

PDV 08 2017/2018

# Úvod do distribuovaných systémů

Michal Jakob

[michal.jakob@fel.cvut.cz](mailto:michal.jakob@fel.cvut.cz)

Centrum umělé inteligence, katedra počítačů, FEL ČVUT





Paralelní  
výpočty

Distribuované  
výpočty

# Plán (ne zcela finální)

T	Přednáška	Cvičení
9	Úvod do distribuovaných systémů (DS). Detekce selhání.	Seznámení se s frameworkem.
10	Čas a uspořádání v DS.	Látka z přednášky. Zadání úlohy #6 (2b.)
11	Globální stav a distribuovaný snapshot. Doručovací protokoly.	Látka z přednášky. Zadání úlohy #7 (2b.)
12	Konsensus v DS.	Látka z přednášky. Zadání semestrální práce #2 (14 b.)
14	Koordinační algoritmy: volba lídra (koordinátora) a distribuované vyloučení procesů.	Látka z přednášky

# Studijní materiály

Slidy

Hlavní kniha: Maarten van Steen, Andrew S. Tanenbaum: **Distributed Systems (3.01 Edition), 2017**, k dispozici online: <https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/>

Sekundární kniha: George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair: **Distributed Systems: Concepts and Design (5th Edition), 2011**

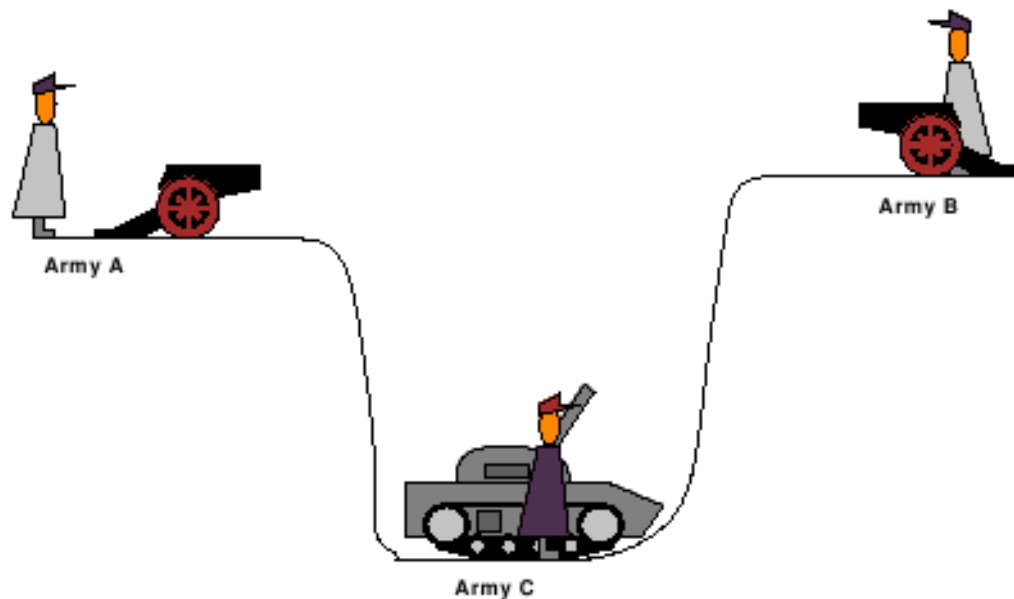


# Úvod do Distribuovaných Systemů

# THE TWO GENERALS PROBLEM



# Dva generálové



Pouze **synchronizovaný útok uspěje** → generálové se potřebují dopředu **shodnout na čas**, kdy oba zaútočí.

Komunikují skrze **posílání zpráv**. Poslané zprávy se mohou **ztratit...**

Jak **zaručit**, že zaútočí ve stejný čas?

# Dva generálové

## Možné řešení:

- Generál A: zaútočit za úsvitu!
- Generál B: potvrzují, zaútočit za úsvitu!
- Generál A: potvrzují, potvrzují, zaútočit za úsvitu!
- Generál B: potvrzují, potvrzují, potvrzují, zaútočit za úsvitu!
- ...

Lze ukázat: Řešení problému Dvou generálů za těchto předpokladů **neexistuje!**



# Reality Check

## Příklad: Výběr z bankomatu

- **vyberete** si z bankomatu v Praze 1000 Kč
- ze zůstatku z vašeho účtu se **odečte** 1000 Kč

Problém **atomického commitu**: atomický commit je sada více operací, které jsou provedeny jako jedná operace.

- zúčastněné systémy musí zkoordinovat zda a kdy tyto operace budou provedeny

# Pragmatické řešení problému Dvou Generálů



Předpokládejme, že pravděpodobnost chycení kurýra je  $p$  a že opakované zachycení kurýra jsou nezávislé jevy.

Možné řešení:

- Generál A pošle  $n$  kurýrů a **každopádně** zaútočí za úsvitu
- Generál B zaútočí za úsvitu pokud k němu dorazí **aspoň jeden** kurýr

Pravděpodobnost **nekoordinovaného útoku** je  $p^n$ .

- snížit pravděpodobnost nekoordinovaného útoku můžeme posláním více kurýru ...
- ... za cenu nutnosti poslat více kurýrů

**Jistoty** koordinovaného útoku ale **dosáhnout nemůžeme** (pokud  $p > 0$ ).

# Obecně

## Řešitelnost problémů v distribuovaných systémech

- Řada zdánlivě jednoduchých problémů nemá v distribuovaných systémech 100% řešení...
- ...ale mají pragmatická řešení.
- Některé problémy 100% řešení mají...
- ... ale jen za určitých předpokladů.

Pochopit, které problémy jsou které, a jak se dají řešit je cílem tohoto předmětu.



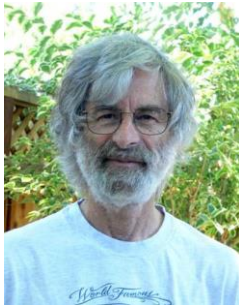
Co to je distribuovaný  
systém?

# Co to je DS?



## Definice (optimistická)

Soubor (*collection*) **autonomních výpočetních elementů**, které se uživateli jeví jako **jeden koherentní systém**. [Andrew Tanenbaum]



## Definice (pesimistická)

Systém, ve kterém selhání počítače, o které jste vůbec nevěděli tušení, že existuje, učiní váš vlastní počítač nepoužitelný. [Leslie Lamport]

## Definice (pragmatická)

Soubor (*collection*) nezávislých, **autonomních výpočetních elementů** propojených **komunikační sítí**. Výpočetní elementy komunikují formou **posílání zpráv** za účelem určité formy **spolupráce**.

# Příklady distribuovaných systémů



Mezibankovní  
platby

MMOG /  
Pokémon Go

Uber / Taxi  
dipečink

Sensorové  
sítě

P2P síť

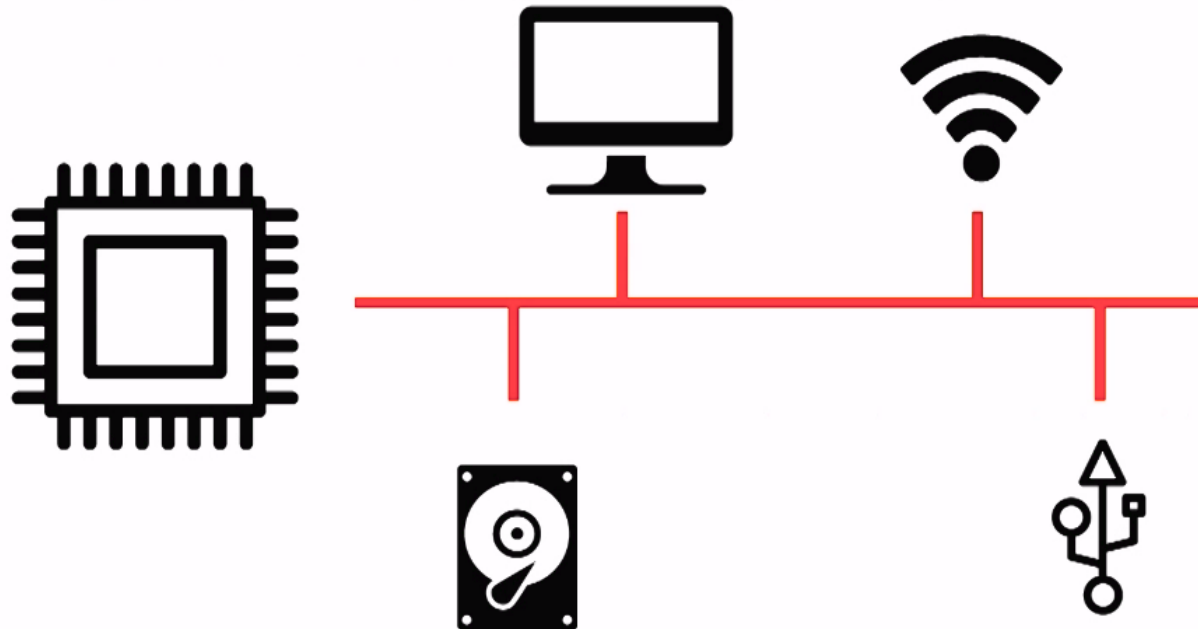
Blockchain

Řízení výrobní  
linky

Datové  
centrum

Internet

# Je počítač distribuovaný systém?



Není. → Proč?

# Charakteristiky DS

## V distribuovaném systému:

1. Počítače pracují **souběžně** (concurrently)
2. Počítače **nesdílejí** globální **hodiny**
3. Počítače **selhávají** **nezávisle**

## Naopak:

## Paralelní systémy sdílejí

1. **Paměť**: předávání stavu používá sdílenou paměť a synchronizační mechanismy
2. **Hodiny**: přístup ke společným hodinám
3. **Osud**: selže buď nic nebo vše



# Nezávislá (a částečná) selhání

Nezávislá selhání komplikují život vývojářům

**Závislosti** mezi jednotlivými počítači DS **zesilují následky** selhání

$n$  závislých procesů,  
pravděpodobnost selhání  $p \rightarrow$   
**dostupnost  $(1 - p)^n$**



Nezávislá selhání usnadňují život uživatelům

**Replikace** pomocí více nezávislých počítačů **zvyšuje odolnost** vůči selhání

$n$  nezávislých procesů,  
pravděpodobnost selhání  $p \rightarrow$   
**dostupnost  $1 - p^n$**

Nejen odolnost, ale taky např.  
**výkonnost**

# Proč existují DS?

## Inherentní distribuce

Aplikace vyžadující sdílení zdrojů nebo šíření informace mezi geograficky nebo organizačně **vzdálenými** entitami jsou „**přirozené**“ distribuované systémy.

Např. banka se několika pobočkami, senzorová síť, ...



## Distribuce jako nástroj

Distribuce může být cílenou **volbou** v **softwarovém návrhu** směřující k splnění specifických **požadavků**: odolnost vůči selhání, zvýšený výkon nebo poměr cena / výkon, nebo minimální QoS.

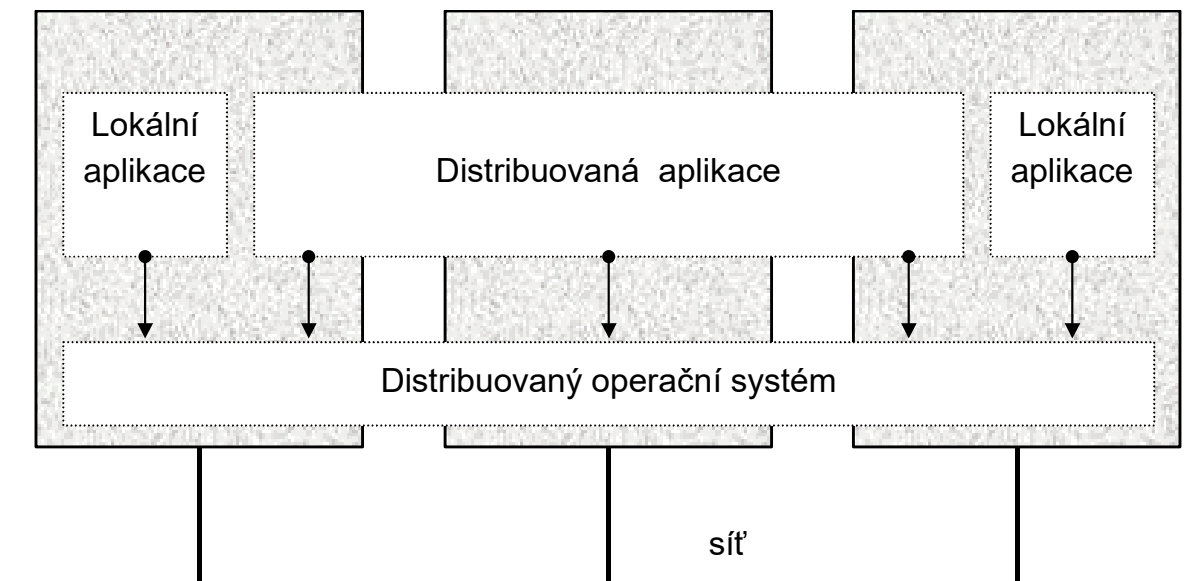
Např. replikované servery

# 199x: Distribuované operační systémy

Operační systém vyvinutý speciálně pro potřeby DS

- “zlatý věk”: Amoeba, Sprite, V, Chorus, Mosix ... T4
- **součástí jádra** OS podpora distribuovanosti, komunikace, synchronizace
- aplikace: **transparentní** využití distribuovaných služeb

Neexistuje rozdíl mezi lokální a distribuovanou aplikací

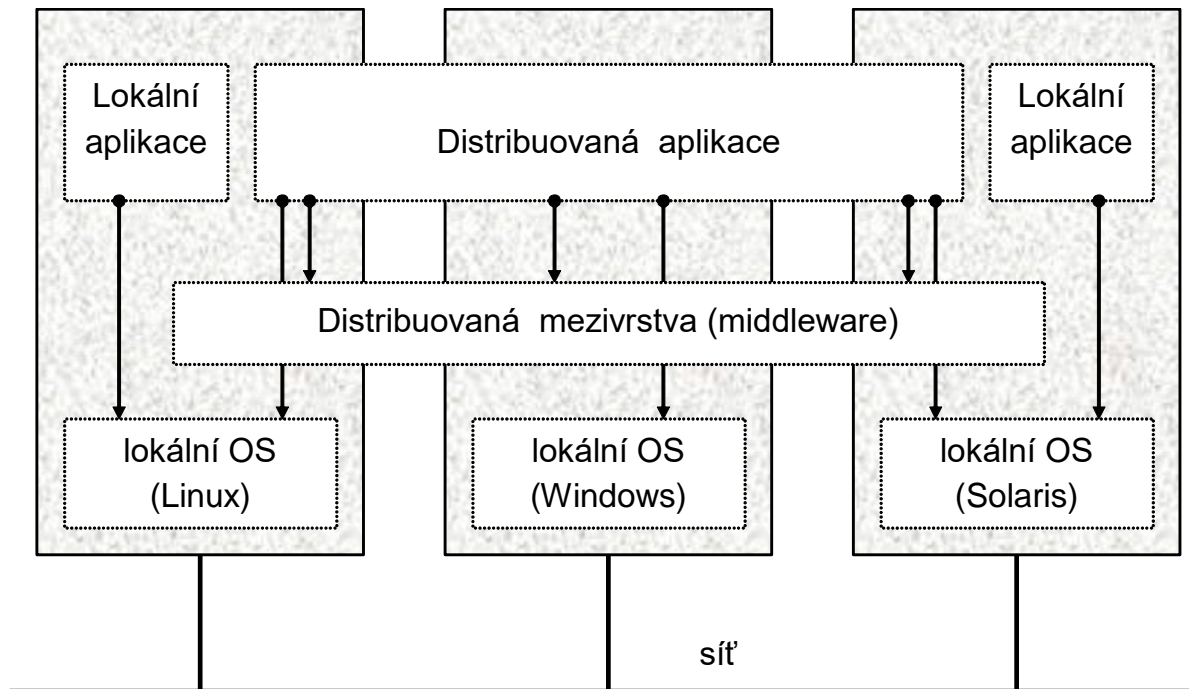


**Nesplněná očekávání:**  
žádný systém  
nedotažen do masově  
použitelného stavu

# 200x - Distribuované frameworky a aplikace

Lokální OS (Win/Ux) + rozšiřující vrstva pro distribuované aplikace

Prostředí pro distribuované aplikace → **middleware** (OSF DCE, CORBA, DCOM, Globe, ..)

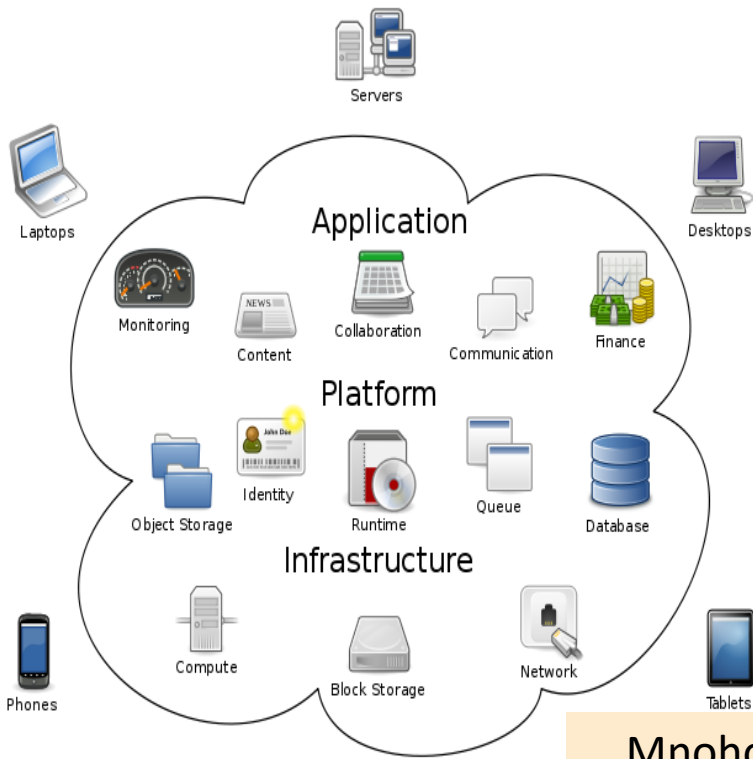


**(Relativně) úspěšné**  
využití pro cluster /  
grid computing

# 201x - Cloud computing, XaaS

## Service-oriented computing: SaaS, PaaS, IaaS

- nutná podpora **virtualizace**
- kompletní infrastruktura připravená k použití:  
Windows Azure, Google App Engine, Amazon ECC, OpenStack, ...



Cloud Computing

Mnoho různých  
úspěšných aplikací

<i>Execution Models</i>	Virtual Machines	Web Sites	Cloud Services		
<i>Data Management</i>	SQL Database	Tables	Blobs		
<i>Networking</i>	Virtual Network	Connect	Traffic Manager		
<i>Business Analytics</i>	SQL Reporting	Hadoop			
<i>Messaging</i>	Queues	Service Bus			
<i>Caching</i>	Caching	CDN			
<i>Identity</i>	Windows Azure Active Directory				
<i>High-Performance Computing</i>	HPC Scheduler				
<i>Media</i>	Media Services				
<i>Commerce</i>	Marketplace				
<i>SDKs</i>	.NET	Java	PHP	Python	Node.js

# Hlavní typy DS

DS pro vysoko  
výkonnostní výpočty  
(high-performance  
computing)

Distribuované  
informační systémy

DS pro pervazivní  
výpočty (pervasive  
computing)

# DS pro spolehlivé a vysoko výkonnostní výpočty

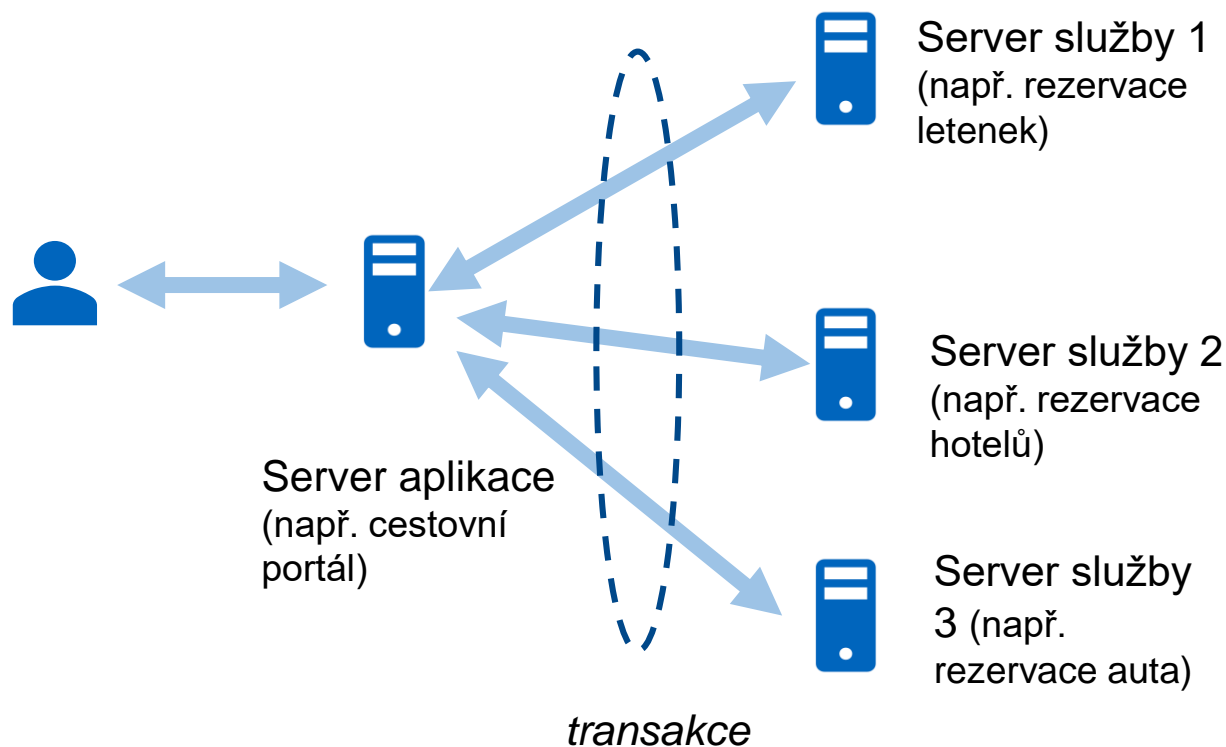
**DS jako nástroj** pro zvýšení výkonů, spolehlivosti a ekonomické efektivity.

Model sdílené paměti se na multi-počítačové architektury **nepodařilo rozšířit** → Další škálování možné jen v paradigmatu distribuovaných výpočtů založených na posílání zpráv.



# Distribuované informační systémy

Distribuce vynucena příslušností jednotlivých výpočetních uzlů do **různých organizací**.



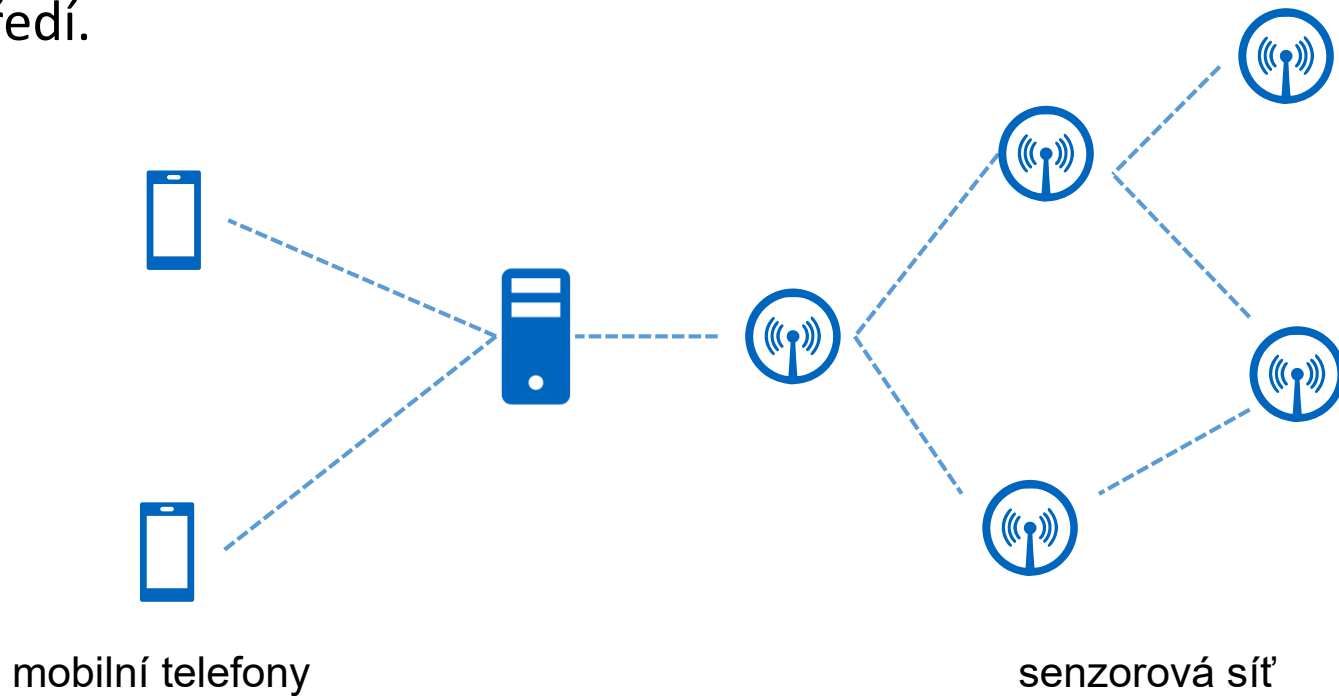
Hlavní výzva: **spolehlivost a konzistence.**



# Pervazivní DS

Malé, mobilní výpočetní uzly, často **vtělené do fyzického prostředí**.  
Omezení na dosah a rychlost komunikace, a na spotřebu energie.

Distribuce vynucena nutností **fyzické blízkosti** uzlu k uživateli nebo prostředí.



Hlavní výzva: **efektivita, dostupnost, ....**



# Modelování DS

# Klíčové elementy DS

Výpočet

Interakce

Selhání

Čas

# Výpočet

**Proces:** jednotka výpočtu v distribuovaném systému (někdy nazýván uzel, hostitel, element, ...)

**Množina (skupina) procesů:** označována  $\Pi$  – je složena ze souboru  $N$  jednoznačně identifikovaných procesů  $p_1, p_2, \dots, p_N$ .

**Klasické předpoklady:**

- množina procesů je **konstantní** ( $N$  je dobře definováno)
- procesy se navzájem **znají**
- BÚNO: všechny procesy provádějí kopii stejného **algoritmu** (souhrn těchto kopií vytváří distribuovaný algoritmus)

V tzv. **extrémních** distribuovaných systémech tyto předpoklady nemusí být splněny.

# Interakce

Procesy komunikují **zasíláním zpráv**

- $send(m, p)$  pošle zprávu  $m$  procesu  $p$
- $receive(m)$  přijme zprávu  $m$

Zprávy mohou být v některých případech jednoznačně **identifikovány**

- odesílatelem zprávy
- sekvenčním číslem, které je lokální odesílateli

Klasickým předpoklad: každý pár procesů je propojen **obousměrným** komunikačním **kanálem**

- plně propojená topologie může být realizována pomocí směrování (routingu)

# Selhání

Jak procesy, tak komunikační kanály mohou v DS selhat.

## Selhání procesu

- **havárie (crash/fail-stop):** proces přestane vykonávat algoritmus (a reagovat na zprávy)
- **libovolné (byzantské) selhání:** proces může pracovat dále (a reagovat na zprávy), ale vykonává chybný algoritmus (z důvodu softwarové chyby nebo úmyslu)

## Selhání kanálu

- **ztráta zprávy (message drop):** zpráva není doručena cílovému procesu (např. kvůli přetížení sítě nebo přetečení zásobníku v OS u přijímacího procesu)
- **rozdělení (partitioning):** procesy jsou rozdělené do disjunktních množin (oddílů - partitions) tak, že v rámci oddílu je komunikace možná, ale mezi oddíly nikoliv

V případě synchronních DS definujeme ještě **selhání časování**, pokud doba odezvy procesu nebo přenosu zprávy po síti vybočila z dohodnutého **časového rozmezí**.

# Předpoklad na komunikační kanál

**Dokonalý**  
(perfect)  
kanál

**Spolehlivé doručování:** Pokud proces  $p$  pošle zprávu procesu  $q$  ani  $p$  a ani  $q$  nehavaruje, pak  $q$  nakonec zprávu obdrží.

Žádná **duplikace:** Žádná zpráva není doručena vícekrát než jednou.

Žádné **vytváření:** Je-li zpráva  $m$  doručena procesu  $p$ , tak zpráva  $m$  byla dříve poslána nějakým procesem  $q$  procesu  $p$ .

**Garantované pořadí doručování:** Odešle-li proces  $p$  procesu  $q$  zprávy  $m_1$  a  $m_2$ , tak pokud byla  $m_1$  odeslána dříve než  $m_2$ , tak  $m_2$  nemůže být doručena aniž by předtím byla doručena  $m_1$ .

**Spolehlivý**  
kanál

**FIFO** kanál

# Čas

**Přesné globální hodiny** by umožnily globální uspořádání výpočetních kroků v DS. Bohužel **neexistují**.

Každý proces má své **lokální hodiny**.

Lokální hodiny **nemusí** ukazovat **přesný čas**.

**Synchronizace** lokálních hodin je možná **jen** s určitou **přesností**.



# DS a Čas

V distribuovaných systémech je obtížné uvažovat o čase nejen kvůli absenci globálních hodin, ale také proto, že obecně nelze dát časové limity na **komunikaci** a **délku výpočtů**.

Různé možné modely:

- **Asynchronní DS**
- **Synchronní DS**
- **Částečně synchronní DS**

# Synchronní vs. Asynchronní

## Asynchronní systém

- Žádné časové limity na **relativní rychlost** vykonávání procesů.
- Žádné časové limity na **trvání přenosu** zpráv.
- Žádné časové limity na **časový drift** lokálních hodin



## Synchronní systém

- **Synchronní výpočty:** známe horní limit na relativní rychlost vykonávání procesů.
- **Synchronní komunikace:** známé horní limit na dobu přenosu zpráv.
- **Synchronní hodiny:** procesy mají lokální hodiny a je znám horní limit na rychlosti driftu lokálních hodin vzhledem k globálním hodinám.

# Částečně synchronní DS

**Částečná synchronicita:** pro většinu systému je relativně snadné definovat časové limity, které platí **většinu času**. Občas se ale mohou vyskytnout období, během kterých tyto časové limity neplatí.

- zpoždění procesů: např. swappování, garbage collection
- zpoždění komunikace: přetížení sítě, ztráta zpráv (vyžadující jejich opakovaný přenos)

Prakticky užitečné systémy **jsou částečně synchronní**

→ Umožňuje **v praxi vyřešit** problémy, které nejsou za předpokladu plně asynchronních DS řešitelné.

- některé z časových úseků synchronního běhu DS jsou dostatečně dlouhá na to, aby distribuovaný výpočet skončil

# Chybné předpoklady při vývoji DS

Řada DS je zbytečně komplexních. Komplexita je způsobena chybami, které je třeba později záplatovat. Chyby vycházejí z mylných předpokladů.

## Typické mylné předpoklady

- Síť je spolehlivá
- Síť je zabezpečená
- Síť je homogenní
- Topologie sítě se nemění
- Síť má nulovou latenci
- Neomezená kapacita sítě
- Systém má jednoho administrátora