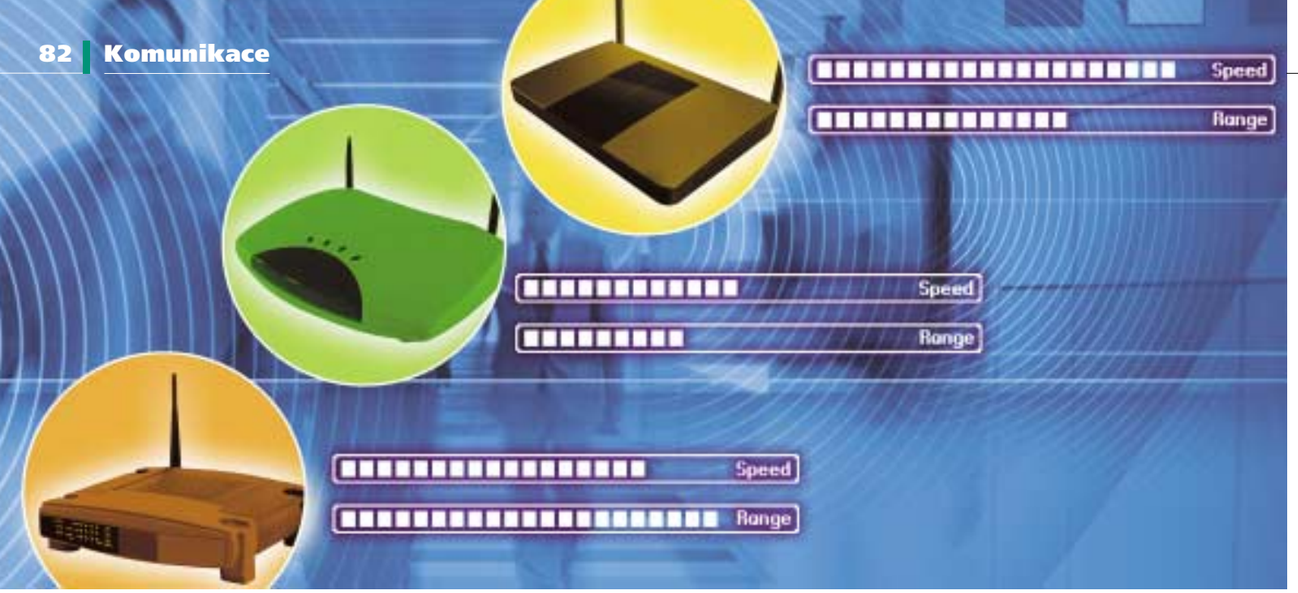


Báječný svět počítačových sítí

Část VIII. – Přenosové techniky v prostředí počítačových sítí (2)

JIŘÍ PETERKA

Při přenosech dat je třeba zajistit, aby příjemce správně rozpoznal a vyhodnotil to, co mu odesilatel vlastně posílá. To se zajišťuje jinak u synchronních přenosů, jinak u přenosů asynchronních či arytmických. A co je bitový proud (bitstream) a jak fungují isochronní a ne-isochronní přenosy?

Každý přenos digitálních dat je ve své podstatě přenosem jednotlivých bitů. Alespoň na nejnižší úrovni (na úrovni fyzické vrstvy) nás ještě nezajímá, zda jednotlivé bity nějak „patří k sobě" (třeba proto, že společně tvoří nějaký datový rámec či paket) a každý z nich přenášíme samostatně a nezávisle na ostatních bitech.

Přenos každého jednotlivého bitu ale vždy určitou dobu trvá, resp. má nějakou délku v čase. Pokud by tomu tak nebylo a přenos každého bitu by trval jen nekonečně krátký okamžik, mohli bychom dosahovat nekonečně velkých přenosových rychlostí.

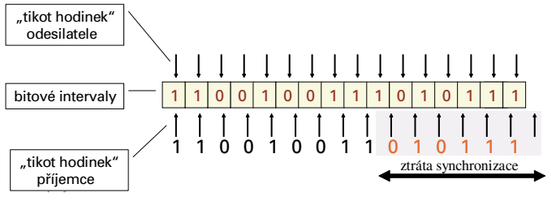
Místo toho musíme v praxi počítat s tím, že přenos každého bitu probíhá po určitý časový interval, kterému se ne náhodou říká „bitový interval". Během tohoto bitového intervalu se příjemce musí rozhodnout, zda vyhodnotí přijímaný bit jako 1 nebo 0. Můžeme si to představit tak, že někde uprostřed tohoto intervalu příjemce sejme stav přenášeného signálu a podle něj se rozhodne, zda jde o 1 či 0.

Potřeba synchronizace

Již na této jednoduché představě (vzorkování přijímaného signálu během bitového intervalu) si můžeme názorně ukázat jedno velké nebezpečí. Příjemce totiž musí správně odměřit začátek i konec každého bitového intervalu, aby se správně „trefil" se vzorkováním přenášeného signálu.

Asi není těžké si představit, co by se stalo, pokud by příjemce odměřil bitový interval nesprávně, v důsledku pomalejšího nebo naopak rychlejšího „tikotu" svých hodinek – strefil by se do jiného bitového intervalu a vyhodnocoval (vzorkoval) by stav signálu v době, kdy již reprezentuje nějaký jiný bit. Něčemu takovému se říká „ztráta synchronizace" a je to samozřejmě nežádoucí.

Co je naopak žádoucí, resp. pro korektní přenos dat nutné, je udržování synchronizace během přenosu všech relevantních datových bitů. Můžeme si to představit také tak, že obě strany (odesilatel i příjemce) mají u sebe vlastní hodinky, pomocí kterých odměřují jednotlivé bitové intervaly – a tyto hodinky musí tikat v dostatečném souběhu (tzv. **synchronně**), aby se navzájem příliš nerozešly a nedošlo k výše popisované nežádoucí situaci (ztrátě synchronizace).

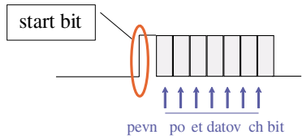


Představa ztráty synchronizace mezi příjemcem a odesilatelem.

Arytmický přenos

Požadavek na zajištění potřebné synchronizace nemusí nutně znamenat, že by pomyslné hodinky na straně odesilatele a příjemce musely tikat ve vzájemném souladu (synchronně) trvale, po libovolně dlouhou dobu. Požadovaného efektu lze dosáhnout i tak, že přenos dat bude probíhat po částech (po určitých skupinách bitů) a potřebná synchronizace bude udržována pouze po dobu přenosu těchto skupin bitů.

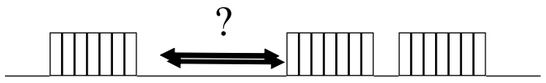
Znamená to tu výhodu, že pokud budou uvedené skupiny bitů dostatečně malé, nebudou na přesnost hodinek příjemce kladeny nijak extrémní nároky. Postačí, když se vhodně seřídí na začátku každé jednotlivé skupiny bitů a vydrží tikat v potřebném tempu alespoň po dobu přenosu této krátké skupiny. Pak se už hodinky příjemce mohou klidně rozejít (ztratit synchronizaci s hodinkami odesilatele), protože na začátku další skupiny bitů dojde k novému seřízení, a vše se opakuje.



Znak při arytmickém přenosu.

To už jsme ale popsali princip tzv. arytmického přenosu. Skupinkám bitů, které jsou tímto způsobem přenášeny, se v praxi říká **znaky** a mají vždy pevnou velikost. Dnes je to nejčastěji 8 datových bitů, ale v úvahu připadá i pevné nastavení na 7, 6 či třeba 5 bitů.

Kromě toho se na začátek každého znaku přidává jeden režijní bit, v rámci tzv. **start bitu**. Právě na něm se hodinky příjemce seřídí a začnou odměřovat bitové intervaly. Na konci znaku pak může (ale nemusí) následovat ještě tzv. stop bit pro zajištění určitého minimálního odstupu od dalšího znaku, pak také tzv. paritní bit, sloužící potřebám zabezpečení přenosu a umožňující detekovat případné chyby v přenosu. Ale k tomu se dostaneme příště.



Při arytmickém přenosu mohou být mezi znaky libovolné odstupy.

Zdůrazněme spíše jinou věc: mezi přenosem jednotlivých znaků mohou být libovolně dlouhé prodlevy. Nemusí tedy být dodržován stejný „rytmus" odesílání jednotlivých znaků (stejné odstupy mezi nimi). Právě tomuto aspektu vděčí tento způsob datového přenosu za své označení: **arytmický**.

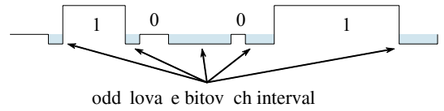
Arytmický, nebo asynchronní?

Po přečtení předchozích řádků vás možná napadne, zda je popis arytmického přenosu míněn vážně a zda nejde o nějaký omyl – když se v praxi popsanému způsobu přenosu se stejně velkými znaky, start bity atd. říká „asynchronní" a nikoli „arytmický".

Je zde skutečně drobný terminologický zádrhel: věcně správné je označení **arytmický přenos**, ale v praxi se mu neřekne jinak než **asynchronní přenos**. Je to tedy další příklad toho, kdy se v počítačové branži říká A, ale myslí se B.

Jiným podobným příkladem je použití zkratky RAM (z anglického Random Access Memory). Ačkoli správně označuje paměť s tzv. přímým přístupem (Random Access), běžně se zkratkou RAM myslí paměť pro čtení i zápis, která je protipólem k paměti ROM (Read Only Memory). Správně by ale paměť pro čtení i zápis měla být označována jako paměť RWM (Read/Write Memory).

Ovšem zpět k asynchronním a arytmickým přenosům: skutečně asynchronní přenos je takový, který zcela postrádá jakoukoli synchronizaci mezi příjemcem a odesilatelem (proto: **a**synchronní). Jak ale potom příjemce pozná, kdy začíná a kdy končí jednotlivé bitové intervaly? Pouze tak, že mu to odesilatel explicitně řekne. Tedy kromě dvou stavů přenášeného signálu, které reprezentují „užitečné" datové hodnoty 1 a 0, existuje ještě třetí možný stav, který indikuje právě začátek a konec bitového intervalu. Je k tomu tedy zapotřebí alespoň tzv. tříhodnotová logika, což je pro reálně využití problematické.



Představa (skutečně) asynchronního přenosu s využitím třetího stavu pro oddělení jednotlivých bitových intervalů.

Spíše zajímavostí je pak to, že jednotlivé bitové intervaly u (skutečně) asynchronního přenosu mohou být různě dlouhé, jak ostatně naznačuje i obrázek. Otázkou ale je, k čemu by to bylo dobré.

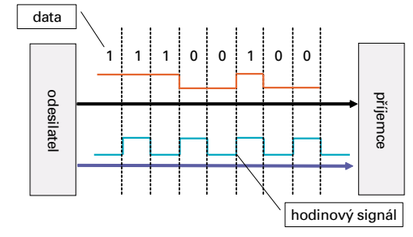
Synchronní přenos

Vedle asynchronního (správně: arytmického) přenosu je běžně používanou variantou také tzv. **synchronní přenos**, resp. „plně synchronní" přenos. Jak už jeho označení napovídá, je synchronizace mezi odesilatelem a příjemcem udržována dlouhodobě – buď trvale, nebo alespoň po dobu přenosu (libovolně velkého) datového bloku. Předností je pak vyšší efektivnost přenosu: jelikož se synchronizace neztrácí, ale udržuje trvale, není nutné oddělovat od sebe skupinky bitů (znaky). Místo toho je možné přenášet data „souvisle" po větších datových blocích (v zásadě libovolně velkých).

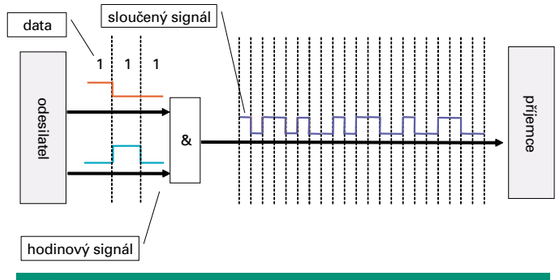
Jak ale dosáhnout toho, aby odesilatel a příjemce zůstali trvale synchronizováni, resp. aby se jejich hodinky nikdy nerozešly a stále „tikaly" v dostatečném vzájemném souladu? Tady už se nedá aplikovat podobný přístup jako u asynchronního přenosu (správně arytmického): seřídit (synchronizovat) na začátku datového bloku obě strany (analogie start bitu) a pak se spoléhat na to, že během přenosu celého bloku se hodinky příjemce nerozejdou více, než je přípustné. To jde u malých znaků, tvořených jen několika bity, ale už ne u datových bloků o stovkách bytů či ještě delších.

V případě (plně) synchronního přenosu je nutné udržovat synchronizaci průběžně. Tedy průběžně seřizovat hodinky příjemce. Ale jak?

Principiálně nejjednodušší by bylo přenášet tikot hodinek odesilatele až k příjemci, a to pomocí vhodného „hodinového" signálu (pravidelně se měnícího signálu, který odměřuje tikot hodinek odesilatele). Pak by na straně příjemce bylo vše jednoduché – stav dat by se vyhodnocoval v okamžicích, určených tímto hodinovým signálem.



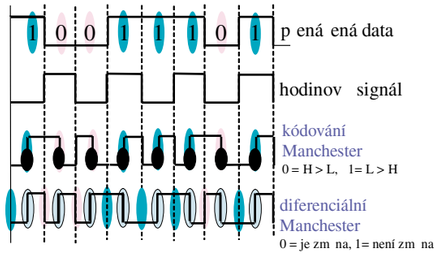
Představa využití samostatného (hodinového) signálu pro zajištění trvalé synchronizace.



Představa sloučení hodinového signálu a signálu nesoucího data do jediného signálu.

Ovšem samostatný hodinový signál pro zajištění plné synchronizace je poměrně velký luxus, který si většinou můžeme dovolit jen na krátkou vzdálenost. Ke svému přenosu totiž vyžaduje samostatný vodič (resp. pár vodičů), což je na delší vzdálenost příliš drahé. Proto se v praxi používá spíše jiné řešení, které z hlediska efektu vyjde vlastně nastejno, ale vystačí si jen s jedním vodičem. Spočívá v tom, že se použije pouze jeden vodič, po němž se přenáší oba signály (hodinový i „datový") současně, sloučené do jednoho výsledného signálu. Představu zachycuje obrázek.

Konkrétních způsobů, jak sloučit hodinový a datový signál do jednoho výsledného signálu, existuje více. Poněkud nesprávně se tomu říká „kódování". Dvě taková často používaná kódování ukazuje další obrázek. Jde o kódování Manchester a tzv. diferenciální Manchester.



▲ Kódování Manchester a diferenciální Manchester.

U kódování Manchester, které se používá například v původní desetimegabitové verzi Ethernetu, jsou oba signály sloučeny tak, že v každém bitovém intervalu dochází nejméně k jedné změně signálu. Právě tato změna (uprostřed bitového intervalu) pak nese užitečná data – jde-li o změnu z vysoké úrovně signálu do nízké (ve smyslu obrázku), jde o 0, zatímco hodnota 1 je reprezentována opačnou změnou (z nízké na vysokou úroveň). Současně tato změna slouží i potřebám synchronizace – při každé takové změně si příjemce může znovu seřídit své hodinky.

Aby však v každém bitovém intervalu mohlo dojít k takové změně signálu, která potřebným způsobem reprezentuje přenášená data, může být nutná ještě jedna další změna. Například když se přenáší více stejných bitů (více 0 nebo naopak více 1), pak je každý z těchto bitů reprezentován stejně orientovanou změnou. Aby však taková změna mohla v každém bitovém intervalu nastat, musí být vždy provedena ještě jedna opačná změna – viz předchozí obrázek.

Takže obecně musí na každý bitový interval (datový bit) připadnout dvě změny přenášeného signálu. To je docela plýtvání, které se nevyhýbá ani druhé variantě – diferenciálnímu kódování Manchester. Zde je hodnota 0 reprezentována změnou signálu (bez ohledu na pointaci této změny), hodnota 1 zase naopak absencí změny signálu – a to na začátku bitového intervalu.

Delší posloupnost hodnoty 1 by však mohla způsobit ztrátu synchronizace. Proto se u diferenciálního kódování Manchester v každém bitovém intervalu provádí ještě jedna změna (uprostřed bitového intervalu), která tentokrát již slouží pouze potřebám časování. Takže zatímco u „normálního" (nikoli diferenciálního) kódování Manchester slouží jedna hrana oběma účelům současně (a případná druhá hrana vlastně jen připravuje půdu pro novou změnu), u diferenciálního kódování Manchester má každý účel svou vlastní hranu.

Bit stuffing

Potřeba dvou změn signálu na každý datový bit, vynucená potřebami synchronizace, je skutečně velký luxus. Ve své podstatě znamená, že „spotřeba" (ve smyslu počtu změn, resp. modulační rychlosti) je dvojnásobná oproti „užitku" (přenosové rychlosti v bitech za sekundu). Něco takového je možné si dovolit tam, kde je přenosové kapacity dostatek, ale to není zdaleka všude.

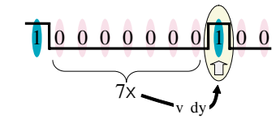
Snaha odstranit naznačené plýtvání je dobře patrná například u Ethernetu, jehož původní desetimegabitová verze používá kódování Manchester, ale jeho rychlejší verze (stomegabitový Ethernet, gigabitový Ethernet atd.) již používají efektivnější metody.

Všechny tyto efektivnější metody jsou přitom založeny na společném principu: místo toho, aby se „jedna změna navíc", nutná k zajištění synchronizace, přidávala ke každému jednotlivému datovému bitu (resp. k „datové" změně signálu), což vlastně představuje 100% režii, přidává se vždy až ke skupince několika datových bitů (resp. „datových" změn signálu). Pokud se například přidává ke čtyřem datovým bitům (změnám) jedna režijní změna pro potřeby zajištění synchronizace, už místo 4 užitečných změn přenáší změn 5, a to odpovídá o mnoho výhodnější režii ve výši 25 %. Takovéto kódování se pak označuje jako 4B/5B a používá se konkrétně u stomegabitového Ethernetu.

Pokud bychom v tomto trendu pokračovali co možná nejdéle, mohli bychom se s výší režie limitně přiblížit až k nule. V takovém ideálním stavu by byly všechny změny využity pro „užitečná" data, s tím, že příjemce by udržoval potřebnou synchronizaci právě a pouze podle těchto „datových" změn.

I v tomto ideálním stavu bychom ale stále museli myslet na to, že určitá posloupnost dat (v závislosti na použitém způsobu kódování, resp. reprezentování datových bitů prostřednictvím přenášeného signálu) by mohla způsobit, že příjemce nedostane včas žádnou hranu a jeho hodinky se „rozejdou" více, než je přípustné.

Řešení naštěstí není nijak komplikované a spočívá v umělém zařazení „bitu navíc" těsně předtím, než by hrozila ztráta synchronizace. Naznačuje to i další obrázek: pro případ, že 1 a 0 jsou reprezentovány vysokou, resp. nízkou úrovní signálu a hodinky příjemce nevydrží v synchronizaci déle než 7 bitových intervalů. V takovém případě se pak po každé posloupnosti sedmi bitů napevno zařadí do přenášeného toku dat jeden opačný bit.



▲ Představa bit stuffingu.

V angličtině se této technice říká **bit stuffing**, doslova „vkládání bitů".

Bitstream (bitový proud)

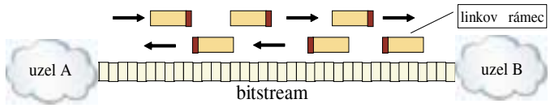
Když už víme, co je (plně) synchronní přenos, můžeme si jej poněkud zobecnit – do podoby jednoduché přenosové služby, realizované fyzickou vrstvou a označované jako tzv. **bitový proud** (anglicky **bitstream**). Zajišťuje dvoubodové spojení (tedy vede přímo mezi dvěma uzly), je obvykle obousměrný a chová se jako synchronně fungující roura: z jedné strany se do ní postupně vkládají jednotlivé bity a z druhé strany zase tyto bity ve stejném pořadí vystupují.

Hlavní výhodou takovéto přenosové služby, přenášející nijak nečleněný proudu bitů, je fakt, že nad ní lze realizovat v zásadě jakoukoli přenosovou službu vyšší úrovně. Tedy nejen přenos dat na principu přepojování paketů (či rámců), ale také přenos dat na principu přepojování okruhů. První možnost je vhodná pro datové služby (v praxi například pro přístup k internetu), zatímco druhá je zase výhodná pro multimediální služby (například pro přenos hlasu či obrazu), s potřebou garantované kvality a garantovaných parametrů (například přenosového zpoždění a jeho pravidelnosti).

Bitový proud (bitstream) je tedy jakýmsi „ideálním podložím", nad kterým lze budovat široké portfolio služeb – od negarantovaných datových služeb až po garantované služby, vhodné pro multimédia. Samotný bitstream se dá realizovat například na místních smyčkách (účastnických vedeních) pomocí technologií xDSL.

V ČR však podobný bitstream není dostupný. Český Telecom, který vlastní prakticky všechny místní smyčky, nabízí ostatním operátorům až přenosové služby vyšší úrovně, realizované na principu přepojování paketů a fungující negarantovaným způsobem. Jinými slovy: místo samotného „podloží" v podobě bitstreamu, nad kterým by alternativní operátoři mohli implementovat své vlastní přenosové služby (vhodné pro data i pro multimediální přenosy), jim Telecom povinně přidá (a nechá si zaplatit) své vlastní přenosové služby – fungující na principu přepojování paketů, konkrétně na bázi protokolu IP. Teprve tyto služby pak zprostředkovává alternativním operátorům na velkoobchodní bázi.

Rozdíl je v tom, že pokud by alternativní operátoři měli přístup k bitstreamu, mohli by sami určovat to, jaké přenosové služby nad ním vybudují. Pak by například mohli nabízet ADSL přípojky k internetu s takovými rychlostmi a stupni sdílení (agregace), jaké si zvolí sami. Místo toho jsou vázáni tím, co rozhodl a určil Telecom (např. z hlediska rychlostí a agregace) a co zabudoval do svých přenosových služeb nad bitstreamem.



▲ Představa bitového proudu (bitstreamu), nad kterým je realizován přenos paketů (rámců).

Isochronní a ne-isochronní přenos

Pro docenění toho, jak univerzální je bitový proud (bitstream), si ještě řekněme, co je tzv**. isochronní přenos**. Samotný přívlastek „isochronní" je možné přeložit jako „probíhající ve stejném čase" nebo alespoň „rovnoměrně v čase". Isochronní přenos je tedy takový, který doručuje data dostatečně rychle (s garantovaným maximálním zpožděním) a také pravidelně (s garantovaným maximálním rozptylem doby zpoždění). Naopak „ne-isochronní" je takový přenos, který toto nenabízí.

Jistě není těžké nahlédnout, že multimediálním službám (například živému hlasu a obrazu) svědčí isochronní přenosy, zatímco typickým datovým aplikacím (jako třeba elektronické poště, přenosu souborů, WWW atd.) plně postačují i neisochronní přenosy.

Ne-isochronní jsou přitom zejména přenosy na principu přepojování paketů (anglicky *packet switching*). Tedy například přenosy, realizované protokolem IP, z rodiny TCP/IP. To proto, že tento přenosový protokol negarantuje, jak dlouho se jednotlivé pakety „zdrží po cestě", zejména uvnitř přepojovacích uzlů (směrovačů). Všem totiž měří stejně, a tak se rychlost „zpracování" každého jednotlivého paketu odvíjí od momentálního souběhu s dalšími pakety, od jiných přenosů.

Naproti tomu isochronní jsou přenosy na principu tzv. přepojování okruhů (anglicky *circuit switching*). Je to dáno tím, že při přepojování paketů je každému přenosu vyhrazena určitá přenosová kapacita, o kterou již nemusí soutěžit s jinými přenosy. Díky tomu může být „zdržení" dat po cestě dostatečně malé a také dostatečně pravidelné (stejné).

Isochronní způsob fungování zachovává také tzv. časový multiplex, který jsme popisovali v minulém dílu tohoto seriálu (každému dílčímu přenosu pevně vyhrazuje určitou přenosovou kapacitu). Naproti tomu tzv. statistický multiplex již isochronní není, protože obecně negarantuje přenosovou kapacitu žádnému z dílčích přenosů (ale rozděluje ji podle momentální potřeby s tím, že se „nemusí dostat na všechny").

Pokud bychom chtěli nějakým způsobem dodatečně vylepšit přenosové služby na principu přepojování paketů, aby se co možná nejvíce blížily isochronnímu způsobu fungování, pak by určité možnosti existovaly. Obecně se tomu říká „zajištění kvality služeb" (QoS, Quality of Service), ale je to drahé a komplikované.

Mnohem jednodušší a efektivnější je vybudovat souběžně oba druhy služeb – garantované služby isochronního charakteru a negarantované služby ne-isochronního charakteru – nad takovým „podložím", které jejich souběh umožňuje. A takovým podložím je právě bitstream. Ten je svou podstatou sám isochronní, a tak lze jeho kapacitu rovnoměrně rozdělit na dvě části. Jednu můžeme využít pro isochronní přenosové služby (např. pro potřeby přenosu hlasu a obrazu) a druhou pro ne-isochronní služby (pro datové služby). Nebo ji lze využít celou pro ne zcela isochronní služby, ale s lepšími přenosovými parametry. Podstatné je, že možností je celá řada a nad bitstreamem není žádná z nich vyloučena.

5 0601/BAM ❑