

# Návrh softwarových systémů - Sdílení dat

Jiří Šebek    *Návrh softwarových systémů  
(B6B36NSS)*



```
public final void onSensorChanged(SensorEvent event)
{
    m_flightIntensity = event.values[0];
    m_etAmblight.setText("" + m_flightIntensity + " lx");
}

... resume()
... light, ... (NORMAL);
```

# Sdílení dat

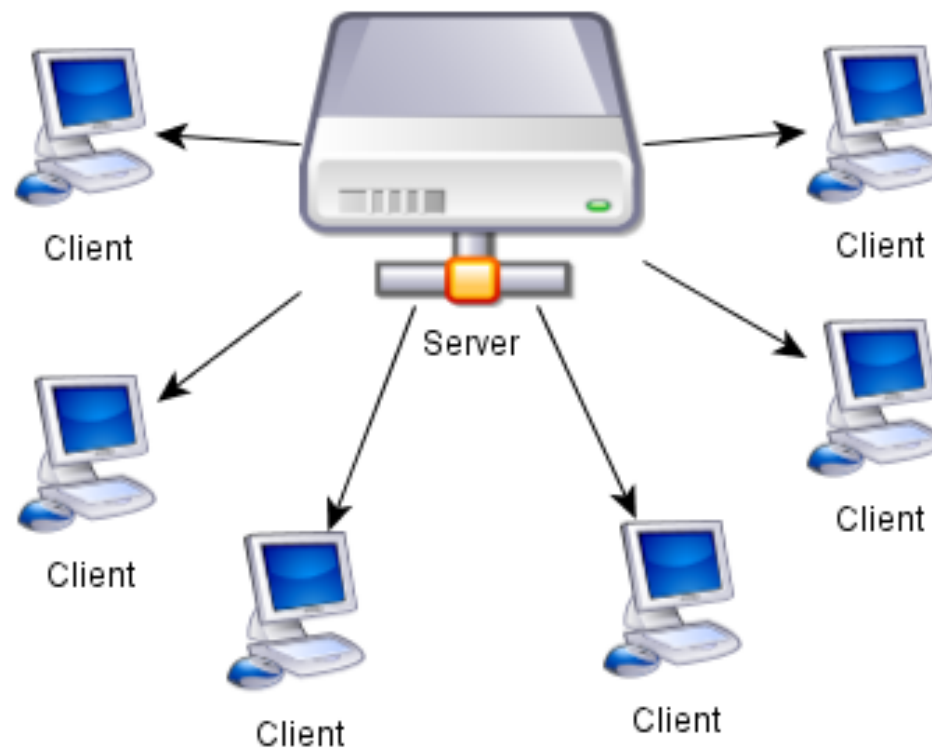
- Sdílení dat může probíhat různě podle architektury
- V našem případě : client-server
- Replikace
  - žádná
  - částečná
  - úplná
    - primary copy model
    - gossip architecture

# Architektury P2P

- P2P = peer to peer
  - Rovný s rovným
  - Protějšek k klient - server
- P2P = point to point
  - přímé spojení mezi dvěma síťovými uzly

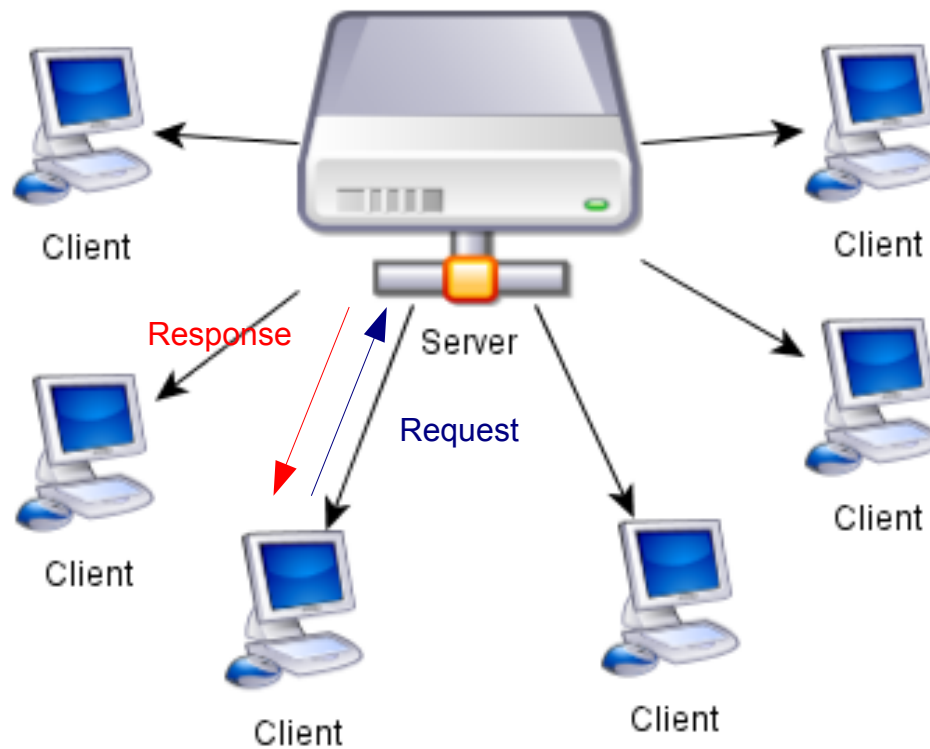


# Sdílení dat



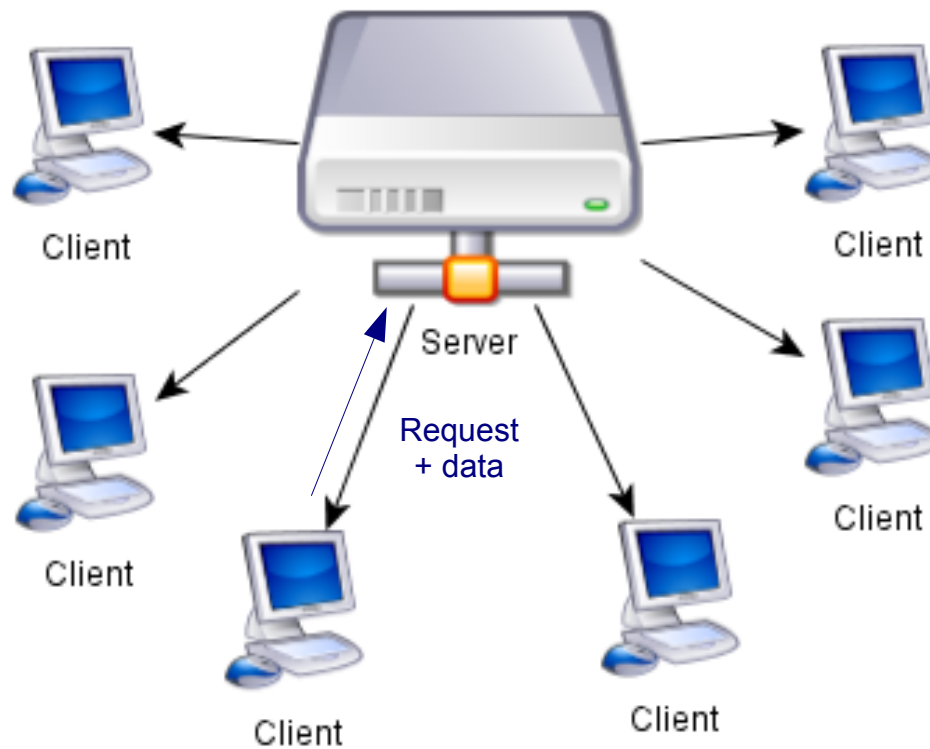
# Sdílení dat

čtení



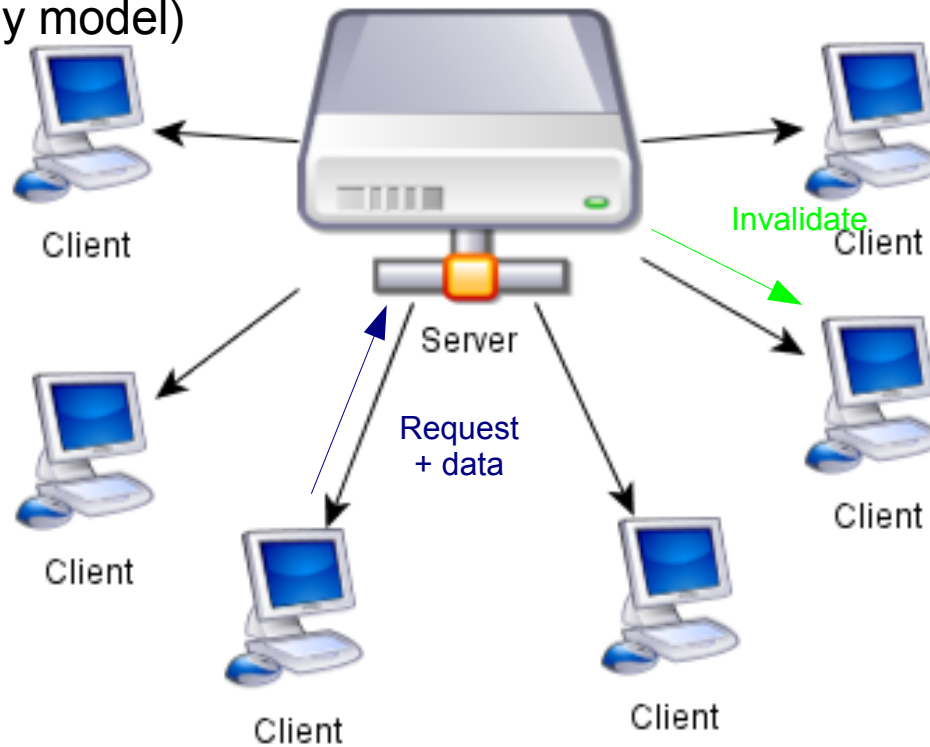
# Sdílení dat

zápis



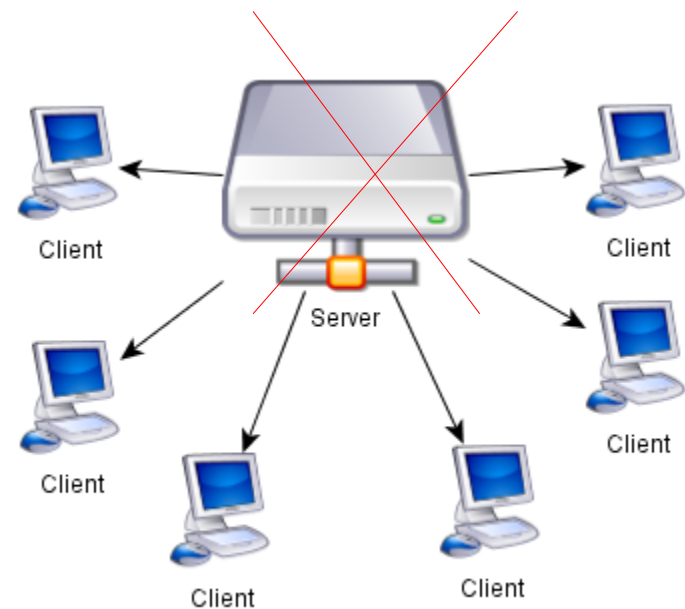
# Sdílení dat

Zápis – využití cache  
(read-only model)



# Mobilita

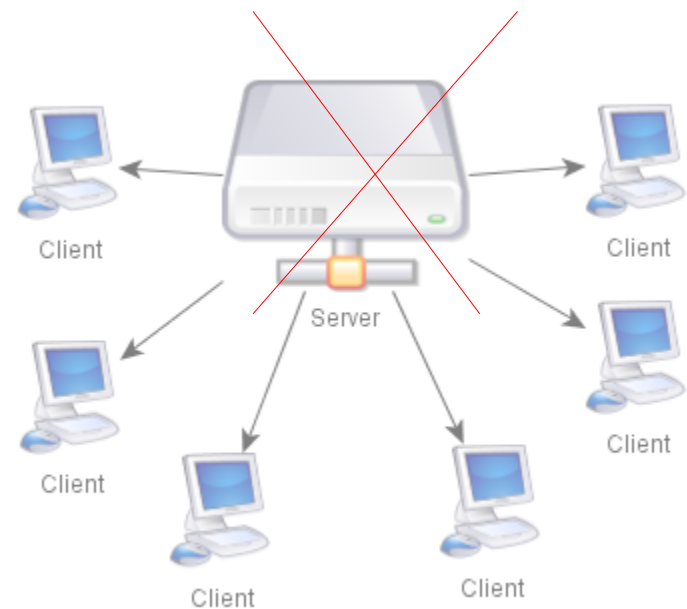
- Vynecháme centrální prvek architektury (pouze klienti)
- Čtení – klient 1 pošle request a klient co data má mu je pošle
- Zápis – klient 1 pošle invalidate packet klientům co mají stará data





# Mobilita

- Plná replikace
  - Čtení – každý klient má lokálně data uložené
  - Zápis – je potřeba rozselet data všem klientům (silně zahlcuje síť)



# Replikace dat

- Plná replikace
  - Primary-copy model
    - Existuje jeden primární prvek pro uchovávání dat ke kterému přistupují klienti nebo servery
    - Tento prvek se poté zálohuje do «Replic »
  - Gossip architektura
    - Existuje více prvků s daty ke kterým přistupují klienti nebo servery
    - Tyto datové prvky spolu komunikují pomocí gossip zpráv (výměna, aktualizace dat)

# Konzistence a koherence dat

- Konzistence
  - Pravidla pro všechny paměťové operace
- Koherence
  - Pravidla pro přístupy k jednotlivým paměťovým místům
- Cache
  - Vyrovnávací paměť (skrytá)
  - Různé úrovně (L1-3)
  - Každá má svoji rychlost a velikost (L1 nejmenší nejrychlejší)

# Konzistence a koherence dat

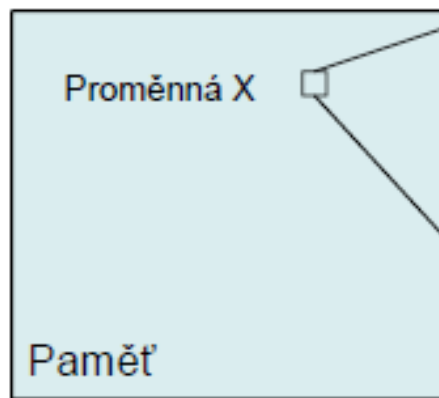
Řekneme, že paměťový systém multiprocesorového systému je **koherentní**, jestliže výsledek jakéhokoli provádění programu je takový, že pro každé paměťové místo je možné sestavit myšlené sériové pořadí čtení a zápisů k tomuto paměťovému místu a platí:

1. Paměťové operace k tomuto paměťovému místu pro každý proces jsou provedeny v pořadí, ve kterém byly spuštěny tímto procesem.
2. Hodnoty vrácené každou operací čtení jsou hodnotami naposledy provedené operace zápis do tohoto paměťového místa vzhledem k sériovému pořadí.

# Konzistence a koherence dat

```
Proces P1:  
X=0;  
if (X ==0) {  
    y=fun ();  
    X = 1;  
}
```

```
Proces P2:  
X=0;  
while (X ==0)  
    { ; }  
X = 2;
```



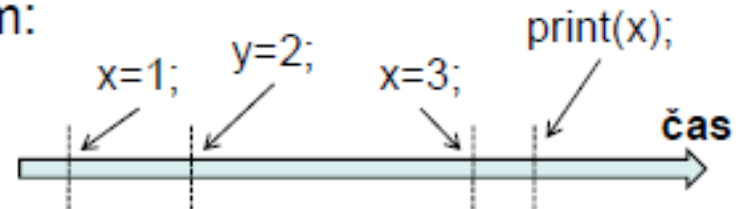
P2: X=0;	P2: read(X)
P1: X=0;	P2: X=0;
P1: read(X)	P1: X=0;
P2: read(X)	P1: read(X)
P2: read(X)	P2: read(X)
P1: X=1;	P1: X=1;
P2: read(X)	P2: read(X)
P2: read(X)	P2: read(X)
P2: X=2;	P2: X=2;

Máme jistotu, že když P2 uvidí  $X==2$ , bude funkce `fun()` volaná procesem P1 se všemi důsledky vykonána?

# Striktní konzistence

- Jednoprocesorový systém:

```
x = 1;  
y = 2;  
x = 3;  
print (x) ;
```

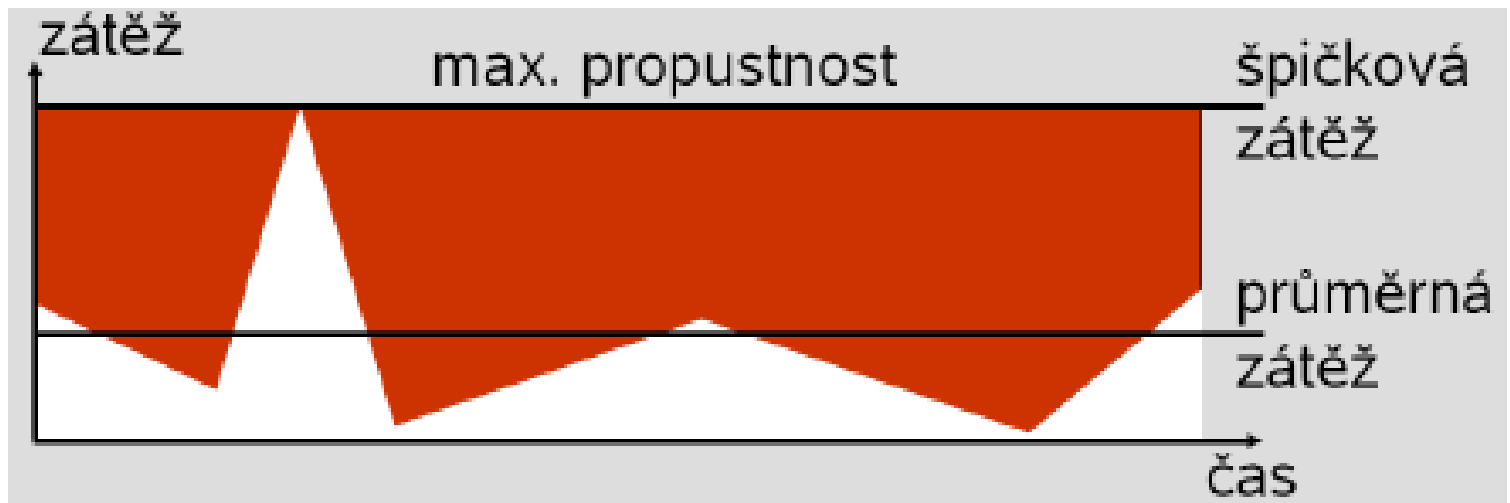


(Jakékoliv čtení z paměti z adresy  $x$  vrátí hodnotu uloženou při posledním zápisu na adresu  $x$ .)

- Víceprocesorový systém:
  - podmiňuje existenci přesného globálního času ve všech uzlech a okamžité propagování změn
  - nerealistický až absurdní požadavek

# Přetěžování sítě

- Jak zajistit eliminace peaků ve využití sítě (přístupu k aplikacím) + zamezit nevyužitý propustnosti ?
- → Cloud (openshift.. )



**Děkuji za pozornost**

Jiří Šebek

