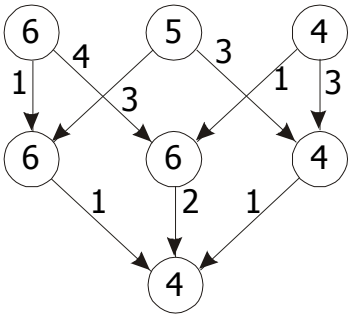
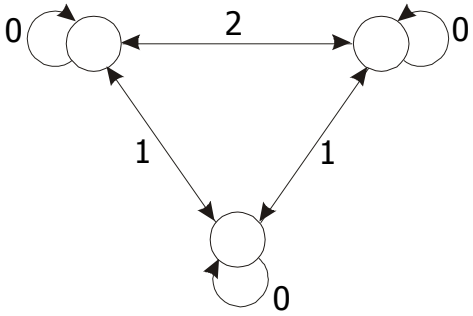
Pokročilé techniky urychlení výpočtu v distribuovaném prostředí

Co ovlivňuje rychlost výpočtu?

- Virtuální topologie, komunikační schéma, distribuované aplikace
- Prezentovaná síťová topologie
- Fyzická topologie sítě
- Výkonnost jednotlivých uzlů v síti
- Výpočetní model distribuované aplikace
  - Každý proces může běžet podle vlastního programu



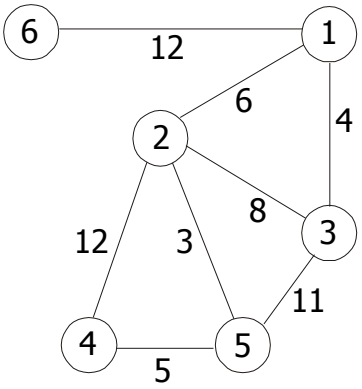
Precedence process model



Communication system model

Process	Cost on A	Cost on B
1	5	10
2	2	Infinity
3	4	4
4	6	3
5	5	2
6	Infinity	4

Computation cost



Communication cost

Řešení obecně

- Umístit procesy na uzly sítě takovým způsobem, aby běžely co nejrychleji a zároveň bylo komunikační zpoždění co nejmenší
  - Může se jednat i o protichůdné požadavky
- Zatížení a dostupnost uzlů se může měnit v čase

- Jsou tři typy úloh
  - S konečným časem výpočtu – distribuovaná aplikace má jenom něco spočítat a pak skončí
  - S teoreticky nekonečným časem výpočtu – distribuovaná aplikace poskytuje službu
    - Neplatí, že uzel musí být zcela vytížen výpočtem po celou dobu
  - Processing While in Transit – výpočet se nad daty provádí během jejich přenosu
    - Např. Active Network

Load-Sharing

- Předpokládá se prostředí pracovních stanic, které nemusejí být vždy plně vytíženy
- Jeden uzel se vyhradí jako master, kde se spustí aplikace
  - Ostatní uzly se označují termínem slave
- V okamžiku, kdy je master vytížen na maximum, zkusí se vyhledat nevytížený uzel

Červ

- jednotlivé procesy se mohou replikovat na nevytížených uzlech
- červ se skládá z několika segmentů – procesů
- počet segmentů je buď pevně stanoven, nebo se při pokročilejší implementaci stanovuje dynamicky podle okolností
- nápadně připomíná šíření virů
- červ musí být věrohodný pro uživatele pracovních stanic

- komponenty červa
  - inicializační kód ke spuštění master
  - inicializační kód ke spuštění slave
  - výpočetní program

- ze života červa
  - nalezení nevytíženého uzlu
    - žádný uzel nezná globální stav sítě a proto se každý segment musí postarat o nalezení volné stanice sám za sebe
    - lze hledat například pomocí broadcastu
    - hodila by se synchronizace, protože několik segmentů může soupeřit o stejný uzel
  - uvolnění uzlu
    - segment se musí postarat, aby uzel byl zase viditelný jako dostupný i pro ostatní

- kontrola růstu
  - čím větší červ, tím vyšší rychlost výpočtu, ale vznikají další problémy
    - rychlost nemusí růst lineárně
    - synchronizace
    - stabilita

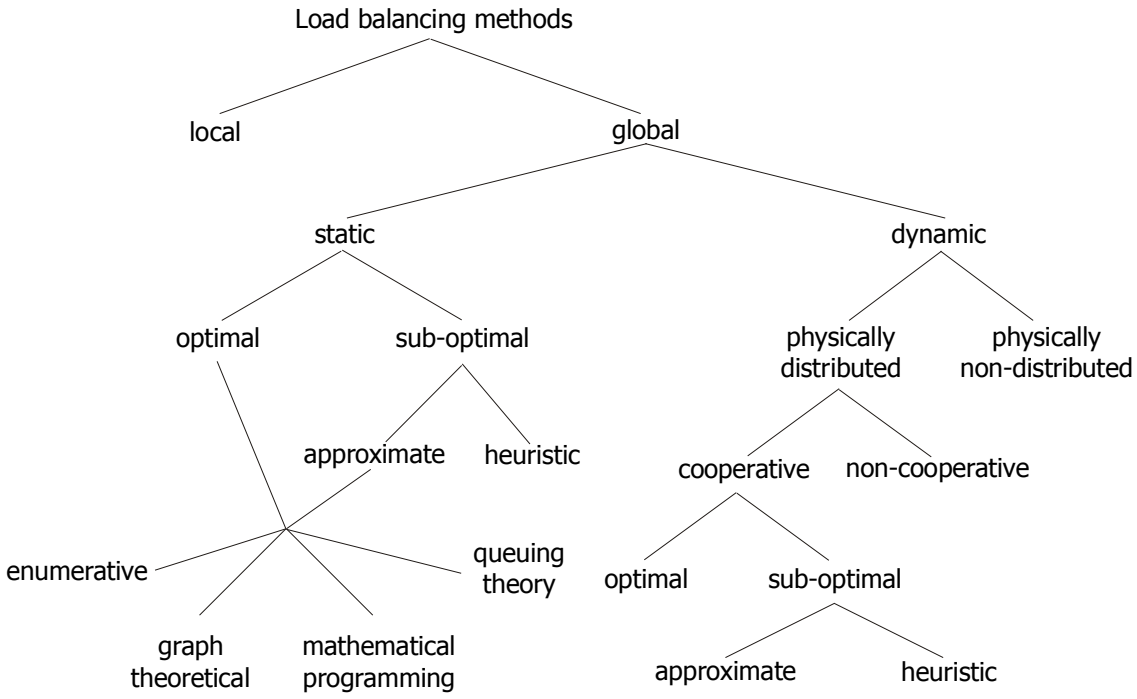
Condor

- snaží se o fér využívání všech uzlů
- pokud některý uzel selže, proces se restartuje jinde
- arbitr, který rozhoduje o tom, kde se spustí proces
  - sám hledá použitelné uzly
  - centralizované místo
    - možnost selhání

- checkpoints
  - procesy lze relokovat za běhu distribuované aplikace
    - používá se ukládání obrazu procesu v paměti, ne rekonstrukce stavu
      - uzly musejí být identické
      - co s otevřenými soubory?
  - checkpoint se ukládá periodicky
    - pokud selže výpočet, použije se poslední checkpoint
  - pre-emptivnost procesů je implementována pomocí checkpointů

Load-Balancing

- na rozdíl od load-sharingu se předpokládá, že celá síť je dostupná pro výpočet
- existuje několik různých metod, které se liší použitelností, účinností, náročností na zdroje (paměť, procesor, …), přesností, spolehlivostí, …



- statické
  - výpočet přiřazení procesů na uzly je proveden ještě před spuštěním distribuované aplikace
    - výpočet může běžet libovolně dlouho, abychom dosáhli požadované přesnosti předpovědi – pokud ji metoda umožňuje dosáhnout
  - nelze reagovat na dynamické změny v prostředí
  - vyžadují předem spoustu informací o chování sítě a aplikace (např. kom. zpoždění, doby běhu procesů)
    - nereálné požadavky nelze splnit
      - vliv na přesnost a tedy i rychlost výpočtu

- dynamické
  - výpočet přiřazení procesů na uzly sítě se provádí za běhu distribuované aplikace
    - výpočet se odehrává v reálném čase a nemůže si proto dovolit konzumovat příliš mnoho zdrojů
  - umí se vyrovnat s dynamickými změnami
    - procesy musí umět pre-empci
  - potřebné informace lze zjistit až za běhu aplikace
    - nebo si jich část vyžádat předem

- pre-emptivní
  - procesy lze přerušit během výpočtu a přemístit je na jiný uzel, aby bylo možné kompenzovat změny v síti
    - např. některý z procesů mohl skončit svoji činnost, nebo se odebral do dlouhodobého wait-stavu
  - pokud tuto vlastnost procesy nemají, na změny lze reagovat až při vytváření

- centralizované
  - mají jeden centrální prvek, arbitr, který rozhoduje o rozdělování zátěže na jednotlivé uzly
  - centrální prvek je slabé místo, co se stane, když selže?
  - centralizované správa vyžaduje komunikaci jednoho uzlu se všemi
    - možnost přetížení
  - arbitr má přehled o známé síti a proto lze očekávat, že dokáže zátěž rozdělovat celkem efektivně bez rizik, která jsou jinak spojena s distribuovaným principem

- distribuované
  - rozhodování o rozdělování zátěže provádí několik, až všechny, procesy
    - mohou být distribuovány na několik uzlů
  - když jeden selže, nic se neděje, pokud ho výpočetní model aplikace nutně nepotřebuje k životu
  - procesy, které provádějí rozhodování mohou být buď specialisté, anebo to mají jako „vedlejšák“ ke své hlavní činnosti

- adaptivní
  - síť prochází změnami během výpočtu – mění se stav uzlů
  - adaptivní metody berou do úvahy i několik předchozích stavů při rozhodování o přidělení zátěže

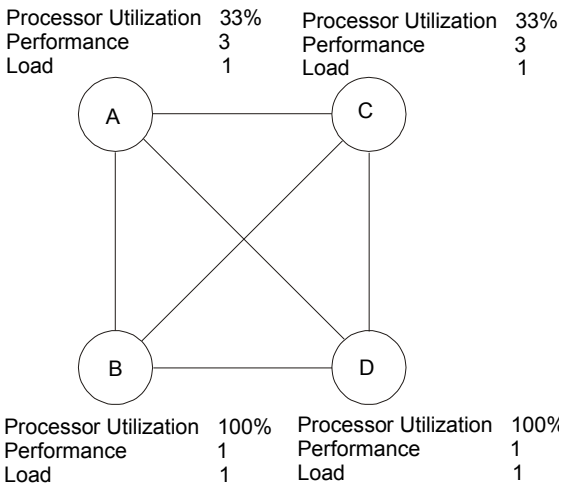
- kooperativní
  - každý proces se může rozhodovat buď sám za sebe, nebo může na rozhodnutí spolupracovat s ostatními
  - přímo – procesy spolupracují nad konkrétním rozhodnutí
  - nepřímo – procesy dávají informace o svých rozhodnutích k dispozici ostatním a ty je použijí při svých rozhodnutích

- sender-initiated
  - v okamžiku, kdy je uzel zatížen přes určitou mez, začne vyhledávat jiné uzly, kam by přemístil část své zátěže
  - je to režie navíc, protože čas potřebný na vyhledávání nových uzlů mohl být použit na běh procesů

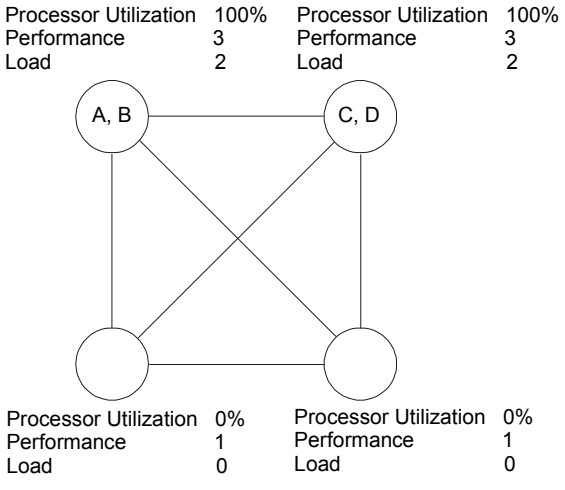
- receiver-initiated
  - v okamžiku, kdy zátěž uzlu klesne pod určitou mez, začne vyhledávat jiné uzly, odkud by mohl převzít jejich zátěž
  - režie se projeví zvýšenou komunikací, uzel má dost volného výpočetního času, který může alokovat pro vyhledávání zátěže

Load Redistribution

- load-balancing tradičně měří zátěž v počtu procesů, což není zrovna to nejlepší
- následující text bude o Load-Redistribution Method in Distributed Environment
  - autor je stejný, jako u těchto přednášek ;-)
- metoda vyžaduje pokročilou síťovou architekturu jako jsou Aktivní sítě



Traditional Load Balancing



Optimal Assignment of Processes

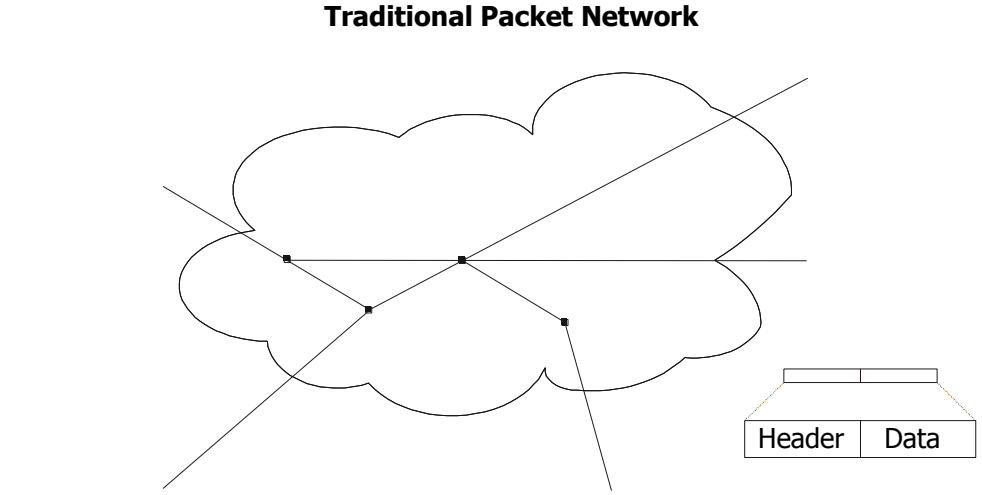
Aktivní sítě

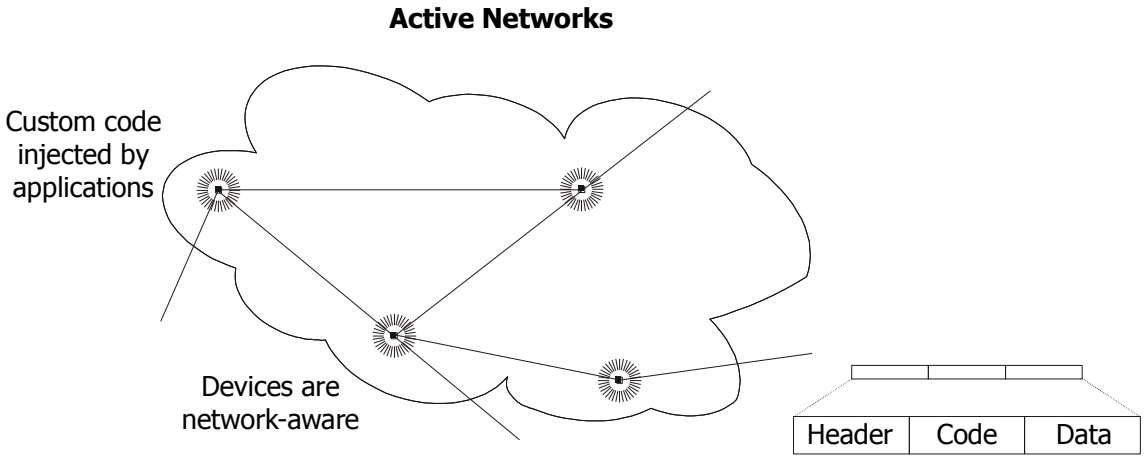
- Odpověď Pentagonu na nedostatky IP protokolu a všech ostatních protokolů, které jsou jimi inherentně postižené
  - Největší problém je standardizace, viz čas, který si vyžádal IPv6 a to ještě nelze mluvit o principiálním řešení stávajících problémů
    - Např. problém s veřejnými adresami byl vyřešen pouze zvětšením adresového prostoru – tj. oddálen
    - AnyCast je možný až s IPv6, PAMCast nenajdeme u IP, ale u aktivních sítí

- The Active Networks program's goal was to produce a new networking platform, flexible and extensible at runtime to accommodate the rapid evolution and deployment of networking technologies and also to provide the increasingly sophisticated services demanded by defense applications.

- IP
  - Dvojice vysílající a příjemce
  - Vysílající odešle paket na konkrétní adresu, včetně portu, kde předpokládá příjemce
  - Pokud tam není, paket se zahodí a vysílající zjistí chybu až timeoutem

- Aktivní síť
  - Paket, nazývaný capsule, je asociován s kódem, který se spustí na každém uzlu, kterým capsule prochází
    - Vždy je přítomný příjemce
  - Kód může manipulovat s daty, vlastnostmi (cíl, TTL, atd.) a vykonávat další uživatelsky definované činnosti
  - Proces se označuje termínem aktivní aplikace
    - Distribuovaná aktivní aplikace se skládá z několika aktivních aplikací, které mohou injektovat capsule a zároveň capsule může injektovat aktivní aplikace

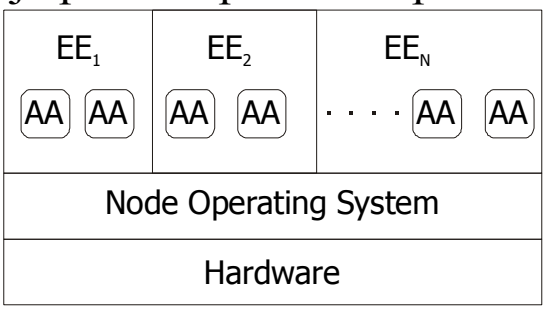




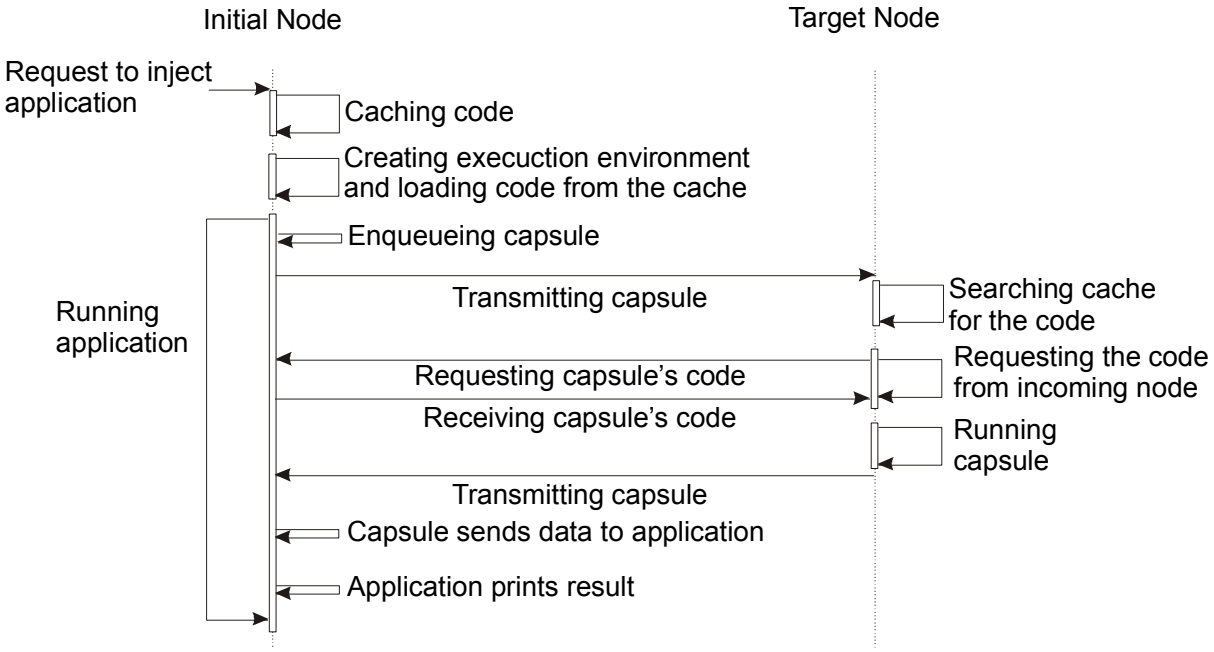
- Komunikační model sám-sobě
  - Je možné injektovat capsuli do sítě, aby nasbírala potřebná data a pak je předala procesu, který ji injektoval
    - U IP by bylo nutné mít dopředu na každém uzlu, který by capsule mohla navštívit, spuštěný specializovaný proces

- Migrace procesů
  - Migrující proces změní svoji síťovou adresu, ale ještě ji nedal na vědomí ostatním procesům
  - Informaci o své nové síťové adrese zanechal na uzlu, odkud migroval
  - Capsule, která má doručit data, dorazí na uzel, odkud proces odmigroval, tam ho nenajde, ale použije svůj kód, aby si přečetla novou adresu a pouze změní svůj cíl
  - Ostatní procesy si mohou aktualizovat záznamy až později – lazy update, u IP je nutné vyřešit předem

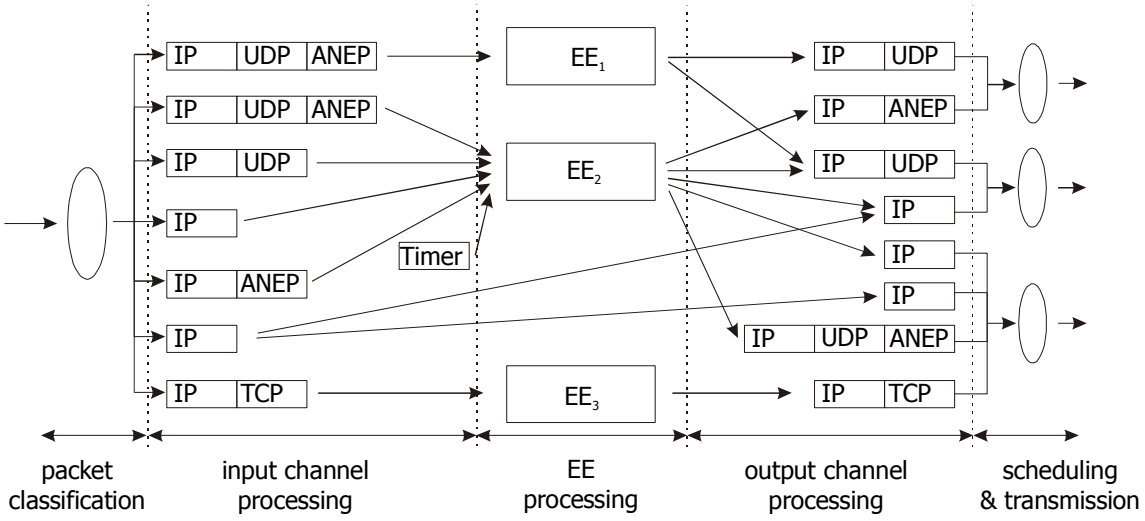
- V aktivní síti je zapotřebí standardizace pouze dvou věcí
  - Programového kódu
    - Kód vykonává Execution Environment (EE)
      - Na jednom uzlu může být několik EE
  - Code distribution protocol
  - Vše ostatní je pak už aplikačně specifické



AA = Active Application
EE = Execution Environment



Co všechno se může stát, když uděláme ping



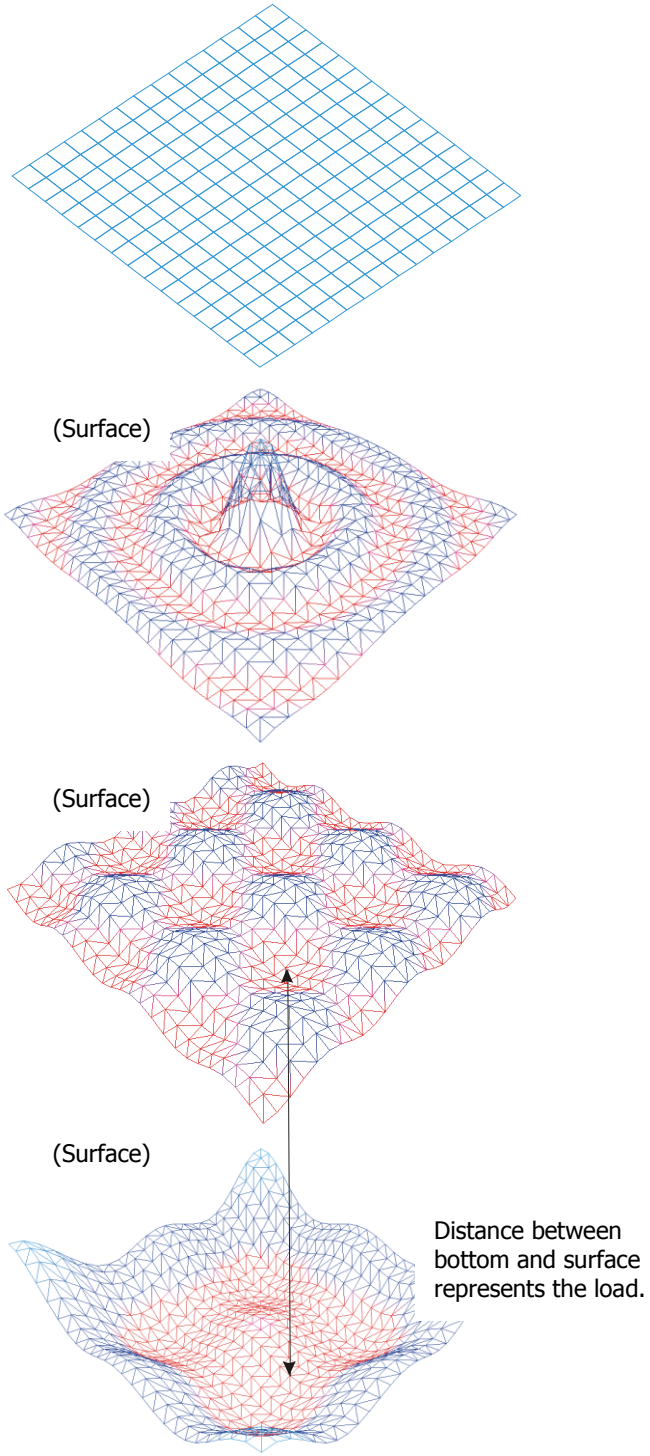
Tak jde život v aktivním uzlu

Load-Redistribution Method in Distributed Environment

- aneb o čem je práce doktoranda
  - aneb s čím se dá propracovat do IEEE knihovny
  - prezentované výsledky a informace, a nejen ty, byly nasbírány právě během doktorského studia

- předpokládá se prostředí aktivních sítí, nebo sítí poskytující stejné možnosti
- metoda je naprosto nezávislá na konkrétní aplikaci a síťové topologie, výkonnosti uzlů, atd.
  - vše si dokáže zjistit sama za běhu
  - programátor ji jenom použije jako knihovnu a napíšu podporu pro migraci procesů
    - ukládání a rekonstrukce stavu, protože v uzly nemusejí být identické a tím pádem ani nelze obnovovat paměťový obraz
    - jak obnovit platné handles systémových objektů na jiném uzlu?

- Princip
  - Procesy se rozhodují samy za sebe,
  - Periodicky zkoumají své okolí
  - Nepřímo spolupracují
  - => chovají se jako kolonie organismů, které mají jednoduchá pravidla, ale dohromady vytvářejí inteligentní chování a dokáží se adaptovat na změny
  - Metoda se nesnaží dosáhnout dobře vyváženého stavu hned na první pokus, ale postupně



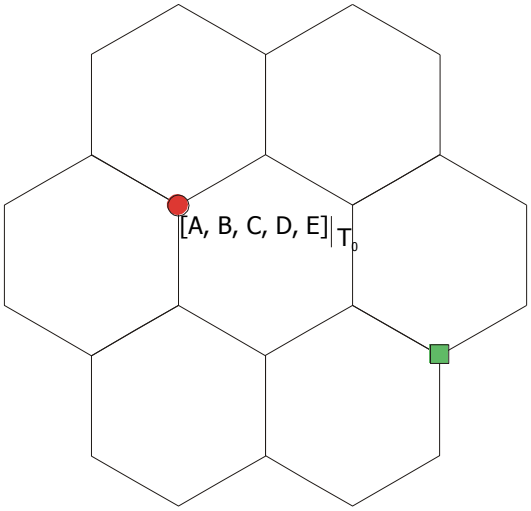
All running applications are ideally balanced from the point of view of computation power of available nodes. If communication is well balanced then this is well-balanced system.

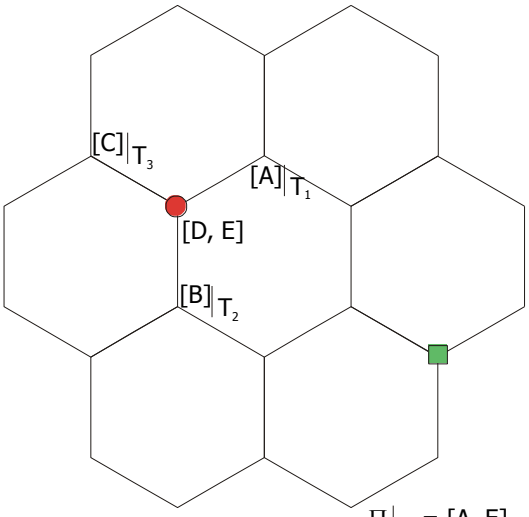
Several processes composing a single application were ran in the initial node causing its high local overload. This also had an impact on load of adjoining nodes as processes started to migrate from the initial node.

Application is running for some time causing waves as nodes are unequally loaded and the communication relationships are established.

Varying depth represents power of each node.

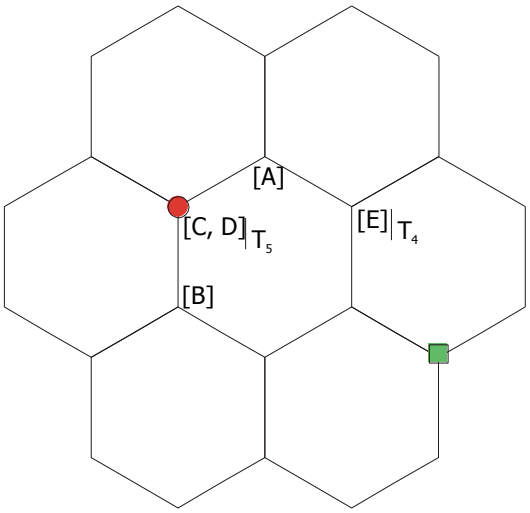
Model





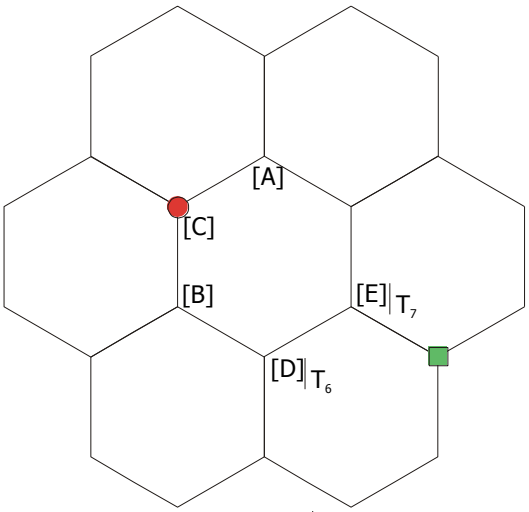
$\Pi|_{T_3}$ = [A, E]

E → A



$\Pi|_{T_{4,5}}$ = [B, C]

C → B



$\Pi|_{T_6}$ = [B, C, D, E]

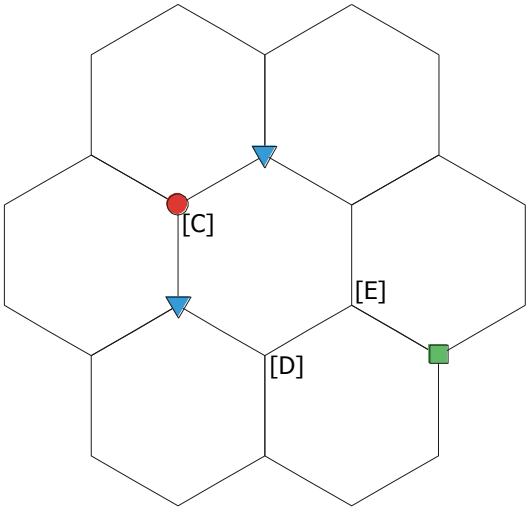
B → C

E → D

$\Pi|_{T_7}$ = [A, B, C, D, E]

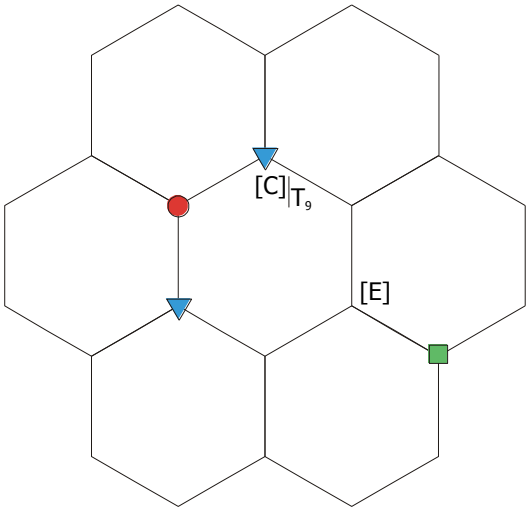
C → A

B → E → D



$\Pi|_{T_8}$ = [C, D, E]

D → E ← C



$\Pi|_{T_9}$ = [C, E]

C → E

- Síťové prostředí
  - Procesy injektují speciální capsules – PerformanceScouts
  - Taková capsule nejen, že zjistí, jak vypadá síťové okolí uzlu z hlediska topologie, ale také výkonnost, zatížení, komunikační zpoždění a další informace o navštívených uzlech
  - Takto zjištěné informace se pak využijí při hledání uzlu, kam by mohlo být vhodnější odmigrovat proces, protože by tam mohl běžet rychleji
  - Network topology discovery

- Komunikace
  - Pokud je distribuovaná aplikace dobře napsaná, pak procesy tvoří skupiny, které mezi sebou komunikují
    - Communication cluster
  - Může být tolik clusterů, kolik je procesů
  - Clustery jsou definovány virtuální topologií – komunikační schéma distribuované aplikace
  - Vyplatí se držet procesy z daného clusteru blízko sebe, aby se snížilo komunikační zpoždění
  - Čas od času může proces komunikovat i s jiným
  - procesem, který do jeho clusteru nepatří
    - ideální cluster

  - Máme tři procesy s komunikací A – B, B – C
    - Existují tři clustery
      - A, B
      - B, C
      - A, B, C

- Poskytnutím komunikačních funkcí lze trasovat komunikaci a odhalit tak komunikační clustery
  - Potencionálně se nemusí odhalit celá virtuální topologie, protože tok zpráv může být závislý na vstupních parametrech
  - Z hlediska efektivního snižování komunikačních zpoždění nemá smysl brát do úvahy komunikaci, za celou dobu běhu, ale pouze za poslední dobu
  - Virtual Topology Discovery

- Počítáme-li s migrací procesů, je výhodné začít používat adresování nezávislé na adresách uzlů a zjednodušit tak život programátorovi
  - Zajištění integrity, konzistentnosti a adresovatelnosti

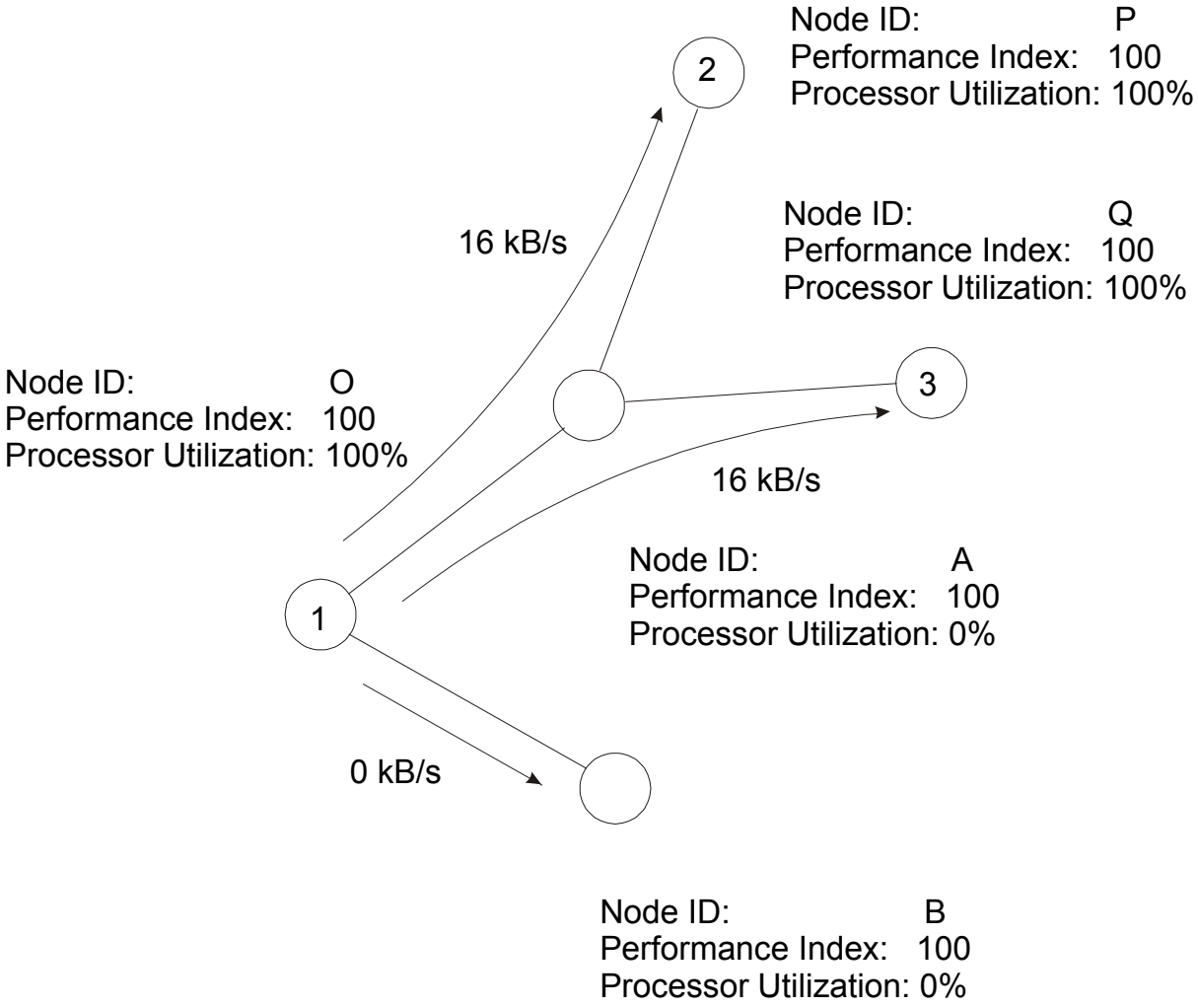
Process ID	0	1	2	3	4	5	6	7	8	9	10	11
0		**408**		**425**		**441**		**1206**		**838**		**391**
1			**1206**	**425**	376		**435**	**836**	**719**	**832**	367	366
2	**825**				**447**	387	**1228**	**855**	**444**		**768**	
3	**442**	**1217**	**843**		386	**796**	365	**856**	384			**756**
4	**451**	**452**	**375**	**407**			**886**			369	**445**	
5		**417**	**452**	**402**	**869**		**773**	**379**			367	
6	**822**	**434**		**822**	**415**	**2051**			**453**		**430**	
7	**431**	**414**	**1135**	406		**430**	**902**				**799**	
8			360		**415**	**1209**	**447**	**783**		**421**		**367**
9			**427**	**874**	**390**	372			**1172**		**822**	**384**
10	**1150**	**390**	**1228**			362		**407**		**406**		**1262**
11		356	**1698**	**431**	**1331**	368	**408**		403	**791**	**1681**	

Reprezentace komunikačních clusterů

- Výkon a rychlost běhu procesu
  - Lze změřit, kolik procesorového času proces konzumuje na lokálním uzlu a následně porovnat, zda by mu jiný uzel mohl poskytnout více procesorového času
  - Pomocí benchmarku se dá porovnávat výkon na různých typech procesorů
  - Lze identifikovat několik uzlů, kde by proces mohl běžet rychleji
    - Vybere se ten, kde dojde k největší minimalizaci komunikačního zpoždění
      - Pokud bychom dali požadavek na dostupný procesorový čas a požadavek na minimalizaci komunikačních zpoždění na stejnou úroveň, v praxi by se pak většinou jednalo o protichůdné požadavky

$$\forall N_i \in N : \frac{load(N_k) \times delay(N_k)}{amount(N_k)+1} = \min_i \left(\frac{load(N_i) \times delay(N_i)}{amount(N_i)+1} \right), \text{ where } 1 \le i \le |N|$$

Dispersion Formula



Výběr nového uzlu

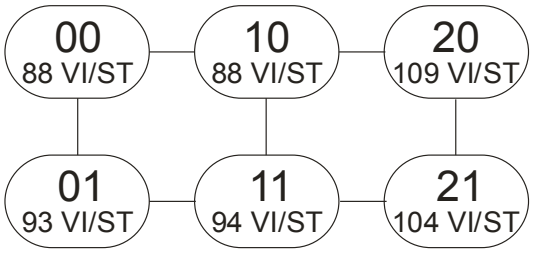
- Synchronizace
  - Každý proces se rozhoduje sám za sebe, ale nepřímo spolupracuje s ostatními pomocí blackboards, které jsou podporovány přímo aktivním uzlem
    - Grade32
      - referenční implementace
      - www.kiv.zcu.cz/~txkoutny/download/grade32.zip
    - SAN - Smart Active Node
      - Nástupce Grade32
      - sourceforge.net/projects/smartactivenode/

- Rizika
  - Masová migrace
    - Několik procesů, i z několika různých uzlů, si může jako nový cíl vybrat stejný uzel, který se stane přetíženým
  - Oscilace
    - Proces se může pohybovat po síti, aniž by cokoliv spočítal
    - Zavedení kreditů
      - Dokud toho dost nenapočítá, nikam se migrovat nebude
    - Extrémní forma je migrace mezi dvě uzly
  - Zbytečné migrace procesů
    - Viz oscilace
    - Pokud mají dva uzly přibližně stejné množství procesorového času, který mohou nabídnout procesu, může dojít k migraci, aniž by došlo k urychlení výpočtu
    - Procesorový čas poskytovaný lokálním uzlem se dynamicky „papírově“ navýší
      - Pokrývá režii spojenou s migrací
      - Zaručuje, že nebude docházet k migraci, pokud je systém v dobře vyváženém stavu

```
var a[1:n]:int
    sum[1:n]:int
    old[1:n]:int
    d:int=1
//initialize elements of sum
sum[i:1..n]::
  sum[i]:=a[i]
barrier
{SUM: sum[i]:=a[i-d+1]+…+a[i]}
while d<n do
  //save old value
  old[i]:=sum[i]
  barrier
  if i-d>=1 then
     sum[i]:=old[i-d]+sum[i]
  barrier
  d:=d*2
end while
```

Application Specific Parameters

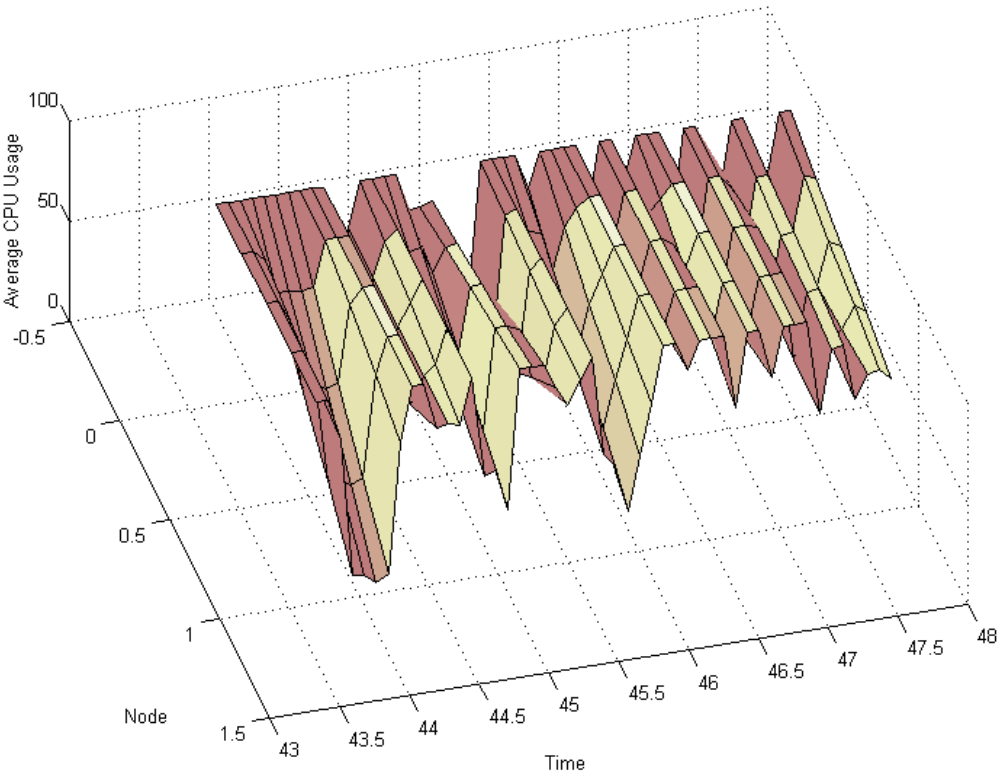
Working Time	Poisson, 62700 VI
Waiting Time	Poisson, 540 VI
Working Probability	85%
Waiting Probability	7%
Sending Probability	8%
Processes	12
Migration Interval	627 VI

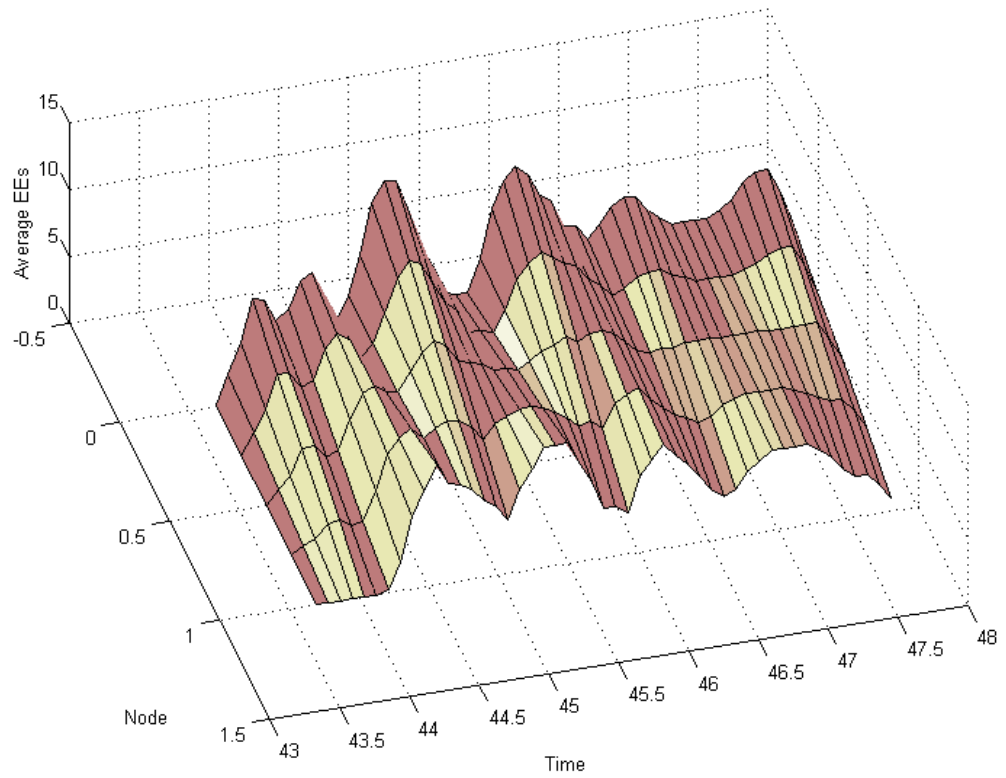


Testovací síť
100 VI/ST ~ Intel Core Duo 1.86GHz

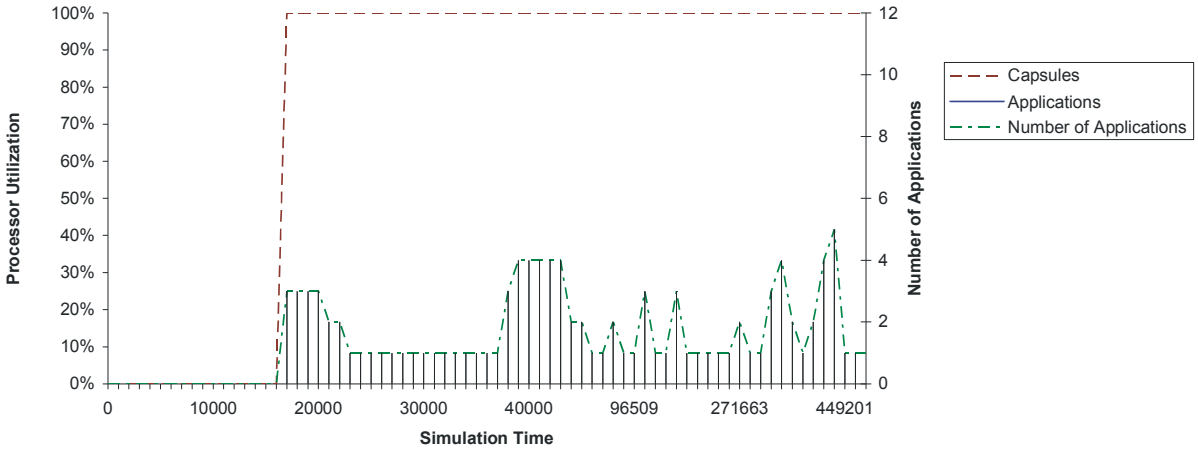
Capsule Specific Parameters

Working Time	Poisson, 10 VI
Size	Uniform, 304B – 504B
Link Speed	100B/ST

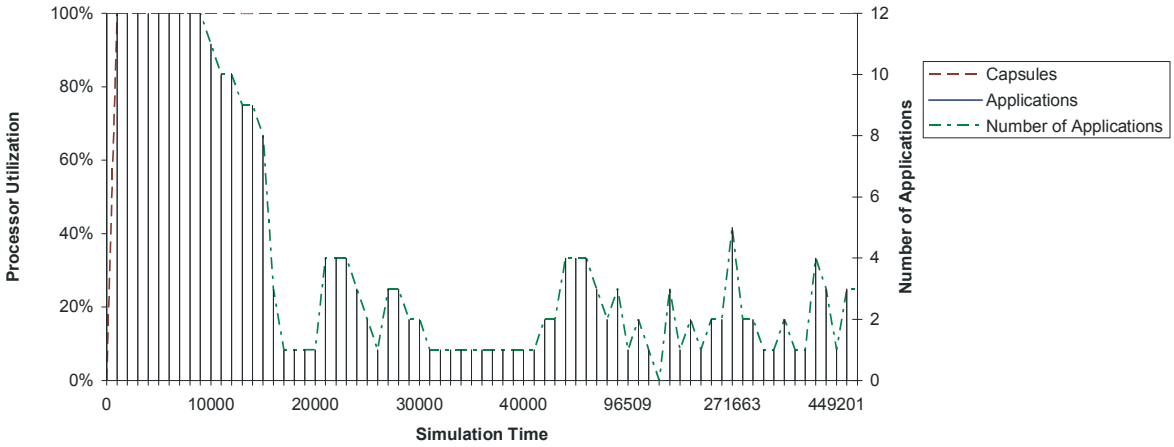




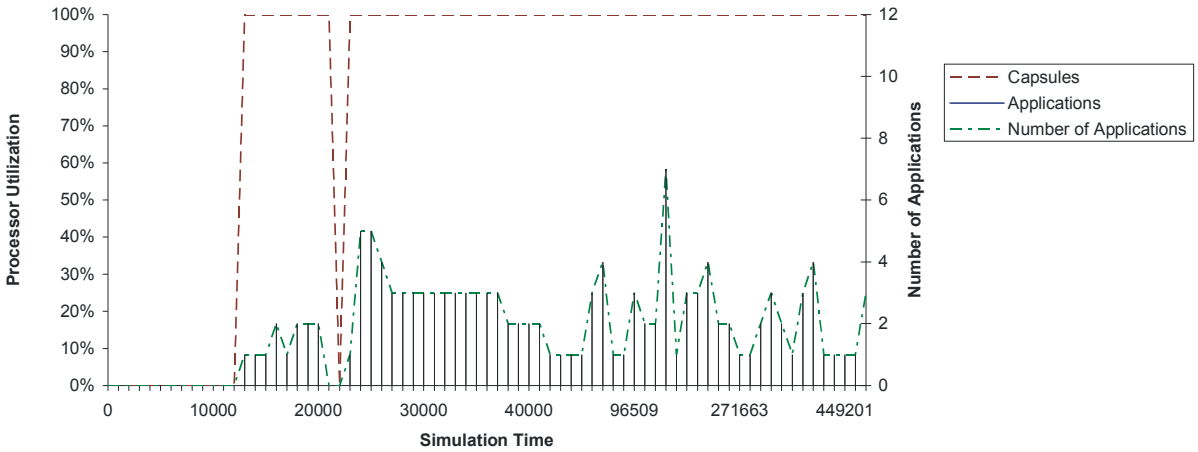
Runtime Results



Node 01



Node 11



Node 21

Simulation Results

TimeWarp simulace

- Distribuovaná simulace
- Optimistická simulace, která pokračuje kupředu a předpokládá, že vše proběhlo v pořádku

- pokud přece jenom něco neproběhlo v pořádku, pak se zašle antimessage
  - procesy spolu komunikují pomocí zpráv
  - pokud příjemce ve své frontě příchozích zpráv zjistí i anti-zprávu, příslušnou zprávu jednoduše smaže
  - pokud anti-zpráv přišla pozdě, provede se rollback zpět do doby, než se příslušná zpráv zpracovala

- existují troje hodiny
  - reálné
  - wallclock – jak dlouho už simulace běží
  - simulační
  - je potřeba znát minimální čas, kdy jsou všechny zprávy potvrzeny a už pro ně nepřijdou anti-zprávy
    - není tak jednoduché určit, protože některé zprávy mohou být na cestě

- A kdeže se můžete setkat s touto zázračnou simulací?

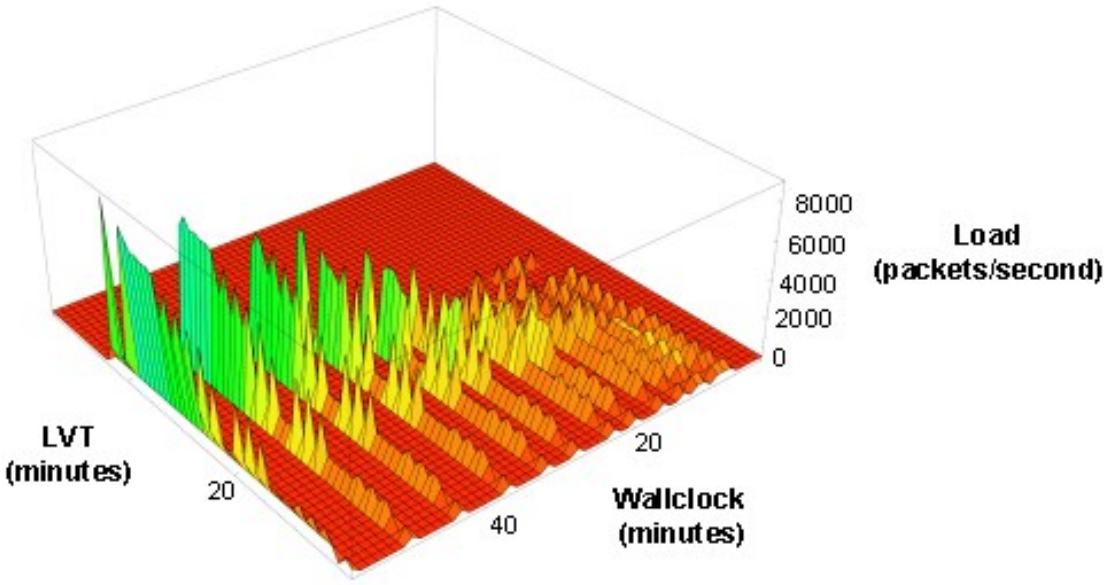
AVNMP

- Active Virtual Network Management Prediction
- http://www.crd.ge.com/~bushsf/AVNMP.html

- Vytvoří se simulace sítě pomocí algoritmu z rodiny TimeWarp a tou se „překryje“ skutečná síť
- Díky časové kompresi simulace, simulační čas běží rychleji než reálný, se za nějakou dobu dozvíme, co se stane se sítí – predikce budoucnosti s nějakou pravděpodobností

- Streptichron
  - Capsule, která nese informaci, zprávu, o stavu sítě
  - Při průchodu jednotlivými uzly může předpověď aktualizovat podle jejich stavu
    - Data processing while in transit

- Využití
  - Při přerozdělování zátěže lze tak dopředu predikovat stav, kterému by bylo záhodno se preventivně vyhnout i za cenu planého poplachu
    - Nebo jsou ztráty v celkovém součtu přijatelné
  - Algoritmu lze zadat vlastní parametry, které mohou reprezentovat možné výchozí situace a spustit několik simulací najednou
  - Prevence proti útokům jako je DDoS
    - AN poskytují možnosti pro-aktivních opatření
      - Detekce
      - Spolupráce uzlů
      - U IP se „jenom díváte“



http://www.crd.ge.com/~bushsf/an/terena01.pdf

- LVT – Local Virtual Time, běží napřed
- Wallclock – čas simulace
- Load – počet kapsulí v síti

Příklady využití zmíněných metod

- load-sharing, load-balancing, load-redistribution
  - urychlení rozsáhlých výpočtů
    - předpověď počasí
    - matematicko-fyzikální modely
      - Alcubierre Drive?
        - www.nasa.gov/centers/glenn/research/warp/ideachev.html
      - proč je tvar F-117A Nighthawk hranatý a B-2 Spirit zaoblený?
        - Náročnost a technika výpočtu
    - rendering – Shrek IV?
    - SETI@Home
    - Distribuované simulace
  - Fault-tolerant systémy

- Load-redistribution/AN
  - Možná obrana proti jakékoliv formě DoS útoku
    - Server se „prostě sebere“ a odstěhuje z napadeného uzlu
  - Data processing while in transit
    - Video broadcasting
  - Síť se může měnit za chodu a změny se nemusí „hlásit“ distribuované aplikaci
    - A network that can turn on a dime.
  - NASA Asteroid Exploration with Autonomic Systems – množství malých, spolupracujících robotů
- Aktivní sítě
  - Load-redistribution
  - AVNMP
  - SAN
    - umožňuje dynamické vytváření síťových adres, které jsou stále adresovatelné
    - adresa pak nemusí určovat místo, kde se fyzicky nalézá počítač tak, jak to známe z IP
      - anonymita
        - za jak dlouho se to objeví ve výměnných sítích?
  - Pro-aktivní přístup k řešení nejrůznějších problémů
    - Civilní i vojenské řešení
      - Např. vytvoření mobilní, plně zabezpečené, sítě na bojišti
        - Podpora ze strany Pentagonu
      - ESA vypsala grant na studii možnosti vysílání digitální televizi právě pomocí aktivních sítí

K opravdu zajímavým věcem se bez Ph.D. nedostanete!