コンピュータ・サイエンス2

第7回
プログラミングとアプリケーション

人間科学科コミュニケーション専攻
白銀 純子

第7回の内容

- プログラミング
- プログラミング言語
- アプリケーションの種類
- 情報ネットワーク

プログラミングとプログラミング言語

プログラミング言語の変遷[1](p. 76)

- プログラム: コンピュータに命令を伝えるための文書
- プログラミング言語: プログラムを記述するための言葉

初期: 機械語でプログラムを記述

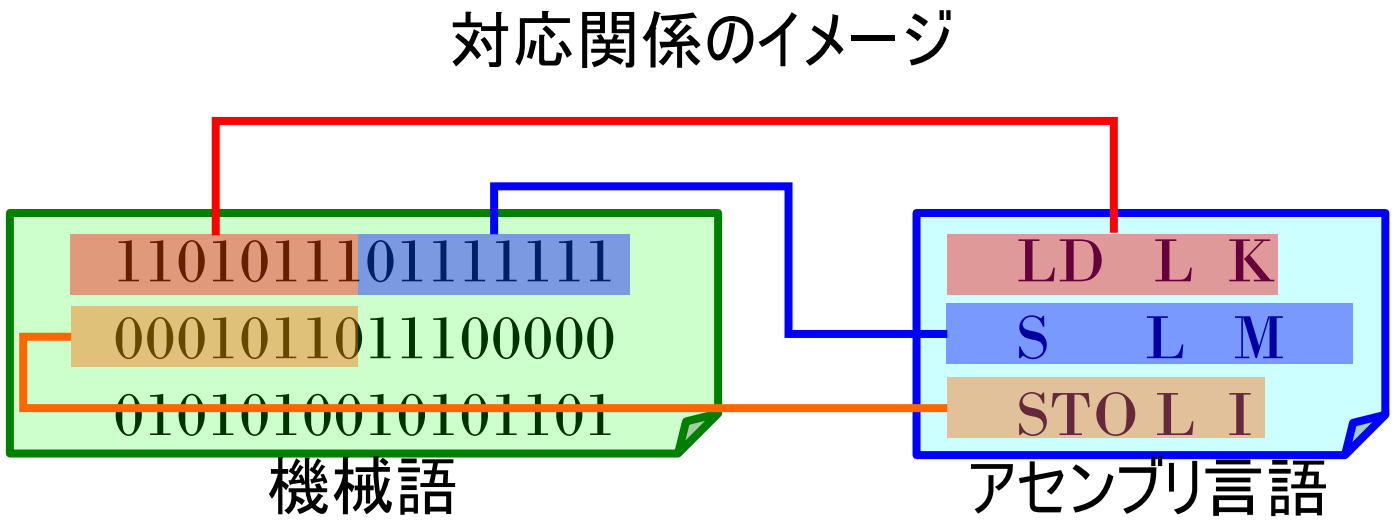
- 機械語: 0と1の2進数の形式の言葉

→ 不便!

→ アセンブリ言語が登場

アセンブリ言語

- 英語に似せた言語
- 機械語と1対1で対応
- アセンブリ言語のプログラムを機械語に翻訳
  - 翻訳ソフトウェア: アセンブラ



プログラミング言語の変遷[2](p. 76)

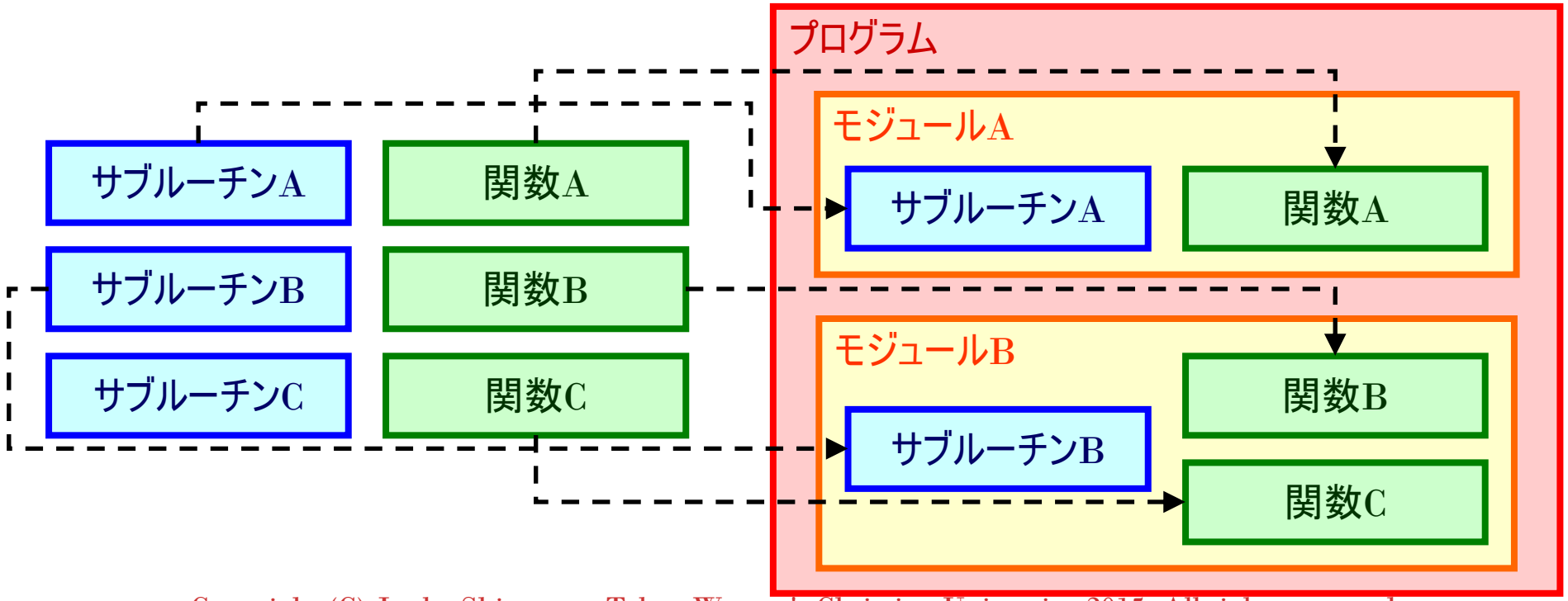
- 1954年にFORTRAN(FORmula TRANSlator)
  - IBM社が科学技術用言語として提唱
- 1959年にCOBOL(Common Business Oriented Language)
  - アメリカ国防省が商用言語として提唱
- 1962年にBASIC(Beginner's All purpose Symbolic Instruction Code)
  - ダートマス大学で初心者でも使える言語として提唱
  - ビル・ゲイツがよく利用し、Microsoft社が開発に注力

プログラミング言語の変遷[3](p. 76)

- 1972年にC言語
  - ベル研究所がOSなどの基幹ソフトウェアの開発用言語として開発
  - UNIXがOSとして初めてC言語で記述
- 1972年にSmalltalk
  - ゼロックス社のパロアルト研究所でオブジェクト指向言語として開発
- 1995年にJava
  - サン・マイクロシステムズ社(現オラクル社)でネットワーク対応言語として開発

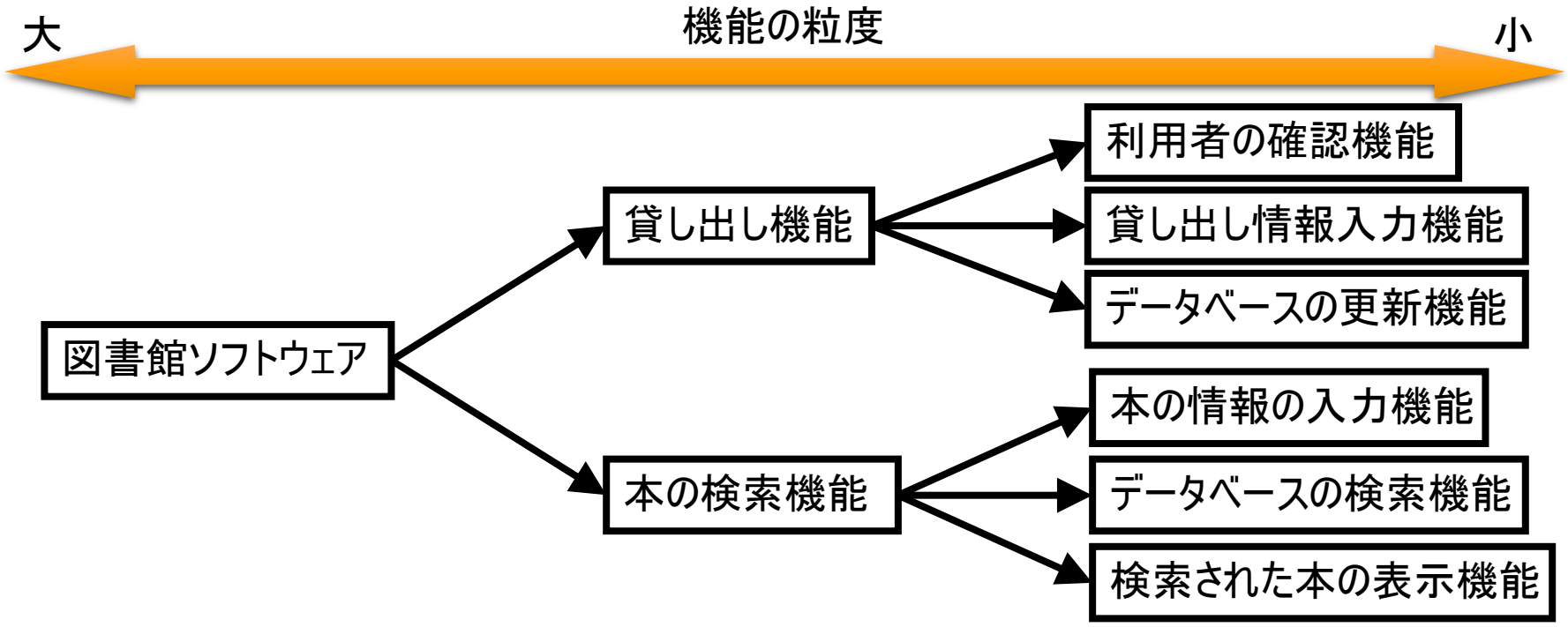
プログラミングの方法[1](p. 77)

- 手続き型プログラミング
  - プログラムを処理の単位ごとに分割(モジュール)
  - サブルーチンと関数でモジュールを構成
    - サブルーチン: あらかじめ用意された、よく使う処理のまとまり
    - 関数: 自分で定義する、よくつかう処理のまとまり



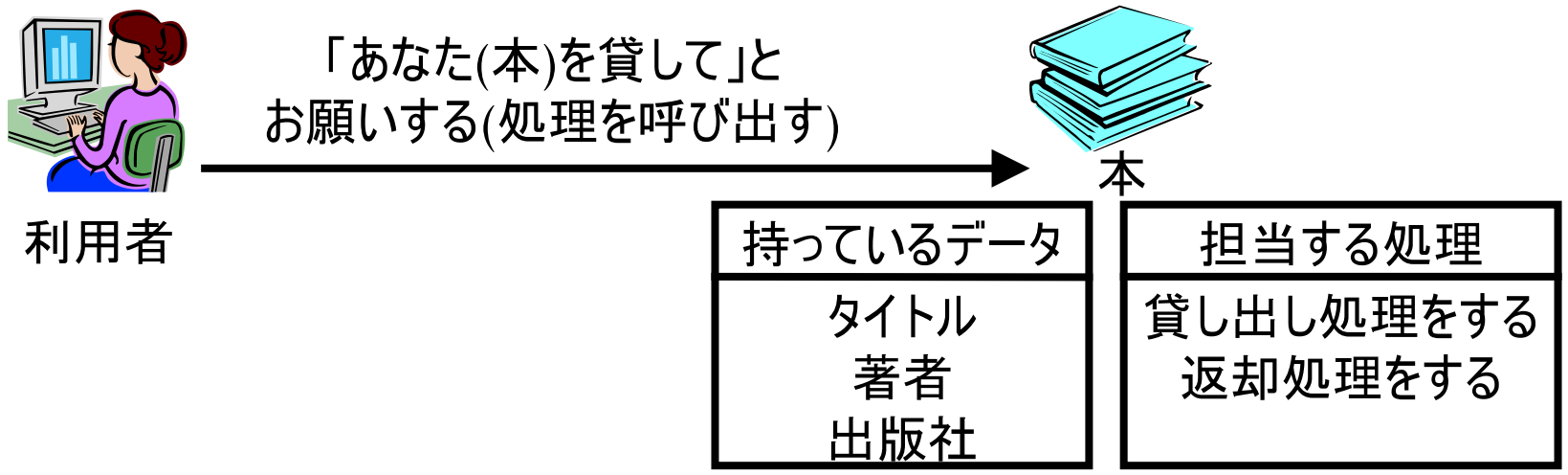
プログラミングの方法[2](p. 77)

- 構造化プログラミング
  - プログラムを小さな機能に分解し、それを組み合わせると完成するという考え方
  - 連続(順番に処理する)・判断(状況によって異なる処理をする)・反復(同じ処理を繰り返す)の組み合わせ



プログラミングの方法[3](p. 77)

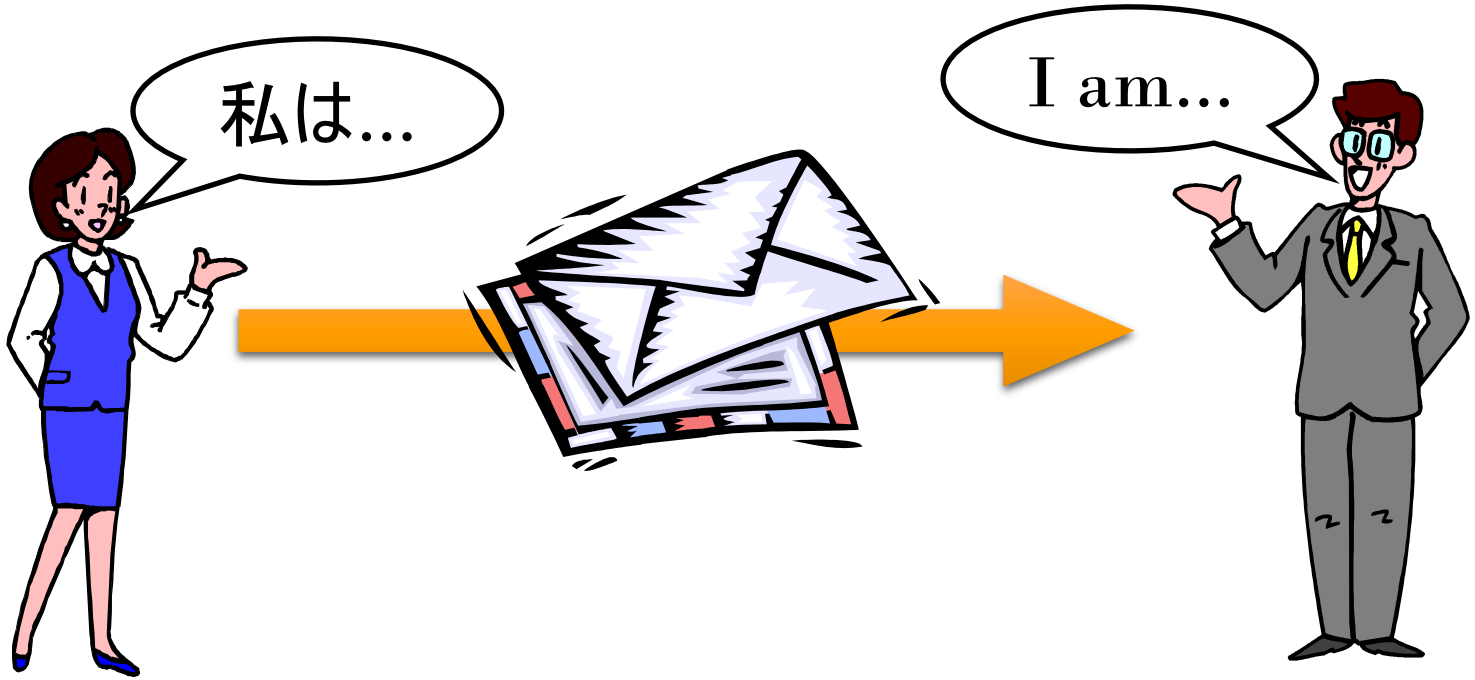
- オブジェクト指向プログラミング
  - 構造化プログラミングの欠点を克服することを目指して考案
  - 現実世界の「もの(オブジェクト)」に着目したプログラムの作成方法
    - 図書館システムであれば「本」や「利用者」など
  - 処理は、他のオブジェクトから依頼されて、オブジェクト自身が実行する、という考え方



プログラミング言語

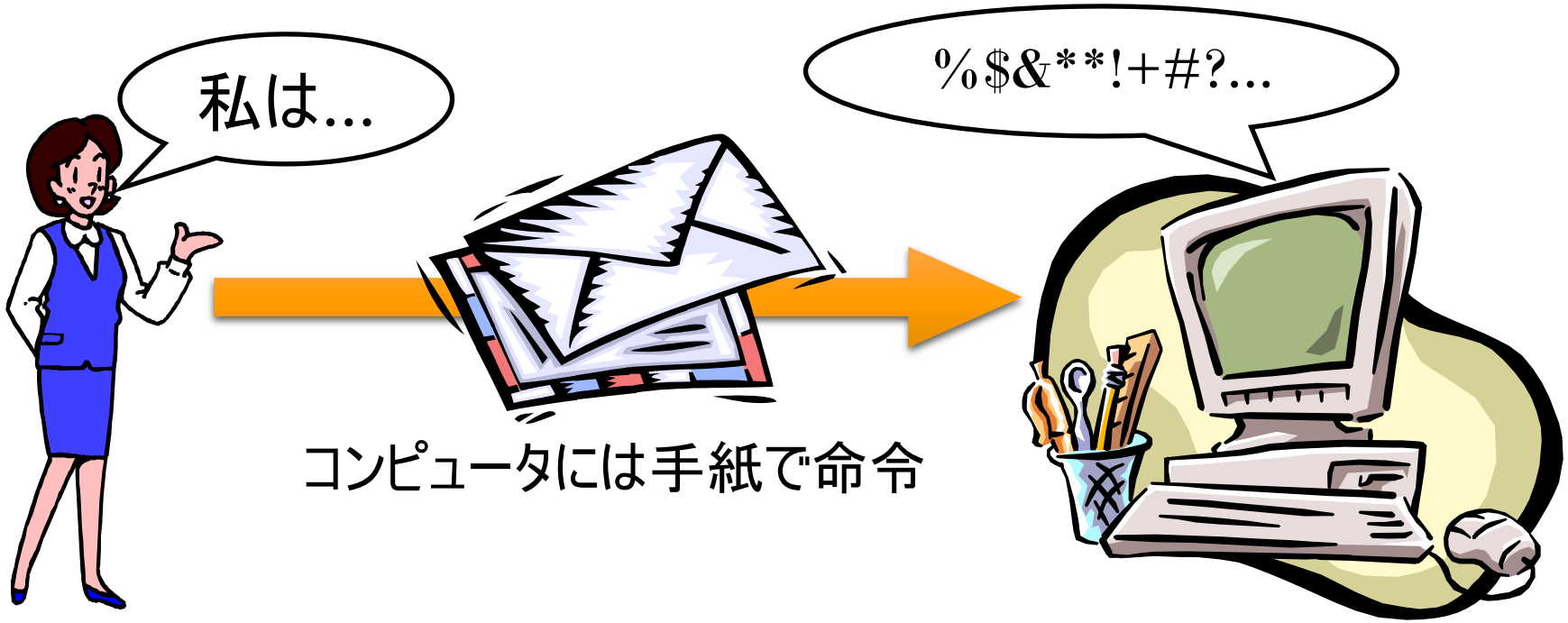
外国人に手紙を書く場合どうする??

- 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



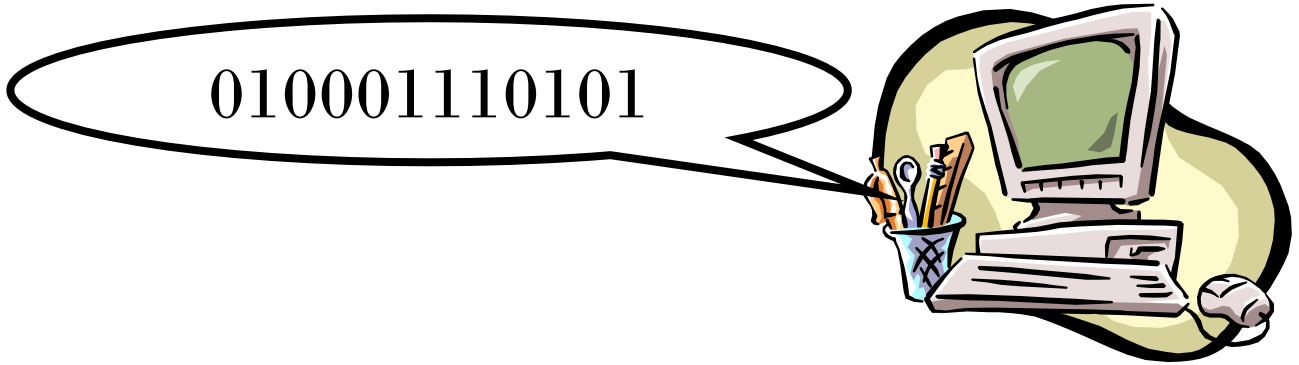
コンピュータが理解できる言葉は?

- 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

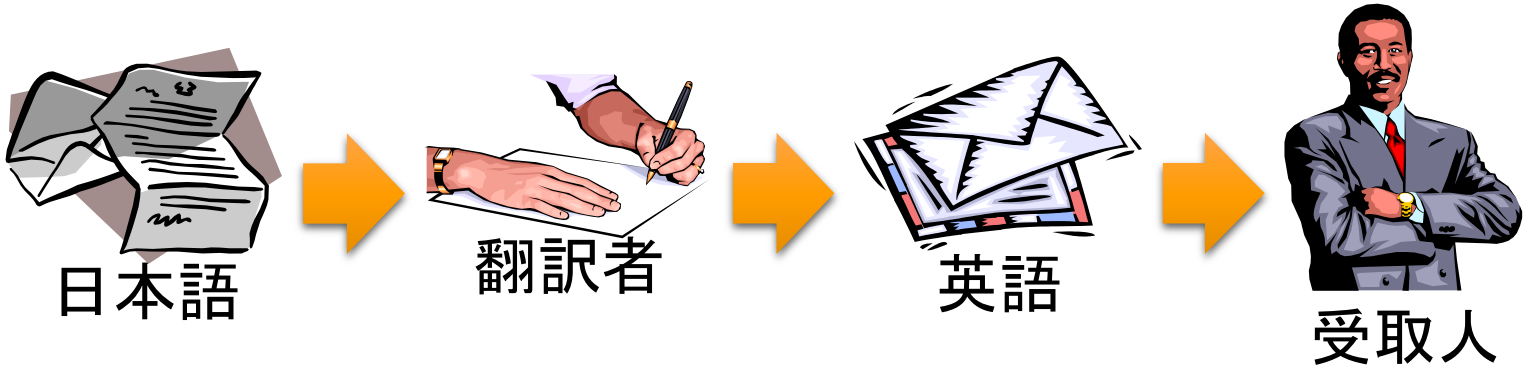
プログラミング言語

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

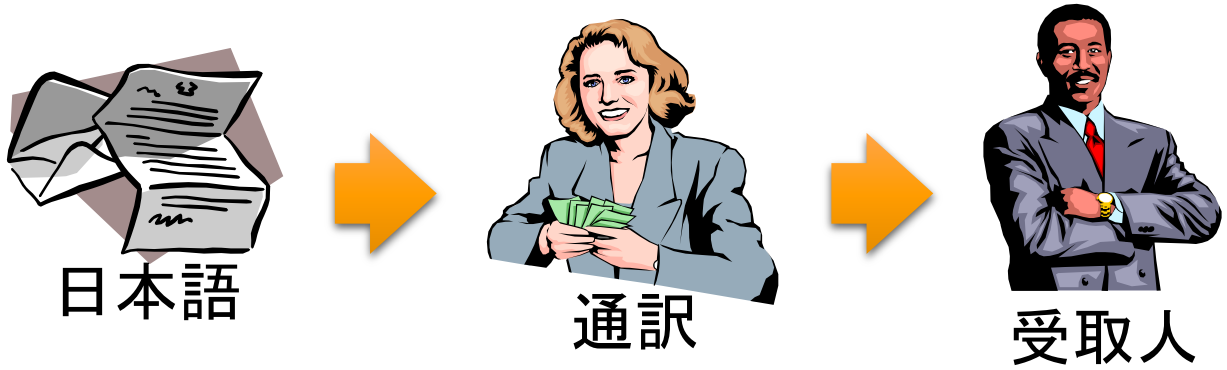


手紙を訳すには?

- 手紙を翻訳する



- 手紙を通訳する



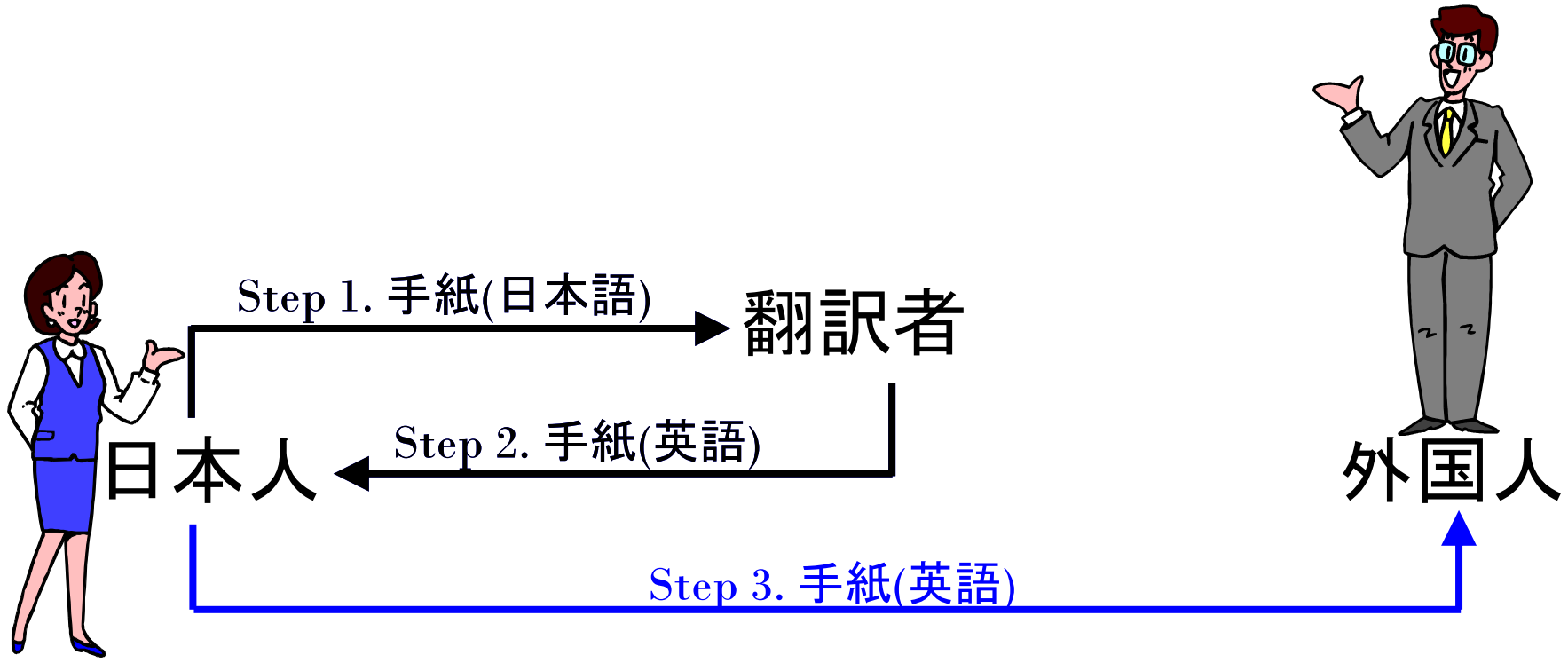
プログラムも、同じように機械語に訳す

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

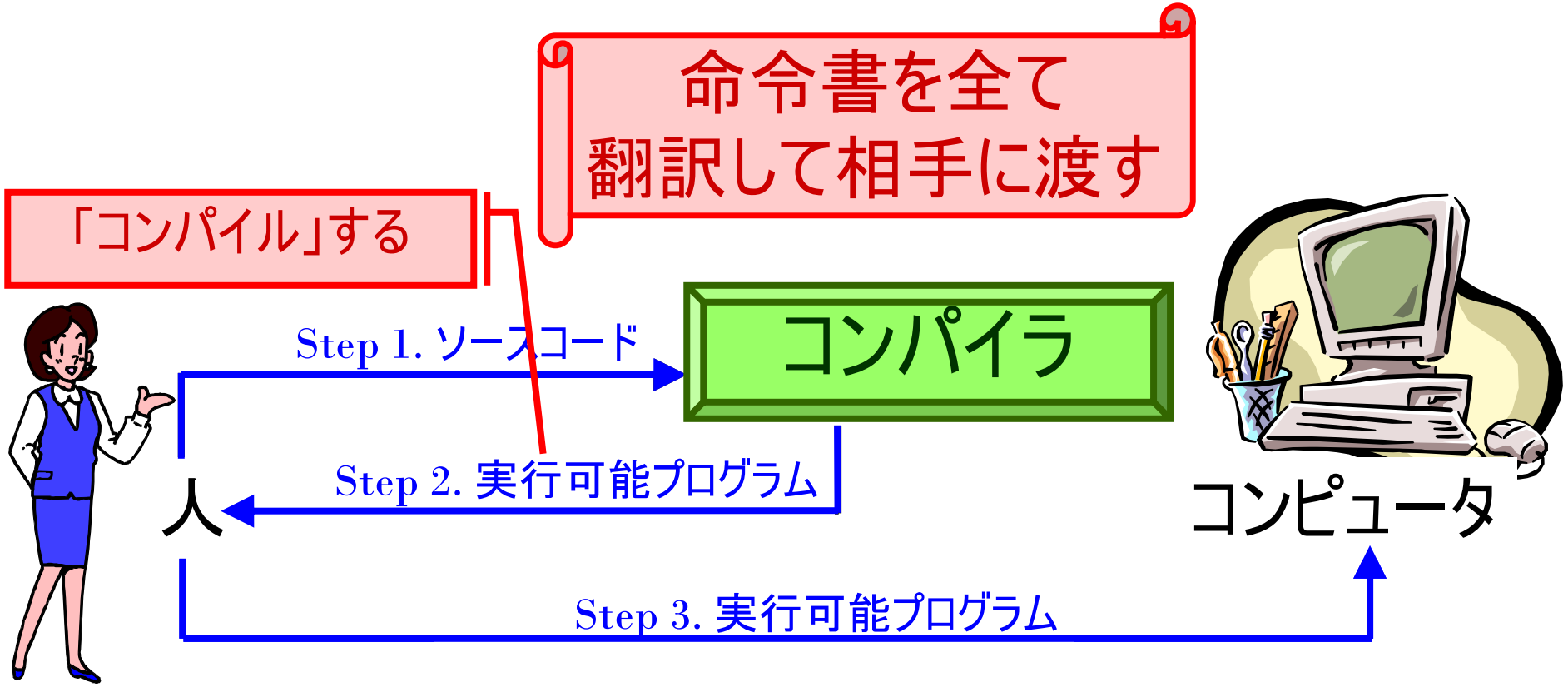
コンパイラ[概要](p. 78)

- 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

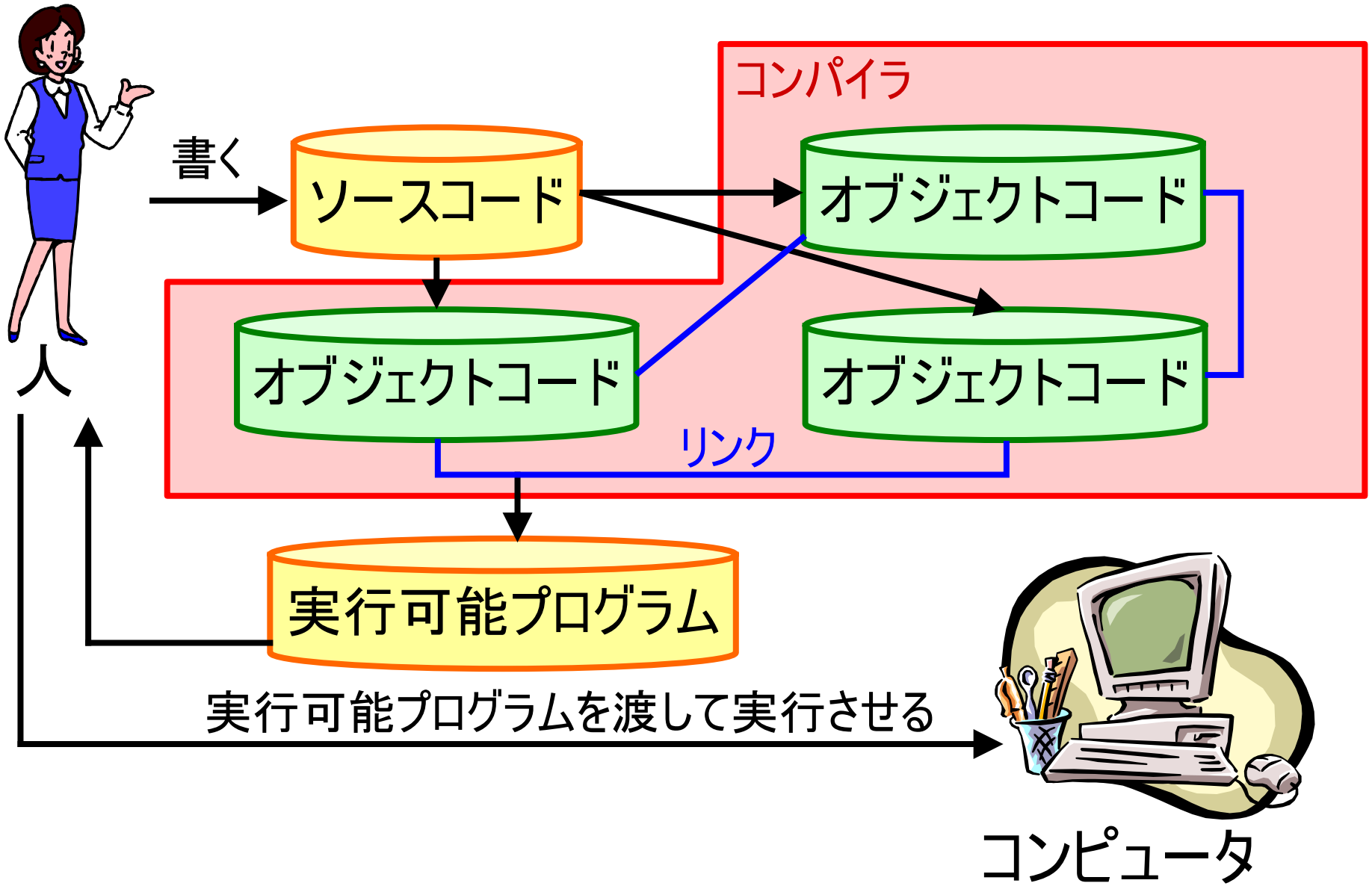


コンパイラ[概要](p. 78)

- 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)

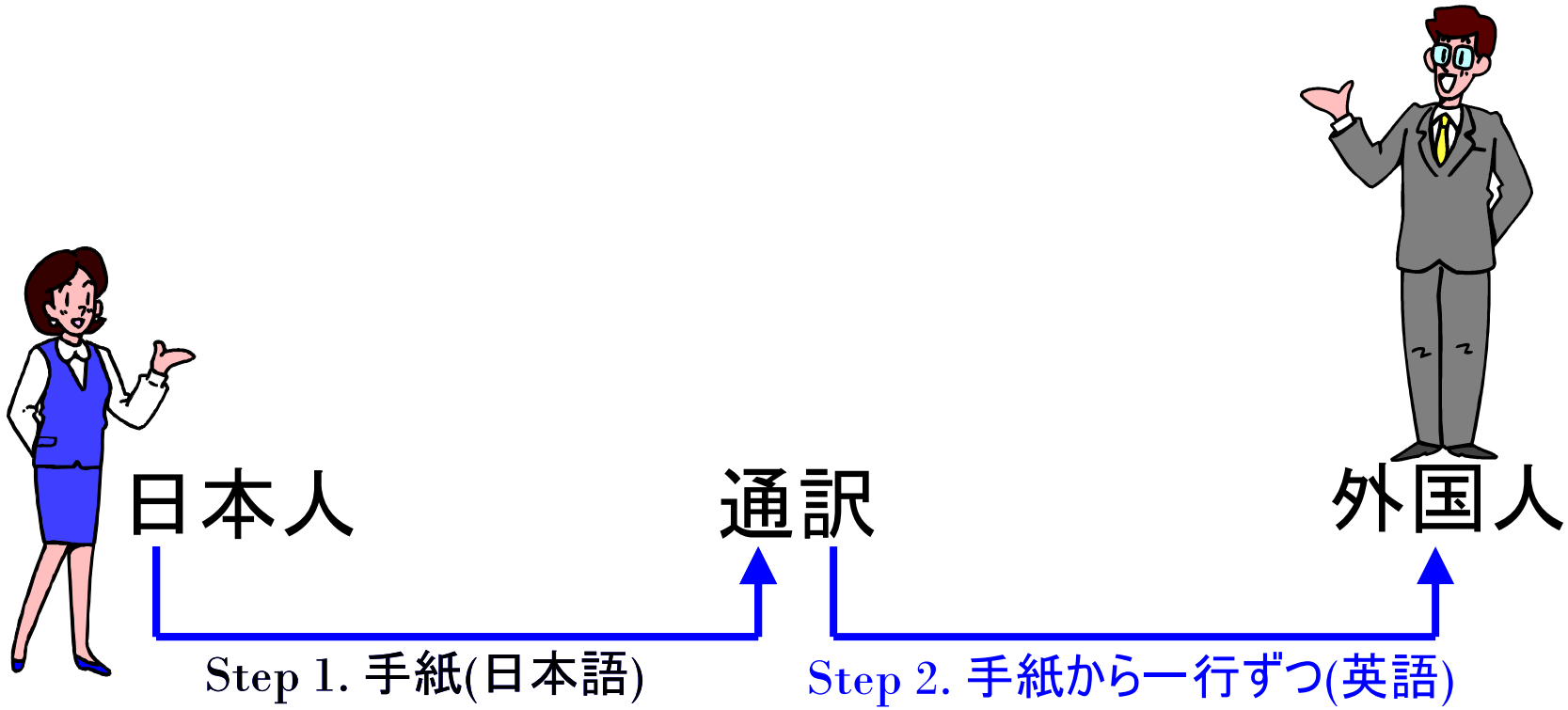


用語[1](p. 78)

- ソースコード: プログラミング言語で記述した命令書
- オブジェクトコード: ソースコードを翻訳したもの
- 実行可能プログラム: オブジェクトコードを連携させて、動作可能な形にしたもの
- プログラム: ソースコードと実行可能プログラムの双方の意味
  - どちらの意味で使われるかは、その時々で異なる

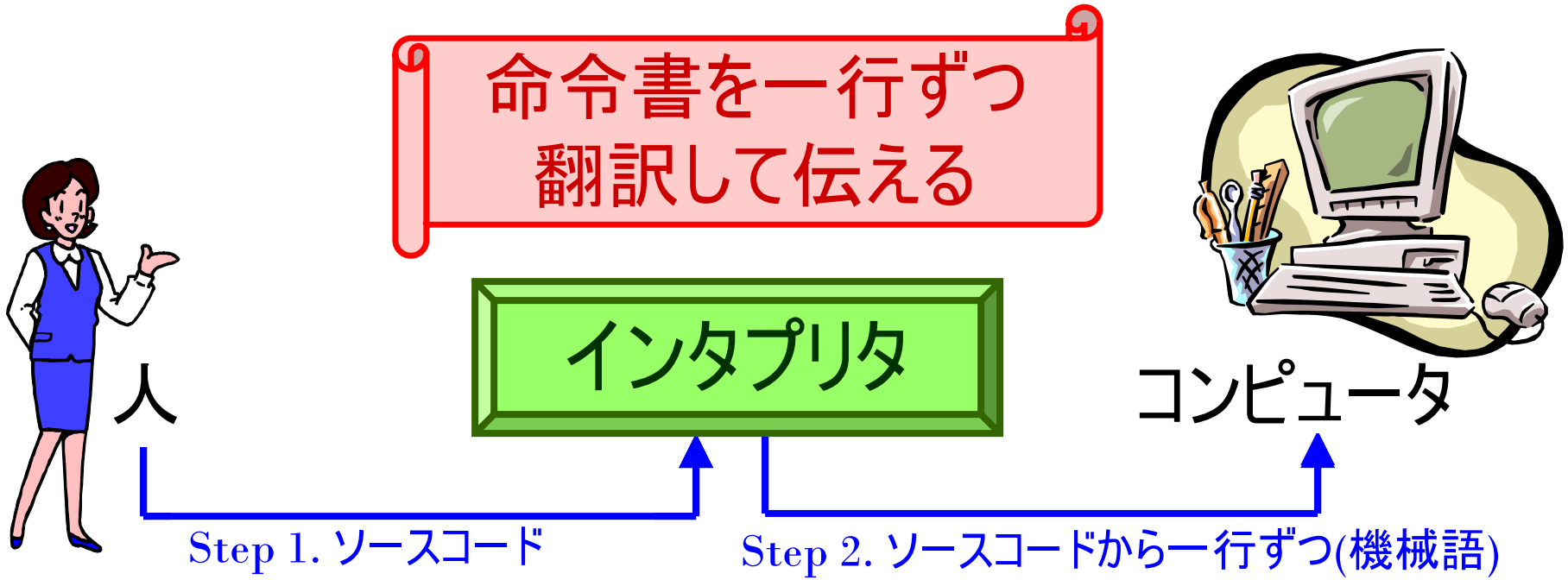
インタプリタ

- 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



アプリケーション

アプリケーションの種類(p. 79)

- アプリケーション: 人間が直接操作して、様々な処理をするためのソフトウェア
  - OSはソフトウェアではあるが、アプリケーションではない
- 種類
  - 専用アプリケーション
  - オフィススイート
  - 画像・音楽・映像用アプリケーション
  - コミュニケーションツール
  - セキュリティ対策アプリケーション
  - eラーニング用アプリケーション
  - 組込みソフトウェア

専用アプリケーション(p. 79)

- 企業や組織、大学などの業務に特化したもの
  - 自分のところで独自に開発や、開発会社に開発を依頼
  - 業務の形態が変わると、開発のしなおし
    - 銀行の統合、大学の入試システムの変更、etc.
  - ものによっては、汎用の機能を持つものをカスタマイズ
    - Ex. 大学の履修登録システムや図書館システムなど
- 例
  - 銀行のオンラインシステム
  - POS(Point Of Sales)システム
    - コンビニやスーパーなどの販売管理システム
  - 大学の業務システム
    - 東女のCampus Squareなど

オフィススイート[1](p. 80)

- 事務作業でよく使われるアプリケーションをまとめたもの
  - ワープロソフト
    - 文書を作成するためのアプリケーション
    - 以前はワープロ専用機を利用、現在はコンピュータ上のアプリケーションを利用
  - 表計算ソフト
    - データの集計や計算をするためのアプリケーション
    - グラフ作成なども可能
  - データベースソフト
    - 業務に利用するデータをため込んで管理し、容易に集計・検索・抽出するためのアプリケーション
  - プレゼンテーションソフト
    - アイデアや調査内容などの発表をするためのアプリケーション

オフィススイート[2](p. 80)

- 例
  - Microsoft Office
  - OpenOffice.org
  - etc.

画像・音楽・映像用[1](p. 81)

- グラフィックスソフト
  - 写真やイラストなどの画像の作成・加工のためのアプリケーション
  - ペイント系ソフト
    - ビットマップ画像を処理するためのアプリケーション
    - ビットマップ画像: 1つ1つの点が何色かで内容を表現する画像
      - 画像の内容についての情報(○とか□とか)を持たないので、拡大・縮小すると、画質が劣化するが、写真などの規則性を持たない画像に向く
  - ドロー系ソフト
    - ベクトル画像を処理するためのアプリケーション
    - ベクトル画像: 内容を方程式の形で表現する画像
      - 画像の内容についての情報(○とか□とか)を持つので、拡大・縮小しても、画質は維持されるが、写真などの規則性を持たない画像には向かない

画像・音楽・映像用[2](p. 81)

- 3D画像の処理アプリケーション
  - 平面図・立面図・側面図を書く機能: モデリング
  - 2次元の画面に投影する機能: レンダリング
  - CPUやビデオカードに高い性能が必要
- キャプチャソフト
  - TVやビデオカメラなどの映像をコンピュータに取り込むアプリケーション
- 音声の処理アプリケーション
  - 音楽や声をマイクなどからコンピュータに取り込み
- CAD(Computer Aided Design)
  - 回路や建築などの設計図を作成するためのアプリケーション

コミュニケーションツール[1](p. 81)

- コンピュータを通じて他者とのコミュニケーションをするためのアプリケーション
  - メーラ(メールソフト)
    - 現在は、ブラウザ上で利用するものが多い
  - ブラウザ
    - Webページを管理しているコンピュータと通信し、必要なファイルを入手し、内容を解析して表示するアプリケーション
    - 1990年ごろにCERN(欧州原子核研究機構の前身)の研究者が研究情報のやりとりのためにHTMLを考案
      - HTML(Hyper Text Markup Language): Webページの内容を記述するための言語
    - この研究者が、W3C(World Wide Web Consortium)を設立
    - W3CがHTMLの文法など、WWWに関する国際規格を策定

コミュニケーションツール[2](p. 81)

- 現在は、様々なアプリケーションがブラウザ上で実現
  - BBS: 電子掲示板
  - ブログ(Weblog): WWW上での雑記帳
    - Webページの紹介や覚書、論評などを記録
  - RSS(Realty Simple Syndication): Webサイトの更新情報を簡単にまとめて配信するための文書フォーマット
    - RSSを読むためのアプリケーション: RSSリーダ
    - BBSやブログと連携

セキュリティ対策アプリケーション(p. 82)

- コンピュータをウィルスや不正アクセスの被害から守るためのアプリケーション
  - ウィルス: コンピュータを病気のような状態にするソフトウェア
  - 不正アクセス: コンピュータの利用権限を持たない人が勝手にコンピュータを利用すること
    - ソフトウェアの不具合や、利用権限を持つ人のパスワードの流出などにより行われる
  - 頻繁なアップデートが必要
    - ウィルスは毎日のように新種が出現
    - 不正アクセスは、様々な手口が考案

eラーニング用アプリケーション(p. 83)

- コンピュータでの学習を支援するためのアプリケーション
  - 大学での語学の授業
  - 企業での研修, etc.

- LMS(Learning Management System)
  - eラーニングのためのコンテンツの作成支援
  - 個人の学習履歴の記録や学習者間、教員と学生間のやりとり
  - 例: 東女のWebClass

組込みソフトウェア

- 家電製品などの特定のハードウェアに特化したアプリケーション
  - 小規模なものは、OSとアプリケーションが一体化
  - 大規模なものは、PCのOSなど、汎用のOS上で動作するものも
- 例
  - クーラー
  - 冷蔵庫
  - カーナビ, etc.

オープンソースソフトウェア

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.

情報ネットワーク

通信手段の発展の経緯[1](p. 88)

- のろし
  - 煙や火を使って遠く離れた人に情報を伝える手段
    - 伝えたい人が火を焚いて煙を上げ、見た人がその意図を了解
    - 複雑な内容の表現は困難
    - 見逃すこともあり、環境が悪い時には伝えるのは不可能
  - 古代中国や中世イギリス、弥生時代の日本などで利用
- 飛脚・郵便
  - 文字が書かれた紙などの「もの」を届ける手段
    - 紙だけでなく、お金や荷物なども運搬
    - 運ぶための人や時間が必要で、非効率的
  - 日本では、鎌倉時代から江戸時代まで飛脚として存在
  - 明治時代に、欧米式の郵便制度が導入

通信手段の発展の経緯[2](p. 89)

- 電信
  - 文字や数字などを符号化して電気信号にして伝送
    - 有線・無線のどちらもが利用
    - 日常の連絡手段だけでなく、緊急時にも利用
  - 19世紀半ばから利用
    - モールス通信から開始
    - グラハム・ベルによる電話機により、個人対個人の情報交換が可能

現代の情報ネットワーク[1](p. 89)

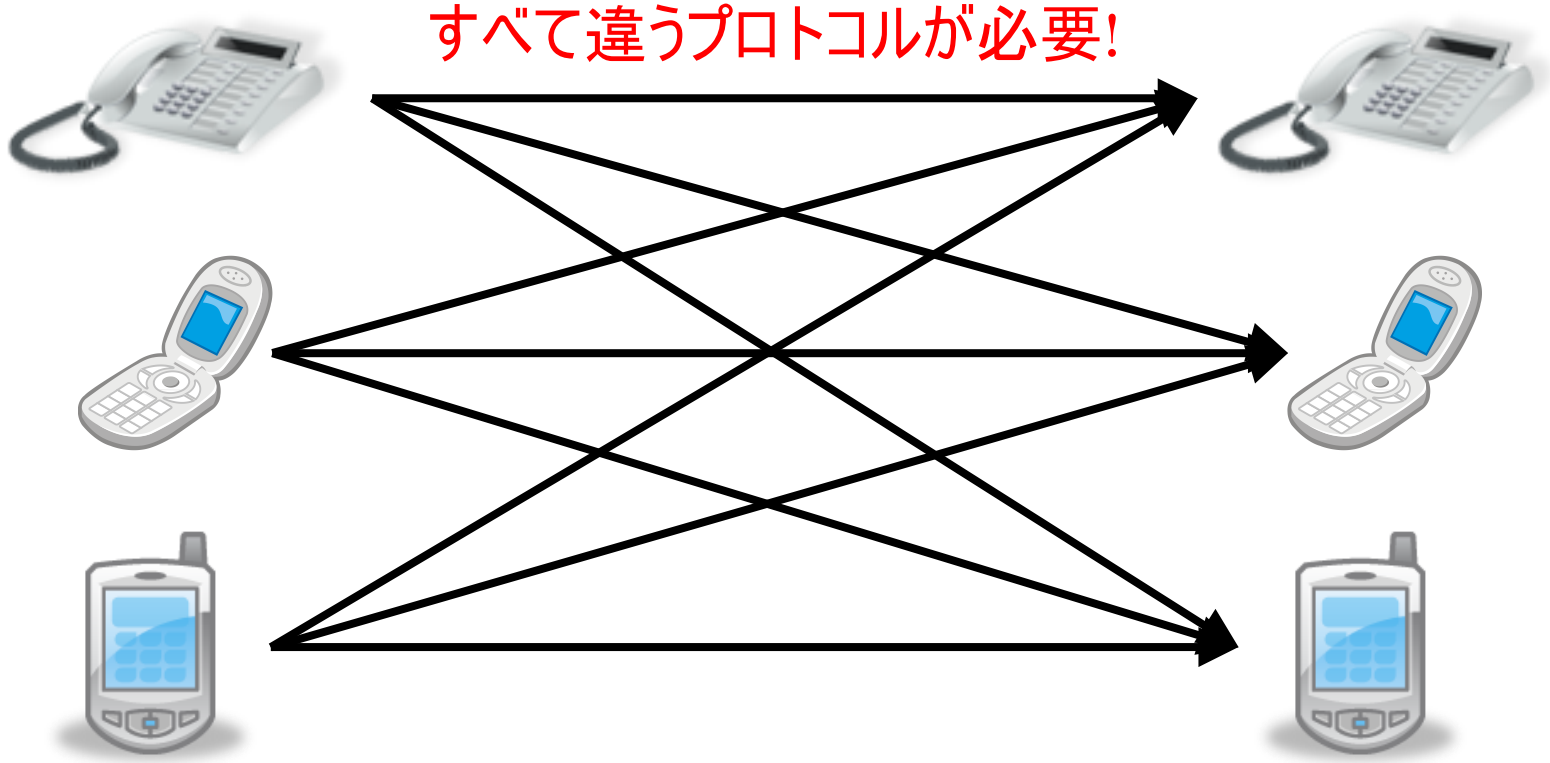
- 電気通信技術とデジタル化技術の組み合わせ
  - 文字・音声・画像などの情報をデジタル情報として扱い
  - 高速性・信頼性・経済性・利便性など、多くの利点
- 通信路(伝送路)によって、情報を伝達
  - 伝送媒体や通信機器などで構成
  - 多数の約束事(通信プロトコル)が必要
    - 通信路に正しく情報をのせ、届いた情報の意味を正確に理解するため
  - 多種多様な情報を、情報ごとに目的の場所に届けるための交換機能も必要
    - どこに届けるかを識別し、選択するための仕組み

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
  - 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.

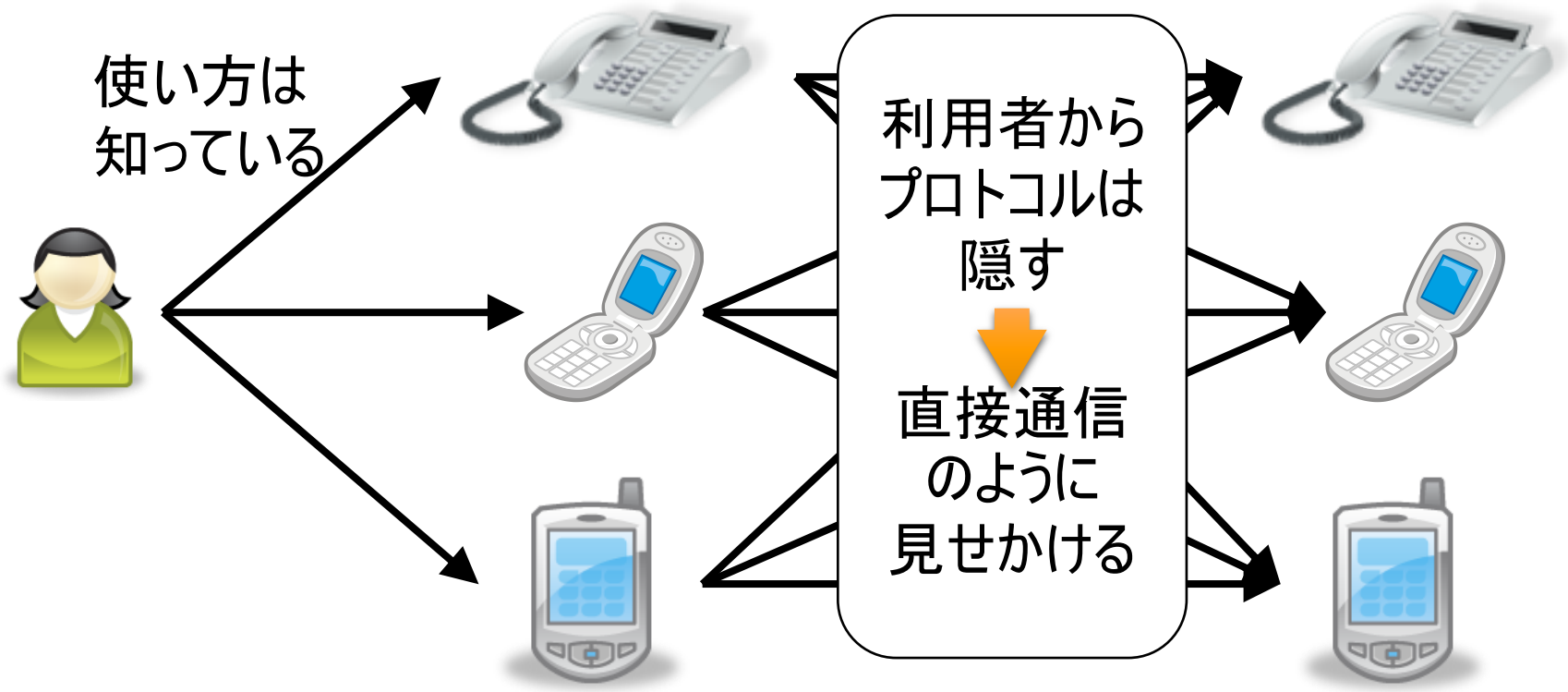
仮想化[1](p. 91)

- 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

OSI参照モデル

標準化とOSI参照モデル(p. 94)

- 以前は、企業や組織ごと、通信機器ごとに様々なプロトコルが利用
  - 異なる企業・組織のネットワーク同士を接続することは困難
  - プロトコルの異なる通信機器同士を接続することは困難

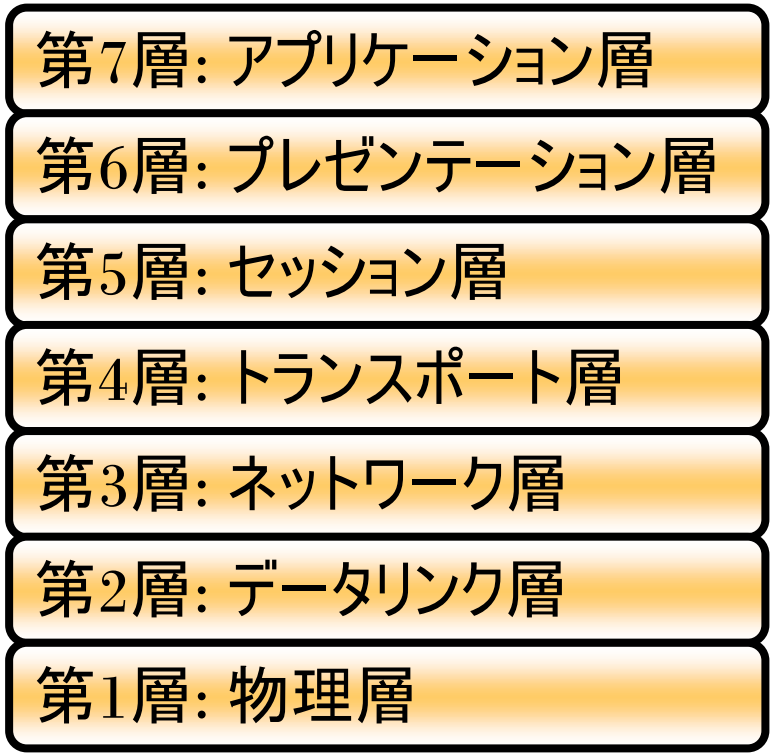
異なる相手同士でも、お互いにやりとりするには?

➡ ネットワーク通信の構造の標準化

OSI参照モデル
(Open Systems Interconnection Reference Model)

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



第1層: 物理層(p. 94)

- データを電気信号や光信号の形で送受信するときの方式
  - 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.
- 通信ケーブルなどの規格

第2層: データリンク層(p. 94)

- ネットワーク機器同士での通信方式
  - 物理層での通信に誤りがないかをチェック
  - 誤りがあれば、再送信を要求

- スイッチングハブなどの規格
  - スイッチングハブ: コンピュータ同士をネットワークに接続するための機器

第3層: ネットワーク層(p. 94)

- データの宛先を特定して送受信
  - データを送る経路の選択
  - データに宛先の情報を付加

- ルータなどの規格
  - ルータ: データを別のネットワークに向かって送り出したり、中継するための機器

第4層: トランスポート層(p. 94)

- データ送受信の信頼性を確保
  - ネットワーク層の通信内容に誤りがないかをチェック
  - 誤りがあれば、それに応じてエラーの出力や是正，再送の要求
- データをネットワークで送信できる大きさに分割
  - データは、大きさによっては、そのままでは送信できない

第5層: セッション層(p. 94)

- 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

第6層: プレゼンテーション層(p. 94)

- 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
    - 圧縮: データをある手順に従って、元のデータよりも小さくすること
  - 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

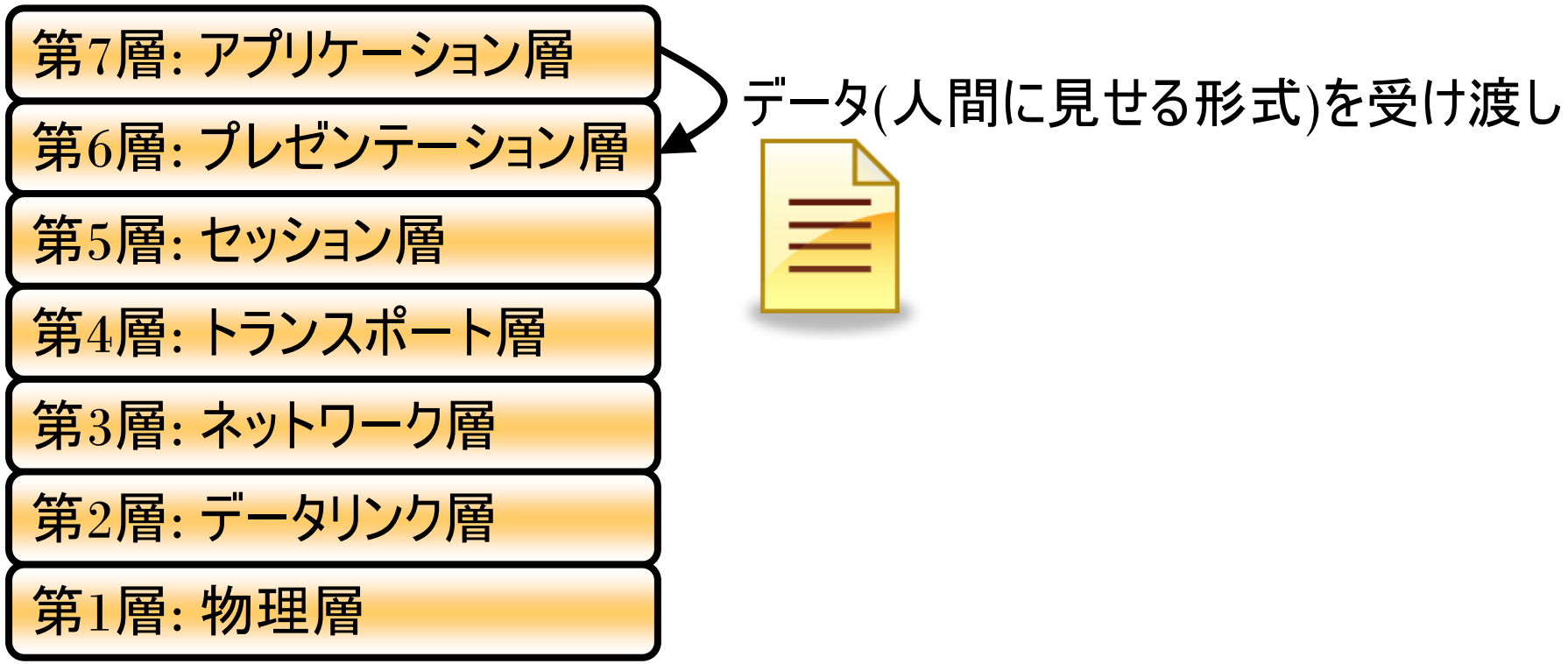
第7層: アプリケーション層(p. 94)

- 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)

データの送信(p. 94)

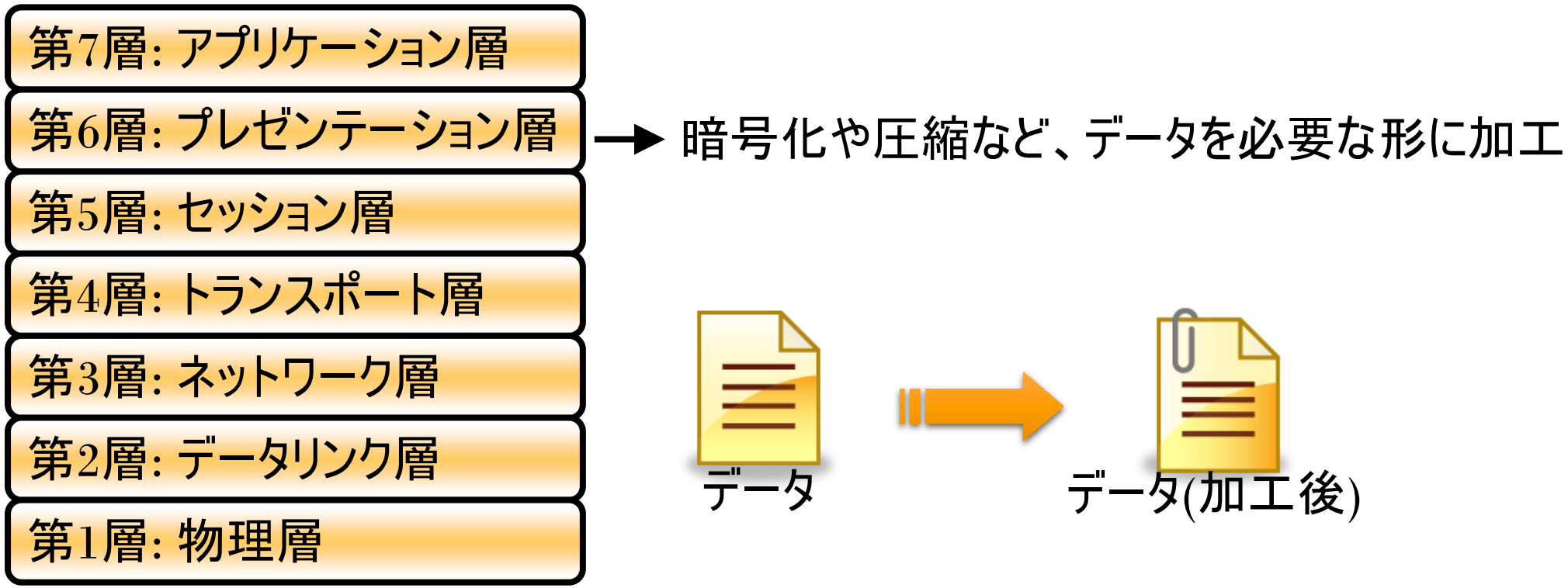
データ送信[1](p. 94)

- データ送信側



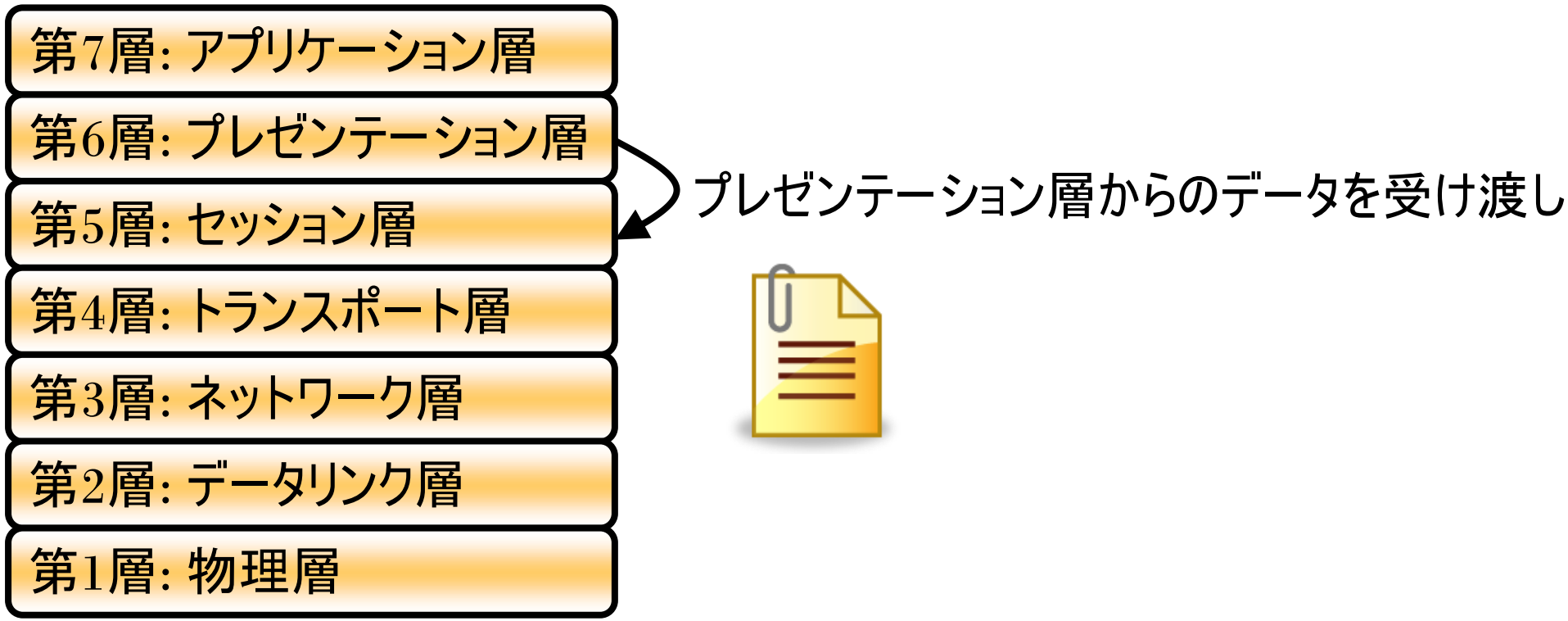
データ送信[2](p. 94)

- データ送信側



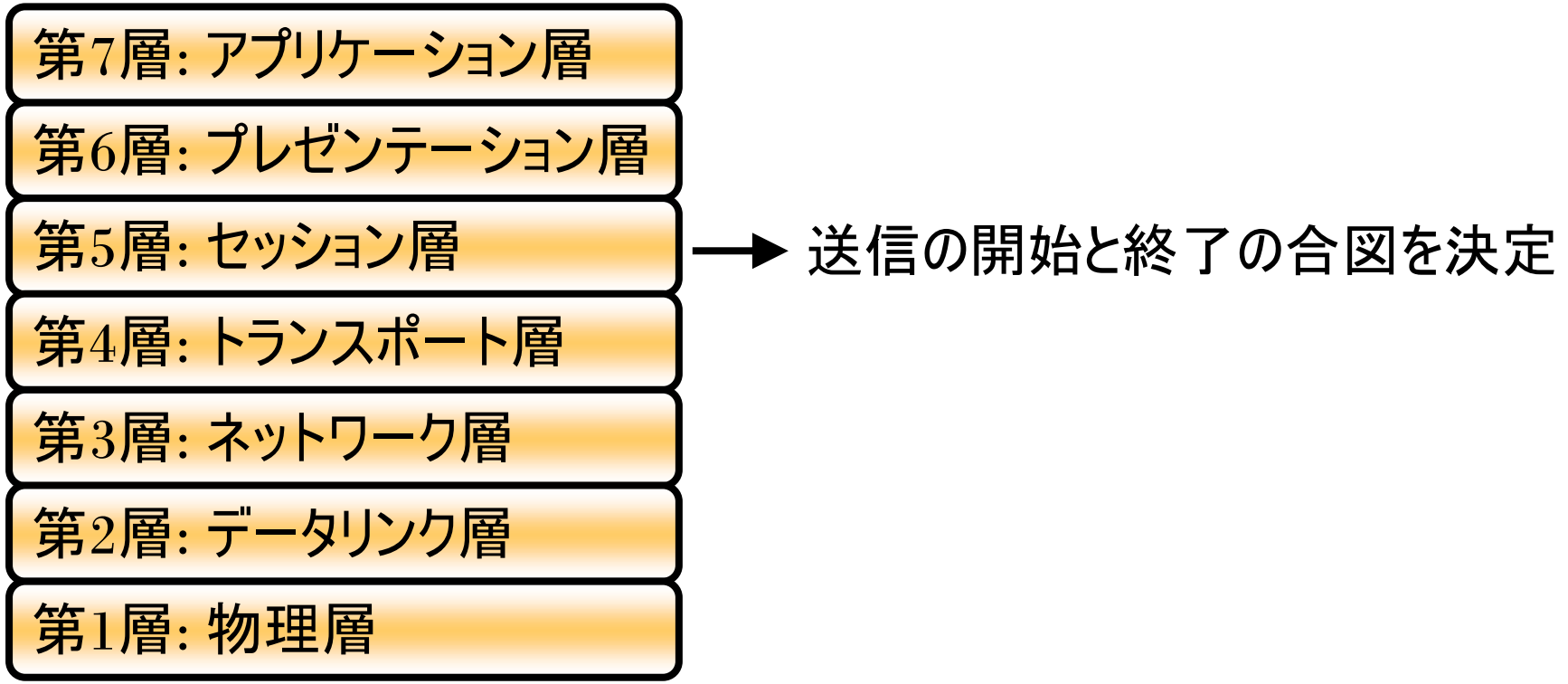
データ送信[3](p. 94)

- データ送信側



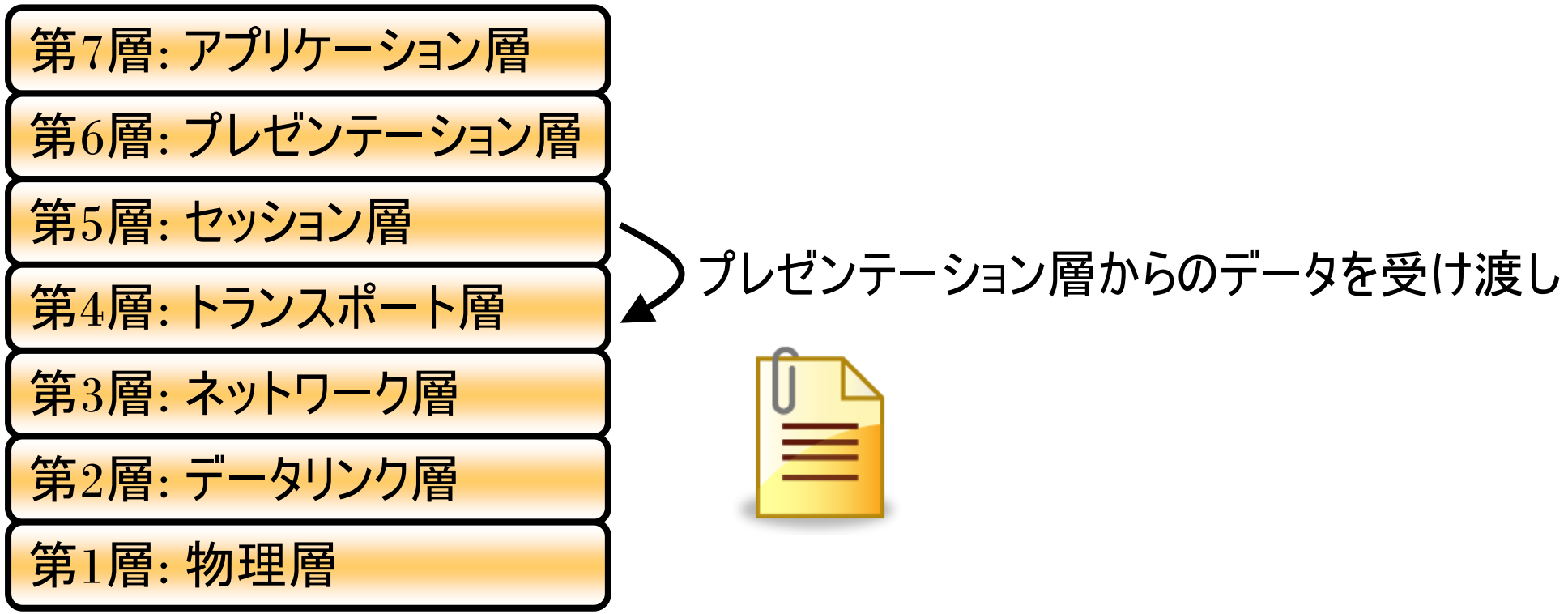
データ送信[4](p. 94)

- データ送信側



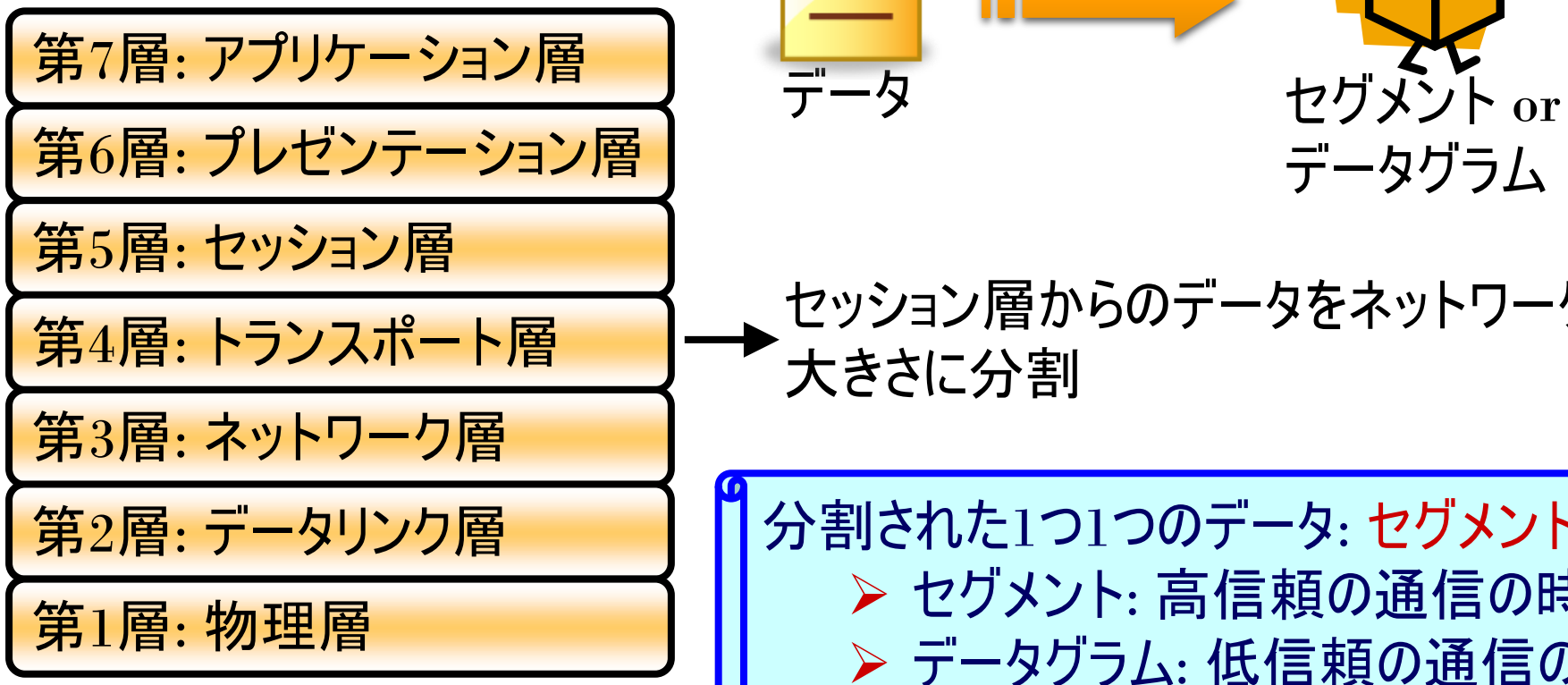
データ送信[5](p. 94)

- データ送信側



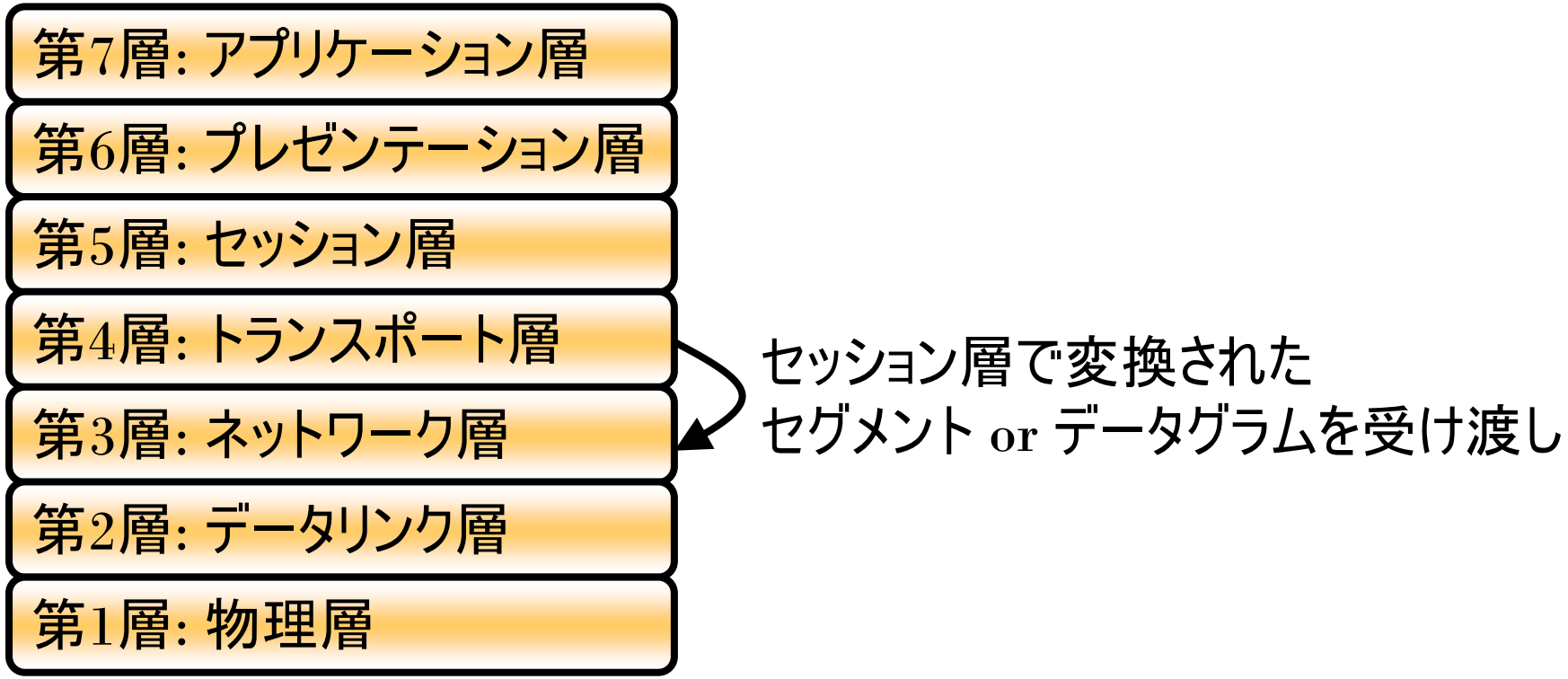
データ送信[6](p. 94)

- データ送信側



データ送信[7](p. 94)

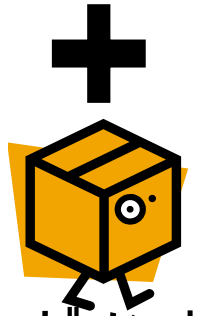
- データ送信側

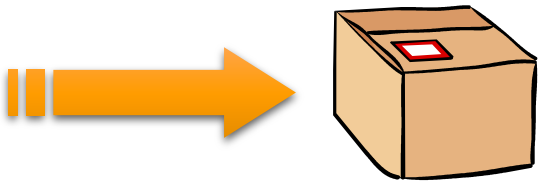


データ送信[8](p. 94)

- データ送信側

送信元・あて先





パケット

セグメント or
データグラム

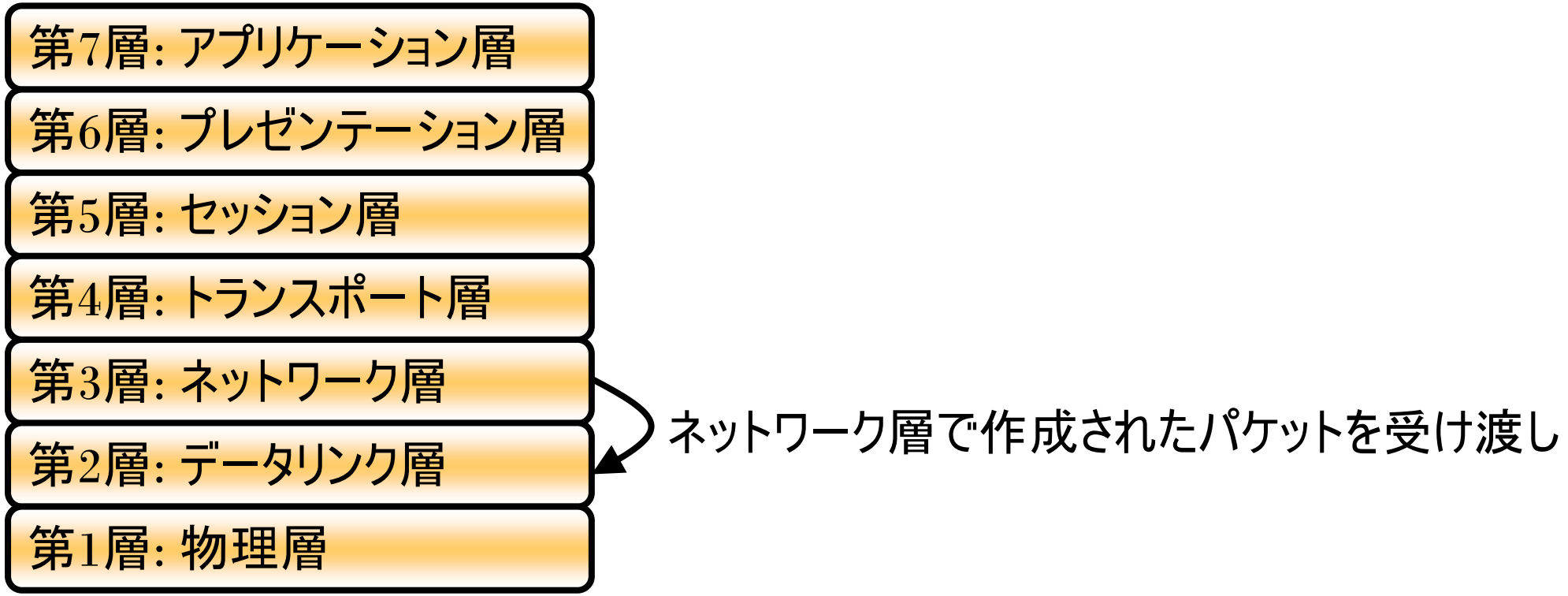
第7層: アプリケーション層
第6層: プレゼンテーション層
第5層: セッション層
第4層: トランスポート層
第3層: ネットワーク層
第2層: データリンク層
第1層: 物理層

→ 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

付加されたセグメント or データグラム:
パケット

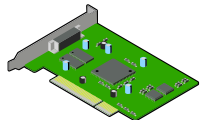
データ送信[9](p. 94)

- データ送信側



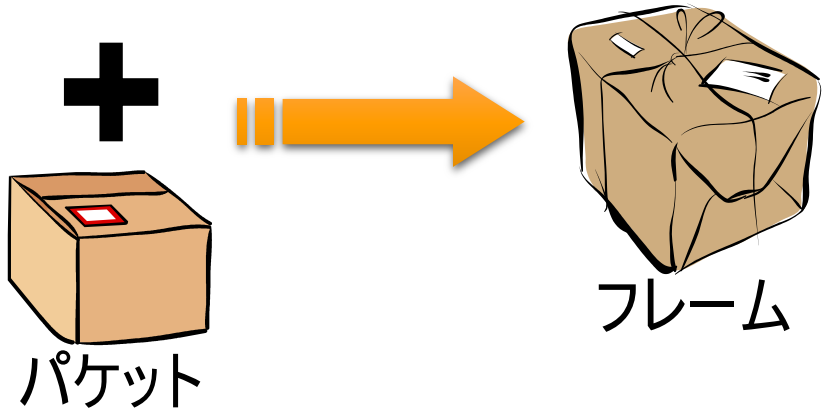
データ送信[10](p. 94)

- データ送信側



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

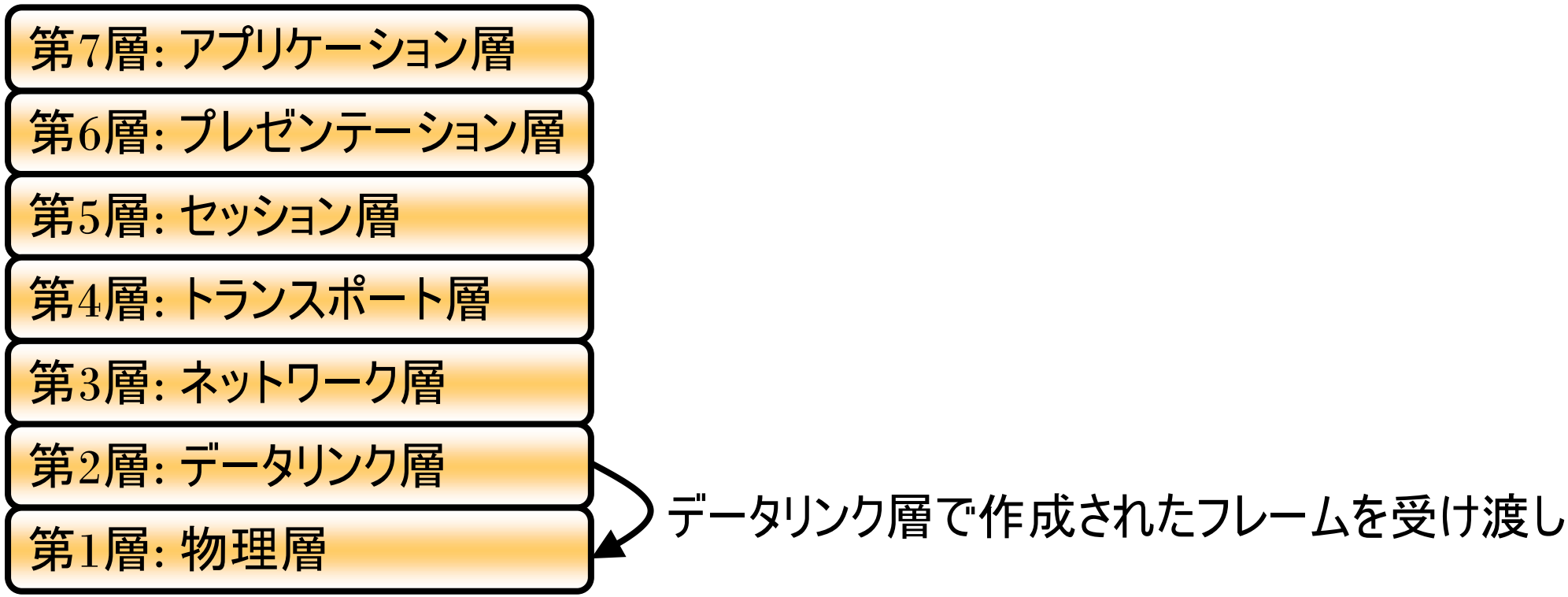


→パケットに送信先のMACアドレスの情報を付加

付加されたパケット: フレーム

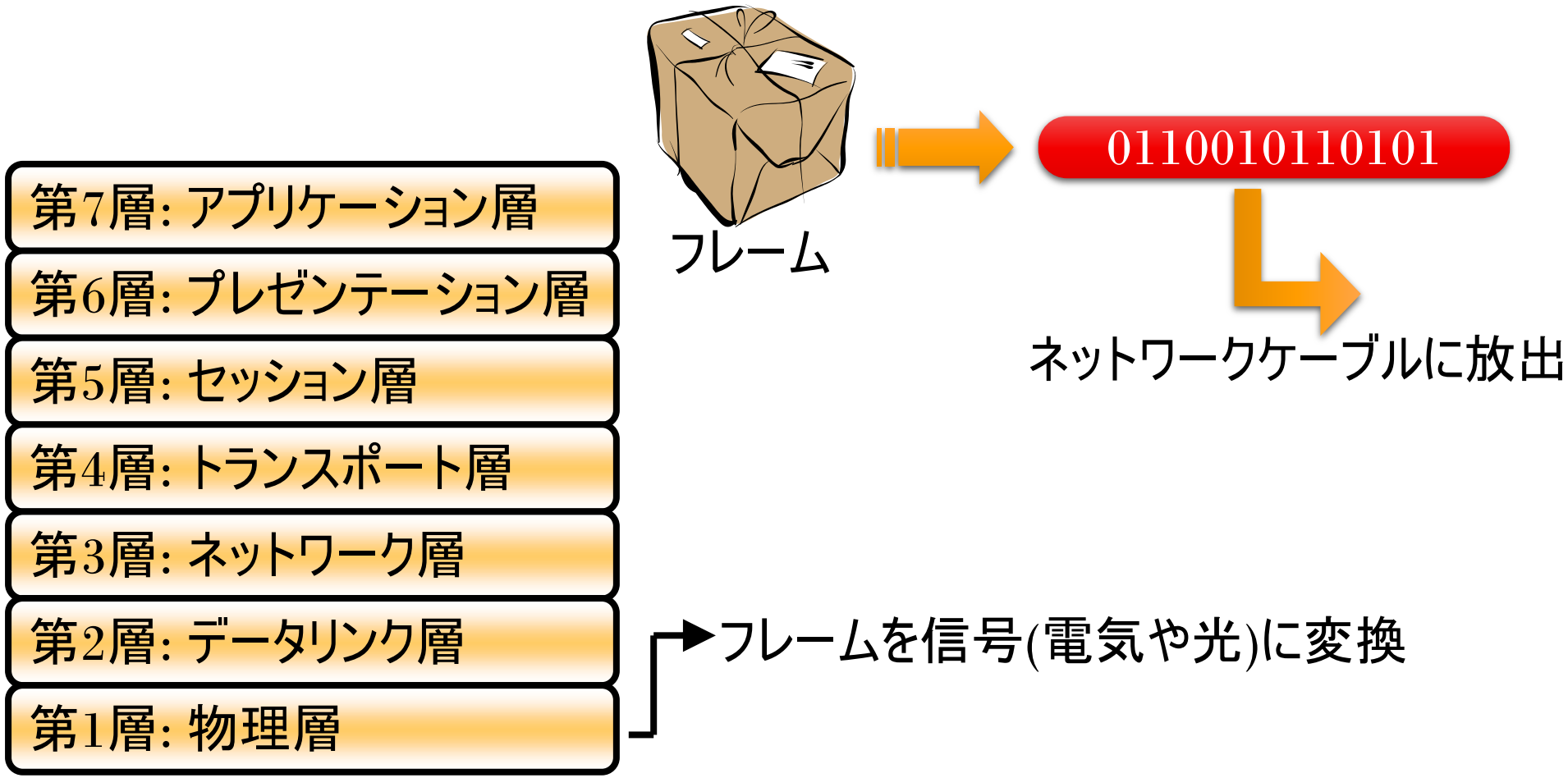
データ送信[11](p. 94)

- データ送信側



データ送信[12](p. 94)

- データ送信側



カプセル化(p. 93)

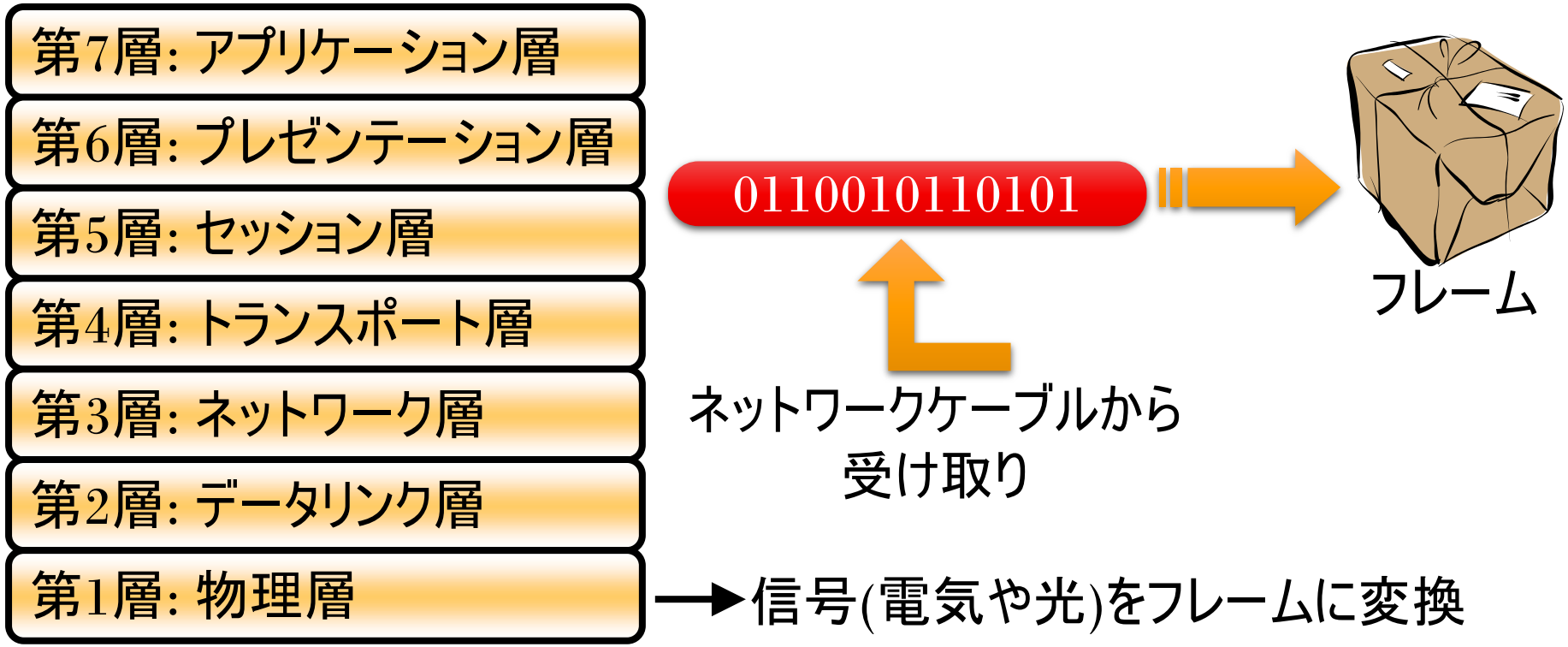
- 送受信するデータには、送受信のために必要な情報がつけられていない
  - 送信する宛先
  - 正しく届いたかどうかのチェック情報, etc.

必要な情報をデータに付加すること: カプセル化

データの受信

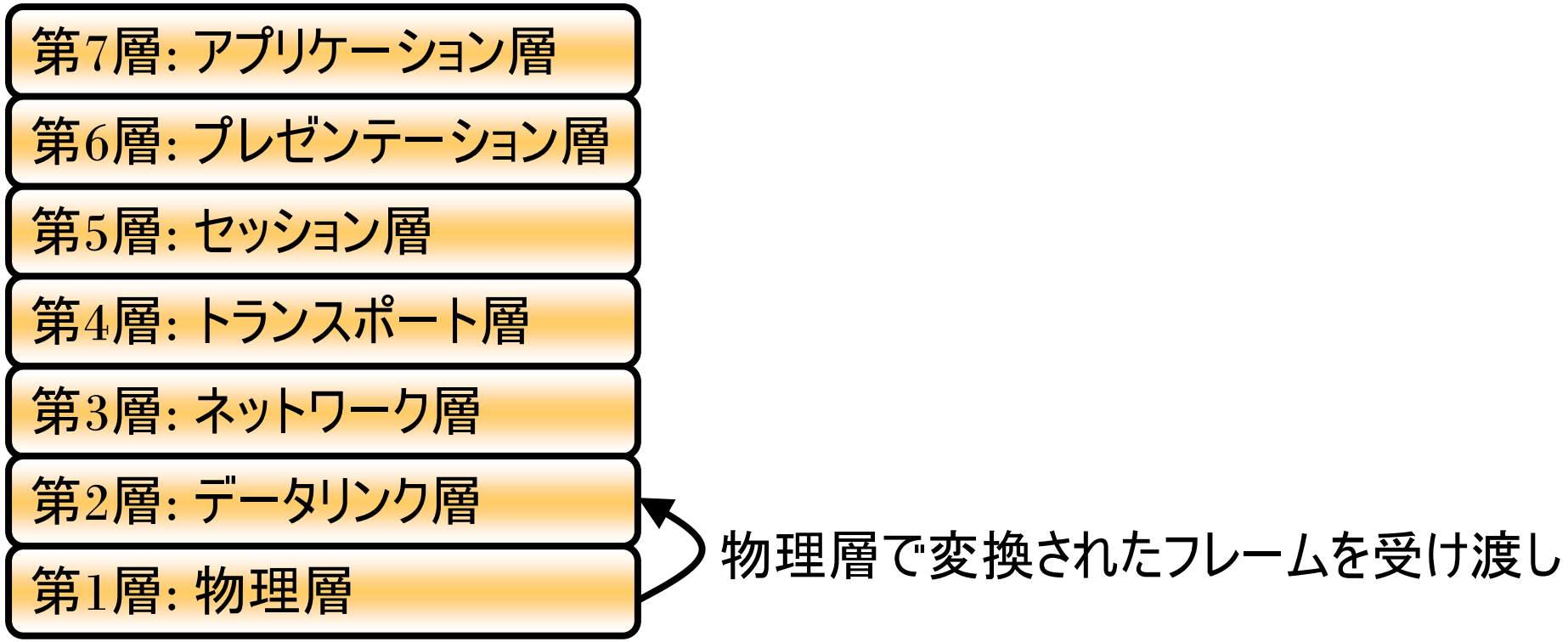
データ受信[1](p. 94)

- データ受信側



データ受信[2](p. 94)

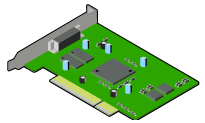
- データ受信側



データ受信[3](p. 94)

- データ受信側

フレーム



MACアドレス
(ネットワークカードに固有の番号)

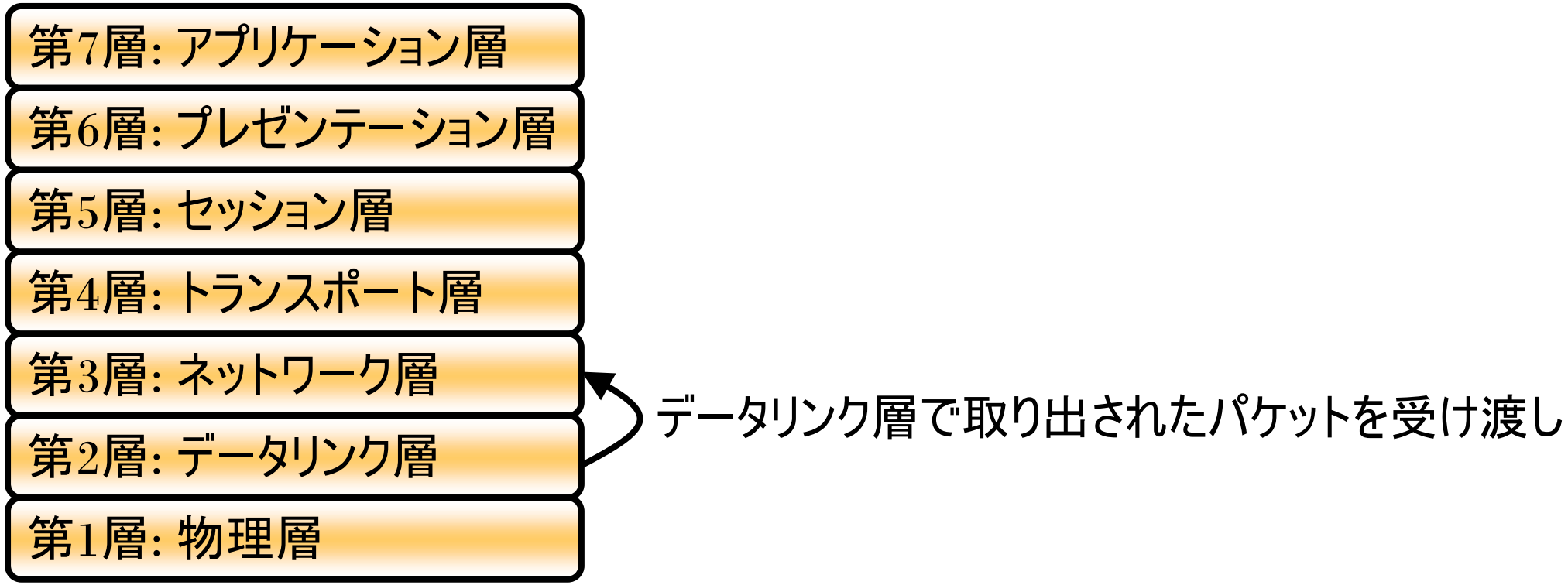
パケット

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)

- データ受信側



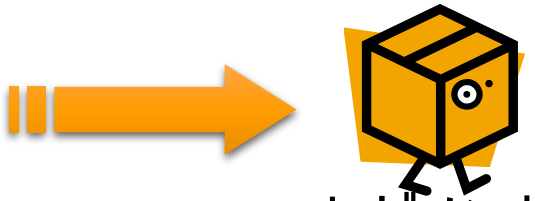
データ受信[5](p. 94)

- データ受信側

パケット

送信元・あて先

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

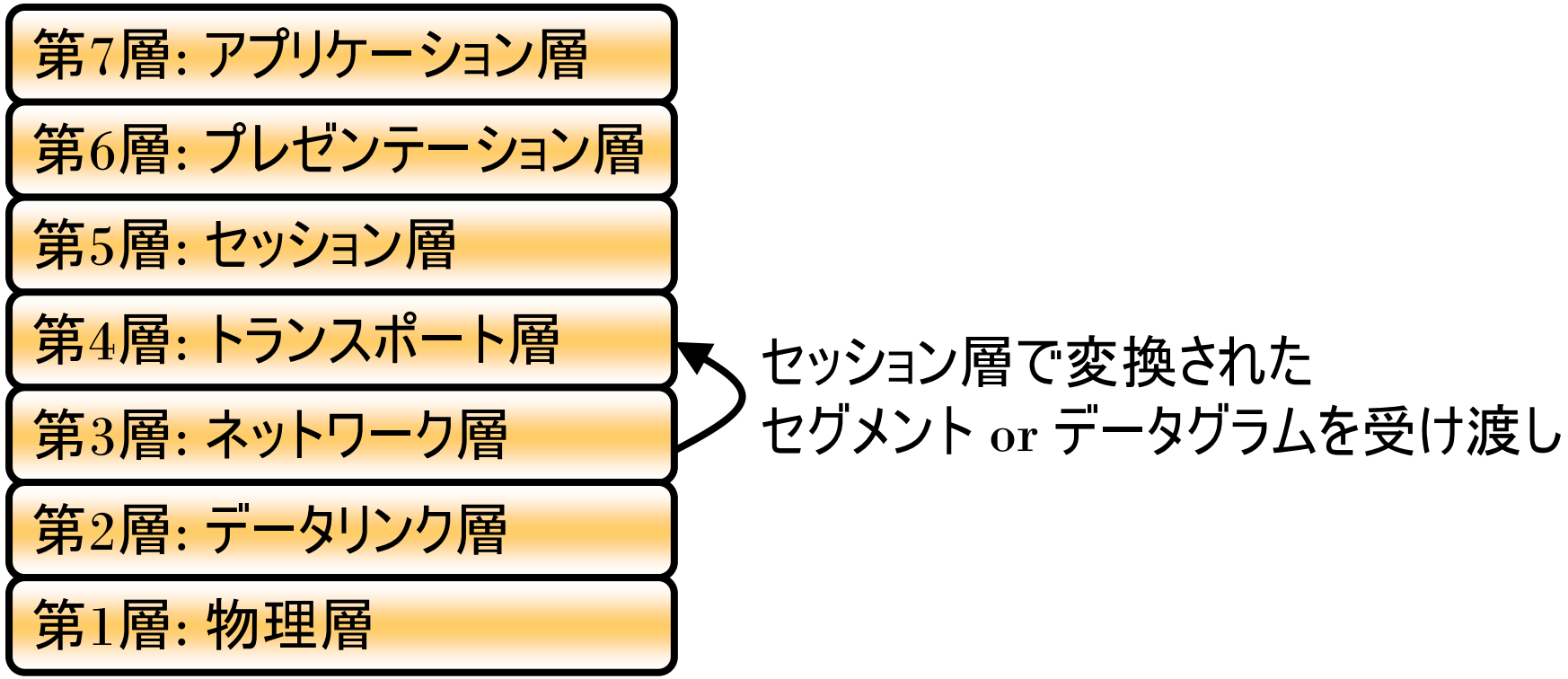


セグメント or
データグラム

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

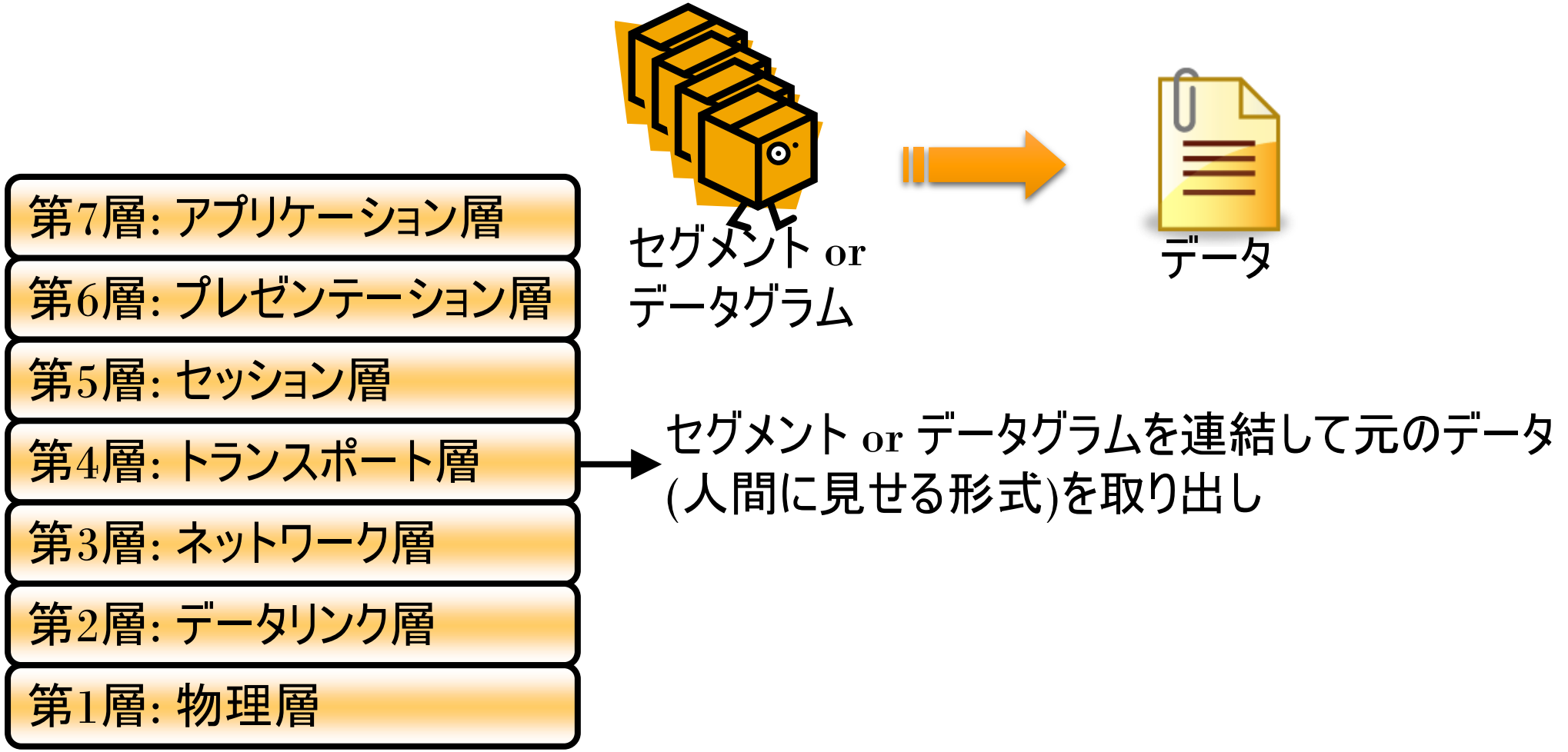
データ受信[6](p. 94)

- データ受信側



データ受信[7](p. 94)

- データ受信側

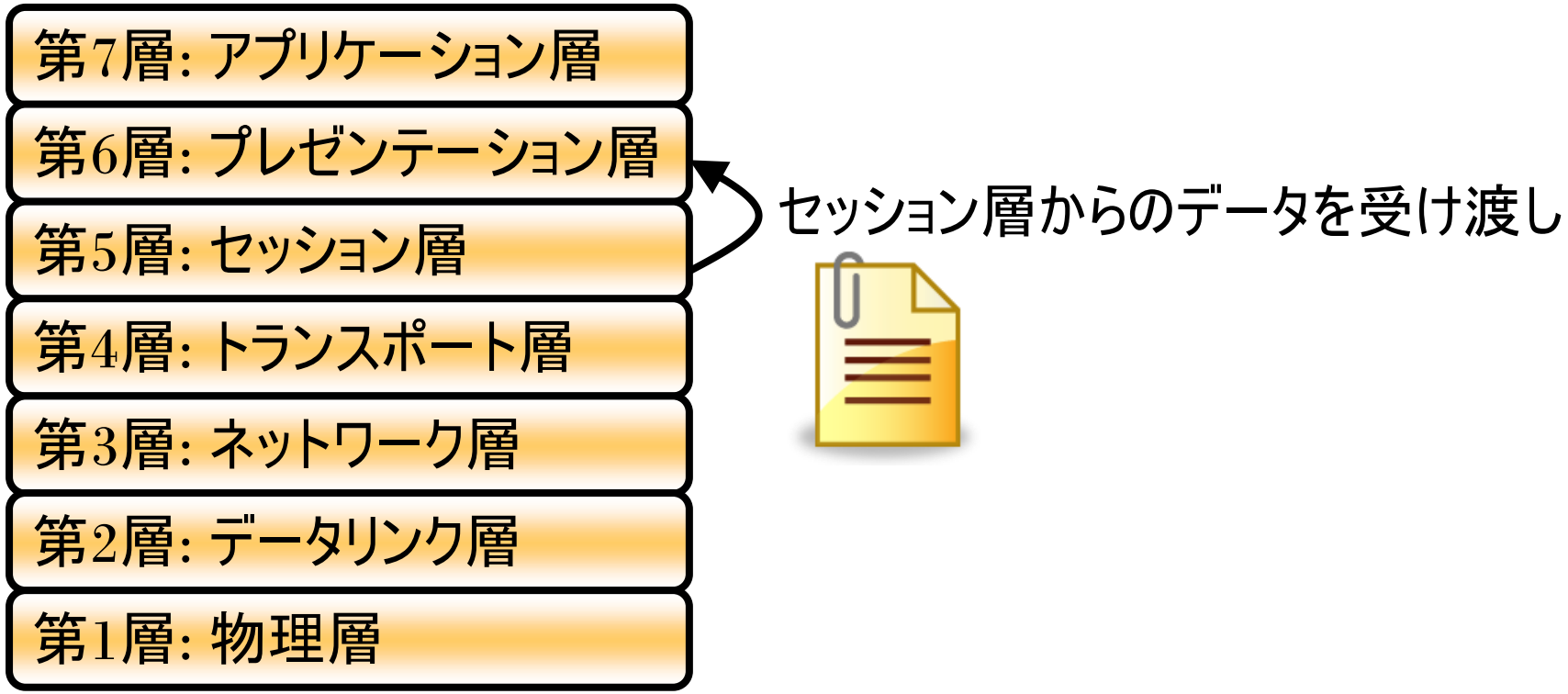


データ受信[8](p. 94)

- データ受信側

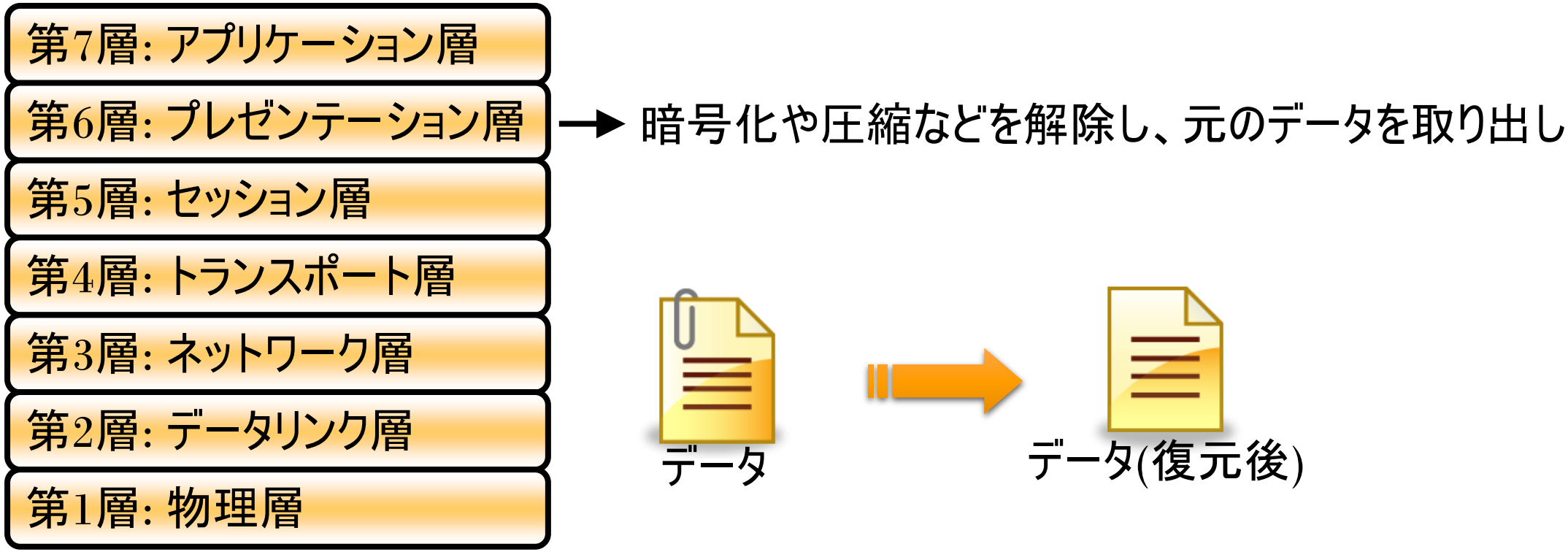
データ受信[9](p. 94)

- データ受信側



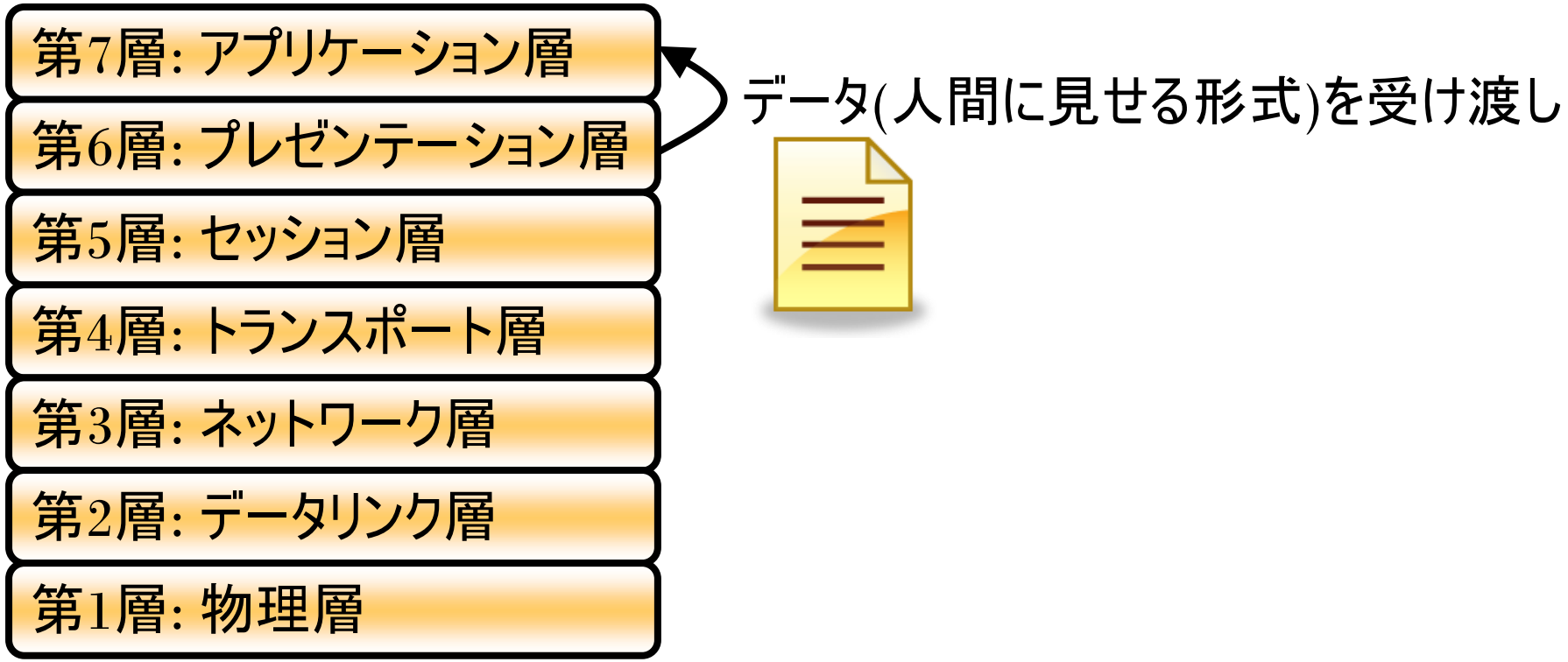
データ受信[10](p. 94)

- データ受信側



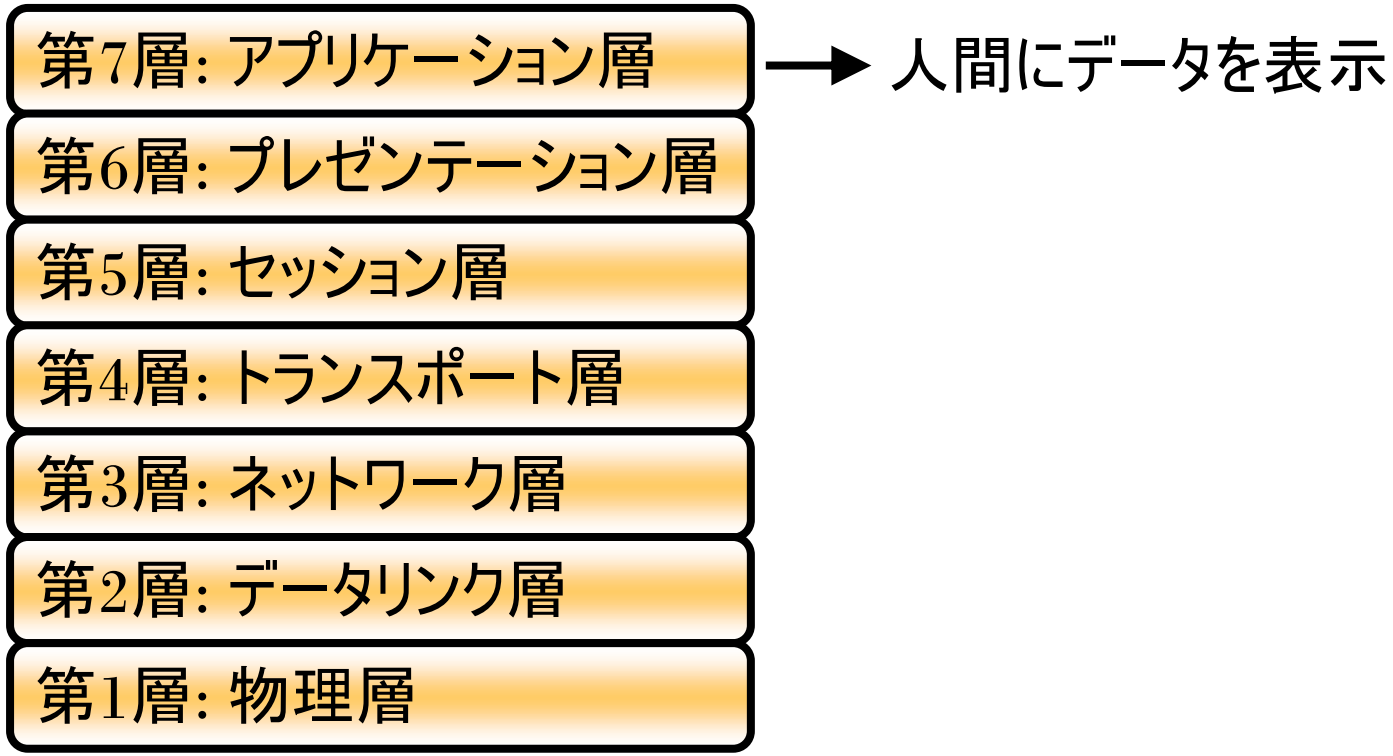
データ受信[11](p. 94)

- データ受信側



データ受信[12](p. 94)

- データ受信側



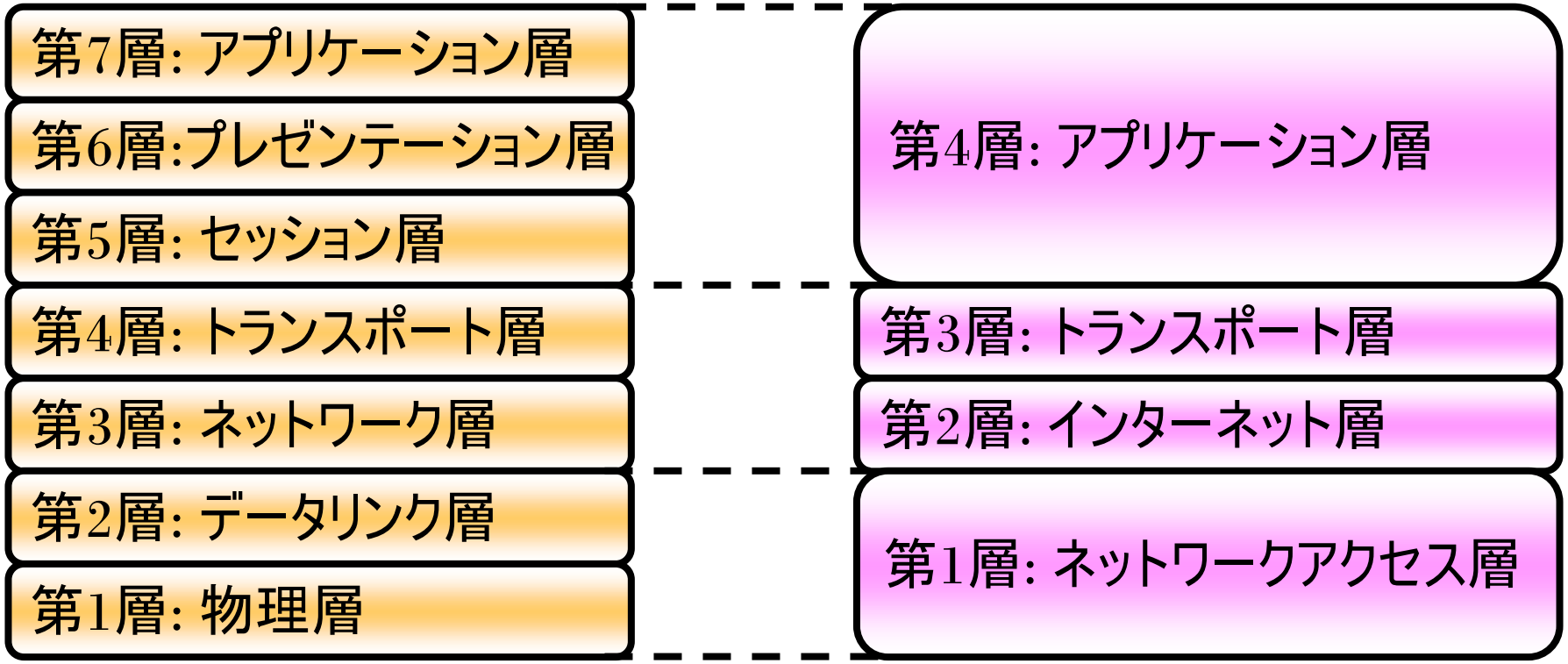
TCP/IPモデル

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



アプリケーション層(p. 96)

- OSIのアプリケーション・プレゼンテーション・セッション層の役割を含む
  - 人間が直接操作するソフトウェアに関する規定
  - データの表現形式の規定
  - 通信の開始時・終了時の合図を規定

トランスポート層(p. 96)

- 送信: アプリケーション層からのデータを分割し、エラー発見用の情報を付加してインターネット層へ受け渡し
- 受信: インターネット層からデータを受け取り、エラーの有無を確認

- プロトコルの規格: TCPとUDP

インターネット層(p. 96)

- 送信: トランスポート層からのデータに送り先の宛先を付加し、ネットワークアクセス層に受け渡し
- 受信: ネットワークアクセス層からデータを受け取り、送り先の宛先の情報を除いてトランスポート層に受け渡し

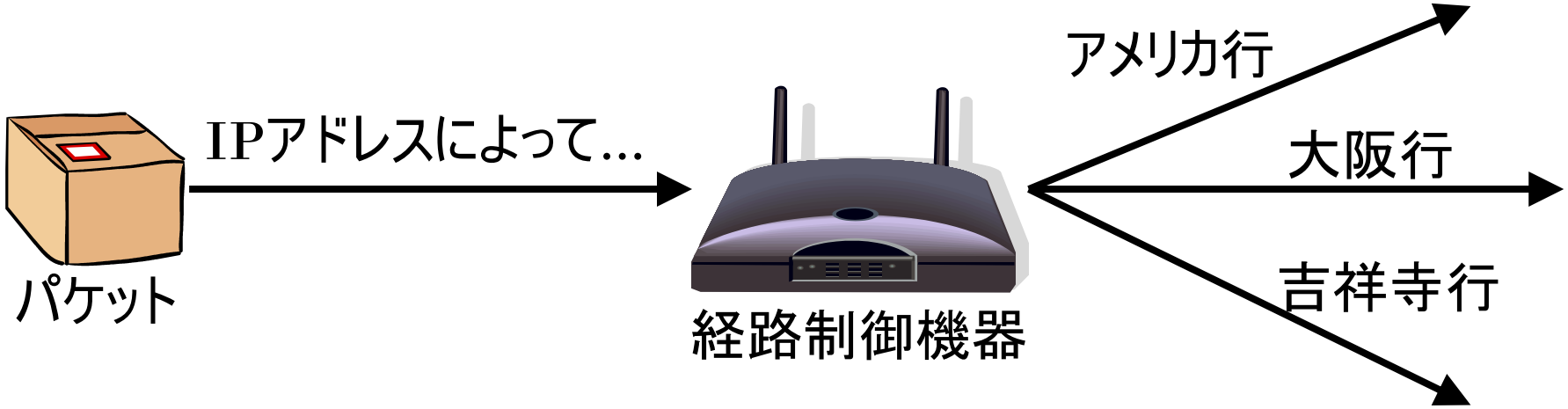
- プロトコルの規格: IP

ネットワークアクセス層(p. 96)

- 送信:
  - インターネット層からのデータにMACアドレス等の情報を付加
  - MACアドレス等の情報を付加されたデータを0と1の信号に変換して送り出し
- 受信:
  - 0と1の信号をもとのデータに復元
  - データからMACアドレス等の情報を削除し、インターネット層へ受け渡し

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

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