規格を規定

送信元・あて先





パケット

セグメント or データグラム

セグメント or データグラムに送信元やあて先の住所を付加

- 第7層: アプリケーション層
- 第6層: プレゼンテーション層
- 第5層: セッション層
- 第4層: トランスポート層
- 第3層: ネットワーク層
- 第2層: データリンク層
- 第1層: 物理層

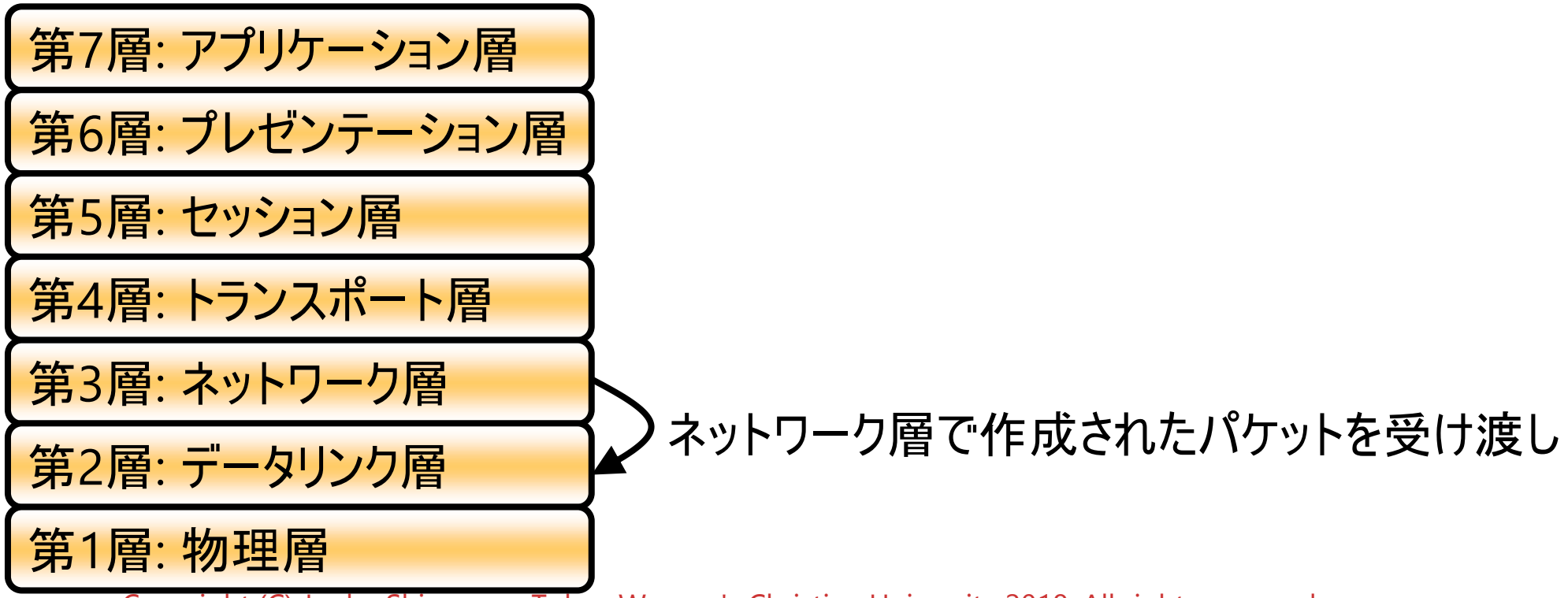
付加されたセグメント or データグラム: パケット

カプセル化(p. 93)

- 送受信するデータには、送受信のために必要な情報がつけられていない
  - 送信する宛先
  - 正しく届いたかどうかのチェック情報, etc.

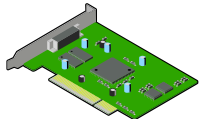
必要な情報をデータに付加すること: カプセル化

データ送信[9](p. 94)



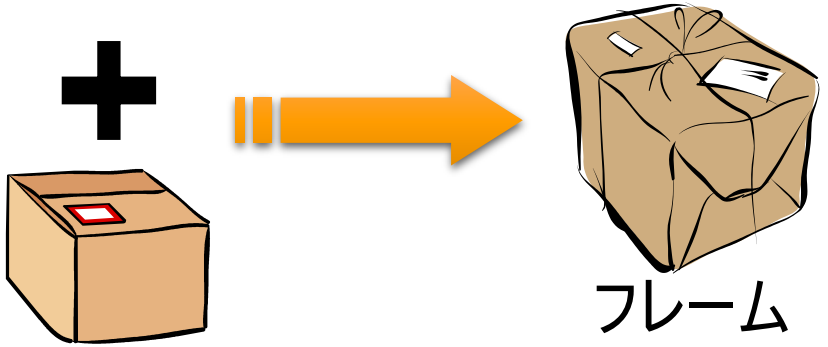
データ送信[10](p. 94)

- データリンク層の役割
  - 物理層での通信に誤りがないかをチェックし、誤りがあれば、再送信を要求
  - スイッチングハブ(コンピュータ同士を接続するための機器)などの規格を規定



MACアドレス
(ネットワークカードの固有の番号)

第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

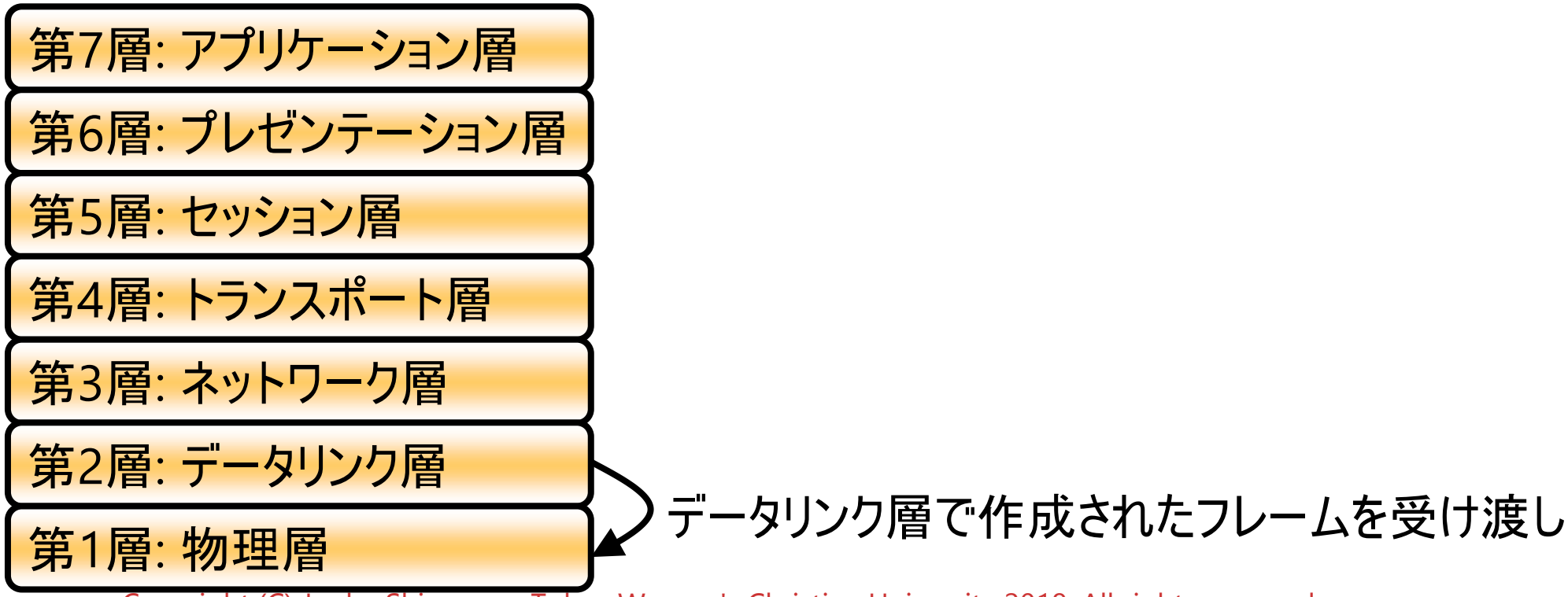


パケット　フレーム

→パケットに送信先のMACアドレスの情報を付加

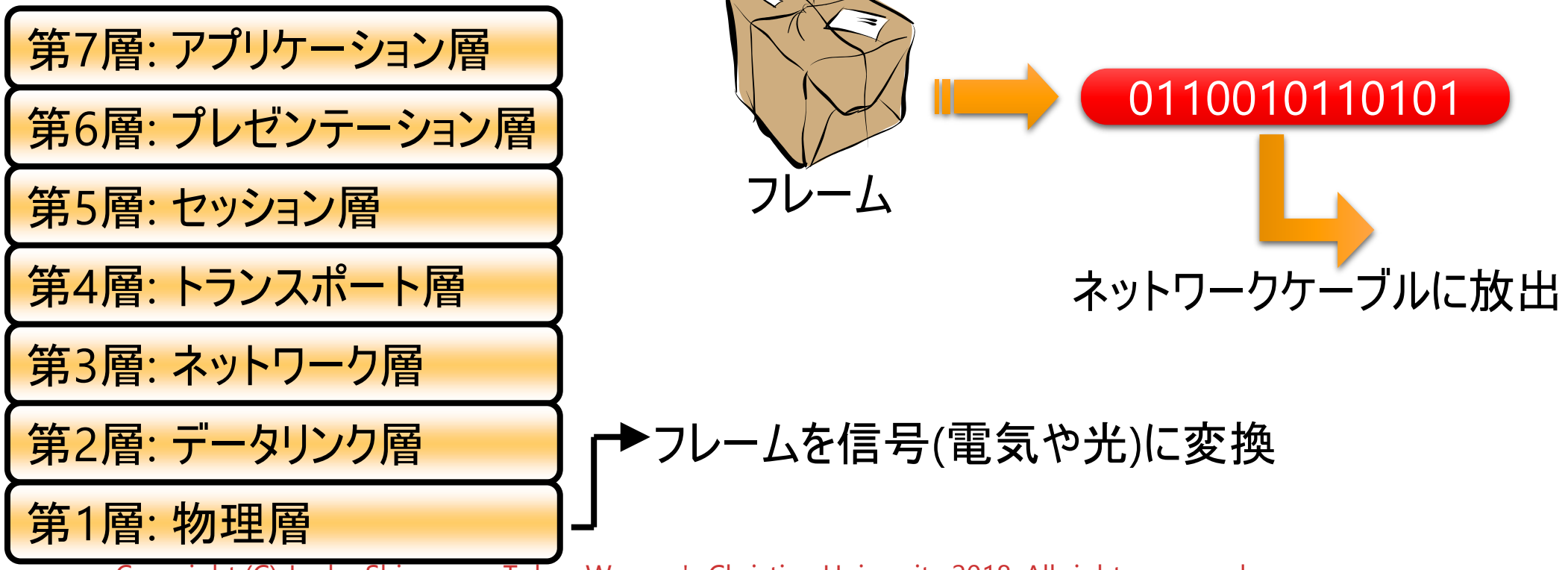
付加されたパケット: フレーム

データ送信[11](p. 94)



データ送信[12](p. 94)

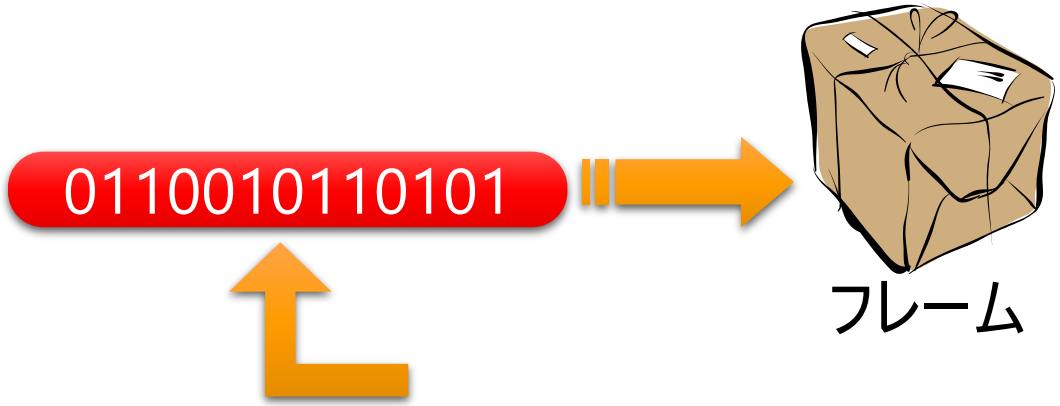
- 物理層の役割
  - ケーブルの規格を定め、やデータを電気・光信号の形で送受信
    - データと電気信号との間の変換
    - ケーブルに関する様々な規格(材質, コネクタの形状, etc.)



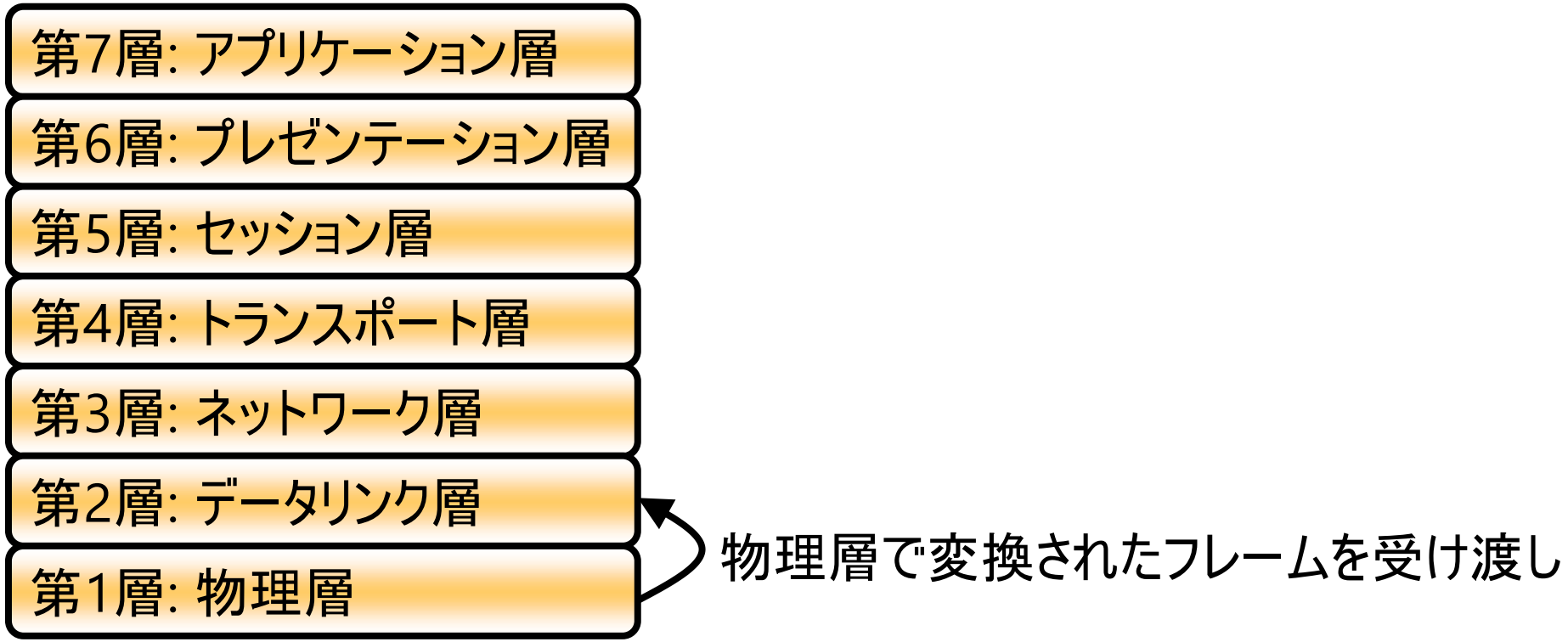
データの受信

データ受信[1](p. 94)

第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層 → 信号(電気や光)をフレームに変換

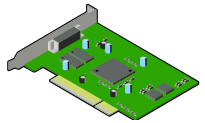


データ受信[2](p. 94)



データ受信[3](p. 94)

フレーム



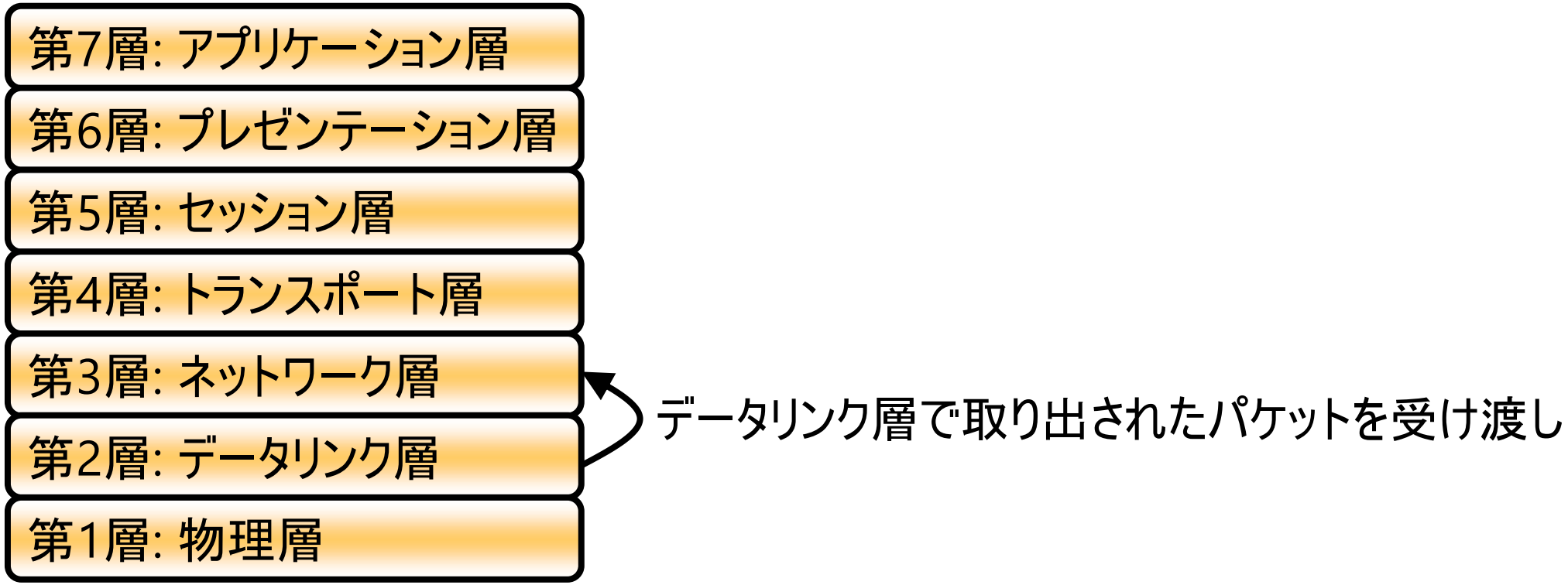
MACアドレス
(ネットワークカードに固有の番号)

パケット

第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

フレームから送信先のMACアドレスの情報を
削除してパケットを取り出し

データ受信[4](p. 94)



データ受信[5](p. 94)

パケット

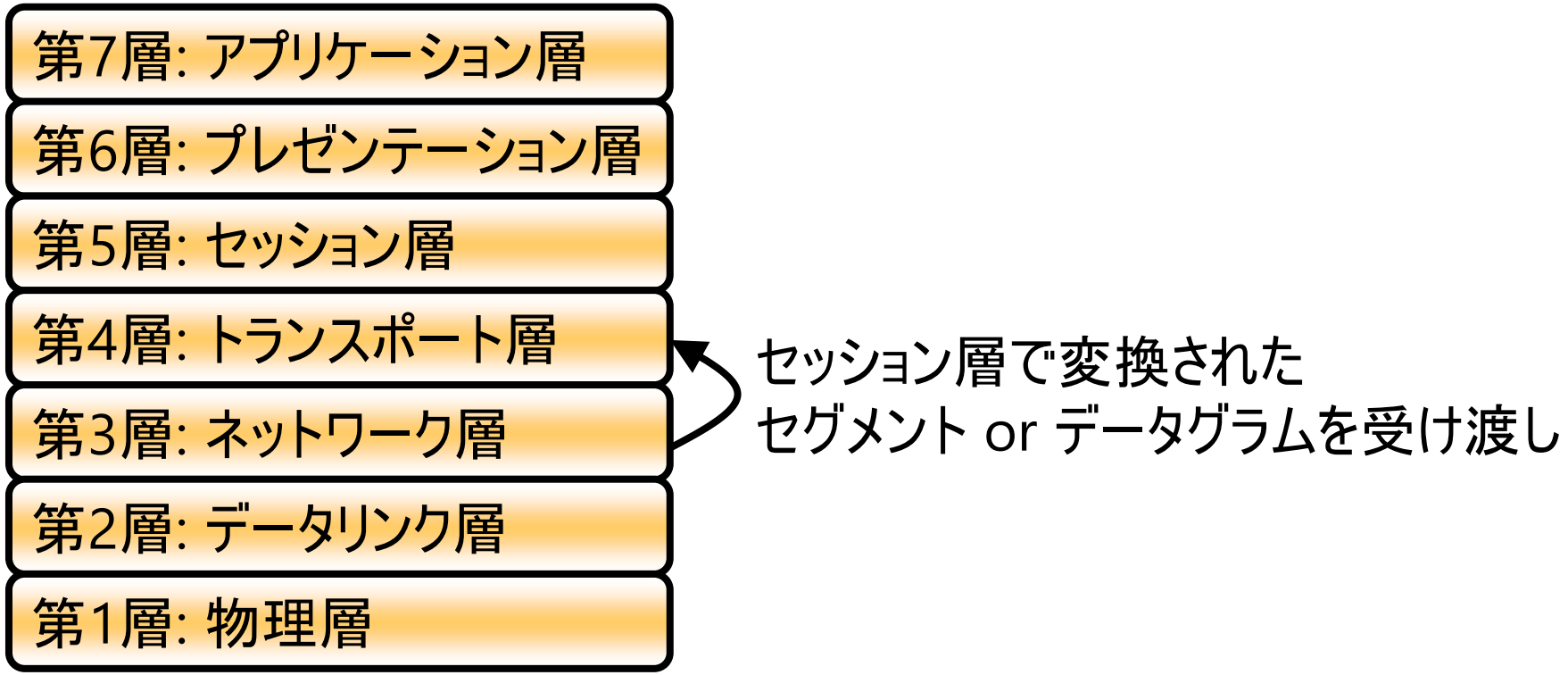
送信元・あて先

セグメント or データグラム

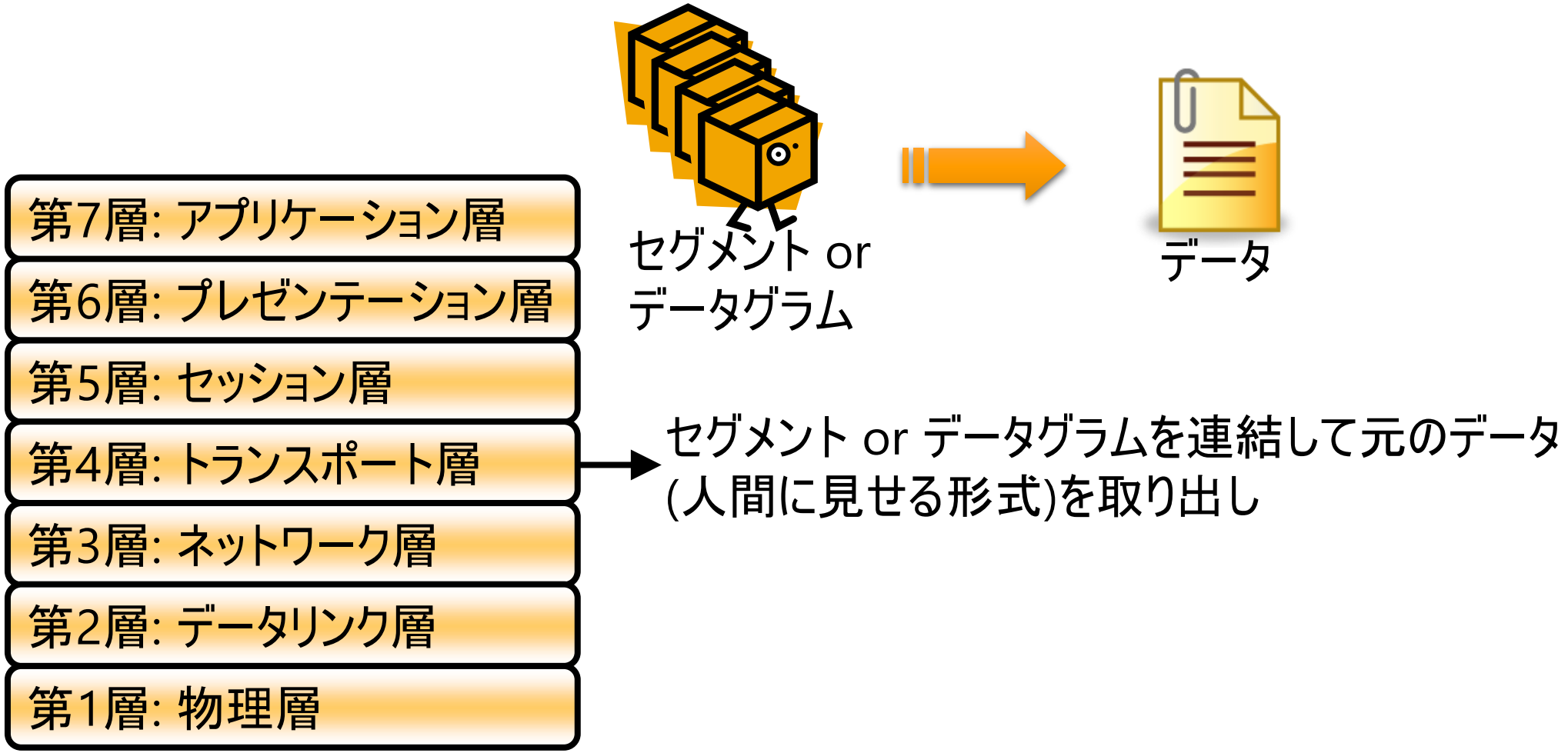
第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

パケットから送信元やあて先の住所を削除して
セグメント or データグラムを取り出し

データ受信[6](p. 94)

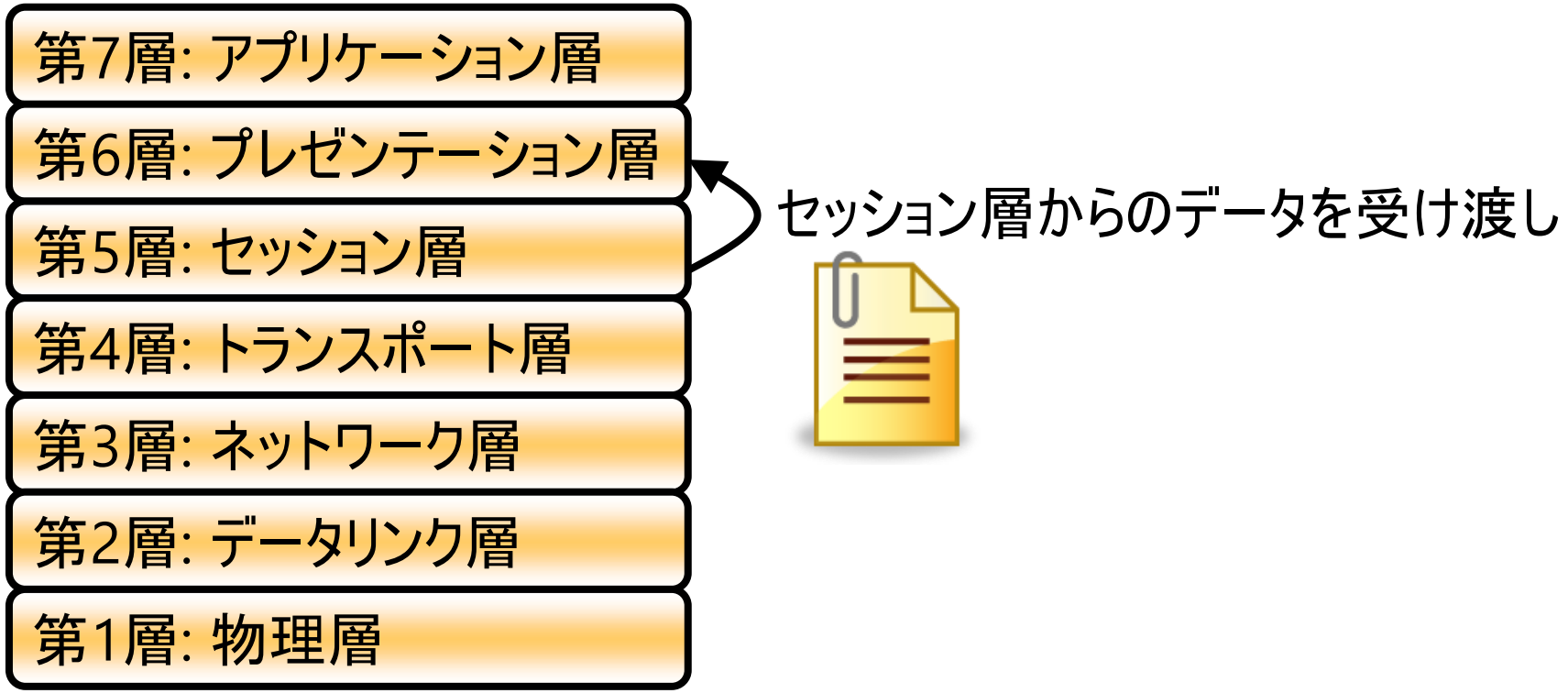


データ受信[7](p. 94)

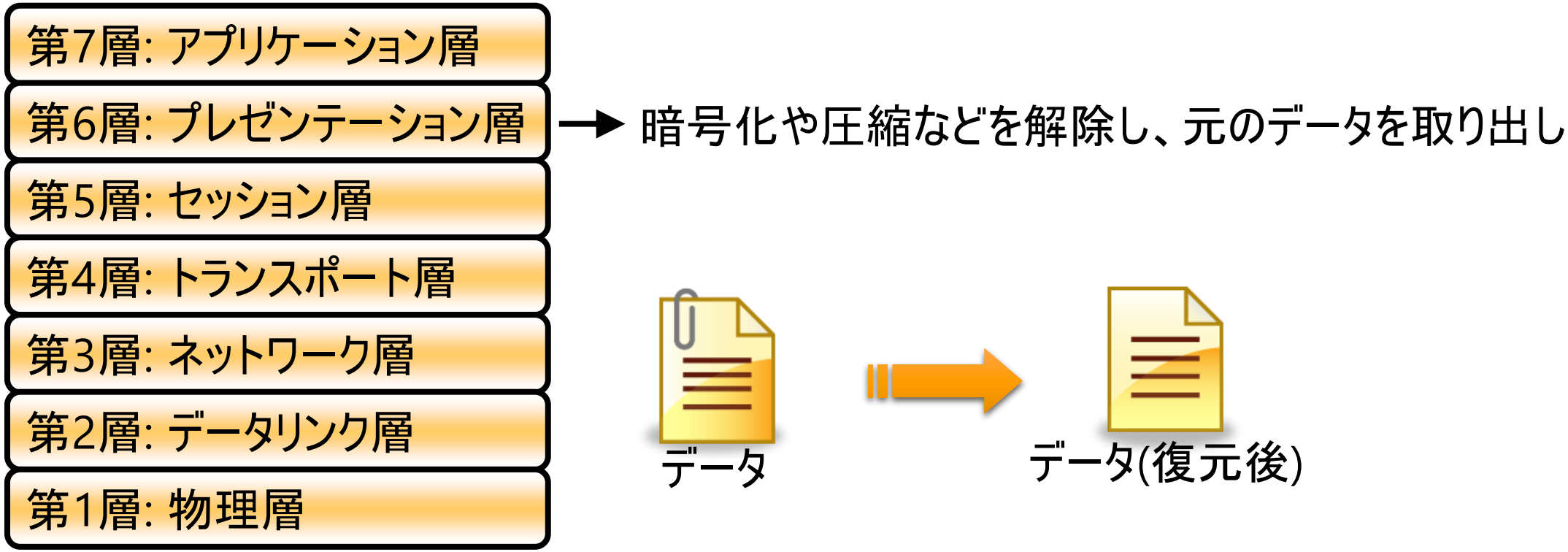


データ受信[8](p. 94)

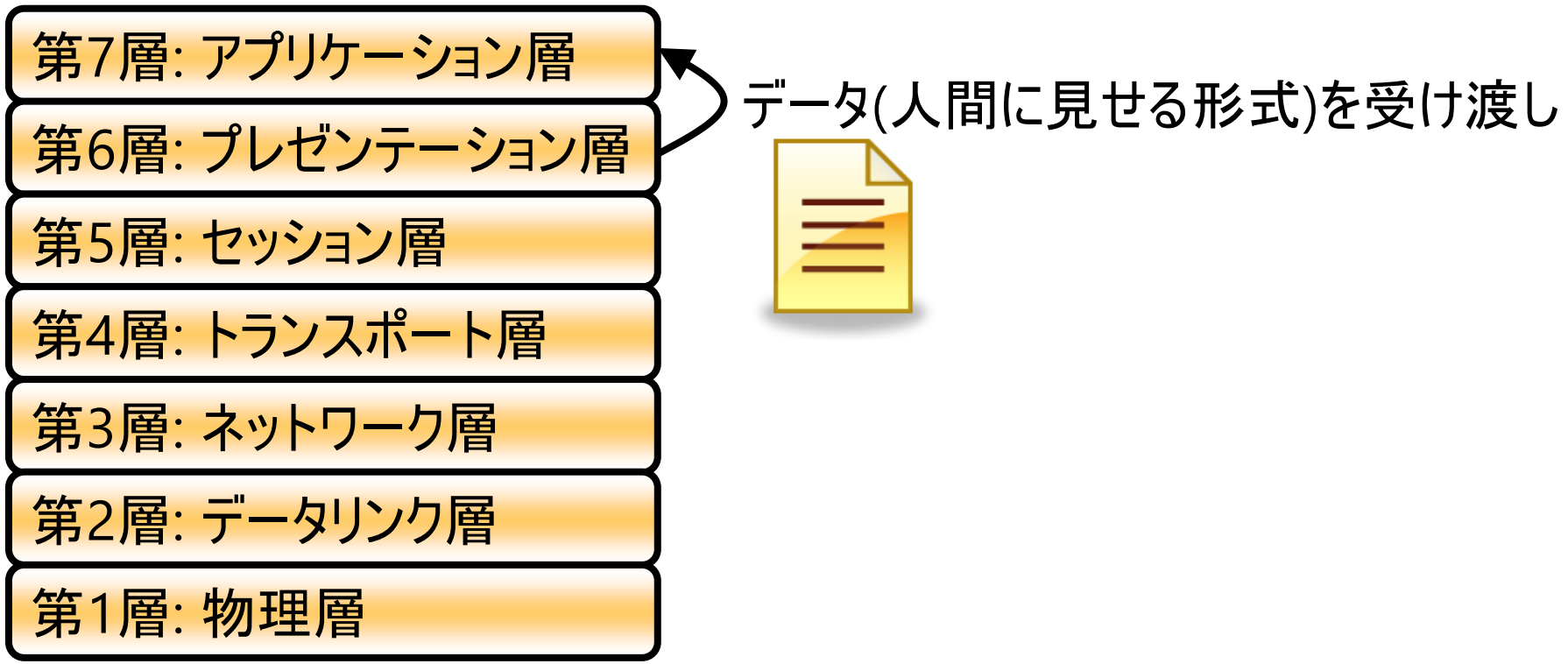
データ受信[9](p. 94)



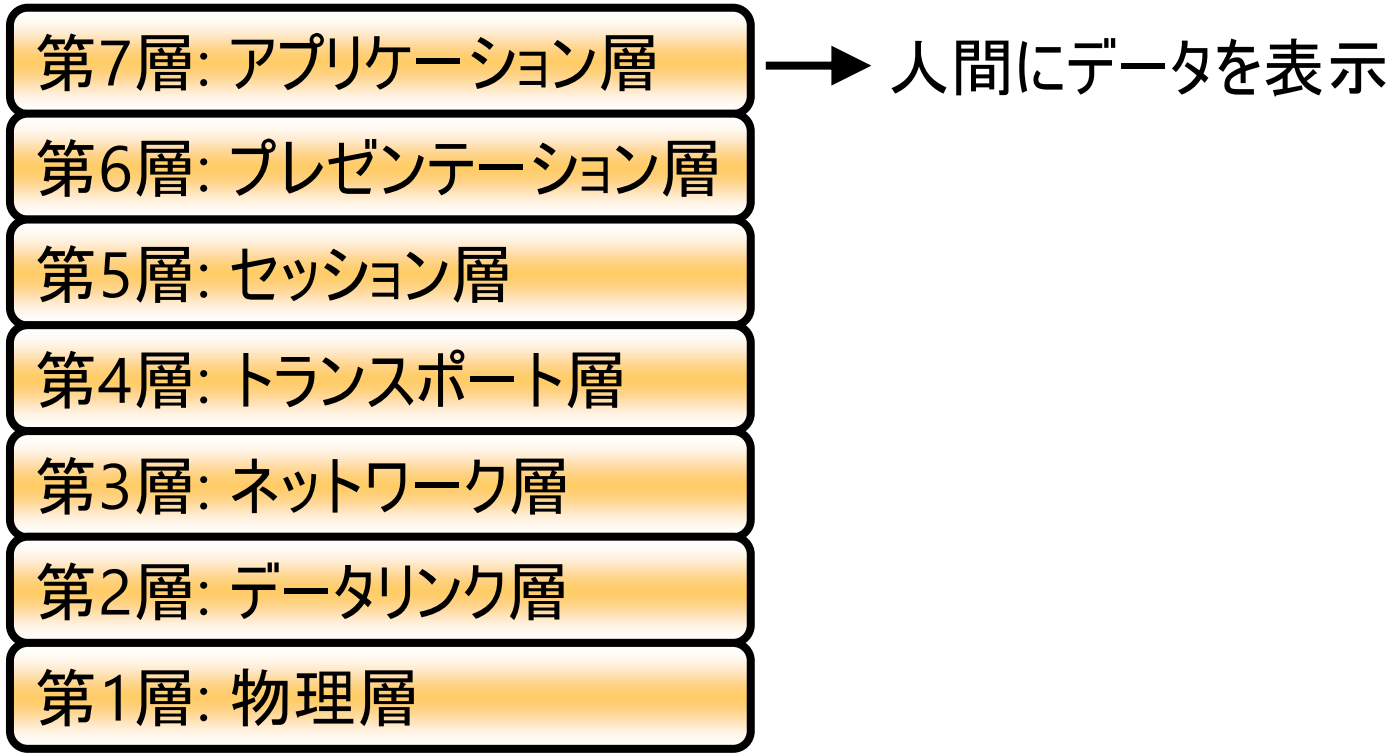
データ受信[10](p. 94)



データ受信[11](p. 94)



データ受信[12](p. 94)



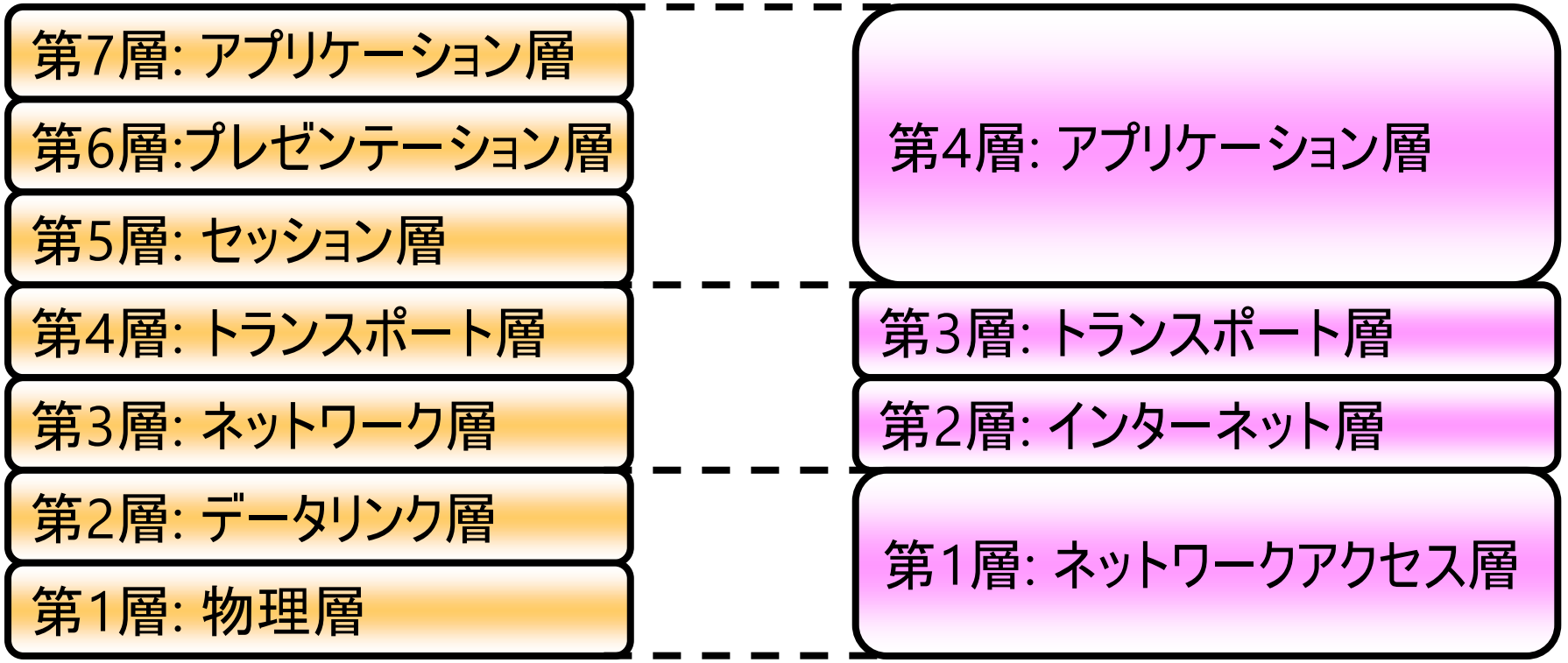
TCP/IPモデル

TCP/IPモデルとは?(p. 96)

- TCP/IP: データがインターネットを通るためのプロトコル
  - Transmission Control Protocol/Internet Protocol
  - インターネットでの標準規格
- コンピュータの通信機能を4つの階層に分割したモデル
  - 各階層ごとに必要な機能(プロトコル)を定義
  - 現在最もよく使われているモデル
    ※OSI参照モデルは、実際に利用するモデルの基礎

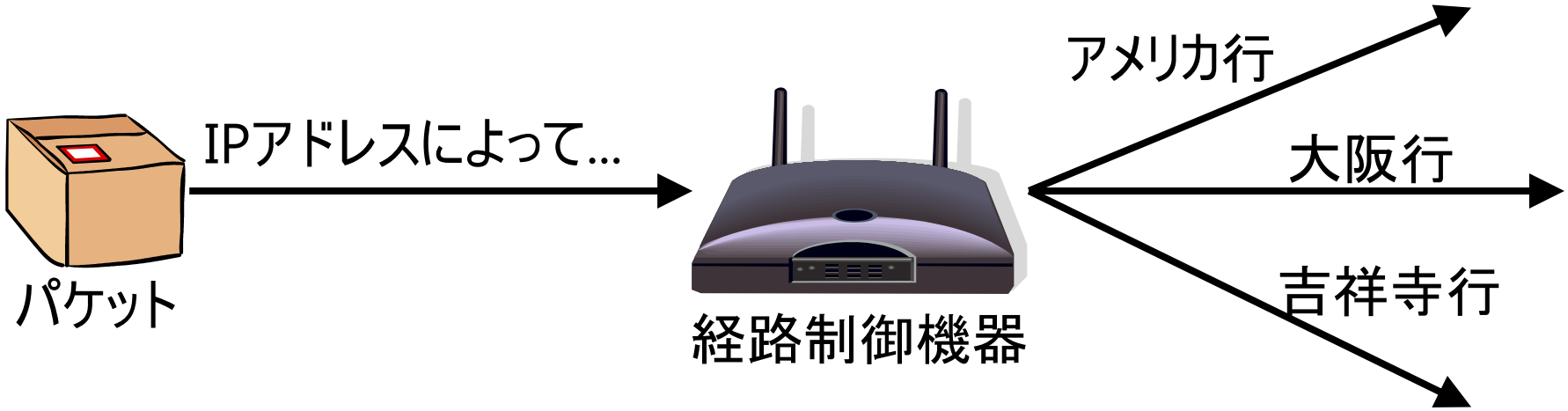
OSIとTCP/IPモデル(p. 96)

- OSIの7層とTCP/IPの4層との対応関係
  - OSI参照モデルと同じ名前の層があるが、必ずしも同じ役割をするわけではない



インターネット層のプロトコル(p. 96)

- IP
  - インターネットの世界でのコンピュータの住所(IPアドレス)を扱うためのプロトコル
    - 通信の宛先として指定される住所
  - IPアドレスに基づいて、送り先を決める経路制御機器で利用



トランスポート層のプロトコル(p. 96)

- TCPとUDP
  - TCP (Transmission Control Protocol)
    - データの通信前に、通信先との道筋を確保し、その上で送受信

コネクション型

  - UDP (User Datagram Protocol)
    - 道筋を確保することなく、いきなりデータを通信

コネクションレス型

TCPとUDP[データの受け渡し](p. 96)

- 上位の層からのデータの受け渡し
  - TCP: データを分割して受け渡し
    - 分割したデータを「セグメント」
  - UDP: データのかたまりをそのまま受け渡し
    - データのかたまりを「UDPデータグラム」
    - データのサイズがある一定以上を超える場合、さらに下位のネットワーク層で分割(IPフラグメンテーション)

TCPとUDP[信頼性](p. 96)

- TCP
  - 通信中にデータの紛失がないかを確認し、紛失があれば送りなおし
    - セグメントに番号をつけ、正しい番号のセグメントが届かなければ再送
    - タイムアウトすれば再送
    - 同じ番号のセグメントが届けば、重複を除去
    - セグメントの番号が順番どおりに届かなければ、順番どおりに並べ替え
- UDP
  - 通信中のデータの紛失については、何もサポートなし

TCPとUDP[シンプルさと軽さ](p. 96)

- TCP
  - 様々な処理をする必要があるので、複雑で遅い(重い)
- UDP
  - データの送受信以外のことをほとんどしないので、シンプルで速い(軽い)

TCPとUDP[使いどころ](p. 96)

- TCP
  - 大きなファイルの送信
    - 第7層か第3層あたりでデータを分割する必要があるが、送信確認をしないと、一部が紛失する可能性
- UDP
  - 小さなメッセージの送信
    - TCPを使うと、小さいデータに様々なものが付加されることになり、非効率
  - 実況中継
    - TCPを使うと、再送などがあってリアルタイム性が問題
  - ブロードキャスト
    - TCPを使うと、再送が起こったときに複数個所に再送が困難

Question!

デファクトスタンダード

デファクトスタンダード(p. 97)

- TCP/IPモデル
  - 階層化が不十分で厳密性が不足
  - but 最も広く使われていて、ネットワークの事実上の標準

デファクトスタンダード

デファクトスタンダード

- 市場で広く使われるようになったために、標準となること
  - ✓ 国際機関などが公的標準として定めたものではない
- 一度標準になると、関連する企画や商品が出て、さらに標準が地位を強化

標準化の流れ(p. 97)

- インターネット関連のプロトコル
  - RFC(Request For Comments)という文書により実現
    - IETF(Internet Engineering Task Force)という技術者組織の技術者が、新しい技術を提案(提案文書: RFC文書)
    - 提案に対して様々ン意見が出され、改良や修正
    - 最終的に、実証実験や正式な会議により、標準化が決定

議論の過程や標準化された規格は広く公開され、誰でも利用可能

➡ 自由で開放的な開発スタイルがインターネットの発展に寄与

but...自由で開放的なために、様々な問題も

- 知的財産の侵害
- コンピュータウィルス
- 不正アクセス