Rozproszone systemy obiektowe

Grzegorz Jabłoński
Mariusz Orlikowski

Program wykładu

Wprowadzenie
Sockets
Internet Communications Engine
Windows Communication Foundation

Co to jest system rozproszony ?

- Poznajemy go po tym, że pracę uniemożliwia nam awaria komputera, o którym nigdy wcześniej nie słyszeliśmy – Leslie Lamport, 1987
- Zespół (być może) heterogenicznych węzłów połączonych w sieć, która zapewnia dostęp do dzielonych zasobów lub usług
- Zespół niezależnych komputerów, który widziany jest przez użytkownika jako jeden spójny system

Główne cechy systemu rozproszonego

- Wiele komputerów
  - Mogą być homo- lub heterogeniczne
- Połączonych w sieć
  - Typowo sieć ogólnego przeznaczenia, nie skonstruowaną celowo na potrzeby systemu rozproszonego
- Współpracujących w celu dzielenia zasobów lub usług
  - Program jest wykonywany na więcej niż jednym komputerze

Trendy technologiczne

- Przetwarzanie równoległe – podział obliczeń między wiele komputerów
- Dwa sposoby budowy takich systemów
  - Systemy wieloprocesorowe – ściśle powiązane, procesory współdzielą pamięć i zegar, komunikacja odbywa się przez pamięć dzieloną, wydzielona wewnętrzna sieć komunikacyjna
  - Systemy rozproszone – luźno powiązane, procesory nie dzielą pamięci i zegara, komunikacja odbywa się poprzez zewnętrzną sieć ogólnego przeznaczenia
- Tanie, wydajne komputery osobiste
- Wydajne sieci

Motywacja dla systemów rozproszonych

- Dzielenie zasobów
- Przyspieszenie obliczeń
  - Części składowe mogą działać współbieżnie
  - Równoważenie obciążenia
- Niezawodność
  - Przy odpowiedniej nadmiarowości częściowo uszkodzony system może pracować nadal
- Komunikacja
  - Wsparcie dla aplikacji do pracy grupowej

Terminologia

- Z punktu widzenia każdego procesora w systemie rozproszonym reszta procesorów i ich zasoby są zdalne, jego własne są lokalne
- Jednostki obliczeniowe w systemie rozproszonym bywają różnie nazywane
  - Hosty, węzły, komputery, maszyny, ...

Systemy operacyjne

- Sieciowe systemy operacyjne
  - Każdy komputer ma własny SO
  - SO zawiera usługi sieciowe umozliwiające dostęp do zdalnych zasobów
  - Użytkownik jest świadomy istnienia wielu komputerów i musi się tym jawnie zajmować
- Rozproszone systemy operacyjne
  - SO wielu komputerów współpracują ze sobą tworząc wrażenie pojedynczego systemu
  - Użytkownik jest nieświadomy istnienia wielu komputerów
- Middleware
  - Warstwa oprogramowania między SO i aplikacjami wspierającymi przetwarzanie rozproszone

Sieciowe systemy operacyjne

- Typowym przykładem są r-polecenia w systemie UNIX
  - Remote login (rlogin)
    - Przezroczyste, dwukierunkowe połączenie
  - Remote file transfer (rcp)
    - Lokalizacja plików nie jest przezroczysta dla użytkownika
    - Brak prawdziwego dzielenia plików
  - Rsh, rexec etc.
    - Usługi są implementowane jako procesy na zdalnym komputerze czekające na połączenia

Rozproszone systemy operacyjne

- Wbudowane wsparcie dla
  - Migracji danych
    - Całego pliku z miejsca A do miejsca B lub jedynie jego aktualnie niezbędnych części
  - Migracji obliczeń
    - Przeniesienie obliczeń w inne miejsce może być bardziej efektywne niż przeniesienie danych
  - Migracji procesów
    - Proces nie jest zawsze wykonywany na komputerze na którym został uruchomiony (równoważenie obciążeń lub niezawodność)

Cele projektowe systemów rozproszonych

- Przezroczystość
  - Z punktu widzenia użytkownika system rozproszony powinien wyglądać jak konwencjonalny, scentralizowany system
- Dlaczego trudno to osiągnąć
  - Inne typy uszkodzeń
  - Wydajność zależy od podziału obliczeń
- Innym aspektem przezroczystości jest mobilność użytkowników
  - Użytkownik siadający przy dowolnym komputerze w systemie powinien widzieć to samo
- Przezroczystość nie zawsze jest korzystna
  - “Dobre udogodnienie to takie, które można wyłączyć”
  - Potrzeba wiedzy o rozproszonej naturze w celu optymalizacji aplikacji

Inne cele: niezawodność

- Odporność na uszkodzenia
  - Dodanie komputerów do systemu zwiększa prawdopodobieństwo, że któryś z nich ulegnie uszkodzeniu
  - Tak jak w przypadku macierzy dyskowych RAID, chcemy z wady uczynić zaletę: stosujemy replikację stanu obliczeń, dzięki czemu system może pracować (być może mniej wydajnie) nawet przy występowaniu uszkodzeń
- System rozproszony powinien wspierać automatyczną reintegrację komputerów

Inne cele: skalowalność

- Zdolnośc systemu do adaptacji do zwiększonego obciążenia
  - Zmniejszenie wydajności i zajęte zasoby
  - Konieczność zapewnienia rezerwowych zasobów w celu zapewnienia niezawodności i poprawnej pracy przy zwiększonym obciążeniu (np. wytrzymanie “slashdot effect” w przypadku serwerów www)
- Zasada: zapotrzebowanie na usługi pojedynczego komponentu systemu powinno być ograniczone z góry przez pewną stałą, niezależnie od liczny węzłów w systemie

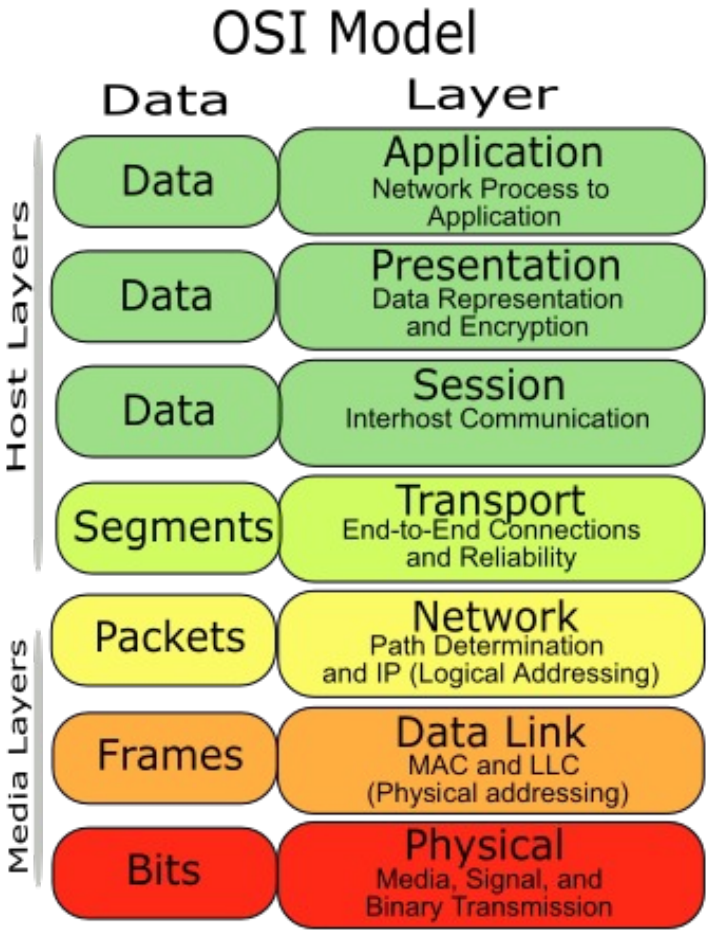
Architektura klient-serwer

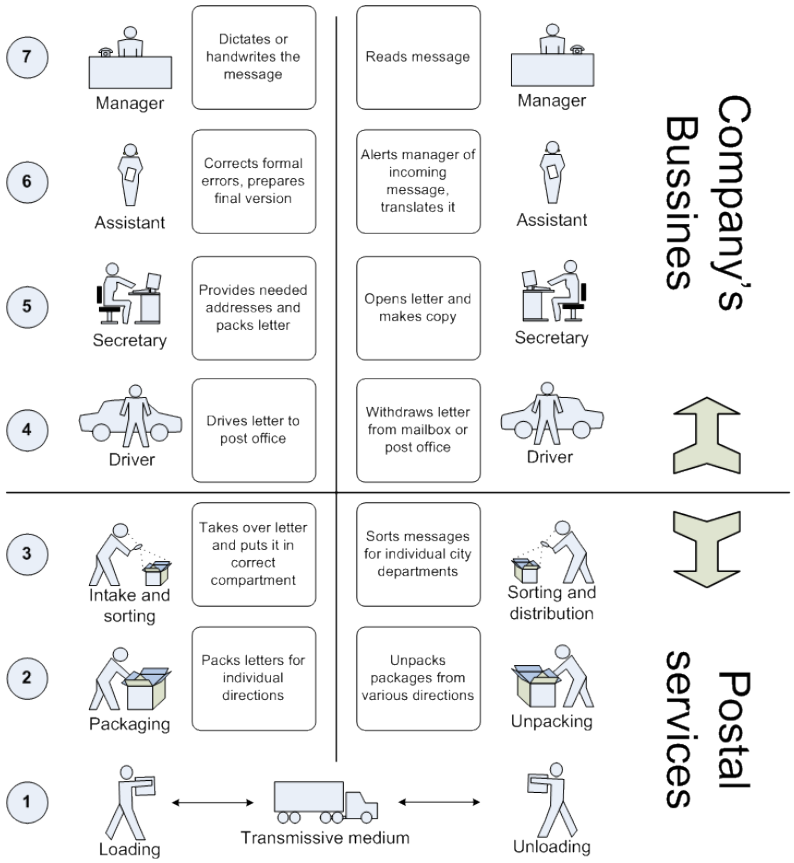
- Zadania aplikacji są podzielone między komputery klienckie i serwery
- Typowo serwery posiadają jakieś szczególne cechy, niemożliwe do bezpośredniego uzyskania w dużej liczbie klientów
  - Więcej pamięci, więcej lub szybsze procesory, licencje na kosztowne oprogramowanie etc.
- Wybór sposobu podziału aplikacji zależy od chrakterystyki klienta, serwera, aplikacji, sieci komunikacyjnej i spodziewanego obciążenia systemu
- Thin client vs. thick client (rich client)

Komunikacja w systemach rozproszonych

- Brak fizycznej pamięci dzielonej, konieczność przesyłania komunikatów
- Podstawowe operacje to Send() i Receive()
- Client wysyła zapytanie i czeka na odbiór odpowiedzi. Serwer odbiera zapytanie i wysyła odpowiedź
- Dane muszą być w jakiś sposób zakodowane w ciele komunikatu
- Komunikaty mogą być zawodne lub niezawodne (w przypadku zawodnych, wyższa warstwa musi zapewnić niezawodność transmisji)
- Operacje mogą być blokujące i nieblokujące
- Może być oparta na specjalizowanym protokole lub protokole ogólnego przeznaczenia (np. TCP/IP)

OSI Reference Model





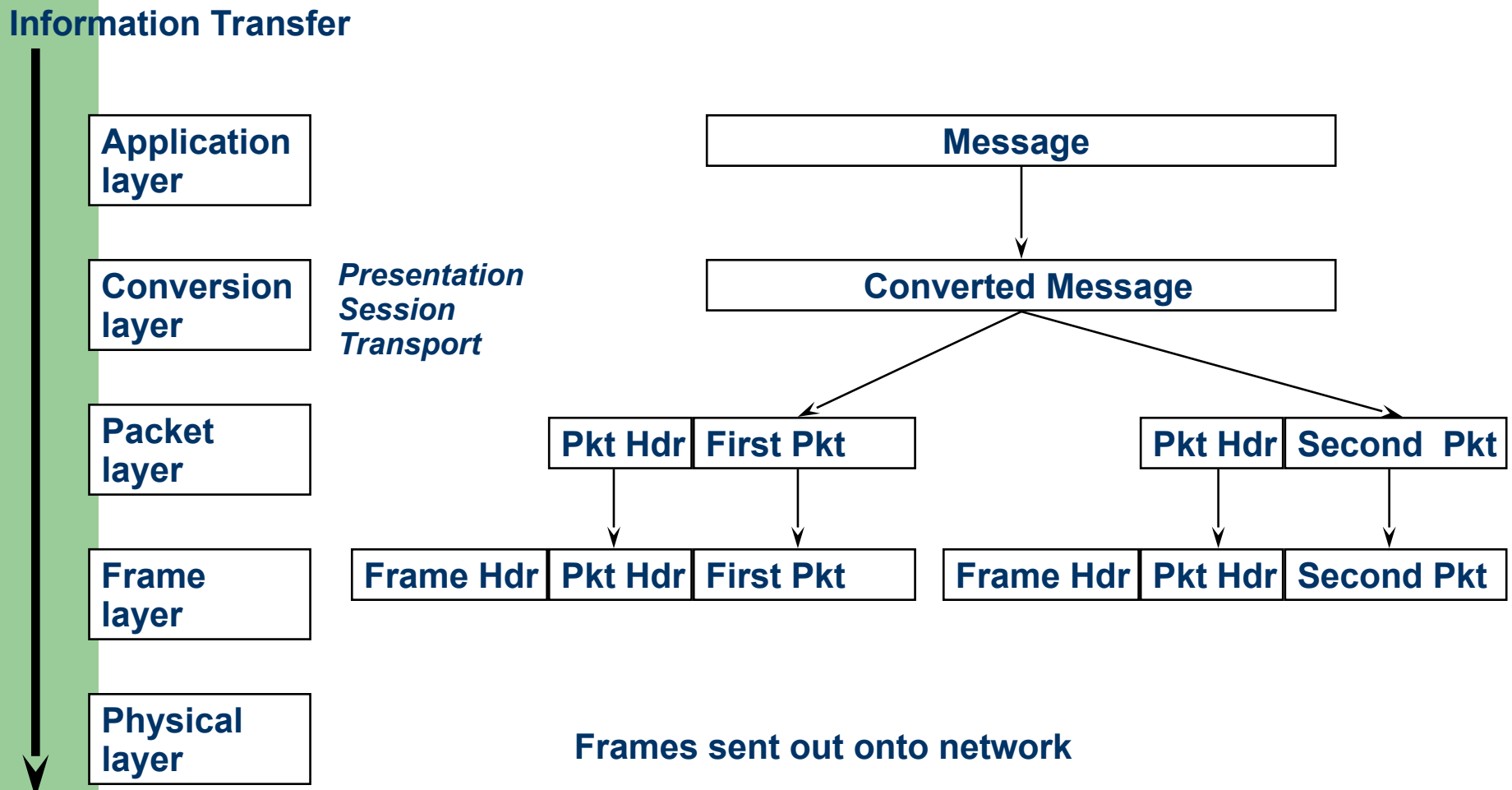
RM – OSI and letter communication parallel

1 April 1990 - RFC 1149

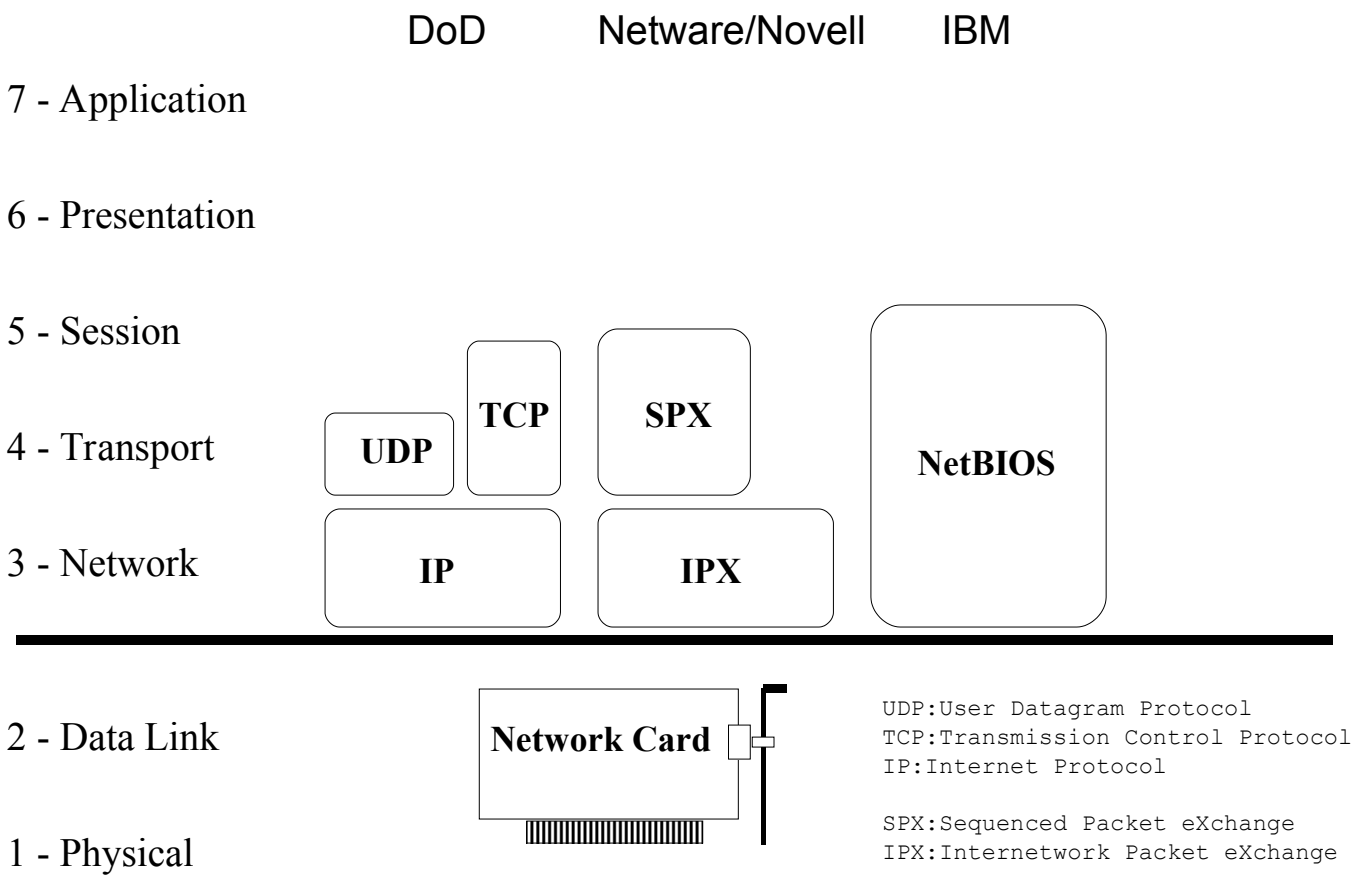
Standard for the transmission of IP datagrams on avian carriers
Implementation: http://www.blug.linux.no/rfc1149/

```
Script started on Sat Apr 28 11:24:09 2001
vegard@gyversalen:~$ /sbin/ifconfig tun0
tun0   Link encap:Point-to-Point Protocol
       inet addr:10.0.3.2  P-t-P:10.0.3.1  Mask:255.255.255.255
       UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:150  Metric:1
       RX packets:1 errors:0 dropped:0 overruns:0 frame:0
       TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0
       RX bytes:88 (88.0 b)  TX bytes:168 (168.0 b)

vegard@gyversalen:~$ ping -i 900 10.0.3.1
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=6388671.9 ms

--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss
round-trip min/avg/max = 3211900.8/5222806.6/6388671.9 ms
vegard@gyversalen:~$ exit

Script done on Sat Apr 28 14:14:28 2001
```

Implementacja warstw



Kilka ważniejszych protokołów



Implementowany protokół

- Zaimplementować serwer pierwiastkujący liczby rzeczywiste i podający bieżący czas na serwerze
  - Oparty na protokole TCP
  - W celu zapewnienia przenośności między różnymi procesorami wszystkie liczby przesyłamy w standardzie Big Endian
  - RQ ID pozwala na rozróżnienie różnych zapytań

Pytanie o pierwiastek:

0	0	0	1	RQ ID	Liczba (IEEE double)

Odpowiedź:

1	0	0	1	RQ ID	Pierwiastek (IEEE double)

Implementowany protokół

- Datę i czas przesyłamy w postaci tekstowej, bez kończącego zera
- Długość podajemy w kolejności Big Endian
- W jednym połączeniu można przesłać kilka zapytań
- Kolejność odpowiedzi może być inna, niż kolejność zapytań

Pytanie o czas:

0	0	0	2	RQ ID

Odpowiedź:

1	0	0	2	RQ ID	Długość (BE)	Data i czas