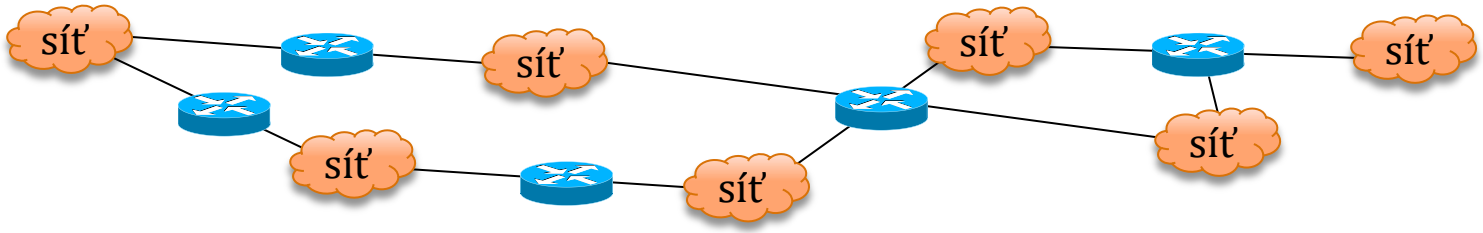
Lekce 8: Síťová vrstva a směrování

Jiří Peterka

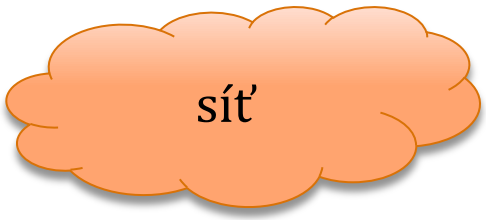
možné přístupy k propojování sítí

- **(obvyklý) základní koncept propojování sítí:**
  - „svět“ je tvořen jednotlivými sítěmi, které jsou vzájemně propojeny na úrovni síťové vrstvy
    - prostřednictvím směrovačů

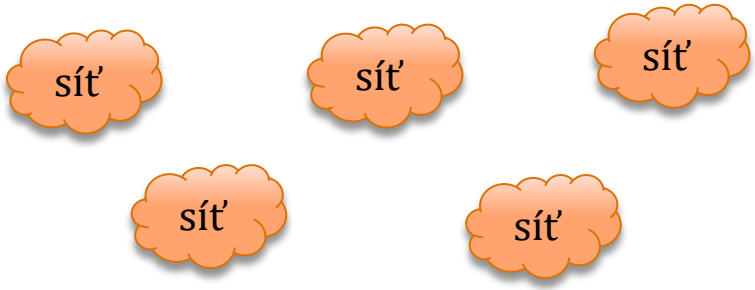
označováno jako: princip katenetu



    - důsledek: vždy existuje (alespoň jedna) cesta mezi libovolnými dvěma sítěmi
      - mezi libovolnými uzly
        - nalezení této cesty je úkolem směrování
- **možné alternativy:**
  - existuje jen 1 síť
    - celý „svět“ tvoří jediná síť
      - pak ani není zapotřebí síťová vrstva
    - příklad: NetBIOS, protokol LAT

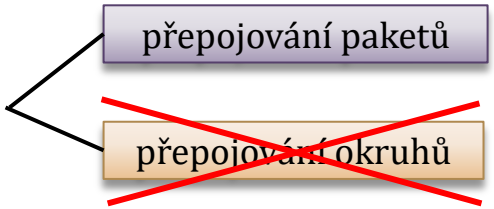


  - existuje více sítí, ale nejsou nijak propojeny
    - a neexistují mezi nimi žádné cesty

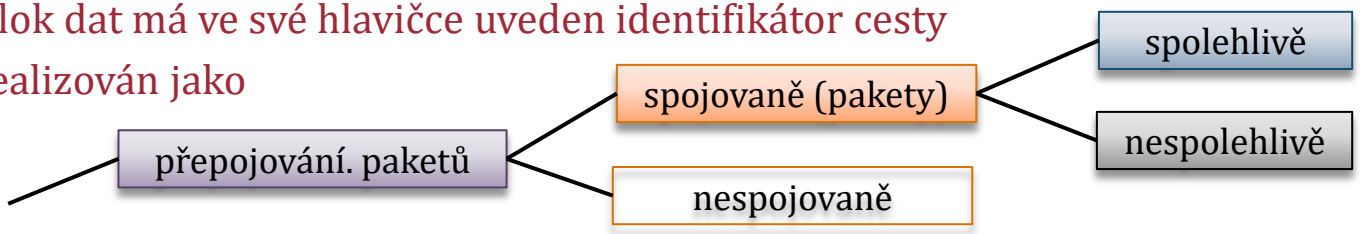


možné způsoby fungování síťové vrstvy

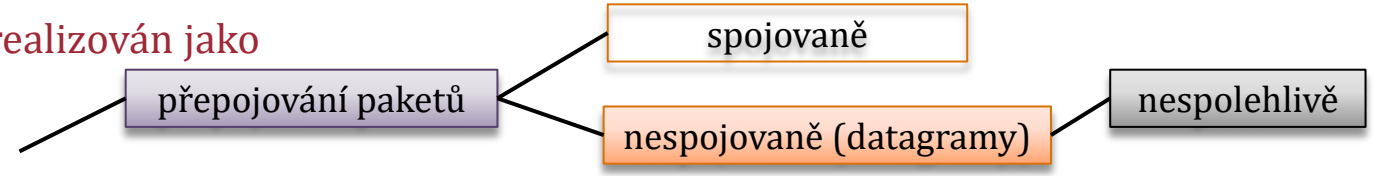
- **připomenutí:**
  - na úrovni síťové vrstvy může „fungovat“ (přenášet data) více přenosových protokolů
    - v TCP/IP: pouze protokol IP (Internet Protocol)
  - (zde) předpokládáme, že celá síť funguje na principu přepojování paketů
    - nikoli na principu přepojování okruhů
      - pak by síť ani neměla klasickou síťovou vrstvu a směrování



- **přenosový protokol síťové vrstvy může fungovat:**
  1. spojovaným způsobem: přenášený blok se obecně označuje jako **paket**
     - na začátku přenosu je (jednorázově) vyhledána cesta od odesilatele k příjemci
       - této cestě je přidělen vhodný identifikátor
     - každý přenášený blok dat má ve své hlavičce uveden identifikátor cesty
     - přenos může být realizován jako
       - spolehlivý
       - nespolehlivý

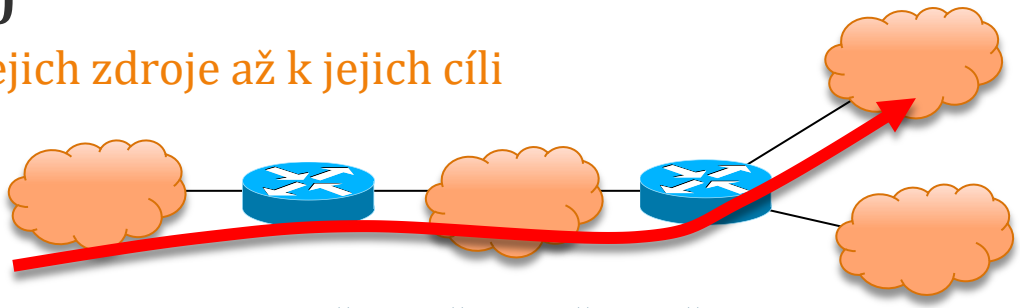


  2. nespojovaným způsobem: přenášený blok se obecně označuje jako **datagram**
     - každý přenášený blok má ve své hlavičce celou adresu svého příjemce
     - k hledání cesty dochází pro každý blok znovu (v každém směrovači „po cestě“)
     - přenos je obvykle realizován jako
       - nespolehlivý



co je úkolem síťové vrstvy (L3)?

- **hlavní úkol: směrování (routing)**
  - dopravovat bloky dat (pakety) od jejich zdroje až k jejich cíli
    - i přes mezilehlé uzly / celé sítě
  - zahrnuje:
    - **volbu směru (routing)**
      - směrování v užším slova smyslu: rozhodování o cestě / směru dalšího přenosu
    - **cílené předávání (forwarding)**
      - samotná manipulace s jednotlivými pakety („předání dál“ ve zvoleném směru)
    - obě tyto funkce jsou obvykle realizovány společně, v zařízení zvaném **směrovač (router)**
      - ale mohou být také oddělené
        - směrování (rozhodování) může být řešeno centrálně, distribuovaný je pak pouze forwarding



- **úkolem síťové vrstvy může být také (je-li to požadováno):**
  - zajištění podpory kvality služeb (QoS, Quality of Service)
    - obvykle: ve spolupráci s dalšími vrstvami, např. transportní
  - předcházení zahlcení (congestion control), viz lekce č. 6
    - eliminace stavů, kdy je přenosová síť zahlcena a nestíhá přenášet všechny požadované pakety
  - řízení toku (flow control), viz lekce č. 6
    - předcházení tomu, aby odesilatel zahltil příjemce

předpoklad: jedná se o síť fungující na principu **přepojování paketů** (nikoli na principu přepojování okruhů)

v sítích s přepojováním okruhů je „všechno jinak“

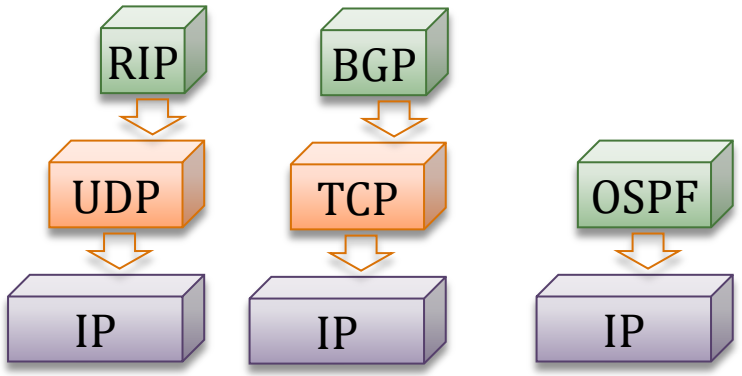
co je směrování (routing)?

- **v širším slova zahrnuje také:**
  - způsob fungování síťové vrstvy
    - přepojování okruhů/paketů, pakety vs. datagramy, ...
  - celkovou koncepci propojování sítí
    - jak a čím se sítě propojují
  - celkovou koncepci směrování
    - které uzly se účastní směrování
      - a do jaké míry
  - řešení směrování v opravdu velkých systémech
    - hierarchické směrování
    - autonomní systémy
  - koncepci síťových adres
  - metody optimalizace směrovacích tabulek
  - směrovací politiky
  - směrovací protokoly
    - RIP, OSPF, BGP, EIGRP, ...
- **v užším slova smyslu:**
  - volba směru pro další předání paketu/datagramu do jiné sítě
    - algoritmy směrování
- **ve skutečnosti zahrnuje:**
  - výpočet optimální cesty
    - je to kombinatorický problém hledání nejkratší cesty v grafu
    - výsledkem jsou "podklady pro volbu směru"
  - uchovávání směrovacích informací (podkladů pro rozhodování)
    - vedení směrovacích tabulek
  - předávání paketů (forwarding)
    - využívání výsledků výpočtů ("podkladů")
  - udržování směrovacích informací
    - aktualizace údajů pro výpočty cest, reakce na změny
  - fungování směrovače (jako zařízení)

jak (a kde) funguje směrování?

- **představa:**
  - ke směrování dochází na síťové vrstvě (L3)
    - směrovače fungují na síťové vrstvě (L3), nemají vyšší vrstvy
- **přitom:**
  - rozhodování o volbě dalšího směru vychází z informací dostupných na síťové vrstvě (L3)
    - v TCP/IP: z IP adres
  - samotná manipulace s pakety či datagramy probíhá na síťové vrstvě
- **ale:**
  - většina souvisejících činností se odehrává na vyšších vrstvách, zejména:
    - hledání cest
    - výměna a aktualizace směrovacích informací
      - prostřednictvím protokolů jako je RIP či OSPF, nebo BGP

- **příklad (TCP/IP):**
  - protokoly RIP a BGP jsou aplikačními protokoly
    - fungují na aplikační vrstvě (L7)
      - RIP využívá služby transportního protokolu UDP (port 520)
      - BGP využívá služby transportního protokolu TCP (port 179)



  - OSPF vkládá svá data přímo do IP datagramů (IP Protocol Type 0x59 = OSPF)
    - patřil by proto na transportní vrstvu
      - ale je také spíše aplikační - má vlastní (zabudovanou) transportní vrstvu

směrovací tabulky

- **jsou datovou strukturou, ve které jsou uchovávány (některé) podklady pro směrování**
  - pro „logické činnosti“: hledání nejkratších cest a výměnu směrovacích informací
  - pracují s nimi („aktualizují je“) protokoly jako RIP a OSPF
- **položky obsahují:**
  - cílovou síť s maskou
    - nebo s prefixem
  - „next hop IP“
    - adresu směrovače
  - odchozí rozhraní
  - ohodnocení
    - v metrice

směrovací tabulka uzlu A

Správce: Příkazový řádek

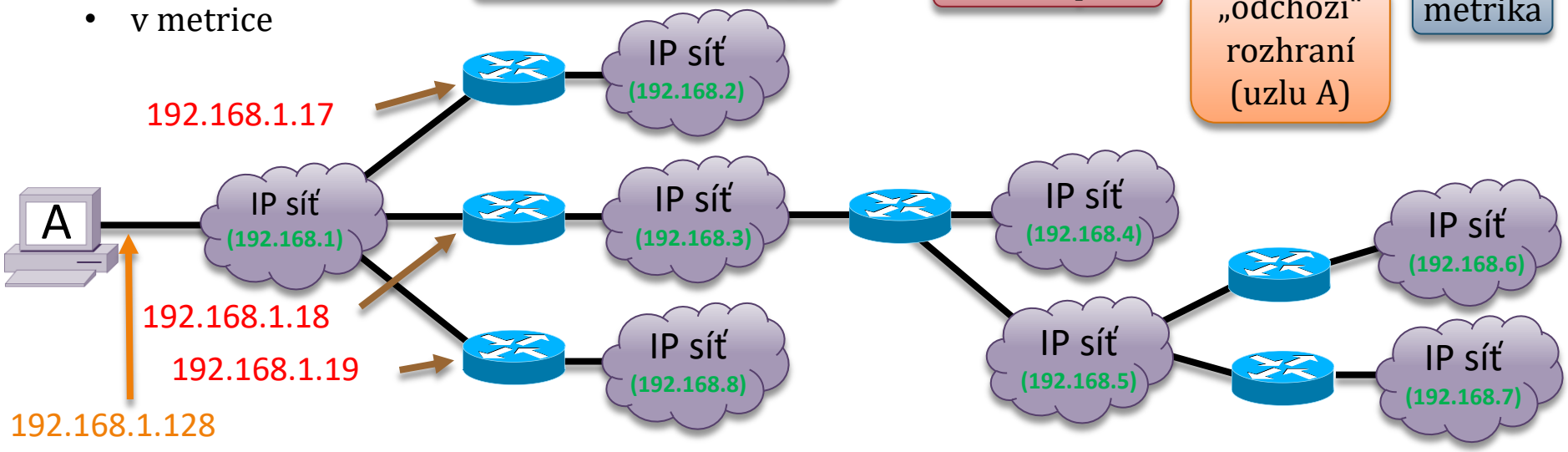
```
IPv4 Směrovací tabulka
===========================================================================
Aktivní směrování:
          Cíl v síti   Síťová maska          Brána       Rozhraní Metrika
      192.168.2.0    255.255.255.0    192.168.1.17    192.168.1.128     11
      192.168.3.0    255.255.255.0    192.168.1.18    192.168.1.128     11
      192.168.4.0    255.255.255.0    192.168.1.18    192.168.1.128     20
      192.168.5.0    255.255.255.0    192.168.1.18    192.168.1.128     20
      192.168.6.0    255.255.255.0    192.168.1.18    192.168.1.128     30
      192.168.7.0    255.255.255.0    192.168.1.18    192.168.1.128     30
      192.168.8.0    255.255.255.0    192.168.1.19    192.168.1.128     11
===========================================================================
```

cílová síť (+maska)

next hop IP

„odchozí“ rozhraní (uzlu A)

metrika



forwardovací tabulky

- **směrovací tabulky jsou „informačně bohaté“**
  - hodí se pro rozhodování, ale nejsou vhodné pro samotnou manipulaci s IP datagramy
    - která musí být co možná nejrychlejší – a u které se již „nepřemýšlí“

cílová síť	next hop IP
192.168.2/24	192.168.1.17
192.168.3/24	192.168.1.18
192.168.4/24	192.168.1.18
192.168.5/24	192.168.1.18
192.168.6/24	192.168.1.18
192.168.7/24	192.168.1.18
192.168.8/24	192.168.1.19

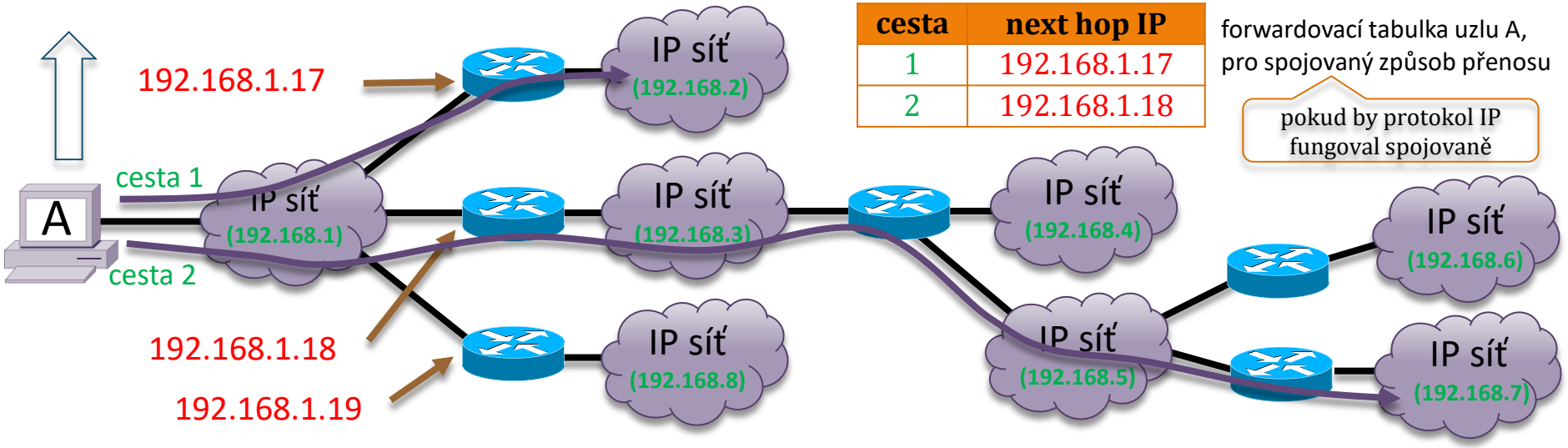
forwardovací tabulka uzlu A, pro nespojovaný způsob přenosu

  - pro samotnou manipulaci s datagramy se používají menší a rychlejší tabulky
    - tzv. forwardovací tabulky (forwarding tables)
  - představa:
    - forwardovací tabulka je „výcucem“ ze směrovací tabulky
      - obsahuje pouze ty cesty, které již byly vybrány jako optimální
        - nepotřebuje např. ohodnocení (v metrice)

cesta	next hop IP
1	192.168.1.17
2	192.168.1.18

forwardovací tabulka uzlu A, pro spojovaný způsob přenosu

pokud by protokol IP fungoval spojovaně



možné přístupy ke směrování

- **podle toho, zda směrování reaguje na dění v síti (změny topologie apod.):**

- **adaptivní (dynamické) směrování**
  - snaží se reagovat na změny
    - vyžaduje aktualizaci informací o stavu celé soustavy sítí
    - vyžaduje průběžné hledání nejkratších cest
  - potřebuje protokoly jako RIP, OSPF, BGP, ...
    - které dynamicky aktualizují obsah směrovacích tabulek
      - základ jejich obsahu může být dán staticky/dopředu
  - nevýhoda:
    - vysoká režie zejména na aktualizaci informací
      - s velikostí soustavy sítí rychle roste
        - otázka škálování

v praxi častější

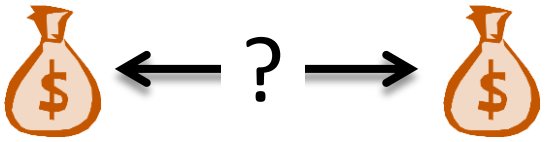
- **neadaptivní (statické) směrování**
  - nesnaží se reagovat na změny v soustavě vzájemně propojených sítí
    - nepotřebuje aktualizace ani spolupráci s ostatními uzly
      - tedy ani protokoly jako RIP, OSPF, BGP
  - obsah směrovacích tabulek je dopředu a pevně dán
    - je statický, nemění se v čase
  - výhody:
    - vyhovuje to zvýšeným požadavkům na bezpečnost
      - nelze napadnout skrze šíření aktualizací
    - není režie na aktualizaci
    - lze vyhovět i speciálním požadavkům na směrování
  - nevýhoda:
    - nereaguje na změny, pokud k nim dojde

bezpečnější

obvyklé přístupy ke směrování

- **destination-based routing**
  - směruje se (jen) na základě cílové adresy
    - zdrojová adresa při směrování nehraje roli
  - směruje se na základě příslušnosti k síti
    - IP sítě: podle síťové části cílové IP adresy
      - a pouze v cílové síti se bere v úvahu také relativní část IP adresy

- **least-cost routing**
  - optimální cesta se volí podle nejnižší „ceny“
    - ve smyslu používané metriky
    - v praxi: metrikou je nejčastěji počet přeskoků (směrovačů) po celé cestě
  - není podporováno více cest se stejnou cenou
    - možnost jejich současného využití



- **hop-by-hop routing**
  - směruje se „per hop“: v každém směrovači se rozhoduje o optimální cestě
    - spojovaná varianta (pakety):
      - rozhoduje se jen 1x, při navazování spojení
    - nespojovaná varianta (datagramy)
      - rozhoduje se pro každý datagram znovu (samostatně)
- **směrování je nezávislé na obsahu a zdroji**
  - algoritmy směrování se neptají na to, co jednotlivé pakety obsahují a odkud pocházejí
    - souvisí s principem best effort a absencí QoS
- **směrování je bezestavové**
  - rozhodování je nezávislé na historii a předchozích datagramech

možné alternativy směrování

- **content switching**
  - snaha rozhodovat se při směrování také podle obsahu/charakteru dat
    - alespoň podle čísel portů
    - alternativa k principu, že při směrování nezáleží na přenášených datech
- **source-based routing**
  - algoritmy směrování se rozhodují i podle toho, odkud data pochází
    - alternativa k principu, že na zdroji dat nezáleží
      - destination-based routing
- **policy-based routing**
  - ještě obecnější koncept: směrování bere v úvahu celou řadu faktorů
    - včetně komerčních zájmů
    - obvykle: všechny tyto „zájmy“ jsou vyjádřeny v tzv. směrovací politice
- **koncept toků (flows)**
    - objevuje se v IPv6
  - při nespojovaném přenosu: jednotlivé pakety/datagramy nějak „patří k sobě“ a podle toho jsou směrovány
    - alternativa k principu, že při směrování nezáleží na historii a předchozích paketech
      - že se každý paket směruje nezávisle na ostatních paketech
- **tag switching**
  - obdoba toků (flows)
    - alternativa k principu, podle kterého se při volbě směru vychází ze síťových adres
      - zde se vychází z „nálepek“ (tags)
    - jakoby přechod z L3 na L2
      - od směrování k přepínání

varianty směrování

- **podle toho, zda a jak spolu jednotlivé uzly (směrovače) spolupracují**

používá se spíše výjimečně

- **izolované směrování**
  - každý směrovač jedná jen „sám za sebe“
    - vůbec nespolupracuje s ostatními uzly
    - například:
      - záplavové směrování
      - source routing
      - metoda horké brambory
      - náhodné směrování
      - metoda zpětného učení
      - .....
- **centralizované směrování**
  - volbu směru (směrování v užším slova smyslu) provádí jedna centrální autorita
    - ostatní uzly provádí její rozhodnutí
      - realizují pouze cílené předávání (forwarding)

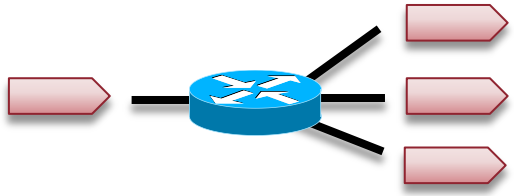
obvyklé (nejčastější) řešení

- **distribuované směrování**
  - směrovače vzájemně spolupracují
    - např. formou distribuovaného výpočtu nejkratších cest
      - každý provádí část výpočtu
        - metody distance-vector
          - sousední směrovače si vzájemně předávají své směrovací tabulky
    - nebo vzájemným sdělováním „podkladů“ pro individuální výpočet
      - každý provádí sám celý výpočet
        - metody link-state
          - každý směrovač monitoruje dostupnost svých sousedů (stav svého spojení k nim: link-state)
          - o případné změně stavu spojení (link-state) informuje všechny ostatní směrovače

může být efektivní a pružné, ale má „single point of failure“

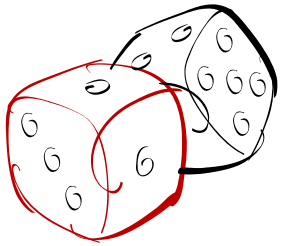
příklad: izolované směrování

- **záplavové směrování**
  - každý síťový paket, který směrovač přijme, rozešle do všech (ostatních) směrů
    - tím vzniká „záplava“ (a duplicitní exempláře stejného paketu)
      - duplicitní pakety je třeba identifikovat a následně eliminovat
    - výhoda: pokud existuje cesta do cílového uzlu, je nalezena
      - používá se tam, kde je zapotřebí vysoká robustnost
        - aby se paket dostal ke svému cíli za všech okolností



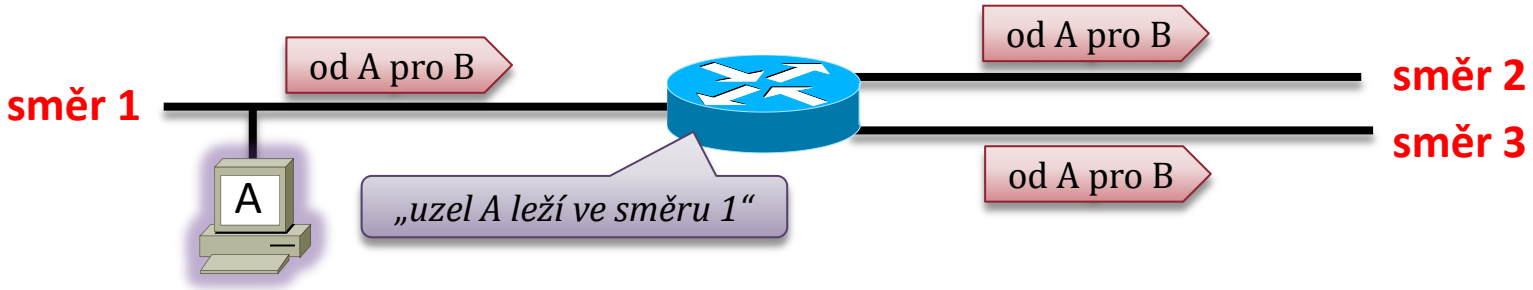
- **metoda horké brambory**
  - směrovač se snaží co nejrychleji zbavit právě přijatého síťového paketu
    - a tak jej odešle tím směrem, kterým „odejde“ nejdříve (nejrychleji)
      - obvykle: kde je nejmenší výstupní fronta
    - obvykle nejde o „správný“ směr (vedoucí k cílovému příjemci)
  - obvyklé využití: jako záložní řešení pro případ hrozícího zahlcení směrovače
    - „normálně“ využívá jiné směrování, ale jakmile začne hrozit zahlcení, přejde na tuto metodu

- **náhodné směrování**
  - obdoba „horké brambory“, směrovač se rozhoduje náhodně
    - *„nechce se mu / nestíhá přemýšlet o správném směru“*

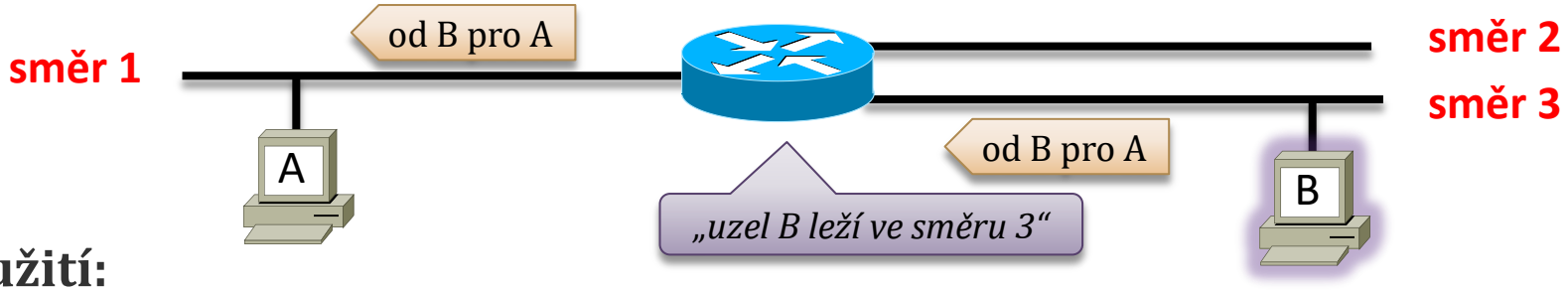


příklad: metoda zpětného učení

- **princip:**
  - na počátku směrovač neví nic o umístění uzlů ve svém okolí
  - když přijme paket od uzlu A, určený uzlu B, „naučí se“, že uzel A leží v příchozím směru
    - a samotný síťový paket rozešle do všech ostatních směrů (záplavovým směrováním)
      - protože ještě neví, kde se nachází uzel B



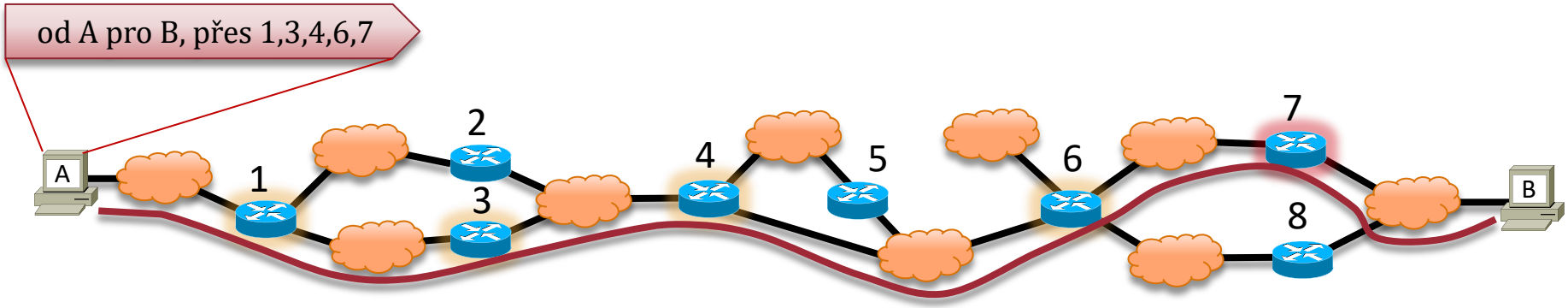
  - když přijme odpověď (paket od uzlu B, určený uzlu A), „naučí se“, kde leží uzel B
    - a jelikož již zná umístění uzlu A, předá paket cíleně již jen do směru 1, kde se A nachází



- **využití:**
  - jako metoda směrování není efektivní: učení by trvalo příliš dlouho
  - používá se na linkové vrstvě (v mostech/přepínačích), v rámci Ethernetu !!

příklad: source routing

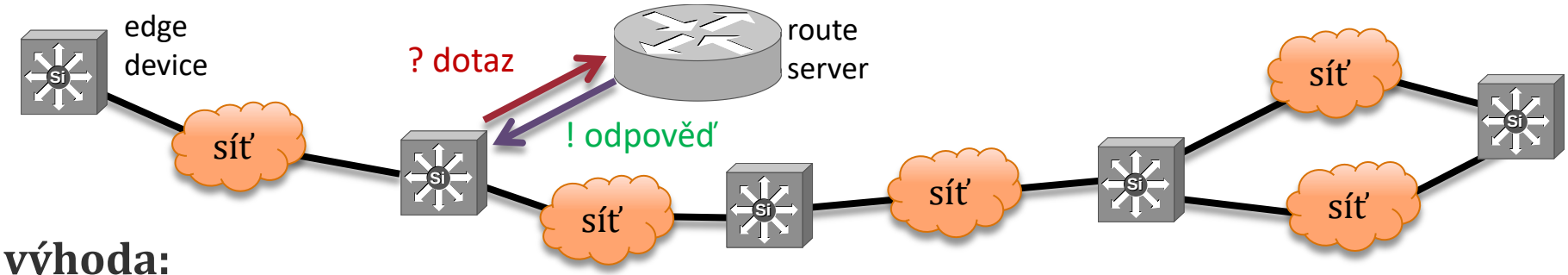
- **doslova: „směrování od zdroje“**
  - zdroj (odesilatel) předepíše paketu, kudy má být přenášen
    - vloží mu do hlavičky seznam „přestupních bodů“ (směrovačů), přes které má projít



    - odesílající uzel zjistí správnou posloupnost směrovačů pomocí záplavového směrování
      - nejprve vyšle „průzkumný“ paket, který se šíří pomocí záplavového směrování a pamatuje si posloupnost uzlů, přes které prošel
        - ten exemplář, který se dostane k cíli jako první, vrátí zpět „svou cestu“ (posloupnost směrovačů)
- **využití:**
  - jako metoda směrování (na síťové vrstvě) jen výjimečně
    - je nutná podpora síťových paketů
      - do jejich hlavičky musí být možné vložit posloupnost směrovačů („přeskoků“)
  - v praxi se používá na linkové vrstvě (mezi mosty/přepínači)
    - v rámci technologie Token Ring

centralizované směrování

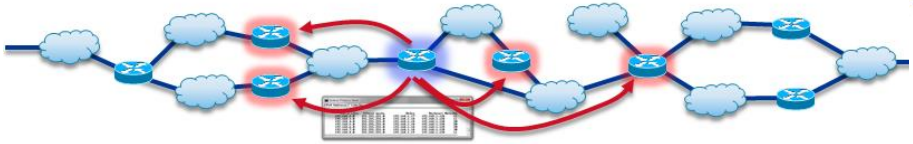
- **princip:**
  - rozhodování o dalším směru přenosu (směrování v užším slova smyslu) je soustředěno do jednoho centrálního prvku (centrální autority)
    - obvykle jde o tzv. **route server**
      - server, který jako svou službu poskytuje informaci o cestě / trase / směru dalšího přenosu
  - ostatní uzly jsou pouze „výkonné“
    - říká se jim různě, např.: edge device („okrajové zařízení“), multilayer switch, ...
    - zajišťují pouze cílené předávání (forwarding), podle pokynů centrální autority (route serveru)
      - když zařízení neví, jak naložit s konkrétním paketem, zeptá se route serveru
        - odpověď route serveru si zařízení pamatuje (ale jen po omezenou dobu, pak ji “řízeně zapomíná“)



- **výhoda:**
  - centrální autorita (route server) má k dispozici veškeré informace, může se rozhodovat (hledat cesty) velmi pružně, může měnit algoritmus svého rozhodování
- **nevýhoda:**
  - single point of failure: s výpadkem route serveru je vše mimo provoz

distribuované směrování

- **připomenutí:**
  - distribuované směrování = jednotlivé směrovače vzájemně spolupracují
    - míra jejich spolupráce se ale může lišit, i dosti výrazně
- **varianta „distance-vector“ (RIP)**
  - každý směrovač má jen neúplnou informaci o topologii celé soustavy sítí
  - výpočet optimálních cest je distribuovaný a průběžný
    - krok:
      - směrovač předá „celou svou informaci“ (svou směrovací tabulku) všem sousedním směrovačům
      - sousední směrovač si z toho „dopočítá“ svou směrovací tabulku
    - opakuje se neustále
      - v praxi (např.) každých 30 sec.
  - nevýhoda:
    - velká režie, špatně škálovatelné



- **varianta „link-state“ (OSPF)**
  - každý směrovač má úplnou informaci o topologii celé soustavy sítí
    - musí sledovat dostupnost sousedních směrovačů
    - pokud nějaký sousední směrovač přestane (nebo začne) být dostupný, musí to oznámit všem směrovačům v celé soustavě
      - stačí jen při změně
  - výpočet optimálních cest je „lokalizovaný“
    - každý uzel si počítá optimální cesty sám
  - výhoda:
    - menší režie, lépe škálovatelné

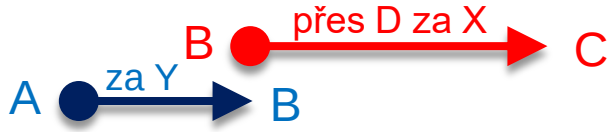
směrování distance-vector

- **princip:**
  - každý směrovač si udržuje tabulku svých nejmenších vzdáleností od všech ostatních uzlů (“vektorů“)
    - proto: distance-vector

  - směrovače si tyto informace vzájemně vyměňují
    - informace typu:
      - já (B) se dostanu k uzlu C za cenu X (přes D)

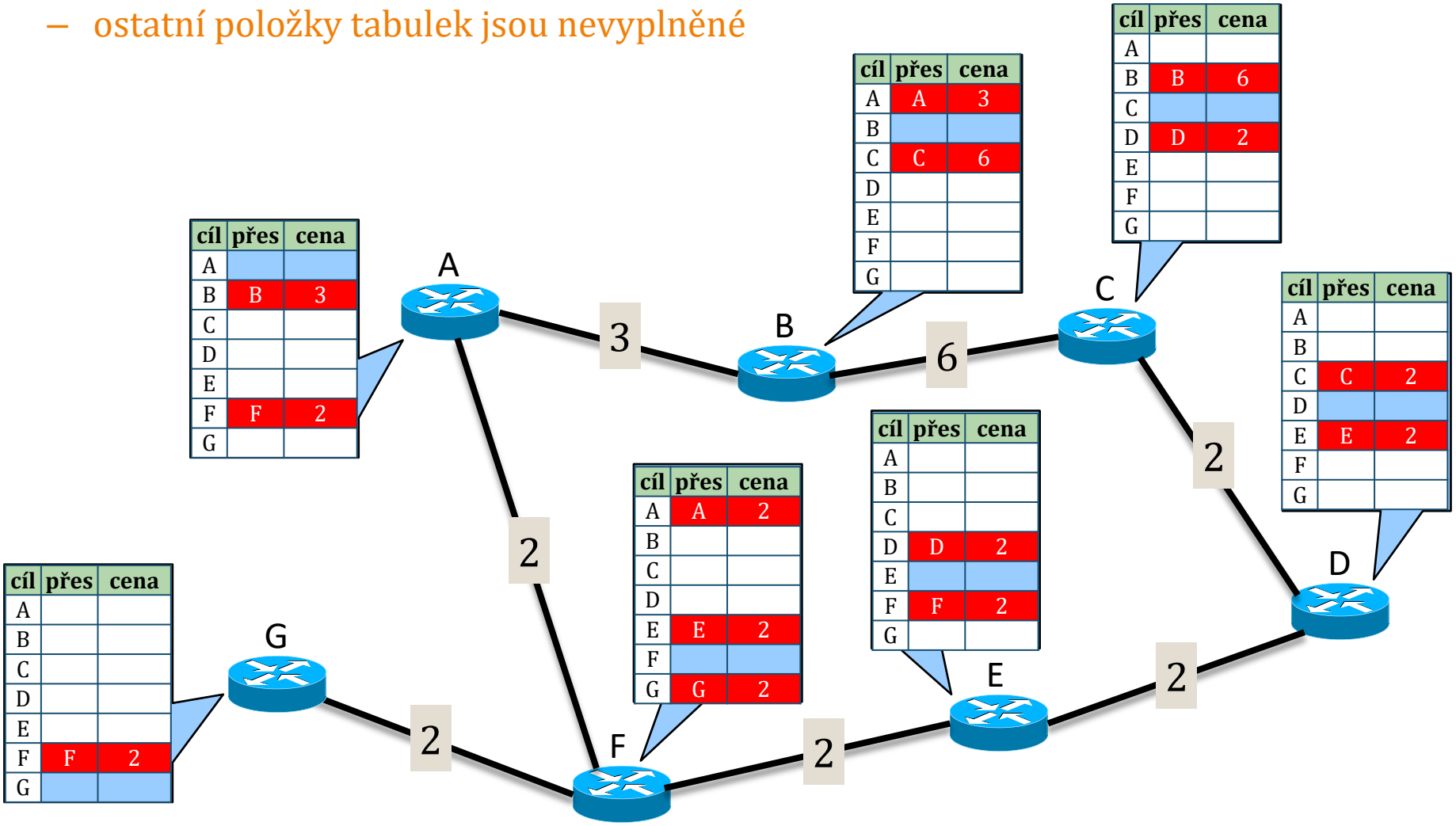
    - jde vlastně o průběžnou výměnu obsahu celých směrovacích tabulek
      - … ale výměna probíhá jen mezi přímými sousedy, ne mezi všemi směrovači sítě !!!!!
  - všechny směrovače si průběžně aktualizují své „vektory“ (nejkratší vzdálenosti)
    - na základě vektorů, které dostávají od svých sousedů
    - výpočet optimálních cest je fakticky distribuovaný
      - když někdo udělá chybu, splete i ostatní

- **problémy:**
  - objemy přenášených dat (pro potřeby aktualizace) jsou hodně velké
    - není to vhodné pro velké sítě
  - problémy jsou i s konvergencí:
    - „dobré zprávy“ se šíří rychle
      - to, že někde existuje kratší cesta
    - „špatné zprávy“ se šíří pomalu
      - to, že někde přestala být cesta průchodná
    - problém „count-to-infinity“
      - hodnota cesty přes neprůchodnou cestu se zvyšuje v každém kroku o 1
      - trvá to hodně dlouho, než se hodnota zvýší tak aby signalizovala neprůchodnost
        - lze řešit úpravou algoritmu hledání cest a způsobu předávání vektorů



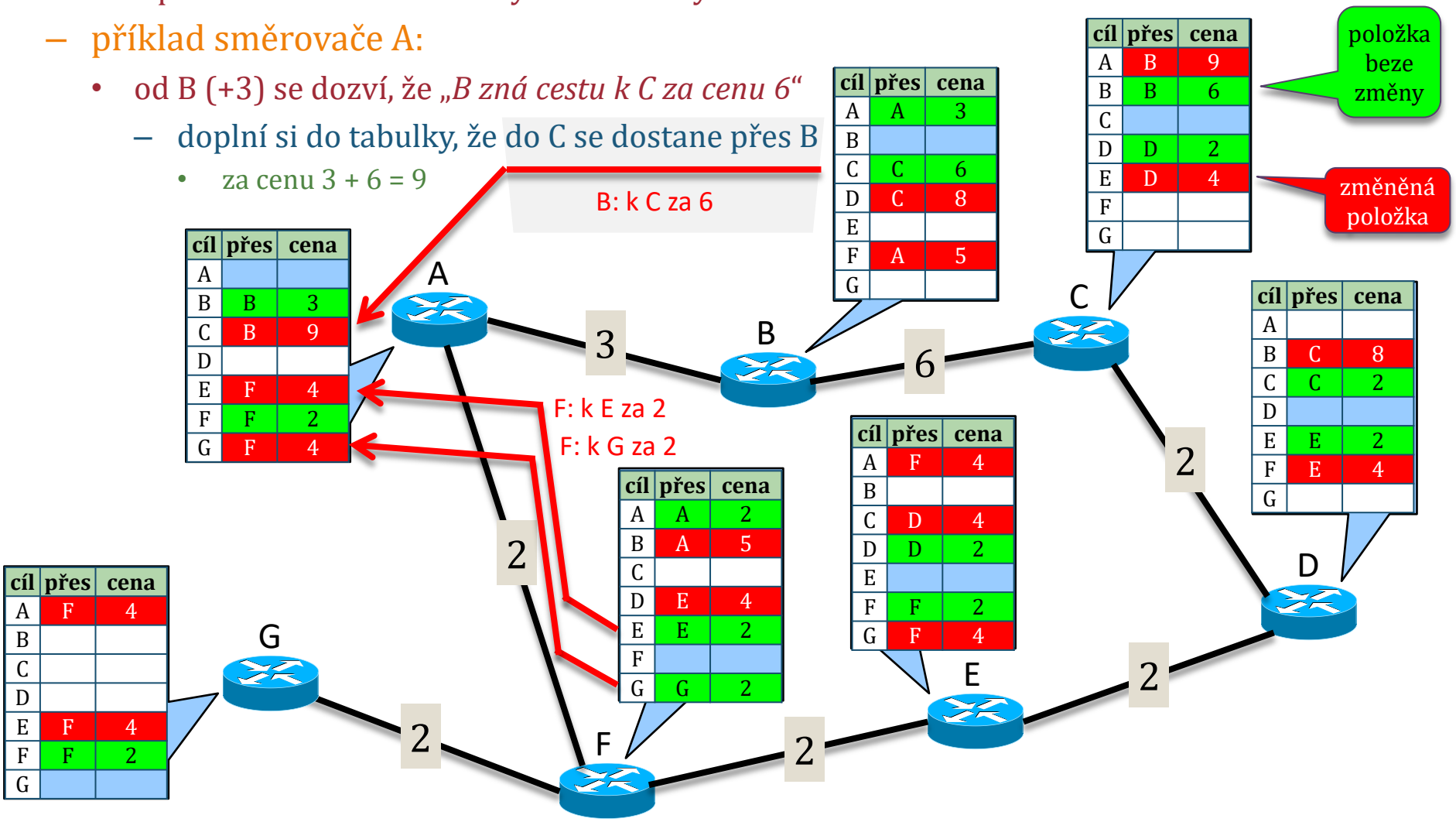
příklad (směrování distance-vector)

- **počáteční stav:**
  - každý uzel „zná“ jen své přímé sousedy
    - “vektory” k nim má zanesené ve své směrovací tabulce
  - ostatní položky tabulek jsou nevyplněné



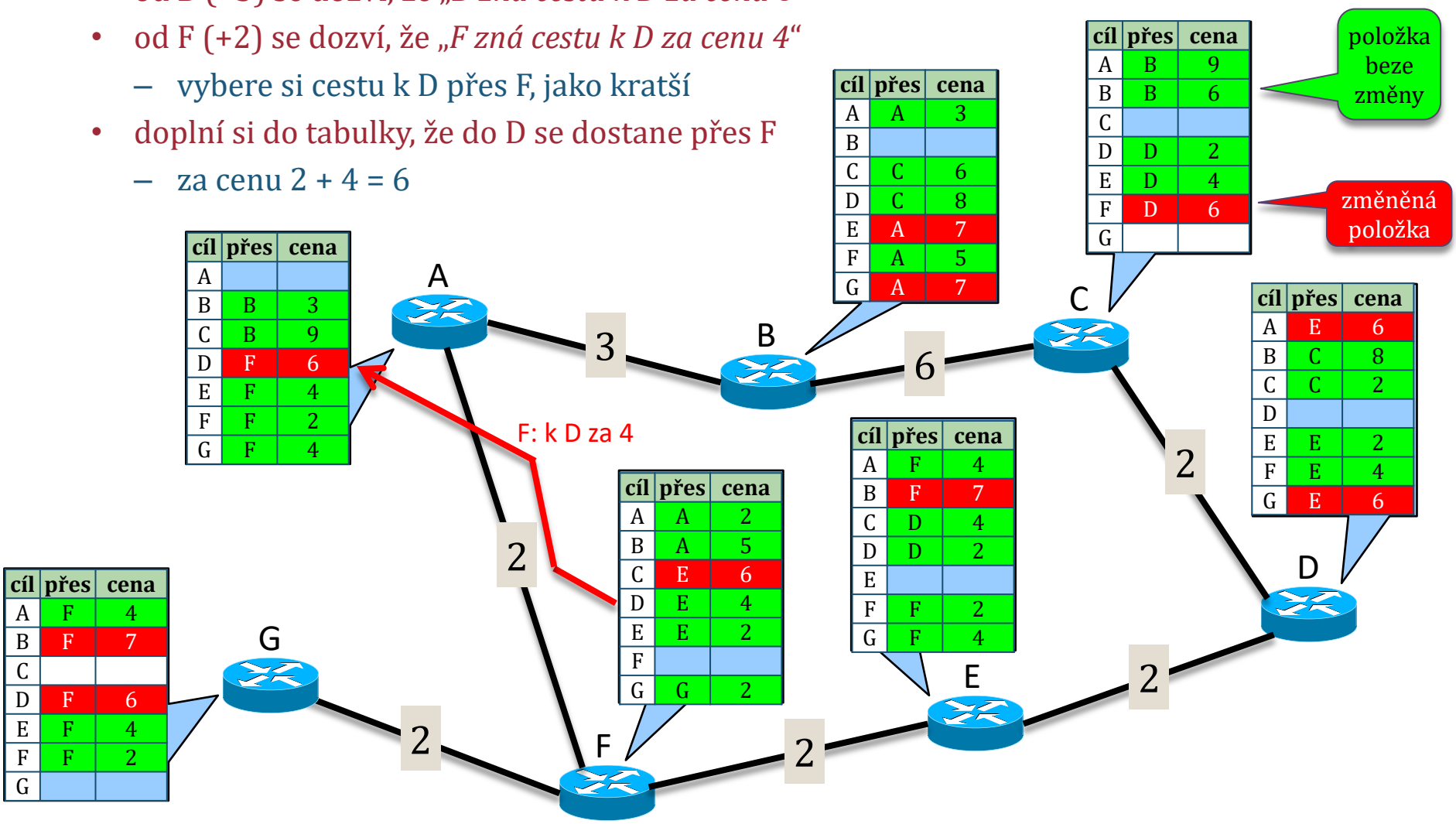
příklad (směrování distance-vector)

- **stav po 1. kroku:**
  - sousední směrovače si předaly své směrovací tabulky
    - a podle nich si aktualizovaly své "vektory"
  - příklad směrovače A:
    - od B (+3) se dozví, že *„B zná cestu k C za cenu 6"*
      - doplní si do tabulky, že do C se dostane přes B
        - za cenu 3 + 6 = 9



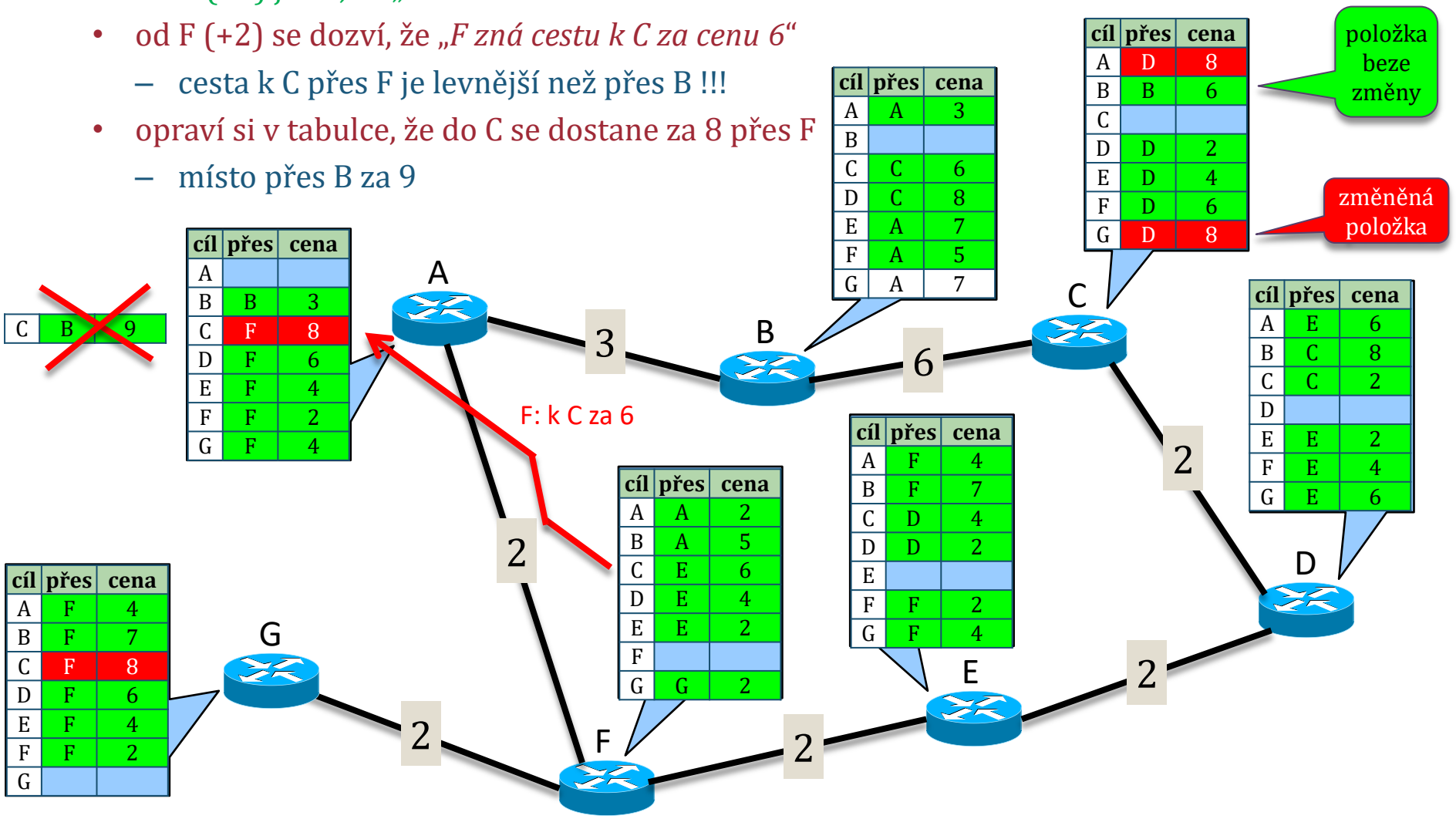
příklad (směrování distance-vector)

- **stav po 2. kroku:**
  - příklad směrovače A:
    - od B (+3) se dozví, že *„B zná cestu k D za cenu 8“*
    - od F (+2) se dozví, že *„F zná cestu k D za cenu 4“*
      - vybere si cestu k D přes F, jako kratší
    - doplní si do tabulky, že do D se dostane přes F
      - za cenu 2 + 4 = 6



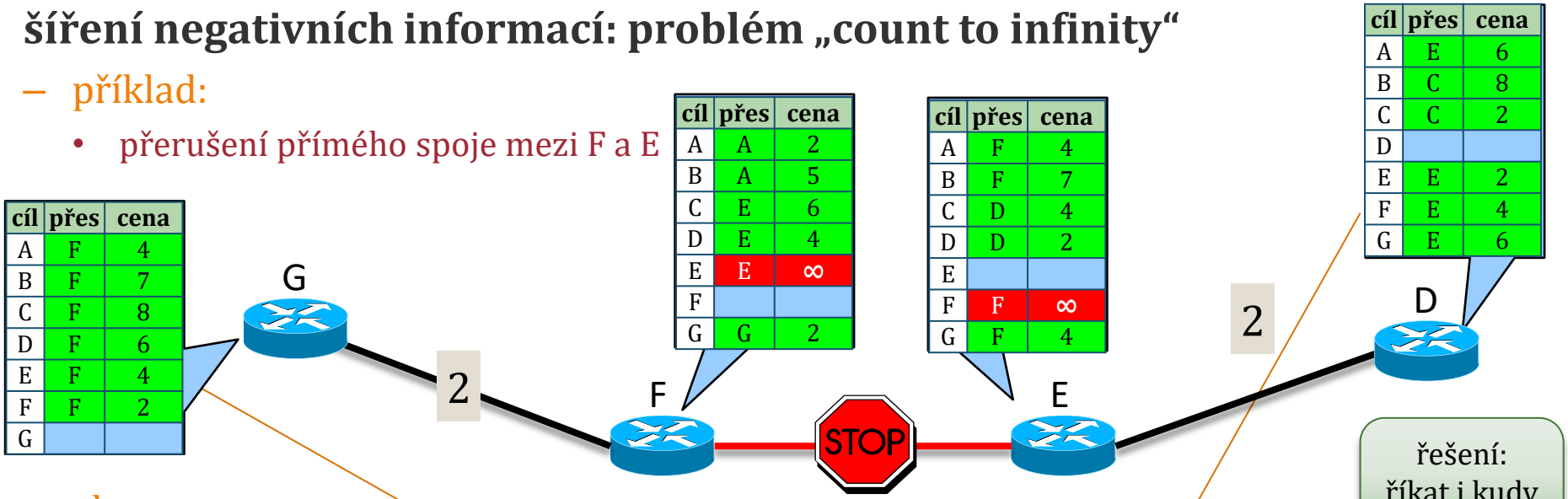
příklad (směrování distance-vector)

- **stav po 3. kroku:**
  - příklad směrovače A:
    - od B (+3) již ví, že „*B zná cestu k C za cenu 6*“
    - od F (+2) se dozví, že „*F zná cestu k C za cenu 6*“
      - cesta k C přes F je levnější než přes B !!!
    - opraví si v tabulce, že do C se dostane za 8 přes F
      - místo přes B za 9

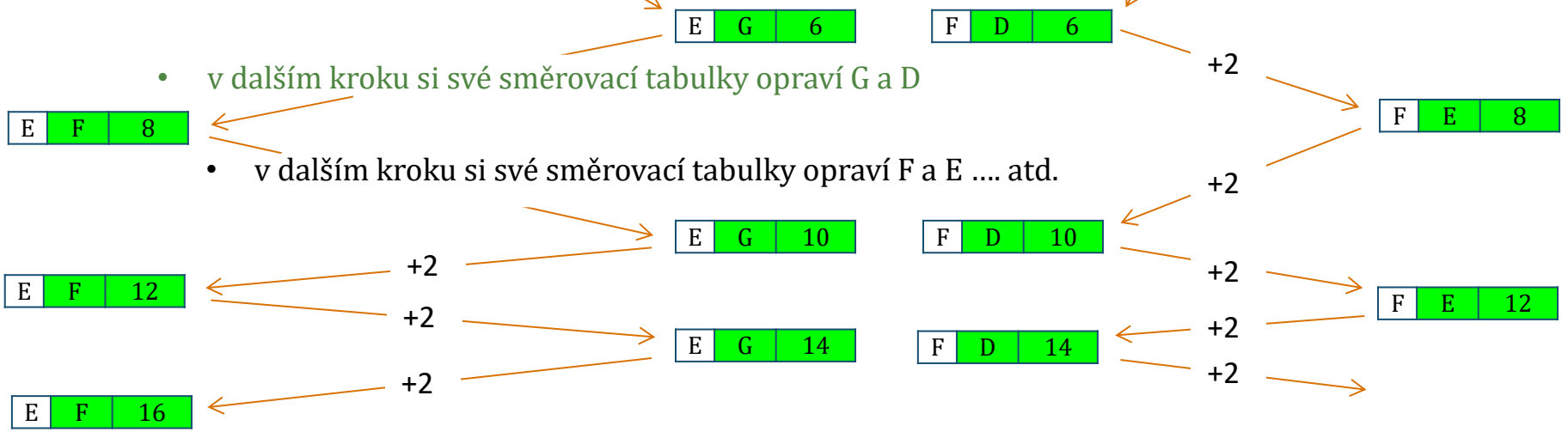


příklad (směrování distance-vector)

- **šíření negativních informací: problém „count to infinity“**
  - příklad:
    - přerušení přímého spoje mezi F a E



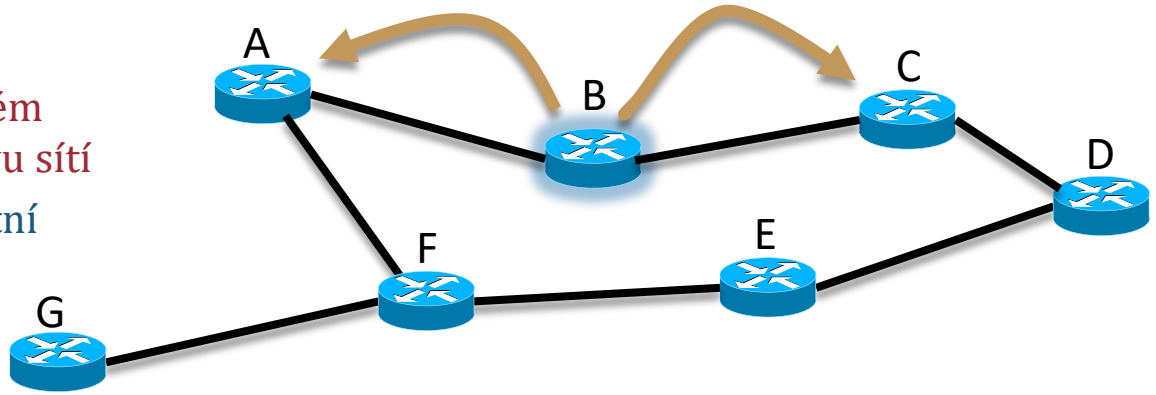
  - ale:
    - F se dozví od G, že „*G zná cestu k E za 4*“; E se dozví od D, že „*D zná cestu k F za 4*“
      - F a E si podle toho opraví své směrovací tabulky



    - v dalším kroku si své směrovací tabulky opraví G a D
    - v dalším kroku si své směrovací tabulky opraví F a E .... atd.

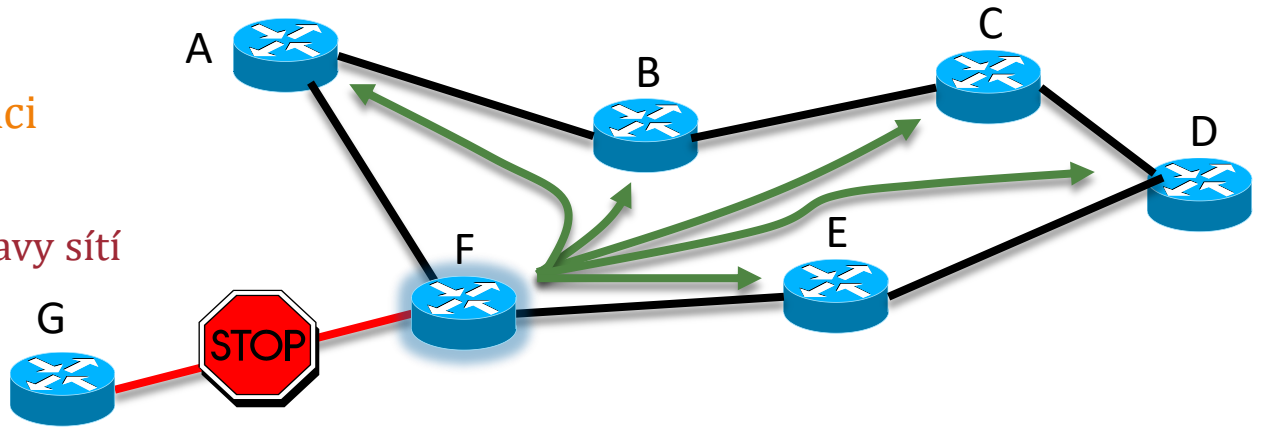
RIP: Routing Information Protocol

- **protokol typu distance-vector**
  - pochází z doby vzniku Internetu a TCP/IP
    - byl zabudován do BSD Unixu již v roce 1982
  - používaná metrika: počet přeskoků
    - ale jen do maxima 15!!!
      - nekonečno = 16
    - protože na vyjádření vzdálenosti má vyhrazeny jen 4 bity!!!
  - nevýhody:
    - nelze jej použít pro větší sítě
      - max. 15 „přeskoků“ (hop-ů)
    - málo stabilní
      - problém „count to infinity“
    - špatně škálovatelný
      - jeho režie rychle roste
    - případná chyba v distribuovaném výpočtu postihne celou soustavu sítí
      - chyba jednoho „splete“ ostatní
- **způsob fungování**
  - všechny své vektory (svou směrovací tabulku) rozesílá každých 30 sekund ke všem sousedním směrovačům
    - obsahuje až 25 cílových sítí
      - vektor obsahuje jen cílovou síť a cenu (nikoli „kudy cesta vede“)
    - rozesílá je vložené do UDP datagramu
      - na port č. 520
    - pokud není "distance-vector" přijat do 180 sekund, je soused/spoj brán jako "mrtvý„ (nedostupný)
  - zpracování aktualizačních informací
    - řeší démon routed na úrovni OS



směrování link-state

- **princip:**
  - každý směrovač má úplnou informaci o topologii celé soustavy (vzájemně propojených) sítí
    - na počátku (po startu) si ji vyžádá od svého souseda (jinak si ji musí budovat postupně)
  - každý směrovač si počítá optimální cesty sám (má k tomu potřebné informace)
    - výpočet není distribuovaný
      - pokud udělá chybu ve výpočtu jeden směrovač, ostatní to nesplete
        - ostatní směrovače mají šanci napravit důsledky chyby daného směrovače
  - aktualizační informace není nutné rozesílat neustále (jako u distance-vector)
    - ale stačí jen při nějaké změně (stavu linky mezi 2 uzly: proto „link-state“)
      - ale pozor: informaci o změně je nutné rozeslat **všem** směrovačům v celé soustavě !!!!
        - nestačí jen těm sousedním, jako u distance-vector
    - lze je rozesílat „1x za dlouhou dobu, pro osvěžení“ (např. 1x za 30 minut)
- **výhody:**
  - menší režie na aktualizaci
  - lépe škálovatelné
    - hodí se i pro větší soustavy sítí

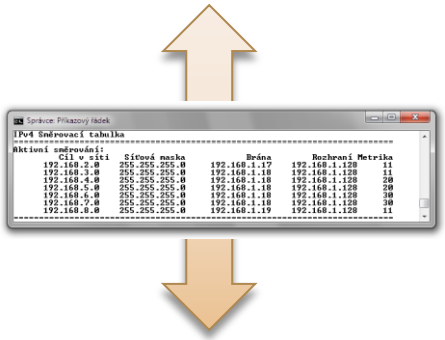


srovnání

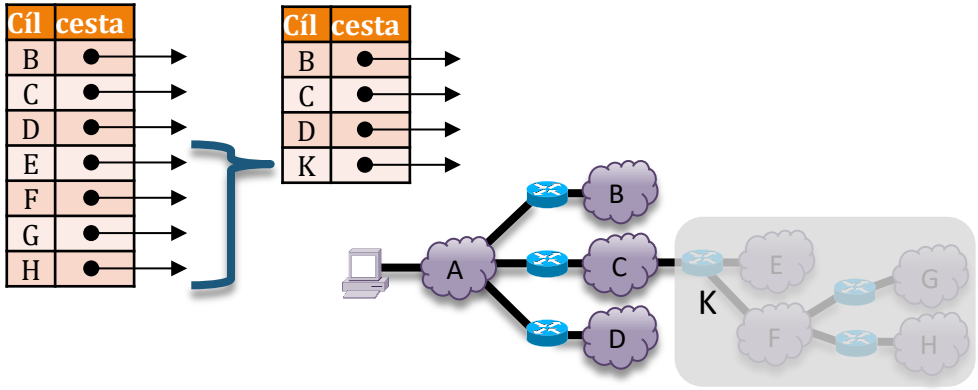
	Distance-vector	Link-state
jak vnímá topologii sítě?	"pohledem svých sousedů"	vnímá celou topologii celé sítě
způsob výpočtu cest v síti	výpočet je distribuovaný (každý něco přičte k výsledku svých sousedů)	každý si počítá všechno sám
konvergence výpočtu	pomalá	rychlá
chyba ve výpočtu	ovlivní ostatní směrovače	neovlivní ostatní výpočty
aktualizace	musí být časté a pravidelné – každých 30 sekund	stačí při změně (jinak pro osvěžení každých 30 minut)
komu se posílají aktualizační informace?	přímým sousedům	všem uzlům v síti
škálovatelnost	špatná (max. 15 hop-ů)	lepší
příklad konkrétního protokolu	RIP (Routing Information Protocol)	OSPF (Open Shortest Path First)

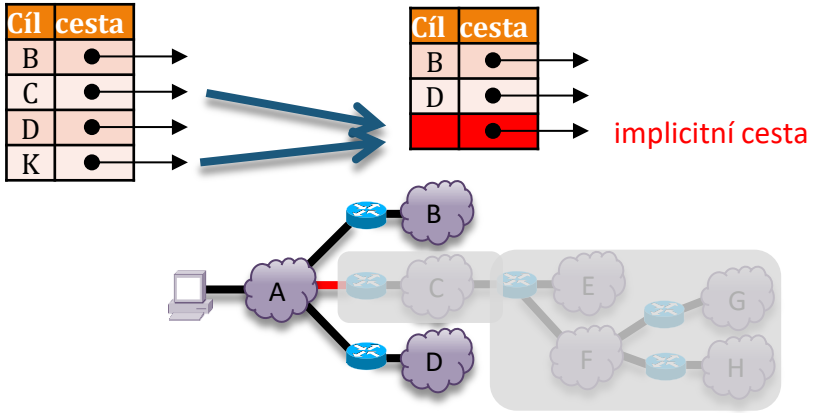
problém velikosti směrovacích informací

- **v praxi nastává problém:**
  - směrovací (i forwardovací) tabulky by měly být co nejmenší
    - aby se s nimi dalo co nejrychleji pracovat, aby nebyly moc drahé
  - **směrovací (i forwardovací) tabulky se neustále zvětšují**
    - s tím, jak roste velikost soustavy vzájemně propojených sítí
      - protože roste objem informací o topologii celé soustavy
        - a ještě rychleji roste objem „aktualizačních“ informací, nutných pro fungování algoritmů distance-vector a link-state



- **pro minimalizaci objemu směrovacích (i forwardovacích) tabulek se používají dvě hlavní metody:**
  - agregace položek
    - skupina položek, které „*vedou stejným směrem*“, se (za určitých okolností) dá sloučit v jednu společnou položku
  - implicitní cesta (default route)
    - „všechno ostatní“ (kromě explicitně určených směrů) se posílá implicitní cestou

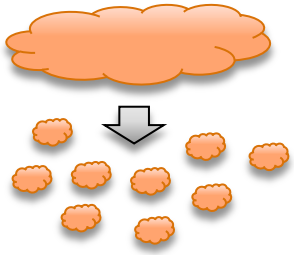




problém velikosti aktualizačních informací

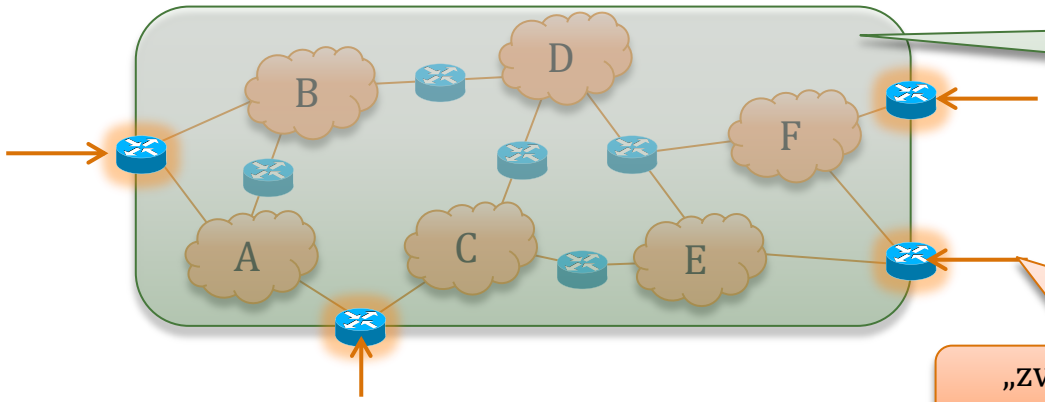
- **velikost směrovacích tabulek**
  - je problémem, ale nějak se daří jej řešit
- **horší problém je:**
  - velikost aktualizačních informací
    - které si vyměňují jednotlivé směrovače
      - v rámci algoritmů distance-vector nebo link-state
  - jejich objem (obvykle) roste ještě rychleji, než objem směrovacích tabulek
    - a pro větší soustavy vzájemně propojených sítí přestává být únosný
      - aktualizace by „spotřebovaly“ rozhodující část přenosové kapacity
- **(jediné) možné řešení:**
  - dekompozice
    - **lokalizovat detailní směrovací informace**
      - „rozbít“ celou soustavu vzájemně propojených sítí na „vhodně malé části“
        - a detailní směrovací informace vést a aktualizovat jen v rámci těchto malých částí

- **připomenutí:**
  - směrování „distance-vector“ je nejhůře škálovatelné
    - hodí se jen pro nejmenší soustavy vzájemně propojených sítí
      - jinak je režie na aktualizaci neúnosně velká
  - směrování „link-state“ je lépe škálovatelné
    - hodí se pro větší soustavy vzájemně propojených sítí
      - má menší režii na aktualizaci
  - ale: pro největší soustavy vzájemně propojených sítí se „link-state“ už také nehodí
    - režie na aktualizaci je už také příliš (neúnosně) velká



směrovací domény

- **směrovací doména**
  - obecné označení pro onu „vhodně malou“ část soustavy vzájemně propojených sítí
    - v rámci které jsou vedeny a aktualizovány detailní směrovací informace
      - v praxi se lze setkat i s dalšími názvy: **oblast** (area), **autonomní systém** (AS), ...



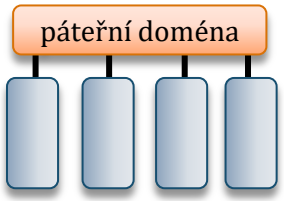
„uvnitř“ směrovací domény jsou dostupné detailní informace o topologii sítí A, B, C, D, E a F

tyto informace nejsou „zvenku“ vidět (dostupné)

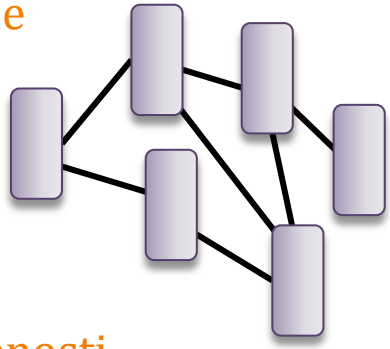
„zvenku“ je dostupná pouze informace o tom, že „uvnitř“ směrovací domény jsou dostupné sítě A až F

  - další vlastnosti směrovací domény:
    - má několik (málo) vstupně/výstupních bodů, skrze které je propojena s dalšími doménami
      - na hranici jsou umístěny tzv. hraniční směrovače 
    - skrze tyto body (hraniční směrovače) „vystupuje ven“ pouze podstatně „menší“ informace
      - obvykle jen výčet (interval) sítí, nacházejících se ve směrovací doméně
  - obecně:
    - je třeba rozlišovat výměnu (a aktualizaci) směrovací informací:
      - uvnitř směrovací domény – používají se „intra-domain“ algoritmy (jako RIP či OSPF)
      - mezi směrovacími doménami – jsou nutné „extra-domain“ algoritmy (jako BGP)

hierarchické směrování



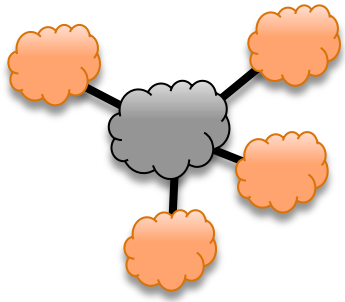
- **princip:**
  - jde o směrování za existence směrovacích domén
    - tj. kdy je celá soustava sítí (z hlediska směrování) rozdělena do více domén
  - jde o jediné řešení, které připadá v úvahu pro hodně velké soustavy vzájemně propojených sítí (jako je celosvětový Internet)
- **proč hierarchické?**
  - protože zpočátku musela mezi doménami existovat určitá hierarchie
    - některé domény musely být „páteřní“ a propojovat ostatní domény
  - dnes už takovéto hierarchické uspořádání není nutné
    - a jednotlivé směrovací domény mohou být uspořádány libovolně



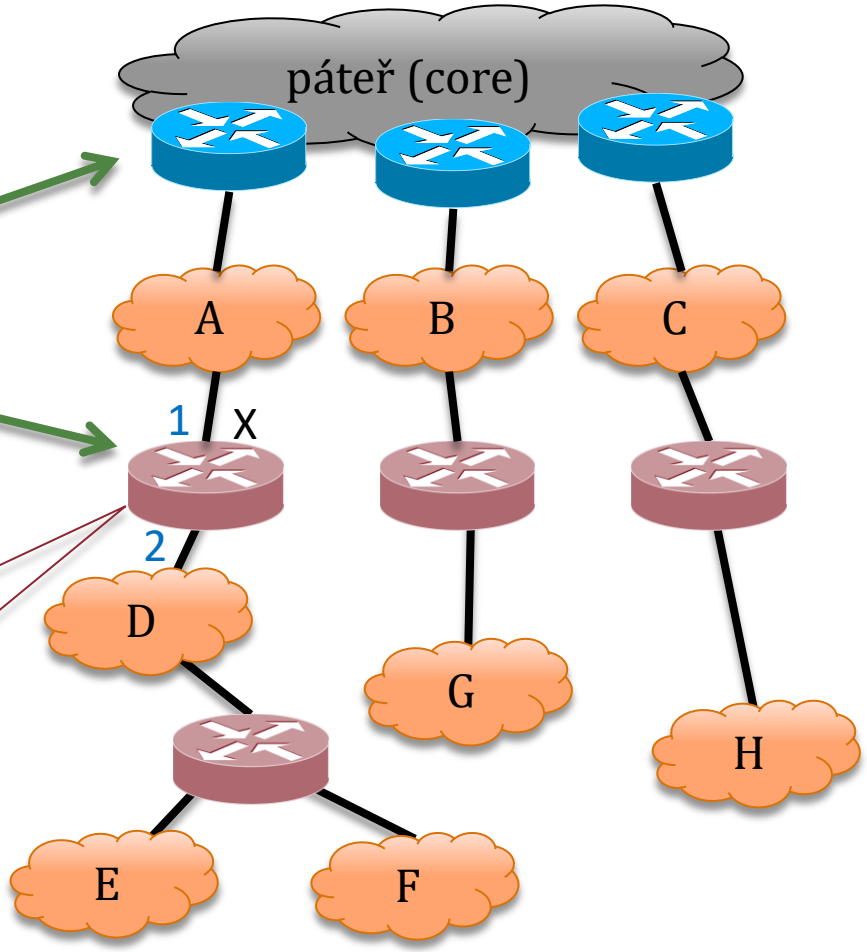
- **významný důsledek:**
  - mezi směrovacími doménami jsou šířeny pouze informace o dostupnosti
    - tzv. **reachability information**
      - „intervalové“ informace typu OD-DO („*v této doméně jsou sítě od A/x do B/y*“)
        - informace o dostupnosti jsou velmi malé (obvykle jen síťové prefixy)
  - mění se logika směrování (hledání cest)
    - (mezi doménami) už není možné hledat optimální cesty
      - protože není k dispozici úplná informace o ceně cesty (ohodnocení hran grafu)
    - místo toho se hledá „alespoň nějaká cesta“
      - taková, která „*vede k cíli*“

příklad: vývoj směrování v rámci Internetu

- **úplně na počátku:**
  - Internet byl jednou jedinou směrovací doménou
    - každý směrovač měl úplnou informaci o topologii celého Internetu
      - časem se to stalo neúnosné – směrovacích informací bylo příliš mnoho



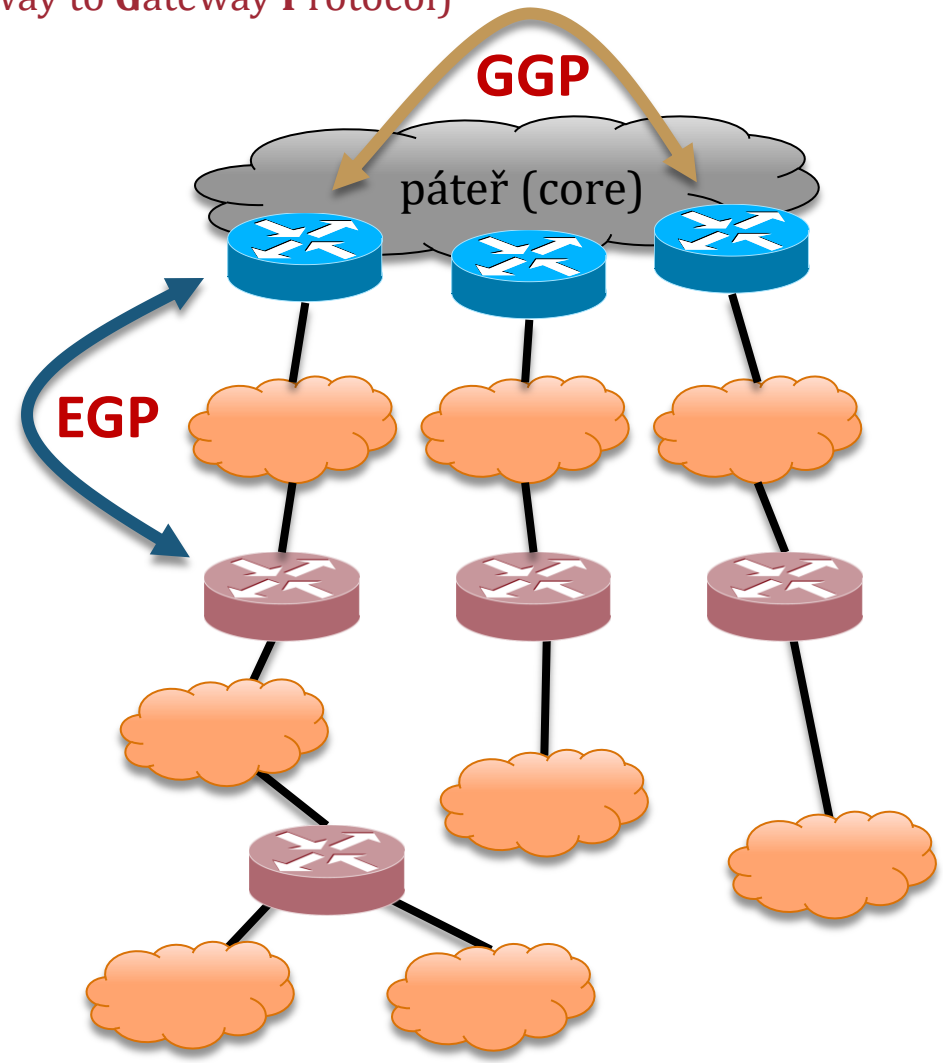
- **později:**
  - Internet byl rozdělen na páteř (core) a "ostatní" (non-core)
    - směrovače v páteři (core gateways) měly úplné směrovací informace
    - směrovače mimo páteř (non-core gateways) měly podrobné směrovací informace jen o své „oblasti"
      - znaly cestu jen do „svých" podsítí
      - vše ostatní směrovaly pomocí implicitní cesty do páteře



směrovač X (mimo páteř) směruje:
- k sítím D, E a F přes rozhraní 2
- vše ostatní přes rozhraní 1 (jako default route)

protokoly GGP, EGP a IGP

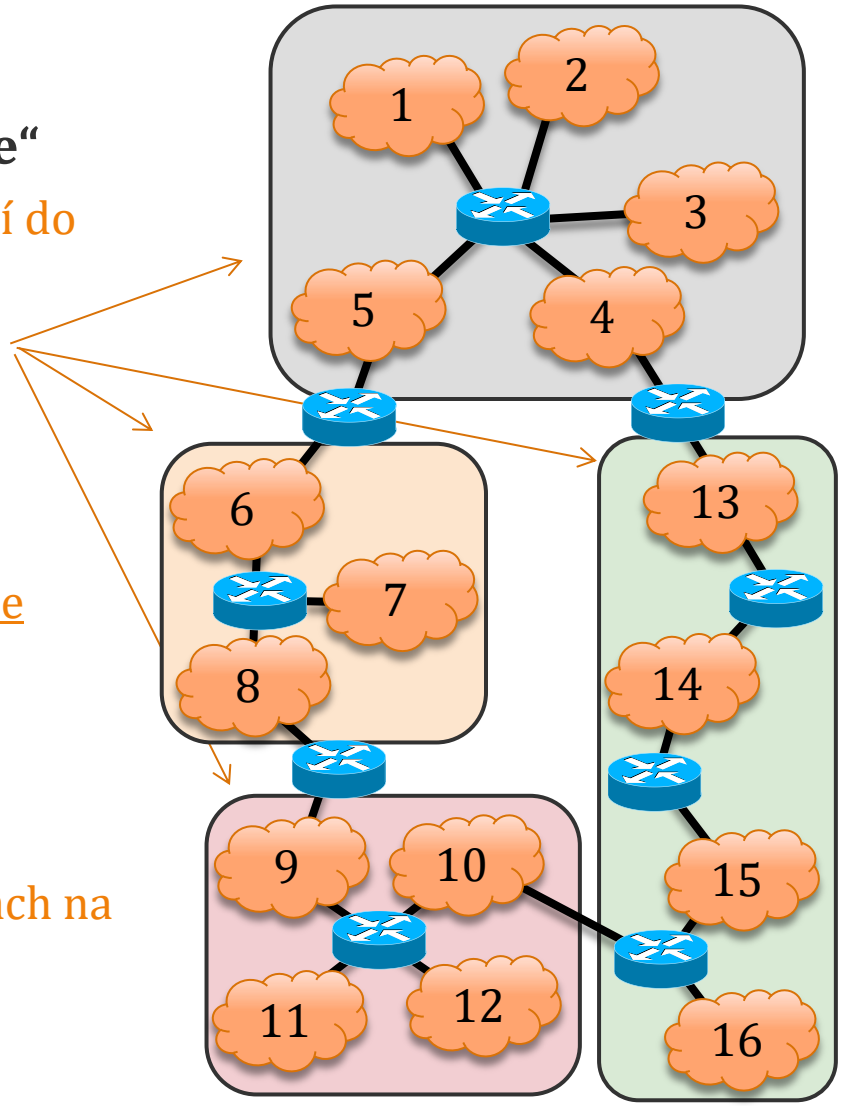
- **nutné předpoklady pro rozdělení na páteřní (core) / nepáteřní (non-core):**
  1. možnost vzájemné komunikace mezi směrovači v páteři (core gateways)
     - k tomu byl vytvořen protokol **GGP** (**G**ateway to **G**ateway **P**rotocol)
  2. možnost komunikace mezi směrovači v páteři (core) a směrovači mimo páteř (non-core)
     - k tomu sloužila celá skupina protokolů, označovaná jako **EGP** (**E**xterior **G**ateway **P**rotocol)



- **jde o „dvouúrovňové“ řešení**
  - které vydrželo jen po určitou dobu rozvoje Internetu
  - ale časem se také stalo neúnosné
    - kvůli velkému objemu směrovacích informací v páteři
      - se kterými musely pracovat páteřní směrovače

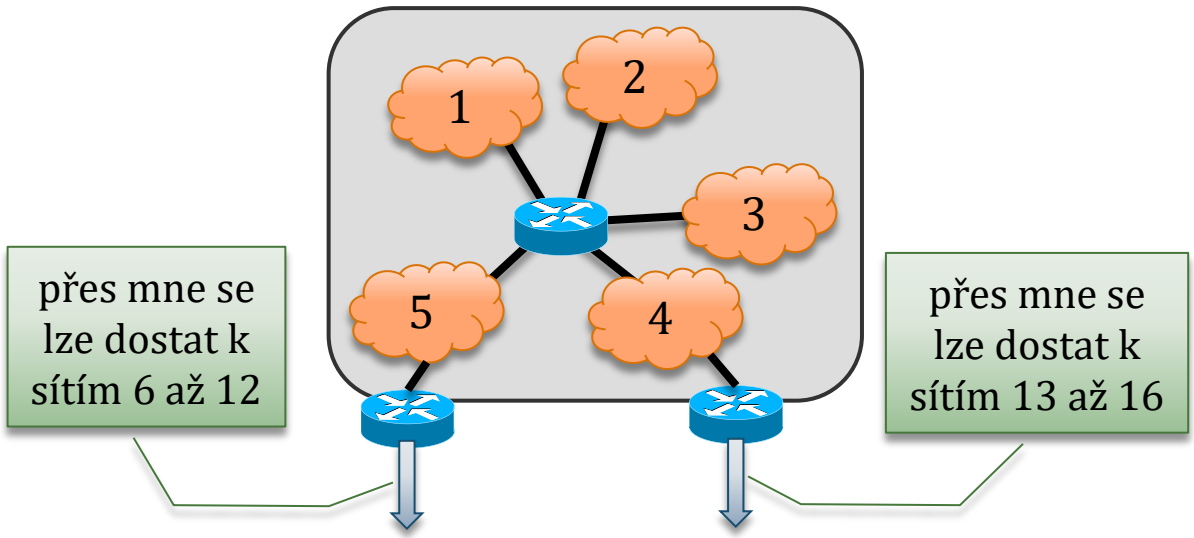
autonomní systémy v Internetu

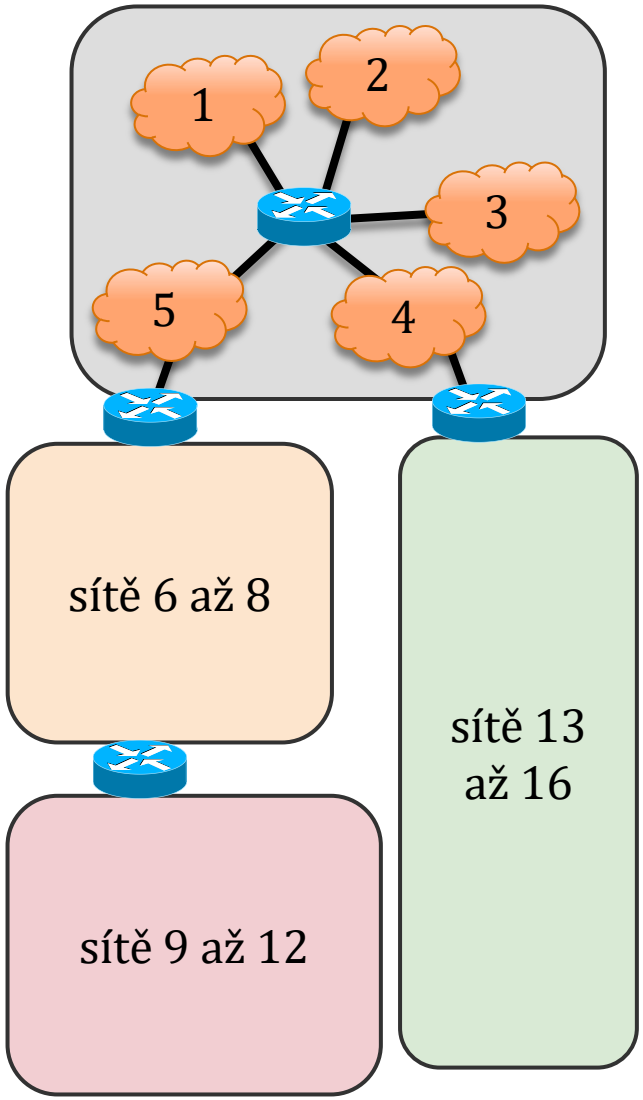
- **ani „dvouúrovňové řešení“ nebylo dostatečně škálovatelné**
  - při určité velikosti Internetu se stalo neudržitelné
    - páteřní směrovače musely pracovat s neúnosně velkými objemy informací
- **řešením byla pouze důsledná „lokalizace“**
  - soustředění detailních směrovacích informací do menších celků
    - směrovacích domén (routing domains)
- **autonomní systémy**
    - tak se v Internetu označují směrovací domény
  - kvůli tomu, že si mohou samy (autonomně) rozhodovat o detailním směrování uvnitř sebe sama
    - obvykle: na principu distance-vector, link-state
      - nezávisle na tom, jaký způsob směrování si vyberou v jiném autonomním systému
  - také si (autonomně) rozhodují o svých vazbách na ostatní autonomní systémy
    - jsou identifikovány čísly, které přiděluje IANA
      - čísly AS: například AS 2852 (AS CESNET-u)



směrování na základě dostupnosti

- **mění se pravidla směrování mezi doménami (mezi autonomními systémy)**
  - místo „optimální“ (nejlevnější) cesty se hledá „alespoň nějaká“ cesta
- **důvod:**
  - algoritmy distance-vector a link-state zde nejsou použitelné
    - protože se nepracuje s ohodnocením hran
      - není známa „cena“ cesty přes směrovací domény
        - nejčastěji je jen jedna možná cesta
  - používají se algoritmy typu path vector
    - které pracují s celými cestami
      - „průchody“ přes směrovací domény



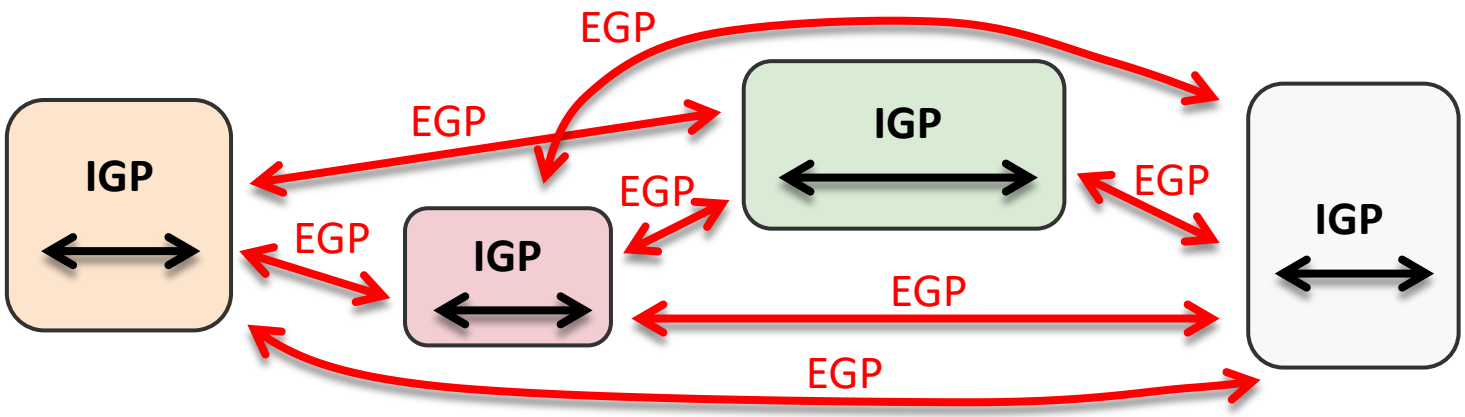


protokoly EGP a IGP

- **pro hierarchické směrování (za existence autonomních systémů, AS) jsou nutné dvě různé skupiny „směrovacích“ protokolů**
  - pro aktualizaci směrovacích informací a hledání optimálních/“alespoň nějakých“ cest

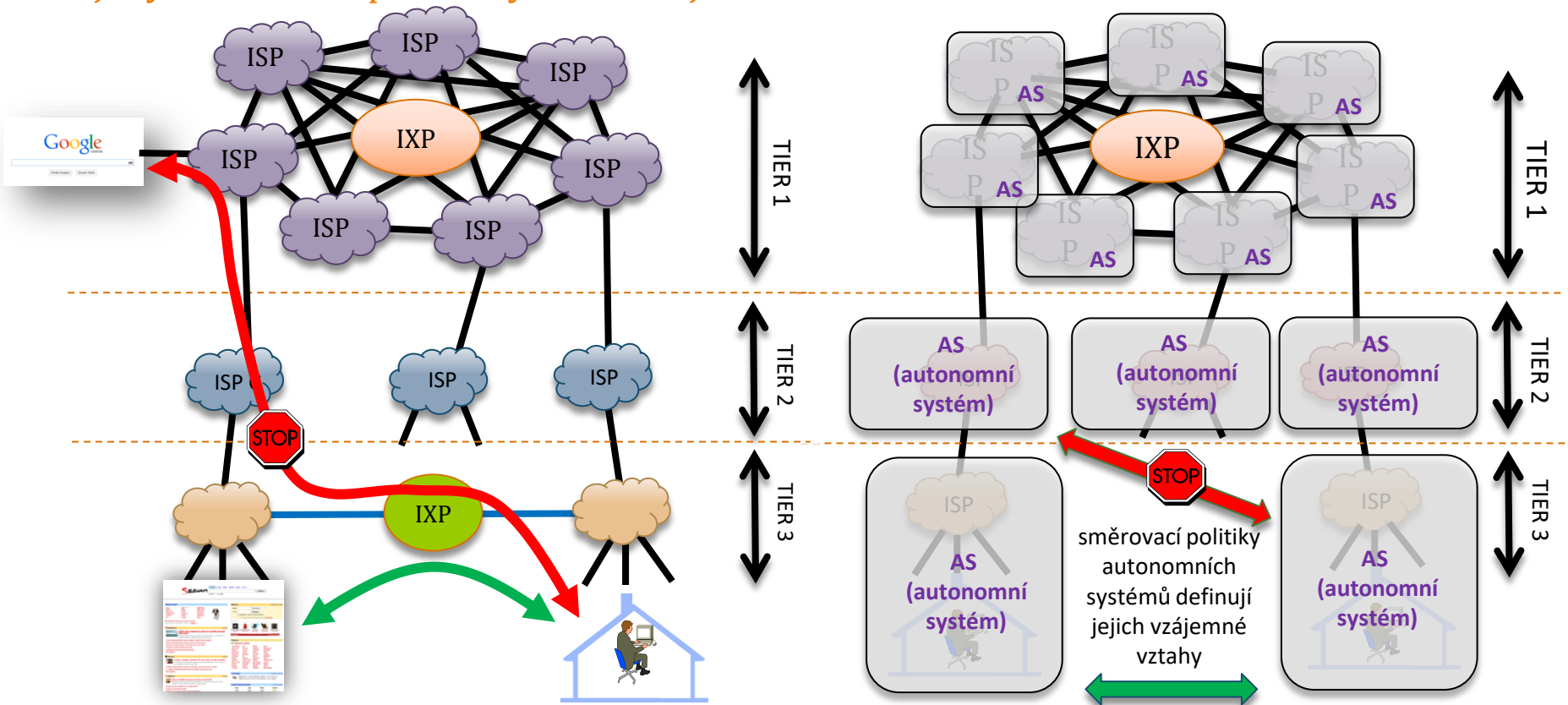
- **IGP: pro směrování „uvnitř AS“**
  - zde se používají stejné protokoly, jako v menších soustavách vzájemně propojených sítí
    - na principu distance-vector nebo link-state
      - např. RIP a OSPF
  - souhrnně se tyto protokoly označují jako **Interior Gateway Protocols**
    - zkratkou **IGP**

- **EGP: pro směrování „mezi AS“**
  - zde se musí používat jiné (speciální) protokoly, než „uvnitř“ AS
    - dnes je v Internetu nejpoužívanější protokol BGP (Border Gateway Protocol)
      - verze 4 (BGP 4)
  - souhrnně se tyto protokoly označují jako **Exterior Gateway Protocols**
    - zkratkou **EGP**



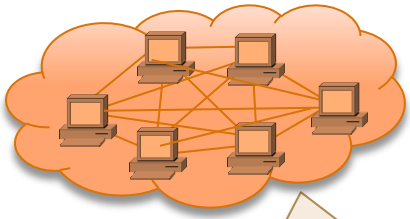
směrovací politiky a peering v Internetu

- **směrovací politika (autonomního systému)**
  - je souhrn pravidel pro směrování „uvnitř“ autonomního systému i „navenek“
    - včetně toho, se kterými (jinými) AS bude daný AS komunikovat a předávat si data
      - typicky: síť (soustava sítí) jednoho internetového providera (ISP) tvoří jeden AS
- **peering**
  - řešení, které umožňuje využít (přímé) propojení různých AS jen pro takový provoz, jaký si vlastníci příslušných AS vzájemně dohodnou



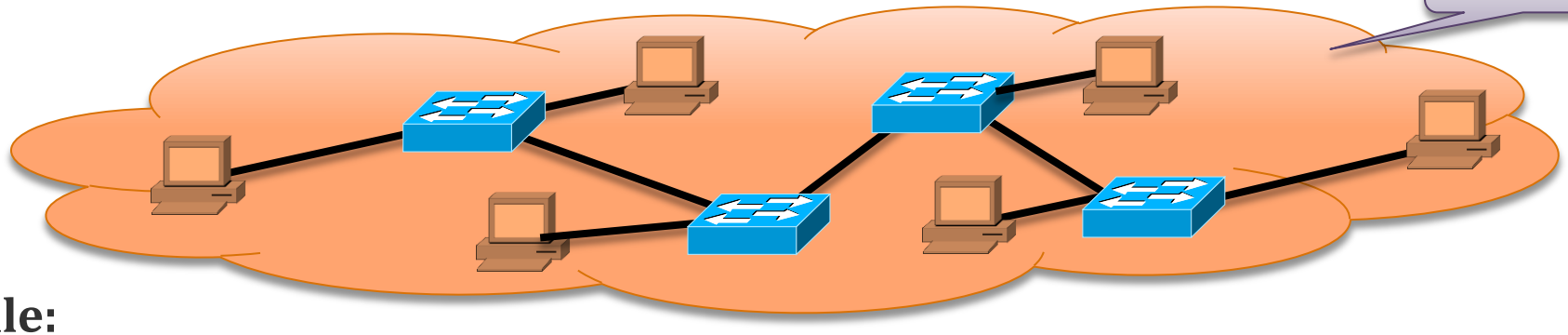
směrování na linkové vrstvě (L2)

- **připomenutí:**
  - směrování se odehrává na síťové vrstvě (L3), zajišťují jej směrovače
    - L3: uzly v dané síti „ví“ o uzlech ve stejné síti i o uzlech v jiných sítích
      - „ví i o tom“, že k uzlům v jiných sítích musí doručovat data jinak, než k uzlům ve vlastní síti
        - je-li příjemce ve stejné síti: jde o tzv. **přímé doručování**
        - je-li příjemce v jiné síti: jde o tzv. **nepřímé doručování** (fakticky: **směrování**)
    - L2: uzly v dané síti „ví“ jen o uzlech ve stejné síti
      - myslí si, že mají přímé spojení se všemi ostatními uzly v dané síti
        - dochází pouze k **přímému doručování**
          - o které se starají přenosové protokoly linkové vrstvy (L2)
      - ve skutečnosti může přenos procházet přes několik „přestupních uzlů“
        - mostů/přepínačů (bridge/switch)
          - mezi kterými je třeba hledat optimální cestu



jde pouze o iluzi

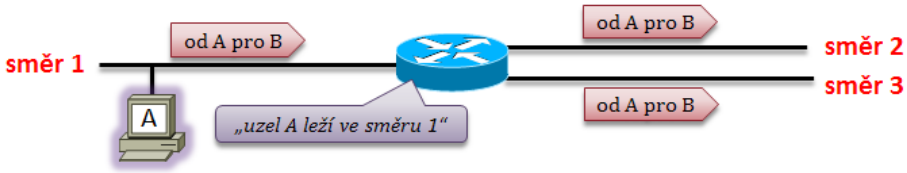
realita



- **ale:**
  - i na úrovni vrstvy linkové může být zapotřebí hledat cestu mezi mosty/přepínači

směrování na linkové vrstvě (L2)

- **nejde o klasické směrování, ale o hledání cesty (přes mosty/přepínače)**
  - k cílovému uzlu, který se nachází v téže síti, jako odesílající uzel
    - jde o nalezení posloupnosti mostů/směrovačů, přes které je nutné data předávat
- **v Ethernetu:**
  - používá se **metoda zpětného učení**
    - původně určená pro síťovou vrstvu (L3)
      - ale pro L3 málo vhodná
        - kvůli velké režii na „naučení se“ topologii ve velké soustavě sítí
          - trvalo by velmi dlouho, než by se mosty/přepínače naučily znát to, co potřebují
          - během doby učení by se mosty/směrovače chovaly velmi neefektivně (jako opakovače)
      - pro linkovou vrstvu (L2) vhodnější
        - potřeba „naučit se“ se týká jen uzlů v dané síti (nikoli v ostatních sítích)
          - učení je rychlé, neefektivnost během učení se není tak velká
    - výhoda: ethernetové mosty/přepínače mohou být plug&play
      - není nutné je konfigurovat, stačí je zapnout
        - samy se naučí vše, co k fungování potřebují znát
    - nevýhoda: v síti nesmí být cykly (brání zpětnému učení)



- **v Token Ring-u**
  - používá se **metoda source routing**