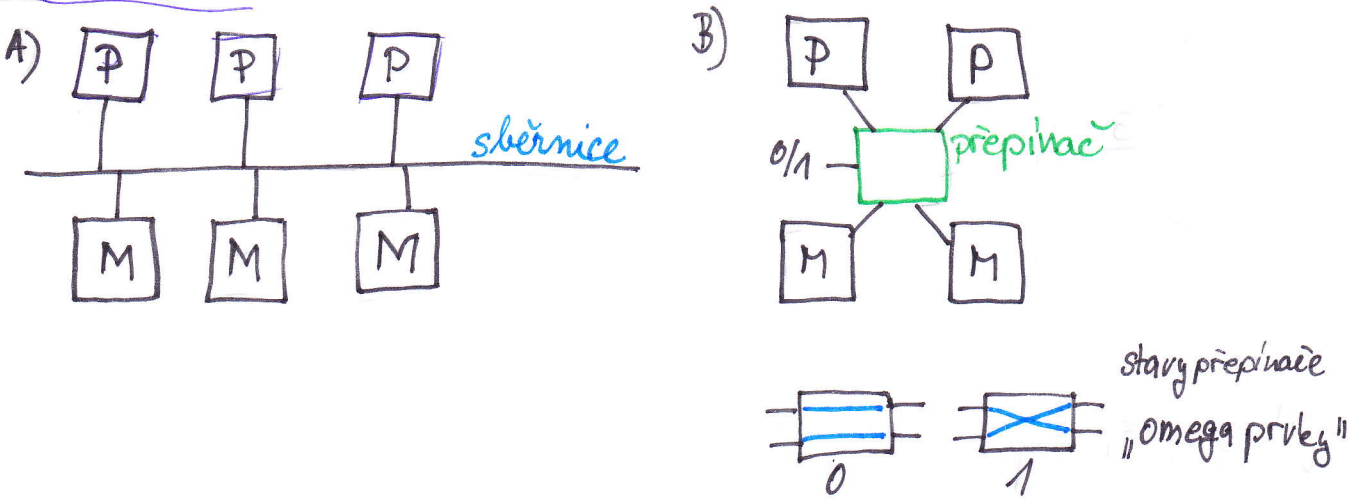
Dělení systémů (Flynn 1972)

SISD - Single Instruction Single Data

SIMD - vektorové

MIMD - paralelní a distribuované

MIMD propojení



MIMD zpožděmí

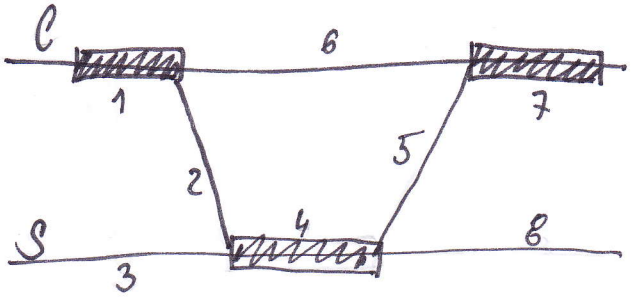
DRUH KOMUNIKACE

- volně vázané - multipočítače (kom. systémy)
- těsně vázané - multiprocesory (sdílená paměť)

V(s): s++
P(s): while (s==0); s--

} ⇒ atomická instrukce (u volně vázaných systémů mnohem složitější realizace)

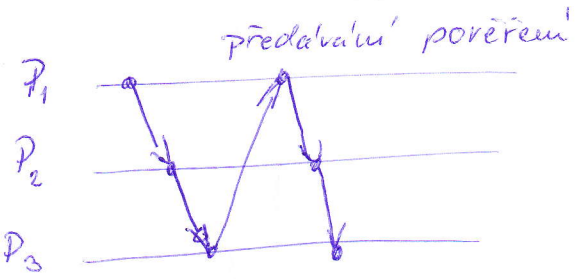
Architektura klient/server

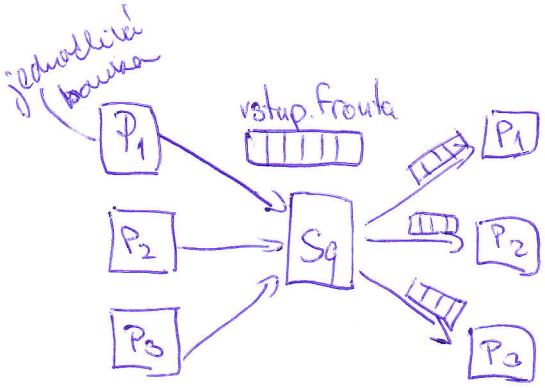


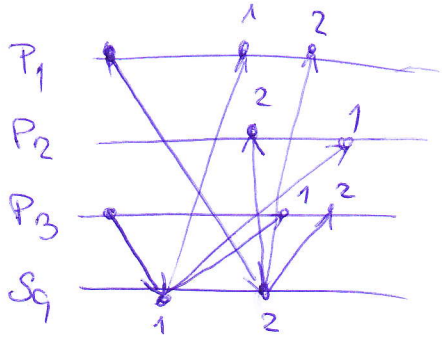
KIV/DS – cvičení 1

1. SP – Sequencer

– OS Linux







– nutné každou událost/transakci řádně označit, aby došlo k jednoznačnému dodržení pořadí

– stačí jeden bankovní účet

Porty:

1 port pro Sq

4 porty pro každou banku

Configurační soubor:

ID | PORT | IP | COLOR

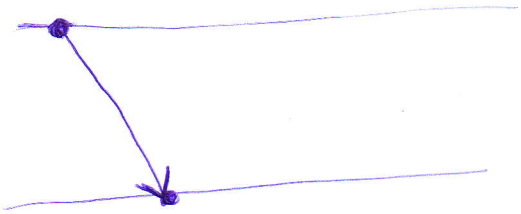
-C <řádka> [<config>]

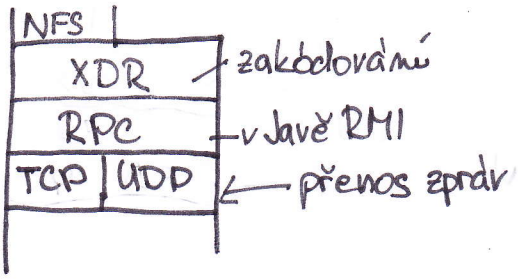
– přidání nové banky za běhu není potřeba řešit

– výstupem je obr. zprac. v prog. „dia"

– v jednu chvíly pouze jeden marker (tj. jedno zjišťování glob. stavu)

2.SP - Globalni stav

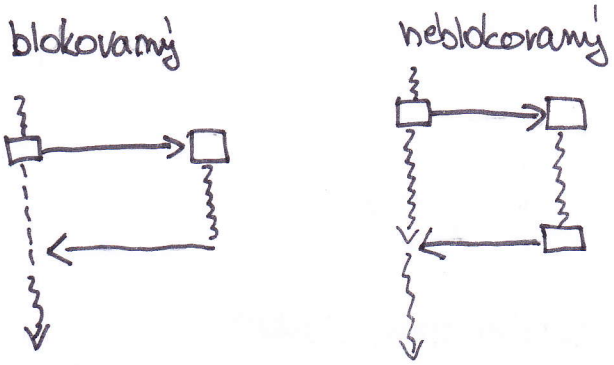




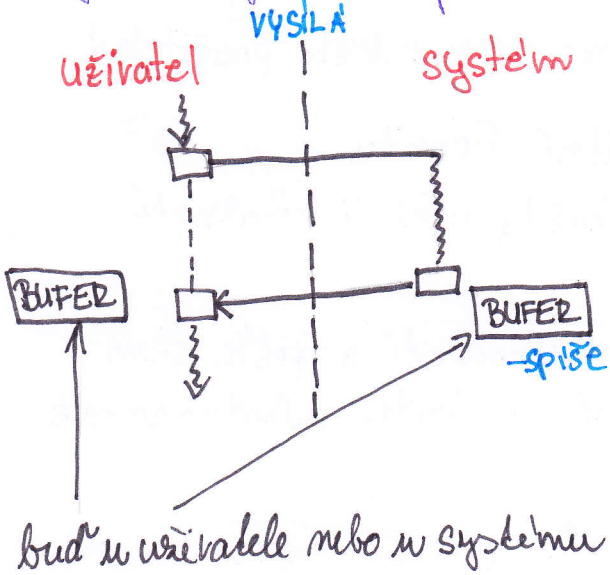
CORBA

- standard pro volání (vzdálených) metod (univerz. pro všechny platformy)
- základní je SW sběrnice

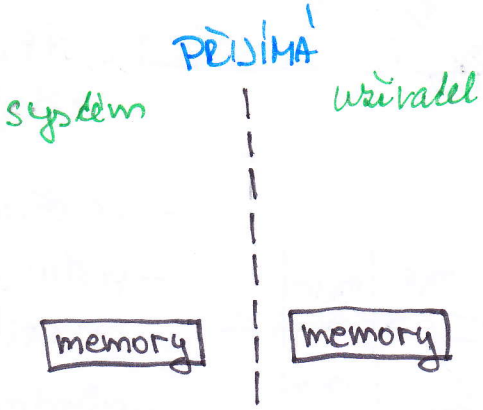
Posílání zpráv



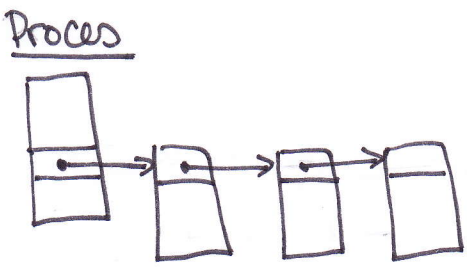
- s využitím vyrovnávací paměti



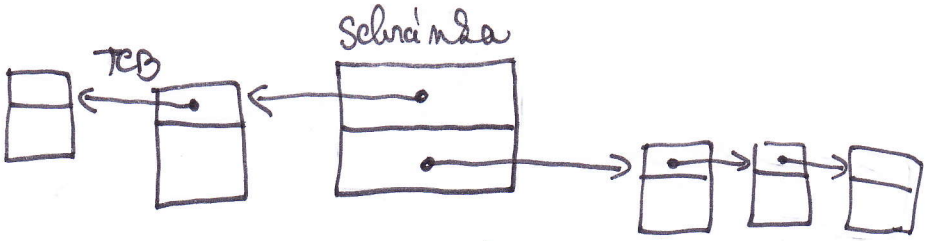
buď u uživatele nebo u systému



Přímá komunikace



nepřímá



- využíval RMX = kiosk OS pro 8080

Pozn.: přednáškové slidy je vhodné přečíst

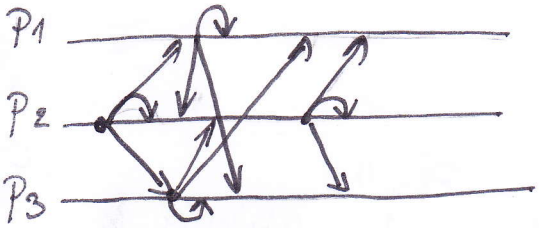
RMI
- mapovalu přes RMI-registry

RPC
- map. přes porty

Spolehlivé protokoly - BROADCAST

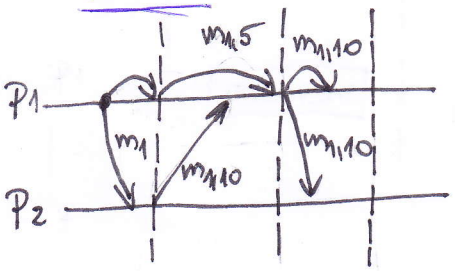
- linkové úrovně
- transportní úroveň (používá se v e-mailu)
  - kopie zpráv, zaskanglen poté jako unicast

- skupina ((příjemců)) < pevná (jednoduché)
  proměnná - nutná operace: join, leave



- P2 pošle zprávu všem (i sobě)
- ostatní zprávu potvrdí, opět všem (i sobě)
- i P2 musí provést potvrzení (a opět všem)
- zbytečně moc zpráv → těžko použitelný

ABCAST



- používá prioritní frontu
- pošlu zprávu, každý uzel ji ohodnotí prioritou
- původní vysílač vyhodnotí a pošle všem v jaké prioritě se bude vyhodnocovat

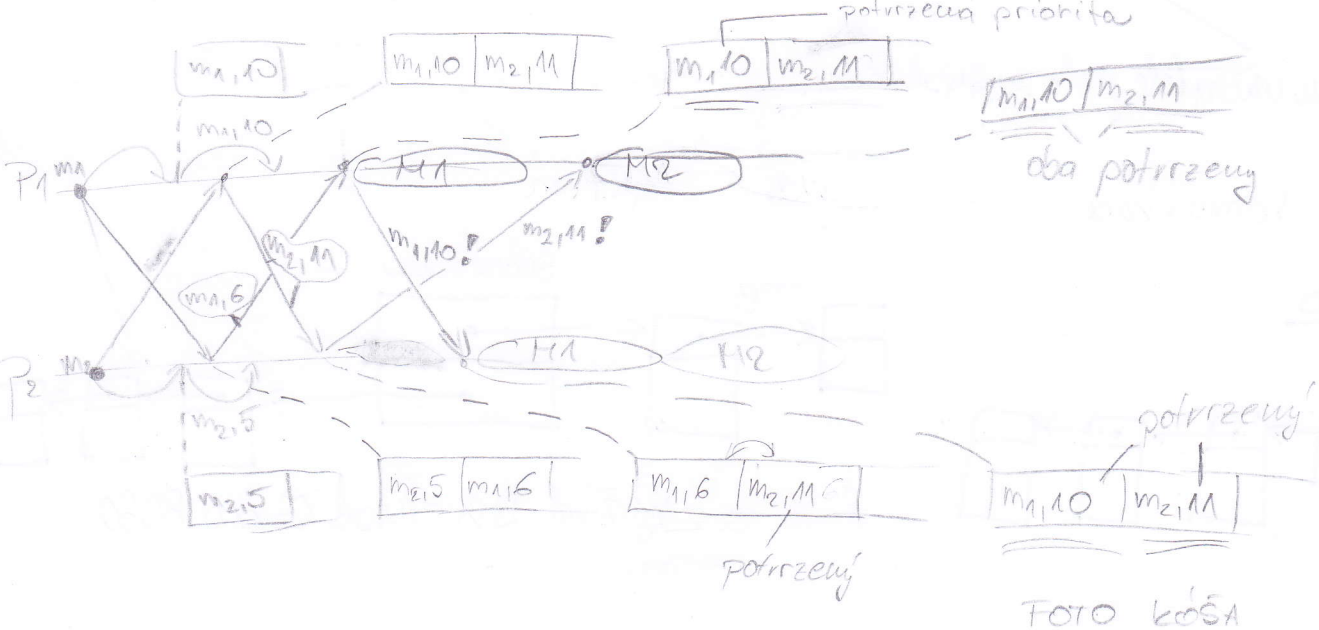
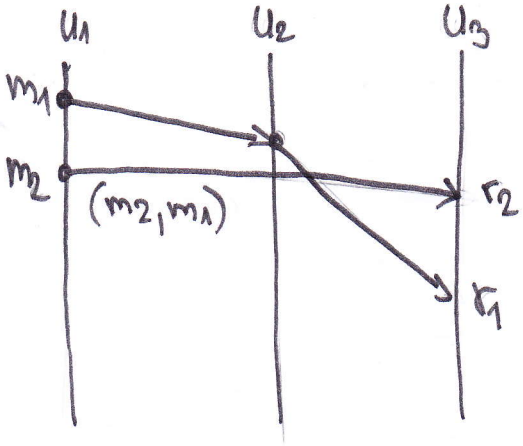


FOTO KÓŠA

CBCAST (podmíněný BCAST)



$m_1 \rightarrow m_2 \Rightarrow r_1 \rightarrow r_2$

– pro zajištění pořadí musím do zprávy očíslovat a přenášet historii

GBCAST

P1	P1	P2
P2	P3	P3

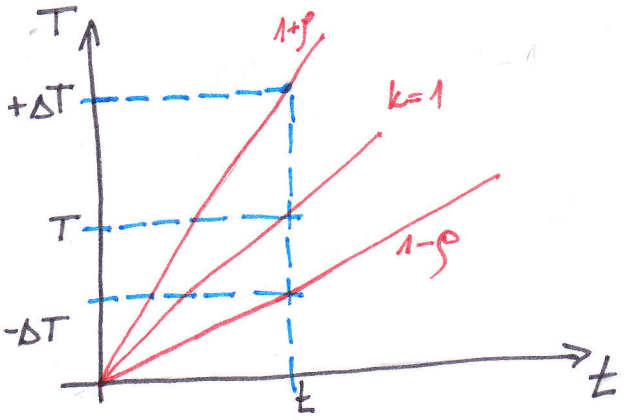
nebo

P1	P1	P1
P2	P2	P2

Synchronizace

– globální čas ∄

– synchronizace „hodin" – fyzické hodiny (nikdy nejdou přesně)
– logické hodiny



$$T = t$$

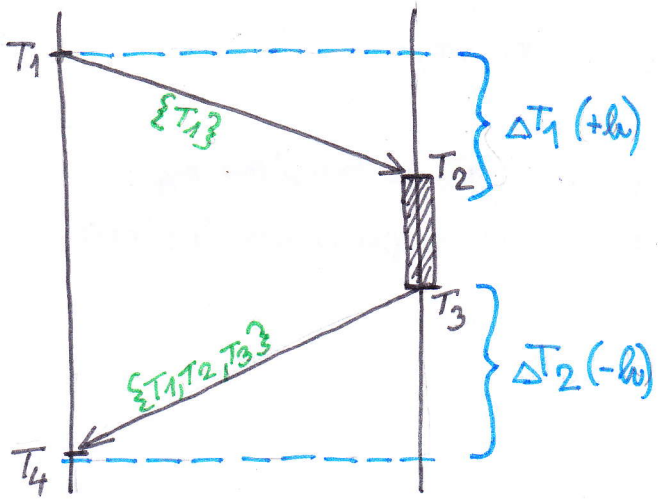
$$T + \Delta T = (1+\rho)\cdot t$$
$$T - \Delta T = (1-\rho)\cdot t$$

$$2\Delta T = 2\rho t$$
$$\Delta T = \rho t \Rightarrow t = \frac{\Delta T}{\rho}$$

– čas nelze měnit skokově (a už vůbec nelze se vracet k předchozím časům)

– čas nelze určit přesně, čas je $T \pm \Delta T$!

Christiánův alg. - synchronizace klienta a time servera (hodin)



$\Delta T_1 = T_2 - T_1$

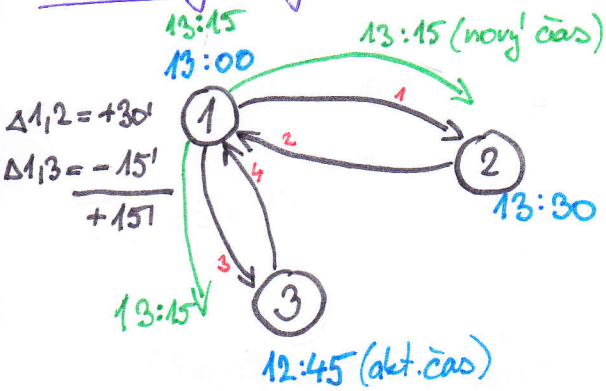
$\Delta T_2 = T_4 - T_3$

$\Delta T_1 + \Delta T_2 = (T_2 - T_1) + (T_4 - T_3)$

$2\Delta T = \text{—}''\text{—}$

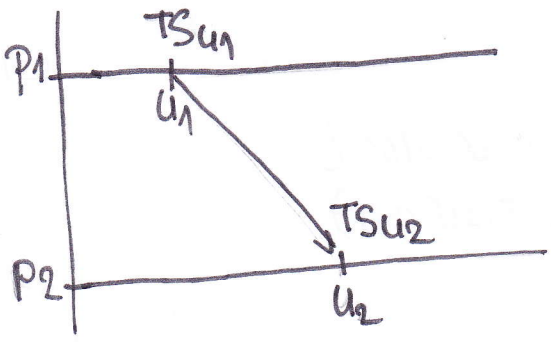
$\Delta T = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}$

Berkley alg. pro synchronizaci čas. serverů - nedbá přesný čas, ale stejný čas



Logické hodiny

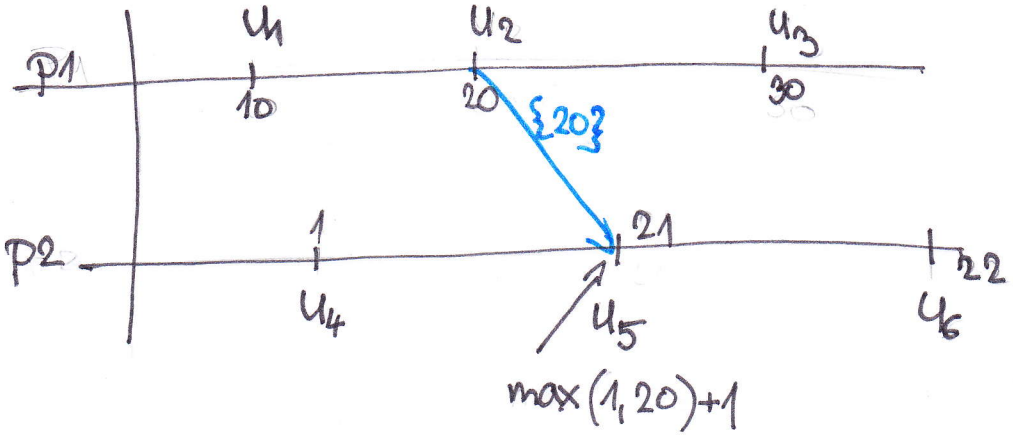
- vymyslel Lamport
- místo hodin jsou časové značky
- $u_1 \rightarrow u_2 \Rightarrow TS_{u1} < TS_{u2}$



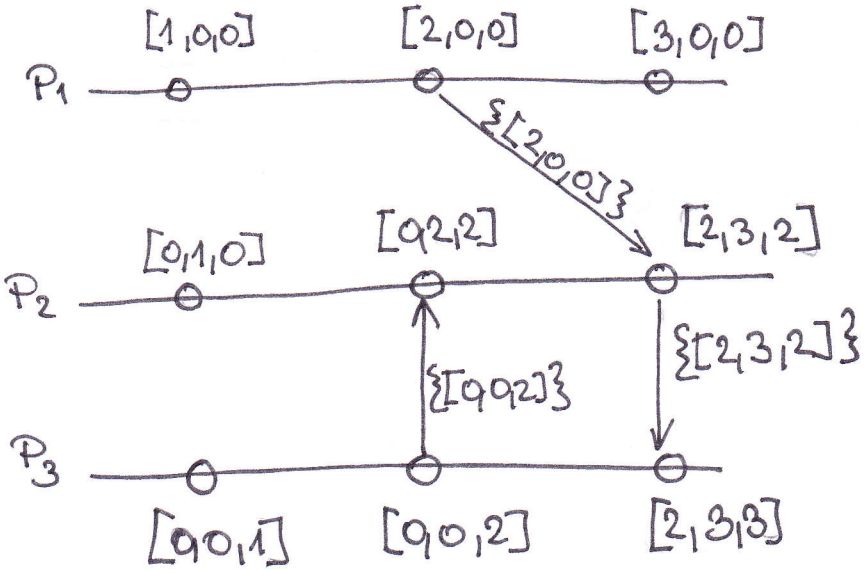
$u_1 \rightarrow u_2 \Rightarrow TS_{u1} < TS_{u2}$

poslání zprávy musí předcházet jejímu přijetí

5.10.15



Vektorové hodiny



$P_1 u_1$? $P_3 u_2$ → nemohu říct

$P_2 u_1$? $P_3 u_3$ → $P_2 u_1$ byla dřív

$u_1 \rightarrow u_2$

$\Downarrow$

$TS_{u_1} \leq TS_{u_2}$

- existují i maticové hodiny – pro synchronizaci souborů

DS

5.10.15

select()

počet descriptorů, které se mají prohlížet

n = select (N, r, w, err, time)

- r: maska, nast. 1 pokud na daném socketu chci číst data
- w: maska, nast. 1 pokud na daném socketu chci data přijímat
- time: NULL – neberu v úvahu; 0 – neblokující; >0 – max. doba čekání

n < 0 chyba

n = 0 vypršel čas

n > 0 počet událostí

TCP: read, accept – accept také vrátí událost

Komunikace

- klient/server TCP/UDP
- fork(), select()
- OOB data

UDP client/server

s = socket(...)
- do struktury dosadit IP adresu, port

bind(...) nebo connect(...)

sendto()
recvfrom()

close()

TCP ~~client~~ server

socket()
bind()
listen()
accept()
recv()
send()
close()

TCP client

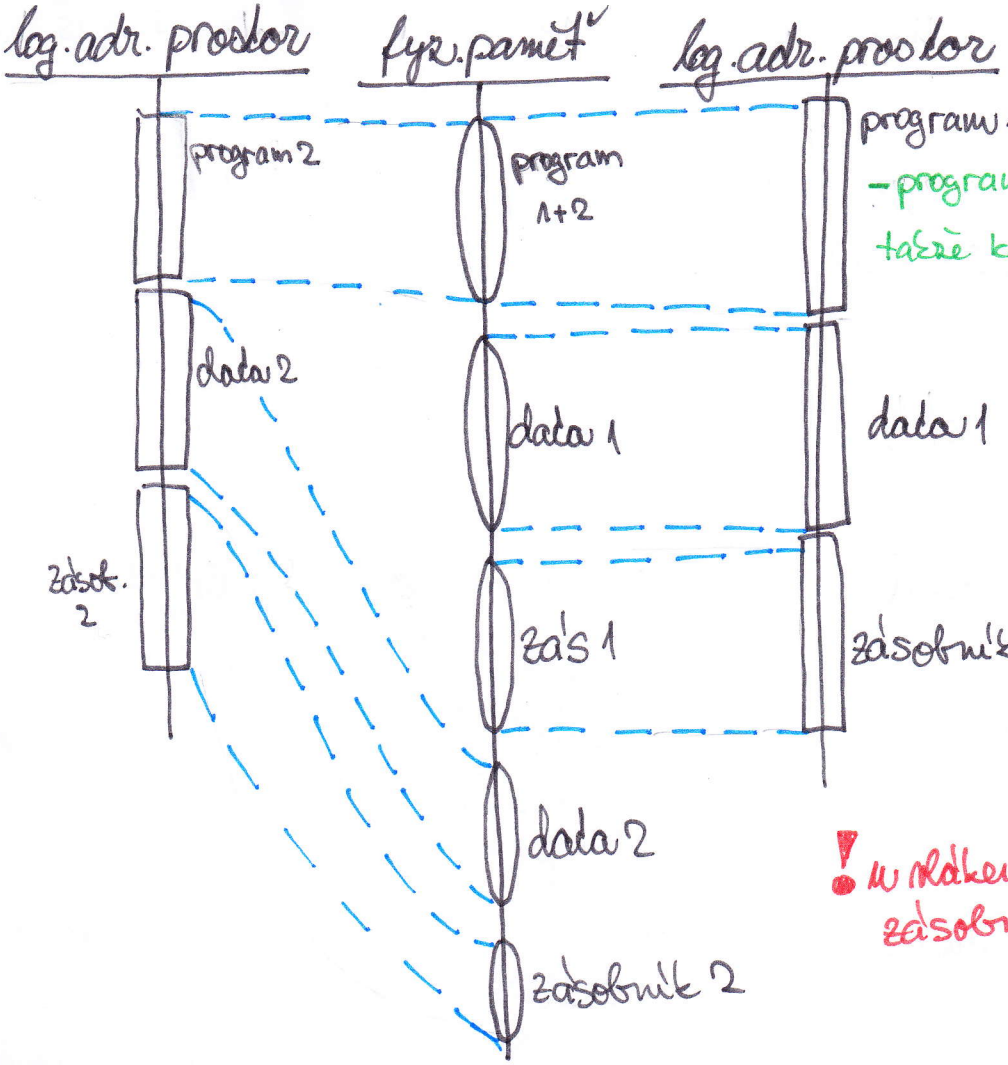
socket()

connect()
send(), write()
recv(), read()
close(), shutdown()

fork()

- rozdělí proces na 2, v potomkovi vrací 0, v rodiči PID potomka

```
s=socket(...)
:
k=accept(...)
PID = fork()
potomek | rodič
        |
        close(k);
readd(k)
write(k)
close(k), close(s)
exit(0);
```



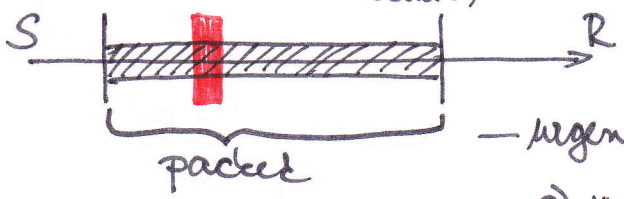
- program se pouze fortnul, takže kód je stále sdílený

! v rámci se mění pouze zásobník

OOB

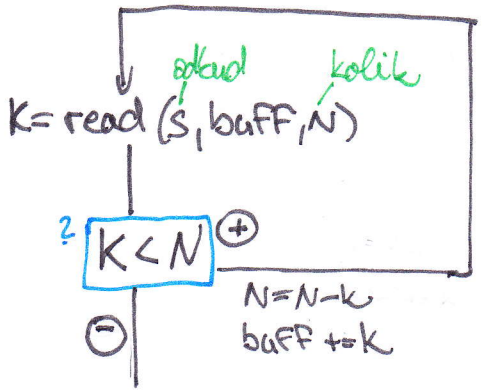
MSG_OOB = parametr u send()

MSG_PEEK = přečte data, ale neodstraní je z bufferu



- urgent data zpracovává na přednostní
  a) výjimka - musím hned zpracovat
  b) až na data narazím, dostanu signál, musím přepnout (IO_CTL) a zpracovat je

př. v Shellu se CTRL+C přenáší jako OOB

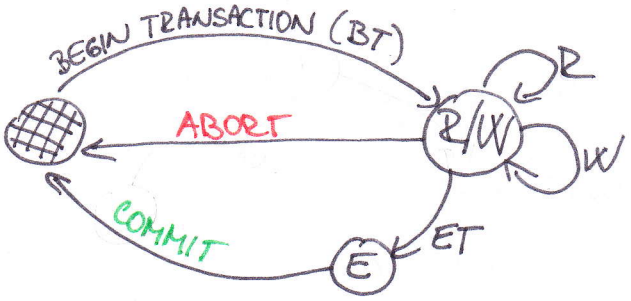


čtení hlavičky určité délky

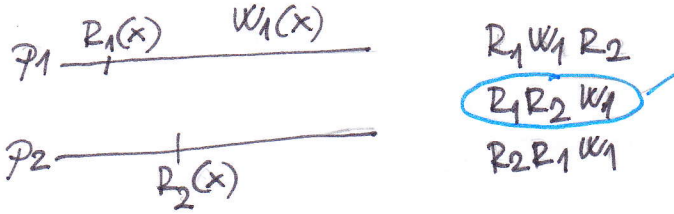
12.10.2015

Transakce

ACID
- Atomicity
- Consistency – nepřepisují si data pod rukou (tzv. špinavá data)
- Isolation
- trvalost
- nevratnost transakce (v.z. bankomat – nevratnost peněz které byly vyjmuty)



serializatelnost transakcí



Souběžné provádění transakcí

zámky
- pro čtení – sdílené
- pro zápis – výlučné

} zapisovat může 1, číst může N, zápis vylučuje čtení

časové značky

N-procesů

- alg. hlasování (decentralizovaný)

$N_W > \frac{N}{2}$ ... $\frac{N}{2}+1 \div N$ (potřebuje)

N_R ... $\frac{N}{2}$ ... 1

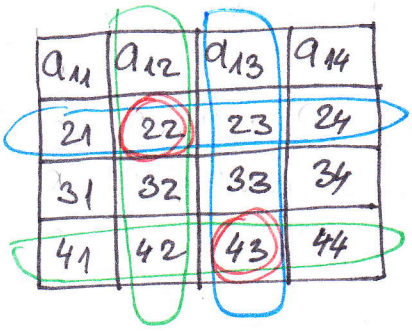
$N_R + N_W = N+1$

– max 1 zapisovat nebo 2 číst

- přidělování hlasů

P1	P2	P3	P4	P5
N	N	N	N	N
N+1	N+1	N+1		
		N+2	N+2	N+2

$N_W = 3$

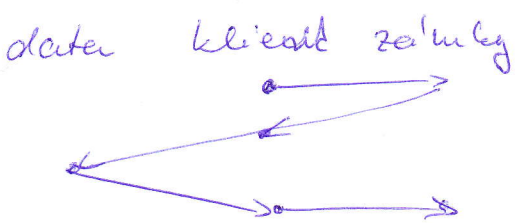


$N_W = 2\sqrt{N} - 1$

- výběr potřebných množin pro zápis
NESMÍ být disjunktní

Uzamykání

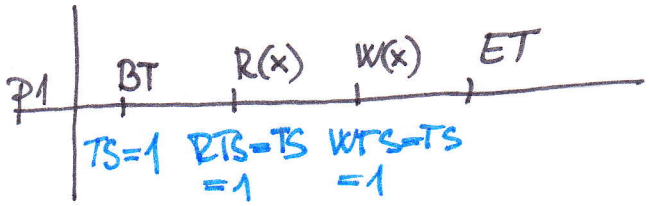
- centralizované × decentralizované

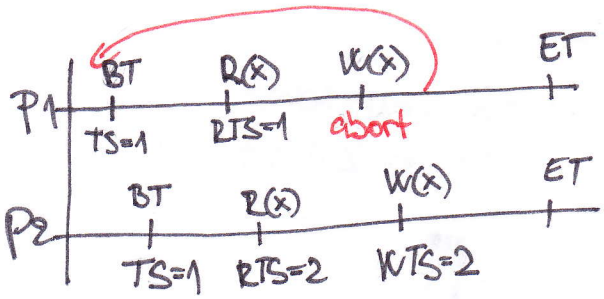


Časové značky (pro přístup k transakcím)

proměnné < RTS (Read Time Stamp) } čtu (RTS ← TS)
WTS } zapisují (WTS ← TS)

Transakce – TS

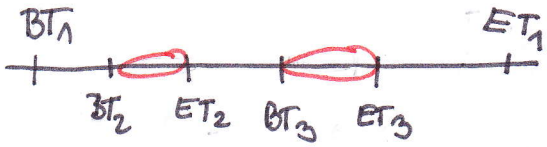


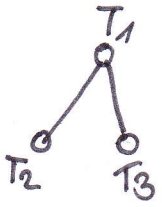


R(x): $WTS(x) \leq TS$

W(x): $WTS \leq TS \wedge RTS(x) \leq TS$

Vnořené transakce



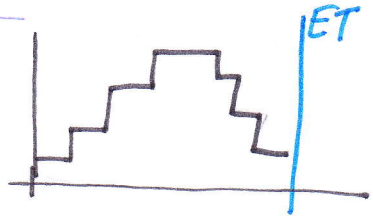


$\forall$ ok $\Rightarrow$ COMMIT

$\exists$ FAIL $\Rightarrow$ ABORT

Ukončení transakcí

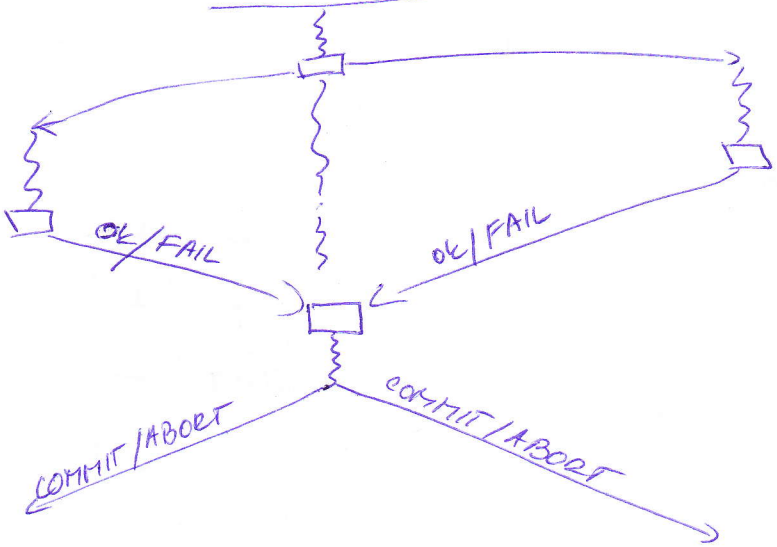
- uvolnění proměnných najednou



- uvolnění proměnných postupně

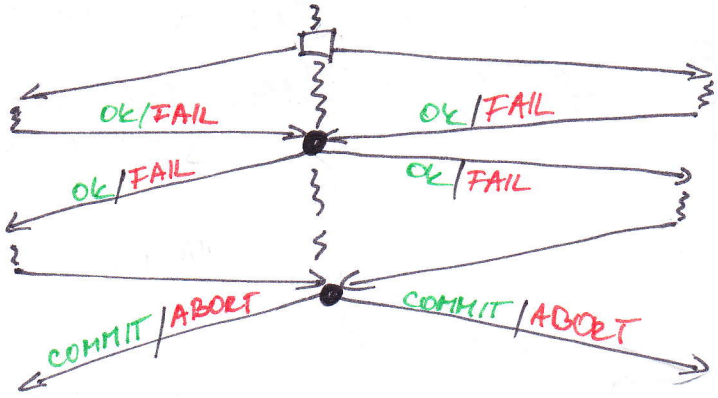
Distribuované transakce

klient koordinátor klient



3-fázové provedení transakce

klient koordinátor klient



Distribuované alg.

- souběh činností
  - soupeření (ethernet)
  - ponětření (token ring)
  - shoda — na hodnotě
    - na vektoru hodnot

Typy:
- vzájemné vyloučení (krit. sekce)
- výběr 1 z N – výběr koordinátora
- alg. shody (Byzantinští generálové)
- detekce ukončení
- obrázek

Semafory a jejich realizace

operace: P(), V()

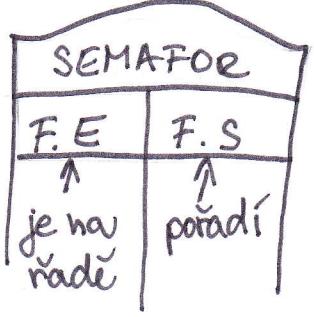
realizace — čekáním smyčkou (P)
- přerušením mezi procesory
- wait, signal – brouky

P	M
+	+
HALT	–
–	–

- realizace pomocí sequencerů a čítačů událostí
  - operace – advance (čítač události): E ← E+1
    - read (proměnná): vrací hodnotu proměnné
    - await (v, E): čeká pokud v ≤ E
    - ticket (čítač událostí): vrátí sequencer a potom atomicky provede: S ← E, E ← E+1

READ ADVANCE

semafor (F) – čítač událostí F.E
\ sequencer F.S



V(F): F.E = F.E+1 (advance(F.E))

P(F): r = ticket(F.S); await(r, F.E)

kritická sekce

F.E = 0; F.S = 0; ← inicializace

P: r = ticket(F.S) || r = 0; F.S = 1

while (r < F.E) r = = 0; F.S = = 1

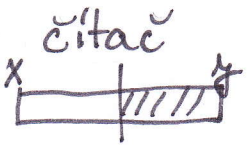
P: r = 1; F.S = 2 r = ticket(F.S)

r = = 1; F.E = = 1 while (r < F.E)

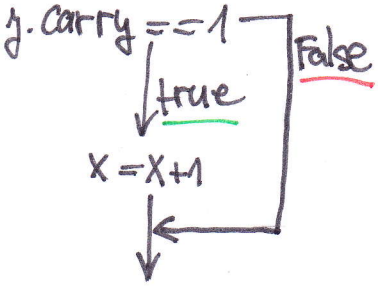
kolize při přístupu k proměnným

	R	W	I
R	−	o	+
W	o	+	+
I	+	+	−

⊖ neradí
⊕ vadí
o nevíme, většinou vadí



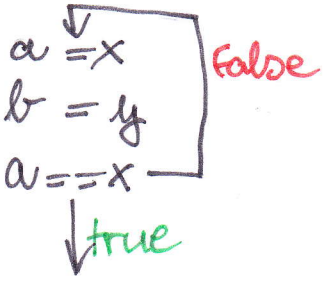
y = y + 1



procesor

b = y } může dojít k chybě ⇒ nutný zámek
a = x }

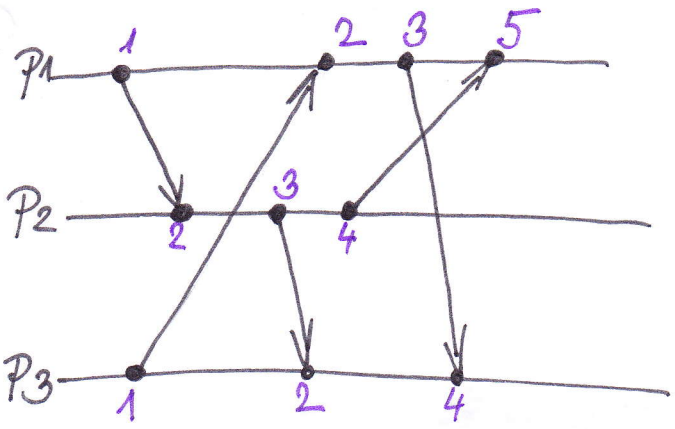
nebo



nepotřebuji zámek

DS cv.4. 12.10.15

časové značky

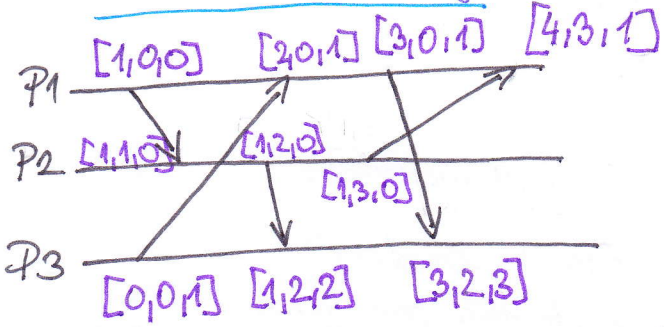


$u_1 < u_2 \Rightarrow TS_1 < TS_2$

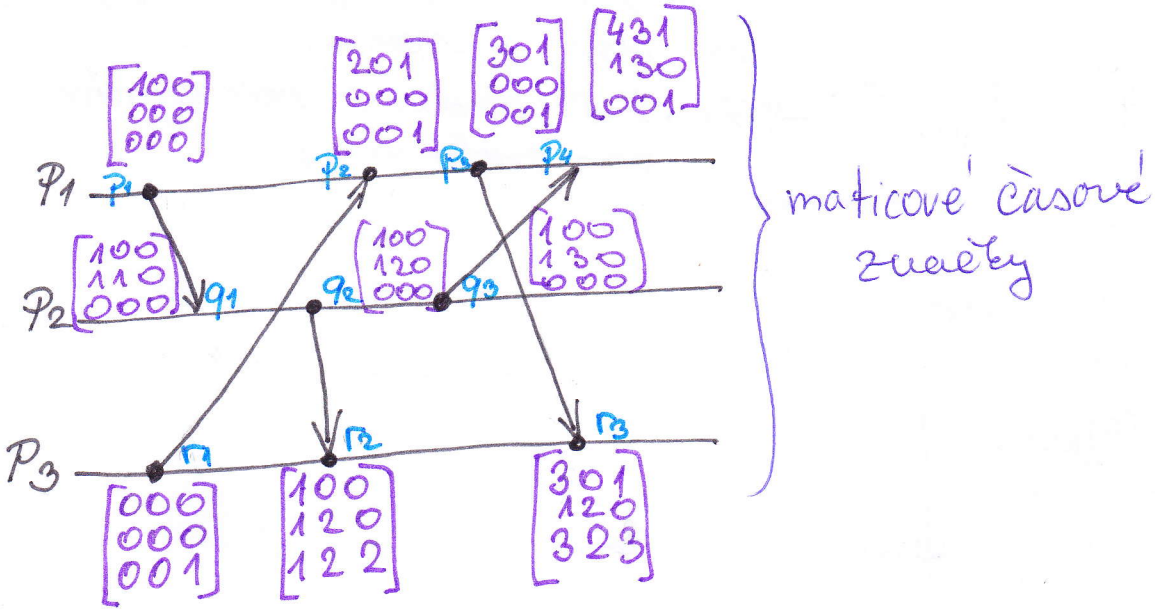
send < recv

$TS = \max\{send, u_{last}\}$

vektorové čas. značky



maticové časové značky

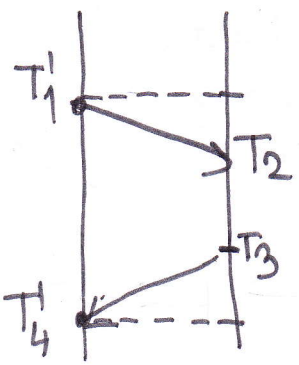


maticové časové značky

Christianův alg → viz. přednášky

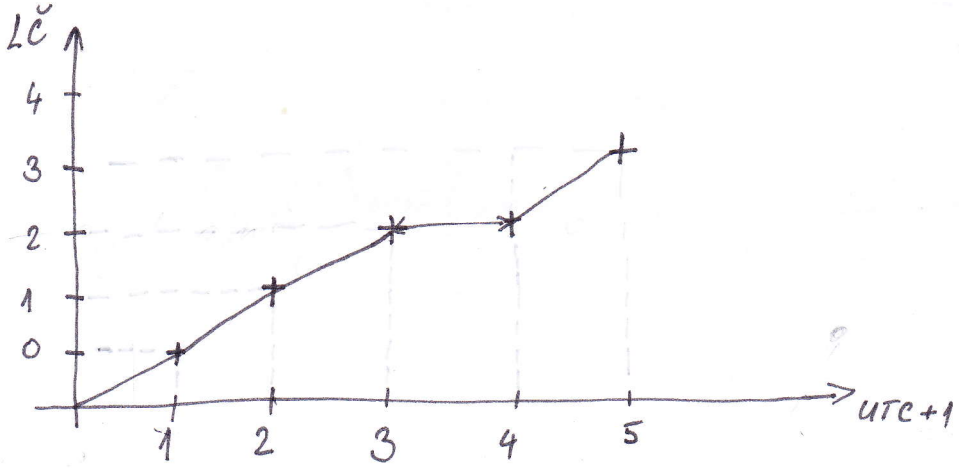
$$lw = \frac{(T_2 - T_1) - (T_4 - T_3)}{2}$$

$$T_1' = T_1 + lw$$



Berkley alg. synchronizace času

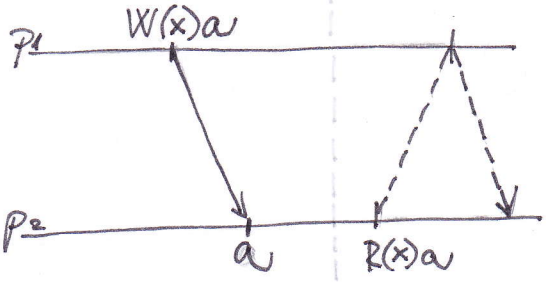
- graf změny letního a zimního času



19. 10. 15

Konzistentnost a replikace

Striktní konzistentnost



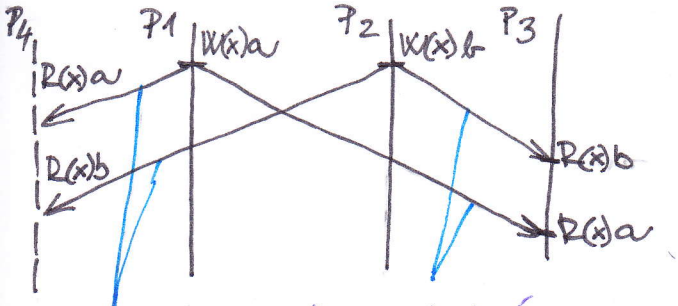
Závislé na implementaci:

- kdo čte a lokálně nebo se dotáže na jeho stav hlavního serveru
- W(x)a: kdo hodnotu v kopii opraví nebo zneplatní

- nelze realizovat distribuovaně

Sekvenční konzistentnost

- ve všech uzlech se operace provedou ve stejném pořadí



není sekvenčně konzistentní

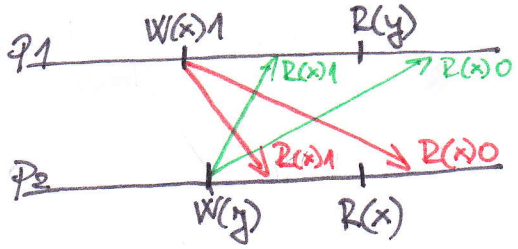
Kauzální (příčinná) konzistentnost

- v případě závislosti jedn. operací, jsou tyto provedeny ve správném pořadí (na všech uzlech)
- více viz slidy z přednášek

19.10.2015

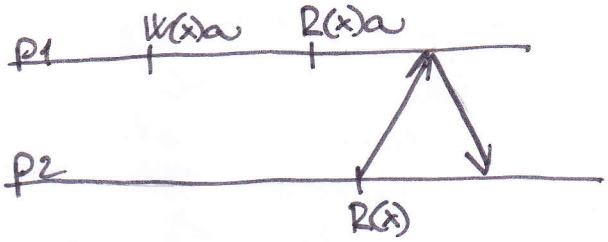
FIFO konzistentnost

- závislost operací ~~na~~ v jednotlivých uzlech

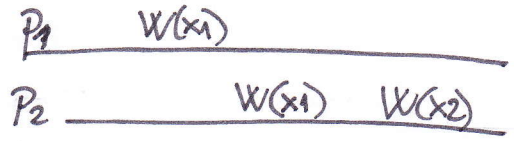


model (pro udržení konzist.)
- data centric (vše předesílé)
- client centric
  - možná (Eventual) konzistentnost
    - LDNS, web

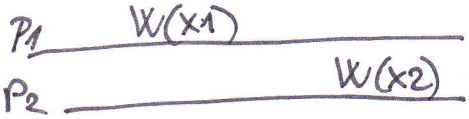
Monotónní čtení



Monotónní zápis

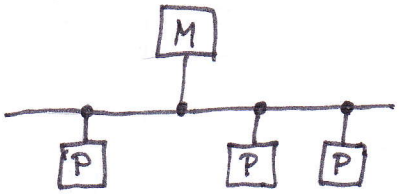


konzistentní



nekonzistentní

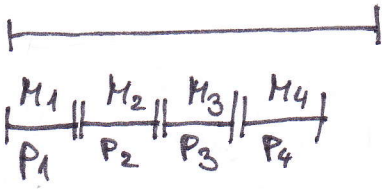
Sdílená paměť

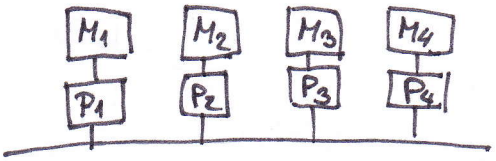


Distr. sdílená paměť

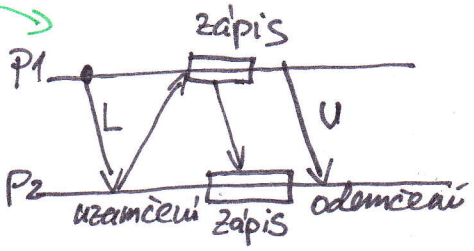
log. adr. prostor

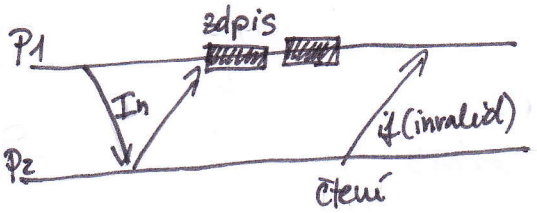
fyz. adr. prostor





sémantika oprav
- write-update
- write-invalidate (zneplatnění dat v ostatních kopiích)





vlastník dat = ten co poslední zapisoval
- → staticky (např. každý proces vlastní urč. oblast)
- → dynamicky
  1) statické rozdělení na začátku
  2) vlastník = poslední zapisovatel
     - → problém s nalezením aktuálního vlastníka (každá oblast musí mít identifikátor vlastníka)
     - → časem může jít o řetěz kdo je vlastník (P_1 myslí že P_2, $P_2 \rightarrow P_3$, P_3 je vlastník)

určení velikosti bloku dat
= objekt v byty → celá paměť
blok rozumné délky
(HW závislost, práva, atd.)

DS

Deadlock

19.10.2015

zdroje < stálé – paměť, procesor, periferie
dočasné – zprávy

podmínky nutné – výlučný přístup (⇒ virtualizace zařízení)
– postupné přidělení (⇒ všechny zdroje přiřadím naráz)
– nepreemptivní plánování (pouze proces, který zámek může odemknout)
– neomezené čekání (předchozí vedou k neomezenému čekání) ⇒ přidělování zdrojů podle priority (vzestupně)

→ stačí porušit jednu podmínku a deadlock nenastane

Odstranění deadlocku

a) detekce + uvolnění – detekční grafy (procesy + prostředky)
b) předcházení – Bankéřův algoritmus
c) prevence – lineární přidělování

ad a) tabulka přidělených prostředků tab. čekajících procesů

prostředek	proces
a	A
b	B
c	C

proces	prostředek
A	b
B	c
C	a

zacyklení

A B C
h(a) h(b) h(c)
h(b) h(c) h(a)
h(c) h(a) h(b)

Bankéřův algoritmus

X_i ... nároky
X ... jistina bankéře

$x_i \leq X, \forall i$
$\sum_i x_i > X$

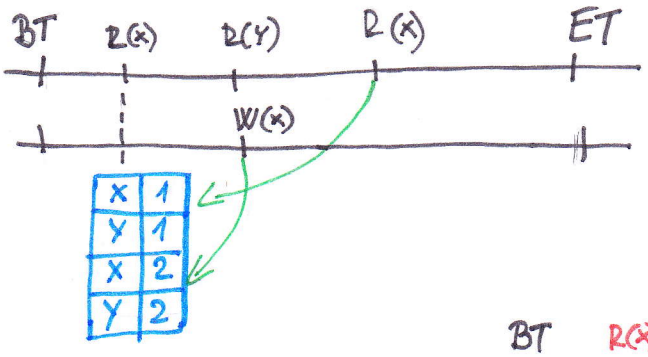
J	a(20)	B(12)	14(6)	19(1)
A	0(10)	B(2)		
B	0(8)		6(2)	
C	0(9)			5(4)
				X

Bankéř. alg. pro více zdrojů

	X	0(1)	1(0)	1(0)
	Y	0(1)	0(1)	1(0)
A	X	0(1)	1(0)	
	Y	0(1)		
B	X	0(1)		
	Y	0(1)		1(0)

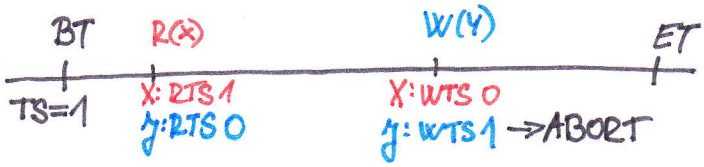
deadlock → k uvolnění potřebuje oba zdroje

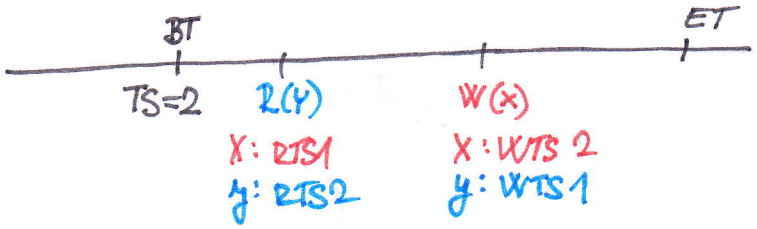
DS cv. 5 19.10.15



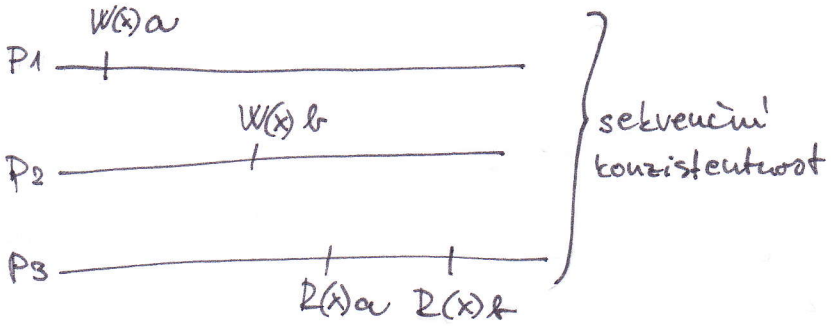
R: WTS ≤ TS

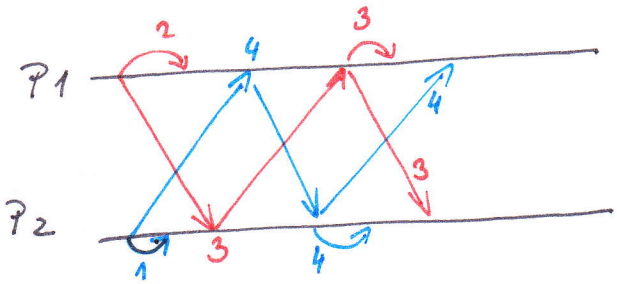
W: RTS ≤ TS ~~&~~ WTS ≤ TS





KONZISTENTNOST

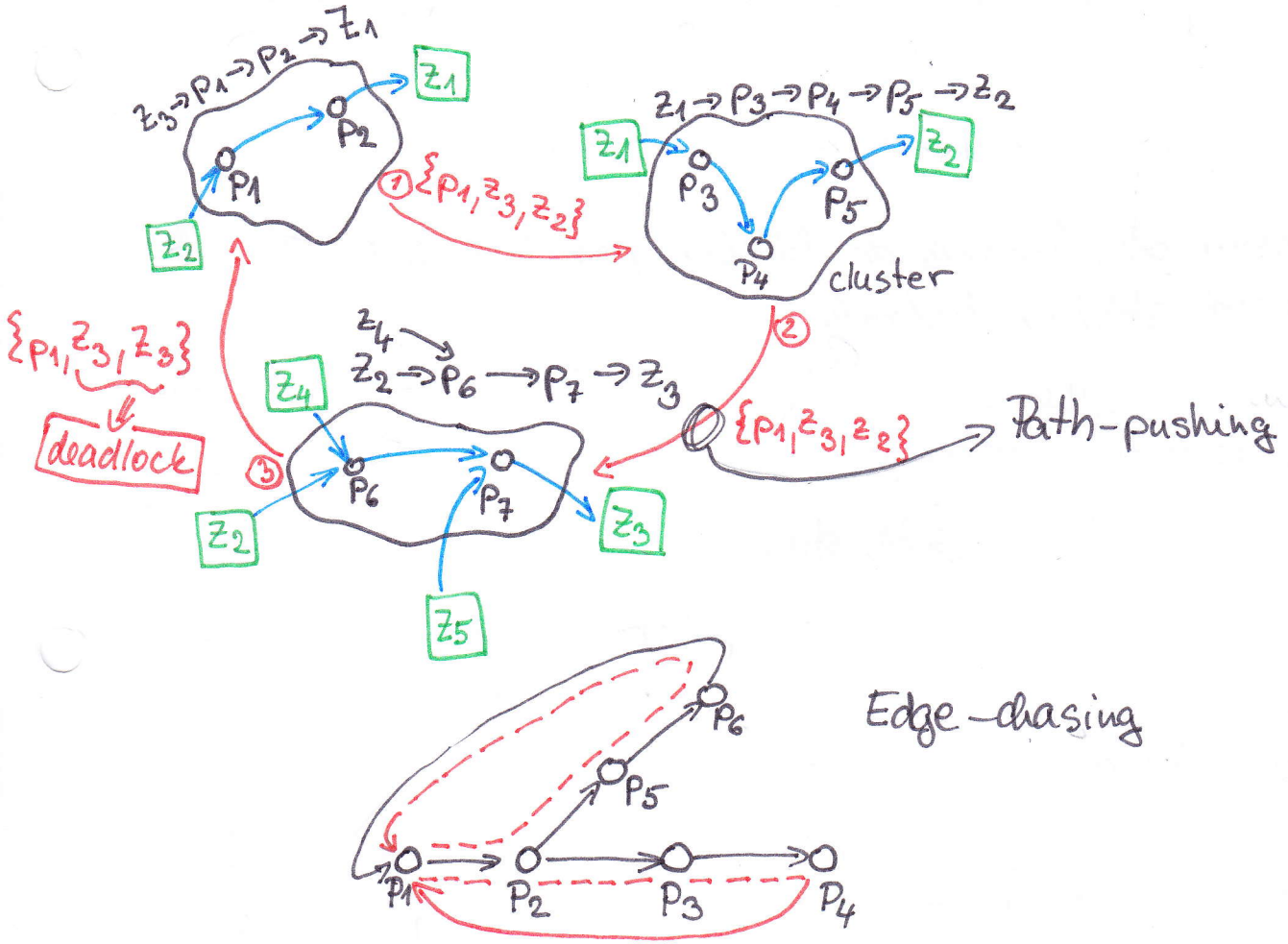




- zajištění sekvenční konzistentnosti

Deadlock v distr. systémech (AND/OR model)

2. 11. 2015



Edge-chasing

DISTRIBUOVANÉ ALGORITMY

- princip soupeření - velká režije
- princip předávání pověření (plánování)
  - centralizované x decentralizované
- dohoda (shoda) - na hodnotě nebo nezáporu hodnot
  odhalí i podvodníka

ad) sdílení zdrojů

- těsné vázání < spol. paměť
  - sběrnice
- volné vázání - není znalost globálního stavu

Fee Ticket (čítač událostí)

- zvedne hodnotu o 1 a vrátí starou (původní) hodnotu

- princip instrukcí „Testaad set", „compare and swap"

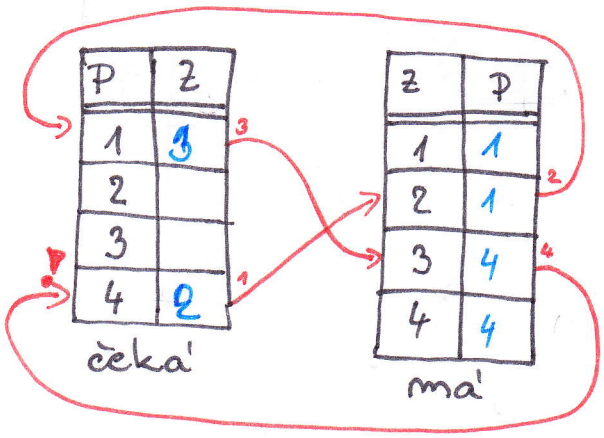
↳ uložení pův. hodnoty do lokální paměti a zároveň změna globální hodnoty

~~Distributed algorithms (rozšíření vyloučení)~~

Lamportův algoritmus (glob. star)

→ Alg. vzájemného vyloučení Ricard–Agrawala optimalizace počtu zpráv na $2(N-1)$

Maekawa – detekce/prevence uváznutí



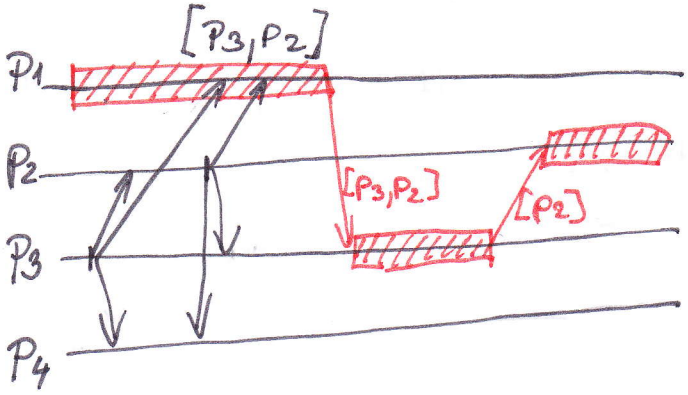
1	2
3	4

P_1 z_1
P_4 z_4
P_1 z_2
P_4 z_3
P_1 z_3
P_4 z_2

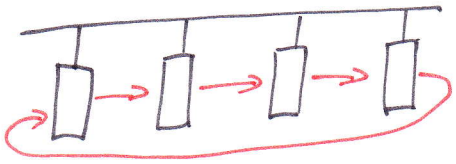
- prevence uváznutí hierarchickým žádáním

Algoritmy předávání pověření

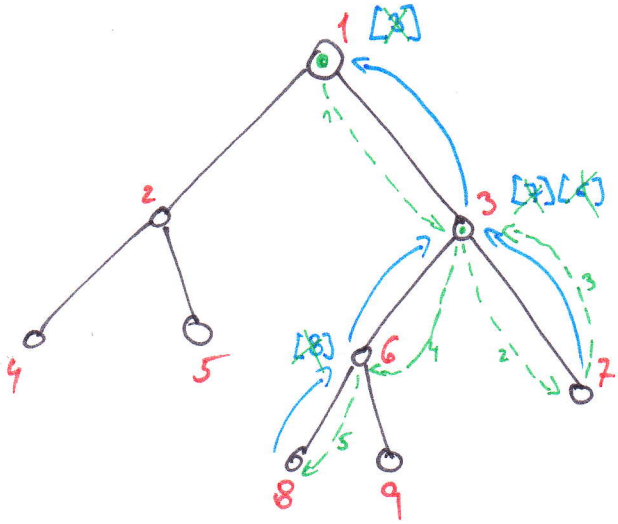
Suzuki a Kasami (broadcast)



LeLann (logický kruh)



Raymond – strom. struktura, rozšíření pro sdílení k identických zdrojů



a) statický kořen

→ možnost, že jedna větev bude dlouho bez pověření

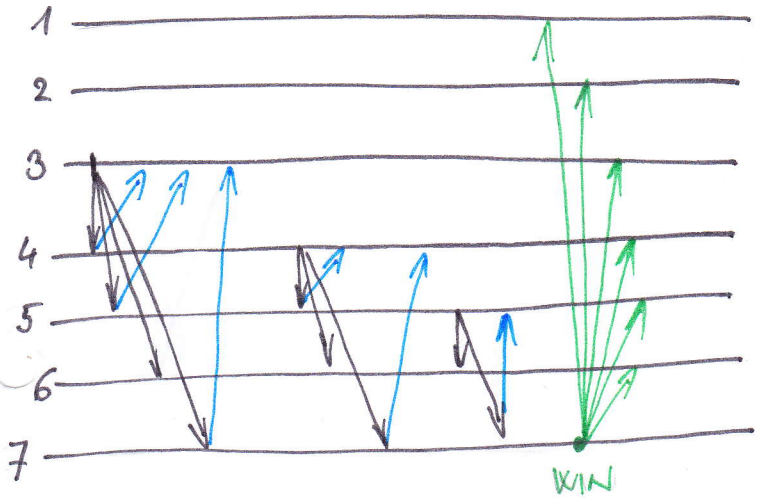
b) dynamická změna kořene

→ ~~jen~~ kdo má pověření je kořen směru

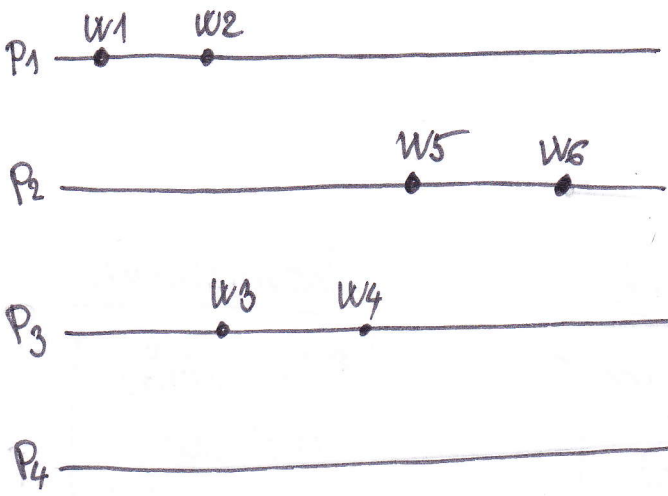
Alg. výběru 1 z N

Bully algoritmus

- vyhrává ten, kdo dostane nejmenší odpověď (ideálně 0)



FIFO konzistentnost – zápisy se provedou ve správném pořadí

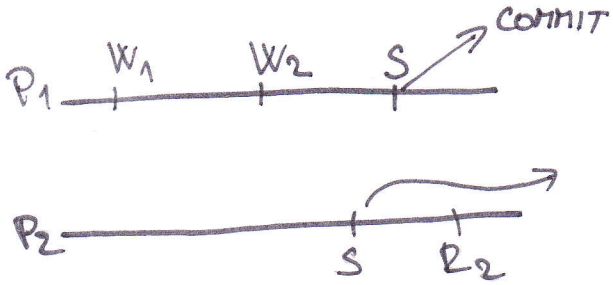


– musím dodržet pořadí zápisů na stejném uzlu

zajišťuje FIFO konzistentnost

Zpráva | Pnum | Wnum | udržuji pořadí

Slabá konzistentnost

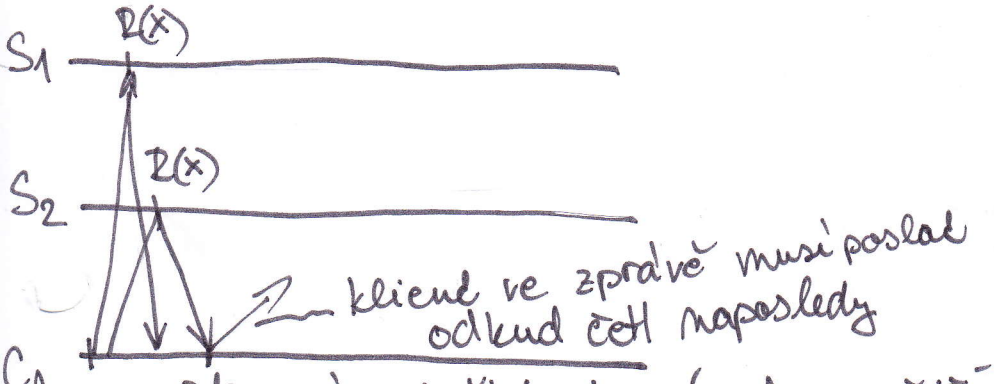


– od synchronizačního bodu musí být zapsané hodnoty přístupné všude

client centric × data centric

– každý klient kouká na data nezávisle (monotónní čtení)

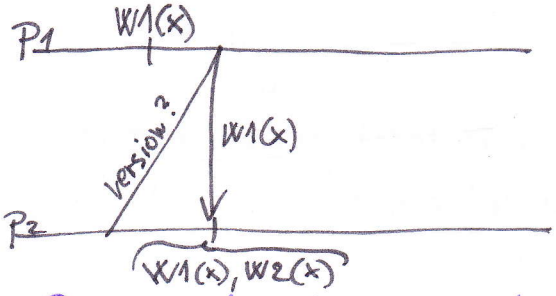
– pro každého klienta jsou data totožná v kterémkoli chvíli



klient ve zprávě musí poslat odkud četl naposledy

zde musím přečíst stejná nebo novější data než v minulém čtení!

monotónní zápis



Producent – konzument

sem_plno[0]
sem_volno[max]

Producent

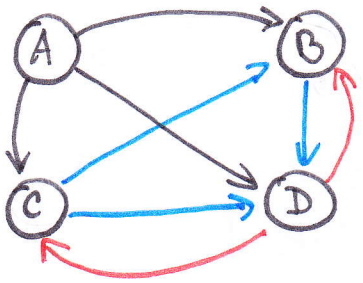
inicializace ()
P(sem_volno)
create ()
V(sem_plno)

Konzument

P(sem_plno)
destroy ()
V(sem_volno)

Byzantské chyby (iii) ← přenášk dat
uzel (k-chyb, N = 3k+1)
→ uzel si vymýšlí

Byzantinští generálové – shoda na vektoru hodnot
– nevýhoda je velké množství zpráv



① ② ③

	A
A	V
B	X
C	V

} chyba v B

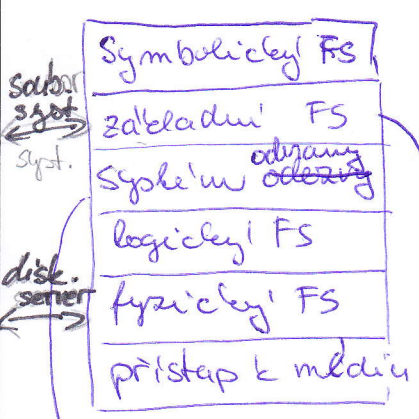
2-fázový a 3-fázový commit (viz. prezentace)

Distribuované souborové systémy

- transparentní přístup (nevím kde přesně je uložen, pracuji lokálně) NFS, ASF,
- netransparentní –"– (FTP, SCP, ...)

– síťové souborové systémy (souborový systém)
  – mapování do lok. souborového systému
  – problémy s přístupovými právy, problémy s chybami
– distr. souborové systémy
  – umísťování na různé úložiště
  – replikace
  – oddělení souborů a adresářů

Souborový systém z Multics



převod jména → id

jméno1, jméno2 → id

jméno → id1, id2

id → adresářové info: uložení, práva, délka

jméno	info, parametry, ...
Symbolický FS ←	→ základní FS

- matice přístupových práv (uživatel × soubor → práva)
- přístupový vektor (vlastník, skupina, ostatní [, systém]) → práva
- capabilities (vlastník generuje vektor, který obsahuje přístupová práva, je možné je předávat a omezovat)

logický FS

- obecné struktury → posloupnost slabik; alokační blok

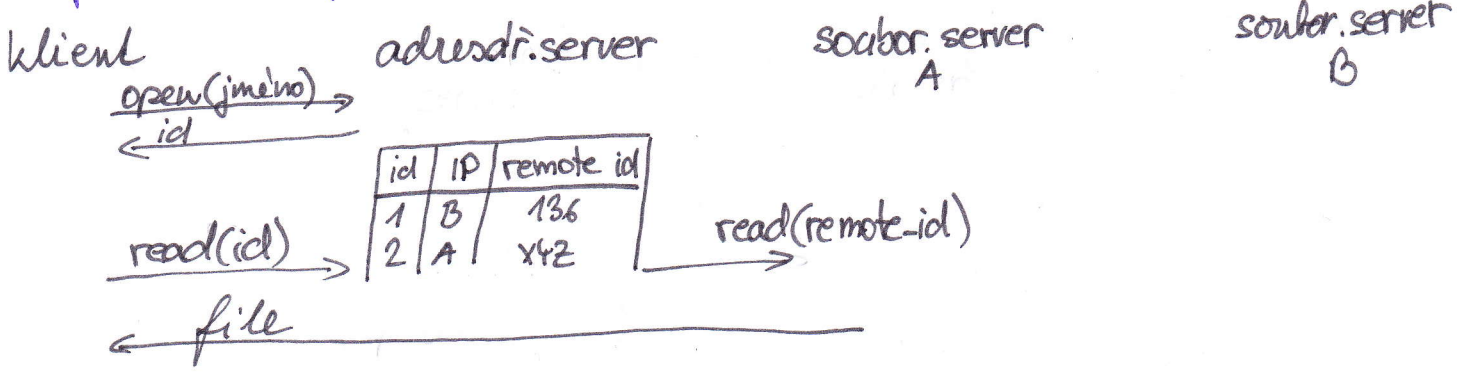
Fyzický FS

- převod log. bloku → fyzický blok

Přístup k médiu (Device Access)

- převod fyz. čísla bloku na skutečné umístění na nějakém médiu (sektor, stopa, cylindr)

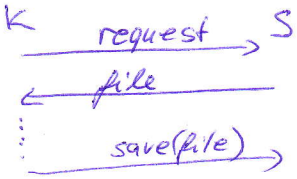
Transparentnost replikací



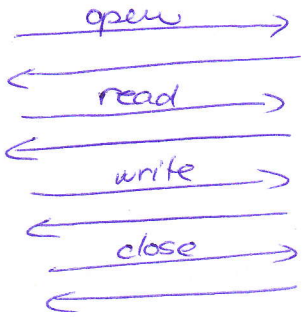
9.11.15

Typy služeb

model download/upload
- např. AFS



model vzdáleného přístupu
- soubor umístěn pouze na serveru
- např. NFS



Sémantika sdílení souborů

- sekvenční – poslední zapsaná hodnota (Unix)
- relační – změny viditelné až po uzavření souboru
- Immutable files – neměnitelné soubory → zápis znamená novou kopii souboru
- transakční –

NFS (Network File System)

- soubory
- identifikace uživatele (uid, gid)
  - NIS
  - NIS++
  - NYS
- používá UDP, postaven nad RPC

Protokoly

Mount protocol – map. exter. stromu ; server vrací „file handle"
NFS protocol – přístup k souborům
-

AFS (Andrew File System)

- vytvořen na Carnegie Mellon University
- používá model download/upload

DS FS

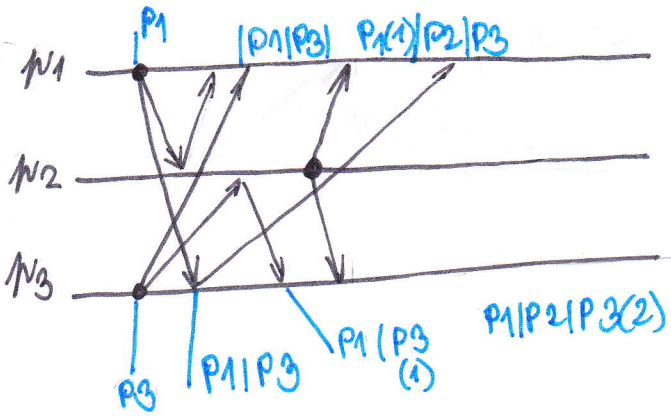
! NFS x AFS

- stavovost, vyrovnávací paměti, mapování (mount)
- replikace < write update
  write invalidate
- uložení dat včetně práv není úplně bezpečné

„Orange book" - TCB

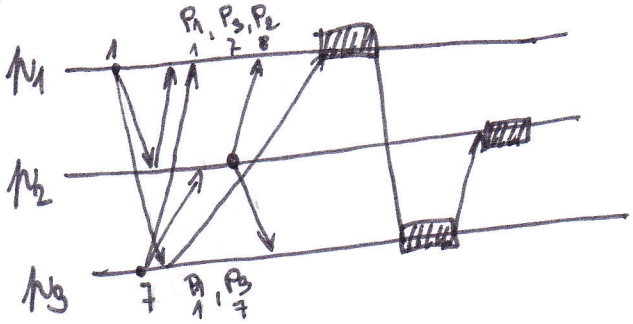
- klasifikace OS < A
  B
  C (Unix, Linux, Windows)
  D (nezabezpečené)

Alg. vzájemného vyloučení (Lamportův algoritmus)



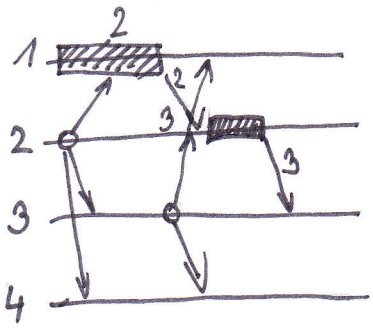
Časové značky

- co potvrdíme, nedáváme do fronty

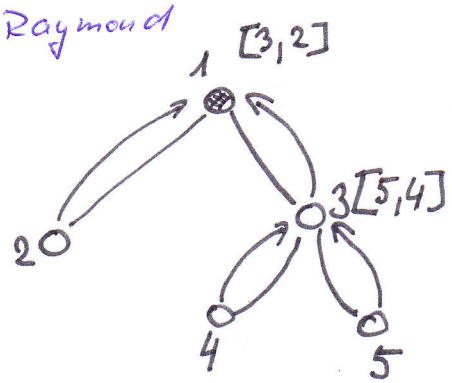


Alg. předávání pověření

- ~~broadcast~~



- Raymond



PORUCHY

- partial failure = částečná chyba (ex: výpadek DNS)
- error isolation = ost. komponenty nejsou zasaženy (izolace chyby) (ex: záložní DNS server)

synchronní — × — asynchronní

- do urč. doby bude reagovat
- chyba se snáze pozná
- těžko se pozná chyba na základě timeoutu (=žádný není)
- myslím, že nastala chyba, kontaktuji záložní server → může se stát, že nakonec odpoví oba

- availability (=dostupnost) - ppst, že je systém v daném datu funkční
- reliability (= spolehlivost) - ppst, že systém neselhal během dané doby
- safety (=bezpečnost) - když systém selže, specifikace se přizpůsobí a nic katastrof. se nestane

DEF:

error (chyba, omyl, odchylka) - část stavu sys., kt. může vést k poruše
fault (porucha, nedostatek)
failure (selhání)
erroneous state (chybový stav)

error → fault → failure

Hard faults × Soft faults

- permanent
- nutný zásah „admina"
- předchozí nebo dočasné
- cca 90% všech chyb

Pozn: slidy jsou dobrý zdroj informací

:

Shoda

- N procesů se dělejí dohodnout na hodnotě
  - 1 hodnota nebo vektor hodnot
- synchronní x asynchronní x failstop
- alg. shody → viz slidy
- interaktivní konzistentnost
- bizantinští generálové
- 2PC (dvou fázový potvrzovací protocol)
  - když vypadne coordinator je to v prdeli
  - → lépe 3PC (3-fázový)

Migrace kódu a procesů

Process = činnost spojená s realizací nějakého požadavku

potřebuje: - program (SW)
- paměť (HW)
- data (SW)
- procesorový čas (HW)

Migrace → důležitý je přenos stavu

- kód: předpokládáme, že poběží od začátku (⇒ snadné)
- proces: běží, musím migrovat celý kontext (vč. periferií)
- kód: možná komplikace v heterogenitě (jiné CPU, jiný OS, ...)

- migrace řešena často pomocí pseudokódu (bytekódu)

P2P

- přímý přenos mezi uzly
- rozdělení / sdílející obsah
  \ vyhledávající obsah
- sdílení / strukturované – musím alg. určit, kam to uložím
  \ nestrukturované – uložím kamkoli, ale musím mít algoritmus pro hledání (+ automatická replikace)
- jiné rozdělení / paralelní výpočty
  – sdílení obsahu, přístup k souborům
  \ Instand messaging
- podle decentralizace / čisté P2P – client i server (servent) bez centrálního serveru/směrovače
  \ hybridní – centr. server pro udrž. informací o členech
  kombinované – dvojí / server odpovídá na dotazy (indexy)
  členové udrž. informací (soubory)

nestrukturované	x	strukturované
- Napster		- Chord
- Gnutella		- Pastry
- Kazaa/FastTrack		

Strukturované P2P sítě

- overlay IPv4
- Realizace směrovacího algoritmu:

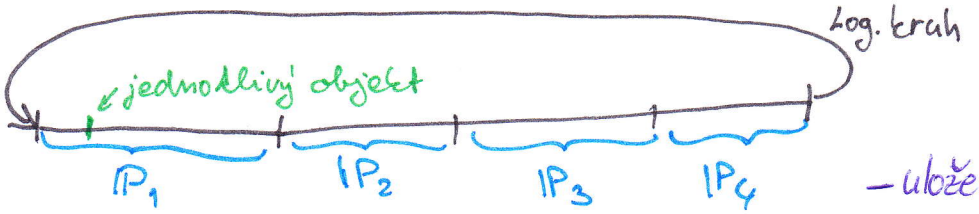
  IP adresa → vyhledání objektu

  ID objektu → IP adresa

  Pozn.: převod realizovaný směrovací tabulkou
- základní operace: lookup; get, put

Chord

- kruhová síť; mapování: objektů (jméno → ID z daného) ; počítačů (IP adr. → — " —)



- uloženo ve směrovacích tab.

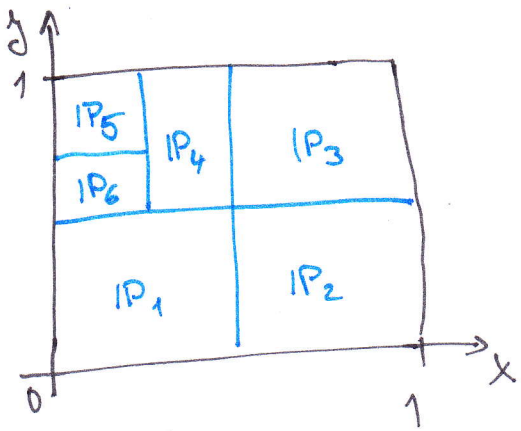
IP	HASH
:	:

IP → hash → rovnoměrné rozmístění

objekt → hash → ID objektu

CAN

- server spravuje všechny obj. ve svém prostoru
- pokud přijde nový stroj náhodně se určí souřadnice a stroj spravující danou oblast se rozdělí na dví



Toreut

- data (objekt) se stahuje z více zdrojů naráz

30.11.2015

CORBA

Common Object Request Broker Architecture

- obj. volání metod v síti
- OMG = Object Management Group – vývoj spec. CORBA
- definice CORBA schémata v IDL = Interface Definition Language
- ORB = Object Request Broker – prostředník
- IIOP = Internet Inter-ORB Protokol
  - komunikace mezi ORB a TCP/IP

Jini

- postavené nad RMI (Java)

Úvod do senzorických sítí

- lokalizace – trilaterace – zjištění polohy na základě intenzity sign.
- bezpečnost
  - šifrování
  - distribuce klíče
    - 1 pro 1
    - párové klíče (předem generované)
    - gen. velké množství klíčů a každý uzel dostane náhodnou podmnožinu
  - asymetr. šifrování
  - eliptické křivky
- čas. synchro.

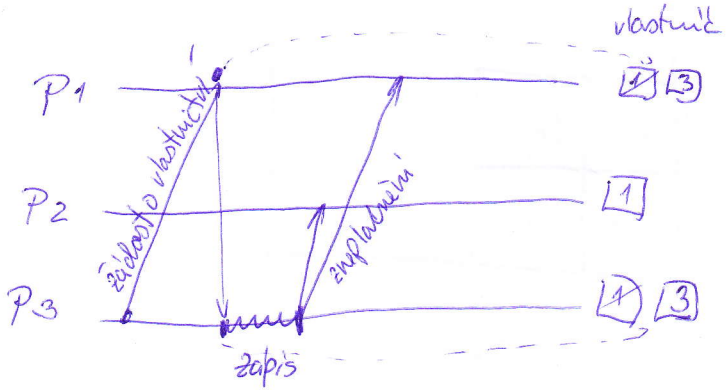
802.11
802.15.4
ultrazvuk, infračervené

Sdílená distribuovaná paměť

30. 11. 15

- dynamické vlastnictví
- přidávání pověření

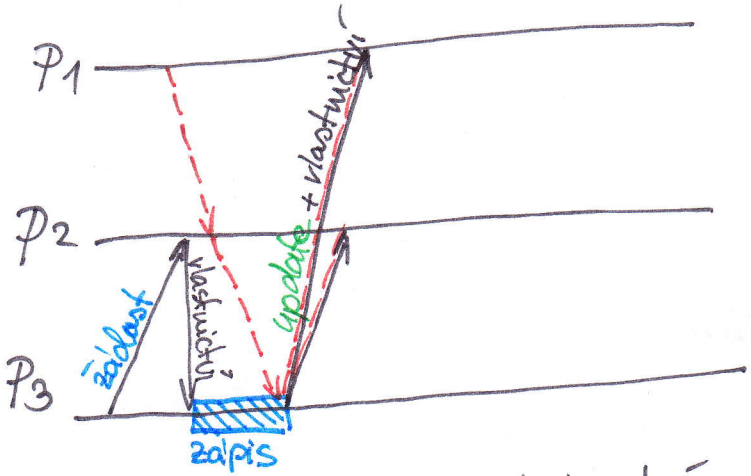
princip WRITE-INVALIDATE



realizace čtení, zápis a invalidace hardwarově

2 příznakové bity < povolen zápis / povoleno čtení > pokud oba na 0, tak je paměť invalidace

princip WRITE-UPDATE



- alg. vzájemného vyloučení
- pouze vlastník může zapisovat

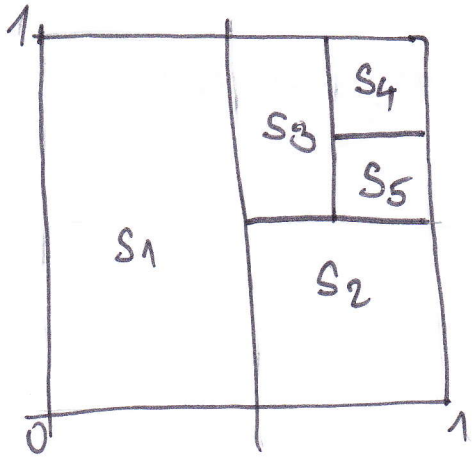
fr. požadavků na vlastnictví

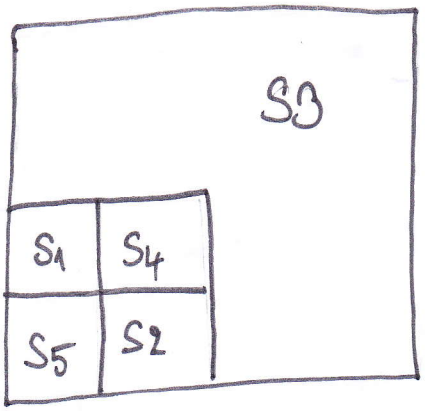
Migrace dat = nejprve musíme zaclylit stav, po migraci je potřeba v cílovém uzlu stav opět obnovit

stav = data, program, context + spojení (ale jak?)

P2P (Peer to peer)

CAN





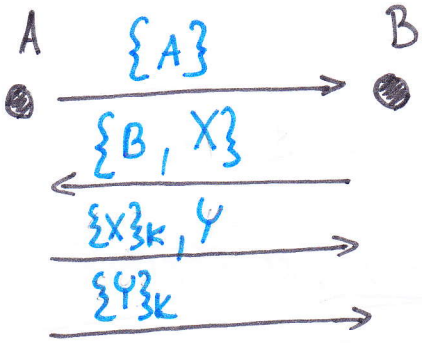
bezpečnost

ověření
- centr. server
- bez centr. serveru

message authentication code

certifikáty

přímé ověření – symetrická šifra



- bankéřův alg. (2 procesy, 2 zdroje)

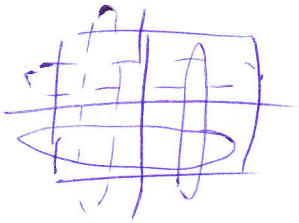
A	P
B	Q

- sdílení souborů v distr.

– idempotentní oprava souborů

musím hodnotu nastavit, nelze přičítávat

- alg. hlasování – Maekawa s vyloučením možnosti deadlocku u 4 procesů



vyloučení deadlocku:
– uzly očíslují a přiřazují vzestupně

- naznačte účelně jak se v chordu určí číslo uzlu na kt. se uloží soubor

- 2 transakce ← 1. READ(x), READ(y), W(y), W(x)
2. R(x), R(y)

– řízení přístupu pomocí časových značek

– je potřeba řešit? → linearizace transakcí
uzamykáním

- nakreslete závislost procesů a prostředků

vysvětlete kdy se jedná o OR deadlock a kdy o AND deadlock

-popište detekci:

- jak se liší fyzické hodiny od logických

jakým způ. zajistíme pomocí log. hodin ~~pořadí~~ posloup. událostí

- monolitic write consistence

monolitní zápis

- přidělování povětří v broadcastové síti

- problém migrace komunikačního kanálu

- ukládání a vyhledávání v P2P typu CAN

obojí stejně → na zákl. minimalizace vzdáleností

- vzájemné
- neprávě ~~symetrického~~ šifrování, pom. symetrického šifr. a výměna relačního klíče

KIV/DS
cvičení

7.12.2015

ŽK: spíš teoretické otázky

- Sémantiky RPC → reakce na chyby
  - při komunikaci
  - na serveru
  - volající skončí

  ⇒ právě jednou (začalo a končilo)
  ⇒ alespoň jednou (čekám na odpověď, může jich přijít více)

- idempotentní podprogramy - mohou se volat vícekrát a nezmění svůj stav

- př. dist. app: konkurenční zpracování, nezávislé chyb. režimy, komunikační zpoždění, bez existence glob. času
  → přístup k DB pomocí transakcí

  → ethernet

- sémantiky sdílení souborů v distrib. soub. systému
  - transakční sémantika
  - unixová sémantika (poslední vrací stav)
  - relační sémantika ξ

- systém údržby replik. ve stálé paměti, možnost výpadku repliky typu fail-stop (celá replika skončí), úplný uspořádání operací, ~~k~~ použité hlasovací alg.)

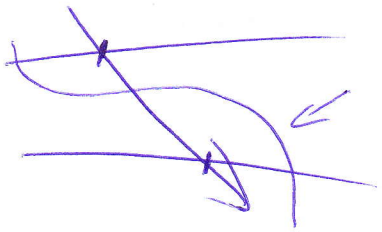
proces má k dispozici operace kopírané:

- lock (data)
- update (time, data, hodnota)
- read (time, data)
- unlock (data)

→ lze i metodou primární kopie

- ~~čtením~~ přečtu od všech hodnotu, ~~hlasy~~, uzamknou (u všech klasifikačních), provedu update a odemknu

- Christianův algoritmus
- problém odstraňování stromů v dist. soub. systému
- co je Oligarchy model a dvarchy model
  - viz přednášky
- schéma ověření účastníka se symetrickým šif. a ověřovacího serveru
- co je migrace stránek ?
  - když mígr. proces musím migrovat i paměť
- líná migrace
  - migrují až když je potřeba
- detekce falešného deadlocku
  - při centralizovaném řešení
- schéma přenosu zpráv pomocí AB cast → zajišťuje sekvenční uspořádání
- co je konzistentní řez ?
  - nekonzistentní



- realizace výrobce – spotřebitel
- monolitické čtení – pokud čte 2x tak dostane buď ty samé nebo novější data
- [illegible]