Introduzione

Ho svolto la traccia 3, consistente in un monitoraggio di rete. Per realizzarlo, mi sono servito principalmente delle seguenti librerie:

- argparse per la gestione dei parametri da riga di comando
- icmplib per le richieste ICMP
- threading per la gestione del parallelismo

Funzionamento

Lo script effettua un monitoraggio concorrente, inviando una singola richiesta per volta ad ogni host attraverso un thread specifico.

Lo script permette di specificare:

- indirizzi (indirizzi IPv4, IPv6, hostname, FQDN)
- timeout (tempo di attesa massimo per ricevere una risposta)
- intervallo di monitoraggio (intervallo di tempo)

Lo script effettua un monitoraggio concorrente: ad ogni monitoraggio, per ogni host un thread specifico si occupa di effettuare la richiesta. Al termine di tutte le richieste, vengono mostrati a video i risultati.

Viene gestita correttamente la scadenza del timeout e gli errori a livello di socket o di protocollo vengono segnalati. Vengono inoltre fatti opportuni controlli volti a garantire che i parametri (indirizzi,timeout,intervallo) forniti siano validi.

Utilizzo

Prima di utilizzare lo script, è necessario installare la libreria `icmplib`:

```
pip3 install icmplib
```

A questo punto è possibile utilizzare lo script, servendosi anche dell'apposito help:

```
python ping.py --help
```

Per interromperlo è sufficiente premere CTRL+C.

Esempi

- host in rete locale:

```
python ping.py -a localhost 192.168.1.1
```

- 3 host con intervallo e timeout ridotti:

```
python ping.py -a wikipedia.org cloudflare.com unibo.it -i 1 -t 0.5
```

- host non risolvibili tramite DNS:

```
python ping.py -a abc1234567.it
```

Considerazioni aggiuntive

Lo script è stato pensato per supportare un numero relativamente basso di host, in quanto viene creato un thread per ognuno di essi. Per ottenere una maggiore scalabilità, si potrebbero suddividere gli host in batch e assegnare un batch a ciascun thread.

La gestione dei tempi, inoltre, non è precissima (è osservabile anche dall'output dei risultati, che include anche le frazioni di secondo), in quanto la funzione `time.sleep()` dipende anche dallo scheduling di altre attività da parte del sistema operativo. Tuttavia, in quanto solitamente gli intervalli di tempo dei monitoraggi di rete variano da alcune centinaia di millisecondi a diversi secondi se non minuti, l'errore che si commette non è tale da causare grandi problemi.