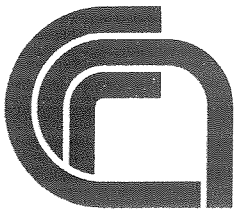


Consiglio Nazionale delle Ricerche

WAMM (Wide Area Metacomputer Manager): Manuale dell'utente Versione 1.1

Ranieri Baraglia, Giancarlo Bartoli, Massimiliano Cosso
Gianluca Faieta, Marcello Formica, Domenico Laforenza, Massimiliano Nicosia

Rapporto
CNUCE-B4-1998-04

Pisa

WAMM (Wide Area Metacomputer Manager): manuale dell'utente

Versione 1.1

Ranieri Baraglia, Giancarlo Bartoli, Massimiliano Cosso
Gianluca Faieta, Marcello Formica,
Domenico Laforenza, Massimiliano Nicosia.

CNUCE - Institute of the Italian National Research Council
Via S.Maria, 36 - I56100 Pisa, Italy
Tel. +39-50-593111 - Fax +39-50-904052
email: R.Baraglia@cnuce.cnr.it, D.Laforenza@cnuce.cnr.it

Indice

1 Introduzione

WAMM (Wide Area Metacomputer Manager) [BFFL] è un'interfaccia grafica che consente di semplificare le operazioni normalmente svolte per configurare ed utilizzare una macchina virtuale parallela basata su PVM. WAMM permette di ottenere una vista *geografica* della macchina virtuale (metacalcolatore): i nodi di elaborazione sono raggruppati secondo una struttura ad albero in accordo alla loro reale dislocazione in reti (WAN, MAN e LAN). Ogni rete viene mostrata in una finestra separata, utilizzando cartine geografiche che facilitano l'*esplorazione* delle risorse computazionali da parte dell'utente.

WAMM comprende una serie di funzionalità per la configurazione ed il controllo di una macchina virtuale PVM; la possibilità di gestire la compilazione di un programma in parallelo su più nodi, permette di ridurre i tempi necessari allo sviluppo delle applicazioni. Inoltre, è possibile effettuare il monitoraggio di applicazioni PVM scritte in linguaggio C. Quest'ultima funzionalità è stata introdotta, mediante l'integrazione in WAMM del tool PVaniM [TSS96a]. Oltre alle capacità di monitoring, WAMM eredita da PVaniM la possibilità di poter cambiare alcuni parametri dell'applicazione durante la sua esecuzione, in modo da poter osservare immediatamente gli effetti indotti da tali variazioni (*steering*).
Per la descrizione degli aspetti implementativi descriventi l'integrazione dei tool WAMM e PVaniM si rimanda a [CN97, BCLN97]

Questo documento descrive le operazioni necessarie per l'installazione, la configurazione e l'uso di WAMM.

2 Installazione

2.1 Requisiti

L'installazione di WAMM richiede che PVM sia già installato su *tutti* i nodi che possono far parte della macchina virtuale. È consigliabile verificare il corretto funzionamento dell'ambiente PVM prima di procedere con l'installazione dell'interfaccia; in particolare, controllare che l'attivazione di PVM e l'inserimento di nodi nella macchina virtuale non presentino problemi. Inoltre è necessario che, per ogni nodo, la directory contenente i programmi PVM (in genere `$PVM_ROOT/bin/$PVM_ARCH`) sia inserita nel path dei comandi.

WAMM si compone dei seguenti programmi:

`wamm`: l'interfaccia grafica;

`PVMTasker`: programma utilizzato da `wamm` per eseguire comandi sui nodi e per ricevere informazioni sui task PVM in esecuzione sulla macchina virtuale;

`olslave`: programma utilizzato per raccogliere informazioni riguardanti il carico sugli elaboratori costituenti la macchina virtuale;

`PVMHoster`: programma usato da `wamm` per controllare l'inserimento di nodi di elaborazione nella macchina virtuale;

`PVMMaker`: programma impiegato da `wamm` per la compilazione parallela;

`PVMKill`: script utilizzato da `wamm` per forzare la terminazione di demoni PVM sui nodi di elaborazione;

`PVANIMSORT`: script che invaca il programma `dualsync` per effettuare la datazione e l'ordinamento degli eventi raccolti durante il monitoraggio;

`PVANIM2PG`: script che invoca i programmi `makepg` e `converter` per effettuare la conversione degli eventi, raccolti durante il monitoraggio, nel formato PICL;

`wammtracelib.a`: libreria delle routine PVM instrumentate per poter effettuare il monitoraggio delle applicazioni.

I sorgenti devono essere compilati utilizzando un compilatore ANSI C (ad esempio, `gcc`). I programmi `PVMTasker`, `PVMHoster`, `PVMMaker`, `PVMKill`, `olslave` e la libreria `wammtracelib.a` devono essere installati su tutti i nodi e richiedono, per la compilazione, unicamente la libreria PVM, versione 3.3 o superiore. Il programma `wamm`, `PVANIMSORT` e `PVANIM2PG` debbono essere installati anche sul solo nodo *locale* (la macchina normalmente impiegata per attivare PVM, poichè i file di traccia eventualmente prodotti vengono raccolti su questo nodo, è consigliabile utilizzare una macchina con buona capacità disco) e richiede, per la compilazione:

- ANSI C;
- PVM, versione 3.3 o superiore;
- X11, release 5 o superiore;
- XPM[1], versione 3.4e o superiore;

[1] `xpm` è una libreria liberamente distribuibile che semplifica l'uso delle pixmap in programmi X11; è disponibile via anonymous FTP all'indirizzo `avahi.inria.fr`

- OSF/Motif, versione 1.2 o superiore.

WAMM è stato sviluppato e provato sui seguenti sistemi:

- HP 9000/720 con sistema operativo HP-UX 9.01;
- Sun SparcStation 2 con sistema operativo SunOS 4.1.3;
- IBM Risc6000 con sistema operativo AIX 3.2;
- IBM Sp2 con sistema operativo AIX 3.2;
- DEC Alpha (OSF/1 3.2)

2.2 Distribuzione dei sorgenti

La prima operazione da compiere per installare WAMM consiste nel prelevarne i sorgenti e distribuirli su tutti i nodi utilizzati. I sorgenti sono disponibili via anonymous FTP all'indirizzo:

```
ftp.cnr.it, /pub/wamm/wamm11.tar.gz % XXX %
```

Ricopiare `wamm11.tar.gz` su ogni nodo (via `ftp` o `rcp`). Per ogni copia, eseguire i seguenti comandi al fine di estrarre i file sorgenti:

```
gzip -d wamm11.tar.gz
tar xvf wamm11.tar
```

I file sono inseriti in una directory di nome `wamm11`.

2.3 Compilazione

Editare, per ogni nodo, il `Makefile` presente nelle directory `wamm11`, seguendo le istruzioni contenute al suo interno. Sul nodo *locale* dare quindi i comandi:

```
make
make install
```

I programmi `PVMTasker`, `PVMMaker`, `PVMHoster`, `PVMKill`, `olslave`, `PVANIMSORT` e `PVANIM2PG` vengono inseriti nella directory usata da PVM come deposito per i file eseguibili; la libreria `wammtracelib.a` viene inserita nella directory in cui sono presenti le librerie PVM (in genere `$PVM_ROOT/lib/$PVM_ARCH`); il programma `wamm` è copiato nella directory indicata nel `Makefile`. Al termine della compilazione inserire il contenuto del file `Wamm.ad` all'interno del file `.Xdefaults` nella home directory dell'utente:

```
cat Wamm.ad >> ~/.Xdefaults
```

In alternativa si può spostare direttamente `Wamm.ad` nella directory contenente i file che definiscono i valori per le risorse X11 dei programmi, togliendo il suffisso `.ad`:

```
mv Wamm.ad $XAPPLRESDIR/Wamm
```

Su tutti gli *altri* nodi eseguire i comandi:

```
make slaves
make install-slaves
```

per compilare e inserire nella directory appropriata i programmi `PVMTasker`, `PVMMaker`, `PVMHoster`, `PVMKill`, `olslave`, `PVANIMSORT` e `PVANIM2PG` e la libreria `wammtracelib.a` (in questo caso `wamm` non viene compilato).

3 Configurazione

Per utilizzare WAMM è necessario preparare un file di configurazione contenente la descrizione dei nodi che possono far parte del metacalcolatore; il file viene letto da `wamm` all'avvio del programma.

Sebbene sia concettualmente analogo ad un *host file* standard di PVM, il file di configurazione di WAMM contiene una serie di informazioni aggiuntive che permettono di raggruppare i nodi in *reti*. Le reti sono strutturate ad albero, secondo lo schema di Figura 1:

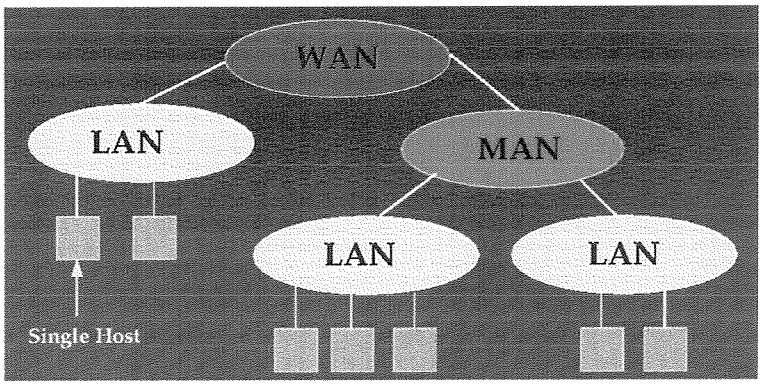


Figura 1: Struttura della rete

- le foglie dell'albero sono costituite dai singoli host;
- gli host hanno per padre reti di tipo LAN (Local Area Network);
- le LAN hanno per padre reti di tipo MAN (Metropolitan Area Network) o di tipo WAN (Wide Area Network);
- tutte le MAN hanno per padre una rete WAN (la radice dell'albero).

In genere la suddivisione delle macchine in sottoreti viene decisa in base alla loro reale dislocazione geografica. Per esempio, tutti i nodi appartenenti ad uno stesso centro di calcolo possono essere inseriti in un'unica LAN; in questo caso si potrebbe usare una MAN per descrivere i centri di calcolo della stessa città.

Un file di configurazione contiene sempre una e una sola WAN. Nel caso in cui una MAN o una WAN contengano un solo nodo, è necessario inserire nell'albero un nodo LAN atto a contenere l'host.

È importante notare che gli host effettivamente inseriti nella macchina virtuale non sono necessariamente tutti quelli dichiarati nel file: utilizzando opzioni descritte nel seguito è possibile fare in modo che, quando il metacalcolatore viene creato, alcuni degli host non siano utilizzati. È possibile variare dinamicamente la configurazione del sistema in un secondo tempo, usando i comandi messi a disposizione dall'interfaccia per aggiungere host o rimuoverli.

3.1 Linguaggio di configurazione

Il file di configurazione viene scritto mediante un semplice linguaggio dichiarativo. I nodi e le reti sono definiti usando una sintassi simile a quella impiegata in C per dichiarare le strutture.

In questa sezione verranno esaminati tutti i costrutti permessi dal linguaggio; è consigliabile fare riferimento al file di configurazione `italy.net`, distribuito con i sorgenti, che descrive un metacomputer completo con più di 60 nodi distribuiti in vari laboratori italiani.

3.1.1 Elementi del linguaggio

Il linguaggio di specifica usato da WAMM si basa sui seguenti elementi:

- **commenti**

  Un commento è una stringa che inizia con il carattere `#` e termina alla fine della riga. Il contenuto dei commenti viene ignorato da `wamm`.

- **direttive;**

  Una direttiva è un'istruzione della forma:

  Nome Arg1 Arg2...

  Esempio:

  ```
  PIXPATH /home/meta/pixmaps
  ```

  Le direttive vengono utilizzate per controllare il parsing del file di configurazione e per costruire le strutture. Gli argomenti possono contenere spazi, ma in questo caso devono essere racchiusi tra apici (").

- **strutture**

  Una struttura descrive un host o una rete. Ha la forma:

Tipo Nome {
    Direttiva1 Arg1 Arg2...
    Direttiva2 Arg1 Arg2...
    ...
}

Il *Tipo* di una struttura può essere `HOST` per descrivere una singola macchina, oppure `LAN`, `MAN` o `WAN` per descrivere una rete. Il *Nome* identifica la struttura: non può contenere spazi e deve essere unico.

3.1.2 WAN

La radice dell'albero che descrive la macchina virtuale viene specificata per mezzo della seguente dichiarazione:

`WAN` *nome_WAN* {
    ...
    ...
}

La dichiarazione di una struttura WAN può contenere le seguenti direttive:

- `TITLE` *titolo_finestra*

  Permette di specificare il titolo da attribuire alla finestra contenente la WAN. Se non viene utilizzata la direttiva `TITLE`, il titolo della finestra sarà uguale al nome assegnato alla struttura WAN (*nome_WAN*).

- `PICT` *nome_file*

  `wamm` mostra le informazioni relative alle reti WAN, MAN e LAN in finestre separate. Questa direttiva permette di specificare una figura (in formato *xpm*) utilizzata come sfondo per la finestra WAN. Generalmente viene utilizzata una cartina geografica che rappresenta l'area su cui si estende la rete. Nel caso in cui non sia possibile mostrare la cartina (colori non disponibili, file non esistente, ecc.), `wamm` produrrà un messaggio di errore.

- `MAN` *nome_MAN pos_X pos_Y*

  Ogni direttiva `MAN` permette di dichiarare la struttura di tipo MAN e nome *nome_MAN* come figlia della struttura WAN corrente. `wamm` mostrerà, sulla cartina specificata con la direttiva `PICT`, un bottone utilizzabile per aprire la finestra della MAN. I parametri *pos_X* e *pos_Y* sono le coordinate X e Y del bottone.

- `LAN` *nome_LAN pos_X pos_Y*

  Ogni direttiva `LAN` permette di dichiarare la struttura di tipo LAN e nome *nome_LAN* come figlia della struttura WAN corrente. `wamm` mostrerà, sulla cartina specificata con la direttiva `PICT`, un bottone utilizzabile per aprire la finestra della LAN. I parametri *pos_X* e *pos_Y* sono le coordinate X e Y del bottone.

Per ogni direttiva `MAN` oppure `LAN` inserita all'interno della struttura WAN è necessario dichiarare una corrispondente struttura MAN o LAN con lo stesso nome, seguendo la sintassi che verrà illusttrata nelle sezioni successive. Non occorre rispettare alcun ordine nella collocazione delle dichiarazioni: la struttura figlia può essere definita prima o dopo la WAN.

L'esempio seguente mostra una possibile struttura WAN per una rete geografica comprendente la rete metropolitana di Pisa e un centro di calcolo di Firenze:

```
WAN italy {
   TITLE "WAN Italia"
   PICT italy.xpm
   MAN pisa 100 130       # Pisa
   LAN firenze 150 120    # Firenze
}
```

3.1.3 MAN

Ciascun nodo MAN viene specificato nel seguente modo:

`MAN` *nome_MAN* `{`

```
    ...
    ...
}
```

La struttura può contenere le direttive:

- `TITLE` *titolo_finestra*

  Permette di specificare il titolo da attribuire alla finestra contenente la MAN. Se non viene utilizzata la direttiva `TITLE`, il titolo della finestra sarà uguale al nome assegnato alla struttura MAN (*nome_MAN*).

- `PICT` *nome_file*

Permette di specificare una figura (in formato *xpm*) da associare alla struttura MAN. Generalmente viene utilizzata una cartina che rappresenta l'area su cui si estende la MAN. Nel caso in cui non sia possibile mostrare la cartina (colori non disponibili, file non esistente, ecc.), `wamm` produrrà un messaggio di errore.

- `LAN` *Nome_LAN pos_X pos_Y*

  Ogni direttiva `LAN` permette di dichiarare la struttura di tipo LAN e nome *nome_LAN* come figlio della struttura MAN corrente. `wamm` mostrerà, sulla cartina specificata con la direttiva `PICT`, un bottone utilizzabile per aprire la finestra della LAN. I parametri *pos_X* e *pos_Y* sono le coordinate X e Y del bottone.

Anche in questo caso, per ogni direttiva `LAN` inserita all'interno della dichiarazione della MAN è necessario dichiarare una corrispondente struttura LAN con lo stesso nome.

Continuando l'esempio precedente:

```
WAN italy {
   TITLE "WAN Italia"
   PICT italy.xpm
   MAN pisa 100 130       # Pisa
   LAN firenze 150 120    # Firenze
}

MAN pisa {
   TITLE "MAN Pisa"
   PICT pisa.xpm
   LAN cnuce 200 100  # Istituto CNUCE-CNR
   LAN sns 280 55     # Scuola Normale Superiore
}
```

3.1.4 LAN

I nodi LAN devono essere specificati in accordo al seguente schema:

`LAN` *nome_LAN* `{`

```
    ...
    ...
}
```

La dichiarazione può includere le direttive:

- TITLE *titolo_finestra*

  Permette di specificare il titolo da attribuire alla finestra contenente la LAN. Se non viene utilizzata la direttiva TITLE, il titolo della finestra sarà uguale al nome assegnato alla struttura LAN (*nome_LAN*).

- HOST *nome_Host*

  Ogni direttiva HOST permette di dichiarare la struttura di tipo HOST e nome *nome_host* come figlia della struttura LAN corrente. wamm mostrerà le informazioni sull'host nella finestra associata alla LAN.

Analogamente ai casi precedenti, per ogni direttiva HOST inserita all'interno della dichiarazione della LAN è necessario dichiarare una corrispondente struttura HOST con lo stesso nome. Le strutture LAN non hanno direttive PICT in quanto le finestre a loro associate contengono le icone degli host contenuti al loro interno.

Ad esempio aggiungiamo tre host alla rete dell'esempio precedente:

```
WAN italy {
   TITLE "WAN Italia"
   PICT italy.xpm
   MAN pisa 100 130       # Pisa
   LAN firenze 150 120    # Firenze
}

MAN pisa {
   TITLE "MAN Pisa"
   PICT pisa.xpm
   LAN cnuce 200 100  # Istituto CNUCE-CNR
   LAN sns 280 55     # Scuola Normale Superiore
}

LAN cnuce {
   TITLE "CNUCE-CNR"
   HOST calpar        # host della LAN: sono i nomi,
   HOST miles         # non gli indirizzi Internet!
   HOST evans
}
```

3.1.5 HOST

L'ultimo tipo di struttura ammesso da WAMM è quello usato per dichiarare gli host:

```
HOST nome_host {
      ...
      ...
}
```

Il nome assegnato alla struttura (*nome_host*) non ha alcuna relazione con l'indirizzo Internet della macchina, specificato invece con la direttiva `ADDRESS`. La struttura può contenere le seguenti direttive:

- `ADDRESS` *indirizzo_macchina*

  Permette di specificare l'indirizzo Internet della macchina. È fondamentale che l'indirizzo venga inserito usando lo stesso formato impiegato da PVM. Un errore comune consiste nello specificare nella direttiva un indirizzo con dominio (es: `pink.foo.org`) quando PVM usa soltanto il nome `pink`.

- `PICT` *nome_file*

  Permette di specificare una figura (in formato *xpm*) da associare alla macchina. Generalmente viene utilizzata un'icona che permette di distinguerne l'architettura (alcune icone di esempio sono distribuite con i sorgenti, nella directory `pixmaps`); l'icona è mostrata all'interno della finestra relativa alla rete LAN che contiene l'host. Le dimensioni dell'immagine da utilizzare devono essere al massimo di 80 $\times$ 60; utilizzando figure più grandi sono possibili sovrapposizioni. Se la direttiva PICT non viene usata, `wamm` tenta di usare per default un file ottenuto aggiungendo il suffisso `.xpm` al nome dell'architettura (cfr. direttiva `ARCH`). Nel caso in cui non sia possibile mostrare l'icona (file non esistente, colori non disponibili, ecc.), `wamm` produrrà un messaggio di errore.

- `ARCH` *nome_architettura*

  Consente di specificare la classe architetturale della macchina. Il nome inserito sarà visualizzato in una riga di informazioni a destra di ciascun host. Generalmente vengono usati i nomi delle architetture così come definiti in PVM (ad es. `SUN4`, `RS6K`, ecc.). Attualmente queste informazioni non vengono utilizzate da WAMM, ma è possibile che versioni future le sfruttino per esegurire operazioni su gruppi di macchine aventi la stessa classe architetturale.

- `OPTIONS` *opzioni_PVM*

  È possibile specificare, tramite questa direttiva, i parametri di attivazione di PVM per lo specifico nodo. Ad esempio è possibile inserire l'opzione `so=pw` che indica a PVM di utilizzare il comando `rexec` anzichè il comando `rsh` per fare partire il processo demone su tale macchina. Possono essere usate più opzioni, purchè siano separate da spazi (in questo caso occorre racchiuderle tutte tra doppi apici). Per poter utilizzare il flag `&`, usato da PVM per indicare che il nodo non deve essere inserito subito nella macchina virtuale, è necessario che venga specificato come primo carattere della stringa *opzioni_PVM*.

- `INFO` *informazioni*

  Utilizzando questa direttiva è possibile associare una riga di informazioni a ciascun nodo. Le informazioni inserite verranno visualizzate in una riga di stato alla destra di ciascun host.

- `CMD` *voce_menu comando*

  Ad ogni host mostrato nella finestra LAN viene associato un menu popup, utilizzabile per richiamare varie funzionalità dell'interfaccia. Questa direttiva permette di aggiungere un comando al menu. Richiamando la voce *voce_menu*, il corrispondente *comando* viene eseguito *sull'host*. Ad esempio, per avere l'elenco di tutti i processi in esecuzione su una macchina, la direttiva da inserire potrebbe essere:

  ```
  CMD "Tutti i processi" "ps -aux"
  ```

  In questo modo, richiamando la voce `Tutti i processi` del menu, viene eseguito sul nodo il comando `ps -aux`. L'output dei comandi è mostrato in un'apposita area messaggi all'interno della finestra LAN a cui appartiene l'host. Non usare `CMD` per i programmi X11: per questi occorre fare uso della direttiva `XCMD`, descritta successivamente.

  Per inserire un separatore nel menu popup è sufficiente utilizzare la direttiva `CMD` nel seguente modo:

  ```
  CMD --- ---
  ```

- `XCMD` *voce_menu comando*

  La direttiva `XCMD` è analoga alla precedente e viene utilizzata per i programmi X-Windows. Gli argomenti richiesti sono: il testo da inserire

nel menu popup e il comando da eseguire. Quando si richiama il comando dal menu, le sue finestre vengono mostrate sullo stesso display usato per le finestre di `wamm`. L'autorizzazione ad accettare connessioni X11 provenienti dall'host su cui si esegue il comando viene automaticamente richiesta da `wamm`. Quando il programma termina, il codice di errore restituito viene mostrato nell'area dei messaggi. Esempio:

```
XCMD XLoad xload
```

Aggiungiamo all'esempio precedente la definizione per la macchina `calpar`:

```
WAN italy {
   TITLE "WAN Italia"
   PICT italy.xpm
   MAN pisa 100 130       # Pisa
   LAN firenze 150 120    # Firenze
}

MAN pisa {
   TITLE "MAN Pisa"
   PICT pisa.xpm
   LAN cnuce 200 100  # Istituto CNUCE-CNR
   LAN sns 280 55     # Scuola Normale Superiore
}

LAN cnuce {
   TITLE "CNUCE-CNR"
   HOST calpar        # host della LAN: sono i nomi,
   HOST miles         # non gli indirizzi Internet
   HOST evans
}

HOST calpar {
   ADDRESS calpar.cnuce.cnr.it   # indirizzo Internet
   PICT /home/meta/pix/HP720.xpm # icona
   ARCH HPPA                     # architettura
   INFO "HP 9000/720"            # informazioni varie
   XCMD "NEdit" "nedit"          # comandi remoti: un editor...
   CMD --- ---                   # ... separatore ...
   CMD "Uptime" "uptime"         # e "uptime" (carico sistema)
}
```

3.1.6 PIXPATH

La direttiva `PIXPATH` definisce il path di default da utilizzare per caricare le immagini specificate con le direttive `PICT` *successive*. Può essere utilizzata in qualsiasi punto del file di configurazione, ma non all'interno delle strutture. Esempio:

```
PIXPATH /home/meta/pix
...
HOST calpar {
    PICT HP720.xpm
    ...
}
```

In questo caso, `wamm` cerca il file `/home/meta/pix/HP720.xpm`. Nel caso in cui la direttiva venga ripetuta, il nuovo path sostituisce il precedente. Per annullare del tutto l'effetto di una direttiva `PIXPATH` si può usare la forma:

```
PIXPATH ---
```

4 Attivazione di WAMM

Per avviare WAMM usare il comando:

`wamm` [*opzioni_X11*] [*file_conf*]

È possibile specificare, come *opzioni_X11*, tutti gli argomenti facoltativi utilizzabili con i programmi X11 (l'elenco completo si può ottenere con il comando `man X`). Il parametro opzionale *file_conf* è il nome del file di configurazione da utilizzare. Se non viene specificato, `wamm` usa per default il file indicato nella variabile di ambiente `WAMMNET` oppure, se questa non è definita, il file `$HOME/.wammnet`.

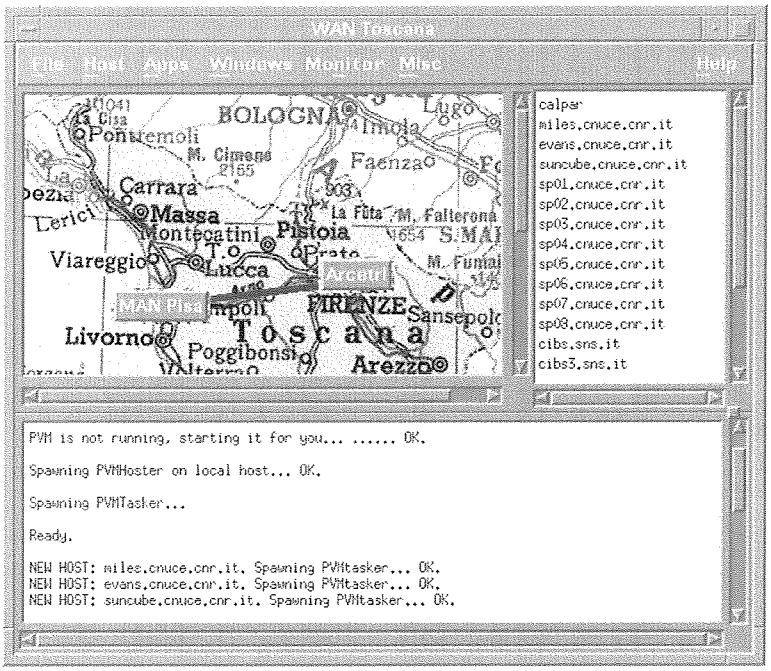
All'attivazione, `wamm` esegue le seguenti operazioni:

1. parsing del file di configurazione;
2. caricamento di tutti i file *xpm* specificati nelle direttive `PICT`; questa fase richiede qualche secondo;
3. apertura della finestra principale del programma, corrispondente alla struttura WAN;
4. se PVM non è attivo, viene fatto partire utilizzando un *host file* temporaneo ricavato dal file di configurazione: tutti gli host privi dell'opzione `&` sono aggiunti alla macchina virtuale. Se PVM è già stato attivato, `wamm` si limita a controllare la configurazione esistente del sistema; in questo caso le opzioni specificate nel file di configurazione usando le direttive `OPTIONS` sono ignorate;
5. attivazione dell'esecuzione del processo `PVMHoster`;
6. su ogni host definito nel file di configurazione e presente nella macchina virtuale viene mandato in esecuzione il processo `PVMTasker`.

L'esito di queste operazioni viene mostrato nell'area dei messaggi presente nella finestra WAN. Terminata la fase di inizializzazione, è possibile utilizzare l'interfaccia attraverso i menu e i bottoni della finestra WAN.

5 Lavorare con WAMM

5.1 Finestra WAN



La finestra WAN costituisce la finestra "base" di `wamm`. Viene utilizzata per lavorare su gruppi di host, anche appartenenti a reti diverse, e per controllare il funzionamento generale della macchina virtuale. È suddivisa in tre parti:

- **mappa geografica** (in alto a sinistra)

  Questa sezione viene utilizzata per mostrare l'immagine specificata all'interno della struttura WAN mediante la direttiva `PICT`. Per ogni direttiva `MAN` e `LAN`, inserita nella struttura per definire una rete figlia, viene posizionato sull'immagine un bottone. Selezionandolo con il tasto sinistro viene aperta la finestra corrispondente alla sottorete. La posizione del bottone è data dalle coordinate specificate come parametri delle direttive `MAN` e `LAN`; l'etichetta corrisponde sempre al titolo della finestra che viene aperta (cfr. direttiva `TITLE`).

- **lista degli host** (in alto a destra)

  Contiene gli indirizzi Internet di *tutti* gli host dichiarati nel file di configurazione. Molte operazioni richiamabili dal menu della finestra WAN richiedono che siano selezionati uno o più nodi della lista; è possibile

selezionarli o deselezionarli tutti richiamando, rispettivamente, `Select All` o `Deselect All` dal menu `Host` della finestra. La selezione di un gruppo di host contigui nella lista avviene utilizzando il tasto sinistro del mouse tenendo premuto [SHIFT] ; per selezionare più host non contigui usare il tasto sinistro tenendo premuto [CTRL] . La lista permette di accedere rapidamente alla LAN contenente un particolare host: è sufficiente un doppio click con il tasto sinistro sul suo indirizzo per aprire la finestra corrispondente.

- **area messaggi** (in basso)

  Quest'area viene utilizzata per mostrare i messaggi di controllo e di errore prodotti da `wamm` o da PVM. In particolare sono mostrate le informazioni relative ad *eventi* quali, ad esempio, l'inserimento o la rimozione di nodi. Se, un qualsiasi task PVM determina una modifica alla configurazione della macchina virtuale, in quest'area vengono prodotti i messaggi che informano l'utente delle modifiche avvenute. Per cancellare i messaggi esistenti si può usare il comando `Clear Messages` del menu `Misc`.

5.1.1 Controllo della configurazione

Una delle operazioni più frequenti, lavorando con un metacalcolatore, consiste nel variarne dinamicamente la configurazione. Molte volte, ad esempio, è necessario reinserire nella macchina virtuale nodi che sono stati esclusi da PVM in quanto temporaneamente irraggiungibili.

Prima di aggiungere uno o più host è sempre consigliabile controllarne lo stato:

1. Utilizzare `PVM Check`, nel menu `Host` della finestra, per verificare se i nodi sono già presenti in PVM. Per ciascuna delle macchine selezionate nella lista degli host `PVM Check` mostra lo stato corrente (`not in PVM` oppure `OK!`) nell'area dei messaggi.

2. Per verificare se l'host è raggiungibile si può usare il comando `Ping`, selezionabile nel menu `Host`. In molti casi, se la macchina non risponde al ping, il problema dipende dalla rete. Il comando `Traceroute`, selezionabile nello stesso menu, può fornire maggiori dettagli. `Ping` e `Traceroute` richiamano gli omonimi programmi Unix, che devono essere disponibili sul nodo locale. L'output dei comandi viene mostrato in finestre separate, una per ciascuna delle macchine selezionate nella lista degli host.

3. Se gli host risultano raggiungibili e non sono in PVM, possono essere aggiunti eseguendo il comando `PVM Add`, presente nel menu `Host`. L'esito dell'inserimento viene mostrato nell'area dei messaggi solo quando questa informazione è disponibile per *tutti* i nodi selezionati. L'operazione può dunque richiedere qualche secondo; per evitare di bloccare l'interfaccia, viene eseguita in background. PVM non consente, tuttavia, di eseguire un nuovo comando `PVM Add` prima che sia terminato il precedente; se tale procedura viene richiamata mentre ne è ancora in corso un'altra, nell'area dei messaggi viene mostrato un errore.

Se PVM è stato avviato da `wamm`, quando si inserisce un nodo vengono utilizzate le opzioni di attivazione eventualmente specificate con le direttive `OPTIONS`. In particolare, per inserire gli host con l'opzione `so=pw` è necessario immettere la relativa password UNIX. In questo caso la password viene chiesta da `wamm` in un'apposita finestra di dialogo.

Spesso capita che l'inserimento di un nodo fallisca, producendo il messaggio di errore `"already in PVM"` anche se il comando `PVM Check` indica che l'host non fa parte di PVM. Questa situazione è quasi sempre determinata dalla presenza di un vecchio file `/tmp/pvmd.<uid>` sul nodo: in questo caso si può provare a "riparare" l'host utilizzando il comando `PVM Repair` del menu `Host`, e rieseguendo il comando `PVM Add`. `PVM Repair` esegue il programma `PVMKill` su tutti i nodi selezionati; il programma elimina tutti i processi `pvmd` dell'utente eventualmente attivi e cancella i file PVM presenti nella directory `/tmp`.

Per rimuovere uno o più nodi da PVM è sufficiente selezionarli ed eseguire il comando `PVM Remove` dal menu `Host`. Un'apposito messaggio di conferma della rimozione viene mostrato nell'area dei messaggi.

5.1.2 Compilazione dei programmi

WAMM mette a disposizione dell'utente alcune facilitazioni per lo sviluppo di programmi per metacalcolatore; in particolare è possibile scrivere i sorgenti per l'intera applicazione sul nodo locale, utilizzando l'interfaccia per distribuirli e compilarli sui nodi remoti. WAMM è in grado di compilare applicazioni scritte in qualsiasi linguaggio (a patto di disporre dei relativi compilatori sui nodi usati); non è necessario che i programmi facciano uso delle librerie PVM.

Per poter usufruire della funzionalità di compilazione remota è necessario definire una directory *top level* contenente il `Makefile`; i sorgenti possono poi essere organizzati all'interno della stessa directory o in sottodirectory interne.

Una volta preparata la directory con tutti i sorgenti dell'applicazione ed il `Makefile`, è possibile selezionare gli host su cui si vuole effettuare la com-

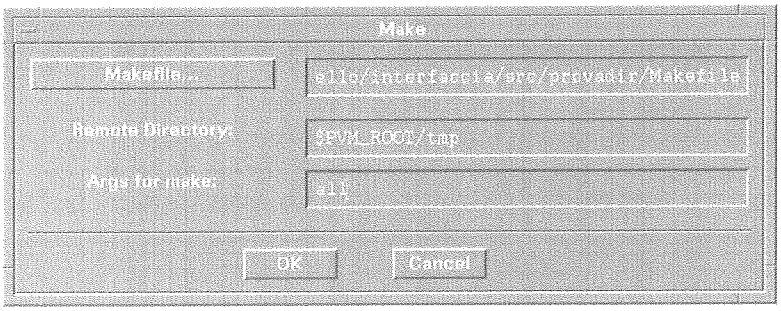


Figura 2: Finestra di dialogo per la compilazione

pilazione ed avviarla richiamando la voce `Make` del menu `Apps` della finestra WAN. Viene mostrata la finestra di dialogo di Figura 2.

Utilizzando il dialog box è possibile specificare alcuni parametri per la compilazione:

Makefile: path completo del `Makefile` da utilizzare per la compilazione. La scelta di un `Makefile` specifica anche, in maniera implicita, la directory *top level*. Il `Makefile` può essere specificato servendosi di un file selection dialog richiamabile con il bottone **Makefile**;

Remote Directory: directory remota da utilizzare per la memorizzazione temporanea dei file sorgenti da compilare. La versione attuale di WAMM usa la directory `$PVM_ROOT/tmp` e non è possibile specificarne una diversa;

Args for make: argomenti da passare al comando `make`. Questa caratteristica permette, ad esempio, di specificare target particolari (`clean`, `install`, ecc.) o di utilizzare un `Makefile` differente da quello standard (`-f my-makefile`, ecc.).

Selezionando il bottone **OK**, `wamm` esegue le operazioni descritte di seguito:

- tutti i file sorgenti sono raggruppati in un unico file, nel formato standard `tar` utilizzato sulle macchine UNIX;
- il file prodotto è compresso usando il comando `compress`;
- su ogni nodo selezionato viene avviato il programma `PVMMaker`; ad ogni `PVMMaker` viene inviato il file compresso.

Le operazioni rimanenti sono gestite in parallelo dai `PVMMaker`, ciascuno sul proprio nodo. Ogni `PVMMaker` riceve il file compresso, estrae i file sorgenti

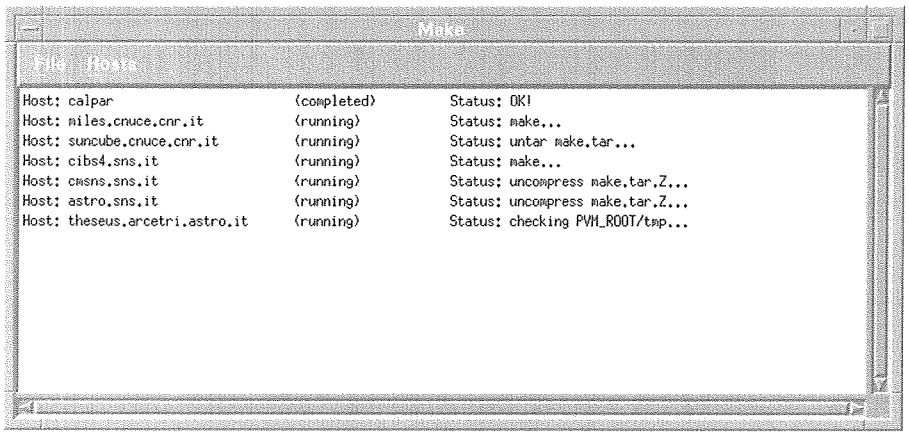


Figura 3: Finestra per il controllo della compilazione dei programmi

dalla directory temporanea `$PVM_ROOT/tmp` e richiama il comando `make` con gli eventuali parametri specificati nella finestra di dialogo (fig. 2).

La directory `$PVM_ROOT/tmp` viene creata in maniera automatica da ogni `PVMMaker`; se essa già esiste, il suo contenuto viene cancellato *prima* della compilazione. I file vengono mantenuti nella directory fino alla compilazione successiva; in questo modo è possibile collegarsi all'host, modificare i sorgenti (se necessario) e avviare manualmente una nuova compilazione.

Ogni `PVMMaker` informa l'interfaccia di tutte le operazioni compiute. I messaggi ricevuti vengono stampati nella finestra di controllo mostrata in Figura 3. La finestra mostra la lista dei processi in compilazione remota, sia essa in corso che terminata. Tale finestra, automaticamente aperta quando si avvia la compilazione, può essere aperta utilizzando la voce `Make` del menu `Windows` della finestra WAN. Su ogni riga vengono mostrati: il nome del nodo, lo stato (**running** oppure **completed**) e l'operazione specifica in corso (es. estrazione dei sorgenti, compilazione, ecc.). Per rimuovere dalla finestra le righe relative a compilazioni già terminate è sufficiente eseguire il comando `Cleanup` del menu `File`.

Nell'esempio riportato in Figura 3, la compilazione è stata avviata su sette macchine. Per ciascuna si può individuare il passo in corso nel momento in cui è stata "fissata" l'immagine. Per esempio, `astro.sns.it` ha ricevuto la propria copia della directory e la sta espandendo, mentre `calpar` ha già concluso con successo la compilazione.

È possibile selezionare uno o più righe e, per queste, aprire altrettante finestre che mostrano l'output del comando `make` (menu `Hosts`, comando `Stdout/Stderr`); una finestra di output può anche essere aperta agendo con

un doppio click sulla corrispondente riga nella finestra Make.

Si può interrompere l'esecuzione del comando `make` in qualsiasi momento, usando il comando `Stop` del menu `File`. È opportuno notare che l'eventuale fallimento della compilazione su un nodo non influenza in alcun modo l'attività in corso sugli altri.

Se la compilazione termina con successo, il `Makefile` usato per la sua esecuzione può copiare i file eseguibili prodotti all'interno della directory

```
$PVM_ROOT/bin/$PVM_ARCH
```

usata dal PVM come "deposito" per i programmi da eseguire. Questa operazione sarà eseguita su ciascun nodo. Il meccanismo di compilazione remota può anche essere utilizzato per distribuire uno o più file (eseguibili, librerie, ecc.) su più nodi: è sufficiente inserire nel `Makefile` una serie di comandi che si occupino di spostare tali file nelle directory desiderate, senza che venga richiamato alcun compilatore.

5.1.3 Esecuzione di un'applicazione

WAMM offre la possibilità di eseguire un'applicazione PVM, scritta in linguaggio C, sotto il controllo di un monitor. L'utente, in tal caso, può sia decidere di eseguire l'applicazione in modo da rilevare tutte le occorrenze di comunicazione PVM e generare così un file di traccia che può essere visualizzato in maniera grafica dagli strumenti `PVaniM` [TSS96a] e `ParaGraph` [HE91], che di campionare periodicamente lo stato dell'applicazione e della macchina virtuale e di vederlo in modalità grafica in un'apposita finestra di WAMM. Per poter sfruttare queste caratteristiche è necessario che il programma da eseguire, venga instrumentato. Va osservato che una volta che l'applicazione è stata instrumentata, può essere rieseguita senza il controllo del monitor senza la necessità di ricompilare il programma.

5.1.4 Instrumentazione di un programma

L'instrumentazione del programma avviene mediante l'utilizzo della libreria instrumentata `wammtracelib.a` che permette la generazione di eventi che segnalano l'avvenuta esecuzione di routine PVM.

In Tabella 1 sono mostrate le funzioni della libreria di PVM che sono state instrumentate dalle corrispondenti funzioni della libreria di WAMM.

In Tabella 2, vengono invece mostrate le routine che sono state aggiunte per offrire nuove funzionalità.

Per poter utilizzare la libreria instrumentata e quindi sfruttarne le capacità di monitoraggio è necessario rispettare alcuni vincoli:

Funzioni PVM	Funzioni WAMM	Descrizione delle routine instrumentate
pvm_exit	wamm_exit	Determina l'uscita di un task da PVM, quando il tracing è attivo provvede alla chiusura ed al strasferimento del file di traccia locale.
pvm_mcast	wamm_mcast	Effettua l'invio in multicast di un messaggio PVM e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_mytid	wamm_mytid	Individua il tipo di monitoraggio da effettuare ed inizializza le strutture dati necessarie per mantenere le informazioni sul monitoraggio. Restituisce il TID assegnato al task PVM.
pvm_nrecv	wamm_nrecv	Effettua una receive non bloccante, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_parent	wamm_parent	Restituisce il TID del task che l'ha generato, nel caso in cui esso sia WAMM, restituisce il valore PvmNoParent.
pvm_precv	wamm_precv	Effettua una receive ad una fase, il messaggio ricevuto viene copiato nello spazio utente, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_psend	wamm_psend	Effettua nello stesso tempo l'impacchettamento e la spedizione di un messaggio e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_recv	wamm_recv	Effettua una receive bloccante a due fasi, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_send	wamm_send	Effettua una send e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.
pvm_trecv	wamm_trecv	Effettua una receive con timeout, fornisce il supporto allo *steering* e provvede ad effettuare il tipo di monitoraggio richiesto dall'utente.

Tabella 1: Elenco delle funzioni PVM che sono state instrumentate.

- Prima di iniziare la fase di monitoraggio, ogni task applicazione deve invocare la funzione `pvm_mytid()`, ciò è necessario perchè la routine che la rimpiazza (`wamm_mytid()`) contiene il codice che inizializza le strutture dati per effettuare il monitoraggio stesso;

- Ogni task per concludere la sua esecuzione deve invocare la `pvm_exit()`, in questo modo la routine che la rimpiazza (`wamm_exit()`) esegue tutte le operazioni necessarie per la conclusione del monitoraggio, compreso, quando necessario, l'invio del file di traccia prodotto dal task PVM.

- Ogni task affinché possa ricevere messaggi di *steering* deve utilizzare una funzione receive con *wild-card* (con parametri (-1, -1)), in questo

Funzioni Aggiuntive	Descrizione delle routine
wammOL_print	Permette la visualizzazione dei messaggi provenienti dai task oggetto del monitoraggio sulla finestra text_output dell'interfaccia.
wamm_close	Permette di chiudere una fase di monitoraggio, consentendo che i task continuino l'elaborazione senza più raccogliere dati di traccia.
wamm_open	Consente di aprire una nuova fase di monitoraggio nel caso in cui sia stata precedentemente eseguita una wamm_close.
wamm_print	Permette di rilevare eventi definiti dall'utente.
wamm_scandouble	Consente di effettuare operazioni di input verso un task, con valori di tipo double.
wamm_scanfloat	Consente di effettuare operazioni di input verso un task, con valori di tipo float.
wamm_scanint	Consente di effettuare operazioni di input verso un task, con valori di tipo int.
wamm_scanlong	Consente di effettuare operazioni di input verso un task, con valori di tipo long.
wamm_scanstring	Consente di effettuare operazioni di input verso un task, con valori di tipo string.
wamm_setsampling	Permette di cambiare la frequenza di campionamento.

Tabella 2: Elenco delle funzioni aggiunte alla libreria fornita da WAMM.

modo il task può ricevere un messagio con qualsiasi TAG da qualsiasi task.

Oltre a ciò, è necessario che l'applicazione includa lo header file `wammOL.h`. Questo deve essere messo immediatamente sotto la specifica dello header file di PVM (`pvm3.h`). Il file `wammOL.h` contiene le macro che sostituiscono le routine PVM con chiamate alle routine della libreria di WAMM (`wammtracelib.a`).
Per sfruttare alcune caratteristiche della libreria di WAMM, è necessario aggiungere al codice dell'applicazione l'invocazione esplicita alla routine desiderata.
Ad esempio, WAMM prevede che come periodo di campionamento possa essere utilizzato quello di default (5 secondi). Se l'utente volesse cambiare la frequenza di campionamento potrebbe procedere in più modi: inserendo immediatamente prima della `pvm_mytid()` l'invocazione alla routine `wamm_setsampling(n,t)`, in questo modo il campionamento verrà effettuato ogni `n` invocazioni a funzioni PVM, se `n` è maggiore di uno, oppure ogni `t` secondi, altrimenti; alternativamente l'utente, se ha utilizzato all'interno della sua applicazione delle *receive* con *wild-card*, può agire a tempo di esecuzione su uno *slider* posto nella finestra di monitoring usata dall'interfaccia.
L'invocazione alle routine `wamm_open()` e `wamm_close()` permette, invece, di

monitorare solo alcune fasi dell'esecuzione dell'applicazione. Questa caratteristica è senz'altro molto interessante perchè nel caso in cui si è interessati solo ad osservare solo piccole frazioni dell'esecuzione, evita che vengano generate grosse quantità di informazioni.
— Altre routine utili sono le `wamm_scan*()` (dove l'asterisco può essere sostituito da `int`, `string`, ecc.) che possono essere aggiunte nel programma per poter effettuare input di differente tipo; in tal caso, durante l'esecuzione verrà aperta un'apposita finestra, come mostrato in Figura 4, indicante il tipo di input richiesto ed il TID del task richiedente.
La routine `wamm_printOL()`, invece, che ridirige l'output del programma ver-

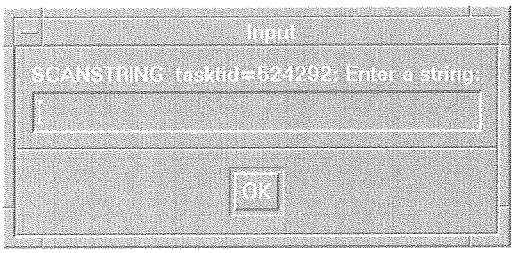


Figura 4: Finestra di controllo per eseguire operazioni di input verso i processi

so l'interfaccia, può essere utilizzata quando non si è interessati ad analizzare tutto l'output prodotto da un task (cosa ottenibile tramite WAMM, mediante l'attivazione dell'opzione **Output**), ma solo un suo sottoinsieme.

Un esempio di uso delle routine instrumentate è mostato in Figura 5.

Infine, mediante l'utilizzo della routine `wamm_print()` (i cui parametri seguono una formattazione simile a quella della `printf()` del C) è possibile rilevare eventi definiti dall'utente.
Una volta effettuate queste modifiche, il codice può essere compilato e per la generazione del modulo eseguibile deve essere utilizzata sia la libreria PVM `libpvm3.a` che la libreria instrumentata di WAMM `wammtracelib.a`. Scrivendo in modo opportuno il `Makefile` (assieme al codice di WAMM, viene fornito all'interno della directory `example`, il codice sorgente relativo ad un programma esempio ed il `Makefile` da utilizzare per la compilazione di applicazioni instrumentate) è possibile utilizzare WAMM per eseguire la compilazione remota del programma su tutti i nodi del metacomputer.

```
Task A: master
.....
#include "wammOL.h"
.....
pvm_mytid();
pvm_spawn ("B",...,nproc,tid);
.....
.....
wamm_print("identificatore evento")
.....
.....
pvm_exit;
```

```
Task B: master
.....
#include "wammOL.h"
.....
pvm_mytid();
wamm_close();
.....
.....
wamm_open();
.....
.....
pvm_exit;
```

Figura 5: Esempio di uso di routine PVM instrumentate.

5.1.5 Attivazione e controllo di processi

Per eseguire un programma PVM è possibile utilizzare la finestra di dialogo richiamabile per mezzo della voce `Spawn` del menu `Apps` (Fig. 6). Il dialog box permette di specificare i parametri di attivazione necessari:

- **Name**: nome del programma da eseguire;
- **Arguments**: argomenti da passare al programma;
- **Copies**: numero di copie del programma di cui si vuole attivare l'esecuzione;
- **Where**: permette di scegliere dove eseguire le varie copie del programma. Le scelte possibili sono:
  - **Auto**: gli host su cui eseguire le varie copie del programma vengono scelti in maniera automatica da PVM, questa opzione corrisponde al flag `PvmTaskDefault` di PVM;
  - **Arch**: è necessario specificare una architettura su cui eseguire i processi. PVM sceglie tra le macchine aventi l'architettura indicata. Se viene attivato il *checkbutton* != (corrispondente al flag PVM `PvmHostCompl`), PVM sceglie una architettura diversa da quella specificata. La voce **Arch** corrisponde al flag `PvmTaskArch` di PVM;
  - **Host**: è necessario specificare il nodo su cui eseguire i processi. Anche in questo caso se viene selezionato !=, PVM sceglie un host diverso da quello specificato. La voce **Host** corrisponde al flag `PvmTaskHost` di PVM;
- **Output**: questo parametro permette di ridirigere l'output dei processi eseguiti verso l'interfaccia. In questo modo è possibile avere l'output dei vari processi in finestre separate. Questa funzionalità viene implementata con la modifica del valore della variabile `PvmOutputTid` di PVM;
- **Debug**: permette di eseguire il processo specificato sotto il controllo di un debugger. Corrisponde al flag `PvmTaskDebug` di PVM;
- **MPP**: avvia i processi sul *front-end* di una macchina multiprocessore anziché sui suoi nodi. Corrisponde al flag `PvmMppFront` di PVM;

- **Sampling**: in questo modo, i processi che vengono attivati, spediranno periodicamente all'interfaccia informazioni descriventi la propria esecuzione, mentre i processi `PVMTasker`, invieranno all'interfaccia le informazioni riguardanti il carico sulle macchine;
- **Tracing**: in questo modo i processi vengono avvertiti che dovranno effettuare la rilevazione degli eventi;
- **Tracefile name**: permette all'utente di scegliere un nome per il file di traccia che verrà generato. Il nome di default è `wammfile.uid.n.anim`, dove `uid` indica il numero identificativo dell'utente, `n` serve a distinguere i vari file di traccia relativi a differenti fasi della stessa esecuzione e `anim` contraddistingue i file di traccia che possono essere visualizzati con lo strumento `PVaniM` [TSS96a]. Il file di traccia così generato verrà collocato, per default, nella directory `/temp`. Tramite il popup menu è possibile selezionare la voce **filename**, in quel caso apparirà una finestra che consentirà di scegliere la directory che dovrà contenere il file di traccia e di specificare il prefisso che si vuole assegnare al file generato.

Una volta impostati tutti i parametri, selezionando il bottone **Spawn!**, viene attivata l'esecuzione del programma specificato. Per ogni copia del proces-

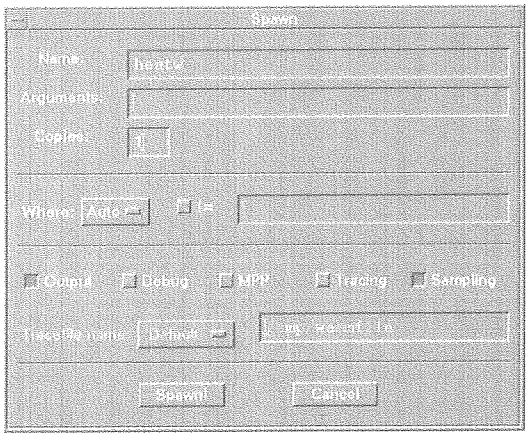


Figura 6: Finestra di controllo per lo Spawn

so avviata con successo verranno mostrati, nell'area messaggi della finestra WAN:

- il nome del processo;
- il TID;

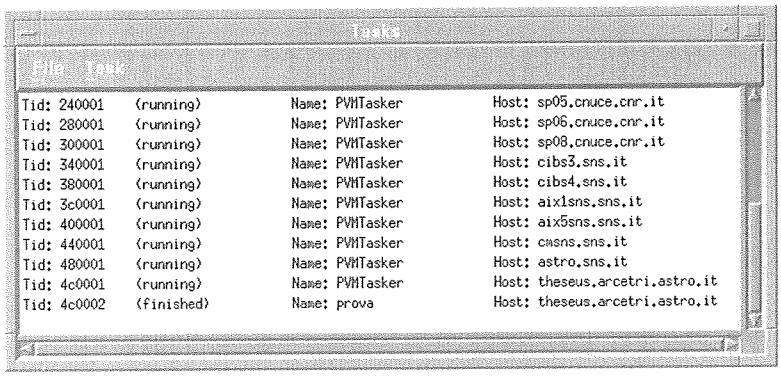


Figura 7: Finestra di controllo per i tasks

- il nome del nodo su cui è stato mandato in esecuzione.

In caso di fallimento, verrà riportato un messaggio di errore.

I task PVM in esecuzione sulla macchina virtuale possono essere controllati utilizzando la finestra mostrata in Figura 7, richiamabile con il comando `Tasks` del menu `Windows`. La finestra contiene una lista dei processi PVM. Per ogni processo vengono mostrati: il TID, lo stato (**running** oppure **finished**), il nome e l'indirizzo della macchina su cui viene eseguito. Per default, la lista dei processi non comprende i task di servizio (`PVMTasker`, `PVMHoster` e `PVMMaker`); possono essere mostrati/nascosti cambiando lo stato della voce `Show system tasks` del menu `File`. Lo stato dei task viene aggiornato automaticamente da `wamm` sulla base delle informazioni provenienti dai processi `PVMTasker` in esecuzione sui nodi della macchina virtuale. Per rimuovere le righe relative a task già terminati occorre richiamare `Remove old tasks` nello stesso menu.

Agendo con un doppio click su una riga, è possibile aprire una finestra che ne mostra l'output prodotto dal corrispondente processo; questa operazione può essere anche svolta su un gruppo di processi, selezionando le righe corrispondenti e richiamando il comando `Output` del menu `Task`. Questa funzionalità è disponibile solo per i processi attivati con l'opzione **Output** (cfr. fig. 6).

La finestra consente anche di terminare uno o più processi: dopo aver selezionato le righe corrispondenti ai processi interessati, deve essere eseguito il comando `Kill` del menu `Task`. In maniera analoga è possibile inviare segnali ai processi selezionati (`Signal`, menu `Task`).

Una volta attivata l'esecuzione dell'applicazione, nel caso in cui sia stata impostata l'opzione **Tracing** e non quella **Sampling**, al termine dell'esecuzione dell'applicazione, verrà effettuata, in automatico, la raccolta dei file

di traccia prodotti e verrà eseguita la fase di elaborazione del file per effettuare l'ordinamento dei dati di traccia; anche in questo caso l'utente verrà informato sull'esito delle operazioni.

Se l'applicazione, invece, viene attivata con l'opzione **Sampling**, apparirà una nuova finestra **Monitor**.

5.1.6 Chiusura dell'interfaccia

La chiusura del programma può avvenire in due modalità differenti, richiamabili dal menu `File` della finestra WAN:

Quit: esce dal programma, chiedendo conferma. I processi di servizio vengono terminati (`PVMTasker`, `PVMHoster`, `PVMMaker`);

Halt: esce dal programma richiamando `pvm_halt()`. La macchina virtuale viene distrutta e tutti i processi PVM in esecuzione vengono terminati. Viene chiesta conferma.

I processi di servizio controllano periodicamente lo stato del processo `wamm`: nel caso di chiusura anomala dell'interfaccia, tali processi terminano autonomamente.

5.2 Finestra Monitor

Nella finestra Monitor vengono mostrate in forma grafica tutte le informazioni raccolte periodicamente durante l'esecuzione di una applicazione PVM.
Come mostrato in Figura 8, la finestra che visualizza le informazioni relative al monitoraggio è divisa in 3 parti: nel riquadro in alto a sinistra, vengono mostrate le varie viste grafiche riportanti informazioni sulle macchine e sui task PVM; a destra, sono elencati i processi sottoposti al monitoraggio ed in basso, vengono mostrati i messaggi relativi sia ai processi che alle macchine monitorate. La parte superiore della vista grafica, mostra le informazioni riguardanti i vari host che compongono la macchina virtuale:

- *Host List*: questa vista illustra, in modo testuale, gli host che compongono la macchina virtuale ed i task allocati su ciascuno di essi. I task sono indicati con un valore intero.

- *Load Info*: questa vista fornisce informazioni sul carico di ogni host mostrando il numero medio di job nella coda di esecuzione.

- *Host Utilization*: in tale vista viene mostrata la percentuale di tempo che ogni processore trascorre in ognuno dei tre stati (computazione,

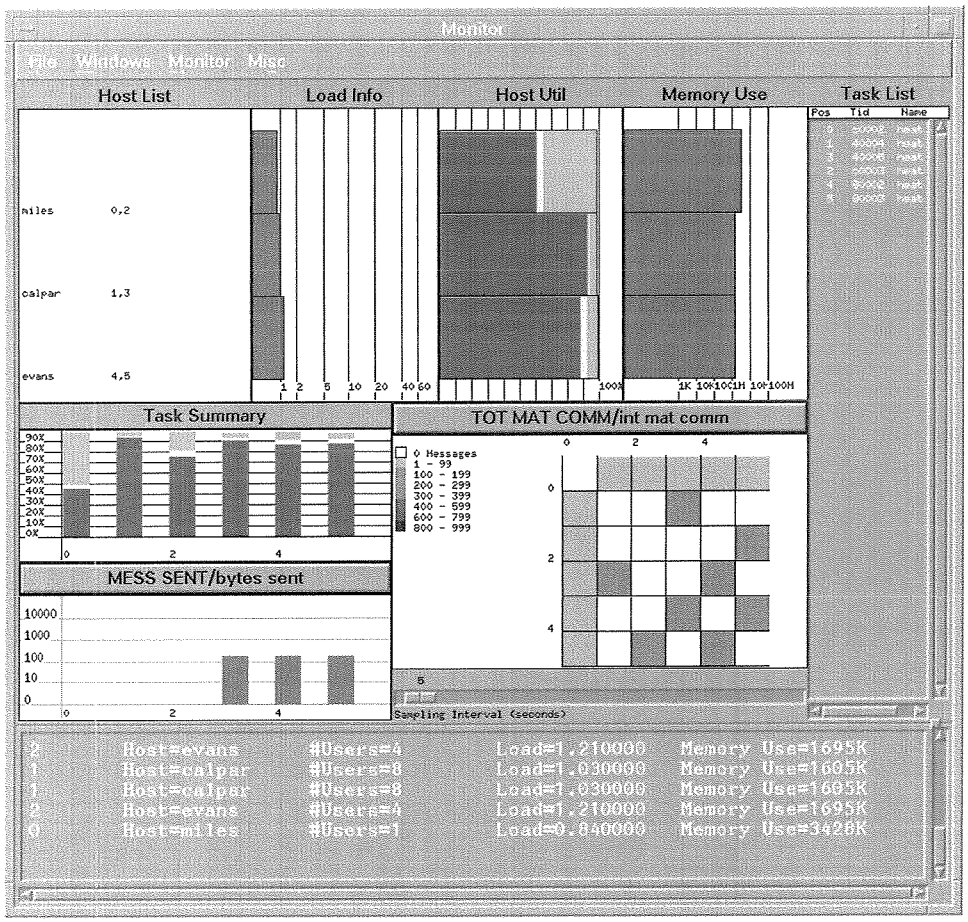


Figura 8: Finestra per la visualizzazione dei dati di trace prodotti durante l'esecuzione di un'applicazione PVM.

spedizione e ricezione di messaggi). Per far questo, ogni processore è associato ad una barra orizzontale suddivisa in segmenti colorati in accordo allo stato in cui esso si trova per l'intervallo di tempo specificato dalla loro lunghezza. I colori verde, giallo e rosso indicano, rispettivamente, computazione, invio di un messaggio e attesa per la ricezione di un messaggio.
La vista Host Utilizzation è analoga alla vista Gantt fornita da ParaGraph, va sottolineato che il livello di dettaglio è sicuramente inferiore, ma ha il pregio di fornire, per ogni processore, una composizione delle informazioni provenienti dall'esecuzione di tutti i task su di esso allocati.

- *Memory Use*: questa vista illustra, per ogni host, l'ammontare di memoria utilizzata dai task PVM su di esso allocati.

Le viste precedenti sono "cliccabili", in questo modo l'utente può ottenere informazioni testuali riguardanti il carico dell'host selezionato, che appariranno in basso nella finestra **Monitor**.
La parte inferiore della vista grafica mostra invece informazioni riguardanti i task PVM e la comunicazione da essi effettuata:

- *Task Summary*: questa vista presenta lo stesso tipo di informazioni fornite dalla vista Host Utilization, ma in questo caso tali informazioni sono relative ai singoli task. Confrontando le informazioni fornite dalla vista *Task Summary* con quelle fornite dalla vista *Host Utilization*, l'utente può comprendere qual'è il task maggiormente responsabile di eventuali degradi delle prestazioni;

- *MESS SENT/bytes sent*: questa vista fornisce informazioni riguardanti sia il numero dei messaggi, che il numero di byte spediti da ogni task. Tali informazioni si riferiscono esclusivamente alla comunicazione avvenuta durante il periodo di campionamento, ma in ogni caso fornisce una idea della comunicazione e permette di individuare dei comportamenti anomali;

- *TOT MAT COMM/int mat comm*: questa vista è rappresentata da una matrice le cui righe e colonne corrispondono rispettivamente ai processi mittente e destinatario. La casella corrispondente ai processori coinvolti nella comunicazione viene colorata in accordo al numero dei messaggi scambiati. Le informazioni fornite possono riguardare le comunicazioni relative al periodo di campionamento, o all'intera esecuzione. Questa vista, se viene impostata per controllare le comunicazioni nel periodo di

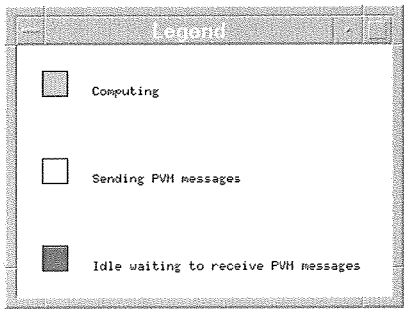


Figura 9: La vista Legend della finestra Monitor.

campionamento, è "cliccabile" e ciò permette di cambiare l'associazione tra i diversi colori ed il numero dei messaggi che essi rappresentano.

Utilizzando il comando `Legend` del menu `Monitor` è richiamabile una finestra che mostra l'associazione tra i colori utilizzati nelle varie viste e gli stati assunti dai processi e dai processori (vedi Figura 9).

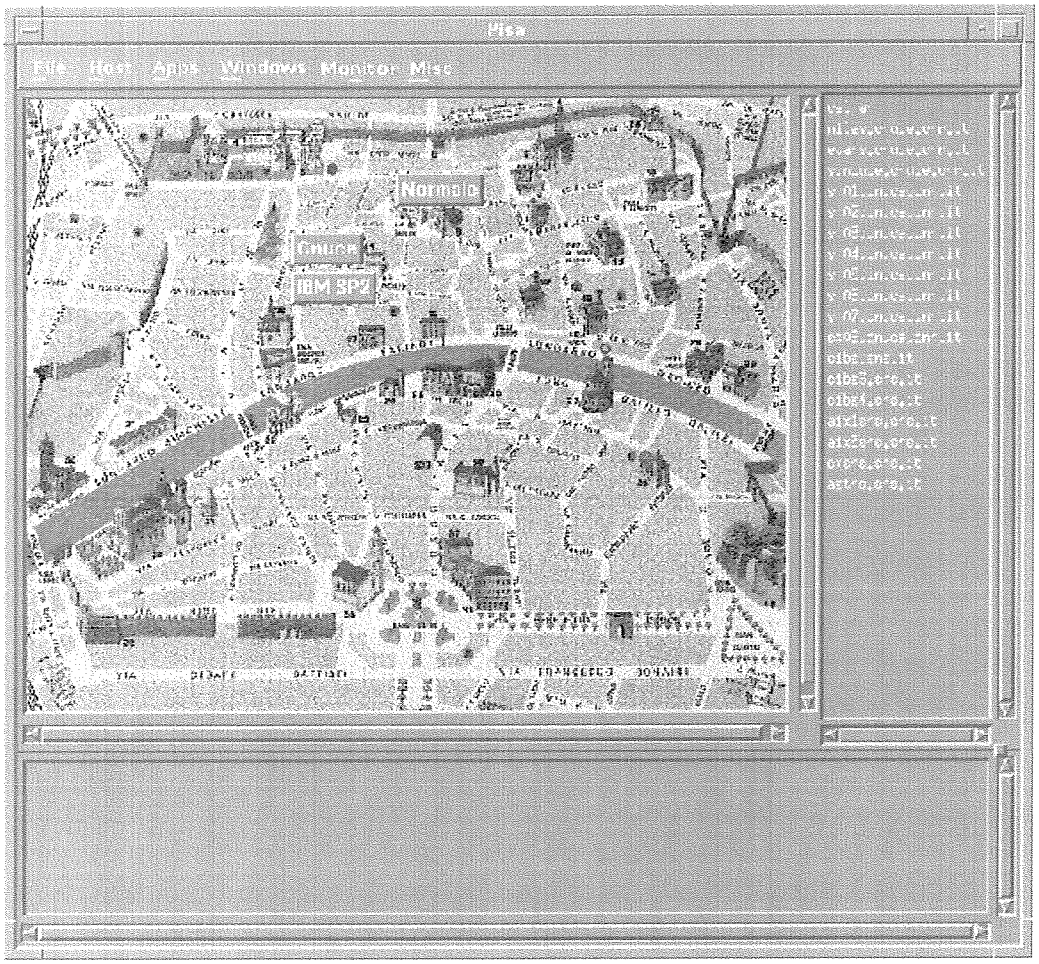
Oltre alle viste sopra descritte, nella finestra **Monitor**, è presente anche uno *slider* che permette all'utente di cambiare la frequenza di campionamento. In questo modo, l'utente può decidere il grado di intrusione dello strumento e il livello di dettaglio delle viste risultanti.

Come si puó osservare in Figura 8, nella *Task List* (riquadro a destra della finestra **Monitor**) vengono elencati i task PVM che partecipano al monitoraggio. Per ogni task vengono presentate le seguenti informazioni: il numero intero che lo identifica, il TID ed il nome del programma. Ogni elemento della lista è "cliccabile" e mostra, in una finestra separata, l'output del task selezionato.

Nel riquadro in basso della finestra, invece, vengono mostrate le informazioni riguardanti l'ingresso e l'uscita dal monitoraggio dei task PVM, le informazioni sul carico degli host ed i vari messaggi di output prodotti dai task PVM, nel caso in cui siano state utilizzate routine `wamm_printOL()`.

Il comando `Converter` del menu `Monitor` consente di attivare una finestra di selezione dei file di traccia in formato leggibile da pvanim, in modo che questi possano essere convertiti nel formato leggibile dal tool ParaGraph.

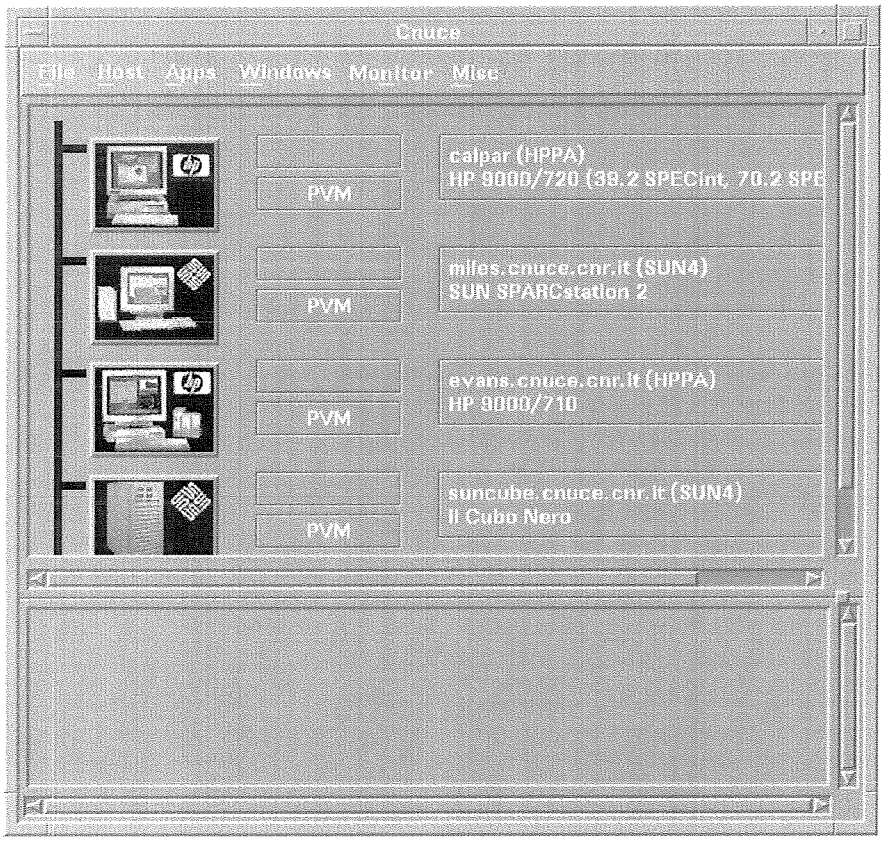
5.3 Finestra MAN



La finestra MAN ha una struttura simile alla finestra WAN. La parte in alto a sinistra contiene l'immagine specificata per mezzo della direttiva `PICT` all'interno della dichiarazione della struttura MAN. La parte in alto a destra contiene una lista di tutti gli host appartenenti alla MAN. La parte bassa della finestra contiene l'area dei messaggi.

Tutte le operazioni richiamabili dalla finestra WAN sono accessibili anche da questa finestra, con la sola limitazione che gli host visibili sono solo quelli appartenenti alla MAN.

5.4 Finestra LAN



La finestra LAN è suddivisa in due parti. La parte in alto raffigura un segmento di rete Ethernet; ad essa sono connessi gli host specificati all'interno della dichiarazione della struttura LAN. Ogni host viene rappresentato per mezzo di una icona. Se le icone non possono essere visualizzate (colori non disponibili, file non trovato, ecc.), al loro posto viene mostrata un'etichetta con il nome della macchina. Nel caso delle finestre LAN la selezione di uno o più nodi avviene premendo l'icona di ogni nodo con il tasto sinistro del mouse. Sono possibili le operazioni di selezione/deselezione di tutti i nodi per mezzo delle voci `Select All` e `Deselect All` del menu `Host`.

Alla destra di ogni icona sono presenti due flag di stato che indicano se il nodo è in PVM e se è selezionato; l'indicazione dell'appartenenza a PVM viene aggiornata automaticamente in caso di variazioni alla configurazione della macchina virtuale. A destra dei flag di stato è presente un riquadro contenente: il nome della macchina, l'architettura specificata per mezzo della direttiva `ARCH` e le informazioni specificate per mezzo della direttiva `INFO`.

La parte bassa della finestra MAN contiene l'area dei messaggi. In ta-

le area vengono mostrati i messaggi di errore restituiti da WAMM nonchè l'output dei comandi eseguiti sui nodi remoti.

Tutte le operazioni richiamabili dalla finestra WAN sono accessibili anche da questa finestra, con la sola limitazione che gli host visibili sono solo quelli appartenenti alla LAN.

5.4.1 Operazioni sui singoli nodi

Ciascuna icona (o etichetta mostrata al suo posto) ha associato un menu popup richiamabile con il tasto destro del mouse. Tale menu è suddiviso in due parti: la prima riproduce alcuni comandi presenti nella barra dei menu principale:

PVM Add: aggiunge il nodo a PVM;

PVM Remove: rimuove il nodo da PVM;

PVM Check: controlla se il nodo fa parte di PVM;

PVM Repair: "ripara" il nodo;

Ping: richiama il comando UNIX `ping` per il nodo associato e mostra l'output in una finestra separata;

Traceroute: richiama il comando UNIX `traceroute` per il nodo associato e mostra l'output in una finestra separata;

Make: avvia la compilazione di un programma sul nodo associato;

Spawn: attiva una o più copie di un processo PVM sul nodo associato.

Richiamando una operazione da un menu popup associato ad un host, implicitamente si seleziona tale macchina come *target* dell'operazione.

La seconda parte del menu popup contiene i comandi remoti specificati dall'utente nel file di configurazione. Prima di eseguire un comando su un nodo remoto, è necessario verificare che esso sia in PVM. L'output dei comandi viene visualizzato nell'area dei messaggi al *termine* della loro esecuzione.

Per quanto riguarda i programmi X11 (specificati per mezzo delle direttive `XCMD` nel file di configurazione), le finestre vengono mostrate sullo stesso display su cui sono visualizzate le finestre di `wamm`. Affinché questo sia possibile, l'interfaccia richiede automaticamente al server X in uso l'autorizzazione per l'host che deve eseguire i comandi remoti. Esistono però situazioni particolari in cui `wamm` non è in grado di abilitare autonomamente le connessioni

al server X. Ad esempio, se `wamm` è in esecuzione su una workstation e le finestre sono mostrate sulla console di un'altra workstation, l'abilitazione deve essere eseguita manualmente dall'utente sulla seconda macchina (quella su cui appaiono le finestre).

6 Risorse X

Ogni elemento grafico di un programma X11 (*widget*) è contrassegnato da un *nome*, una *classe* e un insieme di *risorse* che ne definiscono l'aspetto (ad esempio, il colore, le dimensioni, il font, ecc.). Gli elementi sono organizzati secondo una struttura ad albero: la radice è costituita da un widget la cui classe generalmente deriva il proprio nome da quello dell'applicazione.

Nel caso dell'interfaccia `wamm`, ad ogni finestra WAN, MAN o LAN è associato un proprio albero di widget; la radice è sempre di classe `Wamm` e ha per nome `wamm` nel caso della finestra WAN; per le finestre MAN o LAN viene usato il nome delle strutture corrispondenti dichiarate nel file di configurazione.

Ogni elemento grafico dell'albero è identificabile attraverso una stringa che definisce il percorso radice-nodo. Per esempio, l'area dei messaggi della finestra WAN (un widget di nome `status`) ha per path:

```
wamm.mainform.panedwin.statusSW.status
```

In molti casi è utile abbreviare il path utilizzando il simbolo *:

```
wamm*status
```

identifica lo stesso widget.

Per cambiare il valore di una risorsa di un widget si usa la forma:

```
<path widget>.<nome risorsa>: <valore>
```

Per esempio, per utilizzare il font `fixed` nell'area messaggi della finestra WAN, si può specificare:

```
wamm*status.fontlist: fixed
```

all'interno del file `.Xdefaults` presente nella home directory dell'utente. Esistono modi alternativi per specificare i valori delle risorse di un'applicazione X; per maggiori dettagli si può consultare la documentazione in linea (`man X`).

Si può ottenere l'elenco completo dei widget utilizzati da `wamm` utilizzando il programma `editres`, incluso nella distribuzione di X11. `editres` consente anche di modificare interattivamente i valori delle risorse, senza richiedere modifiche al file `.Xdefaults`.

7 Struttura dei sorgenti

Ogni funzione complessa di WAMM è realizzata da un modulo indipendente. I moduli sono strutturati in maniera omogenea: ciascuno di essi è codificato in un'unica coppia di file sorgenti:

```
modXXX.c
modXXX.h
```

dove `XXX` è il nome del modulo. Il file con suffisso `.h` contiene la dichiarazione delle funzioni esportate dal modulo; il file con suffisso `.c` contiene la loro implementazione. Ogni modulo esporta due funzioni *standard* chiamate:

```
XXXInit()
XXXEnd()
```

che devono essere richiamate, rispettivamente, per inizializzare e per rilasciare le risorse allocate dal modulo (strutture dati, ecc.). I moduli si possono suddividere in quattro gruppi:

Moduli applicativi. Sono moduli ad "alto livello" che implementano funzionalità specifiche di WAMM (es. l'attivazione dei task, il loro controllo, la compilazione):

Spawn: gestisce l'attivazione dei task PVM;

Tasks: controllo e visualizzazione dei task PVM;

Make: coordina la compilazione remota dei programmi;

Internet: implementa le funzionalità `Ping` e `Traceroute`;

Parser: esegue il parsing del file di configurazione.

Moduli grafici. Comprendono tutte le funzioni necessarie per realizzare l'interfaccia grafica del programma:

WAN: gestisce il *look&feel* della finestra WAN;

MAN: analogo al precedente per le finestre MAN;

LAN: analogo al precedente per le finestre LAN;

MONITOR: analogo al precedente per la finestra Monitor;

Redraw: gestisce l'aggiornamento delle viste della finestra Monitor.

Moduli di rete. Sono moduli di controllo che fanno da interfaccia tra l'applicazione e la macchina virtuale sottostante:

PVM: esegue le operazioni di configurazione della macchina virtuale (aggiunta, rimozione, controllo, riparazione di nodi) e raccoglie le informazioni provenienti dalle macchine remote;

NetworkObj: contiene tutte le informazioni sulla struttura della rete, così come specifificata nel file di configurazione.

Moduli di gestione dei file di traccia. Sono moduli che effettuano operazioni sui file di traccia prodotti durante l'esecuzione in modalità Tracing di un'applicazione.

dualsync: effettua la datazione e l'ordinamento degli eventi di traccia;

makepg e converter effettuano la conversione del file di traccia dal formato leggibile da `PVaniM` al formato leggibile da `ParaGraph`.

L'inizializzazione di tutti i moduli, tranne quelli necessari per il monitoraggio, viene eseguita dalla funzione `main` (contenuta in `main.c`) al momento dell'attivazione di `wamm`. I moduli necessari al monitoraggio vengono inizializzati solo al momento del loro utilizzo.