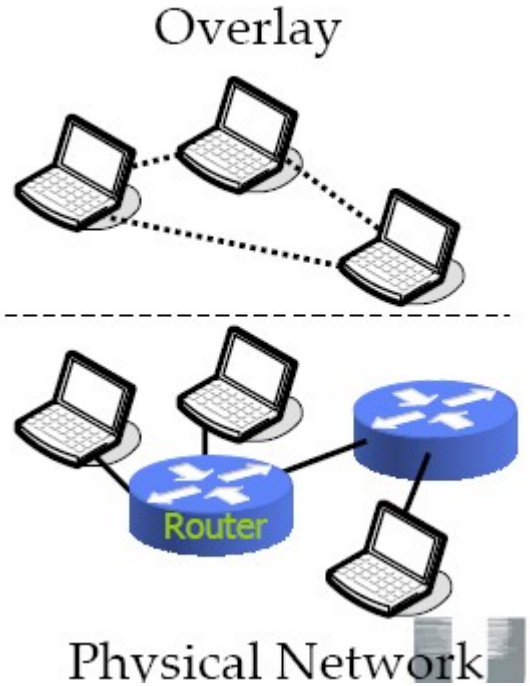
Lezione n.19
22/05/2009
OVERLAY WEAVER
HTTP://OVERLAYWEAVER.SF.NET

Laura Ricci

OVERLAY NETWORKS

Overlay: Una rete costruita 'sopra' un'altra rete

- Internet sopra la rete telefonica
- overlay P2P
  - FastTrack, eDonkey2K, Gnutella, …
  - con più di 1.000.000 di nodi.



Application Level Network

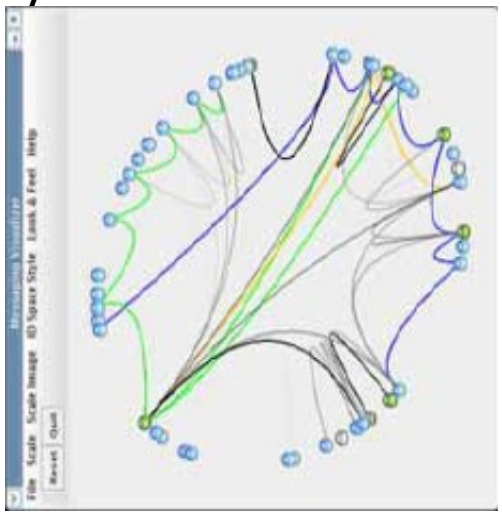
- costruita in modo autonomo e decentralizzatoto
- topologia indipendente dalla rete sottostante (es: Internet)

OVERLAY WEAVER: CARATTERISTICHE GENERALI

- Uno strumento orientato principalmente allo sviluppo di overlay strutturati
  - to weave = tessere, intrecciare
  - weaver = tessitrice
- Gli autori propongono un'analogia tra la tessitura e la creazione di una rete strutturata (es: Chord)
- Una DHT Library
  - sviluppata in Java
  - Apache License 2.0
- Supporta
  - diverse DHT (Chord, Kademlia, Koorde, Pastry,Tapestry, …)
  - application level multicast

OVERLAY WEAVER:CARATTERISTICHE GENERALI

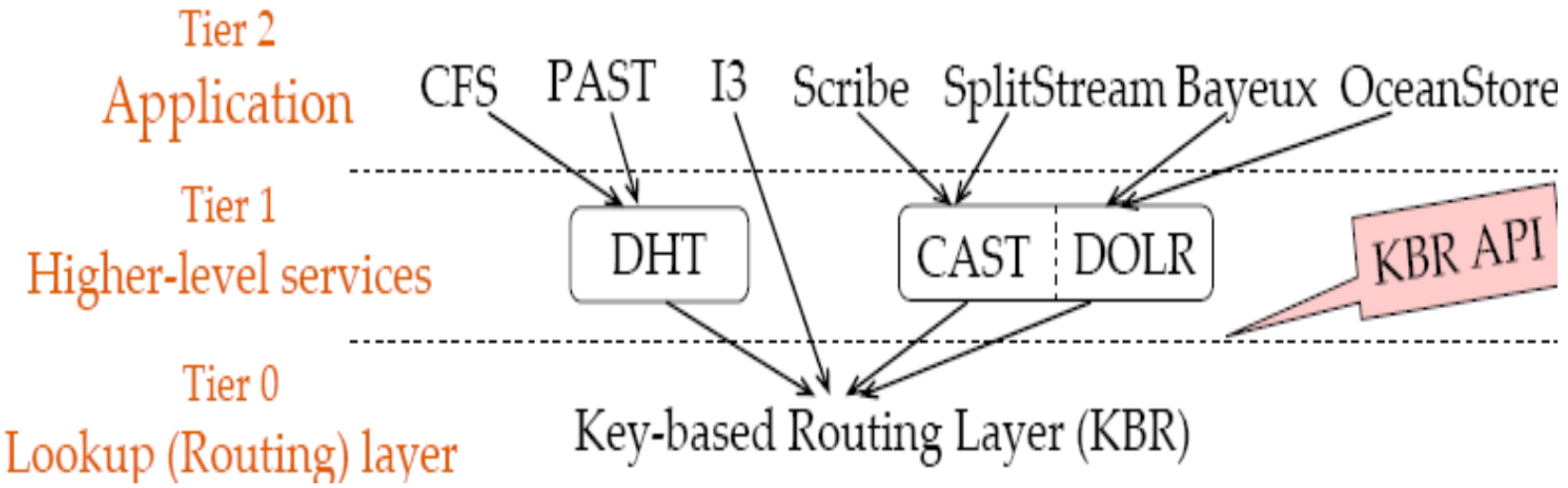
- Possibilità di effettuare esperimenti con strumenti predefiniti oppure mediante lo sviluppo di un numero limitato di linee di codice. Utilizzo di:
  - DHT e multicast shell
  - emulatore per lo sviluppo di esperimenti che consentano di valutare alcuni aspetti (es: numero di messaggi scambiati/ n. di hops,...)
  - tools per la visualizzazione della struttura dell'overlay



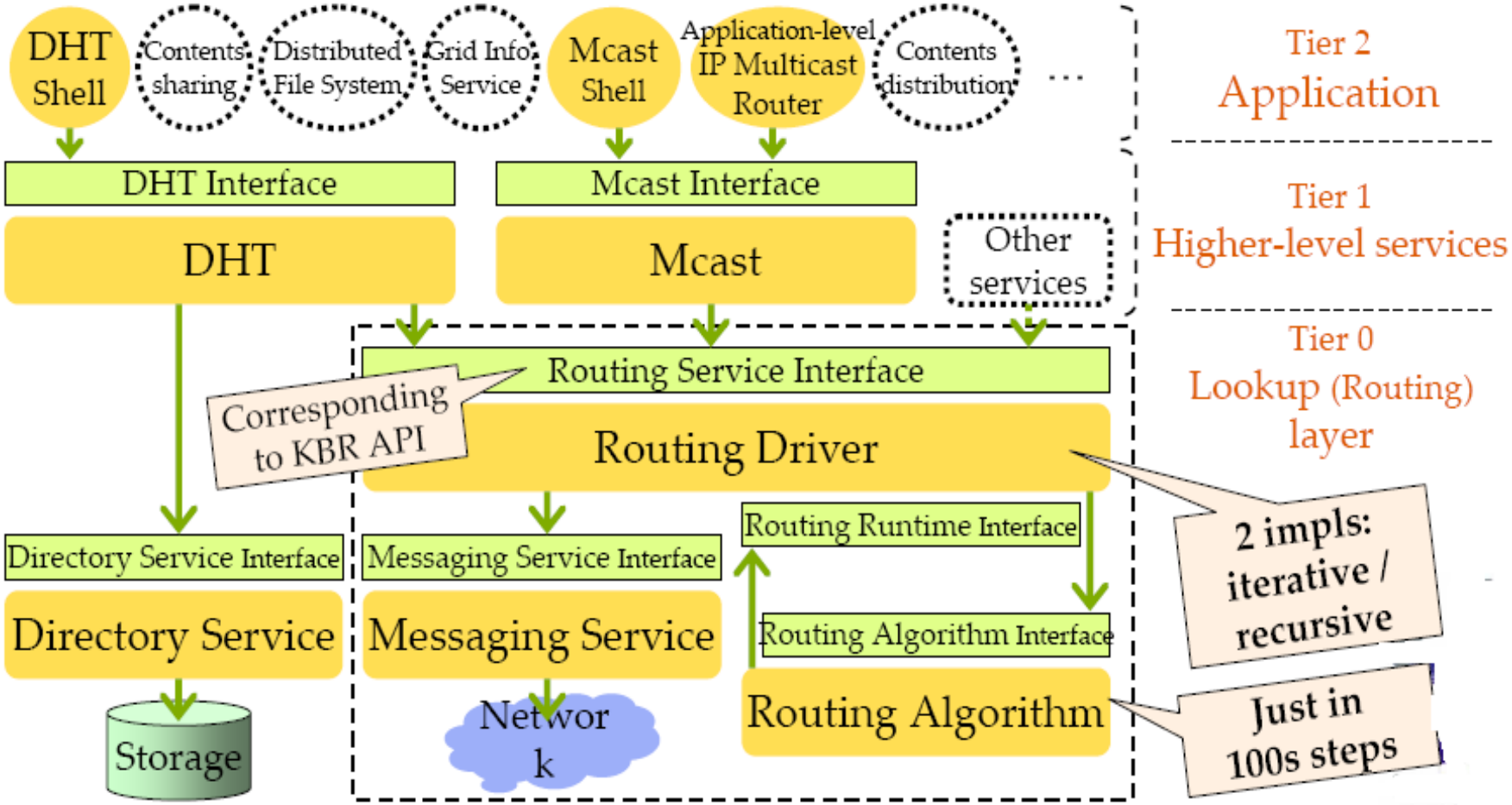
- Caratteristiche dello strumento
  - http://overlayweaver.sf.net/ (SourceForge)
  - May 17, 2009: Version 0.9.4 released.
  - 4.495 downloads al Gennaio 2007
  - utenti
    - sviluppatori di nuovi algoritmi per DHT
    - sviluppatori di nuove applicazioni basate su reti strutturate

OVERLAY WEAVER: IDEE BASE

- Separazione del livello di routing (lookup layer) dai servizi di più alto livello
- Le applicazioni ed i servizi di livello più alto possono essere sviluppati in modo indipendente dal livello di lookup
- Gli algoritmi di routing implementati nelle DHT hanno a comune la caratteristica di essere guidati dalla chiave (key Based Routing)
- Il Look Up Layer include funzionalità comuni a diversi algoritmi di routing tutti basati su una strategia key based

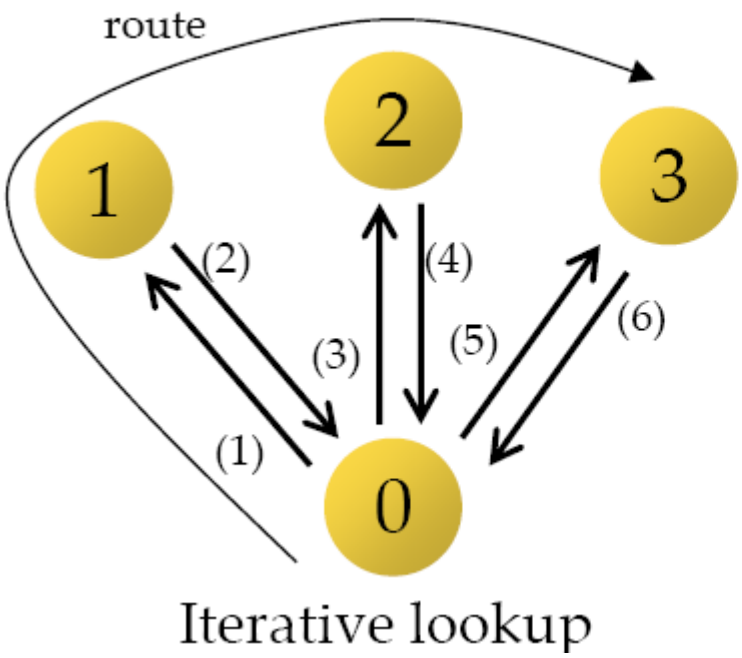


OVERLAY WEAVER: L'ARCHITETTURA

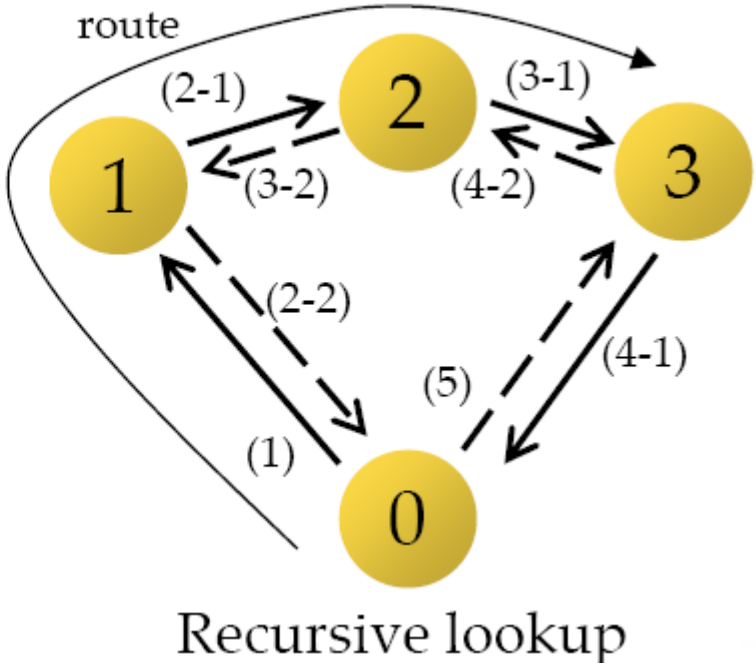


OVERLAY WEAVER: L'ARCHITETTURA

- Diverse implementazioni per ogni livello
- Ad esempio, il livello di routing include diverse strategie
  - Iterativa
  - Ricorsiva

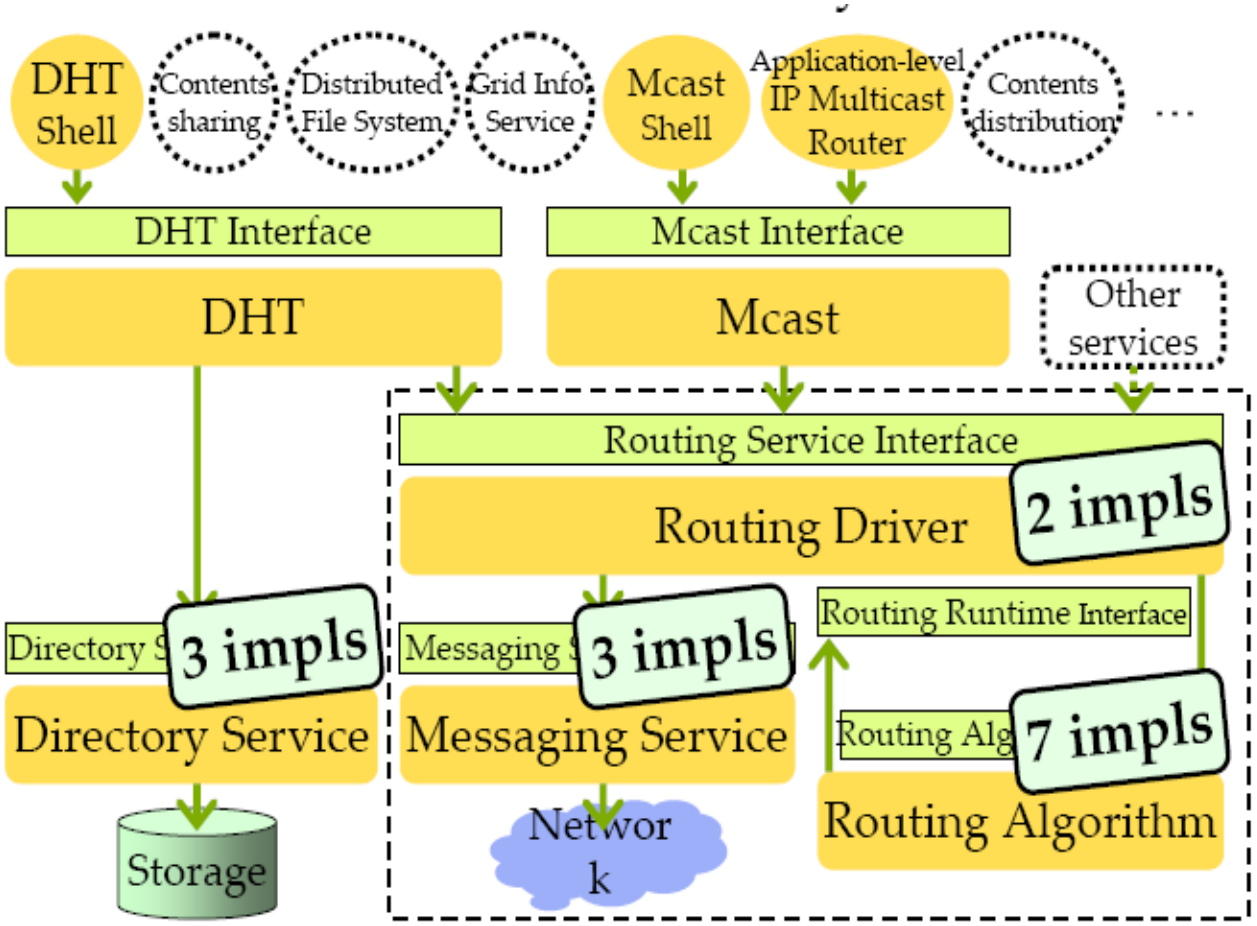


Iterative lookup



Recursive lookup

OVERLAY WEAVER: L'ARCHITETTURA



Routing Driver
- Iterativo
- Ricorsivo

Messaging Service
- UDP
- TCP
- Emulatore

Routing Algorithm
- Chord
- Kademlia
- ....

Directory Service
- Berkeley DB
- ....

OVERLAY WEAVER: DHT SHELL

- E' possibile utilizzare la DHT shell mediante: owdhtshell

  get <key> [<key> ...]
  put <key> <value> [<value> ...] [- <key> <value> [<value> ...] ...]
  remove|delete <secret> <key> [<value> ...] [- <key> [<value> ...] ...]
  ...

- Per default si utilizza Chord con routing iterativo, ma è possibile specificare altre DHT ed altri algoritmi di routing

- status: visualizza la tabella di routing di un nodo

OVERLAY WEAVER: DHT SHELL

put foo bar

status

Last key: 0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33

Last route: [

040aa927c1d39e8a2ad5a91d10548343ec7d8a52:emu0:3997

7e69451702869e2f9142284a65534d082e7744f5:emu2:3997

]

Hash della chiave foo: 0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33

Mostra anche il routing

get foo

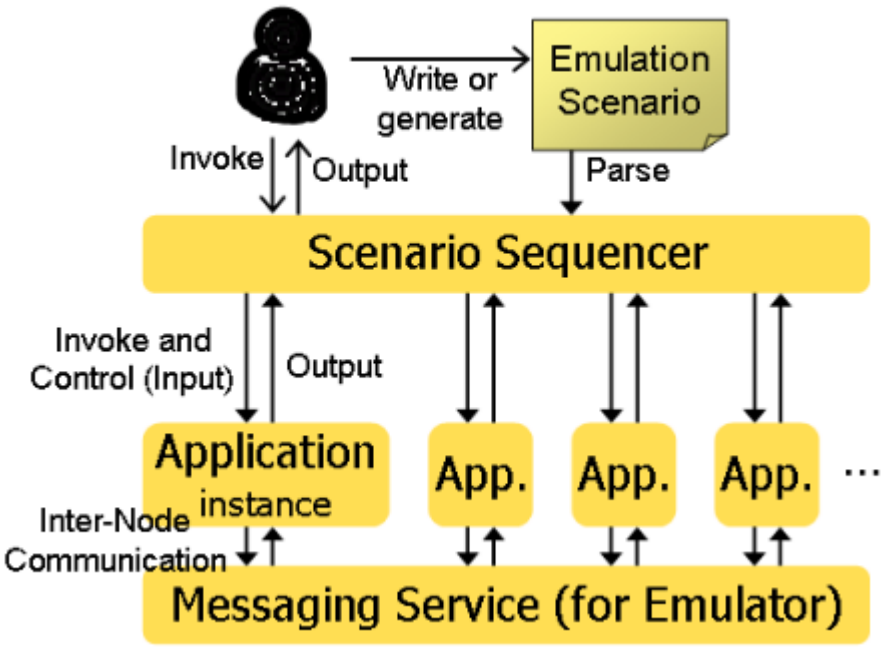
key: 0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33

value: bar 582

DISTRIBUTED ENVIRONMENT EMULATOR

- Da la possibilità di emulare un insieme di nodi della DHT sullo stesso host
- Legge un file scenario che indica quanti nodi attivare, quale programma eseguire su ogni nodo
- Ogni nodo viene eseguito come un thread separato
- Le diverse istanze utilizzano il Messaging Service come se comunicassero su una rete reale
- Ogni nodo può eseguire un'applicazione sviluppata dall'utente
- Tipi di Emulazione
  - Tutti i nodi vengono emulati su un singolo host
  - Distributed Emulation: l'utente invoca un Master Emulator ed un insieme di Worker Emulator su host remoti attivati mediante SSH
  - Anche in questo caso è necessario un file scenario per configurare la emulazione
  - Si utilizza il Messaging Service per l'invio/ricezione di messaggi via TCP/ UDP

DISTRIBUTED ENVIRONMENT EMULATOR



- Utilizzo di un file scenario
- Il file scenario specifica in che ordine vanno invocate le varie istanze, i parametri in ingresso,....
- thread diversi per ogni istanza,

oppure

- istanze associate a nodi diversi

```
# invoke 1000 instances
class ow.tool.dhtshell.Main
arg -p 10000
schedule 0 invoke
arg ptp00.hpcc.jp
schedule 500,500,999 invoke
# put & get
schedule 510000 123 put foo bar 300
schedule 515000 234 get foo
```

A scenario

DISTRIBUTED ENVIRONMENT EMULATOR

- Invocazione dell'emulatore

  owemu -f <host list file> [scenario URL|File]

- Con l'opzione -f l'emulatore lavora in modo distribuito. Nel file specificato devono essere specificati gli host su cui devono essere attivate le diverse istanze dell'applicazione. Ad esempio

  hostname 0 0
  hostname 1 1
  hostname 2 101

  il primo 'virtual node' (emu0) deve essere assegnato ad hostname0, i 'virtual node' da 1 a 100 (emu1,.., emu100) devono essere allocati ad hostname1, e solo l'ultimo virtual node (emu 101) deve essere allocato su hostname2

IL FILE SCENARIO

E' un file di comandi. Alcuni comandi:

- class specifica la classe JAVA in cui si trova il main che viene invocato successivamente dal comando invoke
- arg(arguments) specifica gli argomenti da inviare al main quando viene invocato
- invoke istanzia la classe specificata in precedenza ed invoca il suo metodo

  main
- schedule [+]<time>[,<interval>[,<times>]] <command> [...]

  schedula il comando specificato come secondo argomento (ad esempio un comando invoke) al tempo specificato (in millisecondi).
  - Diverse opzioni disponibili per ilcomando schedule
- quit/exit terminano l'emulatore