

## Aula 18

- USB – Universal Serial Bus
- Características principais
- Topologia
- Codificação e transmissão de dados
- Gestão de energia
- Modelo de comunicação
- Tipos de transferências

José Luís Azevedo, Arnaldo Oliveira, Tomás Oliveira e Silva, Nuno Lau

# Introdução

- USB - Barramento standard desenvolvido por alguns dos principais fabricantes de PCs e indústrias de eletrónica e telecomunicações com vista a interligar dispositivos (tipicamente computadores a periféricos)
  - Compaq, DEC, IBM, Intel, Microsoft, NEC, and Northern Telecom
- Apareceu em 1996 como resposta às dificuldades de instalar periféricos num PC
- Antes do USB, ligar novos dispositivos a um PC envolvia acrescentar hardware específico e fazer a respetiva configuração, evitando conflitos (e.g. gama de endereços atribuídos, linhas de interrupção, etc.)
- Evolução
  - USB 1.0, Janeiro de 1996 (Low Speed, 1.5 Mbit/s)
  - USB 1.1, Agosto de 1998 (Full Speed, 12 Mbit/s)
  - USB 2.0, Abril de 2000 (High Speed, 480 Mbit/s)
  - USB 3.0, Novembro de 2008 (Super Speed, 5 Gbit/s)
  - USB 3.1, Julho de 2013 (Super Speed+, 10 Gbit/s)
- Todas as novas versões mantêm a compatibilidade com as versões anteriores

# Introdução

- Alguns dos objetivos delineados pelas empresas que conceberam o USB:
  - Os utilizadores não devem ter que configurar *switches* ou *jumpers* em placas ou dispositivos
  - Os utilizadores não devem ter que abrir caixas para instalar novos dispositivos de Input/Output
  - Deve haver somente um tipo de cabo para ligar todos os dispositivos
  - Os dispositivos de Input/Output devem poder obter energia do cabo
  - Devem poder ser ligados até 127 dispositivos a um único computador
  - O sistema deve suportar dispositivos de tempo real (áudio, vídeo,...)
  - Os dispositivos devem poder ser instalados com o computador em funcionamento
  - Não deve ser necessário reiniciar o computador após a instalação de um novo dispositivo
  - O novo barramento e seus dispositivos de Input/Output devem ter um baixo custo de produção

# Características principais

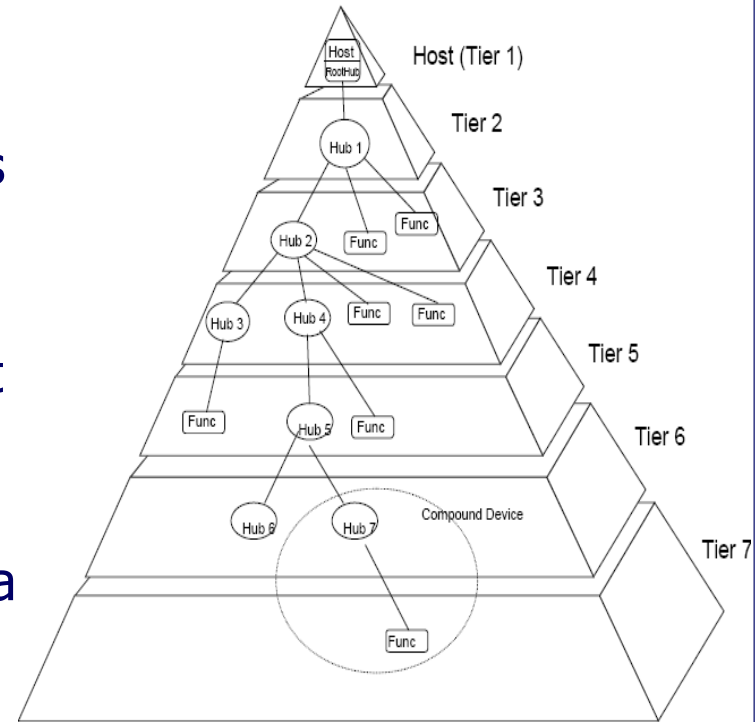
- "Hot plugging": Possibilidade de ligar e desligar fisicamente dispositivos sem necessidade de desligar a alimentação do sistema ou efetuar a sua reinicialização
- Configuração automática ("plug and play")
- Disponibiliza (dentro de certos limites) a alimentação para os dispositivos (+5V)
- Três taxas de transmissão (na versão 2.0)
  - **low-speed, 1.5 Mbit/s** – para comunicação com dispositivos de interação com o utilizador (rato, teclado, joystick, etc.)
  - **full-speed, 12 Mbit/s** – para comunicação com dispositivos que não requeiram taxas elevadas (e.g. impressoras, áudio, ...)
  - **high-speed, 480 Mbit/s** – para comunicação com dispositivos de alto débito (e.g. vídeo, discos, ...)
- Comunicação **half-duplex** (USB 1.0 e 2.0)

# Características principais

- O USB 3.x mantém compatibilidade para trás e acrescenta as taxas de transmissão
  - USB 3.0: **Super-speed mode – 5 Gbit/s**
  - USB 3.1: **Super-speed plus mode – 10 Gbit/s**
- **Comunicação "full-duplex"** no USB 3.x quando são usados os "Super-speed" modes
- Os dados são transmitidos em modo diferencial em par entrançado (nível lógico na receção é discriminado pela diferença de tensão entre os dois sinais)
- O USB usa **comunicação síncrona orientada ao bit**. Um conjunto consecutivo de bits é organizado numa mensagem (packet)
- Os bits são transmitidos continuamente com um ritmo imposto por um relógio que segue para o recetor embebido nos dados (**relógio codificado** - uma das tarefas do recetor é a recuperação desse relógio)

# Topologia

- **Ligação física em árvore** (tiered star), partindo de um sistema central (host controller+root hub) para vários dispositivos ("functions", na "terminologia USB")
- **Máximo de 7 níveis** (incluindo "root hub") e de 127 dispositivos
- Um "function" pode atuar como extremo ou disponibilizar acessos para outros dispositivos operando, nesse caso, como **hub**
- Hub (concentrador): permite que vários dispositivos possam partilhar uma única porta USB
  - Uma única ligação a montante, várias a jusante
  - Permite o aumento da disponibilidade de corrente (alimentação) para os dispositivos



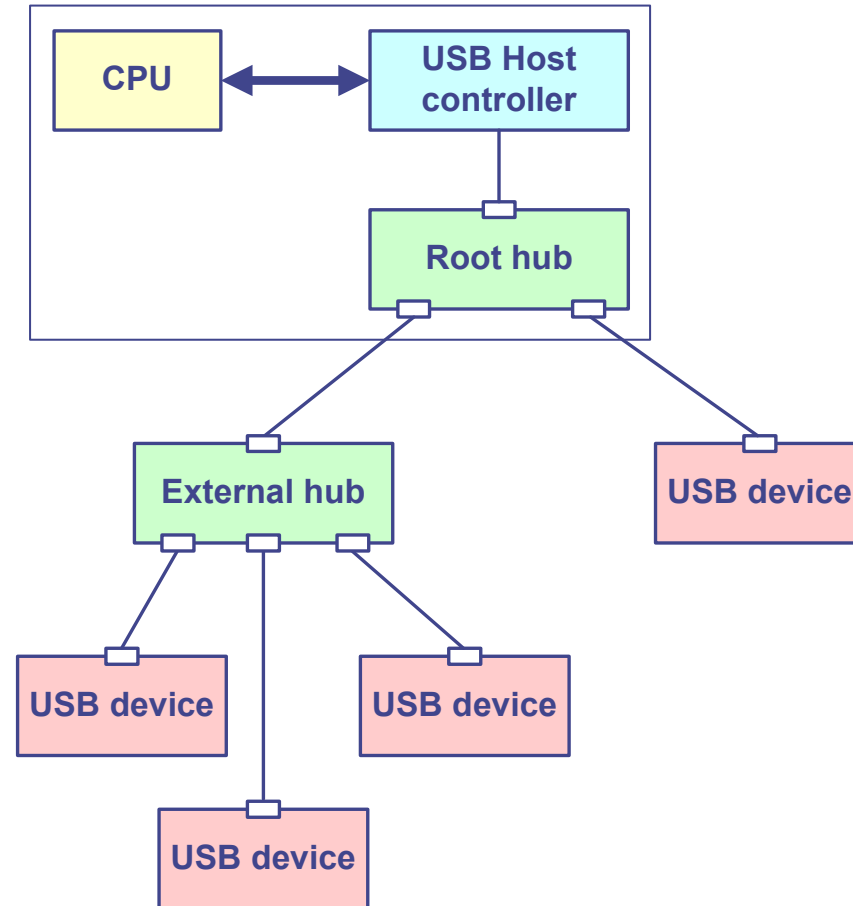
# Topologia

- **Arquitetura "master-slave"**

(i.e., cada dispositivo apenas envia dados após permissão explícita do "host controller")

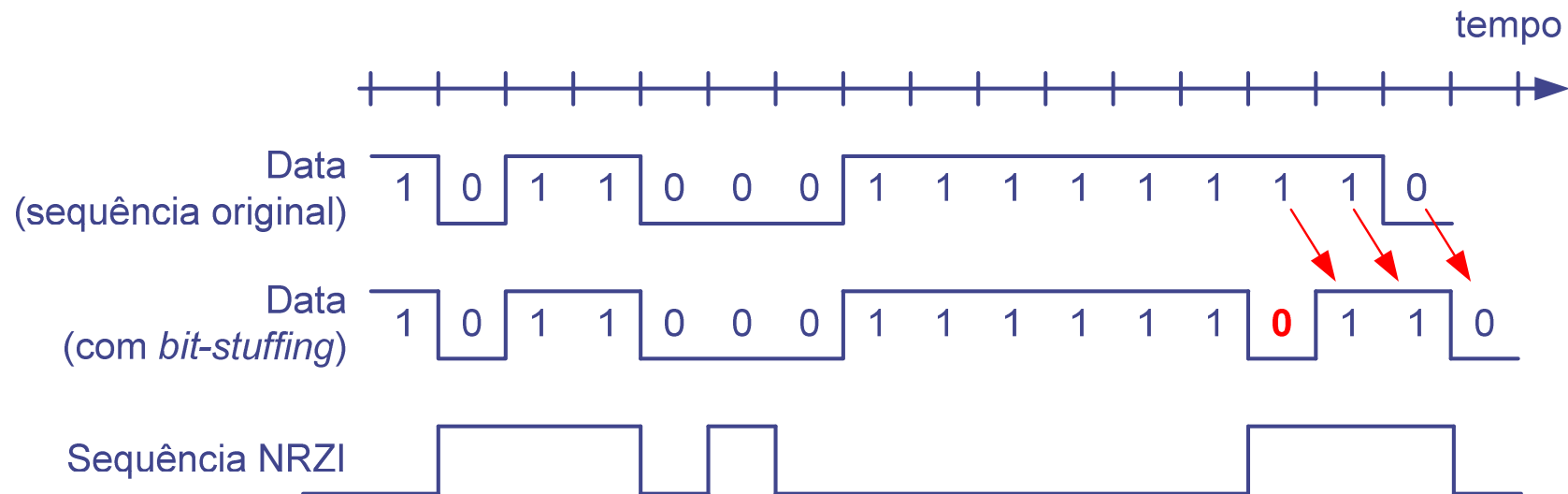
- Não há comunicação entre dois dispositivos USB. Cada dispositivo comunica apenas com o "host controller"

- O controlador USB é encarado, na arquitetura do sistema, como qualquer outro periférico
  - O CPU só comunica com o "host controller"



# Codificação e transmissão de dados

- Codificação NRZI (non-return to zero, inverted)
- Sinal codificado em NRZI não altera a polaridade para "1"s lógicos, invertendo-a para "0"s lógicos
- **"Bit-stuffing"** para mais que 6 "1"s consecutivos
- É inserido um "0" após 6 "1"s consecutivos (na sequência original) de modo a forçar uma transição na sequência transmitida que permita ao recetor manter o relógio sincronizado





# USB pinout

Pin	Function (host)	Function (device)
1	V <sub>BUS</sub>	V <sub>BUS</sub>
2	D <sup>-</sup>	D <sup>-</sup>
3	D <sup>+</sup>	D <sup>+</sup>
4	Ground	Ground

USB 1.0/2.0 conector Pinout

USB 3.x conector Pinout

USB 1.0/2.0

Pin	Color	Signal name ("A" Connector)	Signal name ("B" Connector)	Description
Shell	N/A	Shield		Metal housing
1	Red	VBUS		Power
2	White	D-		USB 2.0 differential pair
3	Green	D+		
4	Black	GND		Ground for power return
5	Blue	StdA_SSRX-	StdB_SSTX-	Superspeed transmitter differential pair
6	Yellow	StdA_SSRX+	StdB_SSTX+	
7	N/A	GND_DRAIN		Ground for signal return
8	Purple	StdA_SSTX-	StdB_SSRX-	Superspeed receiver differential pair
9	Orange	StdA_SSTX+	StdB_SSRX+	

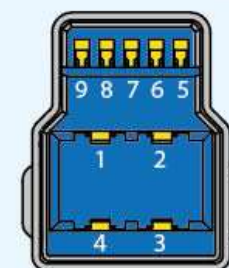
USB 3.x

# USB 3.x – conetores tipo A e tipo B

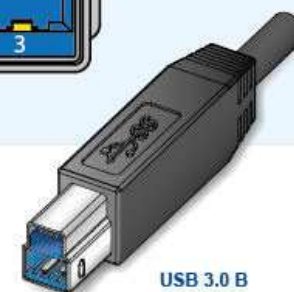
Comprimento dos cabos limitado a 5 m

## USB 3.0 type B pinout, cable Assembly

USB type B Pinout, Cable Assembly		
Pin	Signal Name	Description
1	VBUS	Red
2	D-	White
3	D+	Green
4	GND	Black
5	StdA_SSTX-	Blue
6	StdA_SSTX+	Yellow
7	GND_DRAIN	GROUND
8	StdA_SSRX-	Purple
9	StdA_SSRX+	Orange
Shell	Shield	Connector Shell



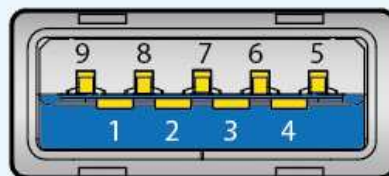
USB 3.0 type B Connector



USB 3.0 B

## USB 3.0 type A pinout, cable Assembly

USB type A Pinout, Cable Assembly		
Pin	Signal Name	Description
1	VBUS	Red
2	D-	White
3	D+	Green
4	GND	Black
5	StdA_SSRX-	Blue
6	StdA_SSRX+	Yellow
7	GND_DRAIN	GROUND
8	StdA_SSTX-	Purple
9	StdA_SSTX+	Orange
Shell	Shield	Connector Shell



USB 3.0 type A Connector



USB 3.0 A

- O conetor tipo A liga ao host
- O conetor tipo B liga ao device

# Gestão de energia

- Dispositivos (incluindo *hubs*) podem ter fonte de alimentação própria ou alimentar-se do barramento
- USB 2.0 – 1 unidade de carga (UC): 100 mA
  - Um *hub* alimentado pelo barramento pode (depois de configurado) fornecer em cada porta um máximo de 1 UC (i.e. 100 mA). A corrente máxima fornecida pelo *hub* está limitada a 5 UC
  - Um *hub* auto-alimentado pode fornecer em cada porta um máximo de 5 UC (i.e. 500 mA)
- USB 3.x – 1 unidade de carga (UC): 150 mA
  - *Hub* alimentado pelo barramento pode fornecer em cada porta 1 UC. A corrente máxima fornecida pelo *hub* está limitada a 6 UC (900mA)
  - Um *hub* auto-alimentado pode fornecer em cada porta 6 UC

# Gestão de energia

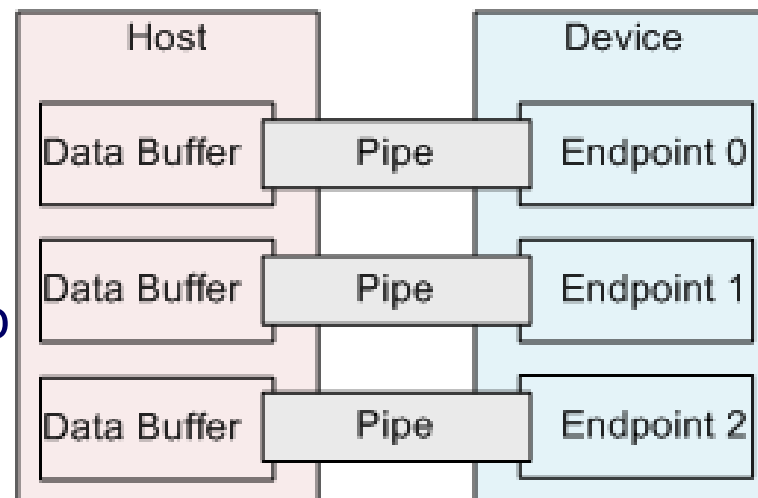
- Após conexão os "devices" são interrogados pelo "host controller" acerca das suas necessidades energéticas
  - Situações de sobrecarga são notificadas e *hubs/devices* podem ser desligados automaticamente
  - Durante a fase inicial os *devices* têm de operar em modo de baixo consumo (low power, max. 1 unidade de carga, passando para o modo normal de operação apenas após autorização explícita do "Host controller")
- A disponibilização de energia entre um *hub* e um dispositivo é também utilizada para a deteção da ligação ou da retirada do dispositivo o que permite implementar o modo de operação "plug-and-play"

# Modelo de comunicação USB

- O "host controller" pode controlar até 127 dispositivos, cada um com um identificador único (entre 1 e 127)
- O identificador 0 é reservado para permitir a ligação de um novo dispositivo; quando um novo dispositivo se liga, o "host controller" atribui-lhe um novo identificador e liberta o 0
- Não há comunicação entre dois dispositivos USB: cada dispositivo USB comunica apenas com o "host controller" através de canais virtuais, designados por "**pipes**" (condutas)
- Cada dispositivo pode incluir várias funcionalidades a que correspondem terminais lógicos (designados por "**endpoints**")
- Um *pipe* estabelece a comunicação entre um *endpoint* e o "host controller"; os *endpoints* estão numerados de 0 a 15 (a cada número correspondem dois *endpoints*, um em cada direção)
- Os *endpoints* são unidirecionais (ou são sinks ou sources ), sendo denominados
  - OUT (sink): do "host controller" para o dispositivo
  - IN (source): do dispositivo para o "host controller"

# Modelo de comunicação USB

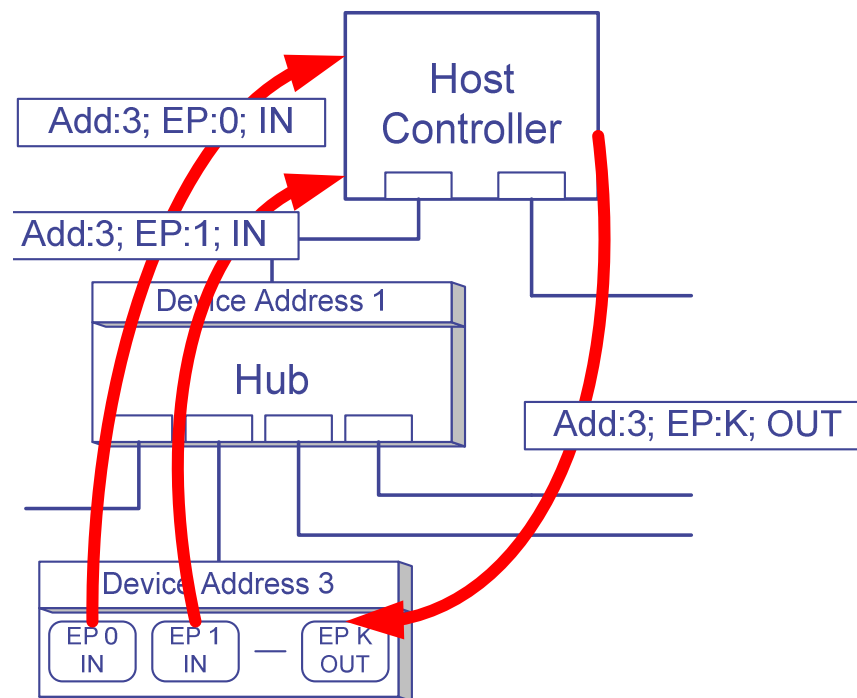
- Todos os dispositivos implementam obrigatoriamente os dois *endpoints* com o número 0
- O **"default control pipe"** está associado ao *endpoint 0*, é usado para efeitos de configuração e verificação de estado



- O "default control pipe" é obrigatório em todos os dispositivos USB – é o canal de comunicação bidirecional básico que se estabelece quando o dispositivo é ligado
- Os restantes canais de comunicação (pipes) são estabelecidos quando o dispositivo é ligado ao barramento, numa fase designada por **enumeração**
- **Cada endpoint inclui uma zona de memória** suficiente para guardar o maior pacote que pode circular no *pipe* a ele associado (configurado na fase de enumeração)

# Modelo de comunicação – endereçamento

- Endereço do dispositivo é atribuído pelo sistema aquando da conexão, sendo único no barramento
- Número do *endpoint*: cada *endpoint* tem um número pré-configurado, único para cada device
- Direcção: IN/OUT



A combinação  
"Endereço de device : Número de Endpoint : Direcção"  
identifica univocamente cada *endpoint* no bus

# Tipos de transferências

- O USB suporta 4 tipos de transferências de dados
- **Não periódicas** – surgem sem qualquer período definido, sob iniciativa do *host-controller*, sem garantia de taxa de transmissão ou de latência:
  - **Control** Transfers
  - **Bulk** Transfers
- **Periódicas** – Na fase de enumeração é definido o período entre envios consecutivos de informação, garantindo-se um limite máximo para a **latência** (atraso máximo de transmissão pelo USB):
  - **Isochronous** Transfers
  - **Interrupt** Transfers
- Cada *pipe* suporta apenas um destes tipos de transferência



# Transferências de controlo

- São utilizadas pelo *host-controller* para o envio de mensagens de **comando e configuração** ou **consulta de estado** de um dispositivo
- Todos os dispositivos USB têm de implementar o "default control pipe" associado ao *endpoint 0*, destinado a tarefas de configuração, controlo e verificação de estado
- As transferências de controlo suportam a comunicação com o *endpoint 0* do dispositivo (IN e OUT). Podem ser criados *endpoints* adicionais deste tipo para satisfazer necessidades específicas dos dispositivos
- Em princípio, a informação a transmitir / receber é de pequena dimensão

# Transferências Bulk

- Transferência de volumes de dados elevados, com instantes de transmissão irregulares
- Permitem a transmissão de dados com garantias de entrega, usando mecanismos de retransmissão em caso de detecção de uma situação de erro (CRC – Cyclic Redundancy Check)
- Não são dadas garantias de largura de banda (quantidade de dados/unidade de tempo) ou de latência, sendo este tráfego tratado em regime de "best effort"
- As transferências deste tipo utilizam a parte da trama que se encontra disponível após a transmissão dos pacotes relativos aos outros tipos de transferências
- Exemplos de dispositivos que usam este tipo de transferências: discos, pen-drives, scanners, impressoras

# Transferências de Interrupção

- Não há sinais físicos dos periféricos para o controlador de interrupções do CPU
- O ***host-controller*** **interroga periodicamente** os dispositivos (registados com este modo de transferência) para saber se algum tem informação para transferir (é um *polling* feito pelo *host-controller*)
- A regularidade com que o *polling* é feito depende do valor da latência negociada na fase de enumeração no arranque do periférico (o sucesso da negociação depende da existência de recursos suficientes para satisfazer os requisitos)
- O período de *polling* pode ser configurado
- No caso em que ocorre um erro de comunicação, a retransmissão é feita no período de *polling* seguinte
- São transmitidas baixas quantidades de informação (até 64 bytes)
- Tipo de transferência tipicamente usado em dispositivos de interação com o utilizador: ratos, teclados, joysticks, etc.

# Transferências Isócronas

- Permitem a transferência de informação com uma **largura de banda garantida** (quantidade de dados / unidade de tempo) e **latência limitada**. Ou seja, é assegurado um ritmo de transmissão sustentado
- Estes parâmetros são negociados durante o processo de arranque do periférico, na fase de enumeração (o sucesso da negociação depende da existência de recursos suficientes para satisfazer os requisitos)
- Os eventuais erros de transmissão não são corrigidos, isto é, não há retransmissão de informação (não há garantias de entrega)
- É, das quatro, o único tipo de transferência que **não faz controle de erros**
- Este tipo de transferência é usado essencialmente em aplicações de áudio e vídeo; i.e., aplicações onde garantir o ritmo de transmissão é mais importante que garantir que todos os dados chegam ao destino

# Enumeração

- No arranque do sistema o *host controller* acede ao *root hub* para:
  - Descobrir se há dispositivos a ele ligados
  - Identificar os dispositivos ligados aos diferentes *hubs* descobertos no sistema
- Passa depois ao procedimento de enumeração de cada um dos dispositivos encontrados: troca de informação inicial, que acontece automaticamente quando o sistema arranca e sempre que um dispositivo é ligado
- A enumeração permite ao *host controller* reconhecer e inicializar um dispositivo. Inclui:
  - Atribuição de um endereço (único)
  - Leitura de informações (e.g. Class, Vendor)
  - Alocação de recursos
  - Atribuição e carregamento de *Device Driver*
  - ...
- É uma tarefa permanente pois podem ligar-se/desligar-se dinamicamente dispositivos a um barramento USB