Universal Serial Bus (USB)

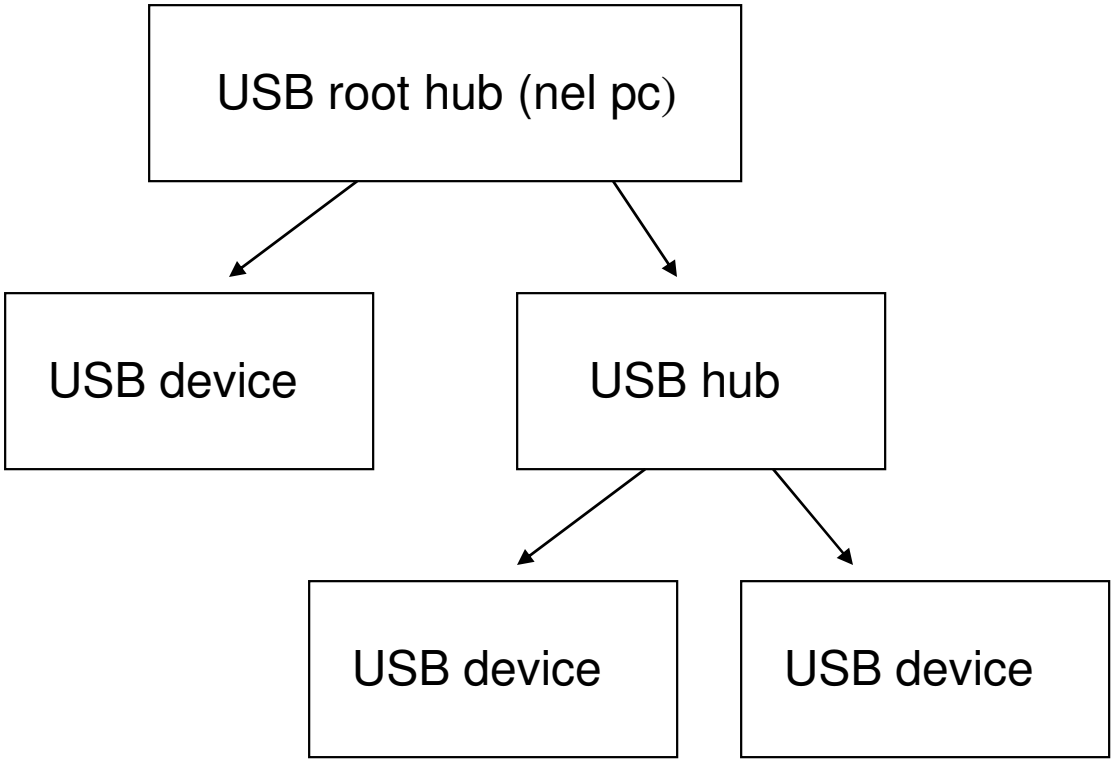
- Standard per la connessione di periferiche al personal computer (proposto verso la metà degli anni ‘90 da un pool di società tra cui Intel, IBM, Microsoft,..)

- Obiettivi principali:
  - connessione al pc dall’esterno del case
  - unico tipo di cavo, **in grado di distribuire anche l’alimentazione**
  - connessione di fino a 127 dispositivi ad un unico computer
  - supporto per dispositivi real-time (audio, video)

- Banda passante di 1.5 MB/s -*low speed* (12 Mb/s – detta *full speed, 480 Mb/s high speed*) Cavo a 4 fili: D+, D- per il segnale (differenziale), VBUS (alimentazione a +5 V), ground.
- Encoding: NRZI (Non Return to Zero Invert):
  - 1: conserva il valore logico precedente
  - 0: commuta rispetto al valore logico precedente
- Clock estratto dalla linea dati; per evitare perdita di sincronismo, è inserito uno 0 ogni sei 1 consecutivi (bit stuffing)

Struttura USB

- Struttura ad albero:



- Un USB device è un dispositivo in grado di connettersi al bus USB
- Un USB hub è un ripetitore con un ingresso e N (tipicamente 4, 7, 8) uscite; a ciascuna delle uscite può essere connesso un USB device o un altro USB hub

USB

- Ciascun device ha un identificatore $\in$ [1, 127]; a un device appena connesso è associato di default l'identificatore 0, in modo che possa essere indirizzabile dalla root che gli invia il suo identificatore definitivo (non c'è ambiguità se non sono connessi contemporaneamente più dispositivi)

- Da un punto di vista logico il bus USB è un insieme di bit pipe (pipe di larghezza 1 bit) tra la root e i device. NB: Nessuna connessione tra i device!, tutto centralizzato attraverso la root.

- In realtà, su ciascun device possono essere presenti fino a 16 endpoint (sorgenti o destinazioni della connessione). Endpoint 0: sempre disponibile a default all'atto della connessione.

- Root: ogni 1.00 $\pm$ 0.0005 ms si ha un nuovo frame. Il frame può contenere più trasferimenti. Ogni trasferimento è indirizzato a un device.

- Il frame è composto di pacchetti, che possono andare da root a device o viceversa.

USB transfer types

- 4 tipi di **trasferimenti**:
  - controllo
  - isocroni
  - bulk
  - interrupt

- controllo: per configurare i dispositivi, inviare comandi e richiedere lo stato: se fallisce, viene reinviato (non è detto nello stesso frame).

- isocroni: per dispositivi real-time quali microfoni e telecamere; hanno un ritardo massimo altamente predicibile e non prevedono ritrasmissione in caso di errori

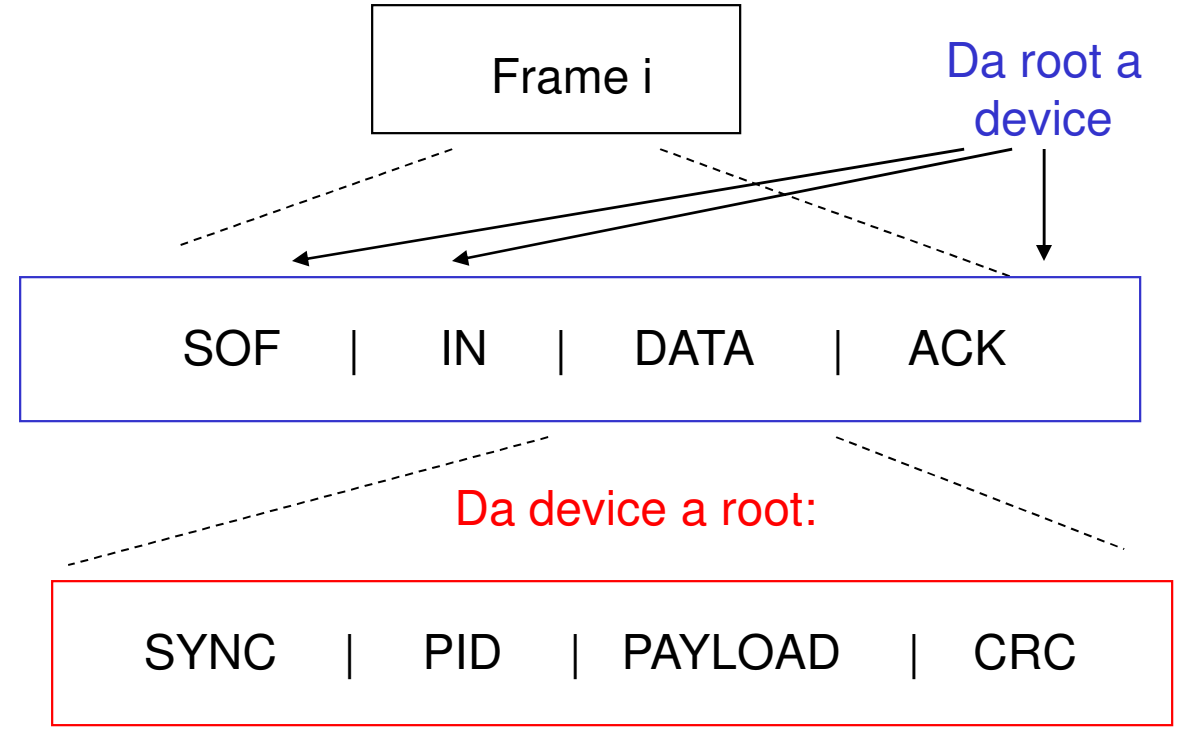
- bulk: per dati non real-time quali file per le stampanti

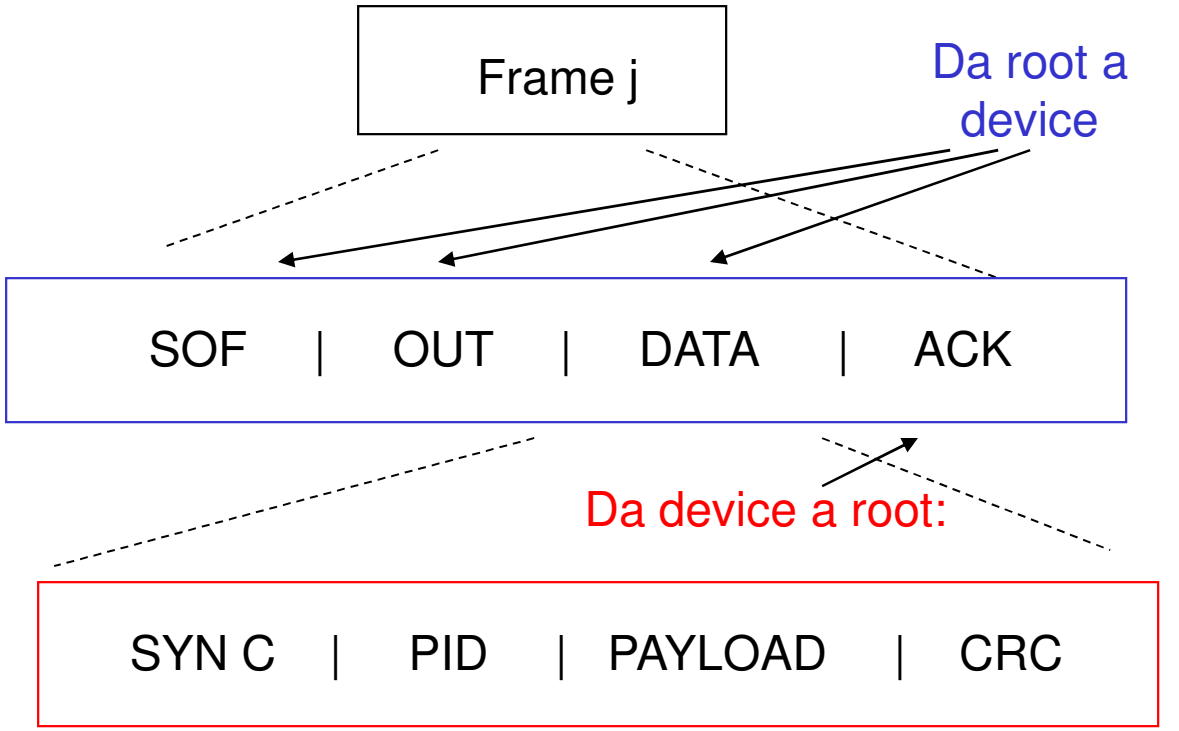
- interrupt: per dati con feedback di utente, tipo digitazione da tastiera o coordinate del mouse; sono in realtà inviati a polling perché non esistono linee fisiche di interrupt nel bus

USB packet

- I trasferimenti sono costituiti da pacchetti (packets); si hanno 4 tipi di **pacchetti**:
  - token
  - data
  - handshake
  - special
- ogni pacchetto è formato da campi:
  - SYNC: sincronizzazione
  - PID: tipo del pacchetto (SOF, IN, OUT, DATA...)
  - parte variabile (DATA per esempio)
  - CRC: Cyclic Redundancy Code
- **token**: dalla root al device per controllo
  - SOF: Start of Frame ("clock del frame"); se non c'è nulla da fare al momento, questo è l'unico pacchetto del frame; la parte variabile è il frame number
  - IN: chiede dati al dispositivo (lettura) e OUT: avvisa il dispositivo che stanno per arrivargli dati (scrittura); la parte variabile è device identifier + endpoint (indirizzamento della sorgente/ricevitore)
  - SETUP: per configurazione
- **data**: la parte variabile (PAYLOAD) sono i dati veri e propri (fino a 1023 byte per trasferimenti tipo isocrono e 64 per gli altri)
- **handshake**: ACK, NACK, STALL ("wait, i'm busy")

Esempi di trasferimenti





USB bandwidth

- Non è possibile raggiungere la banda passante limite del bus di 1.5 MB/s per le informazioni, a causa dell'overhead del protocollo: pacchetti token, ACK e NACK nei casi non isocroni, SYNC, PID, CRC nei pacchetti dati, ...

- maggiore è la lunghezza del payload nel pacchetto dati, migliore è lo sfruttamento della banda passante (che dipende ovviamente dall'overhead introdotto dal protocollo)

- In una situazione tipica è possibile eseguire circa 13-20 trasferimenti con payload da 64 byte per frame, per una banda utile di circa 0.8-1.3 MB/s; per payload più piccoli, questa si riduce a poche decine di KB/s.