I SOCKET IN C#

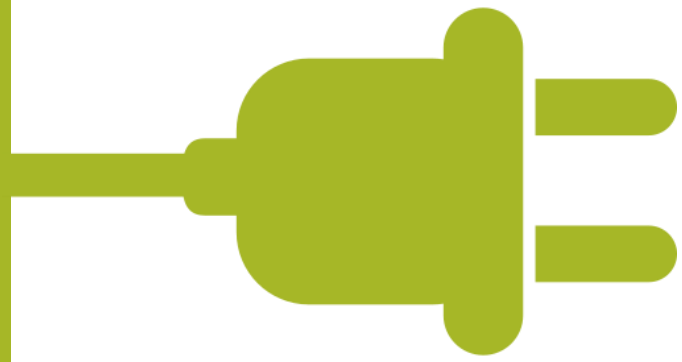
By Nicolò Coltro

Cosa sono?

Socket = Prese…

Utilizziamo i socket (implementati tramite C#), per instaurare una **connessione** tra due applicazioni (TCP L7), per la **trasmissione** di **informazioni**.

Ai due capi, troviamo un **Client** ed un **Server**.



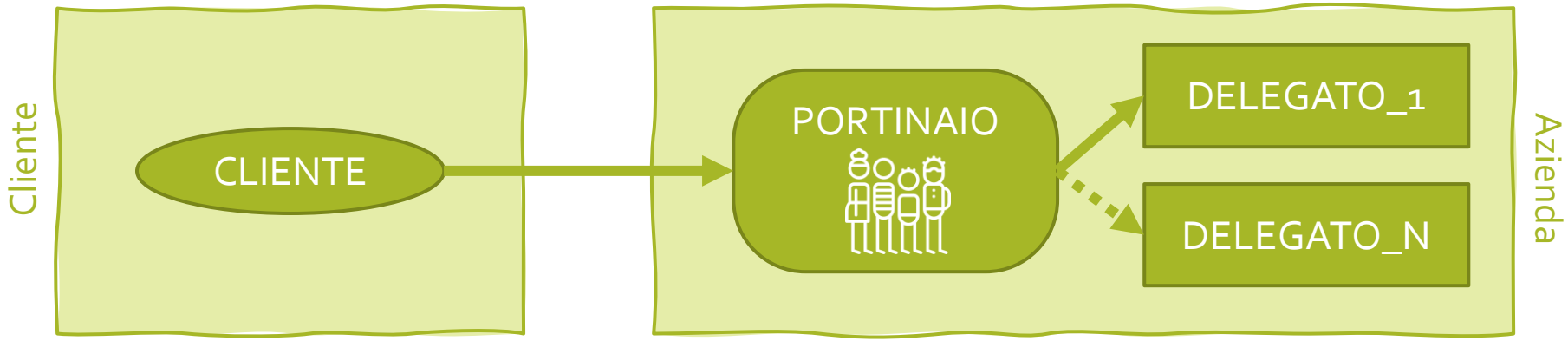
Come li identifichiamo?

IP + Numero di porta (evitare quelle well-known)…

Flow di gestione

Caso applicato «Cliente visita un'azienda dotata di Portinaio».

Come è possibile immaginare, il flow di un Cliente in ingresso viene gestito così:



Server

Composto dai seguenti componenti:

- **TcpListener**, assimilabile al *Portinaio* dell'azienda
- **TcpClient**, assimilabile ad un *Delegato* dell'azienda

Client

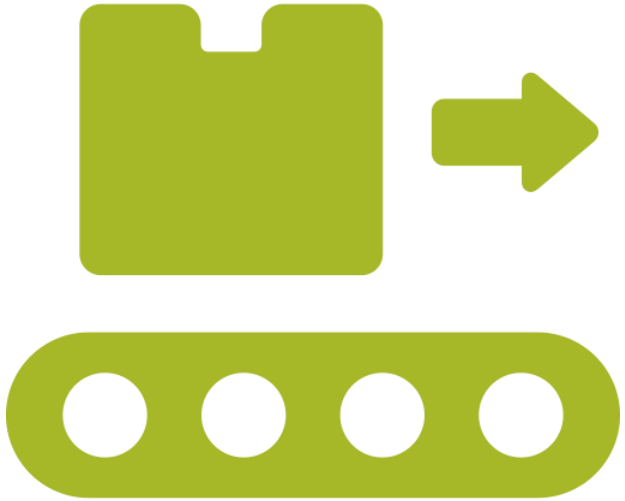
Composto dai seguenti componenti:

- **TcpClient**, assimilabile ad un *Cliente* in azienda

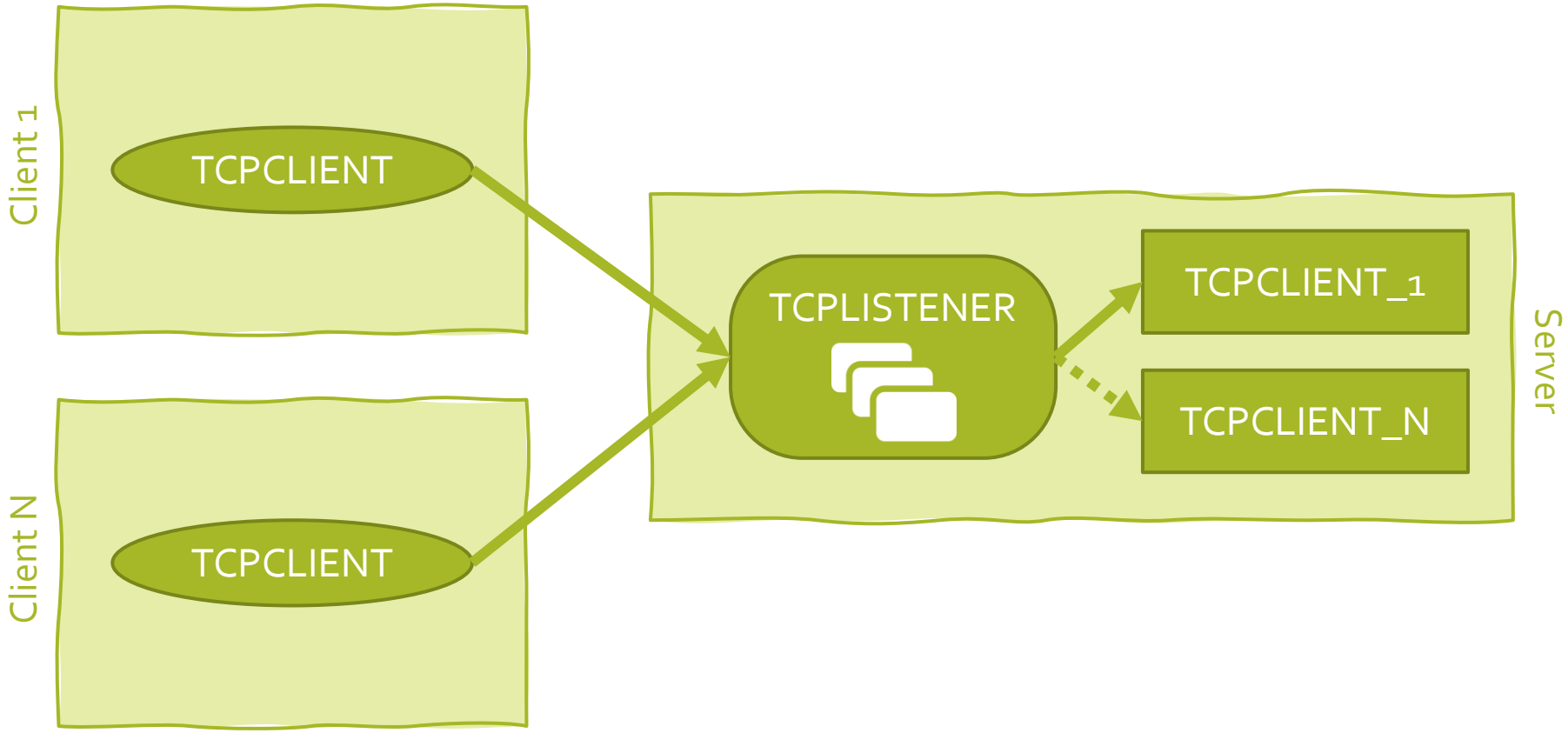
Ulteriori componenti necessari

Sia sul Client che sul Server, sono necessari:

- NetworkStream, **canale** di **comunicazione** messo a disposizione dal **Socket**
- Byte[dimensione], **array** di **Byte** che usiamo come «**scatola**» per i messaggi



Flow di gestione



Schema funzionale

1. Client **instaura** una connessione con il **Portinaio** del Server, che gestisce una **coda**
2. Al **Client** viene assegnato (=istanziato on-the-go) un **Delegato** del Server
3. Avviene lo **scambio** di **dati** tra C&D, avvalendosi di un **canale** di **comunicazione** (=stream) e **pacchetti** di messaggi (=payload salvati in un buffer grande N)
4. Una volta che il messaggio è stato trasmesso nella sua interezza e tutte le operazioni sono state compiute, **chiudo** il **canale di comunicazione** sia su Delegato che su Client
5. **Chiudo** la **connessione** tra Delegato e Client
6. Il **portinaio** gestisce un **altro Client**
7. E così via, **all'infinito** (o finché il processo del Server non è terminato)