ZÁPADOČESKÁ UNIVERZITA
KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY

# Semestrální práce z KIV/DS

## Simulace Chandy-Lamportova algoritmu

**Martin Sloup - A08N0111P - msloup@students.zcu.cz**

**7.11.2010**

# Zadání

## Případ užití, motivace

Banka má několik poboček. Množství hotovosti banky je konstantní, ale jednotlivé pobočky mezi sebou převáží různé obnosy v pomalých nákladních autech na definovaných trasách. Úkolem je zjistit, jaký je stav na pobočkách a kolik peněz je na cestě. To vše bez narušení/zablokování provozu poboček.

## Požadavky

- Implementace v C/C++.
- Překlad programem make.
- Žádné GUI, jen příkazový řádek, neinteraktivní - spustit, spočítat, zalogovat.
- OS Linux.
- Paralelizace volitelně vlákna nebo procesy (dále v textu jen vlákna).
- Množství poboček konfigurovatelné.
- Orientovaný graf bude plně obousměrný (každá pobočka může posílat peníze všem ostatním).
- Komunikace bude realizována protokolem TCP/IP, sokety výstupních kanálů budou NEblokující.
- Pobočky budou v náhodných intervalech na náhodných výstupech odesílat náhodné množství hotovosti.
- Pobočky, které mohou zahájit snímkování, budou v náhodných intervalech spouštět snímkování.
- Vygenerujte jakýmkoliv způsobem datový soubor pro net_flow_vizu
- Pro simulaci zpoždění komunikačních linek využijte socket_retarder.
- Časová zpoždění volte tak, aby byla vizualizace dostatečně názorná.
- Příjem a vysílání zpráv musí být vhodně paralelizované, neměly by se vzájemně blokovat.
- Dokumentace jen stručně v bodech, použité algoritmy, mechanizmy,…

# Řešení

Implementace Chandy-Lamportova algoritmu byla vcelku jednoduchá. Jediný menší problém činilo vytvoření komunikačních linek při spuštění poboček v různý časový okamžik. To nakonec bylo vyřešenou použitím bariéry pomocí pozastavení programu, dokud uživatel nezmáčkne klávesu enter.

Problém je totiž v tom, že každá pobočka může být nebo do konce i je závislá na ostatních pobočkách. Při vytvoření kanálu mezi pobočkami musí totiž již v době připojování všechny pobočky naslouchat. Pokud bychom spouštěli pobočky postupně, nemuseli by se dříve spuštěné pobočky připojit na pobočky spuštěné později.

Tedy po spuštění programu nejprve každá pobočka započne naslouchání na zadaném portu. Zde nastane první bariera, kdy uživatel musí spustit všechny pobočky a následně u všech potvrdit klávesou enter, že jsou všechny pobočky spuštěny a naslouchají. Po zmáčknutí této klávesy dojde k vytvoření všech komunikačních kanálů (hran) – uvažujeme úplný graf propojení mezi pobočkami. Zde se nachází druhá bariéra, protože pro započnutí komunikace (převody peněz, zjišťování globálního stavu) mezi

pobočkami musí být vytvořeny všechny komunikační linky. Po potvrzení klávesou enter začne probíhat provoz mezi pobočkami a není potřeba další interaktivity uživatele s programem.

Na obrazovce jsou pro přehlednost zobrazeny příchozí požadavky a výsledky stavů účtů jednotlivých poboček po dokončení procesu zjištění globálního stavu na jednotlivých pobočkách.

Pobočka vždy po dokončení zjištění globálního stavu předává jiné, náhodně zvolené, pobočce právo na zahájení procesu zjištění globálního stavu. Taktéž, pokud každá pobočka dokončí výpočet svého globálního stravu, odesílá tuto informaci pobočce, která započala zjišťování globálního stavu.

Jednotlivé pobočky se spouští pomocí již předpřipravených souborů branch1.sh až branch3.sh. Tyto BASHové obsahují předpřipravené hodnoty a zároveň nastavují soket retardér. Všechny pobočky používají společný konfigurační soubor `clients.conf` pro vytvoření komunikačních kanálů. Pro tři pobočky může být obsah souboru následující:

```
1 localhost 2001
2 localhost 2002
3 localhost 2000
```

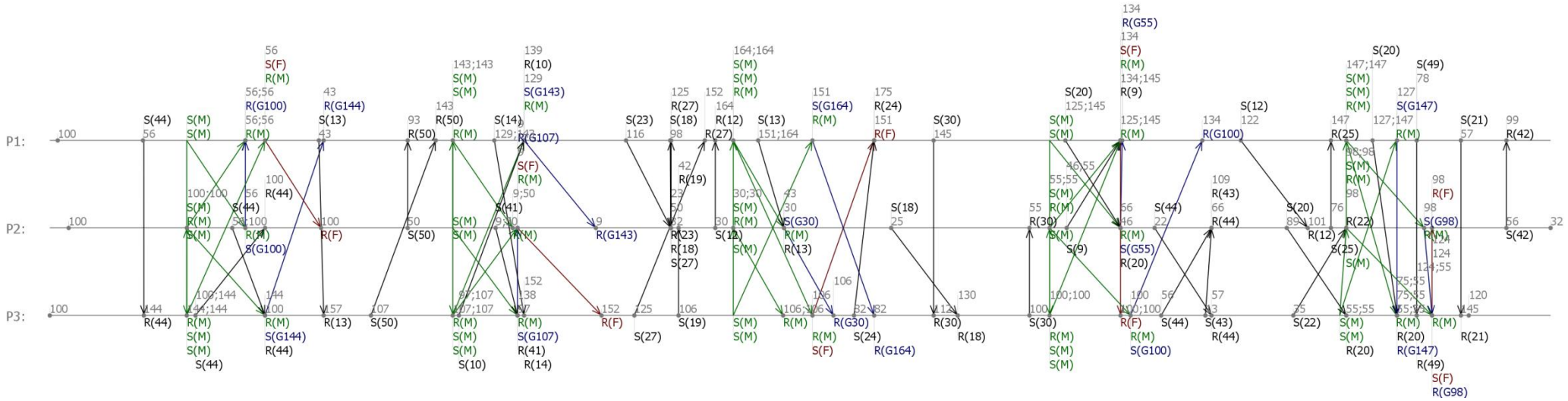
První sloupec udává, pro kterou pobočku platí řádek, a zbytek řádku obsahuje jméno počítače a port, kde naslouchá další pobočka, tedy kam se bude pobočka připojovat. Protože je vytvořené spojení obousměrné, stačí nadefinovat počáteční připojení pouze jedním směrem.

Z každé pobočky jsou vygenerovány dva soubory pro vizualizaci pomocí `net_flow_wizu`. Jde o `Px.log`, ve kterém se nachází příchozí komunikace na pobočku a `Mx.log` (písmeno `x` zde označuje číslo pobočky), kde se nalézá stav účtu pobočky po zpracování příchozího požadavku. Pro tři pobočky budeme mít tedy 6 souborů. Těchto šest souborů složíme do jednoho `flow.log` souboru použitím předpřipraveného BASHového skriptu `combine.sh`. Takto získaný `flow.log` soubor lze použít jako vstup `net_flow_wizu` pro vygenerování diagramu komunikace. Jeden takový diagram se nachází v příloze.

## Závěr

Dle zadání se povedlo vytvořit aplikaci simulující Chandy-Lamportův algoritmus. Správnost algoritmu byla ověřena několika hodinovým během simulace s použitím soket retardéru. Velmi oceňuju možnosti jazyka C++, který zde byl oporou při použití tzv. šablon pro uchovávání potřebných hodnot (např. vector, map). Bez nich by mi přišlo naprogramování této aplikace velmi složité.

# Příloha – diagram komunikace




Legenda:

**S(44) / R(44)** – odeslání / příjem 44 korun z pobočky / na pobočku
**S(M) / R(M)** – odeslání / příjem marker pro výpočet globálního stavu
**S(G100) / R(G100)** – odeslání / příjem globálního stavu (globální stav se vždy odesílá na pobočku, která započala výpočet)
**S(F) / R(F)** – Odeslání / příjem práva na započetí výpočtu globálního stavu
**55** – lokální stav (55 korun) účtu pobočky
**134;145** – lokální (134 korun) a globální (145 korun) stav účtu pobočky