

# **Algorytmy rozproszone**

## **Wykład 1: Wprowadzenie**

**dr hab. inż. Anna Kobusińska, prof. PP**

**Anna.Kobusinska@cs.put.poznan.pl  
www.cs.put.poznan.pl/akobusinska**

# Plan wykładów

- Systemy rozproszone
  - wstęp
  - podstawowe zagadnienia: model systemu, czas wirtualny, stan globalny
- Mechanizmy rozgłaszenia niezawodnego
- Zwielokrotnianie i spójność
- Rozproszone wzajemne wykluczanie
- Problem detekcji zakończenia
- Algorytmy konsensusu

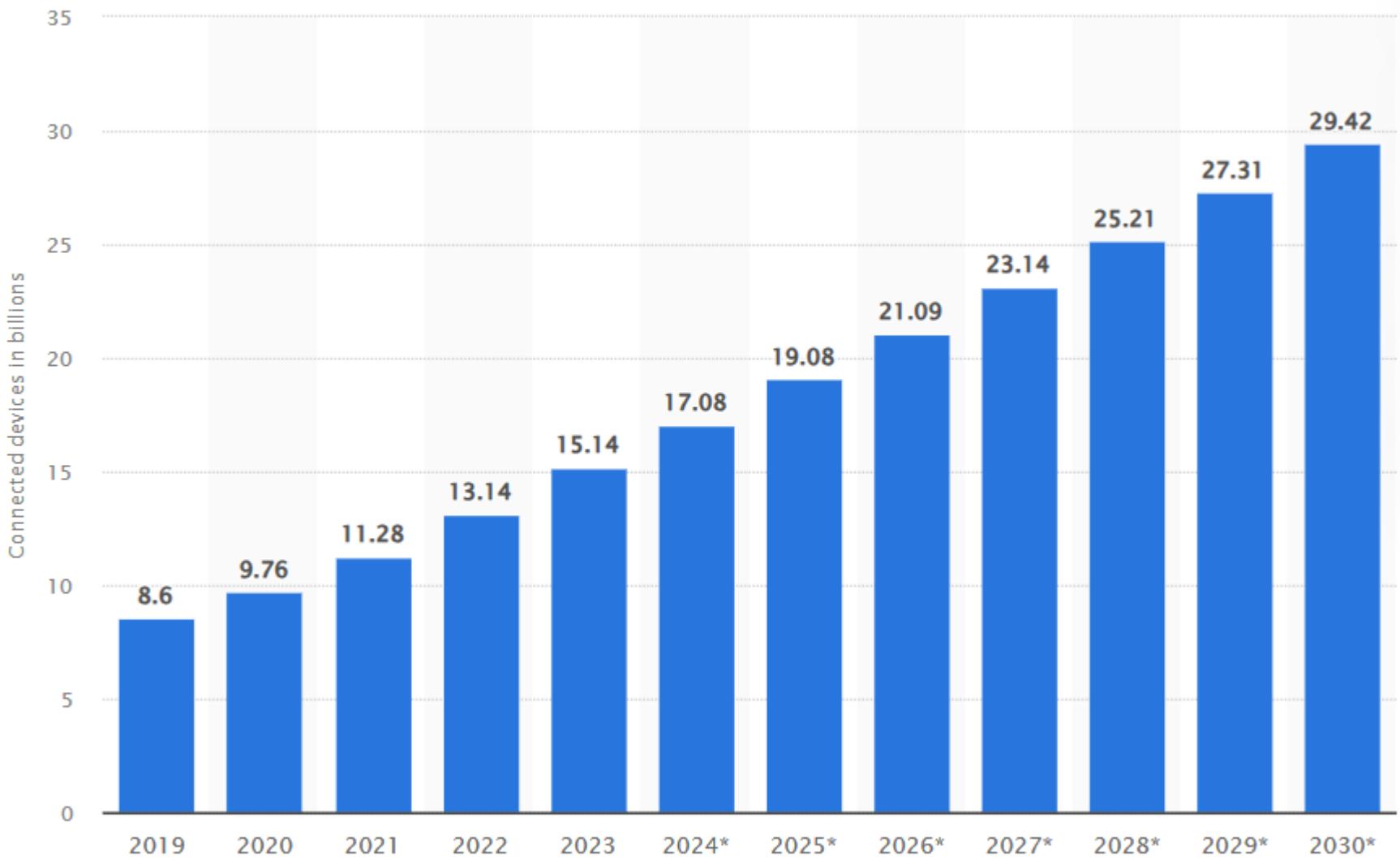
# Literatura

- Andrew S. Tanenbaum, Maarten Van Steen, *Distributed Systems: Principles and Paradigms*
- Cachin, Guerraoui & Rodrigues, *Introduction to Reliable and Secure Distributed Programming*
- George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair, *Distributed Systems: Concepts and Design*
- Martin Kleppmann, *Designing Data-Intensive Applications*

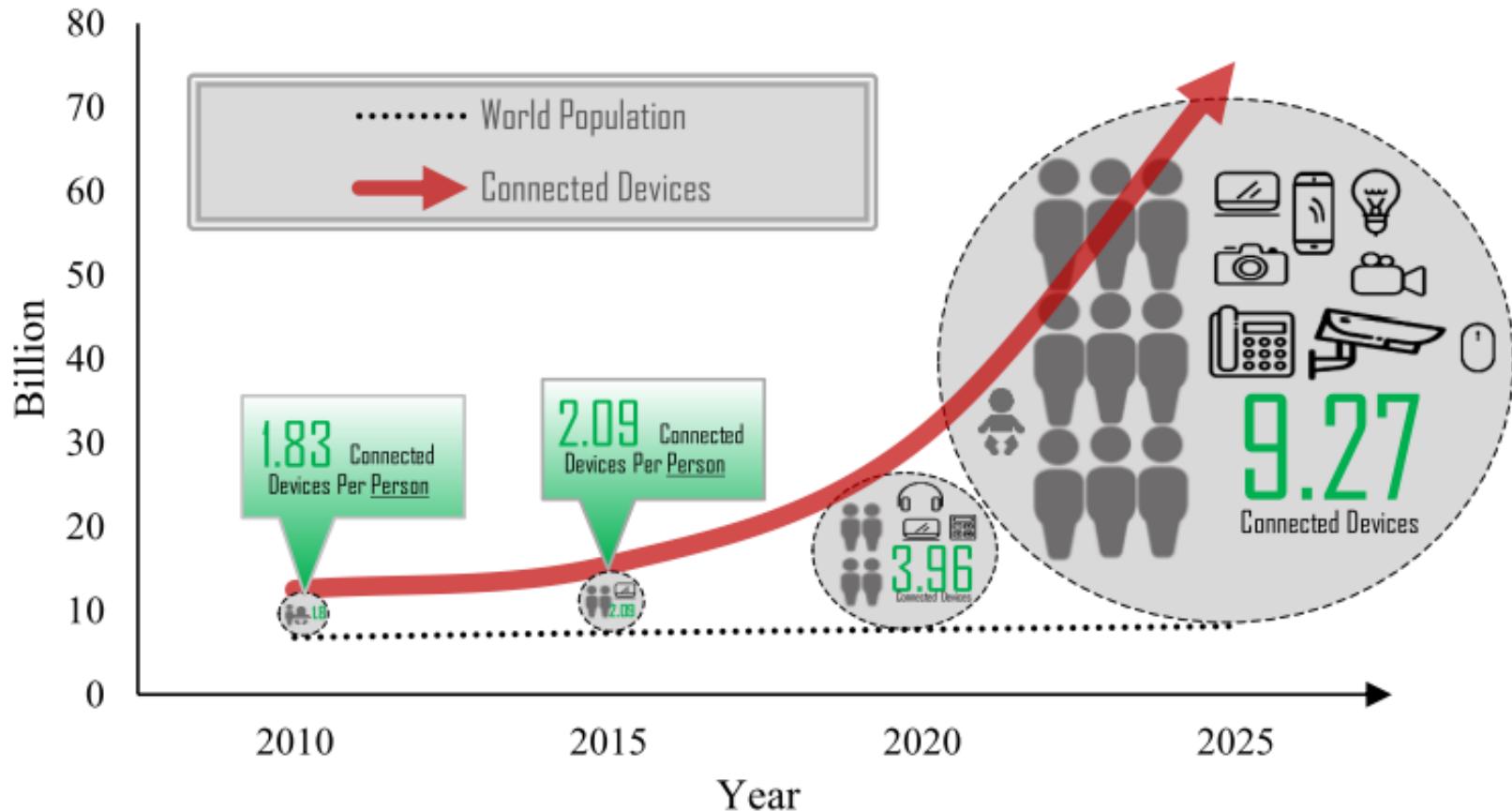
# Rozwój technologiczny

- Postęp technologiczny
  - 10 mln dolarów, 1 instrukcja na sekundę
  - 1000 dolarów, 100 mln instrukcji na sekundę
  - $12^{12}$  razy lepszy współczynnik cena/efektywność
- Zapotrzebowanie użytkowników na usługi zwiększyło skalę systemów (Facebook ma 500 milionów użytkowników)
- Żyjemy w społeczeństwie sieciowym

# Rosnąca liczba urządzeń



# Rosnąca liczba użytkowników



JAN  
2024

# DIGITAL GROWTH

CHANGE IN THE USE OF CONNECTED DEVICES AND SERVICES OVER TIME



TOTAL  
POPULATION



Meltwater

**+0.9%**

YEAR-ON-YEAR CHANGE

**+74 MILLION**

UNIQUE MOBILE  
PHONE SUBSCRIBERS



KEPIOS

**+2.5%**

YEAR-ON-YEAR CHANGE

**+138 MILLION**

INDIVIDUALS USING  
THE INTERNET



we  
are  
social

**+1.8%**

YEAR-ON-YEAR CHANGE

**+97 MILLION**

SOCIAL MEDIA  
USER IDENTITIES



**+5.6%**

YEAR-ON-YEAR CHANGE

**+266 MILLION**

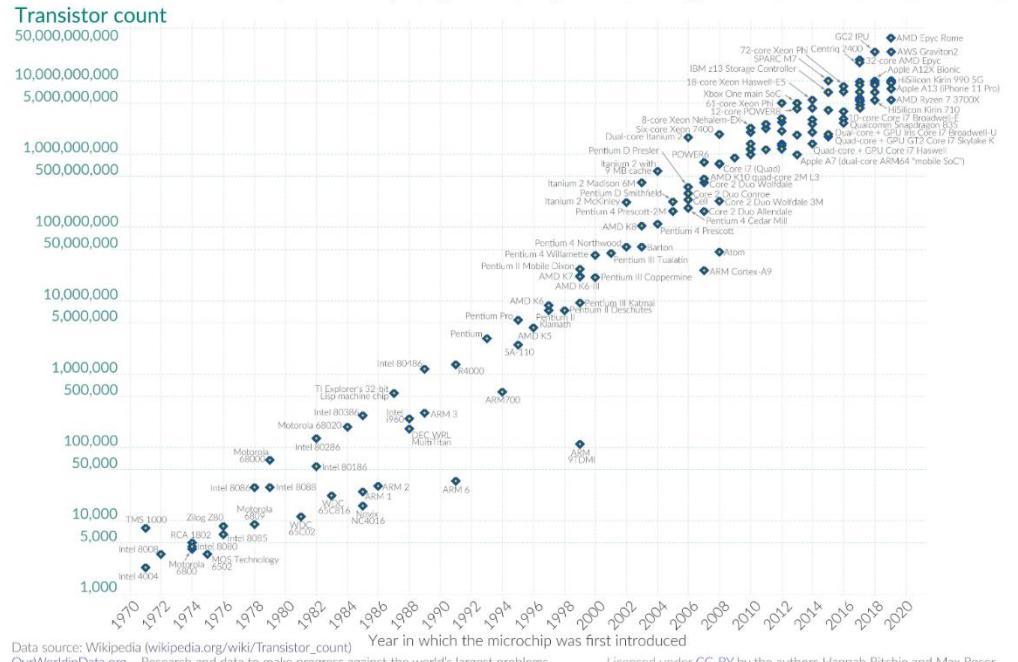
# Wydajność

- Skalowanie pojedynczego systemu ma ograniczenia
  - Technologia coraz trudniej nadąża za prawem Moore'a
  - Istnieją ograniczenia dotyczące rozmiaru matrycy i liczby tranzystorów
  - Więcej rdzeni na chip wymaga programowania wielowątkowego

**Moore's Law: The number of transistors on microchips doubles every two years**

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing - such as processing speed or the price of computers.

**Our World in Data**



Prawo Moore'a: każdego roku liczba tranzystorów w układach scalonych powinna się podwajać, a koszt ich produkcji maleć o połowę.

# Potrzeby obliczeniowe przekraczają możliwości procesorów

Renderowanie filmów:

- Toy Story (1995) - 117 komputerów; 45 minut - 30 godzin na wypłynięcie klatki
- Toy Story 4 (2019) - 60-160 godzin na wypłynięcie klatki

Google

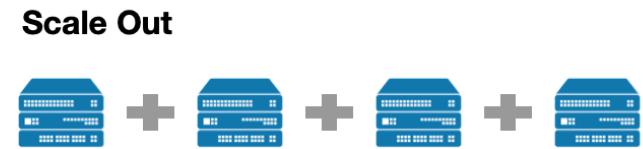
- Średnio ponad 63 000 zapytań na sekundę
- Ponad 130 bilionów zindeksowanych stron
- Wykorzystuje do tego setki tysięcy serwerów

Facebook

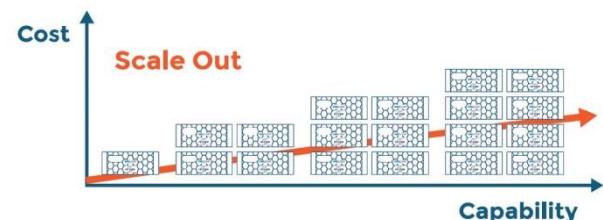
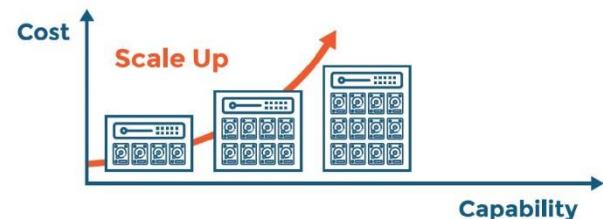
- Około 100 milionów zapytań na sekundę przy 4 miliardach użytkowników

# Co jeśli potrzebujemy większej wydajności niż pojedynczy procesor?

- **Scale up or scale vertically:**  
dodawanie zasobów do pojedynczego węzła w systemie.
- **Scale out or scale horizontally:**  
dodawanie kolejnych węzłów do systemu.
- Scale up: droższe
- Scale out: większe wyzwanie dla zapewnienia odporności na błędy i rozwoju oprogramowania.



## Scale Up vs. Scale Out

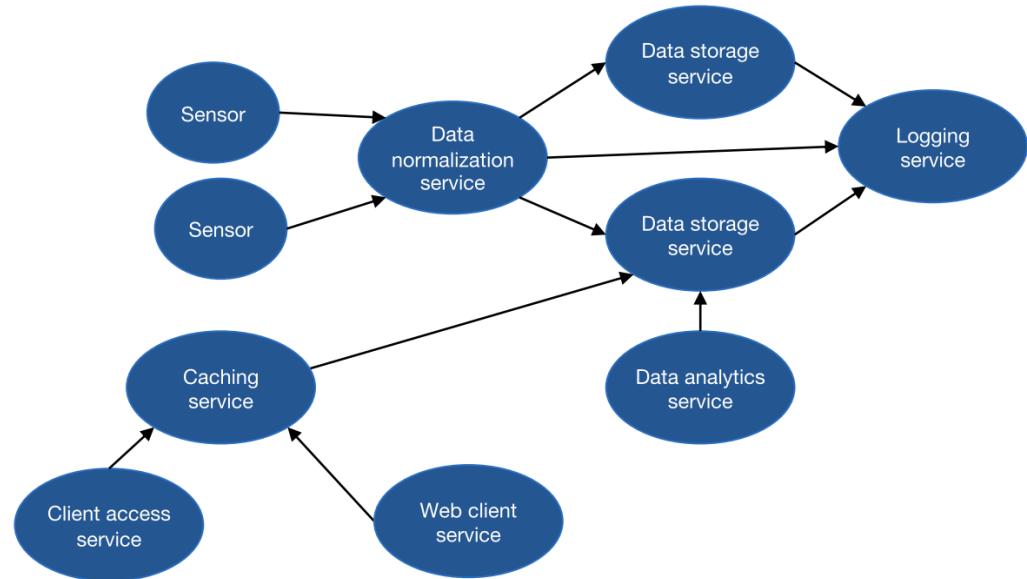


# System rozproszony – definicja

Zbiór niezależnych komputerów połączonych poprzez sieć komunikacyjną, które współpracują, aby osiągnąć określony cel

Wikipedia

System rozproszony to zbiór usług dostępnych za pośrednictwem interfejsów sieciowych



# System rozproszony – definicja

Komponenty (sprzętowe lub programowe) działające na komputerach połączonych siecią, które komunikują się lub koordynują swoje działania tylko poprzez wymianę wiadomości.

Coulouris, Dollimore, Kindberg, Blair

Zbiór autonomicznych elementów obliczeniowych, połączonych siecią, które sprawiają na użytkownikach końcowych wrażenie pojedynczego, spójnego systemu.

Steen and Tanenbaum

# Systemy rozproszone – motywacja

- ogromne rzeczywiste zapotrzebowanie na systemy rozproszone
- dostępność środków technicznych i praktyczne możliwości realizacji systemów rozproszonych
- różnorodność otwartych problemów związanych z konstrukcją i zarządzaniem systemami rozproszonymi

A screenshot of a web browser window on a Mac OS X system. The address bar shows the URL <https://cat.put.poznan.pl/>. Below the address bar, there is a large, dark silhouette of a Tyrannosaurus Rex walking towards the left. Underneath the dinosaur, the text "No Internet" is displayed in a large, bold, black font. Below this, the word "Try:" is followed by a bulleted list of troubleshooting steps:

- Checking the network cables, modem and router
- Reconnecting to Wi-Fi

# Systemy rozproszone - demotywacja

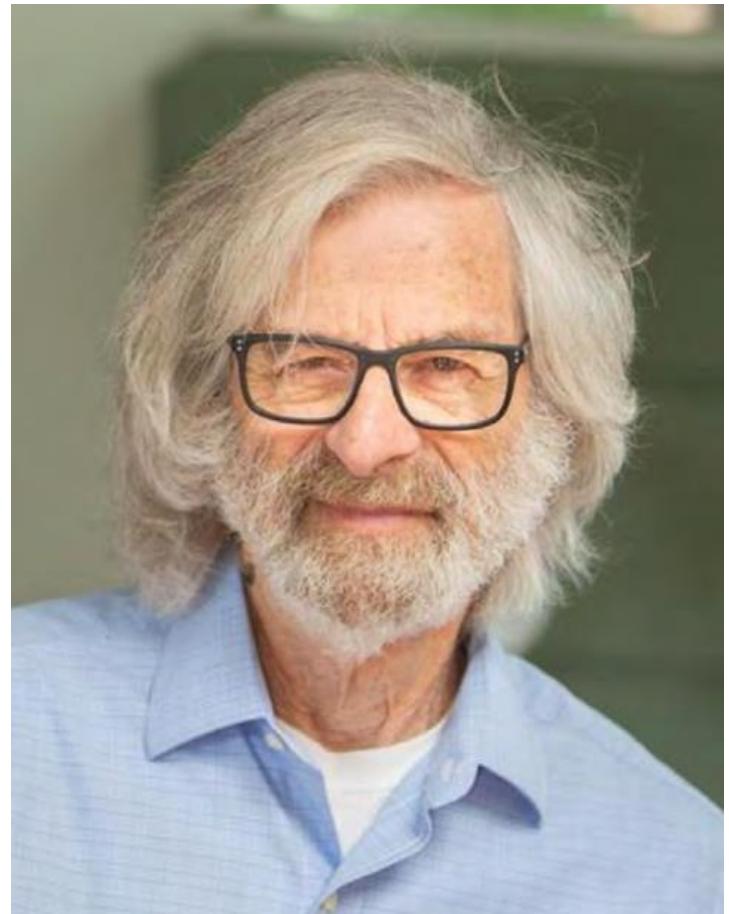
Problem z systemami rozproszonymi:

- Komunikacja może zawieść (a my możemy nawet o tym nie wiedzieć).
- Procesy mogą ulec awarii (a my możemy o tym nie wiedzieć).
- Wszystko to może działać się w sposób niedeterministyczny.

# System rozproszony – definicja alternatywna

System jest rozproszony, gdy awaria komputera, o której nigdy nie słyszałeś, powstrzymuje cię przed wykonaniem jakiejkolwiek pracy.

— Leslie Lamport



# Systemy rozproszone – charakterystyka

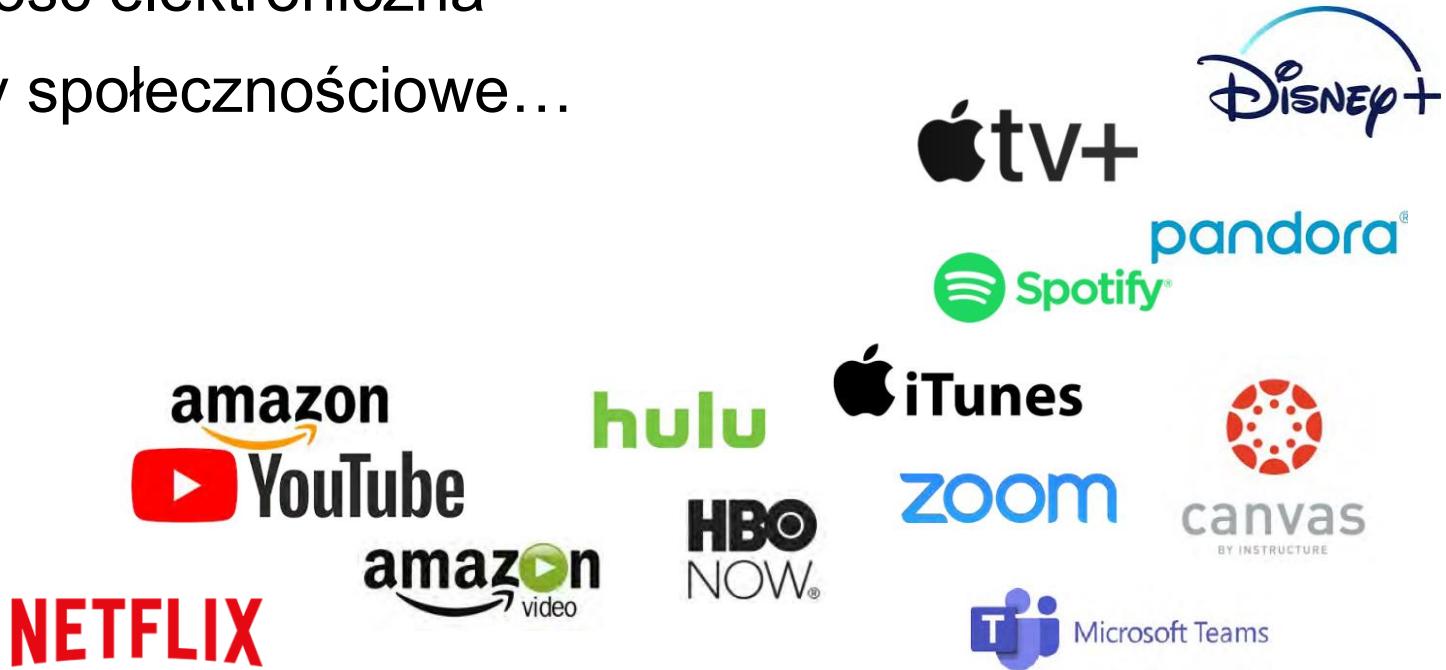
- Naturalność rozproszenia:
  - np. wysyłanie wiadomości z telefonu komórkowego, wykonanie transferu bankowego, dodanie komentarza na FB
- Większa niezawodność :
  - Nawet gdy jeden z węzłów ulegnie awarii, system jako całość działa dalej
- Większa wydajność:
  - Pobieranie danych z węzłów znajdujących się w bliskiej geograficznej odległości
- Możliwość przetwarzania w dużej skali:
  - np. masywnych ilości danych, które nie mogą być przetwarzane na pojedynczym węźle

# Internet



# Przykłady systemów rozproszonych

- Usługi chmurowe (AWS, Azure, GCP, Amazon EC2, S3)
- Gry on-line dla wielu graczy (MMOG)
- BitTorrent (systemy peer-to-peer)
- Blockchain i kryptowaluty
- Bankowość elektroniczna
- Systemy społecznościowe...





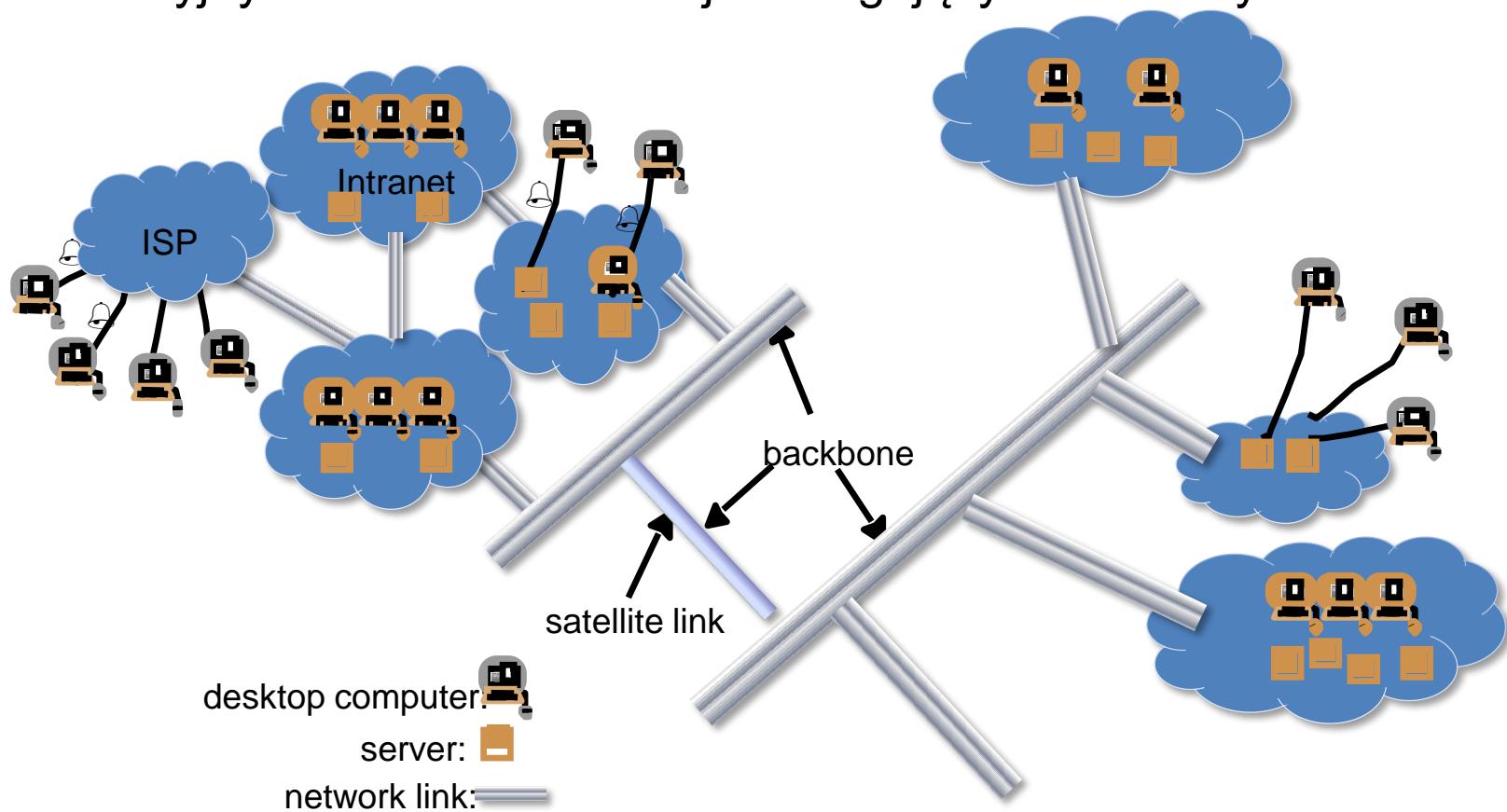
# Przykłady systemów rozproszonych

Są one oparte na znanych i powszechnie używanych sieciach komputerowych:

- Internet
- Intranet
- Sieci bezprzewodowe

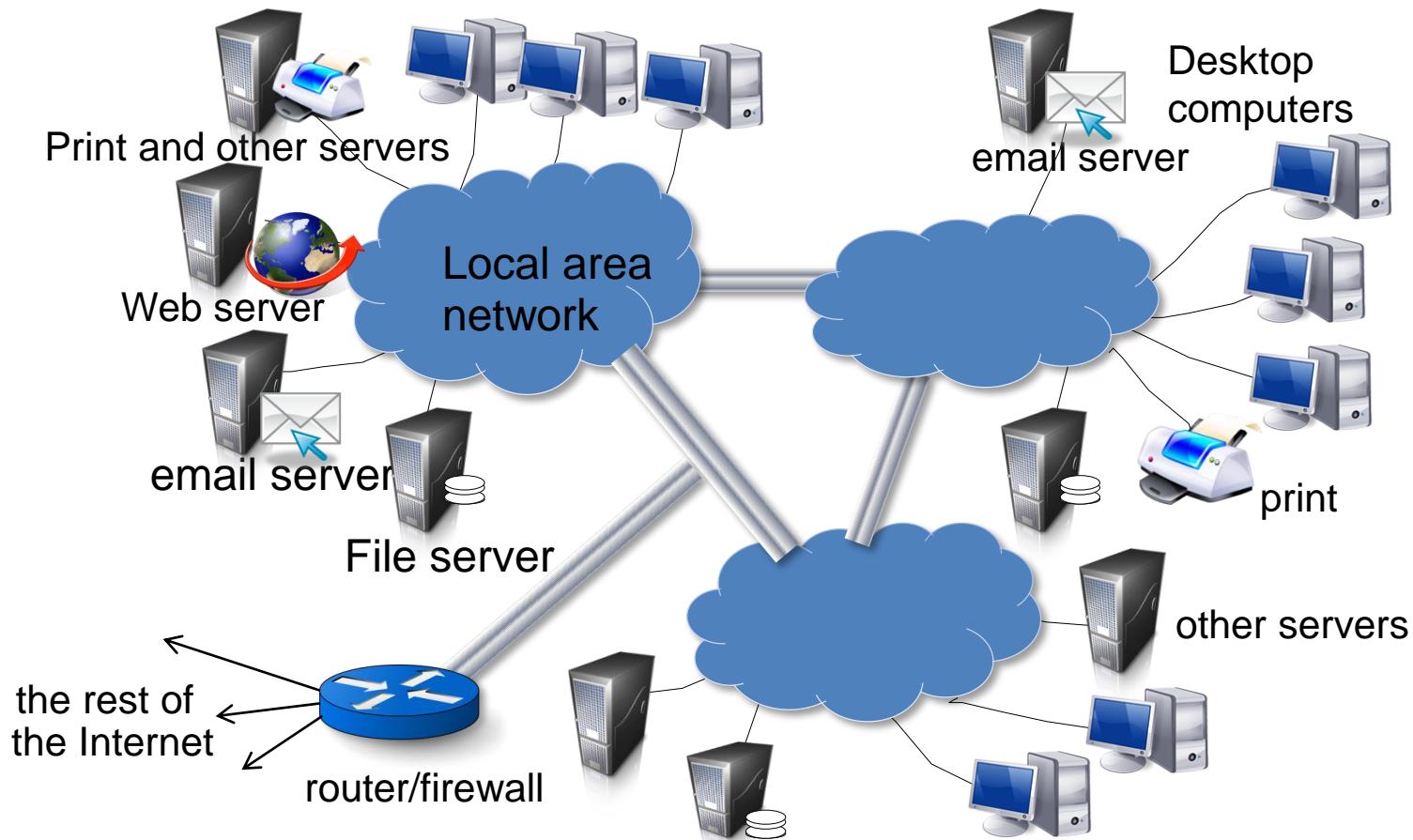
# Internet i jego usługi

- Internet jest rozległym zbiorem sieci komputerowych wielu różnych typów i hostuje różne rodzaje usług.
- Usługi multimedialne zapewniające dostęp do muzyki, radia, kanałów telewizyjnych i wideokonferencji obsługujących kilku użytkowników.



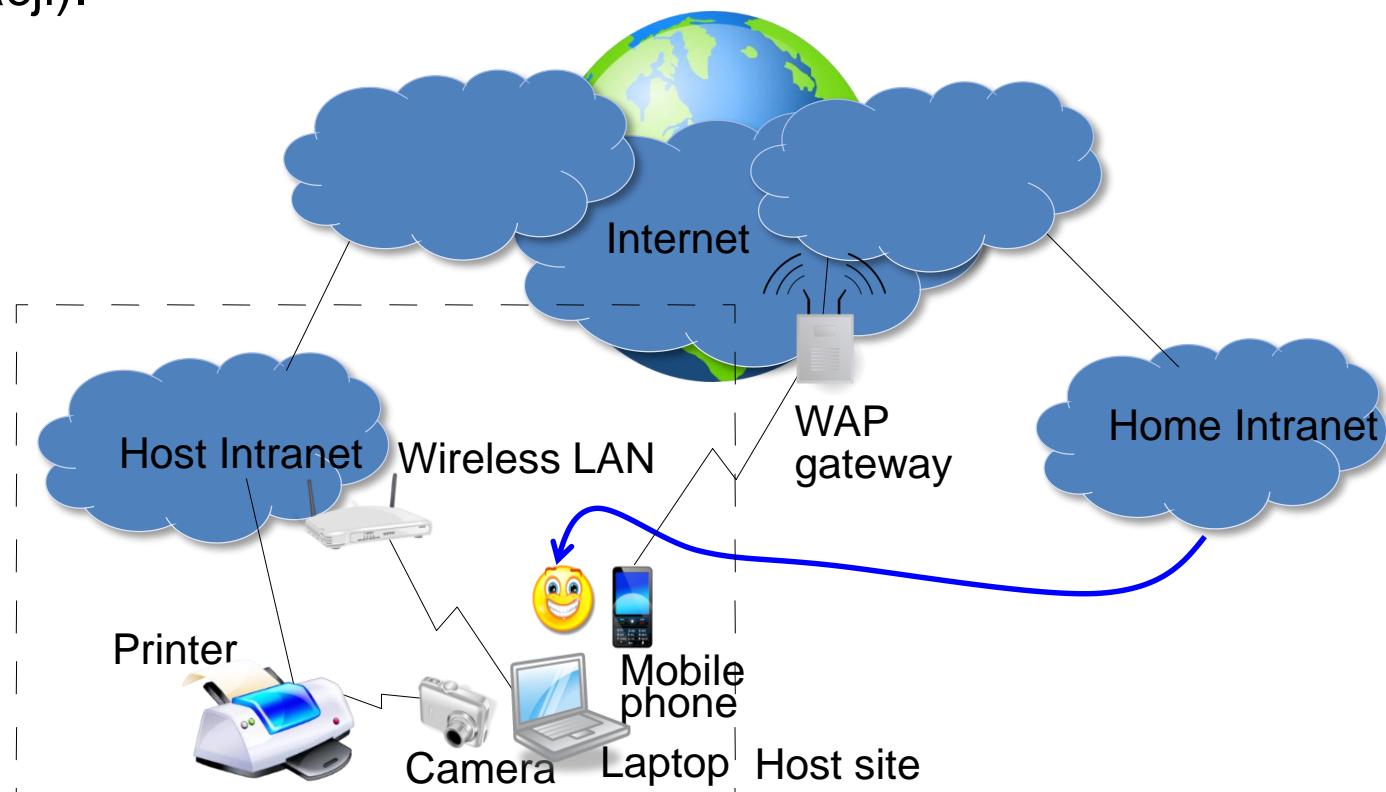
# Interanet

- Część Internetu, która jest oddzielnie administrowana i obsługuje wewnętrzne współdzielenie zasobów (systemy plików/pamięci masowej i drukarki)



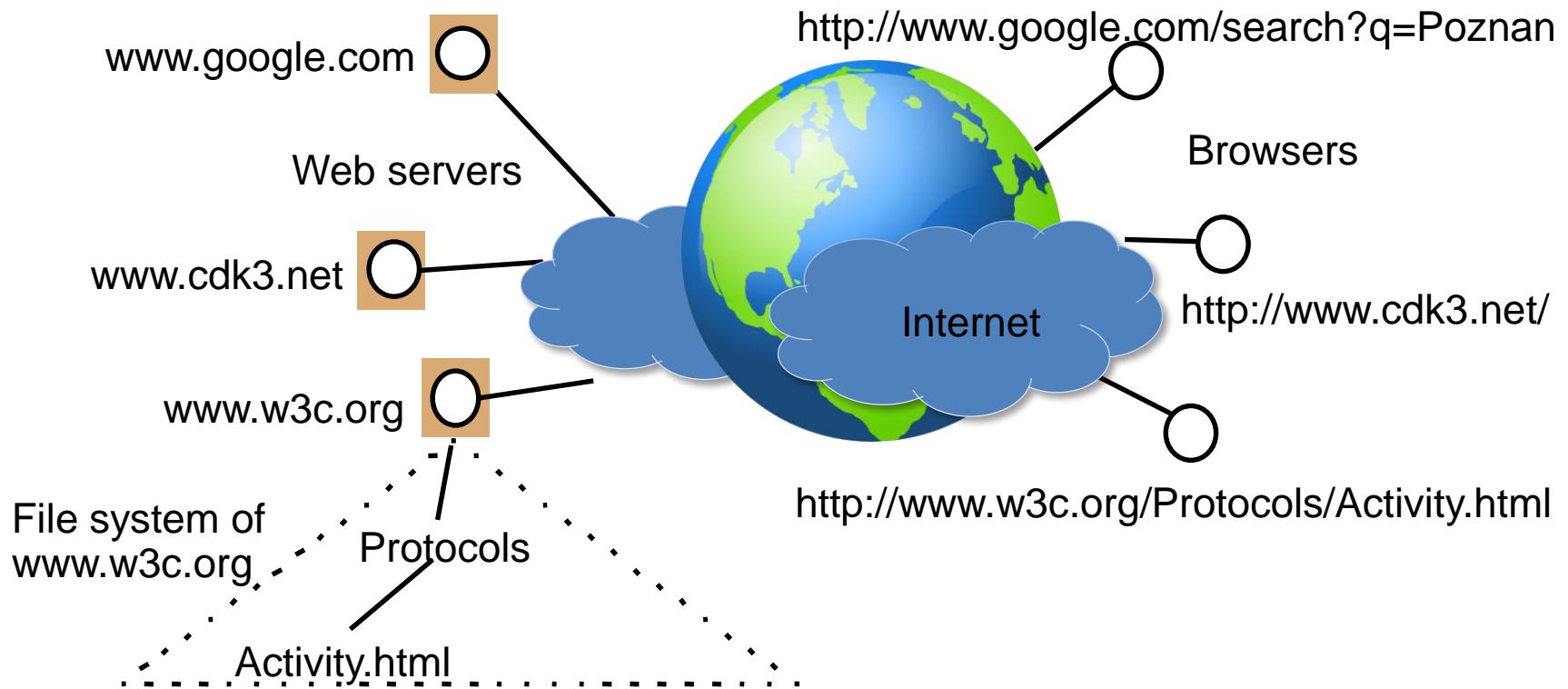
# Komputery mobilne i wszechobecne

- Urządzenia przenośne i podręczne w systemie rozproszonym
- Obsługuje stały dostęp do domowych zasobów intranetowych za pośrednictwem sieci bezprzewodowej i umożliwia korzystanie z zasobów (np. drukarek), które są dogodnie zlokalizowane (przetwarzanie świadome lokalizacji).

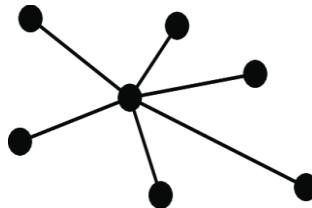


# Współdzielenie zasobów i sieć

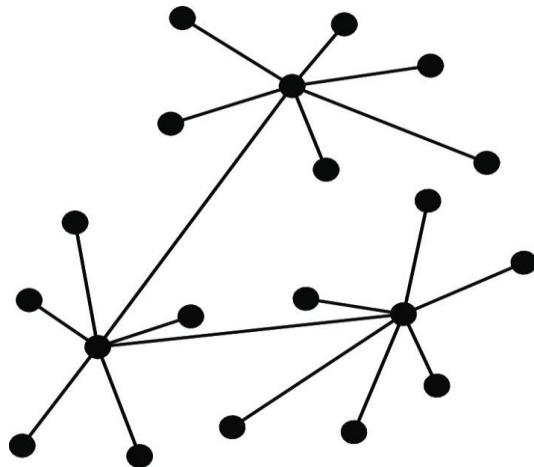
- Otwarte protokoły, skalowalne serwery i przeglądarki z wtyczkami



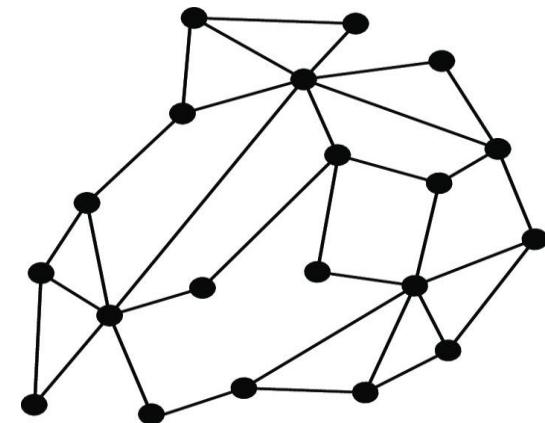
# Rozproszone vs. zdecentralizowane



Scentralizowane  
(ang. centralized)

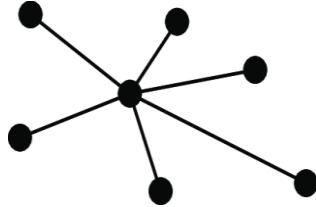


Zdecentralizowane  
(ang. decentralized)

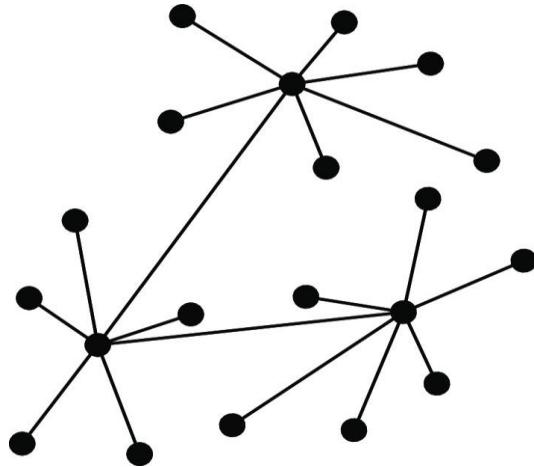


Rozproszone  
(ang. distributed)

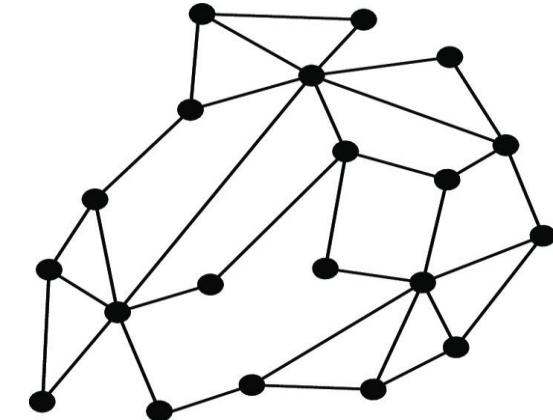
# Rozproszone vs. zdecentralizowane



Scentralizowane  
(ang. centralized)



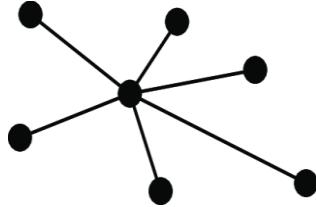
Zdecentralizowane  
(ang. decentralized)



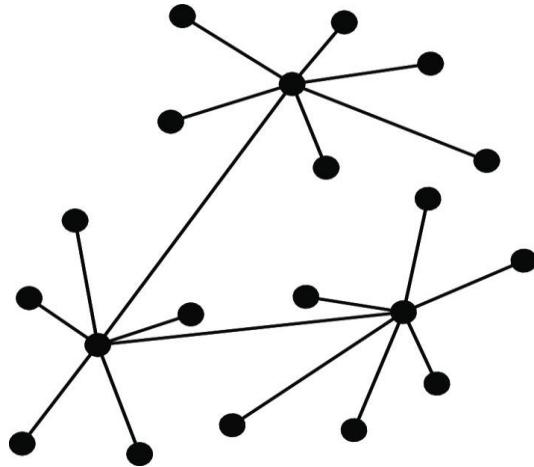
Rozproszone  
(ang. distributed)

Kiedy zdecentralizowany system staje się rozproszony?

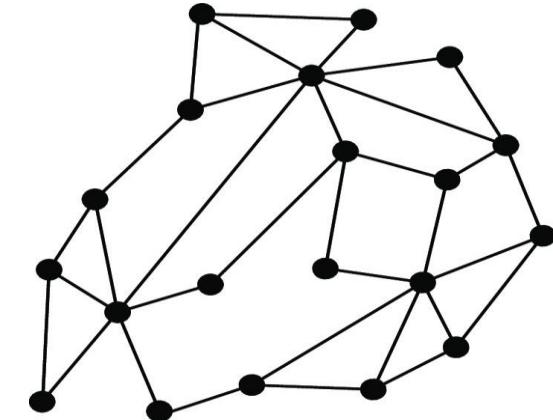
# Rozproszone vs. zdecentralizowane



Scentralizowane  
(ang. centralized)



Zdecentralizowane  
(ang. decentralized)

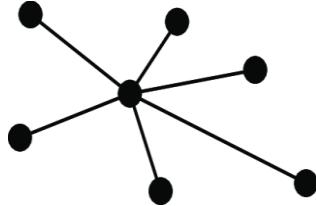


Rozproszone  
(ang. distributed)

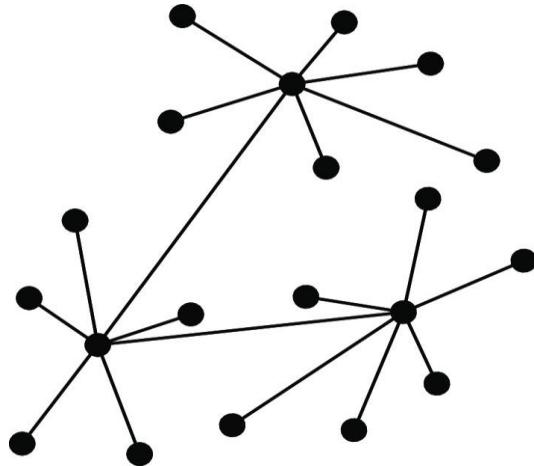
Kiedy zdecentralizowany system staje się rozproszony?

- Dodanie 1 łącza między dwoma węzłami w systemie zdecentralizowanym?

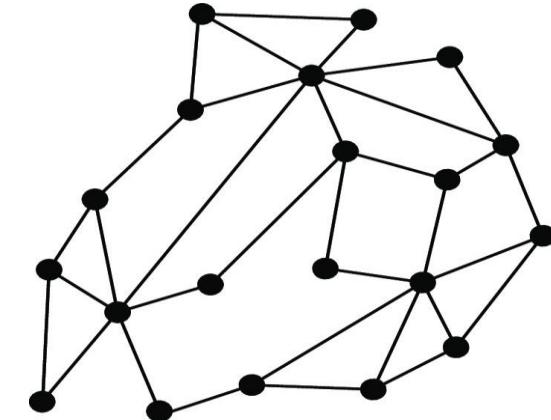
# Rozproszone vs. zdecentralizowane



Scentralizowane  
(ang. centralized)



Zdecentralizowane  
(ang. decentralized)

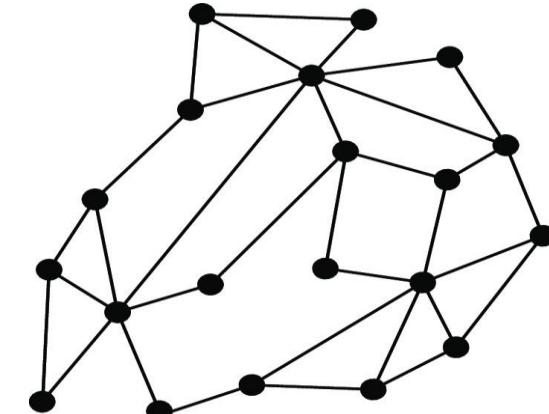
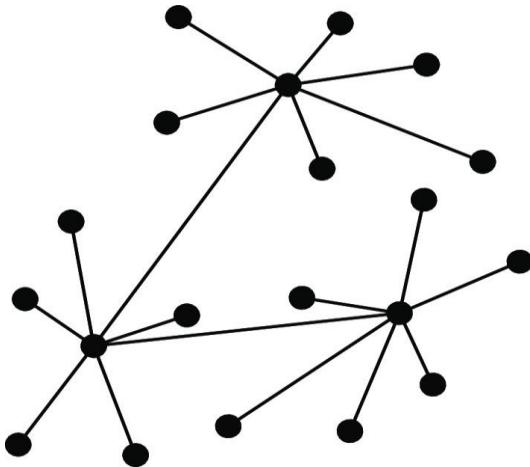
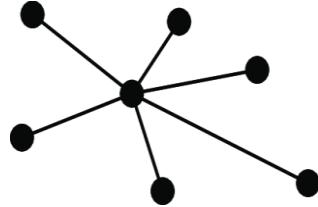


Rozproszone  
(ang. distributed)

Kiedy zdecentralizowany system staje się rozproszony?

- Dodanie 1 łącza między dwoma węzłami w systemie zdecentralizowanym?
- Dodanie 2 linków między dwoma innymi węzłami?

# Rozproszone vs. zdecentralizowane



Scentralizowane  
(ang. centralized)

Zdecentralizowane  
(ang. decentralized)

Rozproszone  
(ang. distributed)

Kiedy zdecentralizowany system staje się rozproszony?

- Dodanie 1 łącza między dwoma węzłami w systemie zdecentralizowanym?
- Dodanie 2 linków między dwoma innymi węzłami?
- Ogólnie: dodanie  $k > 0$  linków....?

# Rozproszone vs. zdecentralizowane

## Systemy zdecentralizowane:

- decentralizacja odnosi się do poziomów kontroli.
- nie ma scentralizowanego komponentu, który ma kontrolę nad całym systemem.
- przykład: Bitcoin

## Systemy rozproszone:

- rozproszenie odnosi się do różnic w lokalizacji - składa się z komponentów, które są fizycznie oddzielone.
- komunikują się one ze sobą za pośrednictwem sieci.
- pozwala to na współdzielenie zasobów i rozkładanie obciążen na wiele maszyn, często poprawiając wydajność i wydajność.
- system rozproszony może jednak nadal mieć scentralizowany komponent, który koordynuje i kontroluje różne komponenty.

# Rozproszone vs. zdecentralizowane

Dlaczego ta różnica ma znaczenie?

- wpływa ona na sposób, w jaki projektujemy, tworzymy i utrzymujemy systemy oprogramowania, które działają w złożonych i dynamicznych środowiskach.
- w zależności od wymagań i ograniczeń domeny problemu, możemy dokonać właściwych wyborów projektowych.

# Cechy systemów rozproszonych

- duża wydajność
- duża efektywność inwestowania
- wysoka sprawność wykorzystania zasobów
- skalowalność
- wysoka niezawodność
- otwartość funkcjonalna

# Systemy rozproszone – własności

- Autonomiczne, heterogeniczne komponenty pracujące niezależnie
- Komunikacja poprzez wymianę komunikatów
  - Brak pamięci współdzielonej
- Współdzielenie zasobów
  - Drukarka, baza danych, inne usługi
  - Brak współdzielonej przestrzeni adresowej
- Brak globalnego stanu
  - Żaden pojedynczy proces nie może posiadać wiedzy o bieżącym globalnym stanie systemu.
- Brak globalnego zegara
  - Procesy mogą synchronizować swoje zegary z ograniczoną precyzją
- Niezależne awarie (komputerów i łączy komunikacyjnych)

# Przykład i wyzwania I

Co jeśli

- Twój klient używa zupełnie innego sprzętu? (PC, MAC,...)...
- innego systemu operacyjnego? (Windows, Unix,...)...
- inny sposób reprezentacji danych? (ASCII, EBCDIC,...)
- **Heterogeniczność**

Lub

- Chcesz przenieść swoją firmę i komputery na Karaiby (z powodu pogody)?
- **Transparentność dystrybucji danych**

# Przykład i wyzwania II

Co jeśli

- Dwóch klientów chce zamówić ten sam produkt w tym samym czasie?
- **Współbieżność**

Lub

- Wystąpi awaria bazy danych z informacjami o zasobach?
- Komputer klienta ulega awarii w trakcie realizacji zamówienia?
- **Tolerancja awarii**

# Przykład i wyzwania III

Co jeśli

- Ktoś próbuje włamać się do systemu w celu kradzieży danych?
- ... wyszukuje informacje?
- ... klient zamawia coś i nie przyjmuje dostawy, twierdząc, że tego nie zrobił?
- **Bezpieczeństwo**

Lub

- Jesteś tak skuteczny, że miliony ludzi odwiedzają Twój sklep internetowy w tym samym czasie?
- **Skalowalność**

# Przykład i wyzwania IV

Podczas projektowania i tworzenia systemu...

- Czy chcesz napisać całe oprogramowanie samodzielnie (sieć, baza danych,...)?
- Co z aktualizacjami, nowymi technologiami?
- **Ponowne wykorzystanie i otwartość (standardy)**

# Przegląd wyzwań I

## Heterogeniczność

- Heterogeniczne komponenty muszą być w stanie współdziałać.

## Transparentność dystrybucji

- Dystrybucja powinna być w jak największym stopniu ukryta przed użytkownikiem.

## Odporność na błędy

- Awaria komponentu (częściowa awaria) nie powinna skutkować awarią całego systemu.

## Skalowalność

- System powinien działać wydajnie przy rosnącej liczbie użytkowników.
- Wydajność systemu powinna wzrastać wraz z włączaniem dodatkowych zasobów

# Przegląd wyzwań II

## Współbieżność

- Współzielony dostęp do zasobów musi być możliwy

## Otwartość

- Interfejsy powinny być publicznie dostępne, aby ułatwić dodawanie nowych komponentów.

## Bezpieczeństwo

- System powinien być używany tylko w zamierzony sposób

# Co chcemy osiągnąć?

Ogólne cele projektowe:

- Wsparcie dla współdzielenia zasobów
- Współdziałanie heterogenicznych komponentów
- Transparentność rozproszenia (ukrywanie faktu rozproszenia zasobów)
- Otwartość
- Skalowalność (łatwość rozszerzania)
- Zwiększoa dostępność

# Heterogeniczność

Heterogeniczne komponenty muszą być zdolne do współdziałania

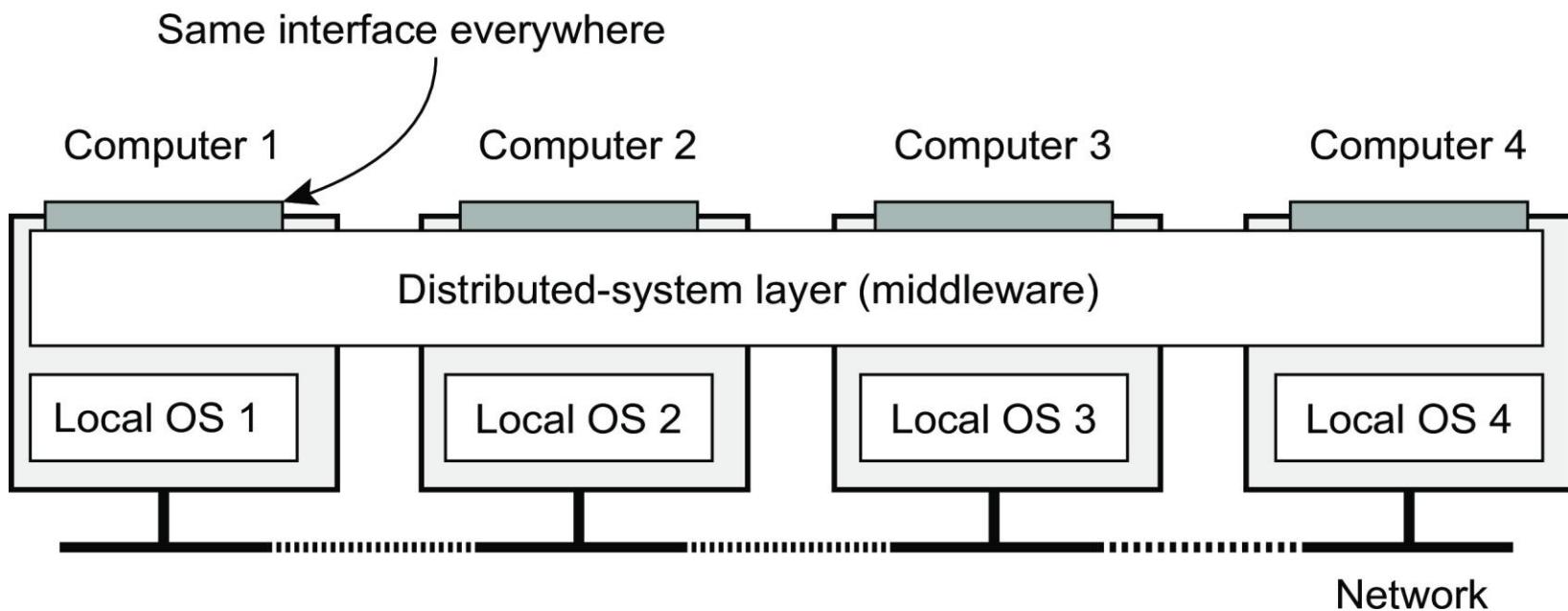
- Systemy operacyjne
- Architektury sprzętowe
- Architektury komunikacyjne
- Języki programowania
- Interfejsy oprogramowania
- Mechanizmy bezpieczeństwa
- Reprezentacja informacji

# Przezroczystość

## Czym jest przezroczystość?

Zjawisko, za pomocą którego system rozproszony próbuje ukryć fakt, że jego procesy i zasoby są fizycznie rozproszone na wielu komputerach, prawdopodobnie oddzielonych od siebie dużymi odległościami.

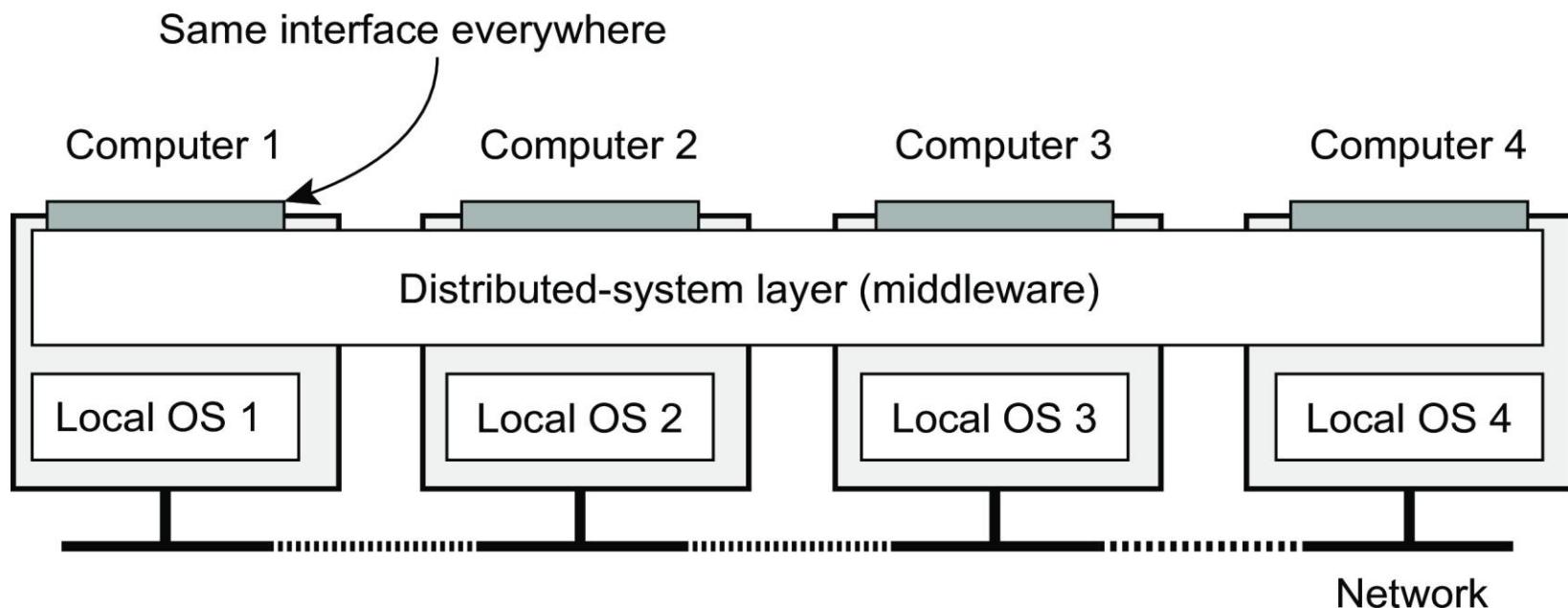
**System przezroczysty** — (transparentny, ang. *transparent*) sprawia wrażenie systemu skonsolidowanego (dla użytkowników i aplikacji).



# Przezroczystość

## Obserwacja

Transparentność dystrybucji jest obsługiwana za pomocą wielu różnych technik w warstwie między aplikacjami a systemami operacyjnymi: warstwa oprogramowania pośredniczącego.



# Przezroczystość

- Przezroczystość dostępu (ang. access transparency) – ujednolicanie metod dostępu do danych i ukrywanie różnic w reprezentacji danych.
- Przezroczystość położenia (ang. location transparency) – użytkownicy nie mogą określić fizycznego położenia zasobu (np. na podstawie jego nazwy czy identyfikatora).
- Przezroczystość wędrówki (ang. migration transparency) – można przenosić zasoby pomiędzy serwerami bez zmiany sposoby odwoywania się do nich.
- Przezroczystość przemieszczania (ang. relocation transparency) – zasoby mogą być przenoszone nawet podczas ich używania

# Przezroczystość

- Przezroczystość z wielokrotniania (ang. replication transparency) ukrywanie przed użytkownikami faktu z wielokrotniania (replikacji) zasobów.
- Przezroczystość współbieżności (ang. concurrency transparency) możliwość współbieżnego przetwarzania danych nie powodująca powstawania niespójności w systemie.
- Przezroczystość awarii (ang. failure transparent) maskowanie przejściowych awarii poszczególnych komponentów systemu rozproszonego.
- Przezroczystość trwałości (ang. persistence transparency) maskowanie sposobu przechowywania zasobu (pamięć ulotna, dysk).

Kompromis pomiędzy dużym stopniem przezroczystości a efektywnością systemu.

# Stopień przezroczystości

- Koszt zapewnienia pełnej przejrzystości rozproszenia może być zbyt wygórowany
- Istnieją opóźnienia w komunikacji, których nie da się ukryć.
- Całkowite ukrycie awarii sieci i węzłów jest (teoretycznie i praktycznie) niemożliwe
  - Nie można odróżnić powolnego komputera od tego, który uległ awarii.
  - Nigdy nie można mieć pewności, że serwer faktycznie wykonał operację przed awarią.
- Pełna przezroczystość będzie kosztować wydajność, ujawniając rozproszenie systemu
  - Utrzymywanie spójnych kopii stanu komponentów jest czasochłonne
  - Natychmiastowe zapisywanie operacji zapisu na dysk w celu zapewnienia odporności na błędy

# Stopień przezroczystości

Ujawnianie rozproszenia może być dobre

- Korzystanie z usług opartych na lokalizacji (znajdowanie znajomych w pobliżu)
- Gdy mamy do czynienia z użytkownikami w różnych strefach czasowych
- Gdy ułatwia to użytkownikowi zrozumienie, co się dzieje (gdy np. serwer nie odpowiada przez długi czas, zgłoś to jako awarię).

Wniosek

Przejrzystość rozproszenia to pożądany cel, ale jego osiągnięcie jest trudne i często nie należy nawet do niego dążyć.

# Otwartość

## Otwarty system rozproszony

- System oferujący komponenty, które mogą być łatwo używane przez inne systemy lub z nimi integrowane.
- Otwarty system rozproszony często składa się z komponentów pochodzących z innych źródeł.

Możliwość interakcji z usługami z innych otwartych systemów, niezależnie od środowiska bazowego:

- Systemy powinny być zgodne z dobrze zdefiniowanymi interfejsami
- Systemy powinny łatwo współpracować
- Systemy powinny wspierać przenośność aplikacji
- Systemy powinny być łatwo rozszerzalne

# Zasady vs mechanizmy

## Wdrażanie otwartości: zasady

- Jakiego poziomu spójności wymagamy dla danych buforowanych przez klienta?
- Na jakie operacje zezwalamy pobieranemu kodowi?
- Jakie wymagania QoS dostosowujemy w obliczu zmiennej przepustowości?
- Jakiego poziomu tajności wymagamy dla komunikacji?

## Wdrażanie otwartości: mechanizmy

- Umożliwienie (dynamicznego) ustawiania zasad buforowania (ang. caching)
- Obsługa różnych poziomów zaufania dla kodu mobilnego
- Zapewniają modyfikowanych parametrów QoS dla każdego strumienia danych
- Oferowanie różnych algorytmów szyfrowania

# Ścisła separacja

## Obserwacja

- Im ścisłej separacja między polityką a mechanizmem, tym bardziej musimy zapewnić odpowiednie mechanizmy, co potencjalnie prowadzi do wielu parametrów konfiguracyjnych i złożonego zarządzania.

## Znalezienie równowagi

- Twarde kodowanie polityk często upraszcza zarządzanie i zmniejsza złożoność za cenę mniejszej elastyczności.

Nie ma oczywistego rozwiązania.

# Systemy elastyczne

## Systemy elastyczne

- łatwość konfiguracji
- łatwość rekonfiguracji (np. wymiana poszczególnych komponentów)

## Organizacja systemu rozproszonego

- projekt monolityczny
- logiczne wydzielenie składowych
- podział na autonomiczne komponenty (koszt: wydajność)
- oddzielenie polityki od mechanizmu

# Niezawodność (ang. dependability)

## Podstawy

- Komponent dostarcza usługi klientom.
- Aby świadczyć usługi, komponent może wymagać usług od innych komponentów ⇒ komponent może zależeć od innego komponentu.

W szczególności komponent C (proces lub kanał komunikacyjny) zależy od C\*, jeśli poprawność zachowania C zależy od poprawności zachowania C\*.

# Niezawodność (ang. dependability)

Wymagania związane z zależnością:

Wymaganie	Opis
Dostępność (ang. Availability)	Gotowość do użycia
Niezawodność (ang. Reliability)	Ciągłość dostarczania usług
Bezpieczeństwo (ang. Safety)	Bardzo niskie prawdopodobieństwo awarii
Utrzymywalność (ang. Maintainability)	Prostota naprawy systemu, który uległ awarii

# Niezawodność vs Dostępność

## Niezawodność $R(t)$ komponentu C

Warunkowe prawdopodobieństwo, że komponent C działał poprawnie w czasie  $[0, t]$ , biorąc pod uwagę, że C działał poprawnie w czasie  $T = 0$ .

Tradycyjne wskaźniki:

- **Średni czas do awarii (MTTF):** Średni czas do awarii komponentu.
- **Średni czas naprawy (MTTR):** Średni czas potrzebny do naprawy komponentu.
- **Średni czas między awariami (MTBF):** MTTF + MTTR.

# Awaria, błąd, usterka

Pojęcie	Opis	Przykład
Awaria	Komponent nie jest zgodny ze swoją specyfikacją	Wadliwy program
Błąd	Część komponentu, która może prowadzić do awarii	Błąd programowy
Usterka	Przyczyna błędu	Niestaranny programista

# Obsługa błędów

Pojęcie	Opis	Przykład
Zapobieganie błędom (ang. fault prevention)	Zapobieganie wystąpieniu błędu	Pozbądź się niestarannych programistów 😊
Tolerancja błędów (ang. fault tolerance)	Budowa komponentu, który będzie maskował wystąpienie błędu	Zbudowanie każdego komponentu przez dwóch niezależnych programistów
Usuwanie błędów (ang. fault removal)	Zmniejszenie obecności, liczby lub powagi błędów	Pozbądź się niestarannych programistów 😊
Przewidywanie błędów (ang. fault forecasting)	Oszacowanie bieżącej obecności, przyszłego występowania i konsekwencji błędów	Oszacuj, jak rekruter radzi sobie z zatrudnianiem niestarannych programistów

# Mechanizmy tolerowania błędów

- Wykrywanie błędów
  - Sumy kontrolne, heartbeat, ...
- Maskowanie błędów
  - Retransmisja uszkodzonych wiadomości,
  - Redundancja, ...
- Tolerowanie błędów
  - Obsługa wyjątków, timeouty, ...
- Odtwarzanie stanu
  - Mechanizmy wycofywania, ...

# Bezpieczeństwo

Obserwacja: system rozproszony, który nie jest bezpieczny, nie jest niezawodny.

Czego potrzebujemy?

- Poufność (ang. Confidentiality): informacje są ujawniane tylko upoważnionym stronom.
- Integralność (ang. Integrity) : Zapewnienie, że zmiany w zasobach systemu mogą być dokonywane tylko w autoryzowany sposób.

Autoryzacja, uwierzytelnianie, zaufanie

- Uwierzytelnianie (ang. Authentication): weryfikacja poprawności deklarowanej tożsamości
- Autoryzacja (ang. Authorization): czy zidentyfikowany podmiot ma odpowiednie prawa dostępu?
- Zaufanie (ang. Trust): jeden podmiot może mieć pewność, że inny wykona określone działania zgodnie z określonymi oczekiwaniami.

# Skalowalność

Co najmniej trzy składniki:

- Liczba użytkowników lub procesów (skalowalność rozmiaru, ang. size scalability)
- Maksymalna odległość między węzłami (skalowalność geograficzna, ang. geographical scalability)
- Liczba domen administracyjnych (skalowalność administracyjna, ang. administrative scalability)

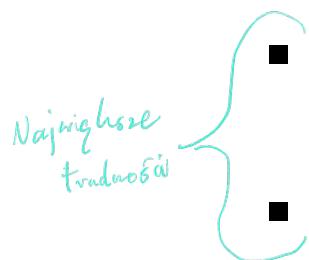
Obserwacja:

- większość systemów uwzględnia skalowalność rozmiaru tylko do pewnego stopnia.
- Często stosowanym rozwiązaniem jest wiele potężnych serwerów działających niezależnie i równolegle.
- Obecnie wyzwaniem nadal jest skalowalność geograficzna i administracyjna.

# Skalowalność

Przyczyny problemów ze skalowalnością rozwiązań scentralizowanych:

- Wydajność obliczeniowa ograniczona przez procesory.
- Pojemność pamięci masowej, w tym szybkość transferu między procesorami i dyskami
- Sieć pomiędzy użytkownikiem a scentralizowaną usługą



Algorytmy zdecentralizowane - *trudności*

- brak informacji globalnej
- decyzje na podstawie informacji lokalnych
- odporność na awarie pojedynczych maszyn
- brak założeń dot. istnienia globalnego zegara

*moje proceder do nie sąjego stanu*

# Skalowalność geograficzna – problemy

- Nie można po prostu przejść z sieci LAN do WAN:
  - wiele systemów rozproszonych zakłada synchroniczne interakcje klient-serwer:
  - klient wysyła żądanie i czeka na odpowiedź. Opóźnienia mogą łatwo uniemożliwić ten schemat.
- Łącza WAN są często z natury zawodne:
  - zwykłe przeniesienie strumieniowego wideo z sieci LAN do WAN jest skazane na niepowodzenie.
- Brak komunikacji wielopunktowej, więc nie można wdrożyć prostej transmisji wyszukiwania.
  - Rozwiązaniem jest opracowanie oddzielnych usług nazewniczych i katalogowych (mających własne problemy ze skalowalnością).

przejście z synchronicznego na asynchroniczne

# Skalowalność administracyjna – problemy

- Istota: sprzeczne zasady dotyczące użytkowania (a tym samym płatności), zarządzania i bezpieczeństwa
- Przykłady:
  - Siatki obliczeniowe: współdzielenie drogich zasobów między różnymi domenami.
  - Współdzielony sprzęt: jak kontrolować, zarządzać i używać współdzielonego radioteleskopu zbudowanego jako współdzielona sieć czujników na dużą skalę?
- Wyjątek: kilka sieci peer-to-peer
  - Systemy udostępniania plików (oparte np. na BitTorrent) Telefonia peer-to-peer (wczesne wersje Skype)
  - Strumieniowe przesyłanie dźwięku wspomagane przez peer-to-peer (Spotify)
  - Uwaga: współpracują użytkownicy końcowi, a nie jednostki administracyjne.

# Techniki zapewnienia skalowalności

- Ukrywanie opóźnień w komunikacji:
  - Wykorzystanie komunikacji asynchronicznej
  - Oddzielny program obsługi dla przychodzących odpowiedzi
  - Problem: nie każda aplikacja pasuje do tego modelu
- Partycjonowanie danych i obliczeń na wielu maszynach
  - Przenoszenie obliczeń do klientów (aplety i skrypty Java)
  - Zdecentralizowane usługi nazewnictwa (DNS)
  - Zdecentralizowane systemy informacyjne (WWW)

# Replikacja w zapewnieniu skalowalności

- Replikacja i buforowanie: udostępnianie kopii danych na różnych maszynach
  - Replikowane serwery plików i bazy danych
  - Lustrzane strony internetowe
  - Buforowanie stron internetowych (w przeglądarkach i serwerach proxy)
  - Buforowanie plików (na serwerze i kliencie)

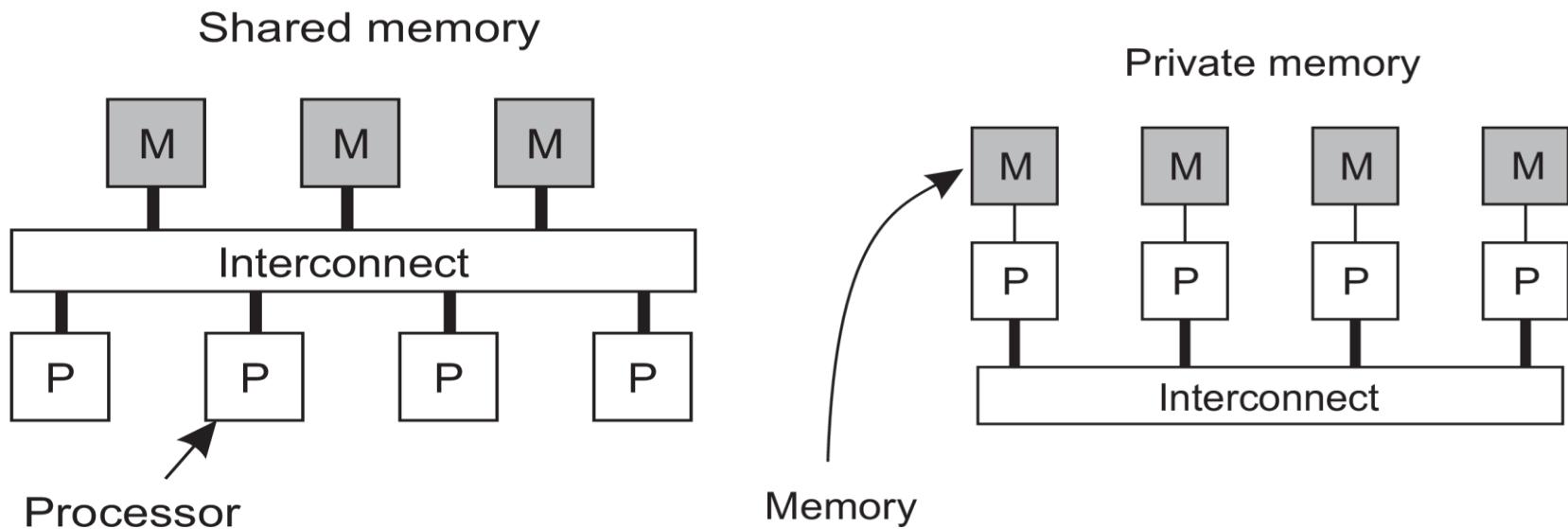
# Replikacja – problem

- Posiadanie wielu kopii (buforowanych lub replikowanych) prowadzi do niespójności: modyfikacja jednej kopii powoduje, że różni się ona od pozostałych.
- Utrzymywanie spójności kopii w sposób ogólny wymaga globalnej synchronizacji przy każdej modyfikacji.
- Globalna synchronizacja wyklucza rozwiązania na dużą skalę.

Obserwacja: jeśli możemy tolerować niespójności, możemy zmniejszyć potrzebę globalnej synchronizacji, ale tolerowanie niespójności zależy od aplikacji.

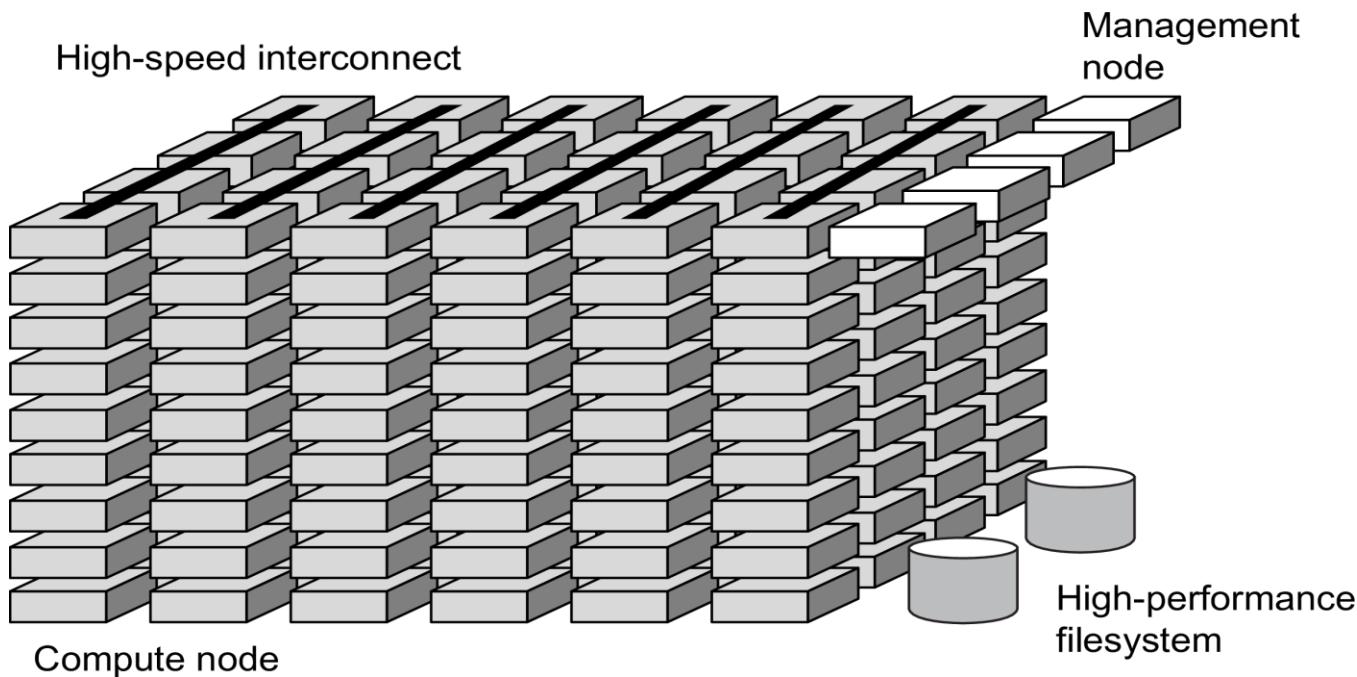
# Replikacja – problem

- Obserwacja: wysokowydajne obliczenia rozproszone rozpoczęły się od obliczeń równoległych
- Wieloprocesorowość i wielordzeniowość kontra wielokomputerowość



# Cluster computing

- Zasadniczo jest to grupa wysokiej klasy systemów połączonych siecią LAN.
- Homogeniczne komponenty: ten sam system operacyjny, niemal identyczny sprzęt
- Pojedynczy lub ścisłe połączony węzeł zarządzający



# Grid computing

Następny krok: wiele rozproszonych węzłów:

- Heterogeniczne
- Rozproszone w kilku organizacjach
- Może łatwo obejmować sieć rozległą

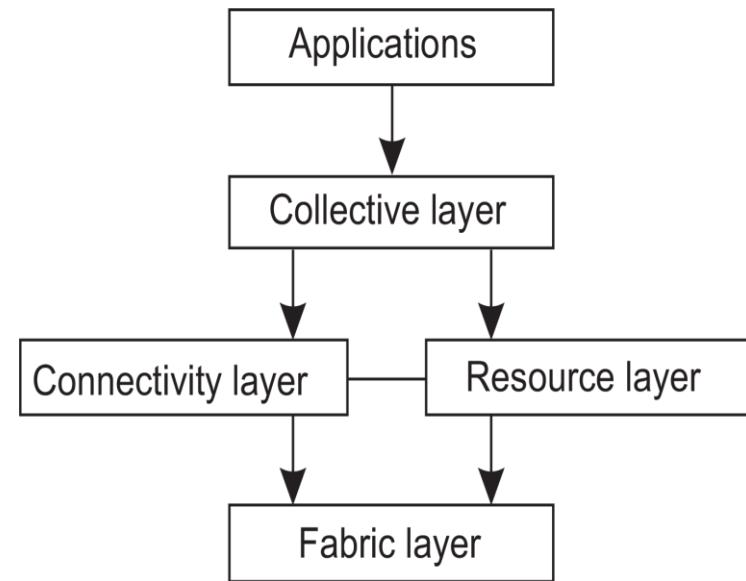
Uwaga:

- aby umożliwić współpracę, siatki zazwyczaj wykorzystują organizacje wirtualne.
- zasadniczo jest to grupowanie użytkowników (lub lepiej: ich identyfikatorów), które pozwala na autoryzację alokacji zasobów.

# Grid computing - architektura

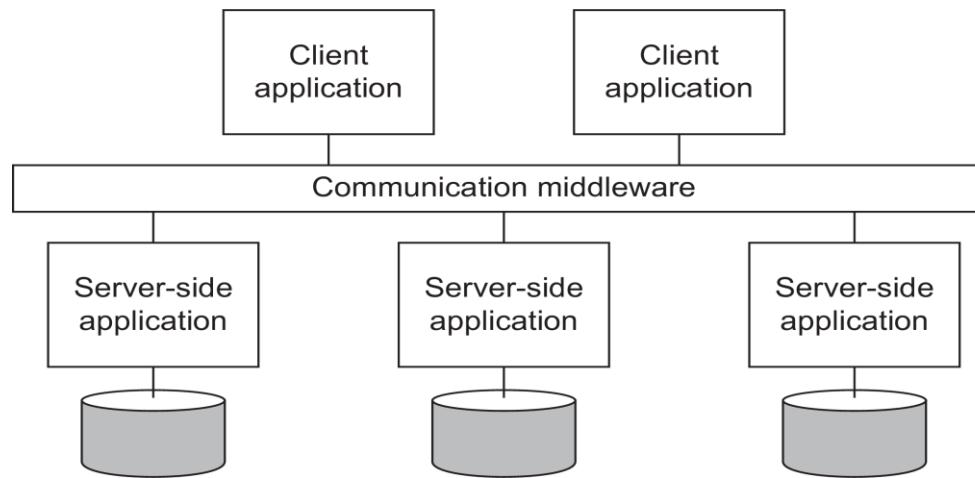
Warstwy:

- **Konstrukcyjna:** zapewnia interfejsy do lokalnych zasobów (do odpytywania stanu i możliwości, blokowania itp.)
- **Łączności:** protokoły komunikacji/transakcji, np. do przenoszenia danych między zasobami oraz protokoły uwierzytelniania.
- **Zasoby:** Zarządza pojedynczym zasobem, np. tworzy procesy lub odczytuje dane.
- **Współdzielona:** Obsługuje dostęp do wielu zasobów: wykrywanie, planowanie, replikacja.
- **Aplikacyjna:** Zawiera rzeczywiste aplikacje gridowe w pojedynczej organizacji.



# Middleware

Oprogramowanie pośredniczące oferuje funkcje komunikacyjne do integracji aplikacji



- Zdalne wywoływanie procedur (**Remote Procedure Call (RPC)**
  - Żądania są wysyłane za pośrednictwem lokalnego wywołania procedury, pakowane jako wiadomość, przetwarzane, wynik jest zwracany jako powrót z wywołania.
- **Message Oriented Middleware (MOM)**
  - Wiadomości są wysyłane do logicznego punktu kontaktowego (published) i przekazywane do subskrybujących aplikacji.

# Integracja aplikacji

- Transfer plików - technicznie proste, ale mało elastyczne:
  - Opracowanie formatu i układu pliku
  - Zarządzanie plikami
  - Propagacja aktualizacji i powiadomienia o aktualizacjach.
- Współdzielona baza danych - znacznie bardziej elastyczna, ale nadal wymaga wspólnego schematu danych obok ryzyka wąskiego gardła.
- Zdalne wywołanie procedury - skuteczne, gdy wymagane jest wykonanie serii działań.
- Przesyłanie wiadomości: RPC wymagają, aby wywołujący i wywołany byli uruchomieni w tym samym czasie. Przesyłanie wiadomości pozwala na rozdzielenie w czasie i przestrzeni.

# Rozproszone systemy wszechobecne (Distributed pervasive systems)

Nowa generacja systemów rozproszonych, w których węzły są małe, mobilne i często osadzone w większym systemie, charakteryzującym się tym, że system naturalnie wtapia się w środowisko użytkownika.

Trzy (nakładające się) podtypy:

- Wszechobecne systemy komputerowe (ang. ubiquitous): wszechobecne i stale obecne, tj. istnieje ciągła interakcja między systemem a użytkownikiem.
- Mobilne systemy komputerowe (ang. mobile): wszechobecne, ale nacisk kładziony jest na fakt, że urządzenia są z natury mobilne.
- Sieci czujników (i siłowników) (ang. sensors and actuators): wszechobecne, z naciskiem na rzeczywiste (wspólne) wykrywanie i uruchamianie środowiska.

# Rozproszone systemy wszechobecne

Podstawowe elementy:

- **Dystrybucja:** Urządzenia są połączone w sieć, rozproszone i dostępne w przejrzysty sposób.
- **Interakcja:** Interakcja między użytkownikami i urządzeniami jest wysoce dyskretna.
- **Świadomość kontekstu:** System jest świadomy kontekstu użytkownika, aby zoptymalizować interakcję.
- **Autonomia:** Urządzenia działają autonomicznie bez interwencji człowieka, a zatem są wysoce samozarządzalne.
- **Inteligencja:** System jako całość może obsługiwać szeroki zakres dynamicznych działań i interakcji.

# Rozproszone systemy mobilne

Cechy wyróżniające:

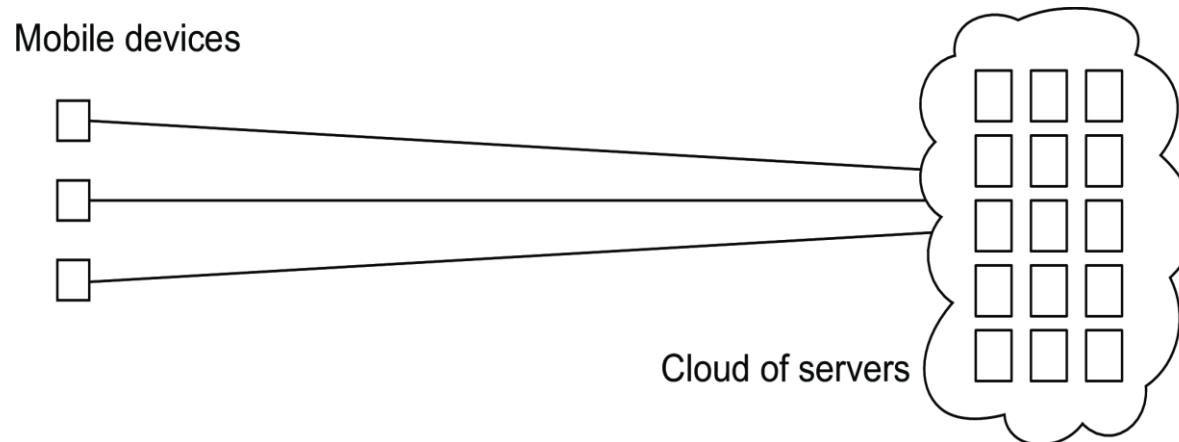
- Mnóstwo różnych urządzeń mobilnych (smartfony, tablety, urządzenia GPS, piloty, aktywne identyfikatory).
- Urządzenia mobilne oznaczają, że lokalizacja urządzenia może się zmieniać w czasie ⇒ zmiana lokalnych usług, możliwości dotarcia itp. Słowo kluczowe: odkrywanie.
  - Utrzymanie stabilnej komunikacji może stwarzać poważne problemy.
- Przez długi czas badania koncentrowały się na bezpośrednim udostępnianiu zasobów między urządzeniami mobilnymi

Podsumowanie:

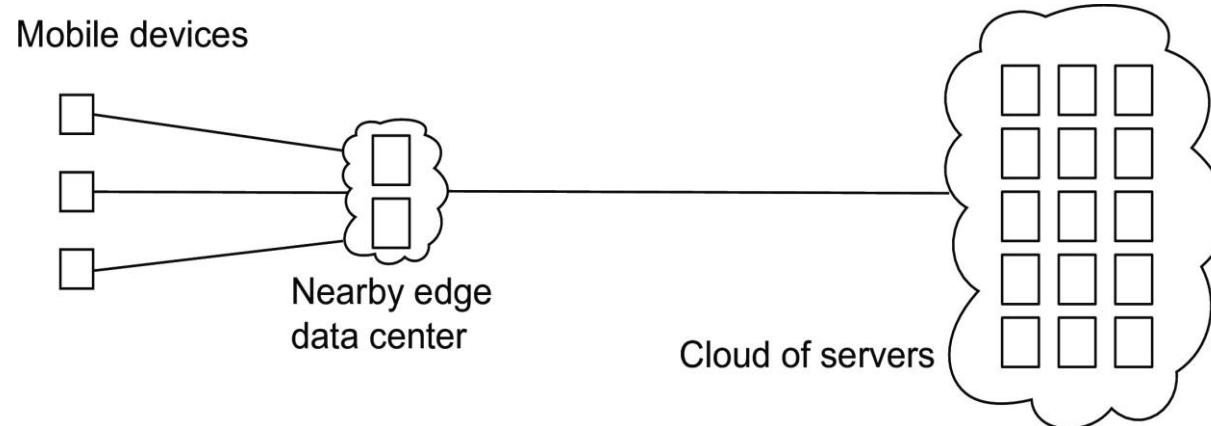
Urządzenia mobilne nawiązują połączenia z serwerami stacjonarnymi, zasadniczo stawiając komputery mobilne w pozycji klientów usług opartych na chmurze.

# Rozproszone systemy mobilne

## Przetwarzanie mobilne w chmurze



## Mobilne przetwarzanie brzegowe

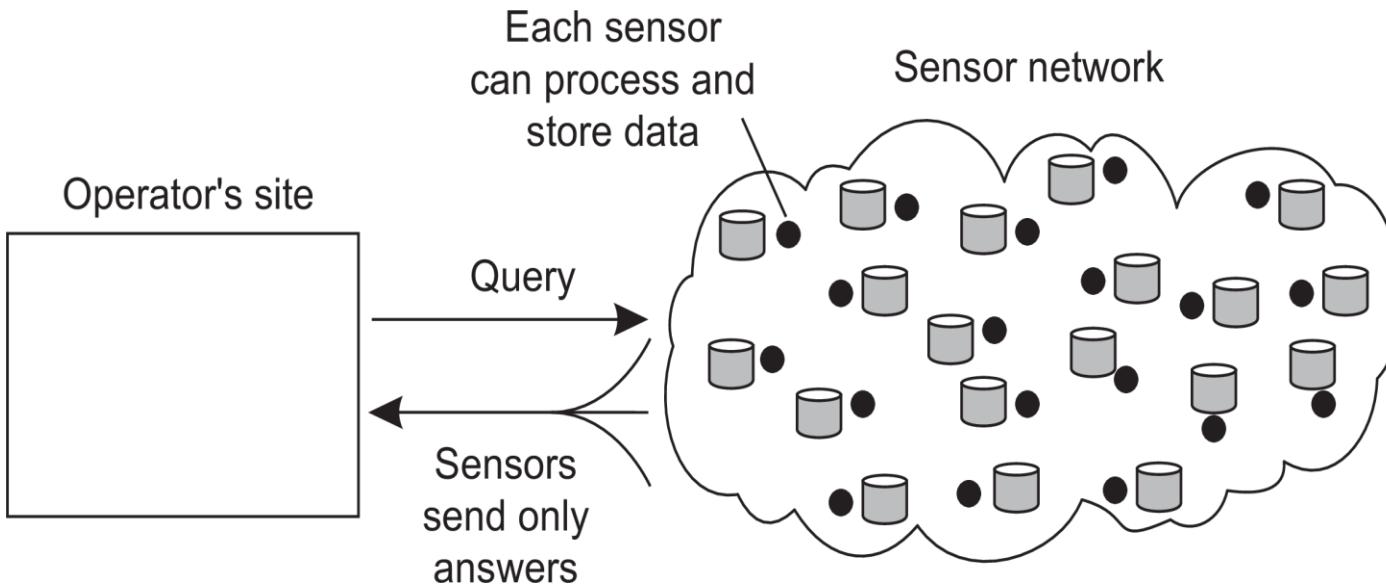
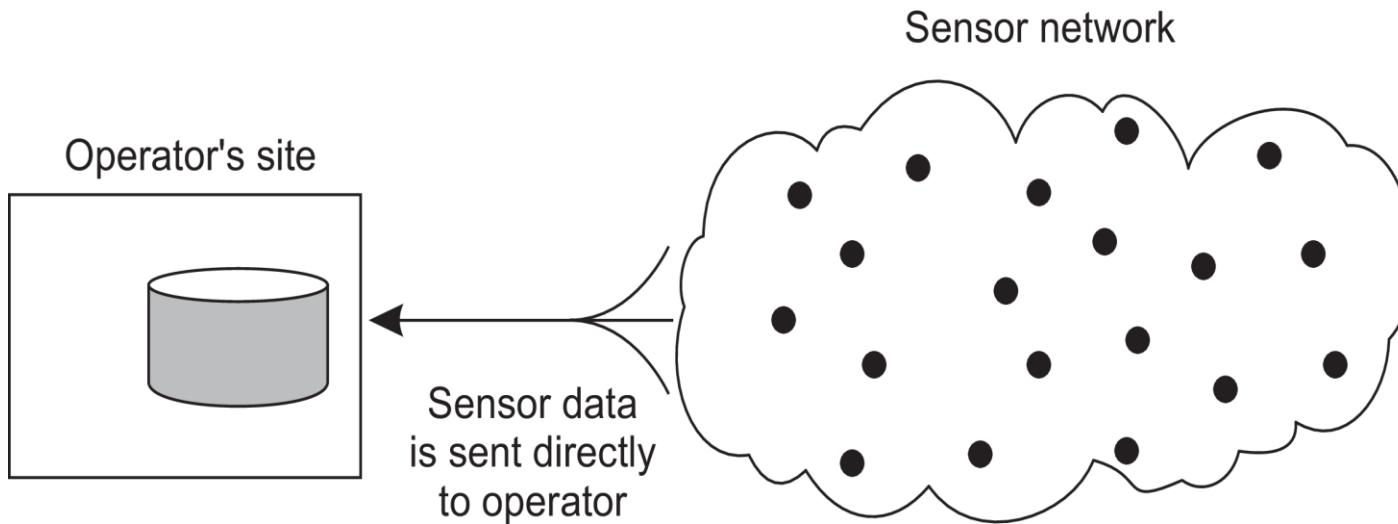


# Sieci sensorów

Charakterystyka – węzły, do których podłączone są czujniki:

- Duża liczba (10s-1000s)
- Proste (mała pamięć/komputer/pojemność komunikacyjna)
- Często zasilane baterijnie (lub nawet bez baterii)

# Sieci sensorów jako rozproszone bazy danych



# Mobilne i wszechobecne systemy rozproszone



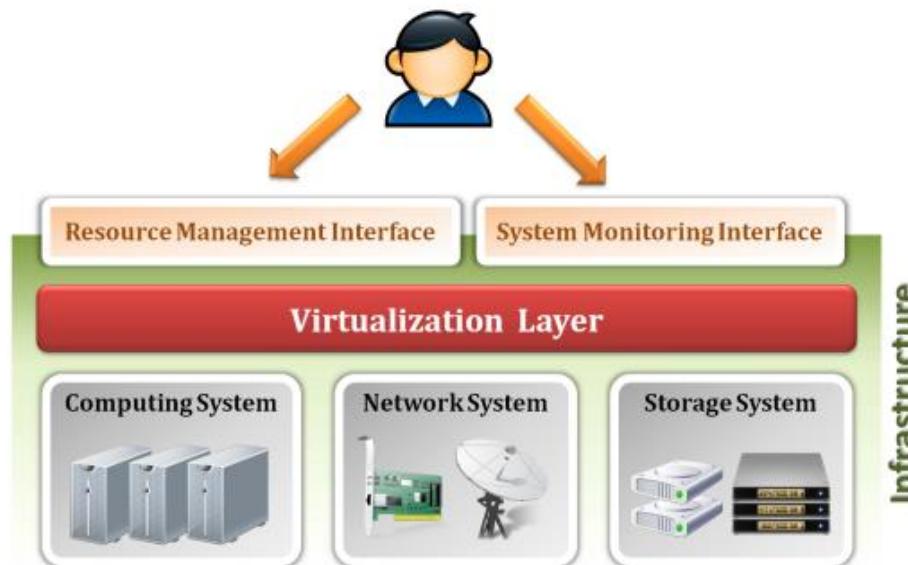
# Przetwarzanie współcześnie

Cisco przewiduje, że do 2023 roku 92% obciążen na rynku IT będzie przetwarzanych przez centra danych w chmurze, podczas gdy tylko 8% będzie przetwarzanych przez tradycyjne centra danych.

# Przetwarzanie w chmurze

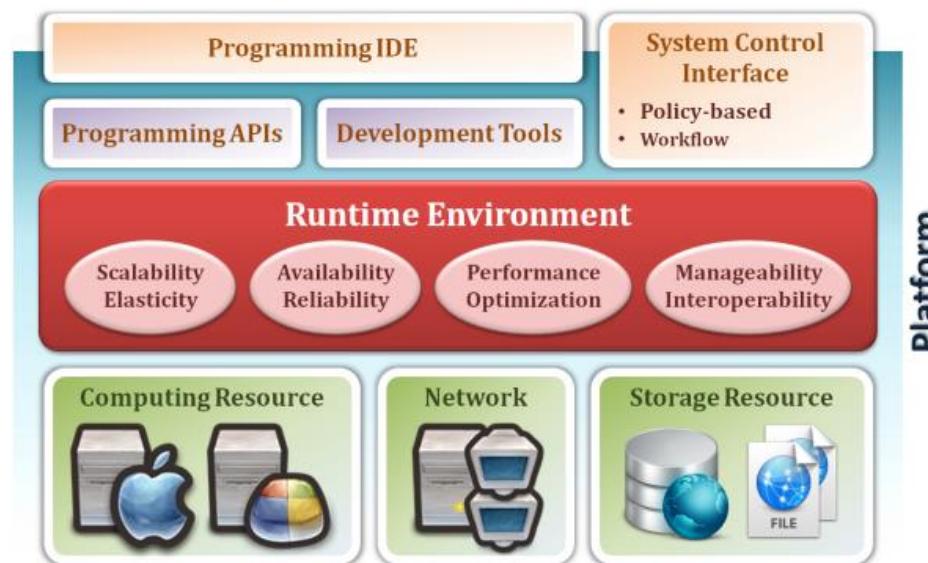
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)
- Function as a Service (FaaS)

- Dostawca zapewnia zasoby, np. przetwarzanie, pamięć masową, sieć, ...
- Klient otrzymuje dostosowane maszyny wirtualne.
- Przykład: Amazon Web Services (instancje EC2 i pamięć masowa S3)



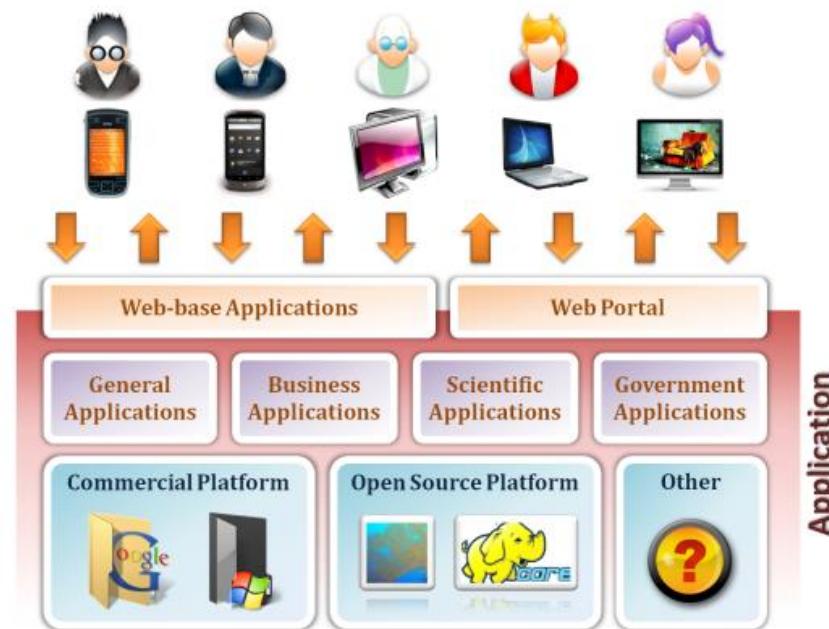
# PaaS

- Dostawca zapewnia sprzęt i środowisko programistyczne.
- Przykład: Silnik aplikacji Google

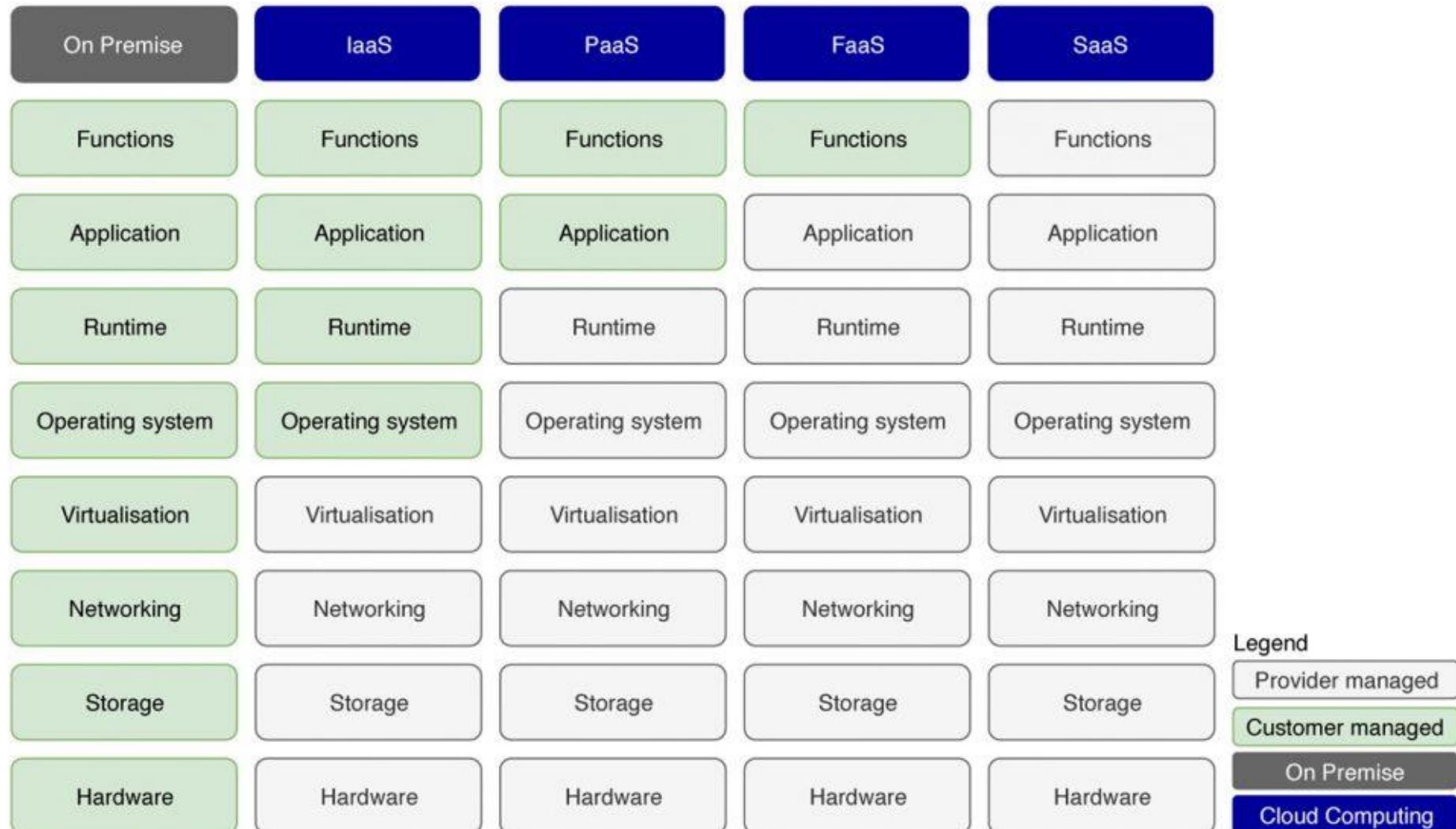


# SaaS

- Dostawca zapewnia aplikacje dostępne przez sieć.
- Przykład: Gmail, Github



# IaaS - PaaS - FaaS - SaaS



# Usługi w chmurze



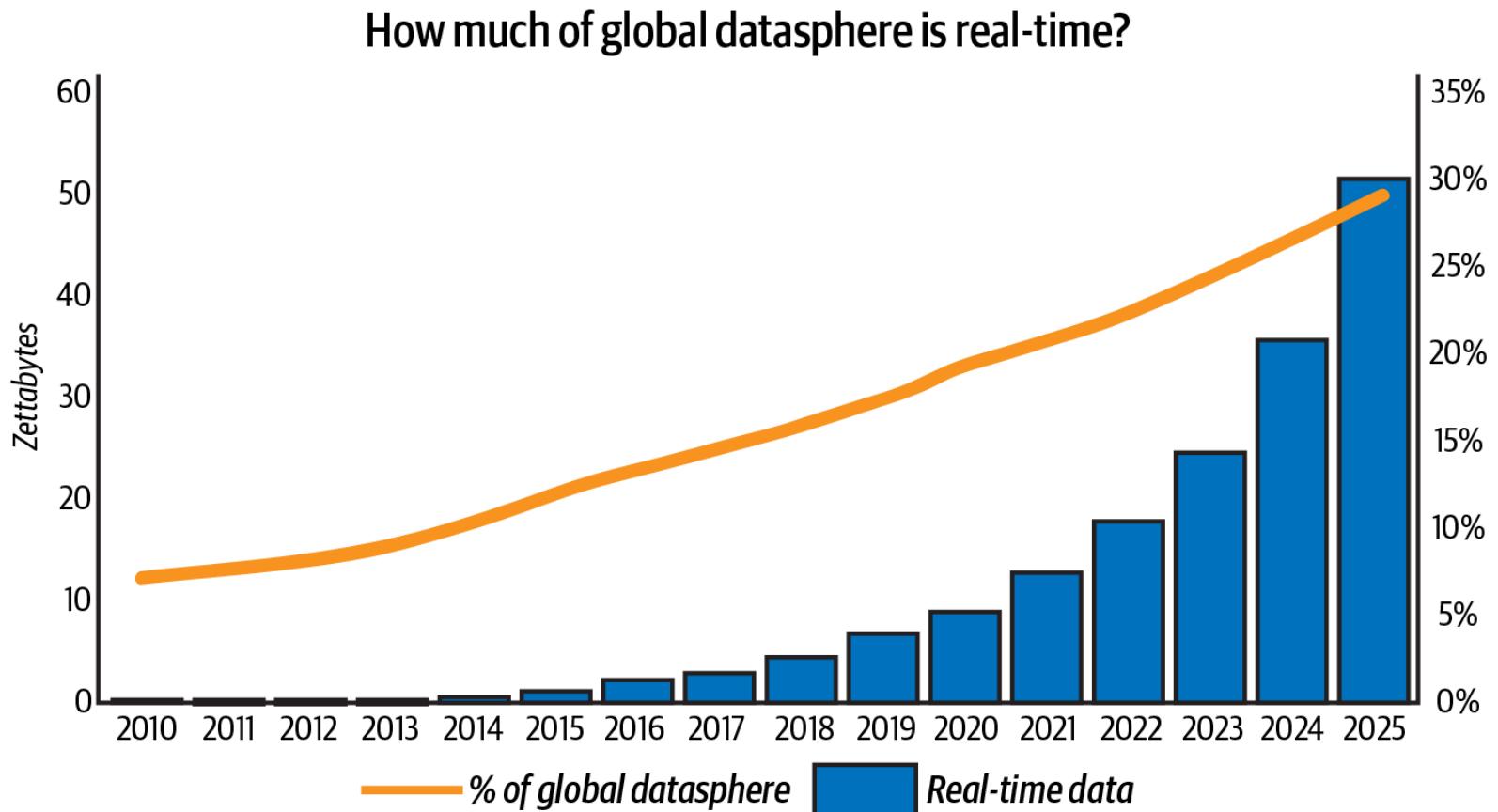
OneDrive



Microsoft 365



# Real-time data



Source: Data Age 2025, sponsored by Seagate with data from IDC Global Datasphere, Nov. 2018

Do 2025 r. 75% danych będzie tworzonych poza centrami danych, gdzie obecnie odbywa się większość przetwarzania

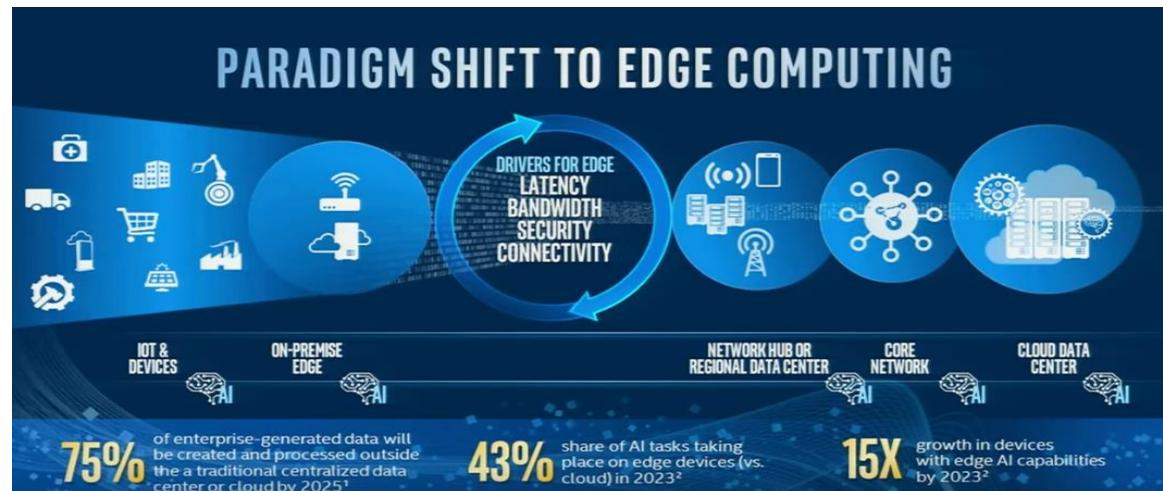
# Systemy czasu rzeczywistego

- Prawidłowe działanie systemu zależy od wykonania w określonym czasie
- Pętle sprzężenia zwrotnego/kontroli
- Sensory
- Twarde systemy czasu rzeczywistego
  - Awaria w przypadku zbyt długiego czasu reakcji.
  - Pamięć zewnętrzna jest ograniczona
- Miękkie systemy czasu rzeczywistego
  - Mniejsza dokładność, jeśli czas odpowiedzi jest zbyt długi.
  - Przydatne w aplikacjach takich jak multimedia, wirtualna rzeczywistość.

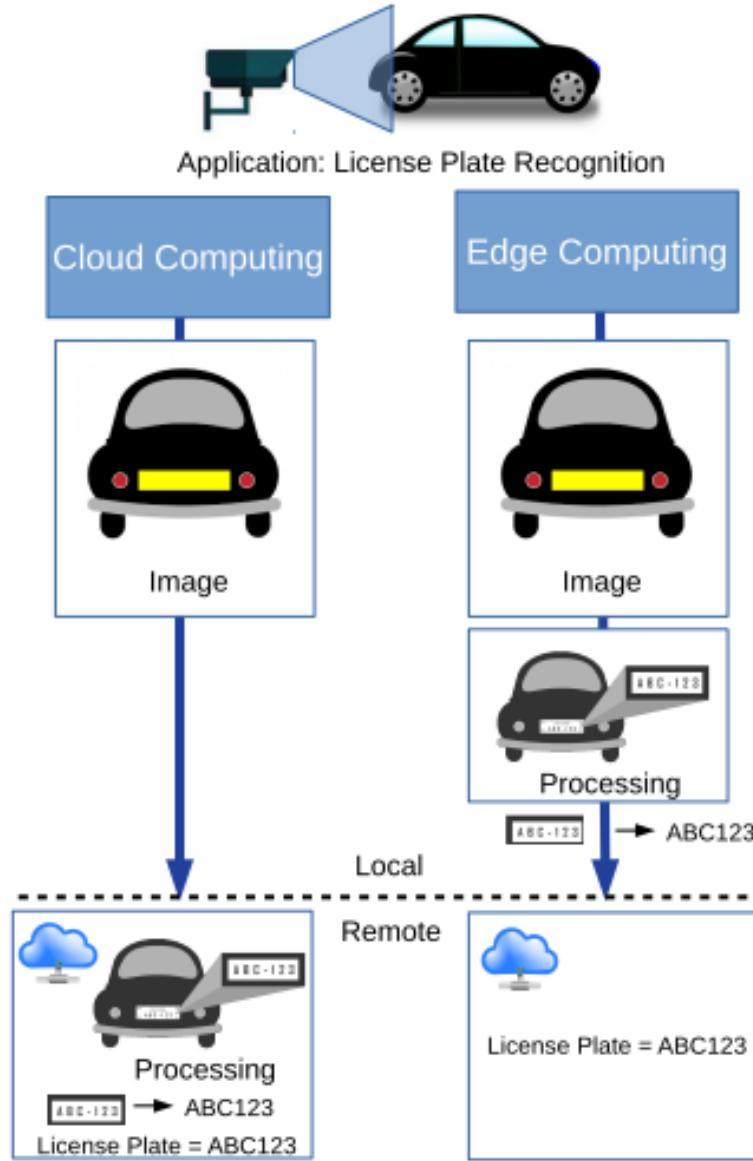


# Edge Computing – motywacja

- Wzrost generowanych (tymczasowych) danych
- Wzrost zapotrzebowania na przetwarzanie danych w czasie rzeczywistym
- Koszty przetwarzania w chmurze
- Prywatność i bezpieczeństwo danych
- Branża półprzewodników – chipy do przetwarzania AI



# Edge vs Cloud Computing



# Edge Computing

Rozproszony paradygmat przetwarzania, który przenosi obliczenia i przechowywanie danych bliżej źródeł danych



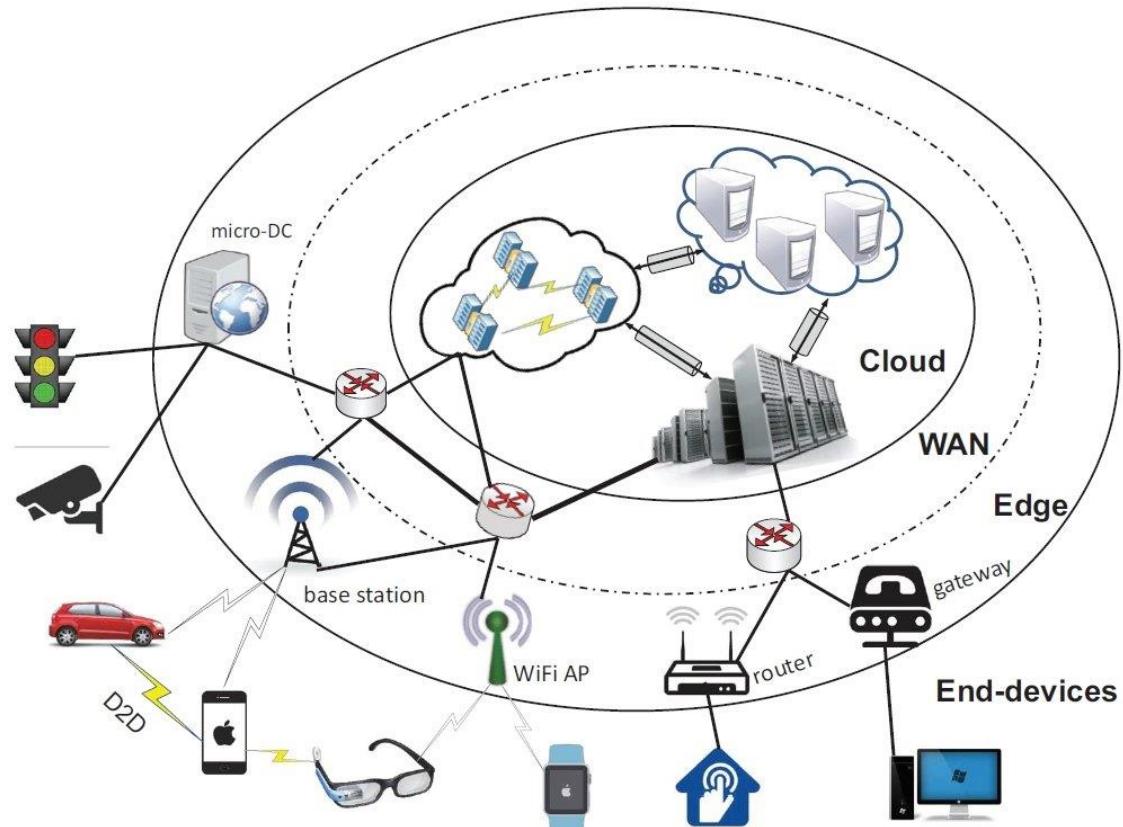
# Edge Computing

Kluczowa architektura przetwarzania informacji w sieci



# Edge Computing

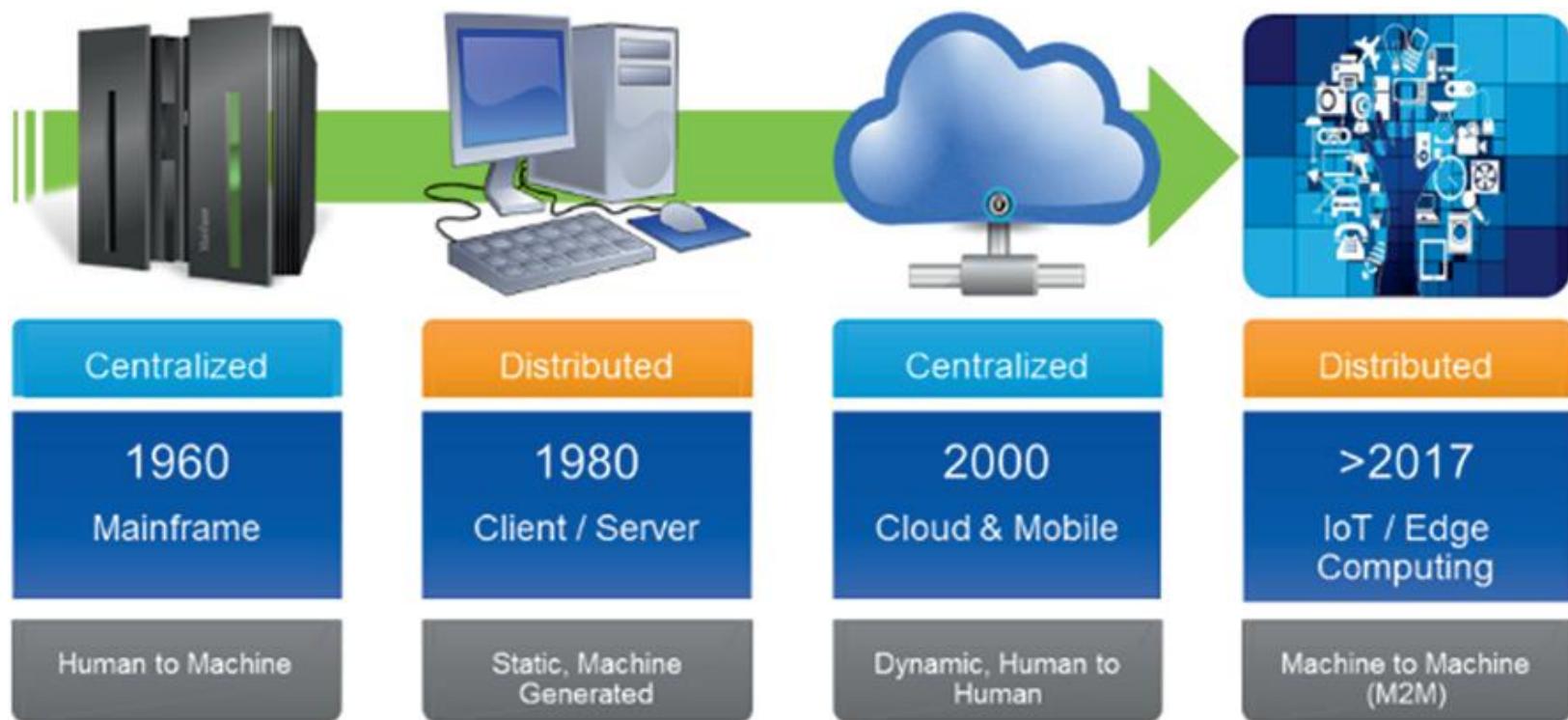
Strategia, w której  
bardzo małe  
opóźnienia i reakcja w  
czasie rzeczywistym  
mają kluczowe  
znaczenie dla  
działania aplikacji i  
zdadowolenia  
użytkownika



# Edge computing

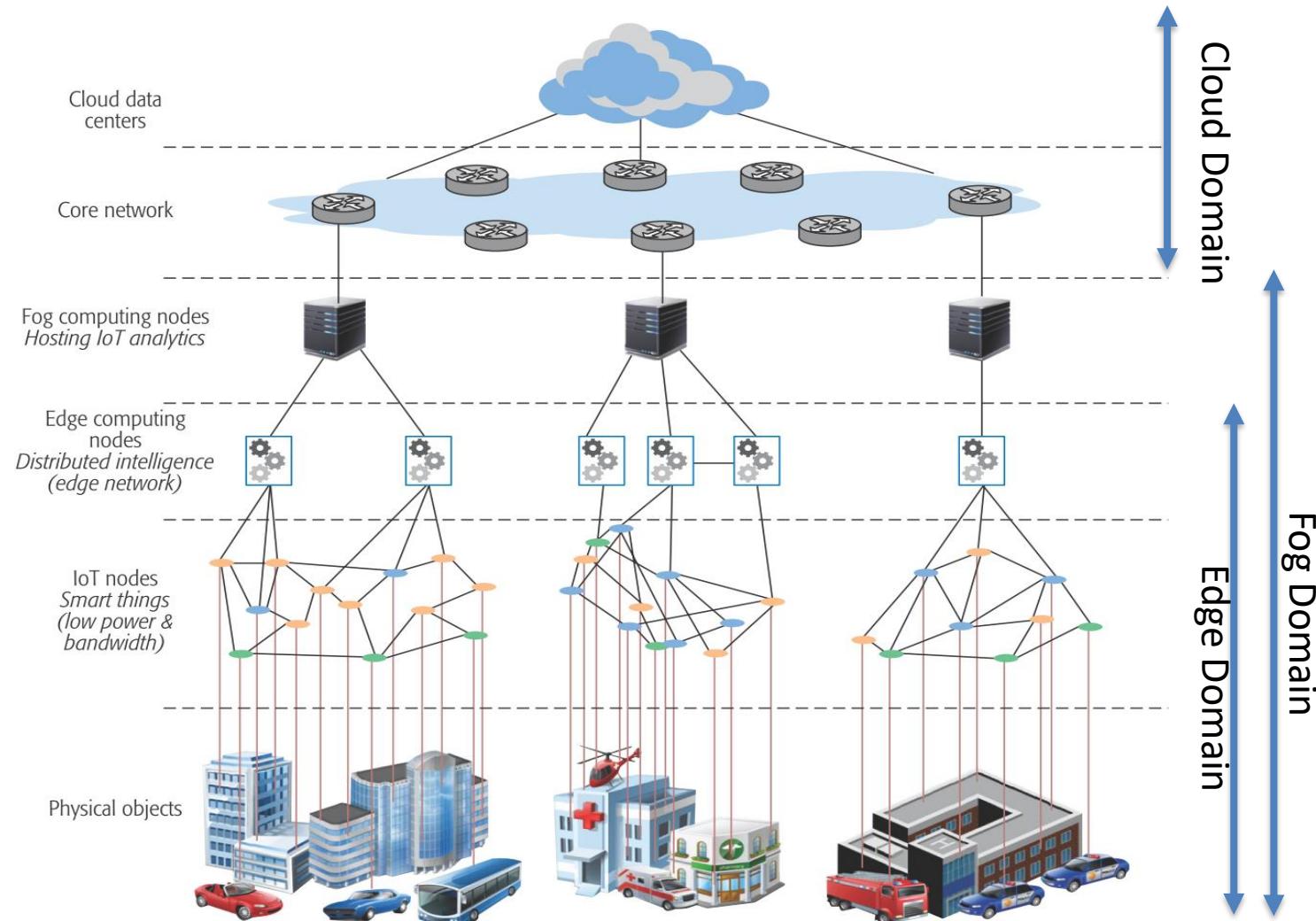


# Główne etapy przetwarzania informacji w sieci



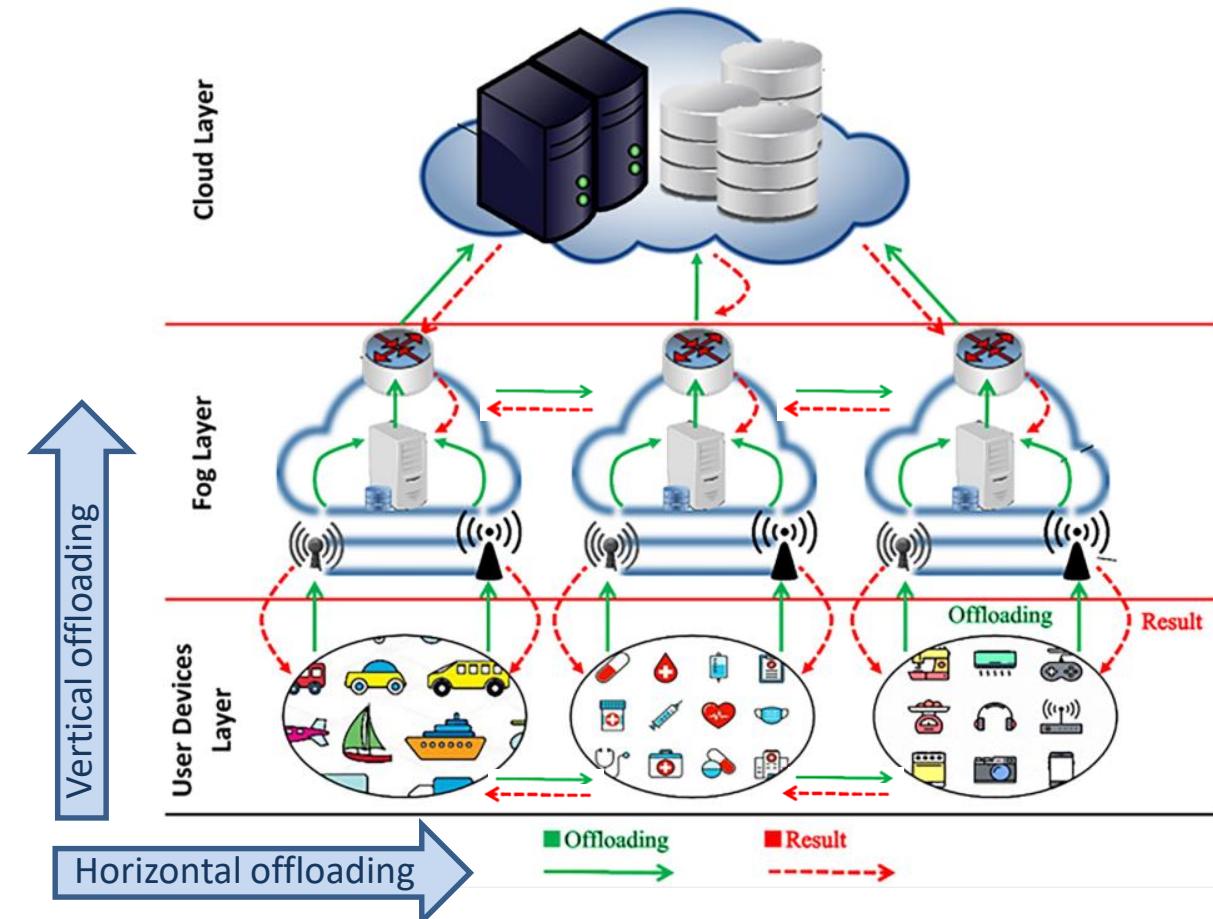
A Review on Evolution of Architectures, Services, and Applications in Computing Towards Edge Computing, 2022

# IoT-Edge-Fog-Cloud Continuum



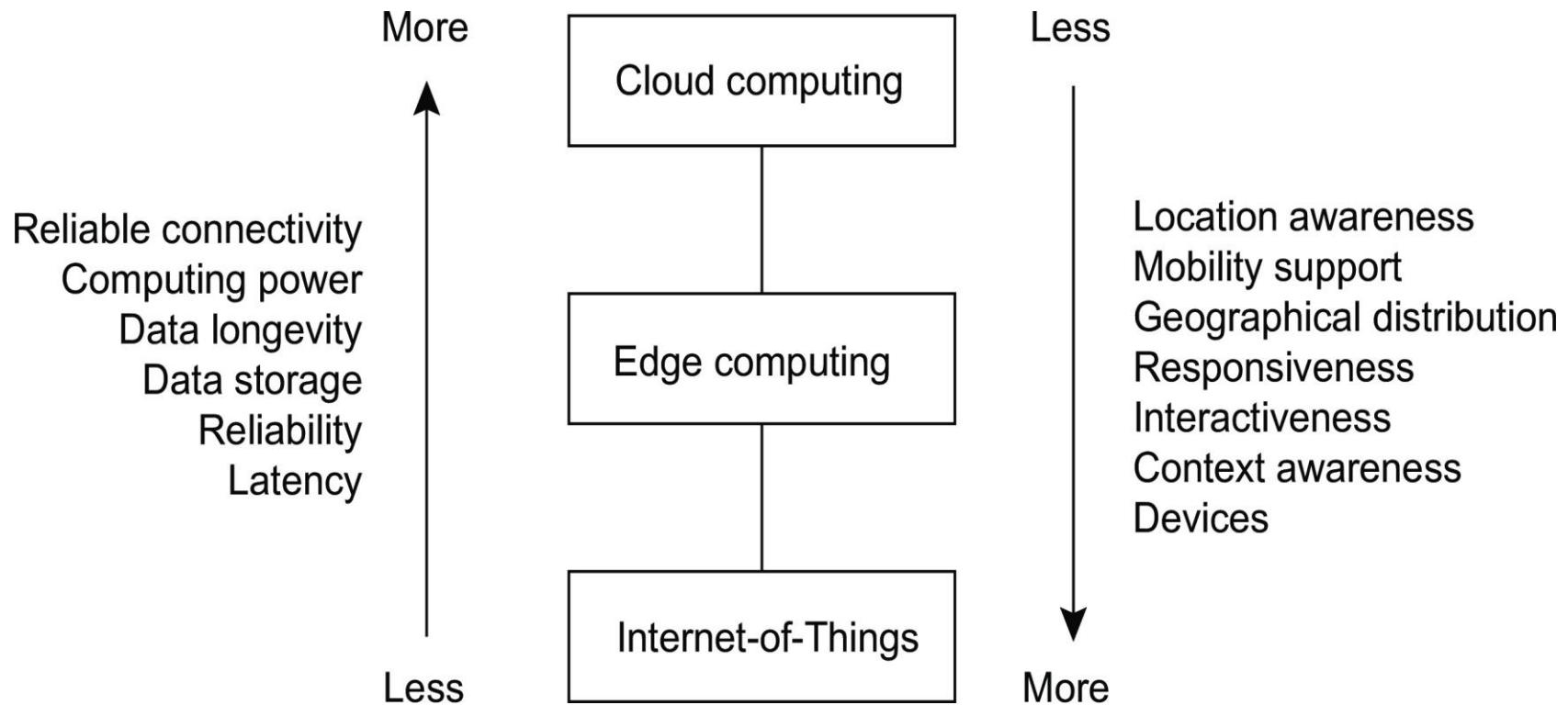
Jak tworzyć oprogramowanie?

# Architektura IoT-Edge-Fog-Cloud



- Koszt komunikacji
- Efektywność energetyczna
- Wydajność obliczeniowa
- Czas wykonania
- Wykorzystanie zasobów
- Bezpieczeństwo

# IoT-Edge-Fog-Cloud continuum



# Systemy rozproszone – różne perspektywy

Systemy rozproszone są złożone - należy przyjąć perspektywę

- Architektura: wspólne organizacje
- Proces: jakiego rodzaju procesy i ich relacje
- Komunikacja: urządzenia do wymiany danych
- Koordynacja: algorytmy niezależne od aplikacji
- Nazewnictwo: jak identyfikować zasoby?
- Spójność i replikacja: wydajność wymaga danych, które muszą być takie same.
- Odporność na błędy: kontynuacja działania w przypadku częściowych awarii
- Bezpieczeństwo: zapewnienie autoryzowanego dostępu do zasobów

# Systemy rozproszone – pułapki

Często przyjmuje się wiele fałszywych (i często ukrytych) założeń (ang. fallacies) :

- Sieć jest niezawodna
- Sieć jest bezpieczna
- Sieć jest jednorodna
- Topologia się nie zmienia
- Opóźnienie wynosi zero
- Przepustowość jest nieskończona
- Koszt transportu wynosi zero
- Jest jeden administrator
- Architektura: wspólne organizacje

[Dziękuję ☺ ]