Role a principy OS
(v distribuovaných systémech zvláště)

PA 150 ◇ Principy operačních systémů

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

Verze : podzim 2020

Osnova předmětu PA150

Vybraná témata charakterizující řešení souběžnosti a perzistence
v OS a pomocí OS a middleware usnadňující efektivní návrh a realizaci
netriviálních aplikačních (cíleně distribuovaných) systémů

- Role a principy operačních systémů, distribuované systémy
- Typové synchronizační úlohy
- Problém uváznutí souběžných činností
- Čas a stav v distribuovaném prostředí
- Distribuovaná řešení typových synchronizačních úloh:
  vzájemné vyloučení, dosažení shody, multicasting, . . .
- Řešení uváznutí v distribuovaném prostředí
- Transakční zpracování, řízení souběžných transakcí a
  obnova transakčního zpracování po výpadku

Předpoklady pro studium v PA150

- Pro porozumění výkladu je absolutně nutná znalost
  principů OS z hlediska řešení
  virtualizace, souběžnosti a perzistence
  alespoň v rozsahu vhodného základního kursu o OS,
  např. PB152, zvláště pak temat:
  - Bázové rysy hardware
    (přerušení, DMA, logický/fyzický adresový prostor, ...)
  - Architektury OS (jádro, mikrojádro, middleware, ...)
  - Procesy / vlákna
  - Správa a virtualizace paměti
  - Plánování činností procesoru
  - Souborové systémy a ovládání IO

Literatura

- Přednášky jsou motivovány učebnicemi

  A. Silberschatz, P.B. Galvin, G. Gange,
  Operating Systems Concepts
  John Wiley, 2018, 10. vydání,
  ISBN: 978-1-119-32091-3

  G. Coulouris, J. Dollimore, T. Kindberg, G. Blair
  Distributed Systems, Concepts and Design,
  Addison Wesley, 2012, 5. vydání,
  ISBN 978-0-13-214301-1

Připomenutí ,,elementárních'' faktů

- Počítač – sestava komponent poskytujících
  - výpočetní schopnost (procesor),
  - paměťovou schopnost (paměť) a
  - komunikačních schopnost (periférie)
- Program – Předpis pro konkrétní činnost počítače
  tvořený sekvencemi instrukcí
- Častý slogan: *program běží v počítači*
  - ✓ Co se děje když program běží v počítači ?

    Procesor interpretuje program pamatovaný v paměti
    a manipuluje s daty uloženými v téže paměti –
    – provádí operace předepsané instrukcemi (mnohdy $\simeq 10^9/s$)

    Provádí $\equiv$ procesor získává instrukci z paměti, dekóduje ji,
    provede odpovídající operaci a přechází k získávání další instrukce
  - ✓ Toto chování odpovídá Von Neumannovu modelu počítání

Připomenutí ,,elementárních'' faktů

- Je přirozeným požadavkem, aby aplikační programy byly proveditelné ve více různorodých hardwarových konfiguracích počítačů
- Vysoký výpočetní výkon potřebný pro řešení mnoha aplikačních úloh je dosažitelný pouze souběžností jejich řešení (a mnohdy i jejich částí)
- Je tudíž potřebný balík programů, který odpovídá za to, že aplikační programy v počítači běží souběžně a efektivně, tj., že jimi řízené výpočty mohou
  - ✓ sdílet paměť
  - ✓ sdílet procesor (procesory, je-li jich v počítači více)
  - ✓ souběžně komunikovat s aplikačním okolím.
  - ✓ Tímto balíkem software je operační systém, OS

Připomenutí ,,elementárních'' faktů

- OS plně spravuje využívání komponent počítače a plně řídí běhy výpočtů podle aplikačních programů – procesy
- Klíčovou úlohou OS je podpora virtualizace
  - ✓ Všem procesům poskytuje jedinečné, výkonné, obecné, snadno použitelné virtuální reprezentace omezených fyzických zdrojů počítače
- Další rolí OS je poskytování rozhraní (API) pro ovládání virtuálních strojů poskytovaných
  - ✓ uživatelům – *GUI, Command Interpreter, shell, ...*
  - ✓ procesům – služby OS – spouštění výpočtů, zpřístupňování pamětí a zařízení, datových objektů, ...
  - ✓ Služby jsou poskytované formou volání OS obvykle zabalených do podprogramů seskupených do nějaké standardní knihovny OS

Připomenutí „elementárních" faktů

- Díky virtualizaci zdrojů počítače, kterou poskytuje OS
  - ✓ v počítači běží více procesů souběžně (sdílí se procesor)
  - ✓ více procesů souběžně přistupuje do paměti pro instrukce a data (sdílí se paměť)
  - ✓ více výpočtů souběžně přistupuje do vnějších pamětí paměti pro data (sdílí se disky), . . .
  - ✓ OS tudíž musí vystupovat v roli manažera zdrojů
- Program x proces x vlákno
  - ✓ Program – text obsahující instrukce/příkazy pro řízení výpočtu
  - ✓ Proces – identifikovatelný výpočet podle programu, kterému OS přiděluje realizační zdroje (paměť, procesor, . . .), schopný souběžného řešení s více procesy v jednom počítači
  - ✓ Vlákno – identifikovatelná dílčí činnost v rámci procesu, které OS přiděluje realizační zdroj (procesor), schopná souběžného řešení s více vlákny v jednom procesu

Připomenutí ,,elementárních'' faktů

- Role OS (a jeho nadstaveb typu *middleware*)
  co manažera zdrojů si vyžaduje splnění podmínek:
  - ✓ musí existovat politika OS určující, který proces získá fyzický procesor pro běh jeho virtuálního procesoru a mechanismy, které ji umožní implementovat (časovač, mechanismus přerušení, . . .)
  - ✓ každý proces se realizuje ve svém virtuálním adresovém prostoru (VAP) a proto musí existovat

    politika OS určující způsob sdílení fyzického adresového prostoru (FAP) počítače virtuálními pamětmi a

    mechanismy, které umožní tuto politiku implementovat (stránkování, přerušení, zobrazování VAP do FAP, . . .)
  - ✓ jsou dostupné nástroje zajišťující řízený (a atomický) přístup k opakovaně (nesdíleně) přístupným zdrojům a synchronizaci mezi souběžně řešenými výpočty
- Řešení podmínek musí být efektivní, bezpečné a spolehlivé

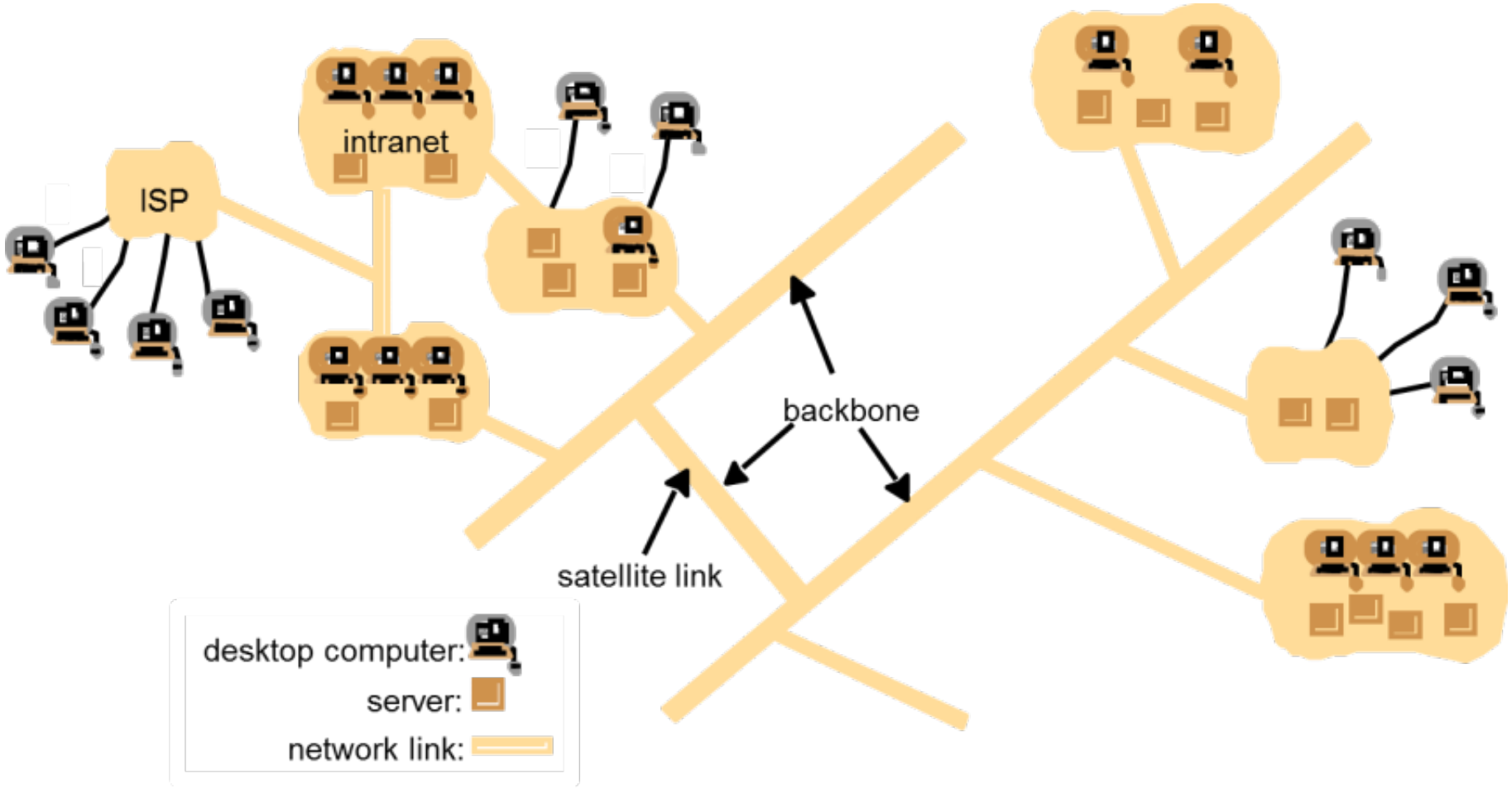
Tři bázové problémy řešené operačními systémy

- Virtualizace zdrojů počítače (procesoru, paměti, . . .)
  - ✓ Jaké mechanismy a politiky jsou implementované v OS pro dosažení virtualizace – iluze pro každý proces, že používá vlastní zdroje ? Jak OS virtualizaci řeší účinně, jakou potřebuje podporu od hardware ?
  - ✓ Odpovědí dává předmět např. PB 152 Operační systémy
- Souběžnost, *Concurrency*,
  - ✓ Jak vytvářet korektní souběžně řešitelné programy ? Souběžné výpočty se řeší mnohdy v jednom fyzickém, reálném, adresovém prostoru. Jaké služby musí poskytnout OS, jaké hardware a jak se používají ?
  - ✓ Odpověd naznačuje předmět PB 152 Operační systémy, detailní výklad poskytuje tento předmět, PA 150
- Trvalost, *Persistence*, jak trvale, dlouhodobě uchovávat data
  - ✓ Většinu odpovědí dává předmět PV 062 Organizace souborů, tento předmět, PA 150, ilustruje problém zajištění perzistence z hlediska potřeb transakčního zpracování v prostředí s výpadky

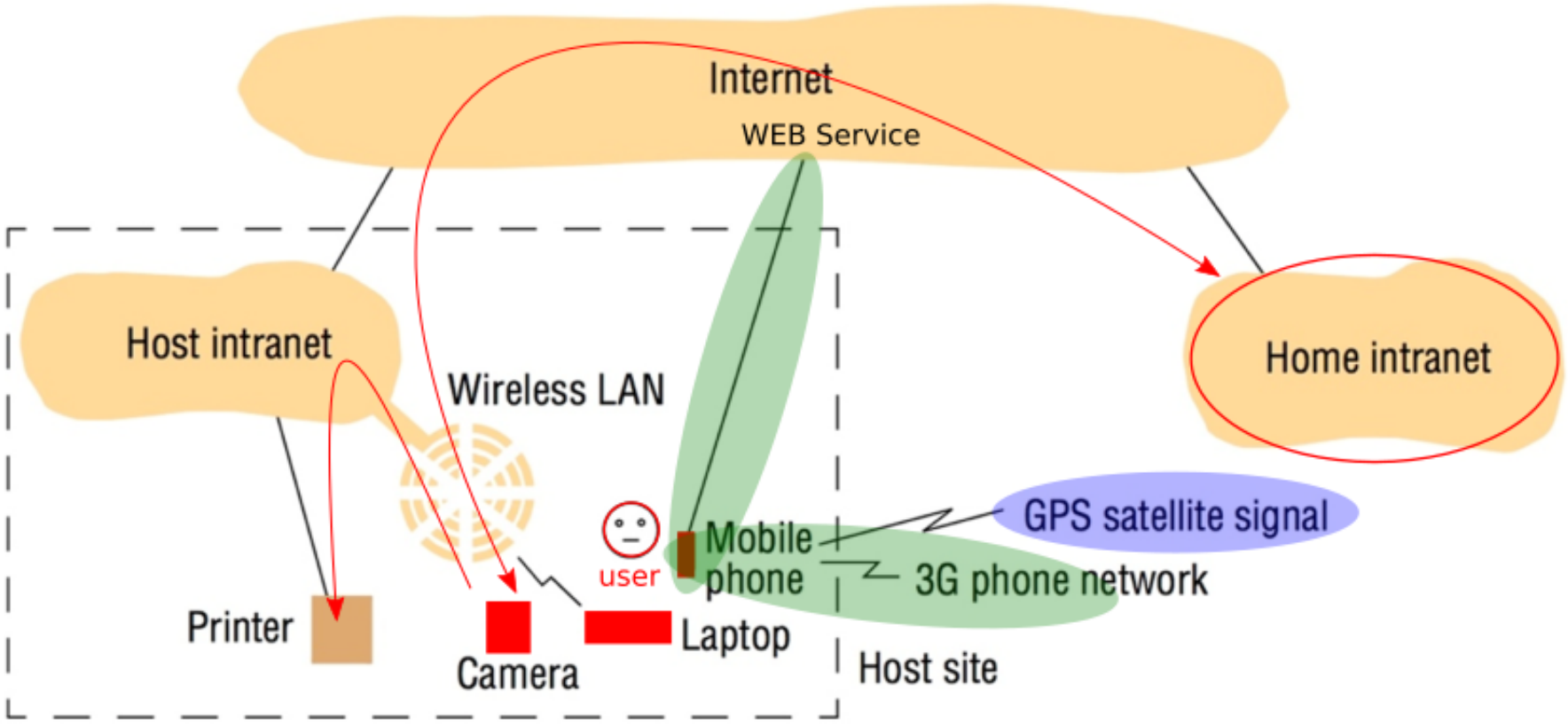
Distribuovaný systém, DS

- DS – systém, jehož hardwarové nebo softwarové komponenty počítačů propojených sítí mohou komunikovat a koordinovat svou činnost pouze předáváním zpráv
- Důsledky
  - ✓ standardní je souběžné řešení programů v DS
  - ✓ v DS neexistuje globální čas a přesto je nutná synchronizace v čase
  - ✓ každá komponenta DS, vč spojů propojovací sítě, může vypadávat a obnovovat svoji činnost nezávisle na ostatních a tyto se o výpadcích nedozvídají
- Soudobé významné trendy v rozvoji DS
  - ✓ extrémní prosíťování
  - ✓ podpora mobility
  - ✓ používání multimediálních služeb
  - ✓ chápání DS jako veřejnou službu využívající a plnící předchozí rysy

Příklad klasického DS – Internet

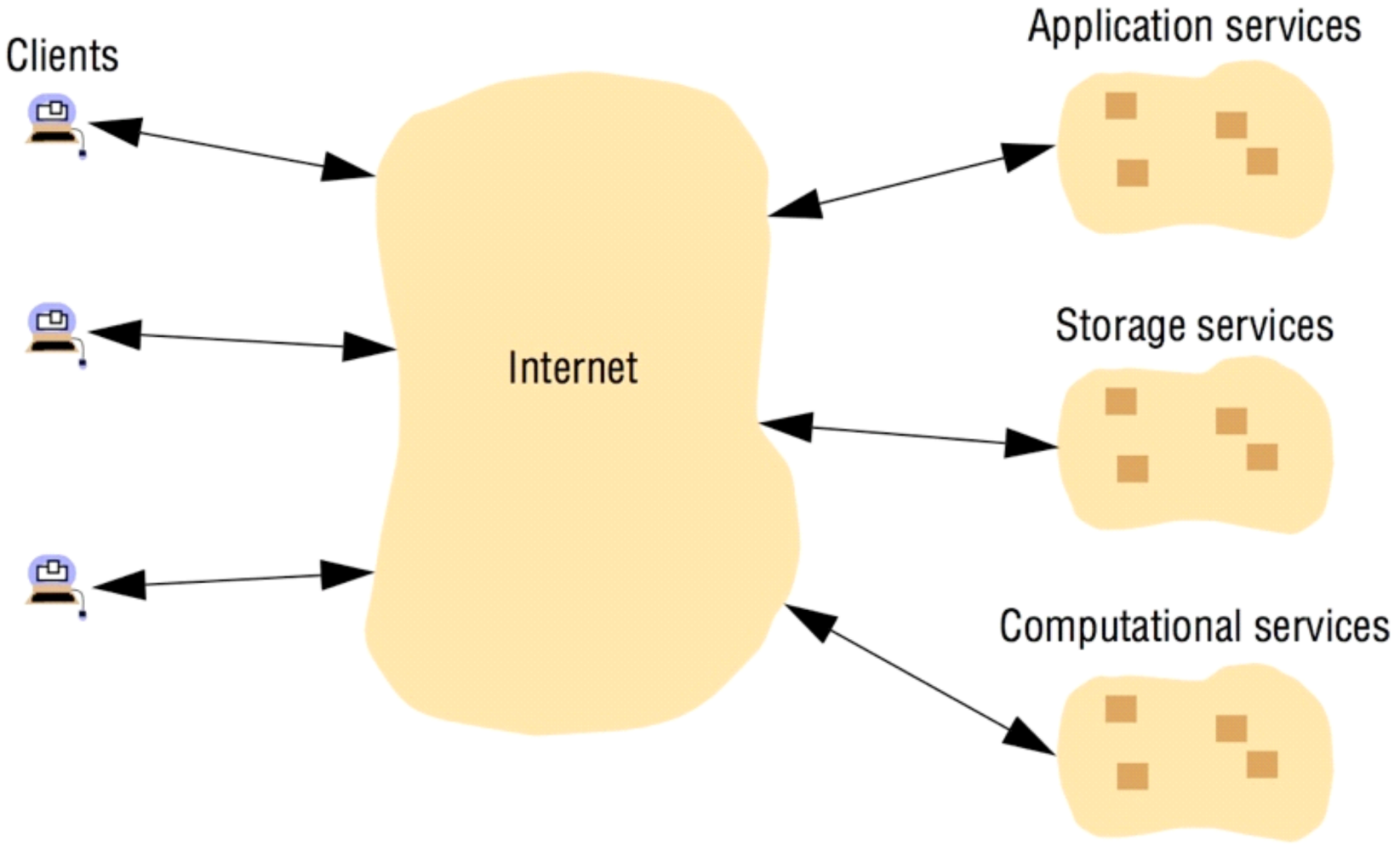


Přenosná / příruční zařízení v DS



- Vyžaduje se přirozená, instinktivní, interoperabilita

Distribuované počítání jako veřejná služba – clouds



Příklad DS – První server Google, 1997, Backrub

Příklad DS – Serverovny Google, Facebook, . . . po 20 letech






Dominantním cílem DS je sdílení zdrojů

- Sdílitelné technické, hardwarové zdroje
  - ✓ procesory, vnitřní paměť, vnější paměť, tiskárny, komunikační cesty
- Sdílitelné logické, softwarové zdroje
  - ✓ webovské stránky
  - ✓ soubory
  - ✓ objekty
  - ✓ databáze
  - ✓ nástěnky
  - ✓ video/audio proudy
  - ✓ exkluzivitní zámky, . . .

Nepominutelné výzvy pro rozvoj DS

- Heterogenita je a vždy bude ve všech relevantních rovinách
  - ✓ sítě, počítače, OS, programovací jazyky, implementace vývojáři
  - ✓ efektivním řešením je middleware – programovací abstrakce maskující heterogenitu podle předchozího bodu
- Smysluplná jsou pouze otevřená řešení
  - ✓ DS musí být rozšiřitelné,
  - ✓ klíčová rozhraní jsou publikovaná
  - ✓ používá se jednotný komunikační mechanismus a jsou publikovaná rozhraní pro přístup ke zdrojům
  - ✓ heterogenta si vynucuje systematické testování a ověřování vyhovění komponent stanoveným standardům
- Řešení DS musí podporovat bezpečnost – důvěrnost, integritu, dostupnost zdrojů

Nepominutelné výzvy pro rozvoj DS

- Škálovatelnost –
  respektování dynamiky nárůstu uživatelů, zdrojů, ...
- Zvládání poruch, výpadků, komponent DS
  - ✓ Obnova činností po výpadku
  - ✓ Redundance zdrojů, komponent
- Souběžnost řešení úloh v DS
- Transparentnost
  - ✓ lokalit, přístupu ke zdrojům, výpadků a poruch, výkonu
  - ✓ dopady distribuovanosti mají být skryté aplikačnímu programátorovi
- Kvalita služeb
  - ✓ adaptibilita, spolehlivost, bezpečnost, výkon, ...

Síťový OS vs. distribuovaný OS

- Síťový OS
  - ✓ Unix, Windows
  - ✓ OS řídicí 1 uzel sítě s vestavěnými schopnostmi pracovat se vzdálenými zdroji v síti
  - ✓ některé zdroje lze zpřístupňovat se síťovou transparencí (NFS zpřístupňující soubory v síti, . . .)
  - ✓ mnohé zdroje si zachovávají uzlovou autonomii (OS řídí procesy ve svém uzlu, plánovat procesy v jiném uzlu nelze, uživatel se musí otevírat relace v jednotlivých uzlech explicitně, . . .
- Distribuovaný OS
  - ✓ zatím v komerční, ekonomicky efektivní rovině neexistuje (experimentální provozování ,,gridově'' orientovaných systémů, . . .)
  - ✓ celá síť se uživateli jeví jako jediný systém

Middleware

- Pojmem middleware se rozumí
  - ✓ Softwarová vrstva ležící mezi aplikacemi a OS poskytující aplikacím programovací abstrakci a maskování heterogenity podpůrných sítí, počítačů, operačních systémů, programovacích jazyků, . . . (API)
  - ✓ Vrstva poskytující aplikačním programátorům jednotný výpočetní model vesměs na bázi paradigmat server-klient, příp. dalších forem navržených pro podporu distribuovaných aplikací v prostředí podporovaném síťovým OS (protože distribuované OS jsou chiméra)
  - ✓ příklady
    - Remote Procedure Call — Klient volá procedury řešené ve vzdáleném uzlu
    - Message Oriented Middleware — Zprávy zaslané klientovi se pamatují v middleware do odebrání (zaměstnaným) klientem
    - Object Request Broker — Zasílání objektů a volání služeb OO prostředí
    - SQL-oriented Data Access — Rozhraní aplikace – DB systém
    - Embedded middleware — komunikační služby a integrace rozhraní software/firmware ve vestavěných aplikacích

Middleware

- middleware = procesy a objekty v počítačích propojených sítí
  + systém výměn zpráv
- Příklady komerčních produktů typu middleware
  - ✓ CORBA (Common Object Request Broker Architecture)
  - ✓ WEB Services
  - ✓ Java RMI (Remote Method Invocation)
  - ✓ DCOM (Distributed Component Object Model, Microsoft),
  - ✓ ...

Middleware

- Nadstavba síťového OS řešící neexistenci distribuovaných OS
  - ✓ OS běžící v uzlu (jádro OS + služby na uživatelské úrovni) poskytuje lokální abstrakce a ty využívá middleware pro implementaci mechanismů pro vzdálené manipulace s objekty a procesy v uzlech

    (řeší se uváznutí, transakce, obnova po výpadku, vzájemné vyloučení kritických sekcí procesů, dosažení shody, . . . , . . .)
- Kombinace middleware a síťového OS je akceptovatelné kompromisní řešení vyváženosti mezi
  - požadavky na autonomii na jedné straně a
  - síťovou transparentostí na druhé straně
- Valná většina problémů studovaných v PA150 spadá do ranku middleware

Architektury DS: komunikační paradigmata

- Komunikující entity
  procesy/vlákna$\longleftrightarrow$procesy/vlákna,
  procesy/vlákna $\longrightarrow$objekty
- komunikační paradigmata
  - ✓ *interprocess communication*, IPC
  - ✓ *remote invocation*, vzdálené vyzývání
  - ✓ *indirect communication*, nepřímá komunikace

Architektury DS: komunikační paradigmata

- IPC
  - ✓ *message-passing*, výměna zpráv mezi procesy, *sockets* (API v Internetu), ...
- vzdálené vyzývání, vzdálená invokace
  - ✓ *Request-reply protocols*

    výměna zpráv mezi procesy podporující model výpočtů klient–server
  - ✓ *Remote procedure calls*, RPC

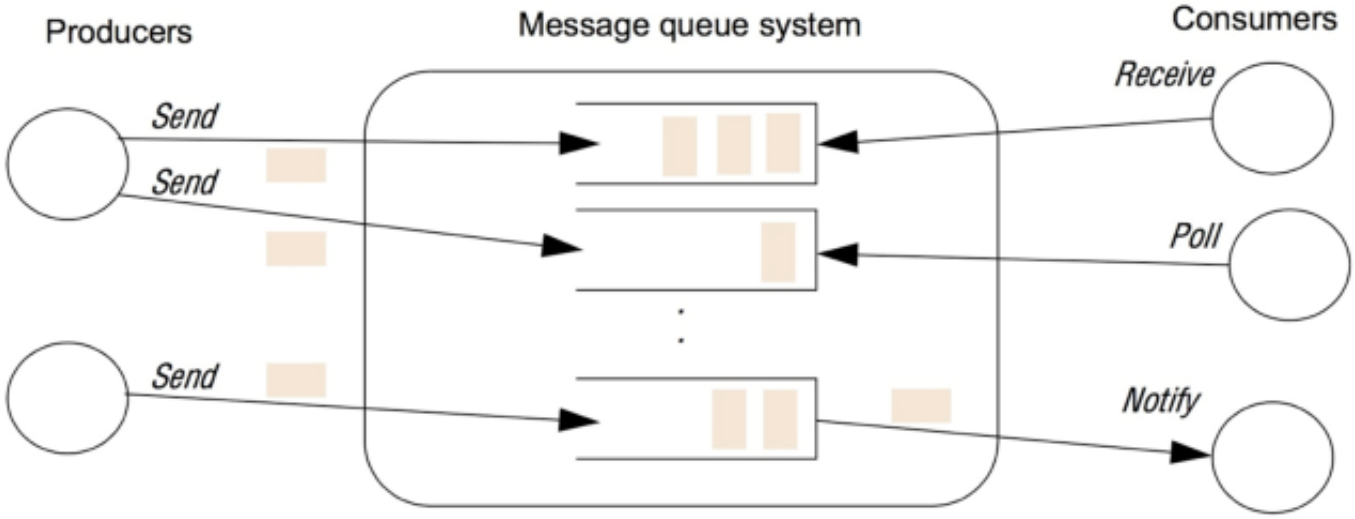
    procedura implementovaná ve vzdáleném procesu se volá jako by se nacházela v lokálním adresovém prostoru

    podpora modelu komunikace v modelu klient–server
  - ✓ *Remote method invocation*, RMI

    princip RPC aplikovaný na volání metod ve vzdálených objektech

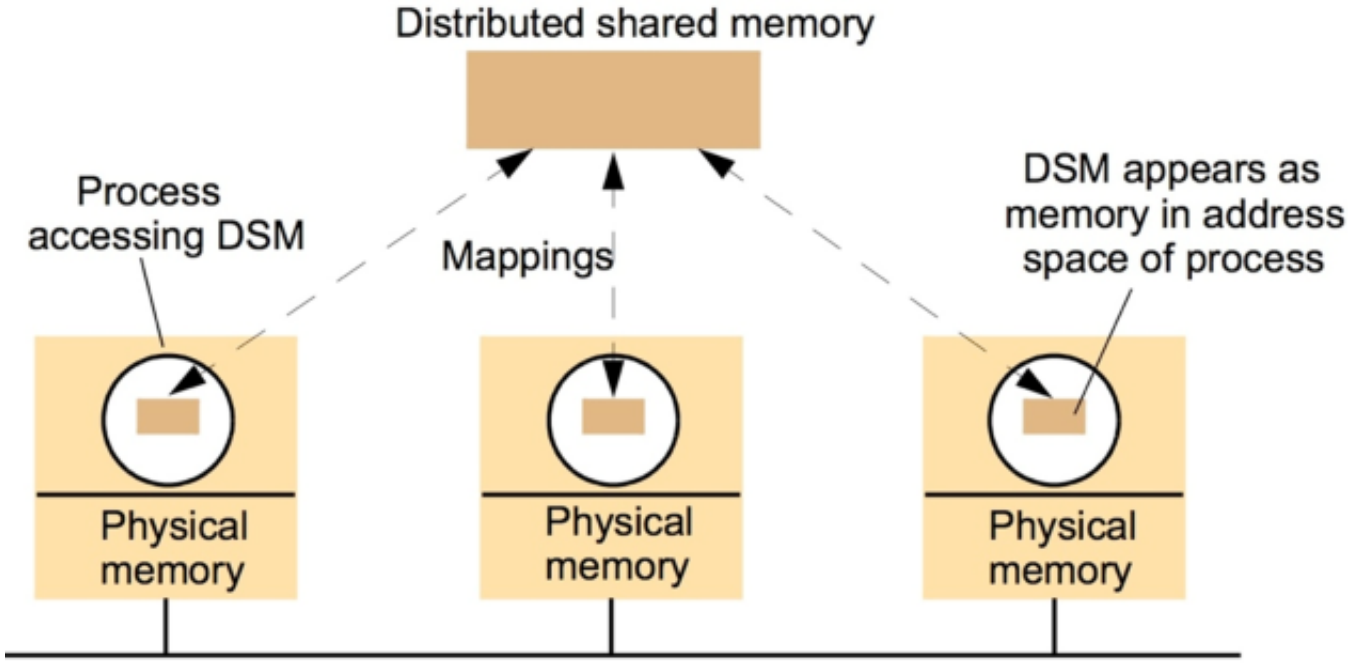
Architektury DS: komunikační paradigmata

- Nepřímá komunikace
  - ✓ komunikace ve skupině, *multicasting*,
    komunikace 1:*n*, rozesílání zpráv procesům ve skupině procesů
    (?? spolehlivost doručení všem procesům ve skupině,
    ?? konzistentnost pořadí doručení u všech příjemců, . . .)
  - ✓ *Message queues*, MQueues,
    p2p služba pro komunikaci mezi producenty a konzumenty

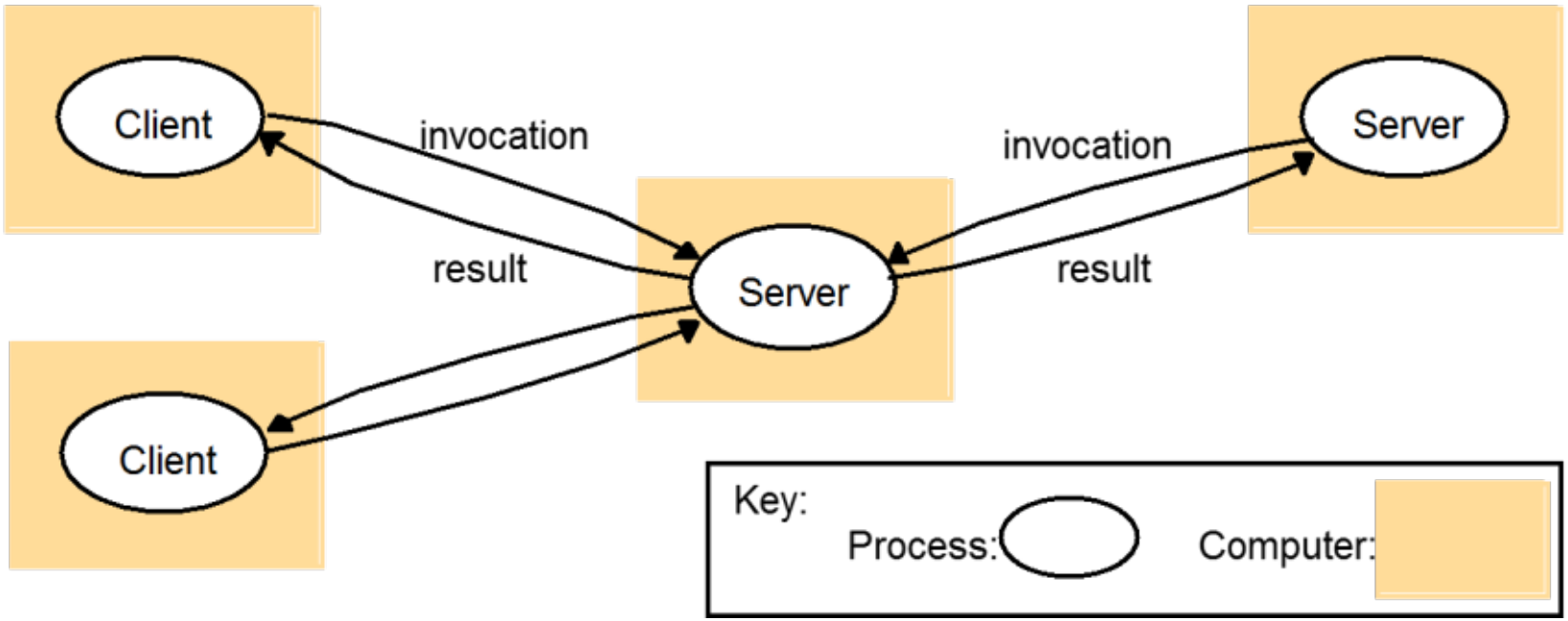


Architektury DS: komunikační paradigmata

- ✓ nástěnky, *Tuple spaces*
  nesynchronizované zavěšování, čtení a mazání zpráv na nástěnky
- ✓ distribuovaná sdílená paměť
  jeví se jako součást adresového prostoru
  každého z participujících procesů

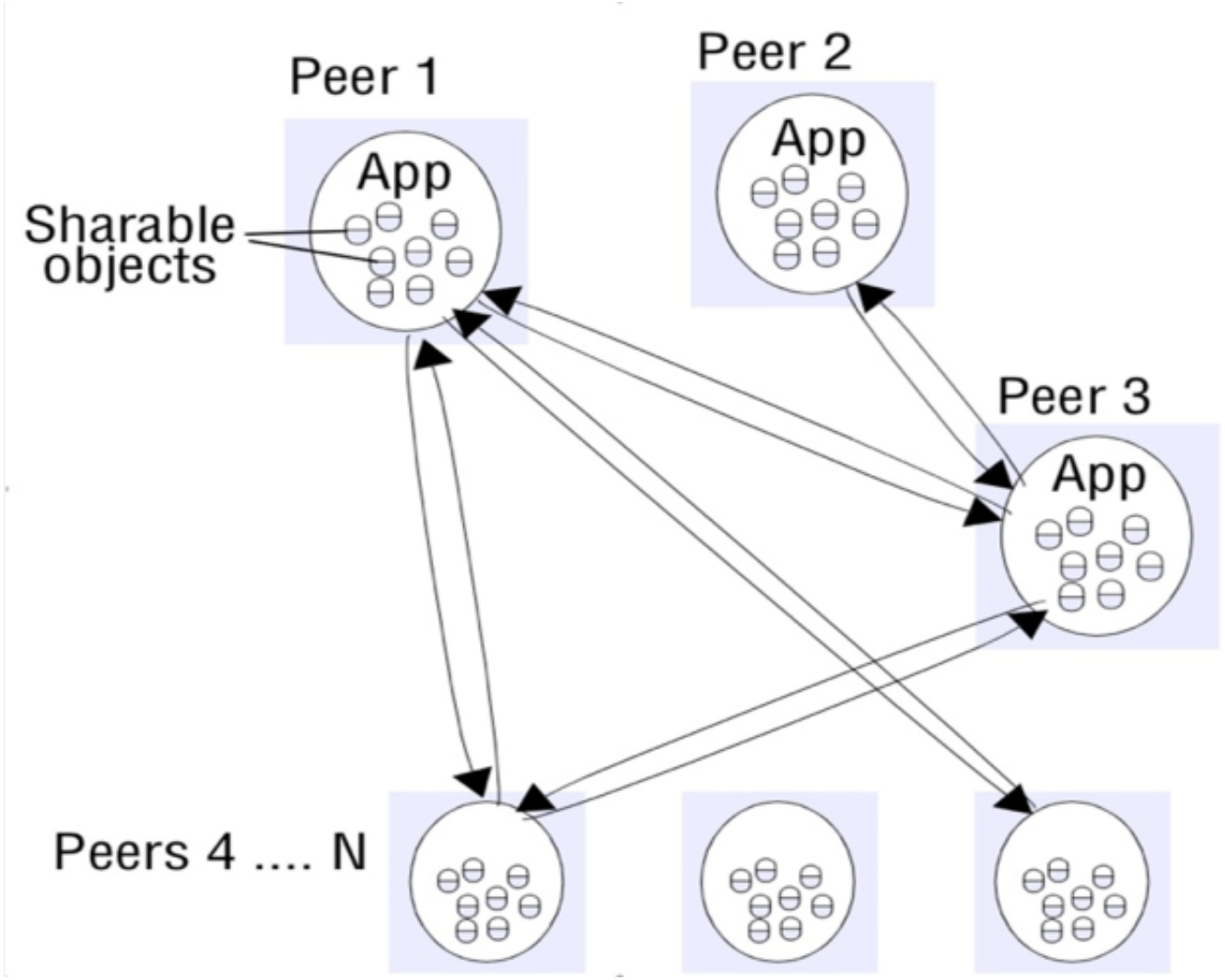


Architektury DS: klient–server

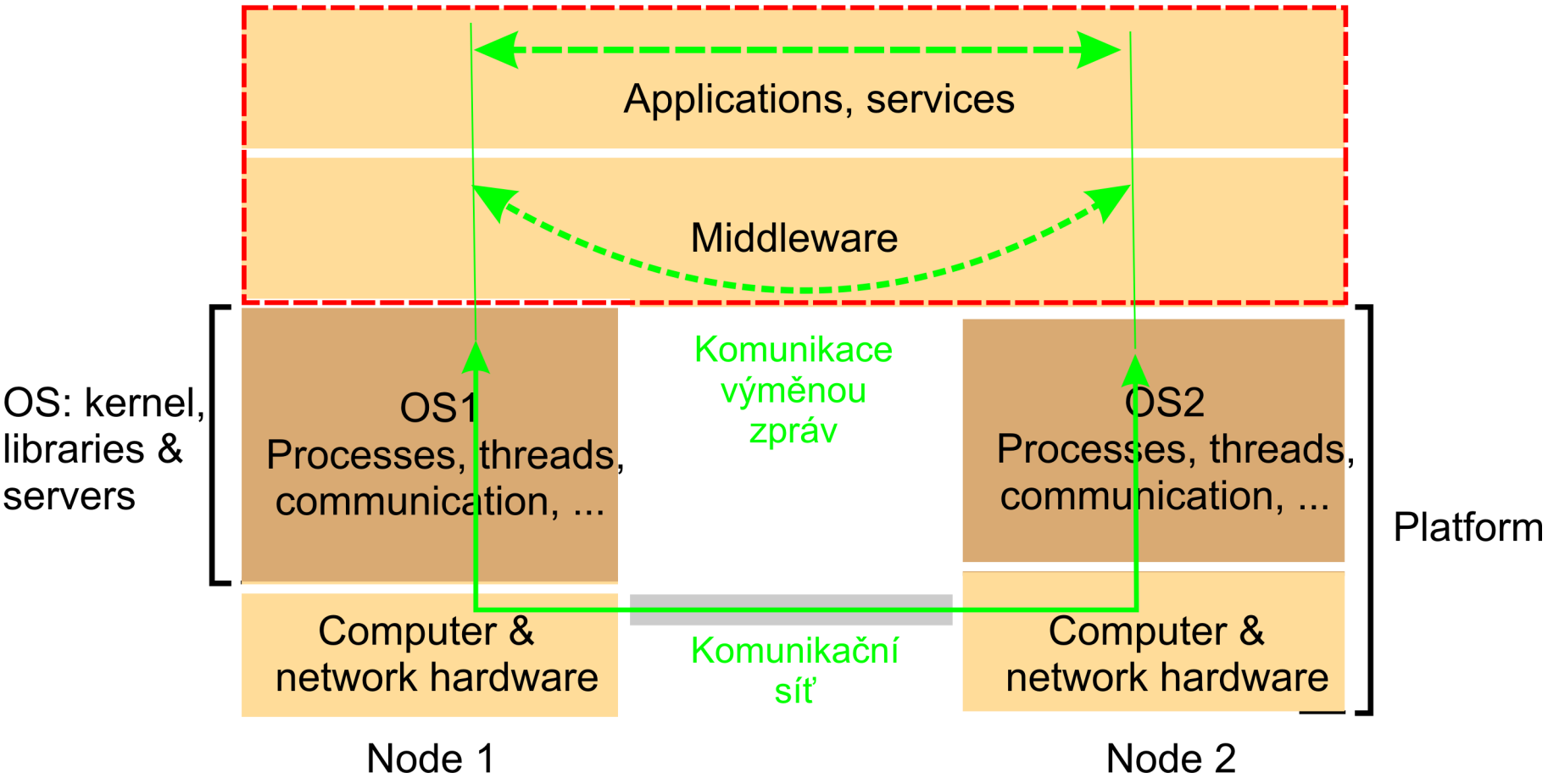


- Server může být klientem vůči jinému serveru
  - ✓ Web servery jsou klienty DNS serverů
  - ✓ Proxy server klientovovi zprostředkovává styk s více servery

Architektury DS: peer2peer



Typové hierarchické uspořádání distribuovaného systému



Distribuovaný systém, distribuovaný algoritmus

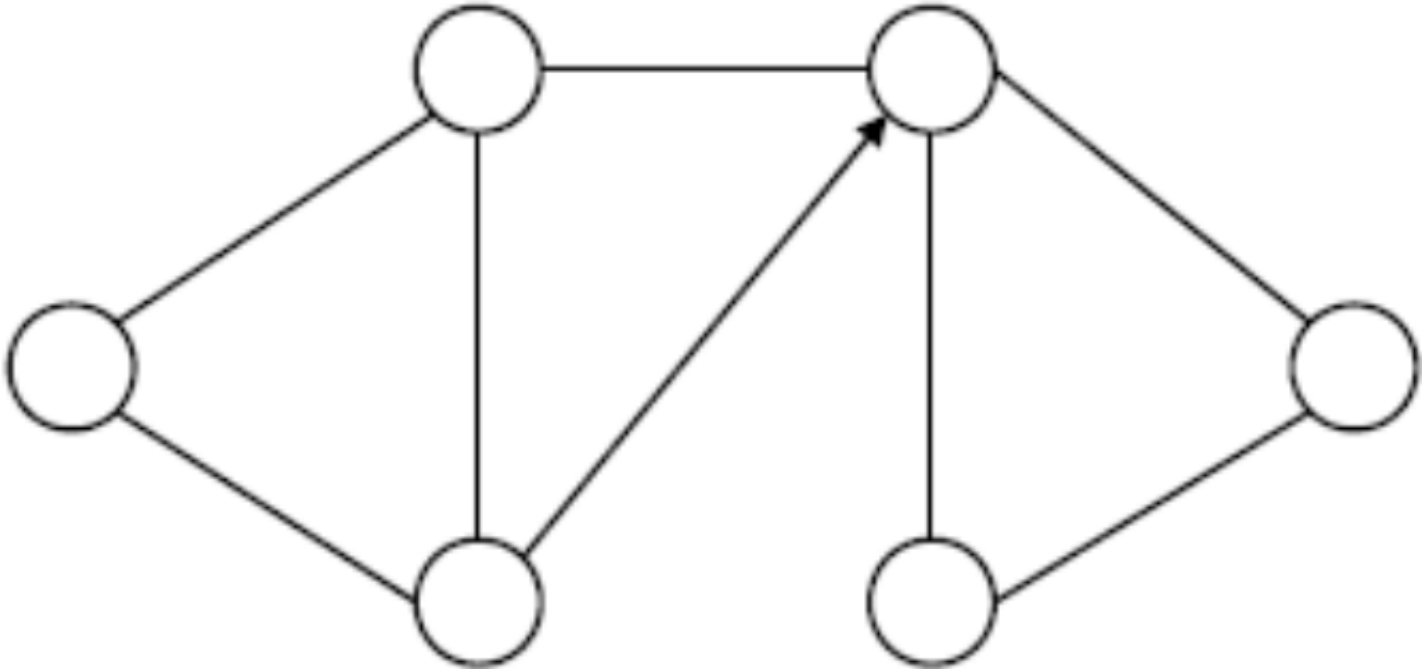
- Distribuovaný systém, DS
  - ✓ množina autonomních výpočetních komponent vzájemně propojených nějakou komunikační strukturou
- Distribuovaný algoritmus, DA
  - ✓ agregace algoritmů běžících v jednotlivých komponentách DS
- Připomenutí pojmu algoritmus
  - ✓ přesný návod či postup, kterým lze vyřešit daný typ úlohy
  - ✓ teoretický princip řešení jisté třídy obdobných problému
  - ✓ skládá se z konečného počtu jednoduchých (elementárních), jednoznačně a přesně definovaných kroků
  - ✓ končí, poskytuje výsledek, v (libovolně velkém) konečném počtu kroků

Rysy vymezující chování DS (model DS)

- Topologie
  - ✓ forma propojení komponent DS (uzlů, počítačů, . . .)
  - ✓ směrovost informačních toků komunikační infrastrukturou
- Plánování, princip řízení
  - ✓ synchronní, asynchronní
  - ✓ lokální algoritmy se startují v jistých časech a běží jistou rychlostí
- Komunikace, výměna dat
  - ✓ synchronní, asynchronní
  - ✓ výměnou zpráv (sdílení paměti je typické pro paralelní algoritmy)
  - ✓ zprávy se doručují v jistých časech, v jistém pořadí (např. FIFO)
- Spolehlivost, poruchovost
  - ✓ možnost/vyloučenost výpadků procesorů a/nebo komunikačních spojů

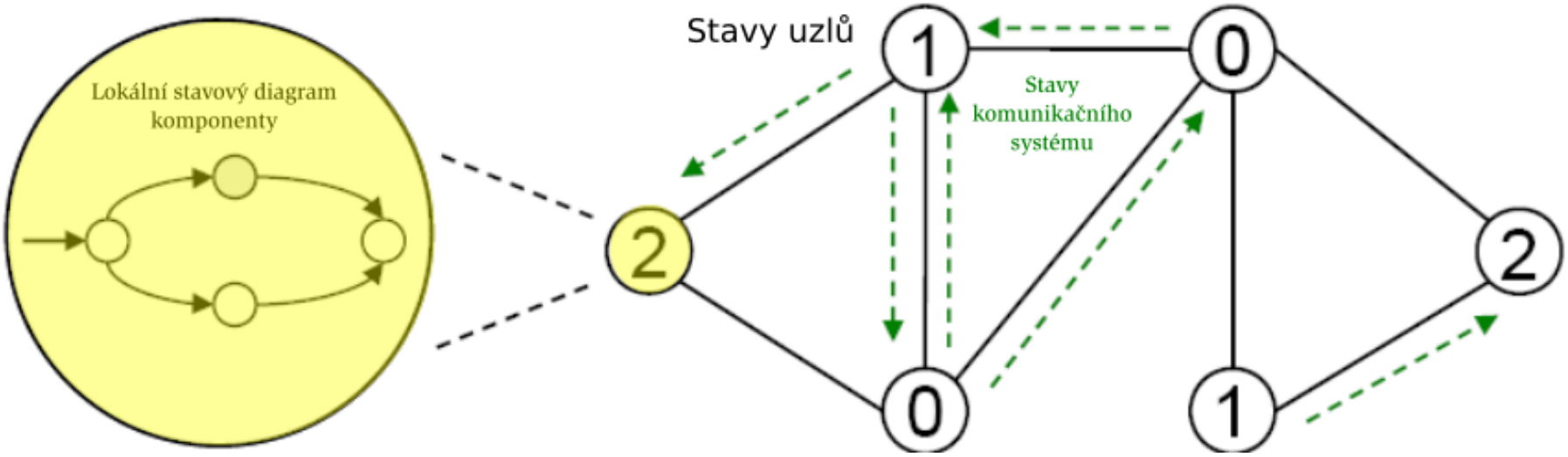
Topologie DS

- počty a formy propojení procesorů, komponent DS
- reprezentace pomocí grafu
  - ✓ uzly – procesory
  - ✓ hrany – komunikační kanály, mohou mít omezenou směrovost



Chování DS

- komponenty DS mění svůj lokální stav a vzájemně interagují
- DS se postupně nachází v jednotlivých konfiguracích DS
- každá jednotlivá konfigurace DS je určená
  - ✓ okamžitými lokálními stavy všech komponent
  - ✓ stavy komunikačních média (např. obsahy komunikačních bufferů)



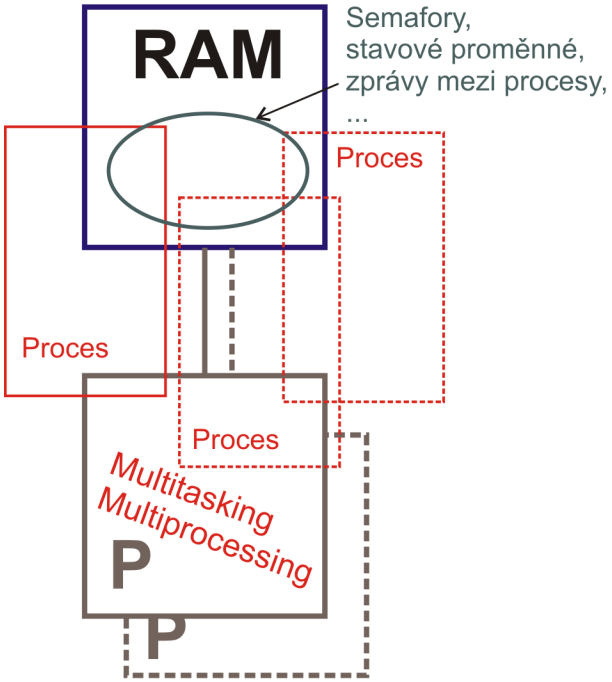
Distribuovaný výpočet

- Provedení výpočtu podle DA v DS
- Posloupnost diskrétních událostí, přechodů mezi stavy DS
- Každá událost je atomickou změnou konfigurace DS
- Všechny lokální výpočty jsou deterministické
  - ✓ K nedeterminismu dochází tehdy, pokud se připustí volitelné plánování lokálních kroků a/nebo komunikační události typu ztráta zprávy, změna pořadí doručení zpráv, . . .

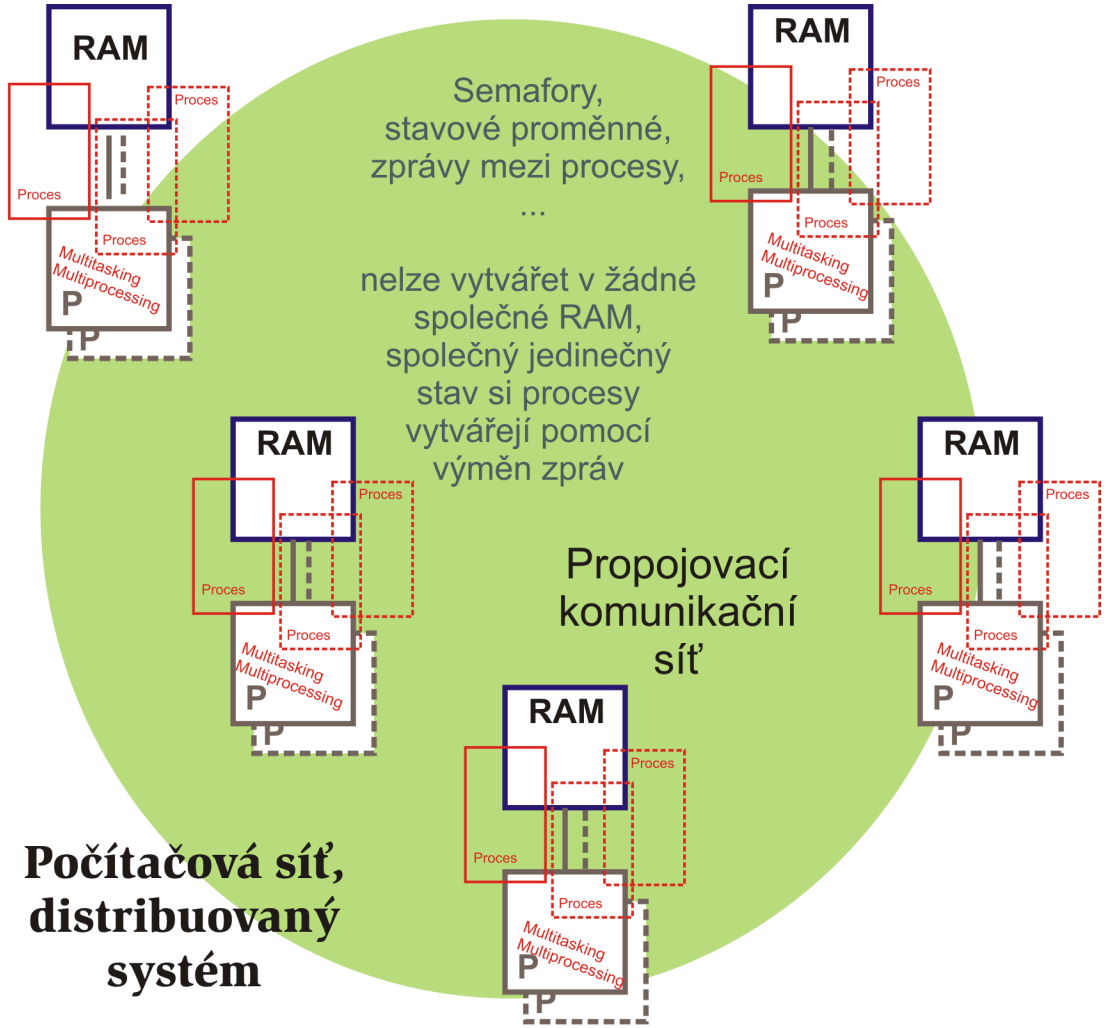
Plánování, časový model DS

- DS se synchronní výměnou zpráv
  - ✓ výpočty se odehrávají v synchronních krocích
  - ✓ běh všech komponent DS se řídí tiky globálních hodin
  - ✓ v každém kroku mohou komponenty vyslat a/nebo přijmout zprávu a provést lokální výpočet a příslušně změnit lokální stav
    - Mooreho synchronie – vyslané zprávy determinuje lokální stav
    - Mealyho synchronie – vyslané zprávy determinuje lokální stav a přijaté zprávy (dále neaplikujeme)
  - ✓ časový interval přenosu zpráv je nenulový, ale vždy menší než interval mezi dvěma tiky globálních hodin
- DS s asynchronní výměnou zpráv (*předmět našeho studia*)
  - ✓ globální čas neexistuje
  - ✓ čas v komponentách DS je řízený jejich lokálními hodinami, které běží libovolnými vzájemně nezávislými rychlostmi
  - ✓ doby přenosů zpráv jsou neomezené

Distribuovaný systém, DS, pro zpracování informací



Počítač - monoprocesor (příp. paralelní systém, multiprocesor)



Počítačová síť, distribuovaný systém

Distribuovaný systém, DS, pro zpracování informací

Pracovní definice DS pro účel našeho (orientačního) výkladu:

- ✓ Systémy typu Internet, síťové vestavěné systémy, ...
- □ Kooperace dějů v DS
  (procesů běžících v různých uzlech, počítačích)
  se děje pouze na základě výskytu asynchronních událostí
- □ Asynchronní události se odvozují
  z asynchronních výměn zpráv mezi procesy
- □ Uzly jsou propojené (případně i nespolehlivým)
  asynchronním komunikačním systémem pro výměnu zpráv
- □ Procesům jsou bezprostředně dostupné pouze
  lokální stavy uzlů
- □ Globální stav DS si musí každý proces odvodit
  pomocí vzájemně vyměňovaných zpráv

Distribuovaný systém, DS, pro zpracování informací

- Zkušený programátor musí mít rozumět algoritmům v DS
- Na bakalářské úrovni
  - ✓ jsou studenti školeni v algoritmickém myšlení vesměs pro nedistribuované prostředí
  - ✓ lze absolvovat kurzy orientované na základní algoritmy, tj. na vyhledávání, třídění, rozpoznávání vzorů, grafové úlohy, ...
  - ✓ lze se dozvědět jak rozpoznat tyto podproblémy v rámci svých programů a jak je lze účinně řešit
- Distribuovaný systém je propojená kolekce autonomních procesů cílených na
  - ✓ Výměnu informací (ve WAN)
  - ✓ Sdílení zdrojů (v LAN)
  - ✓ Paralelizaci pro zvýšení výkonnosti
  - ✓ Replikace pro zvýšení spolehlivosti, ...

Distribuovaný systém, DS, pro zpracování informací

- Výhody DS – vzrůst spolehlivosti, výkonu, škálovatelnost, . . .
- Jestliže procesy DS při dosahování společného cíle kooperují, pak musí mít přístup ke globálnímu stavu DS a přitom:
  - ✓ Pokud může kterýkoliv z procesů kdykoliv vypadnout/selhat, ostatní procesy v DS se o výpadku explicitně nedozví
  - ✓ Výměna zpráv může selhat a i když neselže, vesměs se odehrává v nepředvídatelném čase
  - ✓ A navíc – je nedosažitelný jednotný běh reálného času.

    V jednotlivých komponentách DS je běh času řízený lokálními hodinami běžícími ve těchto komponentách DS
  - ✓ Jednotný běh času si musí procesy DS modelovat (místo reálného běhu času se spoléhají na simulovaný logický čas)

Paralelní vs. distribuovaný systém

- Distribuované systémy se liší od paralelních, (multiprocesorových) systémů ve třech aspektech
  - ✓ Neznalost globálního stavu
    Proces obvykle nezná lokální stavy ostatních procesů.
  - ✓ Nedostupnost globálního časového rámce
    Události nelze uspořádat podle jejich času výskytu
  - ✓ Nedeterminismus
    Souběh procesů je nedeterministický,
    opakovaný běh těchže procesů může generovat různé výsledky.

Typy distribuovaného prostředí podle běhu času

Asynchronní distribuované systémy

- ✓ Nejobecnější případ distribuovaného systému (DS)
- ✓ Předem se neznají (≡ mohou být libovolné)
  - rychlosti běhu procesů,
  - doby zpoždění přenosů zpráv,
  - drifty reálných hodin v jednotlivých uzlech DS
- ✓ Nepoužívá se žádná synchronizace na bázi centrálního reálného času
- ✓ Nic nelze předpokládat o časových intervalech řešení běhů procesů
- ✓ Kooperace procesů musí řešit algoritmy hnanými výměnou zpráv mezi procesy řízenou vhodným protokolem
- ✓ Odeslání a příjem zprávy reprezentují nezávislé události, až na kauzalitu ,,událost přijetí následuje po události odeslání''
- ✓ Smutná pravda: Ne všechny problémy distribuovaného počítání mají asynchronní řešení.

Typy distribuovaného prostředí podle běhu času

- Synchronní distribuované systémy
  - ✓ Vesměs specializované systémy
  - ✓ Znají se horní a dolní meze
    - rychlostí běhu procesů, tj. každého kroku každého procesu
    - dob zpoždění přenosů zpráv,
    - driftů reálných hodin v jednotlivých uzlech DS
  - ✓ odeslání a příjem zprávy je koordinováno tak,
    aby ty dvě akce tvořily jednu událost,
    zpráva se odesílá pouze pokud je její cíl připraven ji přijmout.
- pseudoasynchronní (Internet) – v podstatě cíl našeho studia
  - ✓ asynchronní prostředí s možností
    detekce nesplnění časového limitu doručení zprávy

Typy distribuovaného prostředí podle spolehlivosti

- systémy bez poruch × systémy s výpadky (s poruchami)
  - ✓ systémy s výpadky procesů – ostatní procesy mohou výpadek zjistit pouze podle chybějící odpovědi na invokační (vyzývací) zprávu
  - ✓ systémy s výpadky komunikací – ztráty zpráv může příjemce detekovat časovými limity (a příp. získávat smluvenou implicitní hodnotu)
- systémy s libovolnými (Byzantinskými) poruchami
  - ✓ nejhorší možné chyby
  - ✓ procesy mohou chybně nastavovat svá data,
  - ✓ procesy mohou na výzvu odpovídat lživě,
  - ✓ zprávy mohou během přenosu měnit obsah,
  - ✓ mohou se generovat falešné zprávy,
  - ✓ zprávy se mohou ztrácet, dublovat, . . .
  - ✓ pokud má DS s byzantinskými poruchami komponent řešící DA dospět k validnímu výsledku, musí být vytvořen jako odolný proti poruchám, může se v něm nacházet nejvýše jistý počet chybujících komponent

Předpoklady pro model použitý při našem studiu DA

- ✓ Komunikační síť je silně souvislá,

  každý uzel DS může komunikovat s každým jiným uzlem DS
- ✓ Procesy komunikují pouze výměnou zpráv
- ✓ Komunikace výměnou zpráv je pseudo-asynchronní,

  zdržení zprávy v kanálu může být libovolné, vždy je konečné, výpadky komunikačního přenosu lze detekovat hlídáním časových limitů
- ✓ Komunikační kanály zprávy neztrácejí, neduplikují, nemodifikují
- ✓ Přenos zpráv komunikačními kanály se řídí politikou FIFO
- ✓ Procesy nepadají
- ✓ Každý proces zná pouze své sousedy, nikoli topologii celého DS
- ✓ Procesy mají jedinečné identifikátory (pid)

Připomenutí základních pojmů

Operační systém je . . .

- . . . správce prostředků
  - ✓ spravuje a přiděluje zdroje systému, eviduje jejich využívání,. . .
- . . . řídicí program
  - ✓ řídí bezpečné provádění uživatelských programů a operací I/O zařízení
- . . . poskytovatelem problémově orientované abstrakce bázových fyzických prostředků
  - ✓ procesory, operační paměť, komunikační nástroje, vnější pměti, . . .
  - ✓ programátorovi nabízí k použití formou rozhraní volání systému
    - spíše soubory a záznamy než diskové bloky a vystavovací mechanismus disku
    - spíše schránky (sockets) než přímý přístup k síti
    - spíše procesy a vlákna než procesory, paměťový prostor, . . .

Základní definice související s OS

- jádro OS
  - ✓ logické rozšíření rysů hardware + poskytované služby
  - ✓ vše mimo jádro je řešeno formou procesů
  - ✓ jádro může využívat speciální rysy hardware nedostupné procesům
- mikrojádro OS
  - ✓ minimalistická varianta jádra OS v některých architekturách OS
  - ✓ typicky zabezpečuje
    - správu přerušení,
    - správu paměti,
    - správu procesorů a
    - správu procesů a komunikaci mezi procesy předáváním zpráv – *Interprocess communication* (*IPC*)
  - ✓ ostatní funkce jádra se přesouvají do „procesové“ oblasti (drivery, služby systému souborů, virtualizace paměti, . . .)
  - ✓ mezi procesy se komunikuje předáváním zpráv

Generické komponenty OS

- Správa procesorů
- Správa procesů a vláken
- Správa (hlavní, operační) paměti
- Správa souborů
- Správa I/O systému
- Správa vnější (sekundární) paměti
- Networking (síťování), distribuované systémy
- Systém ochran
- Interpret příkazů
- Systémové programy
  - ✓ stavové informace, podpora jazyků, podpora komunikace, manipulace se soubory, aplikační systémy (databáze, . . .)

Tou či onou formou jsou implementované v každém OS

Generické komponenty OS, popis

- Správa procesorů
  - ✓ dispečer, krátkodobý plánovač běhu procesů / vláken
- Správa procesů a vláken
  - ✓ vytváření a rušení procesů a vláken
  - ✓ pozastavování a obnova běhu procesů a vláken
  - ✓ mechanismy synchronizace procesů a vláken
  - ✓ mechanismy komunikace mezi procesy a vlákny
- Správa (hlavní, operační) paměti
  - ✓ zobrazování LAP do FAP
  - ✓ virtualizace paměti
  - ✓ sledování které části FAP jsou používány a kterými procesy
  - ✓ mechanismy přidělování a uvolňování paměti (FAP) na žádost
  - ✓ střednědobé plánování – potláčení / obnova běhu vybraných procesů

Generické komponenty OS, popis, 2

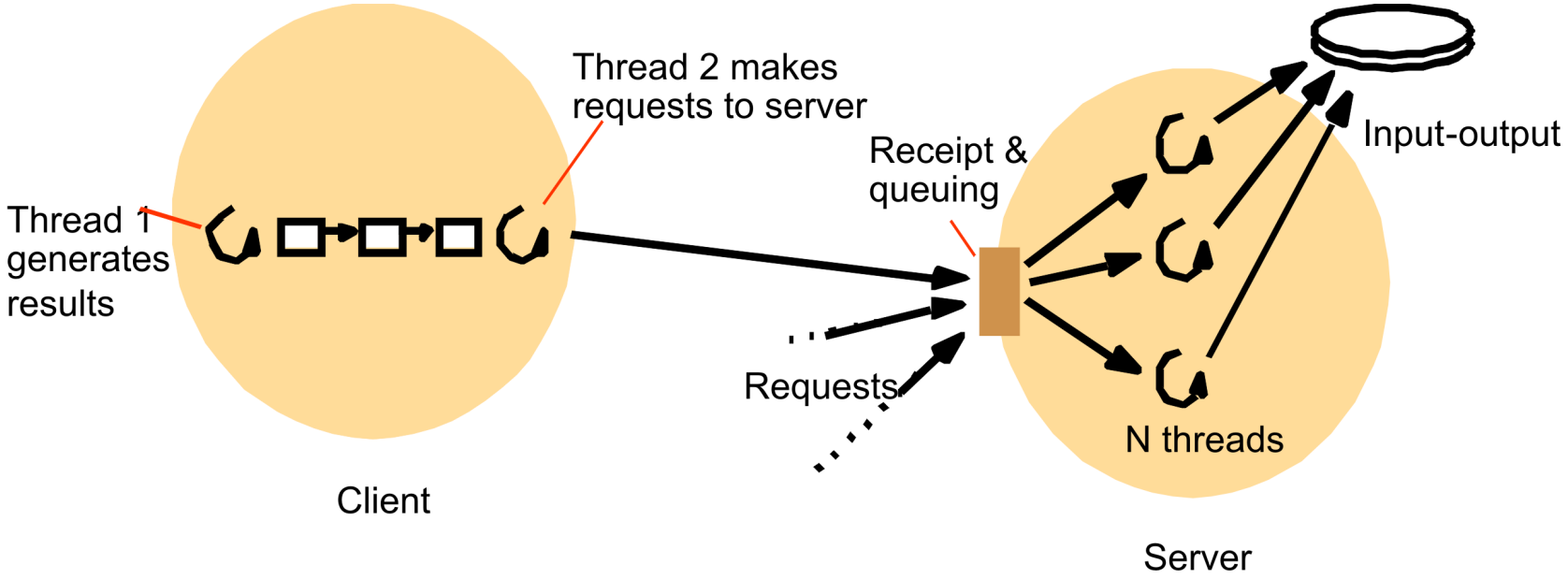
- Správa I/O systému
  - ✓ správa vyrovnávacích paměti
  - ✓ podpora univerzálního rozhraní ovladačů (driverů)
  - ✓ ovladače
- Správa vnější (sekundární) paměti
  - ✓ správa volné paměti
  - ✓ přidělování paměti
  - ✓ plánování optimálního pořadí (diskových) operací
- Správa souborů (systém souborů, *File System*)
  - ✓ manipulace s kolekcemi dat na vnějších pamětech – se soubory
  - ✓ vytváření, rušení, katalogizace, archivace, obnova, . . . souborů

Generické komponenty OS, popis, 3

- Networking (síťování), distribuované systémy
  - ✓ kooperace procesorů nesdílejících ani paměť ani procesor (každý procesor má svou lokální paměť a hodiny)
  - ✓ propojení procesorů (uzlů, počítačů) komunikační sítí
  - ✓ nástroje pro sdílení zdrojů (distribuovaný systém souborů, . . .)
- Interpret příkazů
  - ✓ rozhraní uživatele na služby operačního systému
- Systém ochran
  - ✓ mechanismy pro řízení přístupu procesů a uživatelů ke zdrojům
  - ✓ rozlišování autorizovaných a neautorizovaných přístupů ke zdrojům
  - ✓ specifikace vnucovaných ochranných opatření
  - ✓ nástroje pro prosazování ochranných opatření

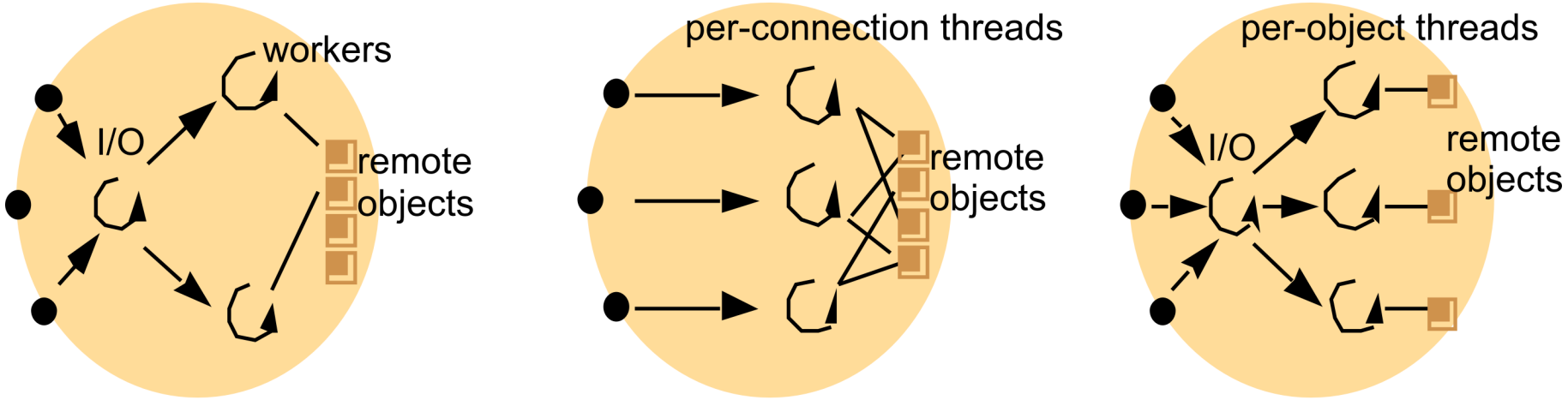
Vlákna

- Proces může definovat více aktivit proveditelných souběžně
  - ✓ Např. server může obsluhovat více požadavků klientů souběžně



Příklady architektur klient-server s více vlákny

- Thread-per-request Architecture
  - ✓ I/O vlákno serveru vytvoří nové vlákno (*worker*) pro řešení každého nového požadavku klienta na zpřístupnění vzdáleného objektu
  - ✓ po splnění služby se vlákno *worker* samo zruší
  - ✓ vlákna nesdílí žádnou frontu – maximální propustnost
  - ✓ časté vytváření / rušení vláken – vyšší režie

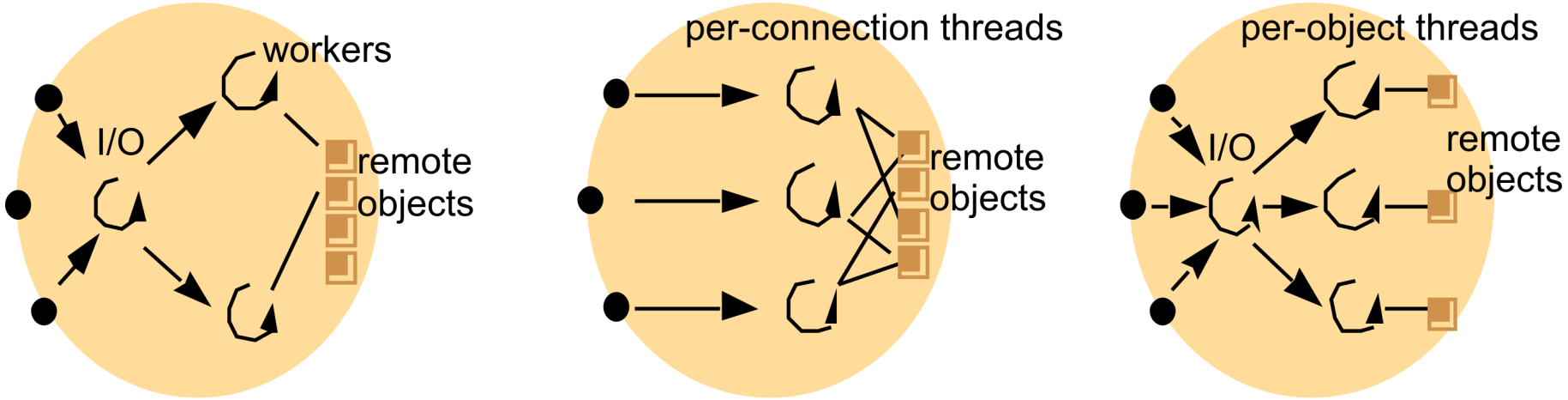


a. Thread-per-request b. Thread-per-connection c. Thread-per-object

Příklady architektur klient-server s více vlákny

Thread-per-connection Architecture

- ✓ Server vytvoří nové vlákno pro každé spojení s jedním klientem a toto vlákno řeší požadavky daného klienta sekvenčně
- ✓ po uzavření spojení s klientem se vlákno zruší
- ✓ menší režie než v případě *Thread-per-request Architecture*
- ✓ potenciálně niší propustnost díky frontování požadavků



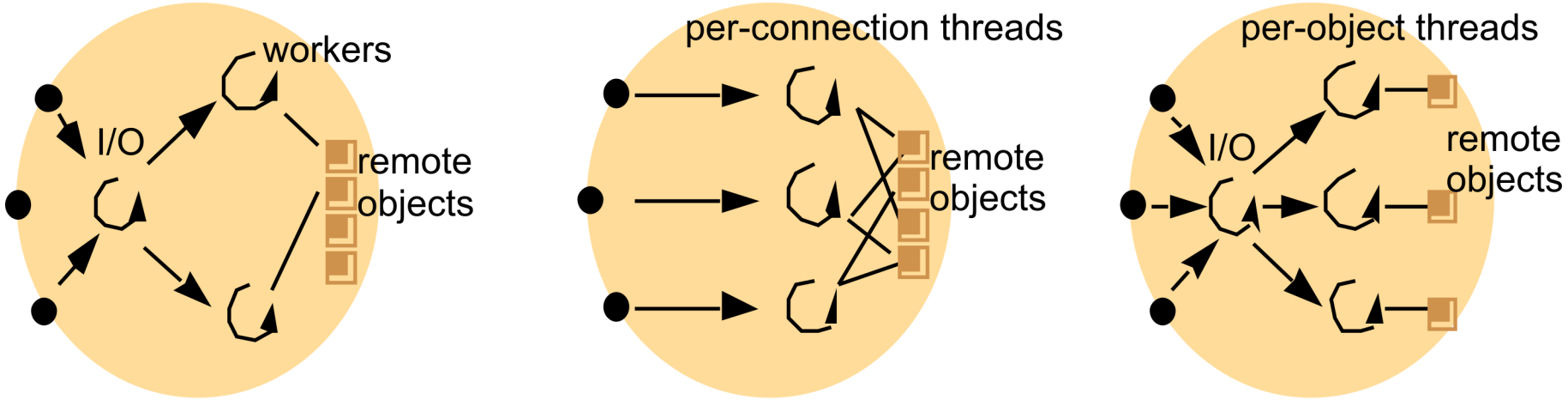
a. Thread-per-request

b. Thread-per-connection

c. Thread-per-object

Příklady architektur klient-server s více vlákny

- Thread-per-object Architekture
  - ✓ Každý zpřístupňovaný objekt serveru je obsluhovaný samostatným vláknem
  - ✓ I/O vlákno přijímá požadavky klientů na zpřístupnění objektů
  - ✓ požadavky na týž objekt se řadí do fronty na objekt
  - ✓ vlákno se zruší při zrušení objektu



a. Thread-per-request b. Thread-per-connection **c. Thread-per-object**

Počítačový systém, komponenty

- hardware
  - ✓ bázové výpočetní zdroje (CPU / procesor, paměť, I/O zařízení)
    CPU – Central Processing Unit
- operační systém + middleware
  - ✓ řídí a koordinuje používání hardware různými aplikačními programy různých uživatelů a propojuje uživatele a jejich aplikace
- aplikační programy
  - ✓ definují způsoby kterými se používají zdroje systému pro řešení výpočetních uživatelských problémů (kompilátory, databázové systémy, video hry, byznys programy, . . .)
- uživatelé (lidé, ale také jiné stroje, jiné počítače, . . .)

Operační systém je . . .

- ✓ . . . program, který funguje jako spojka mezi uživatelem počítače a jeho aplikačními systémy a hardware počítače

Cíle (povinnosti) OS

- ✓ řídit řešení uživatelských (aplikačních) programů
- ✓ poskytnout nástroje pro řešení problémů uživatelů (aplikací)
- ✓ učinit počítač snadněji použitelný
- ✓ vytvářet podmínky umožňující efektivně používat hardware počítače

Cíle (přání) uživatele

- ✓ služby poskytované OS lze používat pohodlně, tj. snadno zvládat
- ✓ OS je spolehlivý, bezpečný
- ✓ požadované služby poskytuje OS pohotově

Cíle provozovatale OS

- ✓ OS je snadno navrhnutelný, implementovatelný a udržovatelný
- ✓ OS je přizpůsobitelný, spolehlivý a bezchybný

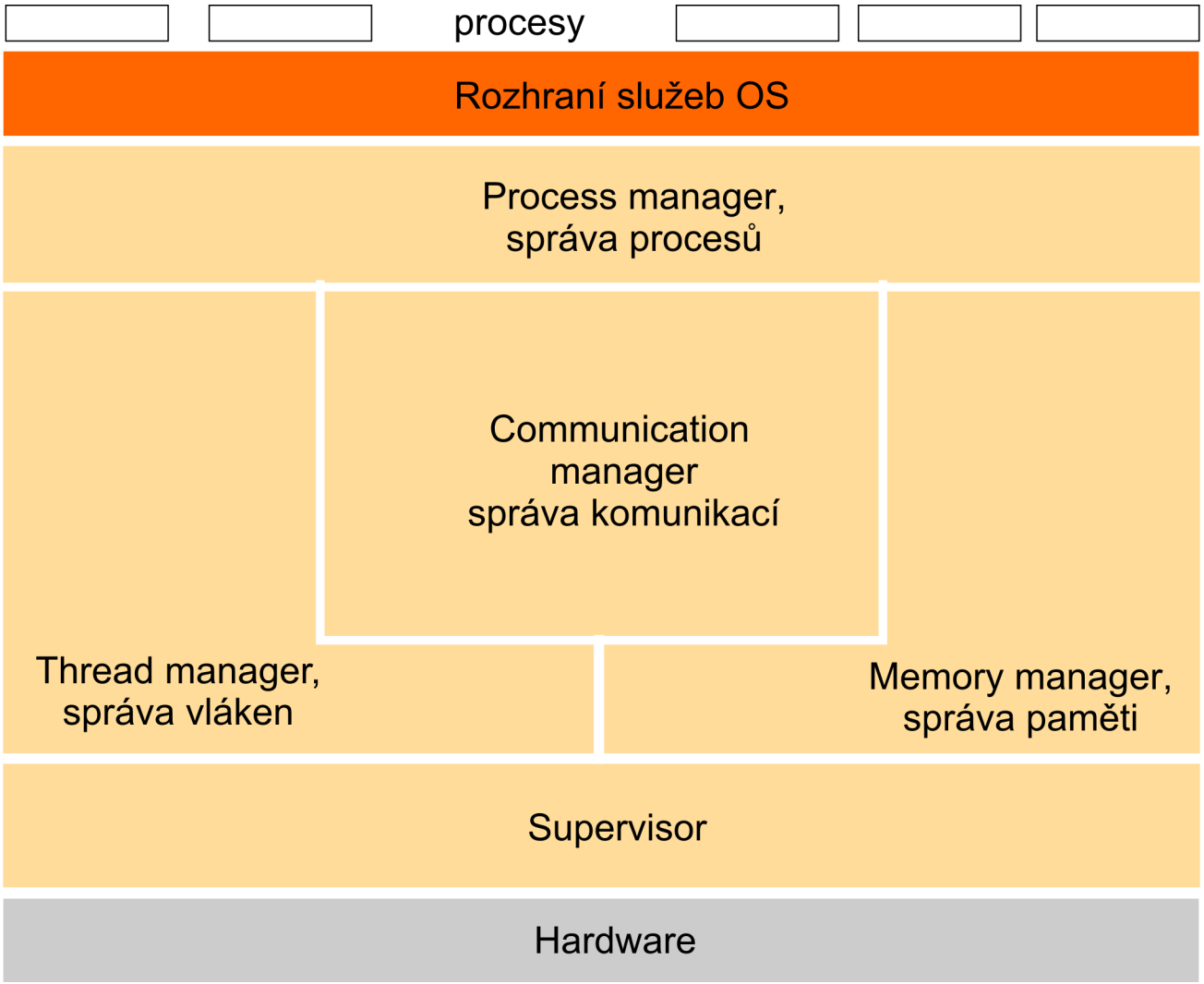
Hierarchické strukturování, pravidla

- Řeší se tím problém přílišné složitosti velkého systému
  - ✓ provádí se dekompozice velkého problému na několik menších, zvládnutelných problémů
- Úplná funkčnost systému se rozloží (uspořádá) do vrstev hierarchicky uspořádaných do jednotlivých úrovní
- Každá úroveň řeší konzistentní podmnožinu funkcí
- Nižší vrstva nabízí vyšší vrstvě (vyšším vrstvám) ,,primitivní“ funkce (služby)
- Nižší vrstva nemůže požadovat provedení služeb vyšší vrstvy
- Pro volání služeb se používají přesně definovaná rozhraní mezi vrstvami

Hierarchické strukturování, pravidla

- Funkčnost přiřazená vrstvě v jedné úrovni může být rozložená mezi více entit
- Pokud se zachová definované rozhraní vrstvy, lze vrstvu uvnitř modifikovat, aniž to ovlivní ostatní vrstvy
- Entity v jedné vrstvě komunikují pomocí výměn zpráv řízených pomocí přesně definovaných protokolů
- V případě distribuovaného systému mohou být entity v jedné vrstvě rozloženy do samostatných uzlů (výpočetních systémů) propojených komunikačním systémem zprostředkovávajícím výměnu zpráv mezi entitami

Funkcionalita OS, typové hierarchické uspořádání jádra OS



Funkcionalita OS, typové hierarchické uspořádání jádra OS

- Správa procesů
  - ✓ Vytváření, řízení, synchronizace, rušení procesů
  - ✓ Proces – jednotka správy zdrojů vč. adresního prostoru, vláken, ...
- Správa vláken
  - ✓ Vytváření, řízení, synchronizace, rušení vláken
  - ✓ Vlákno – jednotka plánování činností v rámci procesu
- Správa komunikací
  - ✓ Výměna zpráv mezi vlákny a procesy typicky v rámci 1 počítače
  - ✓ Komunikace se vzdálenými vlákny/procesy, tj. mezi různými počítači, vyžaduje dodatečné služby síťování poskytovanými typicky nad jádrem OS

Funkcionalita OS, typové hierarchické uspořádání jádra OS

- Správa paměti
  - ✓ správa fyzické a virtuální paměti
- Supervizor
  - ✓ Rozhraní na vlastnosti hardware
  - ✓ správa přerušení,
    řešení výjimek,
    ovladače,
    správa prostředků pro virtualizaci paměti,
    manipulace s registry počítače,
    . . .

Virtualizace výpočetního stroje

- V režimu multiprogramování (multitasking) může běžet na jednom stroji (počítači) současně více procesů (vláken)
  - ✓ OS každému procesu / vláknu poskytuje virtuální stroj dávající vlastníkům procesů iluzi vlastních ,,strojů''
  - ✓ Tento rys zahrnuje sytém ochran chránící každý proces před nežádoucí vzájemnou interferencí procesů
- Dva režimy činnosti počítače
  - ✓ Pro dosažení spolehlivé a efektivní virtualizace stroje jsou podporovány dva režimy činnosti počítače:
    - plný přístup ke všem zdrojů – pro (jádro) OS
    - omezený přístup ke zdrojům – pro procesy

Proces, vlákno

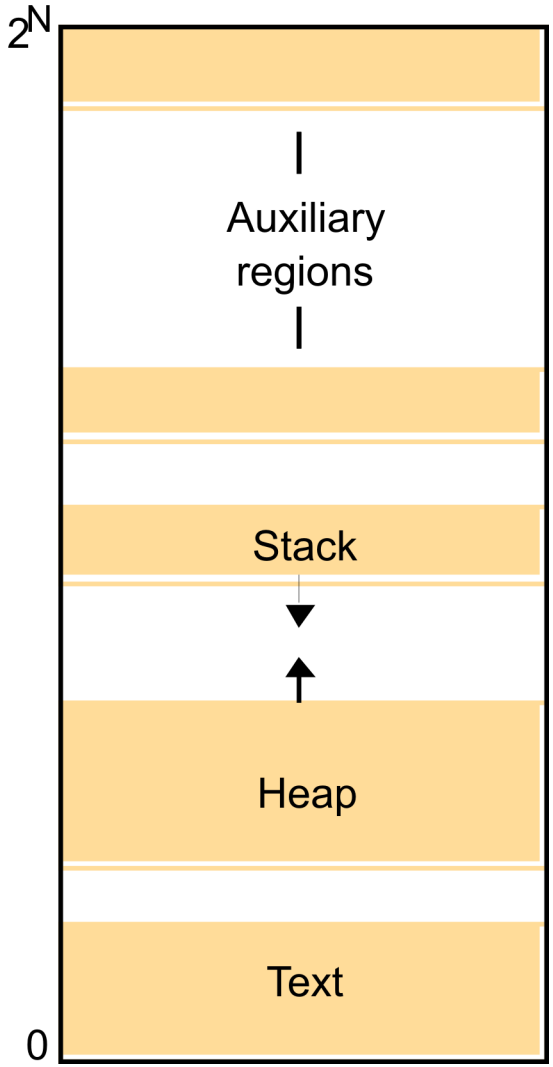
- Identifikovatelná / správní jednotka zpracování v počítači řízeném OS
- Typický proces zahrnuje identifikační a stavové informace, prostředí běhu a jedno nebo několik vláken
- Prostředí běhu procesu
  - ✓ Adresový prostor (logický adresový prostor procesu)
  - ✓ Nástroje pro synchronizaci a komunikaci vláken (semafory, . . .)
  - ✓ Komunikační rozhraní pro síťovou komunikaci (sockets, . . .)
  - ✓ Zdroje vyšší úrovně (soubory, okna, . . .)

  Mapování prostředí procesu na hardwarové/softwarové zdroje zajišťuje OS
- Vlákno
  - ✓ Abstrakce definovatelné sekvenční činnosti v rámci prostředí procesu
  - ✓ Identifikovatelná jednotka v případě vícevláknových procesů

Procesy

- Adresový prostor (Logický adresový prostor procesu)
  - ✓ správní jednotka virtuální paměti pro proces
  - ✓ poskytuje definované oblasti dostupné vláknům procesu
  - ✓ každá oblast
    - má svůj prostor (*extent*) vymezený typicky bázovou adresou a délkou
    - má r/w/x přístupová práva pro vlákna procesu
    - je typicky stránkováním zobrazovaná do fyzického adresového prostoru
    - může se v době běhu procesu zvětšovat / zmenšovat
- Adresový prostor v systémech s OS Unix
  - ✓ pevná nemodifikovatelná oblast text region s programem
  - ✓ halda, heap, oblast rozšiřitelná do vyšších adres virtuálního adresového prostoru procesu
  - ✓ zásobník, stack, oblast rozšiřitelná do nižších adres virtuálního adresového prostoru procesu
  - ✓ libovolný počet dalších pomocných oblastí (auxiliary region)

Adresový prostor v systémech s OS Unix



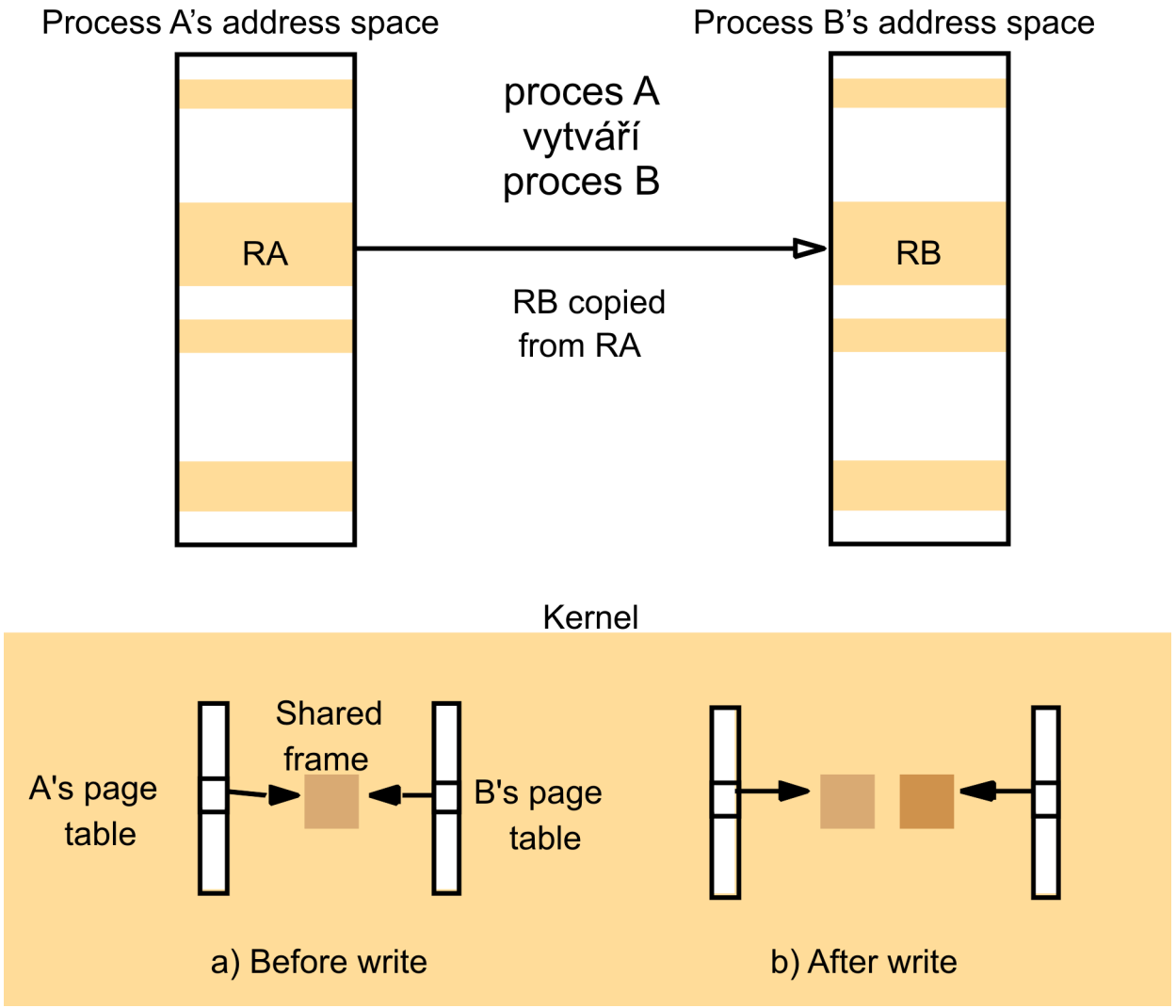
Více k oblastem adresového prostoru

- Halda
  - ✓ inicializovaná z části hodnotami ze souboru s binární verzí programu
  - ✓ dynamicky rozšiřovatelná, dynamicky definovaný obsah
- Zásobník
  - ✓ obecně vždy jeden pro každé vlákno
  - ✓ obsluha typu LIFO (last-in, first-out)
  - ✓ pro ukládání návratové adresy při volání funkce
  - ✓ pro ukládání lokálních proměnných funkce, předávání parametrů, . . .
- příklad pomocné oblasti – oblast souboru (File region)
  - ✓ Podpora zobrazování souborů z vnější paměti do virtuální paměti
- další příklad pomocné oblasti – sdílená paměťová oblasti (Shared Memory Regions)
  - ✓ Pro komunikaci sdílením paměti mezi procesy, mezi procesem a jádrem OS, pro umístění knihovních podprogramů, . . .

Vytvoření procesu

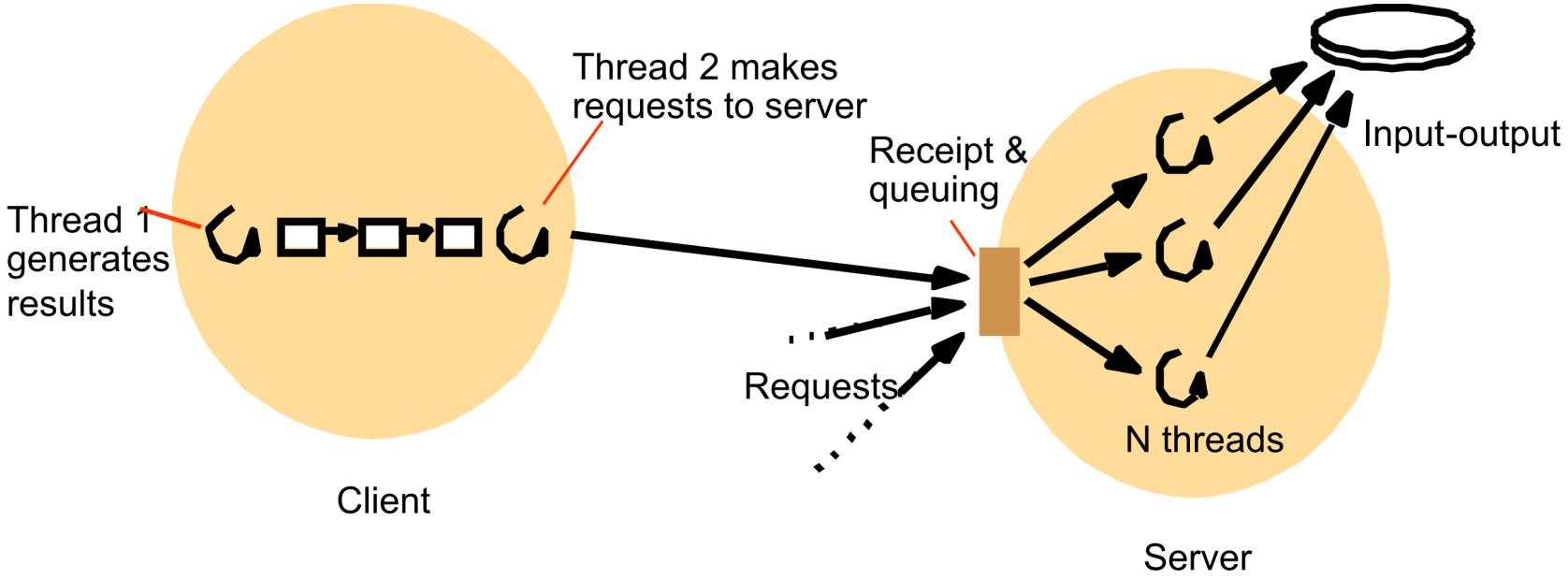
- Nově vytvářený proces požaduje vytvoření nového prostředí běhu procesu
- Tradiční forma vytvoření procesu unixového typu
  - ✓ služba OS *fork* vytvoří nové prostředí běhu kopií prostředí žádajícího procesu + sdělení novému procesu, že je potomkem vytvářejícího procesu, rodiče
  - ✓ služba *exec* umožní volajícímu procesu definovat nový program řídicí proces kopií textu programu z udaného souboru
- Vytvoření procesu způsobem *Copy on Write*
  - ✓ iniciálně nový proces sdílí obrazy stránek v rámcích původního procesu
  - ✓ při zápisu do stránky novým procesem se pro nový proces vytvoří samostatná kopie modifikované stránky

Vytvoření procesu způsobem *Copy on Write*



Vlákna

- Proces může definovat více aktivit proveditelných souběžně
  - ✓ Např. server může obsluhovat více požadavků klientů souběžně



Co je to vlákno

- sekvenční běh (části) programu řídicího proces
- v procesu může být definováno 1 nebo více vláken
  - ✓ většina programů je typu jednovláknový proces
  - ✓ jednovláknový proces =
    sekvenční aktivita nepřipouštějící žádný paralelismus
  - ✓ jednovláknový proces neumožňuje definovat více souběžně řešitelných aktivit
- Např. JVM (*Java Virtual Machine*) umožňuje, aby aplikační proces mohl aktivovat souběžné provádění více vláken
  - ✓ všechna vlákna sdílejí stejný adresový prostor jak pro program, tak i pro data
  - ✓ např. iniciální vlákno procesu potomka sdílí data rodiče

Použití vláken

- Univerzální nástroj pro všechny aplikace
  - ✓ od interaktivního kreslení po hry
  - ✓ např. souběh čtení klávesnice jedním vláknem v době, kdy jiné vlákno vykresluje obrázek
- Efektivní využití multiprocesorových počítačů
  - ✓ možnost skutečně pralelního běhu vláken na různých procesorech místo multitaskingu sdílejícího jediný procesor

Architektury operačních systémů

Monolitické jádro

- ✓ provádí v jednom celku všechny služby poskytované jádrem OS
- ✓ mohutný objekt, obtížně diferencovatelný, obtížně manipulovatelný
- ✓ minimální uplatnění modulovosti, vysoká provázanost funkcí
- ✓ původní řešení OS Unix
- ✓ v rámci jádra mohou fungovat serverovské procesy
  file server, networking, . . .
- ✓ služby jádra jsou typicky řešeny sekvenčně

Mikrojádro OS

- ✓ provádí většinu abstrakcí práce s pamětí, s procesy a vlákny a zajišťuje meziprocesovou komunikaci
- ✓ služby OS jsou poskytované servery běžícími jako procesy nad mikrojádrem
- ✓ nad mikrojádrem lze emulovat prostředí více OS souběžně, poskytuje se souběžně více různých rozhraní volání služeb (různých) OS

Monolitické jádro a mikrojádro OS