Politechnika Wrocławska

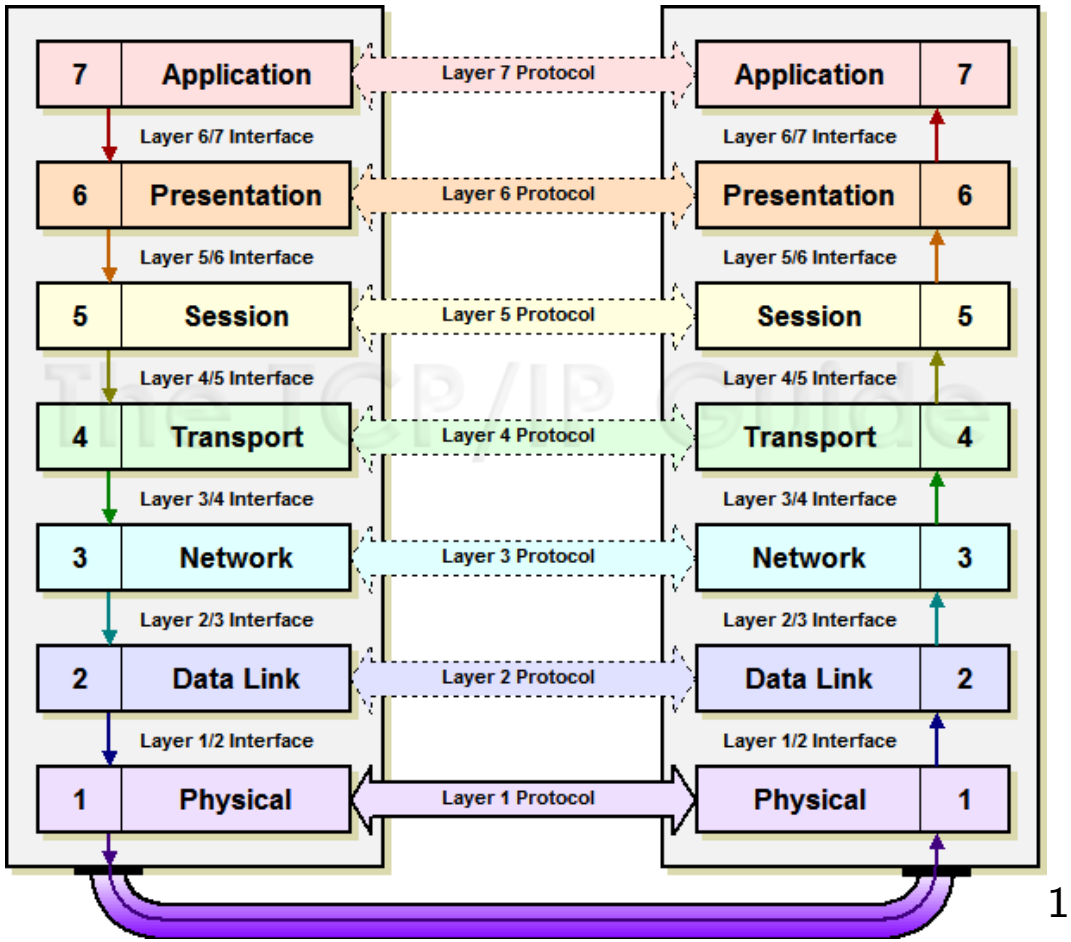
Katedra Cybernetyki i Robotyki

Protokoły komunikacji

Mariusz Janiak
p. 331 C-3, 71 320 26 44

Contents

Model OSI



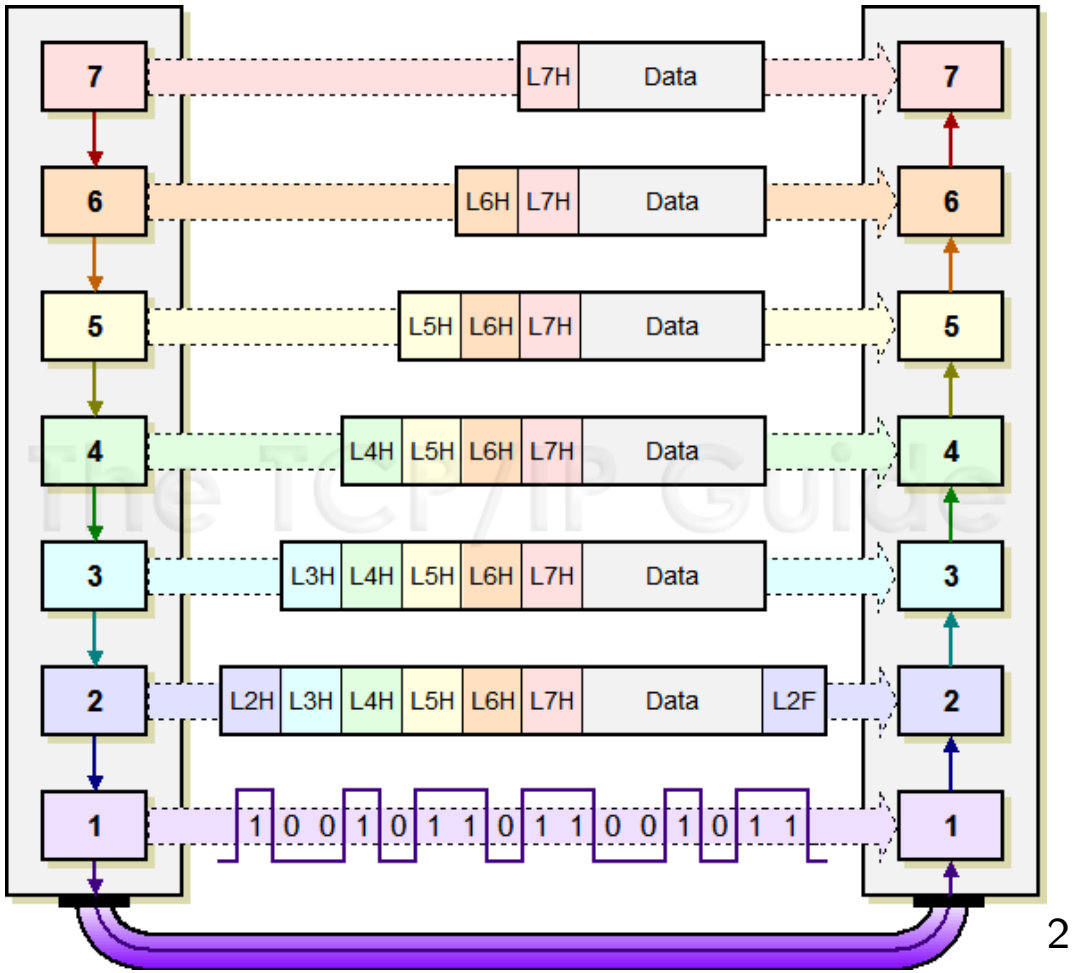
[1]

[1] http://www.tcpipguide.com

Model OSI

1. **Fizyczna** (Bity) – przesyłanie strumienia bitów poprzez fizyczne medium
2. **Łącza danych** (Bity/Ramka) – niezawodne przesyłanie sformatowanych ramek danych pomiędzy co najmniej dwoma węzłami podłączonymi za pośrednictwem warstwy fizycznej
3. **Sieciowa** (Pakiet) – budowanie i zarządzenie topologią sieci zbudowanej z wielu węzłów, adresowanie, trasowanie połączeń, kontrola ruchu
4. **Transportowa** (Segment) – niezawodne przesyłanie bloków danych pomiędzy węzłami sieci, segmentacja, potwierdzanie, multipleksowanie
5. **Sesji** (Dane) – zarządza komunikacją pomiędzy komputerami (sesją): zestawienie połączenia, sterowanie przepływem, synchronizacja, retransmisja,
6. **Prezentacji** (Data) – zarządza reprezentacją informacji, tłumaczy dane przesyłane pomiędzy aplikacjami za pośrednictwem sieci na format zgodny z wewnętrzną reprezentacją systemu docelowego
7. **Aplikacji** (Data) – definiuje interfejs aplikacji, dostarcza metod dostępu do środowiska OSI, pełni rolę okna między zdalnymi aplikacjami

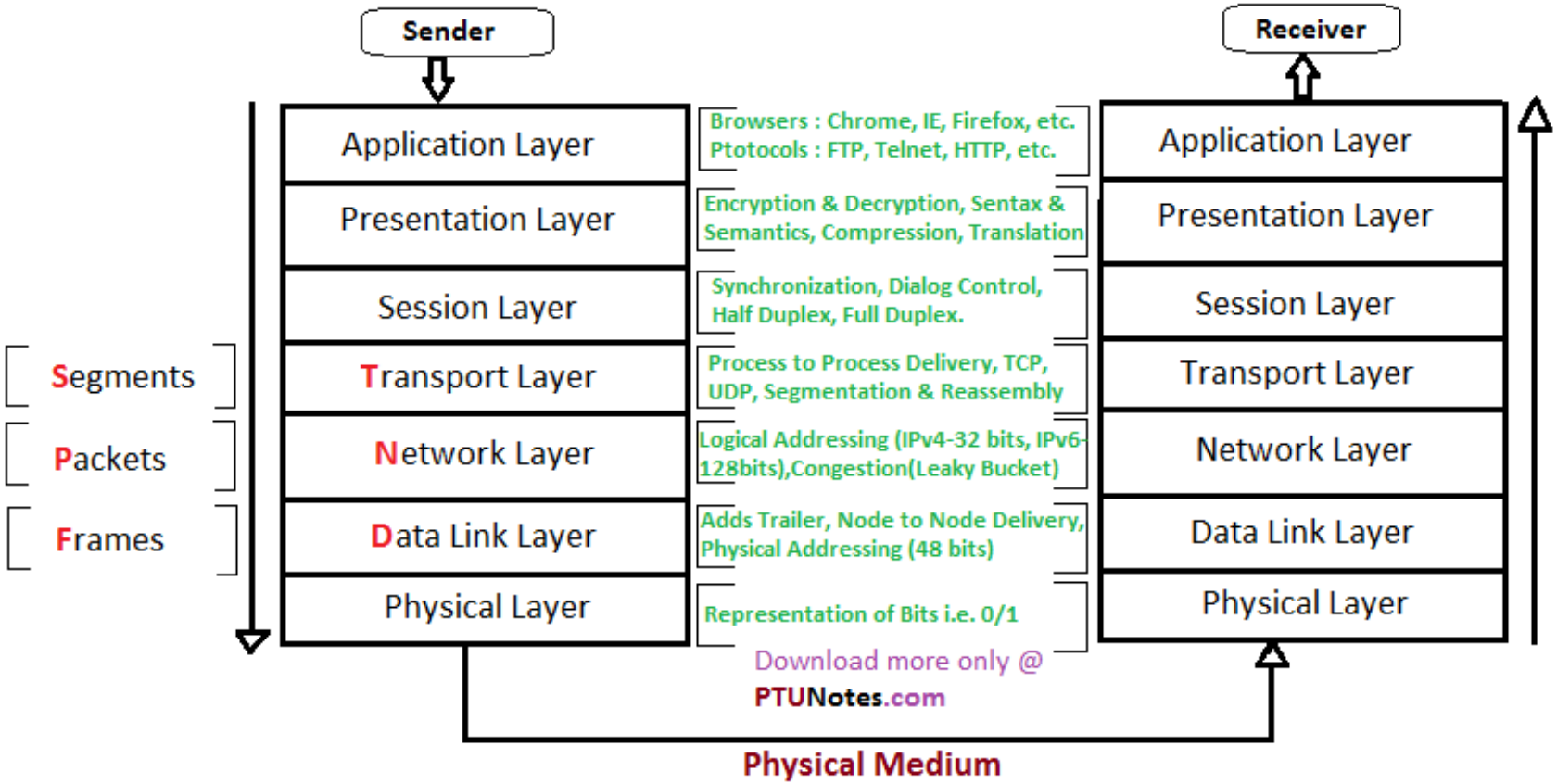
Model OSI



2

[2] http://www.tcpipguide.com

Model OSI



[3] http://ptunotes.com

Interfejsy sieciowe w systemach wbudowanych

Popularne interfejsy przewodowe

- RS232
- RS485/RS422
- CAN
- Ethernet
- USB
- ...

Interfejsy sieciowe w systemach wbudowanych

CAN Bus

- Charakterystyka
  - Rozmiar pola danych do 8B
  - half-duplex
  - Adresowanie 11b (CAN 2.0A) lub 29b (CAN 2.0B)
  - Szybkość transmisji 1Mb/s (to 40m)
  - Narzut – $64/111 = 0.576$ best, $64/135 = 0.473$ worst
  - Czas rzeczywisty – sprzętowy arbitraż.
- Multimaster, MultiCast
- Dobra detekcja błędów, procedury postępowania w przypadku awarii
- Protokoły: CANopen, DeviceNet, SAE J1939

Interfejsy sieciowe w systemach wbudowanych

Ethernet

- Charakterystyka
  - Rozmiar pola danych 1500B
  - Typowo full-duplex
  - Adresowanie 48b
  - Szybkość transmisji – zróżnicowana, obecnie standard 1GB/s
  - Słabe wsparcie dla czasu rzeczywistego – Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
- Może być wykorzystywany z/bez TCP lub UDP
- Wsparcie dla rozległych sieci (przełączniki sieciowe)
- Protokoły czasu rzeczywistego: RTnet, EtherCAT, ProfiNet, Powerlink, Ethernet-IP, Sercos, ...

Interfejsy sieciowe w systemach wbudowanych

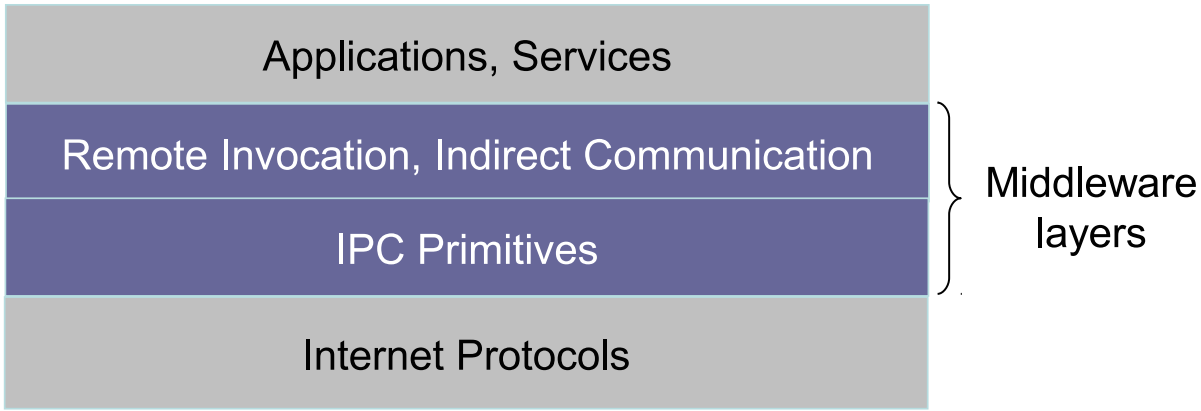
RTnet

- Sieciowy stos czasu rzeczywistego dla *Xenomai* i *RTAI*
- Licencja wolnego oprogramowania
- Wykorzystuje standardowy sprzęt sieciowy
- Wspiera popularne interfejsy sieciowe, również gigabitowe
- Implementuje UDP/IP, TCP/IP (podstawowe), ICMP i ARP w deterministyczny sposób
- Dostarcza standardowy interfejs POSIX socket API dla przestrzeni użytkownika i jądra
- Dostępna implementacja dla systemu FreeRTOS (RTmac, RTcfg, TDMA, Socket API, UDP/IP, ARP)

`www.rtnet.org`

Modele komunikacji

- Komunikacja międzyprocesowa (*Inter-Process Communication*)
- Zdalne wywołania (*Remote Invocation*)
- Komunikacja pośrednia (*Indirect Communication*)



Modele komunikacji

Komunikacja międzyprocesowa (IPC)

- Wsparcie dla niskopoziomowych mechanizmów komunikacji
  - Bezpośredni dostęp do protokołów sieciowych (gniazda)
- Gniazda stanowią zakończenia kanału komunikacyjnego, aplikacje mogą przeprowadzać operacje zapisu/odczytu
- Gniazda wykorzystywana są do przesyłania komunikatów za pośrednictwem warstwy transportowej
- Każde gniazdo stowarzyszone jest z wybranym protokołem warstwy transportowej
  - UDP – bezpołączeniowy, bez gwarancji poprawności danych
  - TCP – połączeniowy, niezawodny

Modele komunikacji

Komunikacja międzyprocesowa – gniazda UDP

- Protokół bezpołączeniowy, bez potwierdzeń i retransmisji
- Mechanizm komunikacji
  - Serwer otwiera gniazdo UDP *SS* na ustalonym porcie *sp*
  - Gniazdo *SS* czeka na nawiązanie połączenia
  - Klient otwiera gniazdo UDP *CS* na losowym porcie *cx*
  - Gniazdo klienta *CS* wysyła komunikat do IP serwera na port *sp*
  - Gniazdo serwera *SS* może odpowiedzieć gniazdu klienta *CS*

Modele komunikacji

Komunikacja międzyprocesowa – gniazda UDP (cd.)

- Komunikaty mogą zostać dostarczone w innej kolejności niż zostały wysłane
- Nie ma gwarancji dostarczenia danych
- Nadawca we własnym zakresie musi podzielić długi komunikat na mniejsze części przed jego wysłaniem (sugerowany rozmiar to 548B)
- Odbiorca powinien zarezerwować bufor odpowiedniej wielkości zdolny do pomieszczenia całej wiadomości

Modele komunikacji

Komunikacja międzyprocesowa – gniazda TCP

- Protokół połączeniowy, z potwierdzeniami i retransmisją
- Mechanizm komunikacji
  - Serwer otwiera gniazdo TCP *SS* na ustalonym porcie *sp*
  - Serwer czeka na połączenie (wywołanie *accept*)
  - Klient otwiera gniazdo TCP *CS* na losowym porcie *cx*
  - Gniazdo *CS* inicjuje połączenie z serwerem na porcie *sp* (wywołanie *connect*)
  - Gniazdo serwera *SS* rezerwuje nowe gniazdo *NSS* na losowym porcie *nsp* dla klienta
  - Gniazdo klienta *CS* może wysłać dane do gniazda serwera *NSS*

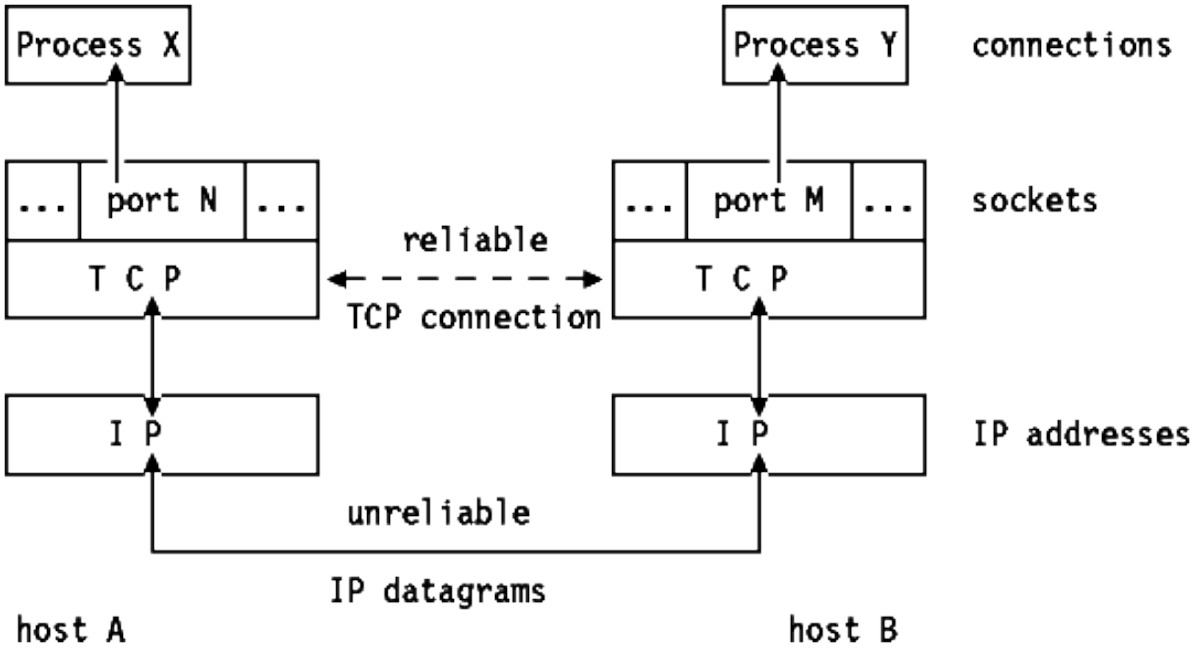
Modele komunikacji

Komunikacja międzyprocesowa – gniazda TCP (cd.)

- Gwarancja dostarczenia komunikatów ustalonej kolejności
- Aplikacje mogą wysyłać wiadomości dowolnej wielkości
- Gwarancja dostarczenia danych dzięki potwierdzeniom i retransmisji
- TCP kontroluje szybkość nadawania, przez co zapobiega przeciążeniu sieci

Modele komunikacji

Komunikacja międzyprocesowa (TCP)



4

[4] www.danzig.jct.ac.il

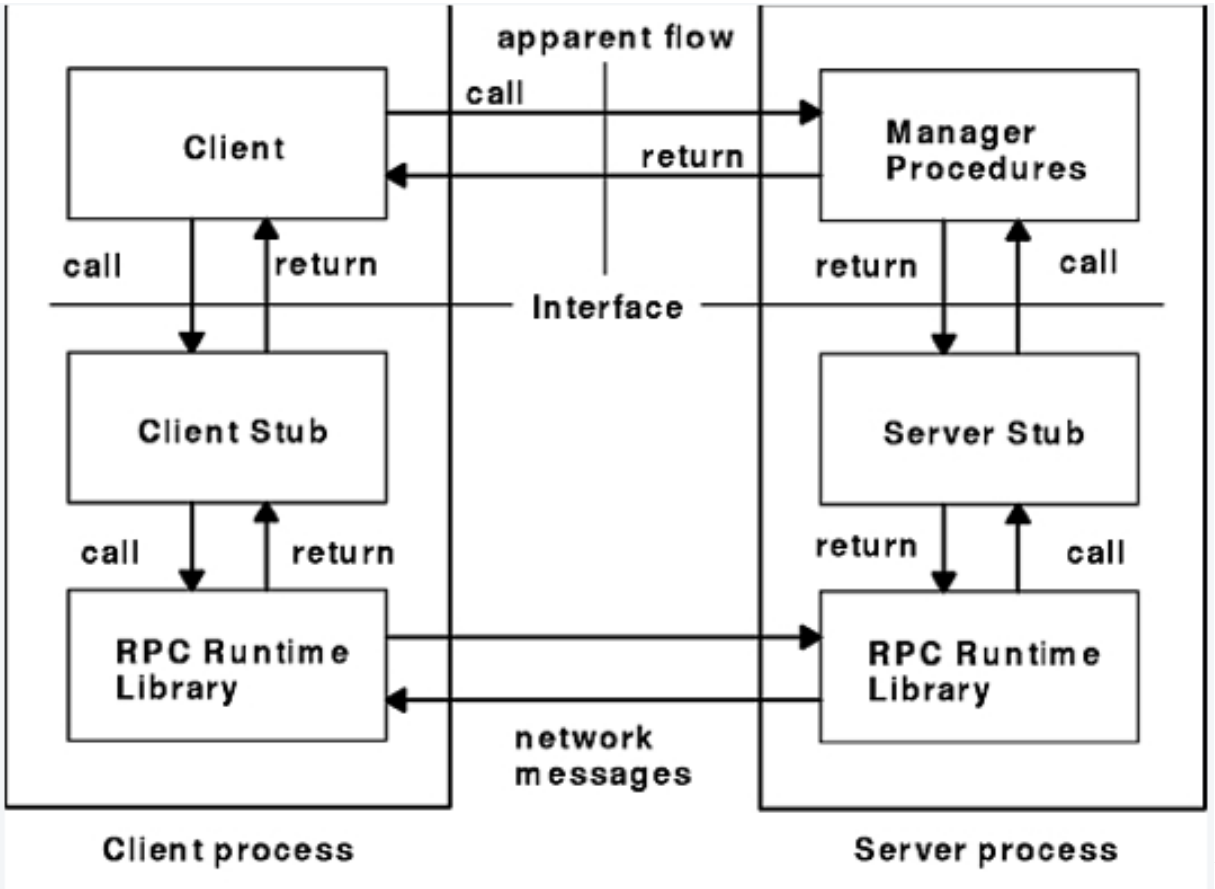
Modele komunikacji

Zdalne wywołania

- Wykonanie wybranej procedury na zdalnej maszynie
  - warstwa pośrednicząca (middleware) zarządza komunikacją niskopoziomową
  - komunikacja przezroczysta z punktu widzenia programisty
- Dwa modele komunikacji
  - Zdalne wywołanie procedury (*Remote Procedure Call*) – Sun RPC XML/RPC
  - Zdalne wywołanie metod (*Remote Method Invocation*) – CORBA, Java RMI
- RMI to obiektowa koncepcja komunikacji RPC

Modele komunikacji

Zdalne wywołania (cd.)



[5]

[5] blog.facilelogin.com

Modele komunikacji

Zdalne wywołania (cd.)

- RPC umożliwia nadawcy komunikację z odbiorcą przy użyciu prostego wywołania procedur – procedury komunikacji i przesyłania komunikatów są ukryte przed programistą
- Parametry przekazywane są z wykorzystaniem *serializacji*
- Reprezentacja danych musi być jednolita
- Przekazywanie danych przez: wartość, referencję (tylko RMI)

Modele komunikacji

Zdalne wywołania (cd.)

- Synchroniczne vs Asynchroniczne
  - Synchroniczne blokuje klienta do czasu odpowiedzi serwera – marnowanie zasobów klienta
  - Asynchroniczne wykorzystywane gdy klient nie oczekuje odpowiedzi serwera – serwer wysyła ACK zaraz po odebraniu wywołania od klienta
  - Odroczone synchroniczne – klient inicjuje asynchroniczne wywołanie po stornie serwera, serwer po ukończeniu realizacji zadania inicjuje wywołanie zwrotne w celu dostarczenia rezultatu zadania do klienta

Modele komunikacji

Zdalne wywołania (cd.)

- sprzężenie w przestrzeni – położenie (adres) zdalnej procedury musi być znany przed jej wywołaniem
- sprzężenie w czasie – po stronie serwera, proces musi oczekiwać na połączenie z klientem by zaakceptować zdalne wywołanie
- Ograniczona niezawodność – kaskadowa propagacja awarii
- Typowo wykorzystywany TCP – ograniczona skalowalność i determinizm czasowy
- Najlepiej przystosowany do niewielkich, zamkniętych systemów

Modele komunikacji

Komunikacja pośrednia

- Wykorzystuje warstwę pośrednicząca (middleware)
  - komunikacja jeden do wielu
  - mechanizmy eliminujące sprzężenie w czasie i przestrzeni
    - aplikacje nie muszą znać wzajemnych tożsamości
    - aplikacje nie muszą jednocześnie nasłuchiwać by się komunikować
- Wsparcie dla różnych protokołów warstwy transportowej
- Lepiej dopasowane do wydajnej komunikacji w czasie rzeczywistym
- Typy komunikacji pośredniej
  - Komunikacja grupowa (*Group Communication*)
  - Publikuj-Subskrybuj (*Publish-Subscribe*)
  - Dystrybucja danych (*Data-Distribution*)

Modele komunikacji

Komunikacja pośrednia – Komunikacja grupowa

- Komunikacja jeden do wielu – multicast
- Abstrakcja grup
  - Grupa w systemie reprezentowana jest przez *groupId*
  - Odbiorcy zgłaszają swój akces do grupy
  - Nadawca wysyła komunikat do grupy, który jest odbierany przez wszystkich jej członków
- Usługi warstwy pośredniczącej
  - Przynależność do grupy
  - Obsługa uszkodzeń jednego lub więcej członków grupy
- Efektywne wykorzystanie pasma
- Tożsamość członków grupy nie musi być znana węzłom
- Sprzężenie w czasie – dane nie są buforowane

Modele komunikacji

Komunikacja pośrednia – Publikuj-Subskrybuj

- Zorientowana na komunikaty, anonimowa komunikacja
  - Komunikaty wysyłane na tzw. Tematy (*Topic*)
  - Wiele odbiorców może subskrybować Temat (z/bez filtrowania)
  - Każdy komunikat dostarczony do subskrybenta (odfiltrowany)
- Uczestnicy są rozprzężeni
  - w przestrzeni: bez połączenia i znajomości wzajemnej tożsamości
  - w czasie: nie muszą działać w tym samym czasie
- Duża liczba producentów rozsyła informację do dużej liczby odbiorców
- Dobre rozwiązanie dla dynamicznych, zdecentralizowanych systemów

Modele komunikacji

Komunikacja pośrednia – Publikuj-Subskrybuj (cd.)

- Tylko wiadomości bez koncepcji danych
- Komunikaty przetwarzane są bez kontekstu
- Komunikaty dostarczane są najczęściej zgodnie z algorytmem FIFO, ewentualnie z priorytetami
- Brak buforowania danych
- Proste QoS: filters, durability, lifespan

Modele komunikacji

Komunikacja pośrednia – Dystrybucja danych

- Definiuje *Globalną Przestrzeń Danych* dostępną dla wszystkich zainteresowanych aplikacji
  - Obiekty danych adresowane przez Domenę, Temat i Klucz
  - Subskrybowanie jest niezależne od publikowania
  - Zestawienie połączenia (*contract*) ustalone za pomocą QoS
  - Automatyczne wykrywania i konfiguracja
- Subsystemy rozprzężone w czasie i przestrzeni
- Komunikaty służą aktualizacji obiektów danych
- Obiekty danych są reprezentowane przez Klucz
- Warstwa pośrednicząca zarządza stanem obiektów danych
- Obiekty danych są buforowane
- Zaawansowane QoS: Ownership, History, Deadline

Modele komunikacji

Sieciowe struktury ramowe

- Adaptive Communication Environment (ACE)
  http://www.dre.vanderbilt.edu/~schmidt/ACE.html
- ZeroC Ice
  https://zeroc.com
- ZeroMQ
  http://zeromq.org/
- OpenDDS
  http://www.opendds.org
- RTI Connext DDS
  https://www.rti.com
- ...

Reprezentacja danych

- Niejednorodność (każdy jest inny)
  - Różne systemy operacyjne
  - Różne języki programowania
  - Różne architektury sprzętowe
- Powstaje problem przesyłania złożonych struktur danych w postaci komunikatów w systemie rozproszonym

Reprezentacja danych

- Węzły mogą wykorzystywać różną reprezentację danych
  - rozmiar liczb stałoprzecinkowych, liczby zmiennoprzecinkowe, znaki (ASCII vs Unicode)
  - Big vs. Little endian
  - rozmieszczenie struktur danych w pamięci
    - wstawianie pustych pól, wyrównywanie danych
  - Wskaźniki i dane ustrukturyzowane
    - Reprezentacja wskaźników może się różnić
    - Drzewa, listy, itd. muszą być szeregowane
  - Obiekty i funkcje (zawierają kod!)
    - Zazwyczaj nie przesyłamy kodu

Reprezentacja danych

Marshalling, *Serialization*

- Serializacja to proces przekształcający zbiór ustukturyzowanych danych do postaci umożliwiającej ich przesłanie i/lub zapis.
- Deserializacja to proces przekształcający strumień danych w równoważny zbiór ustrukturyzowanych danych w miejscu docelowym
- Strategia 1: Odbiornik dopasowuje dane
  - Nadawca wysyła dane w swojej wewnętrznej reprezentacji
  - Odbiorca modyfikuje dane w razie potrzeby
- Strategia 2: Pośrednia reprezentacja kanoniczna
  - Nadawca serializuje dane do postaci kanonicznej
  - Odbiorca deserializuje dane do swojej postaci wewnętrznej

Reprezentacja danych

Schematy danych

- Typowanie jawne – samodokumentujące się dane (znaczniki)
  - dodatkowa informacja dodawana do komunikatu
  - np. ONC XDR (RFC 4506)
- Typowanie domyślne – znany format danych na obu końcach
  - współdziałanie zależne od dobrej specyfikacji protokołu
  - trudno wprowadzać modyfikacje
  - np. ISO's ASN.1, XML, Google Protocol Buffers, JSON

Reprezentacja danych

Struktury ramowe do szeregowanie danych

- Sun XDR
- ASN.1/BER
- CDR
- Java Object Serialization
- XML
- Google Protocol Buffers (Nanopb – C bez alokacji pamięci)
- Apache Thrift
- Boost Serialization
- MessagePack (CMP – C bez alokacji pamięci)
- Lightweight Communications and Marshalling (LCM)
- ...

Koniec

Dziękuję za uwagę.