Dosažení shody v distribuovaném prostředí

PA 150 ◇ Principy operačních systémů

Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze : podzim 2020

Definice problému

- Procesy kooperující v rámci řešení jisté aplikace v rámci definované skupiny si vyměňují informace s cílem se vzájemně dohodnout a nakonec dosáhnout společného názoru (dohody, shody) dříve než spustí nějakou akci specifickou pro řešenou aplikaci.
  - ✓ Poznáme později *commit* protokol pro rozhodování zda se má hotová distribuovaná transakce potvrdit nebo zrušit
- Dosažení shody je determinováno
  - spolehlivostí komunikačního systému a
  - spolehlivostí kooperujících procesů
- Procesy ve skupině se dohadují rozesíláním (*multicast*) zpráv, typicky každý proces může komunikovat s každým procesem ve spolupracující skupině, v DS

Typy distribuovaného prostředí (DS) podle běhu času

- asynchronní
  - ✓ předem se neznají rychlosti běhu procesů, doby zpoždění přenosů zpráv, drifty reálných hodin uzlů. Kooperace procesů se musí řešit algoritmy na bázi logického času hnaného výměnou zpráv mezi procesy
- pseudoasynchronní (Internet)
  - ✓ asynchronní prostředí s možností detekce nesplnění časového limitu doručení zprávy
- synchronní
  - ✓ znají se horní a dolní meze rychlostí běhu procesů, dob zpoždění přenosů zpráv, driftů reálných hodin v uzlech, …
  - ✓ Výměna zpráv a lokální běhy procesů se provádějí v periodicky vymezovaných intervalech

Problém dosažení shody a synchronnost / asynchronnost DS

- Algoritmy pro dosažení shody známe pro synchronní DS
- V asynchronních DS nelze dosažení shody garantovat žádným algoritmem
  - ✓ Nepozná se vypadlý proces od velmi velmi pomalého procesu
  - ✓ Nelze garantovat neznamená, že shodu nelze nikdy dosáhnout, platí: lze dosáhnout dosažení shody pouze s jistou pravděpodobností
- Místo asynchronního systému lze použít částečně (pseudo) asynchronní systém implementovaný např.
  - ✓ detekcí poruch – např. pomocí časových limitů pro získání zpráv a definovaných implicitních hodnot při nezískání zprávy

Typy distribuovaného prostředí podle spolehlivosti

- systémy bez poruch
- systémy s nezhoubnými(beningními, *fail-stop*) poruchami
  - ✓ poruchy lze vesměs ošetřit protokolárně, detekovat, . . .
  - ✓ systémy s výpadky procesů/procesorů, výpadek je trvalý, komponenta DS nelže, buďto funguje nebo je nečinná
  - ✓ systémy s výpadky komunikací, ztráty zpráv např. lze detekovat časovými limity a dodávat smluvenou implicitní hodnotu
- systémy s Byzantskými poruchami – nejhorší možné chyby, komponenty DS mohou selhat i lhát
  - ✓ Byzantská porucha (*fault*)

    Každá chyba prezentující se různým pozorovatelům různými příznaky
  - ✓ Byzantská selhání (*failure*)

    Ztráta služby systému kvůli byzantské poruše v systémech, které vyžadují dosažení shody mezi jejich komponentami

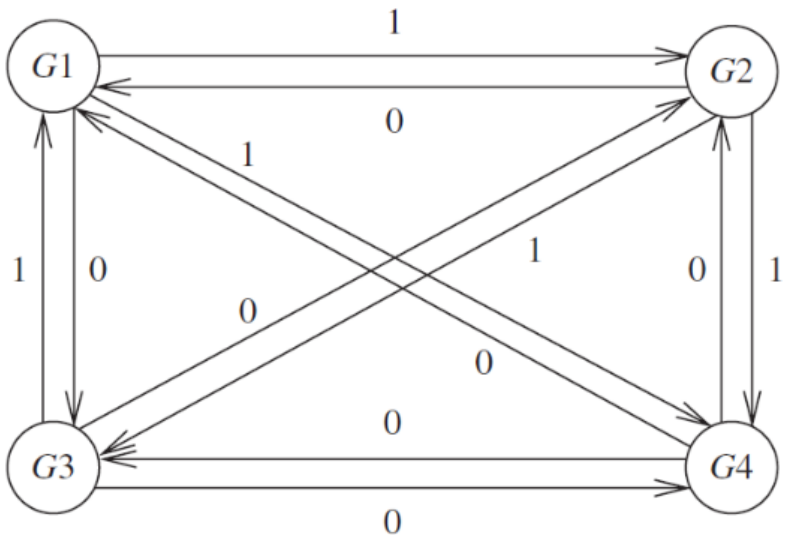
Systémy s Byzantskými poruchami

- Systémy s Byzantskými poruchami na úrovni procesů
  - ✓ proces může vynechat některý krok své procedury
  - ✓ proces může provést krok procedury, se kterým se nepočítalo
  - ✓ proces může své datové položky nastavit na špatné hodnoty
  - ✓ proces na výzvu může vrátit chybnou odpověď, což vyzývající nepozná, poněvadž správnou odpověď předem nezná
- Systémy s Byzantskými poruchami na úrovni komunikací
  - ✓ komunikační kanál předá porušenou zprávu bez detekce chyby
  - ✓ komunikační kanál předá jednu a tutéž zprávu vícekrát
  - ✓ komunikační kanál předá nikým nevyslanou zprávu
  - ✓ KOMUNIKAČNÍ PORUCHY JSOU ŘÍDKÉ – v sítích je protokolárně lze řešit na úrovních vrstev datového spoje, transportní a/nebo aplikační

Ilustrace Byzantinských poruch

- Čtyři tábory útočící armády, každý z nich velí generál,
  táboří kolem obléhané pevnosti, pevnost dobudou pouze
  případě, že na pevnost zaútočí současně,
  musí se dohodnout na čase útoku.
- Komunikují pomocí poslů (zprávami), kteří mohou cestovat
  mezi dvěma tábory libovolně dlouho.
- Výpadek zprávy je modelován zachycením posla nepřítelem.
- Byzantinsky se chovající proces je modelován zrádcem.
  - ✓ Zrádce se pokusí podkopat dohodovací mechanismus tím,
    že dá zavádějící informace informace ostatním generálům.
  - ✓ Zrádce může například informovat jednoho generála,
    aby útočil v 10 hodin a ostatní generály, aby zaútočili v poledne.
  - ✓ Nebo nemusí poslat některému generálovi zprávu.
  - ✓ Nebo může manipulovat se zprávami,
    které dostane od jiných generálů, než je předá dál.

Ilustrace Byzantinských poruch



- Generálové se dohadují na boolovské hodnotě (0, 1)
- Někteří generálové mění hodnotu rozhodovací proměnné, což vede ke zmatku.
- Lze v takovém případě dosáhnout dohody? Pokud ano za jakých podmínek ?
- Je-li dohoda dosažitelná, musí být dosažena protokolem

Výpadky vs. Byzantinské chyby

- DS s $N = 2f + 1$ komponentami může tolerovat současné výpadky až f komponent
- Kolik komponent DS může vykazovat Byzantinské chyby, aby DS mohl spolehlivě poskytovat službu, na kterou byl navržený ?
  - ✓ Služba – např. řízení kosmické sondy N paralelními počítači
  - ✓ Ukážeme si, že komponent DS vykazujících byzantinské chyby musí být méně jak 1/3
  - ✓ Pokud v DS vykazuje byzantinské chyby f komponent, DS musí obsahovat alespoň $3f + 1$ komponent

Klasifikace forem problémů shody v DS

- Komponenty DS podílející se násobně na plnění služby se musí shodnout na výsledku takové služby
- Shoda může mít více forem
- Shoda (*Consensus*), n:1
  - ✓ každý proces deklaruje svoji iniciální hodnotu
  - ✓ všechny validní procesy se musí shodnout na jediné iniciální hodnotě
  - ✓ pokud je iniciální hodnota všech validních procesů stejná, musí se všechny validní procesy shodnout na této hodnotě
  - ✓ každý validní proces musí dojít k rozhodnutí v konečném čase

Klasifikace forem problémů shody v DS

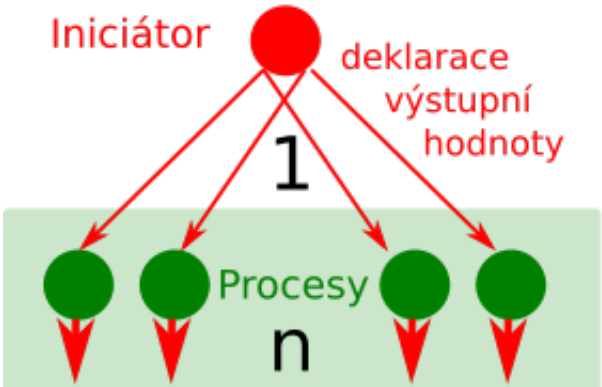
- Byzantská dohoda (*Byzantine agreement*), 1:n
  - ✓ jeden z procesů ve skupině, iniciátor shody, zvolí iniciální hodnotu, a rozešle ji všem ostatním procesům ve skupině
  - ✓ ve skupině procesů se mohou nacházet jak validní procesy, tak i nevalidní procesy (chybující, . . . , generující lživé zprávy, . . .)
  - ✓ všechny validní procesy se musí shodnout na stejné hodnotě
  - ✓ pokud je iniciátor validní, musí se všechny validní procesy shodnout na hodnotě deklarované iniciátorem
  - ✓ všechny validní procesy musí dojít k rozhodnutí v konečném čase
  - ✓ Kolik může být ve skupině nevalidních procesů, aby se validní procesy dokázaly shodnout na jedné hodnotě ?

Klasifikace forem problémů shody v DS

- Dosažení interaktivní konzistence, n:n
  - ✓ každý proces ve skupině deklaruje svoji iniciální hodnotu
  - ✓ iniciální hodnoty jednotlivých procesů popisuje vektor, pole, $A[v_1, \ldots, v_n]$, kde v_i jsou iniciální hodnoty procesů p_i
  - ✓ všechny validní procesy se musí shodnout na jedinečném poli hodnot
  - ✓ Pokud je proces i validní a jeho iniciální hodnota je v_i, pak všechny validní procesy musí dojít k shodě na v_i jako hodnotě i-tého prvku A.
  - ✓ Pokud proces j není validní, pak validní procesy mohou přiřadit j-tému prvku A libovolnou hodnotu
  - ✓ Všechny validní procesy musí dojít k rozhodnutí o poli A v konečném čase

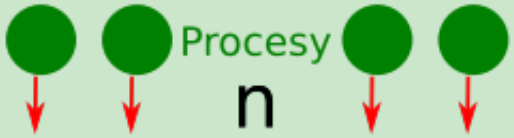
Typové případy shody, přehled

Byzantská shoda

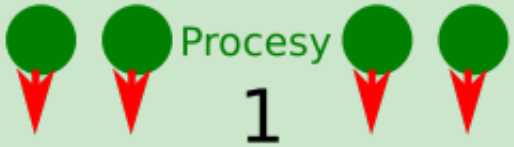


Iniciátor sdělí procesům výstupní hodnotu.
Validní procesy se musí shodnout na jediné výstupní hodnotě.
Pokud je iniciátor validní, musí se všechny validní procesy shodnout na hodnotě deklarované iniciátorem.

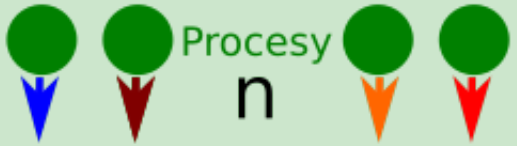
Shoda



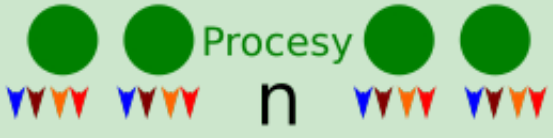
Každý proces deklaruje svoji iniciální výstupní hodnotu.
Validní procesy se musí shodnout na jediné výstupní hodnotě.
Pokud jsou iniciální hodnoty všech validních procesů shodné, musí se všechny validní procesy shodnout na této hodnotě.



Interaktivní konzistence



Každý proces deklaruje svoji iniciální výstupní hodnotu.
Validní procesy se musí shodnout na tomto jedinečném poli výstupních hodnot.



Zahajuje	1 proces	všechny procesy	všechny procesy
Shoda na	1 hodnotě	1 hodnotě	vektoru hodnot

Klasifikace problémů distribuované shody

- Všechny problémy jsou vzájemně ekvivalentní
- Interaktivní konzistenci lze odvodit Byzantskou dohodou
  - ✓ řeší n-krát byzantská dohoda, jednou pro každý z n procesů
- Podobně lze odvodit shodu interaktivní konsistencí
- Podobně lze odvodit Byzantskou dohodu pomocí shody, . . .
- V DS s Byzantinskými poruchami lze dosáhnout shody pouze pokud je DS synchronní
  - ✓ V synchronním DS s n procesy, z nichž f je potenciálně nevalidních, je dohoda dosažitelná v $f + l$ fázích výměn zpráv mezi procesy pouze když platí
    $f \leq \lfloor (n-1)/3 \rfloor$

Problém dosažení shody, motivační příklady

- Procesy se mají shodnout na jisté hodnotě po té, co jeden nebo více procesů navrhují co touto hodnotou má být
  - ✓ V transakci přenášející kapitál z jednoho účtu na jiný účet, se musí participující počítače shodnout na příslušném debitu a kreditu
  - ✓ Při řešení distribuovaného vzájemného vyloučení (DME) se musí procesy shodnout, který z nich může vstoupit do kritické sekce
  - ✓ Probírané algoritmy DME ale netolerovaly ztrátu zprávy v nespolehlivém kanálu
  - ✓ Bankovní operace jsou řešeny transakcemi, které mají účinek pouze když nedojde k žaádnému výpadku během jejich řešení
  - ✓ Co se stane, když se ztratí zpráva ?
  - ✓ Co se stane, když některý z procesů účastných v DS vypadne ?

Problém dosažení shody, motivační příklady

- Robotický fotbalový tým
  - ✓ Každý robot má senzory pro detekci prostředí kolem něj.
  - ✓ Tým robotů se musí dohodnout na strategii (např. obranné nebo útočné) na základě dosažení shody na tom, zda tým má míč pod kontrolou nebo zda ho má pod kontrolou protivník.
  - ✓ Každý robot začíná s hodnotou odvozenou ze svých senzorů (já mám míč pod kontrolou $\Rightarrow$ tým má míč pod kontrolou).
  - ✓ Pomocí algoritmu shody se každý robot rozhodne, zda tým má míč pod kontrolou nebo ne.

Problém dosažení shody, motivační příklady

- Množina procesů kooperující na finálním výsledku úlohy
  - ✓ Každý proces pošle jím navrhovanou finální hodnotu každému z ostatních procesů
  - ✓ Každý proces pak řeší stejnou funkci určující finální hodnotu založenou na znalosti navrhovaných hodnot
  - ✓ V systému bez výpadků procesů každý proces začíná řešit shodu se stejnými vstupy, počítá stejnou funkci a rozhodne se na základě dosažení shody pro stejnou hodnotu, dochází ke konsenzu
  - ✓ V systému s výpadky procesů je pro dosažení dohody zapotřebí více kol takových komunikaci – výměn zpráv

Definice problému dosažení shody

- Modelové prostředí
  - ✓ kolekce p_i ($i = 1, 2, \ldots, N$) procesů komunikujících výměnou zpráv
  - ✓ komunikace jsou spolehlivé,
    procesy mohou vykazovat Byzantské chyby (mohou lhát, vypadávat)
  - ✓ nespolehlivých procesů může být až M ($M \leq N$)
- Všeobecná definice problému dosažení shody
  - ✓ Každý proces p_i ($i = 1, 2, \ldots, N$) začíná dosahování shody
    ve stavu nerozhodnutý a navrhuje jako výsledek jistou hodnotu v_i
    z množiny D ($i = 1, 2, \ldots, N$)
  - ✓ Procesy mezi sebou vzájemně komunikují (případně ve více fázích) a
    vyměňují si mezi sebou jim známé výstupní hodnoty
  - ✓ Po ukončení komunikace každý proces nastaví hodnotu své
    proměnné rozhodnutí d_i na základě znalosti navrhovaných hodnot
    ostatními procesy a přejde do stavu rozhodnutý,
    ve kterém už nemůže rozhodnutí měnit

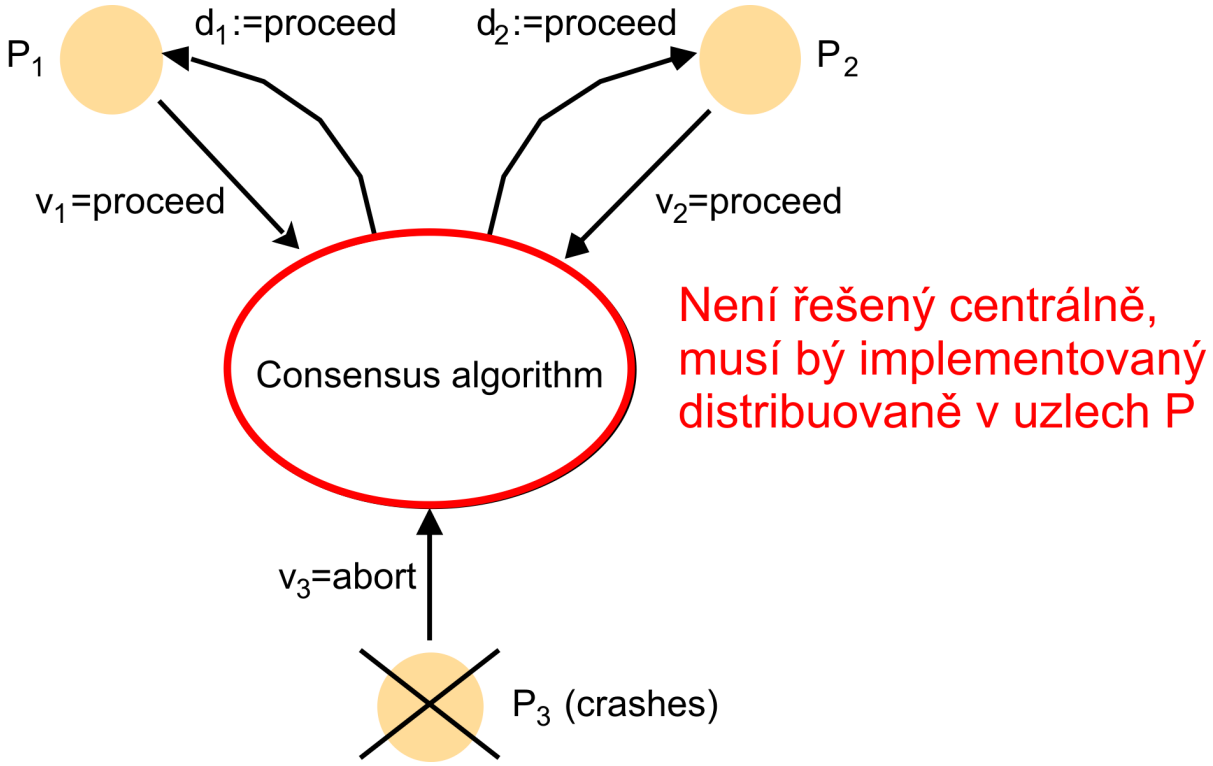
Definice problému dosažení shody

- Požadavky na vlastnosti algoritmu dosažení shody
  - ✓ ukončení – všechny nechybující, korektní, procesy nakonec nastaví svoji proměnnou rozhodnutí – živost, *liveness*
  - ✓ dosažení shody – všechny nechybující, korektní, procesy nastaví svoji proměnnou rozhodnutí na stejnou hodnotu
  - ✓ integrita, resp. validita – jestliže některý nechybující, validní, proces navrhuje hodnotu V, pak každý nechybující, validní, proces ve stavu rozhodnutý zvolí hodnotu V
    – integrita a dosažení shody = *safety*, bezpečnost
- Definici integrity lze ale upravovat podle požadavků aplikace
  - ✓ Např. nechybující procesy mohou navrhovat více různých hodnot, a rozhodnutí může nabýt jedné z takových hodnot podle funkce použité k dosažení shody o rozhodnutí z návrhů

    Funkcí může být majorita, minimum, maximum, . . . , z v_i, případně indikace speciální hodnoty $\perp \notin D$ pokud extrém neexistuje apod.

Dosažení shody, konceptuální pohled

- dva nechybující procesy, každý za sebe, navrhly *proceed*
- třetí proces navrhl *abort* a zkrachoval
- dva nechybující procesy nastavují rozhodnutí na *proceed*



Dosažení shody, procesní model pro synchronní DS

- Model
  - ✓ Komunikační médium je spolehlivé, věrolomné jsou procesy (krachují)
  - ✓ n procesů, zkrachovalých procesů není více než f, $f < n$
  - ✓ Každý proces definuje navrhovanou hodnotu V_i
  - ✓ Pomocí vzájemného zasílání zpráv si každý spolehlivý proces P_i vytváří vektor $X_i = (A_{i,1}, A_{i,2}, \ldots, A_{i,n})$, pro který platí
    - Pokud je P_j spolehlivý proces, pak $A_{i,j} = V_j$
    - Pokud jsou P_i a P_j spolehlivé, pak mají shodné $X_i = X_j$

Dosažení shody, procesní model pro synchronní DS

- Známá řešení mají vlastnosti
  - ✓ Doba provádění algoritmu je úměrná $f+1$ kolům výměn zpráv, v každém kole výměn zpráv každý proces zašle všem procesům hodnoty, které získal v předchozím kole od všech procesů
  - ✓ Z hlediska počtu přenášených zpráv se jedná o nákladný algoritmus, algoritmus je řešený $O(n^{f+1})$ zprávami.

    Může-li selhat nejvýše 1 proces, validní procesy si musí znalost vyměnit ve dvou kolech, komunikační složitost je kvadratická
  - ✓ $f+1$ kol se vyžaduje proto, poněvadž proces může selhat v každém okamžiku, tedy i v průběhu operace vysílání své znalosti.

    Proces může poslat navrhovanou hodnotu procesu j, ale ne už procesu k, . . . a pak se budou po ukončení vysílání znalosti j a k lišit a j a k by mohly rozhodnout odlišně. Ale v příštím kole si j a k vymění své znalosti, takže rozdíly se smažou
  - ✓ Jestliže věrolomný proces zprávu nepošle, spolehlivý proces může zvolit, že od něj získal implicitní hodnotu ($\perp$)

Dosažení shody, procesní model pro synchronní DS

- Příklad – algoritmus pro $f = 1$ and $n = 3$
  - ✓ proběhnou 2 kola výměn zpráv ($f = 1,\ f + 1 = 2$)
  - ✓ 1. kolo – každý proces pošle všem ostatním procesům svoji hodnotu V_i
  - ✓ 2. kolo – každý proces pošle všem ostatním procesům znalost, výstupních hodnot, kterou získal v 1. kole
  - ✓ Po konci 2. kola si každý spolehlivý proces P_i může vytvořit rozhodovací vektor $X_i = (A_{i,1}, A_{i,2}, A_{i,3})$ např. následovně:
    - Pro $i = j,\ A_{i,i} = V_i$,
    - Pro $i \neq j$:
      - Pro určení hodnoty $A_{i,j}$ se použije např. majorita pokud existuje většina na hodnotách oznámených o procesu P_j,
      - Pokud většina neexistuje, použije pro $A_{i,j}$ implicitní hodnotu, $\perp$
      - příp. lze pro určení hodnoty $A_{i,j}$ použít fce min nebo max apod

Dosažení shody, synchronní DS

✓ Pro rozhodnutí se v následujícím programu se používá fce *minimum*

Algorithm for process $p_i \in g$; algorithm proceeds in $f + 1$ rounds

On initialization

$Values_i^1 := \{v_i\}$; $Values_i^0 = \{\}$; // $Values_i^r$ *hodnoty známé* p_i *na počátku kola* r

In round r $(1 \le r \le f + 1)$

B-multicast$(g, Values_i^r - Values_i^{r-1})$; // Send only values that have not been sent

$Values_i^{r+1} := Values_i^r$;

while (in round r)

{

On B-deliver(V_j) *from some* p_j

$Values_i^{r+1} := Values_i^{r+1} \cup V_j$;

}

After $(f + 1)$ *rounds*

Assign $d_i = minimum(Values_i^{f+1})$;

Dosažení shody, synchronní DS

n = 3

f = 0, dohoda proběhne ve 1 kole (r)

rozhodnutí = minimum ()

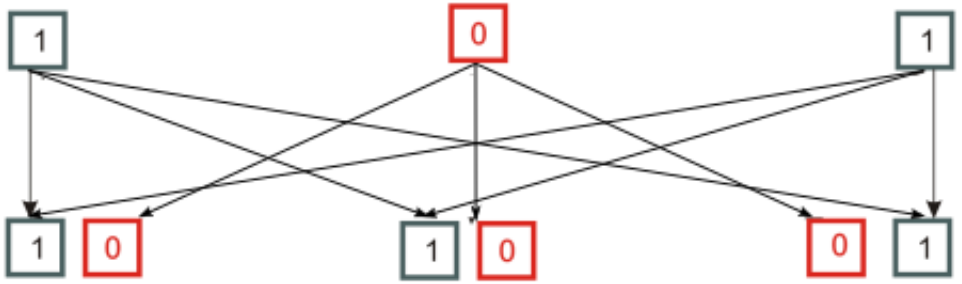
$Values^0_i$

$Values^1_i = v_i$, (co zná p_i na počátku kola r=1)

r=1: B-multicast ($Values^1_i$ - $Values^0_i$)

Kdykoliv p_i přijme zprávu od p_j

$Values^2_i = Values^1_i \cup v_j$



Dosažení shody, synchronní DS

n = 3

f = 1, dohoda proběhne ve 2 kolech (r)

rozhodnutí = minimum ()

$Values^0_i$

$Values^1_i = v_i$, (co zná p_i na počátku kola r=1)

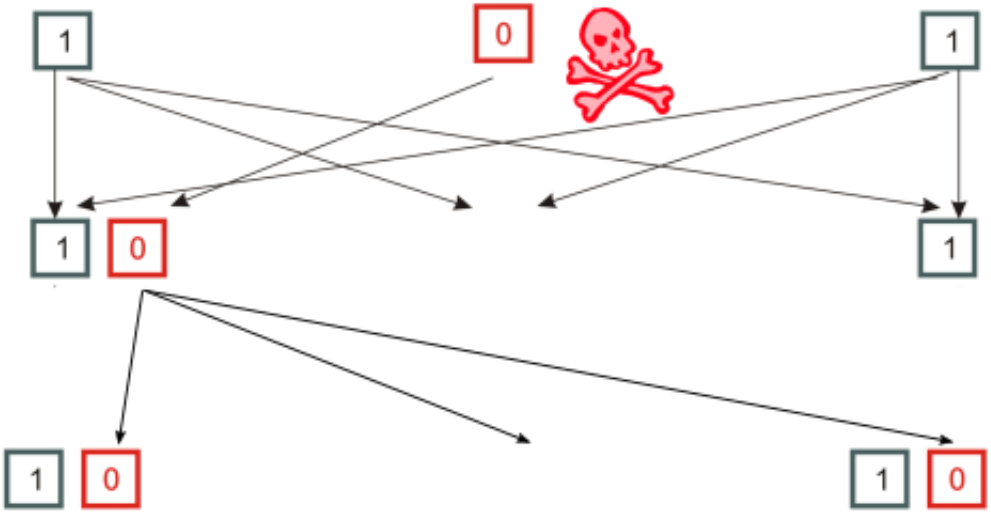
Vyšle 0 procesu 1 a zkrachuje

r=1: B-multicast ($Values^1_i$ - $Values^0_i$)

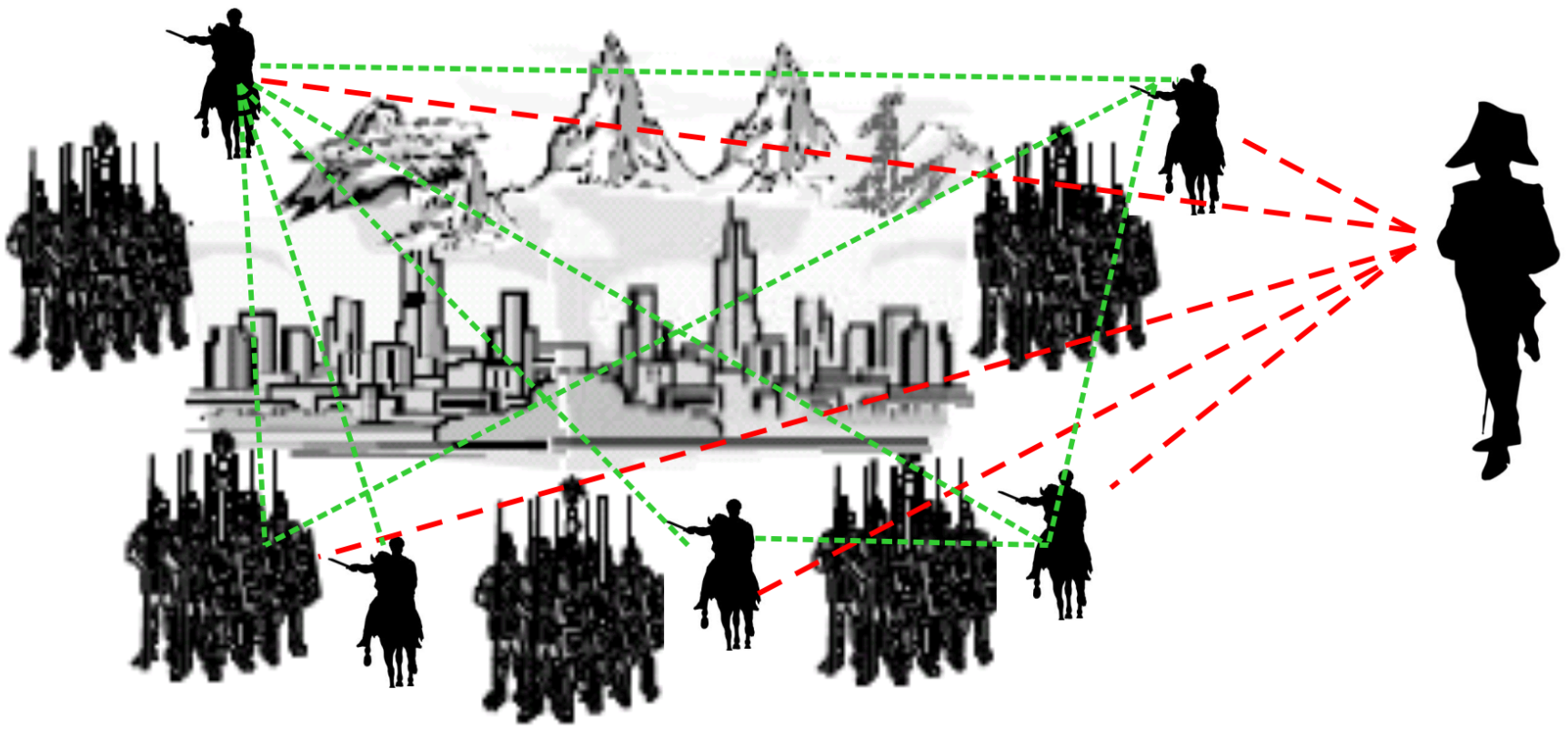
Kdykoliv p_i přijme zprávu od p_j
$Values^2_i = Values^1_i \cup v_j$

r=2: B-multicast ($Values^2_i$ - $Values^1_i$)

$Values^3_i = Values^2_i \cup v_j$



Problém Byzantských generálů (Byzantské shody)



Problém Byzantských generálů (Byzantské shody)

- Na rozdíl od problému dosažení shody v problému BG
  jeden *master* proces sděluje výstupní hodnotu
  a validní *slave* procesy se musí shodnout
  na jediné výstupní hodnotě
- Pokud je *master* validní, musí se *slave* shodnout na jeho výstupní hodnotě
- Komunikace mezi generály (uzly) je spolehlivá
- Příklad
  - ✓ 3 a více generálů se zprávami informují za útočit či ustoupit,
    armádní generál dává rozkaz divizním generálům a
    divizní generálové se musí dohodnou zda se bude útočit či ustupovat,
    protože kterýkoliv generál může být věrolomný (nevalidní)
    - věrolomný armádní g. dává jednotlivým divizním různé příkazy
    - věrolomný divizní g. sděluje ostatním d. g. různé informace
    - nevěrolomní divizní g. se musí na akci shodnout

Problém Byzantských generálů (Byzantské shody)

- Požadavky na vlastnosti algoritmu dosažení shody Byzantských generálů
  - ✓ ukončení – všichni nevěrolomní divizní generálové nakonec nastaví svoji proměnnou rozhodnutí, d_i
  - ✓ dosažení shody – všichni nevěrolomní divizní generálové nastaví svoji proměnnou rozhodnutí na stejnou hodnotu
  - ✓ integrita –

    všichni nevěrolomní divizní generálové vykonají stejný rozkaz bez ohledu na to, zda armádní generál je či není věrolomný

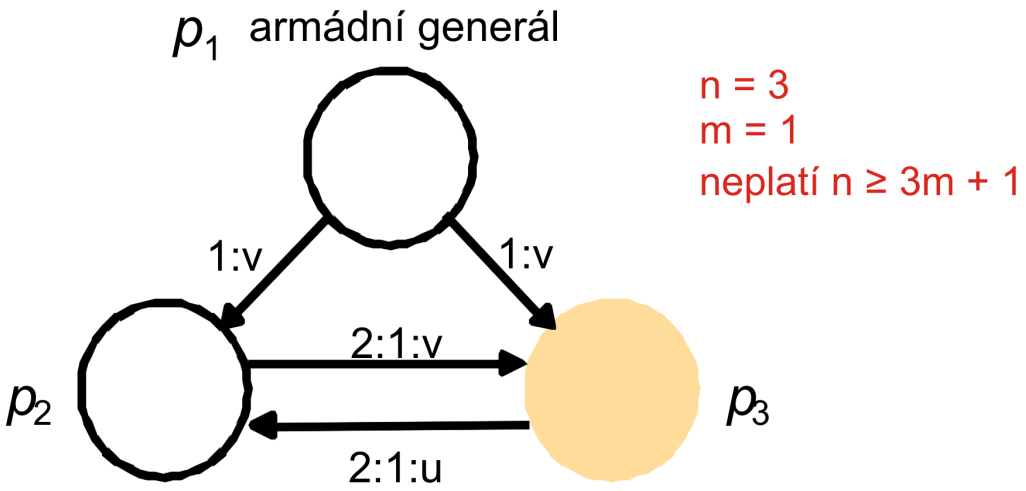
    pokud je armádní generál nevěrolomný, korektní, všichni nevěrolomní, korektní, divizní generálové vykonají jeho rozkaz
- Ukážeme později, úloha BG má řešení pokud je více než 2/3 generálů nevěrolomných, korektních

Problém interaktivní shody

- Na rozdíl od problému BG každý proces (divizní generál) nabízí svoji výstupní hodnotu a žádný armádní generál neexistuje
- Cílem je dosažení shody všech korektních procesů na rozhodovacím vektoru výstupních hodnot jednotlivých procesů, ze kterého lze odvodit rozhodnutí
- Požadavky na vlastnosti algoritmu dosažení shody
  - ✓ ukončení – všechny nechybující, korektní, procesy nakonec nastaví svůj rozhodovací vektor
  - ✓ dosažení shody – rozhodovací vektor všech nechybujících, korektních, procesů je shodný
  - ✓ integrita – je-li p_i korektní proces s výstupem v_i, pak se všechny korektní procesy shodnou, že výstupem p_i je v_i

Problém Byzantských generálů (Byzantské shody), řešení

- ✓ Správný algoritmus je znám jen pro případy kdy $n \geq 3 \times m + 1$, kde n je počet generálů a m je počet věrolomných generálů

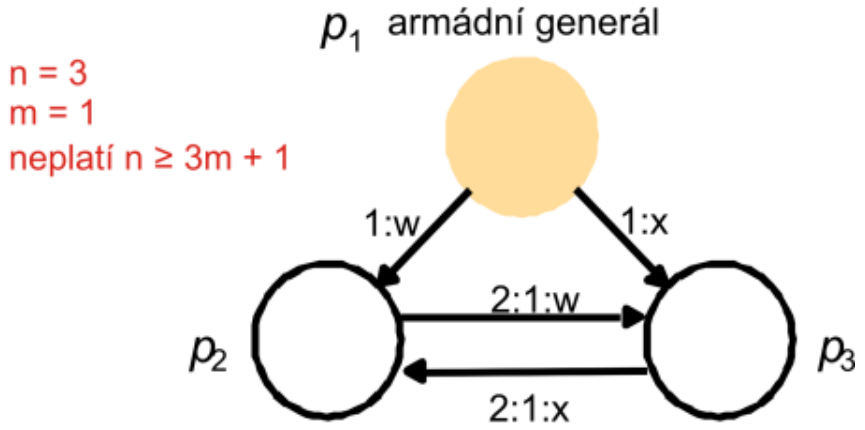


Věrolomné procesy jsou vybarvené

- ✓ věrolomný je divizní generál p_3
- ✓ nevěrolomný divizní generál p_2 nemá možnost udělat rozhodnutí,
  - od armádního generála dostal hodnotu v
  - od divizního generála p_3 dostal zprávu, že armádní generál řekl u

Problém Byzantských generálů (Byzantské shody), řešení

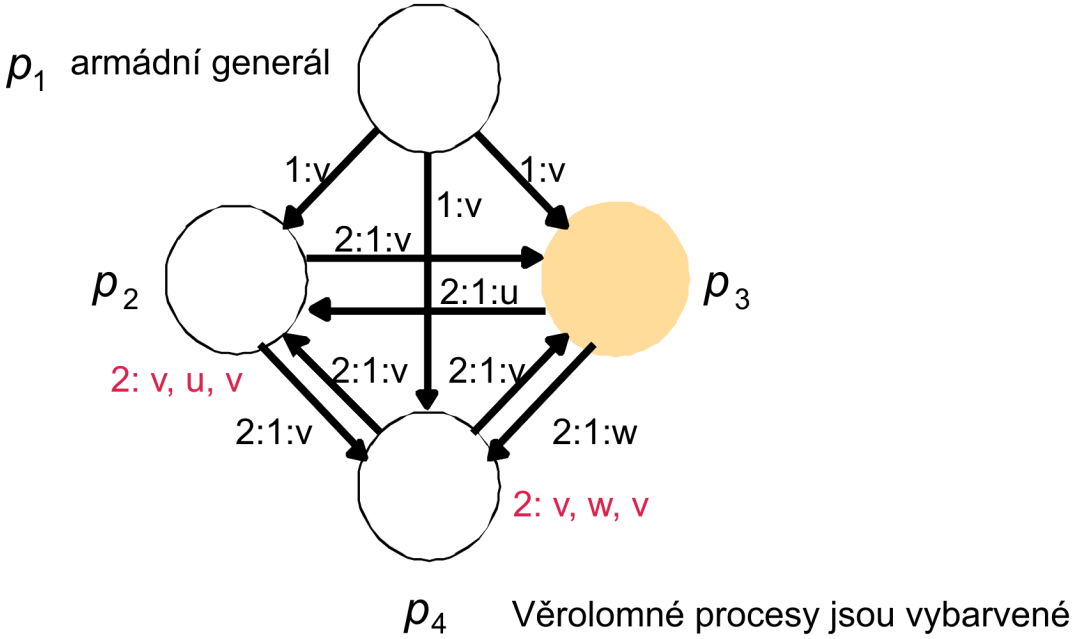
- ✓ Správný algoritmus je znám jen pro případy kdy $n \geq 3 \times m + 1$, kde n je počet generálů a m je počet věrolomných generálů



Věrolomné procesy jsou vybarvené

- ✓ věrolomný je armádní generál p_1,
- ✓ nevěrolomný p_2 nemá možnost udělat rozhodnutí,
  - od armádního generála p_1 dostal hodnotu w
  - od divizního generála p_3 dostal zprávu, že armádní p_1 řekl x
- ✓ Nemožnost rozhodnutí platí i pro p_3

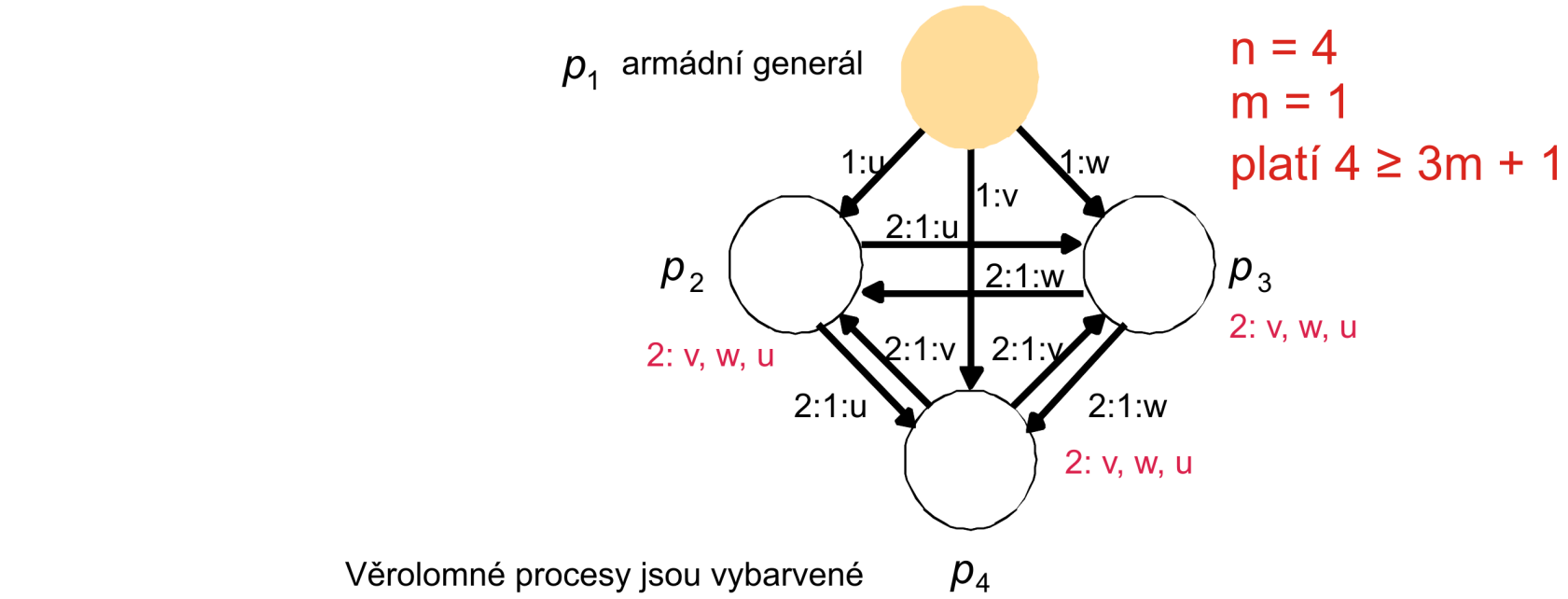
Problém Byzantských generálů (Byzantské shody), řešení



n = 4
m = 1
platí 4 ≥ 3m + 1

- věrolomný je divizní generál p_3
- nevěrolomný p_2 rozhoduje podle $majority(v, u, v) = v$
- nevěrolomný p_4: rozhoduje podle $majority(v, w, v) = v$

Problém Byzantských generálů (Byzantské shody), řešení



- věrolomný je armádní generál p_1
- divizní generálové p_2, p_3 a p_4 rozhodují podle
  $majority(v, u, w) = \perp$, $\perp$ je implicitní hodnota pro případ, že majoritu nelze určit, např. $\perp = v$

Problém Byzantských generálů (Byzantské shody), řešení

- Poznámka, připomenutí podmínky možnosti shody
  - ✓ $n \geq 3m + 1$
  - ✓ při 5 existujících: $5 \geq 3 \times 1 + 1$, nejvýše 1 věrolomný
  - ✓ při 6 existujících: $6 \geq 3 \times 1 + 1$, nejvýše 1 věrolomný
  - ✓ při 7 existujících: $7 \geq 3 \times 2 + 1$, nejvýše 2 věrolomní
  - ✓ ...
  - ✓ při 10 existujících: $10 \geq 3 \times 3 + 1$, nejvýše 3 věrolomní
  - ✓ ...
  - ✓ při 13 existujících: $13 \geq 3 \times 4 + 1$, nejvýše 4 věrolomní
  - ✓ ...
  - ✓ při 16 existujících: $16 \geq 3 \times 5 + 1$, nejvýše 5 věrolomných
  - ✓ ...