

L'interfaccia di programmazione socket

Mario BALDI

<http://staff.polito.it/mario.baldi>

Silvano GAI

[sgai\[at\]cisco.com](mailto:sgai[at]cisco.com)

Fulvio RISSO

[fulvio.risso\[at\]polito.it](mailto:fulvio.risso[at]polito.it)



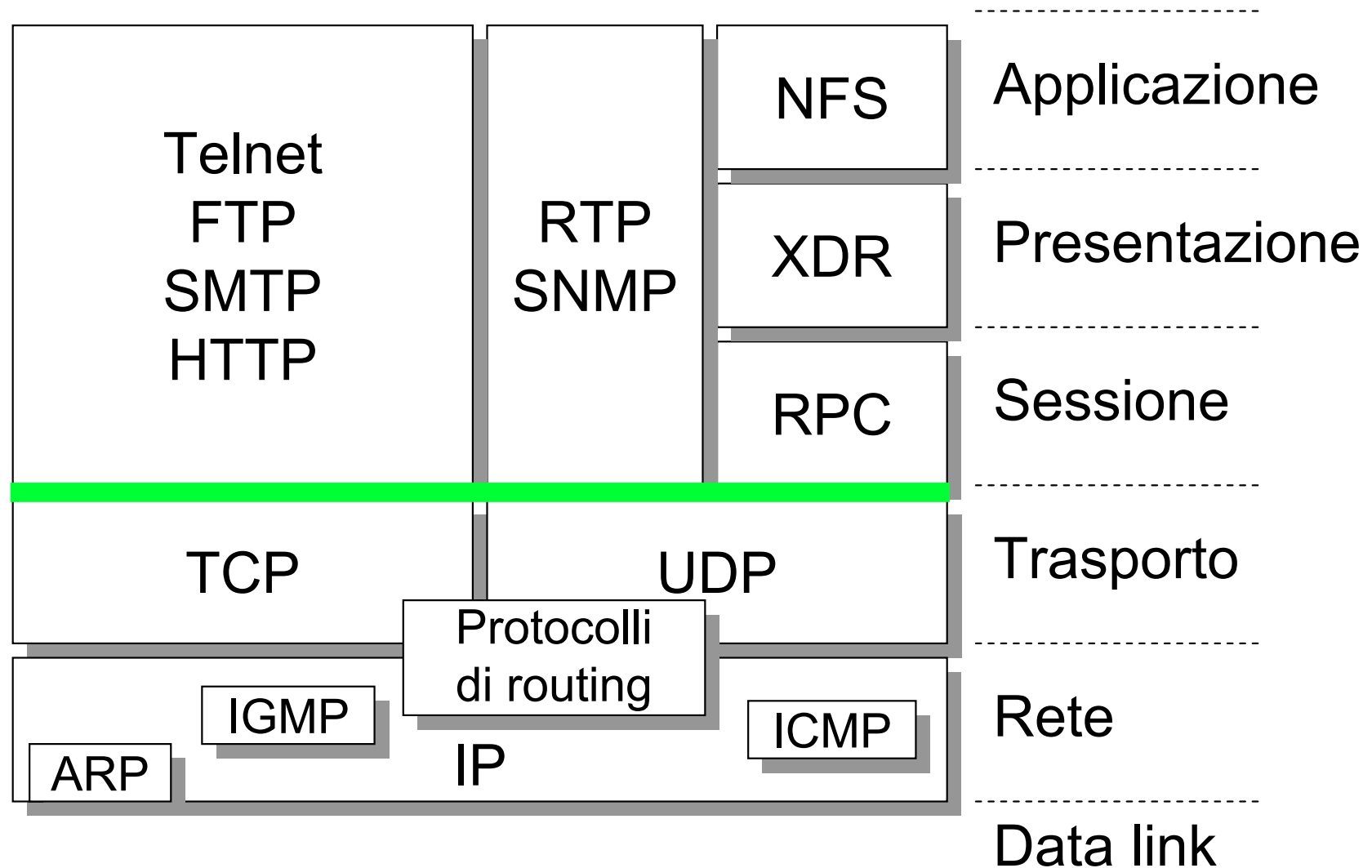
Nota di Copyright

Questo insieme di trasparenze (detto nel seguito slide) è protetto dalle leggi sul copyright e dalle disposizioni dei trattati internazionali. Il titolo ed i copyright relativi alle slide (ivi inclusi, ma non limitatamente, ogni immagine, fotografia, animazione, video, audio, musica e testo) sono di proprietà degli autori indicati a pag. 1.

Le slide possono essere riprodotte ed utilizzate liberamente dagli istituti di ricerca, scolastici ed universitari per scopi istituzionali, non a fine di lucro. In tal caso non è richiesta alcuna autorizzazione.

Ogni altra utilizzo o riproduzione (ivi incluse, ma non limitatamente, le riproduzioni su supporti magnetici, su reti di calcolatori e stampate) in toto o in parte è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte degli autori.

L'interfaccia Socket





I socket

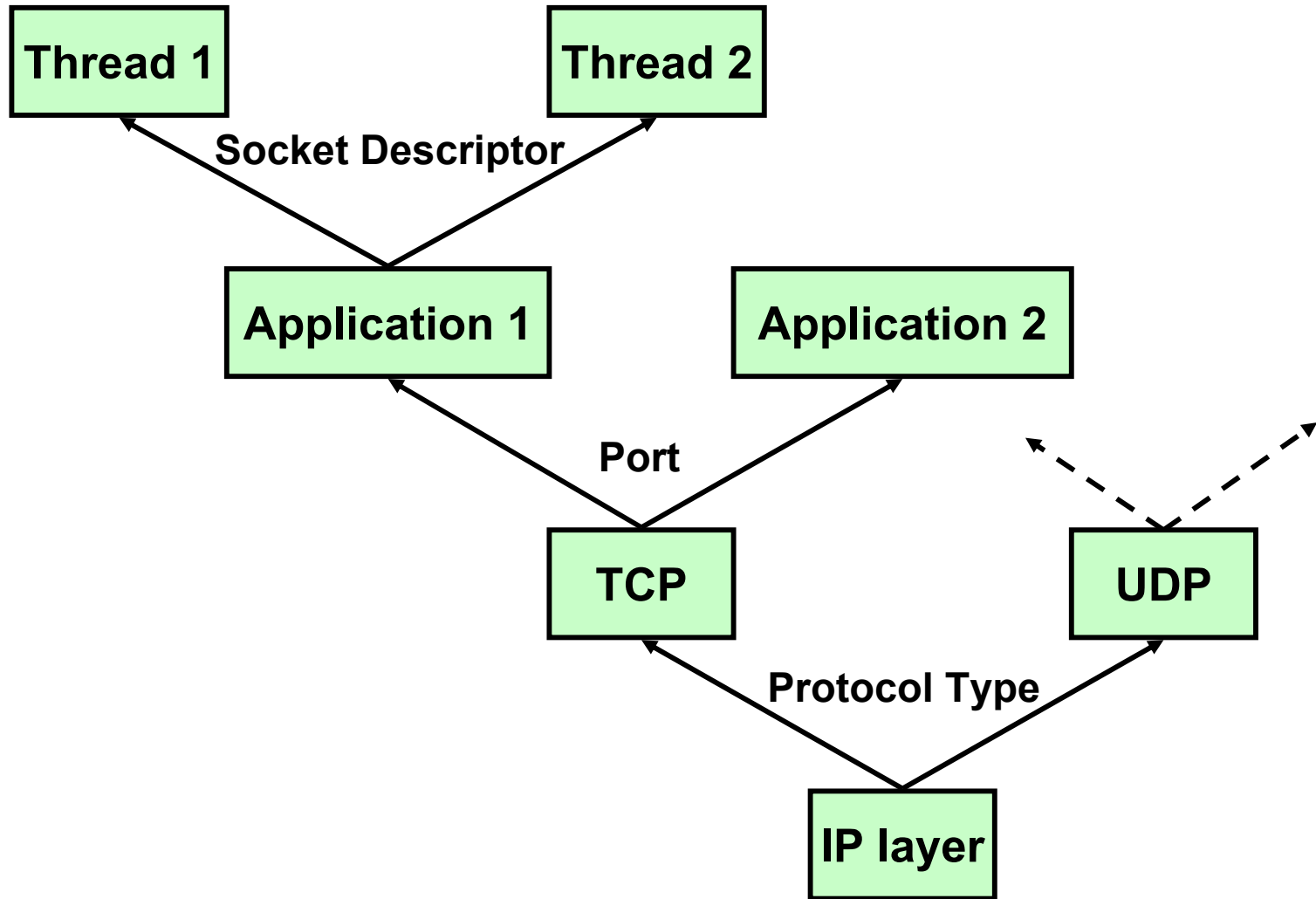
- sono l'interfaccia base per le comunicazioni basate su TCP/IP
- paradigma Client - Server
- sono i punti estremi della comunicazione
- una coppia di socket connessi fornisce un servizio tipo pipe
- identificato da un socket descriptor
 - analogo ad un file descriptor
- Tipi:
 - stream: connessione di tipo affidabile (TCP)
 - datagram: comunicazioni di tipo datagram (UDP)
 - raw: per la manipolazione diretta del pacchetto IP
 - realizzazione di un protocollo di livello 4



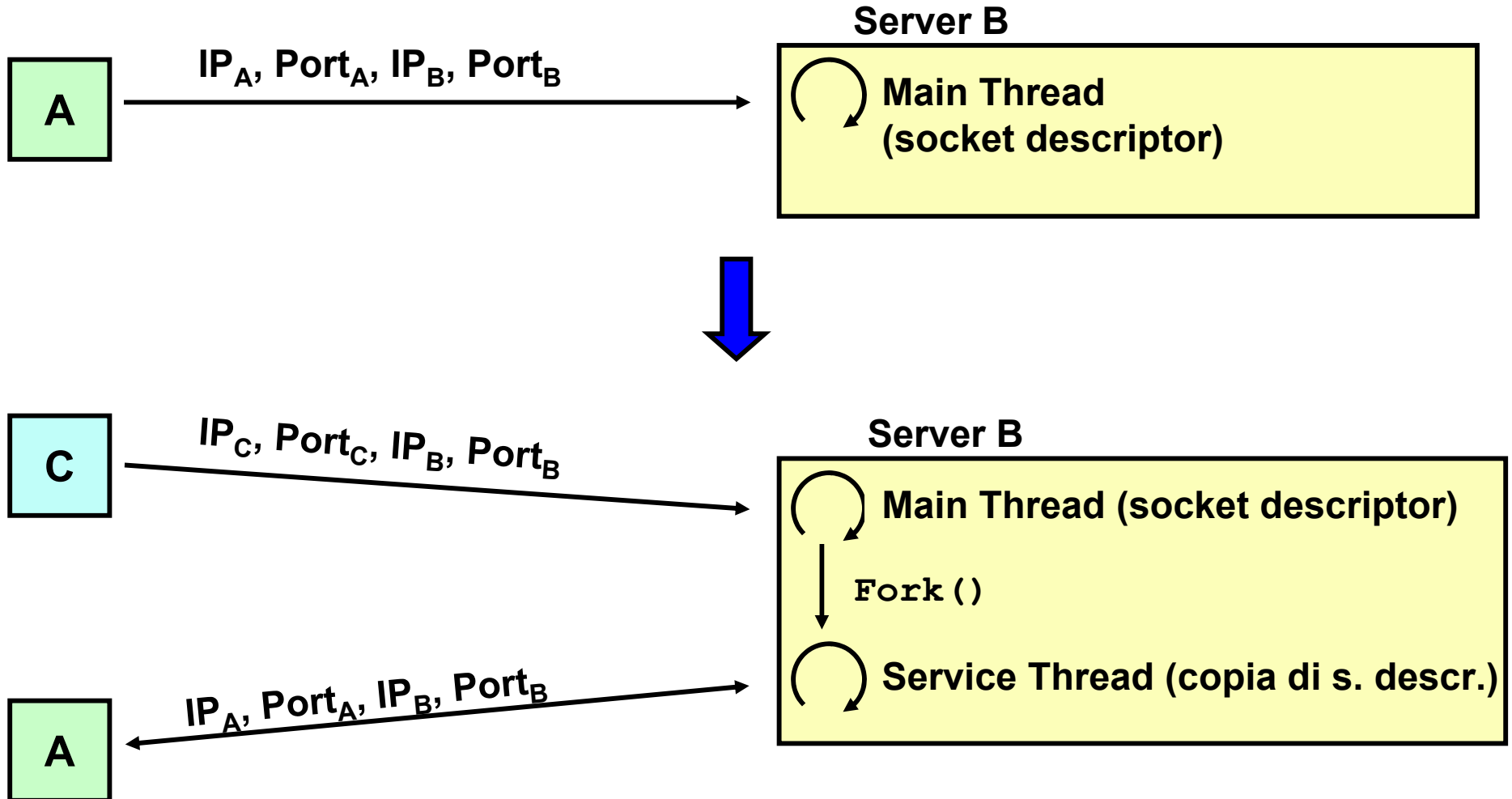
Socket descriptor

- in UNIX è un normale file descriptor riferito ad un socket anziché ad un file
- può essere usato normalmente per la lettura o la scrittura
- si possono usare tutte le system call operanti su file

Socket descriptor



Connessioni multiple con socket TCP





Operazioni con i socket: lato client

- `socket()` : crea un socket descriptor
- `bind()` : collega il descriptor ad una porta locale
- `connect()` : collega il socket locale con uno remoto instaurando la connessione
 - se il socket è di tipo stream viene aperta una connessione TCP
- `read()`, `write()`, ...
- `sendto()`, `recvfrom()`



Operazioni con i socket: lato server

- **socket ()** : crea un socket descriptor
- **bind ()** : collega il descriptor ad una porta locale
- **listen ()** : mette il processo in attesa di richieste di connessioni
- **accept ()** : è bloccante e crea un nuovo socket all'arrivo di una nuova connessione
- Utilizzo
 - un processo server esegue **listen ()** e si blocca in attesa di richieste di connessione
 - all'arrivo di una connessione il server esegue **accept ()** per creare un nuovo descriptor
 - il server clona se stesso (**fork ()**) o crea un thread per la gestione della nuova connessione
 - il processo/thread originale esegue nuovamente **listen ()**