Internet of Threads

Renzo Davoli

Alma Mater Studiorum Università di Bologna, Dipartimento di Scienze dell'Informazione

Abstract. I protocolli di TCP/IP ed il supporto dei sistemi operativi alle reti sono stati disegnati sulla base dei concetti e delle esigenze di networking degli anni 60/70.
In particolare il contributo del lavoro di ricerca descritto in questo articolo analizza come superare due limitazioni progettuali: la presenza di un solo stack TCP/IP nel sistema o nell'ambiente virtuale e l'identificazione delle interfacce (reali o virtuali) come entità indirizzabili.
Questi vincoli possano essere eliminati mantenendo la compatibilità con i protocolli e le applicazioni esistenti, così ogni singolo processo o thread può diventare nodo di rete con un proprio indirizzo di rete e proprie specificità nella comunicazione. Questa nuova prospettiva crea nuove applicazioni e semplifica la struttura software in molteplici casi di uso comune.

1. Introduzione

Cosa collega la rete? Sicuramente la ricerca in questi ultimi anni ha messo in discussione quali siano i *nodi* di una rete. La rete Internet e i protocolli TCP/IP sono nati attorno all'idea che le entità da collegare fossero i computer e che le entità indirizzabili fossero le singole interfacce di rete.

Con l'evoluzione delle reti e, soprattutto, dei servizi di Internet è venuta meno la centralità del singolo computer ed è apparso sempre più innaturale l'indirizzamento delle interfacce come nodi della comunicazione, come dimostrato dai seguenti aspetti:

- nei sistemi ad alta affidabilità [6], vengono assegnati alle interfacce numerosi indirizzi IP suddividendo i servizi fra di essi, per poter far migrare i servizi in caso di malfunzionamento di un nodo riassegnando gli indirizzi ad altri elaboratori;
- i sistemi di virtualizzazione quali macchine virtuali [10] o container [8] creano interfacce virtuali alle quali assegnare indirizzi;
- nei sistemi operativi multiuser risulta molto complesso assegnare indirizzi di rete, QoS differenti a specifici utenti e servizi perché occorre applicare filtri [7] al traffico smistato dall'unico *stack* del sistema. Spesso la scelta più lineare è quella di attivare macchine virtuali o container che generino interfacce virtuali;
- l'utente rimane vincolato dalle scelte di networking dell'amministratore di sistema: senza riconfigurare il sistema appare arduo poter, per esempio, eseguire più browser che funzionino con indirizzi IP diversi di altrettante VPN distinte. Anche in questo caso l'utente può attivare macchine virtuali (QEMU/KVM/User-Mode Linux/VirtualBox) e usare reti virtuali (VDE, Virtual Distributed Ethernet).

Questo articolo presenta una proposta alternativa: l'introduzione della possibilità di avere come nodi di rete singoli processi o gruppi di processi. Il nome scelto per questo concetto è *Internet of Threads*[1] che indica bene il cambiamento di prospettiva come ulteriore evoluzione dei concetti di *Internet of Things* [5] e di *Internet of Services* [9]. La proposta è già corredata di un insieme di strumenti software che costituiscono un *proof-of-concept* operativo di questo cambiamento di prospettiva

[1] Ringrazio Pietro Galliani della University of Amsterdam per avermi suggerito un termine così calzante.

Conferenza GARR_11 Selected Papers. Da 20 anni nel futuro. Bologna 8-10 novembre 2011
ISBN: 978-88-905077-3-1 © 2012 Associazione Consortium GARR

sulla rete.

Il laboratorio internazionale di ricerca sulla virtualità *VirtualSquare* [2, 4] ha realizzato molteplici strumenti, come:

- VDE: la rete Ethernet virtuale e distribuita [1], che consente di connettere in una rete locale macchine virtuali (e altre entità virtuali) degli utenti in esecuzione sullo stesso elaboratore o su sistemi fisici diversi anche se geograficamente distribuiti;
- LWIPv6: uno stack ibrido LWIPv6, IPv6 retrocompatibile con IPv4, interamente implementato come una libreria modulare. Ogni applicazione che utilizza LWIPv6 accede a una rete TCP/IP con propri indirizzi e proprio routing. LWIPv6 supporta autoconfigurazione, DHCP (server e client), NAT e *slirp* (anche IPv6);
- View-OS/umnet/umnetlwipv6: View-OS è un progetto di macchine virtuali parziali che consente di virtualizzare alcune funzionalità del sistema host (per esempio, *umfuse* consente *mount* virtuali di parti del filesystem, *umdev* crea *device* virtuali). Umnet è il modulo di virtualizzazione della rete e il sottomodulo umnetlwipv6 consente di usare la libreria lwipv6 per usare reti virtuali. Con View-OS è possibile usare le applicazioni esistenti in contesti di rete virtuale, per esempio essa consente di attivare un browser collegato ad una VPN mentre il resto del sistema usa la rete locale.

Questo cambiamento di prospettiva consente una estrema flessibilità nella migrazione dei servizi. Una rete ethernet virtuale forma un cloud naturale per le applicazioni direttamente connesse come *nodi* di rete: una volta collegati alla rete virtuale l'applicazione che risponde ad un determinato indirizzo può essere ovunque nella rete virtuale (e a maggior ragione in quella fisica sottostante).

2. Definizione

Con l'Internet of Threads (IoTh), si perde l'idea dello stack unico per sistema (o per container) e lo stack diventa invece una libreria di implementazione di protocolli ai vari livelli del modello. Considerazioni di efficienza e di affidabilità del sistema possono guidare la scelta verso librerie del *kernel* o come librerie utente, entrambe le scelte sono possibili. Il cambiamento fondamentale sta nel fatto che ogni utente può attivare molteplici stack di rete per le proprie applicazioni (processi o threads) e decidere a quali reti collegare ognuna di esse, quindi quali indirizzi e quale routing associare ad ogni applicazione.

Anche il ruolo della rete *Data-Link* viene ridefinito in IoTh (*fig. 1*). Le applicazioni non hanno interfacce fisiche da collegare ad apparati di rete. Per questo vengono create reti virtuali che sono in grado di fornire un collegamento alle macchine virtuali o alle applicazioni IoTh, ma anche di essere interconnesse in modo trasparente a reti fisiche mediante instradamento del traffico o anche in modalità *bridging*. Per tale motivo, le reti virtuali devono essere coerenti con i protocolli di Data-Link comunemente usati: questo è infatti il senso e lo scopo di Virtual Distributed Ethernet. Le applicazioni in modalità IoTh possono così interagire con qualsiasi altro nodo dell'Internet, sia esso sistema reale, virtuale, oggetto (Thing) o altra applicazione IoTh. È anche possibile usare reti VDE per creare intranet distribuite o semplicemente come meccanismo di Inter Process Communication (IPC).

In quest'ottica deve essere anche possibile creare applicazioni che possano operare da gateway fra stack diversi. Ciò ha messo in luce una limitazione strutturale dell'interfaccia

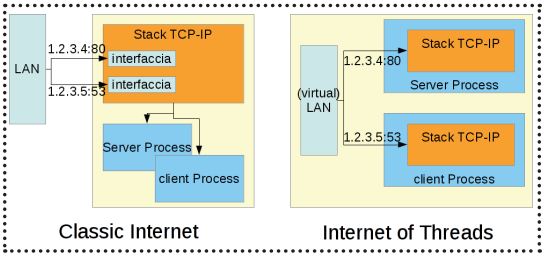

Fig. 1 Ruolo del livello Data-Link per Internet e IoTh

di programmazione Berkeley Sockets, che implicitamente assume che ci sia un solo stack di rete nel sistema. È stato necessario estendere la API per consentire l'utilizzo di più stack contemporaneamente [3]. Questa estensione tra l'altro consente:

- di poter operare contemporaneamente con stack con caratteristiche diverse: ad esempio, politiche di ritrasmissione, ampiezza delle finestre TCP etc;
- la compresenza di stack di produzione e sperimentali. Oggi è comune avere una sola implementazione dello stack anche quando vengono definite diverse reti virtuali, come per esempio tramite i containers [8]. Con una Internet of Threads è possibile far convivere al contempo uno stack affidabile per la produzione e il controllo degli esperimenti mentre si opera anche con uno stack sperimentale per valutarne per esempio correttezza, performance ed affidabilità. Senza Internet of Threads l'uso di uno stack sperimentale potrebbe rendere inaccessibile un nodo remoto in caso di errore.

```
$ # Start the first browser
$ firefox &
$ # Start a local plug for a remote slirp process
$ dpipe vde_plug /tmp/plug[] = ssh us.mydomain.org slirpvde6 -D - &
$ # Start a View-OS virtual machine
$ umview -p umfuse -p umnet bash
rd235 2.6.37-utrace-vm GNU/Linux/View-OS 17473 0
VM$ # Start a new stack
VM$ mount -t umnetlwipv6 -o vd0=/tmp/plug none /dev/net/default
VM$ # Virtualize the resolv.conf file (for DNS resolution)
VM$ mount -t umfuseramfile -o ghost /etc/resolv.conf /etc/resolv.conf
VM$ ip link set vd0 up
VM$ # configure the net by dhcp
VM$ /sbin/udhcpc -i vd0 -q -s ~/etc/udhcpc/default.script
VM$ # start a browser (choose or create nother profile)
VM$ firefox -ProfileManager -no-remote
```

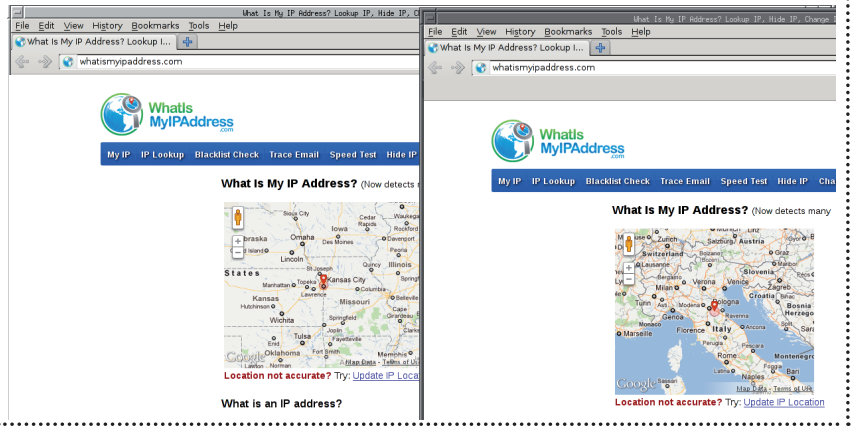

Fig. 2 Due browser della stessa sessione che usano stack TCP-IP diversi. Si noti che quello a destra appare a Bologna (quello in funzione sulla rete reale), mentre il secondo collegato ad una VPN risulta a Kansas City.

3. Alcuni esempi pratici

Come è stato già detto nell'introduzione, VirtualSquare ha realizzato strumenti che possono essere utilizzati per fare esperimenti con l'Internet of Threads. In un sistema GNU-Linux, distribuzione Debian, molti strumenti possono essere installati direttamente come pacchetti ufficiali anche se per poter vedere le caratteristiche più avanzate è necessario trasferire, compilare ed installare le ultime versioni disponibili sul sito Sourceforge[2]. Tutto il software realizzato da VirtualSquare è stato rilasciato con licenze di Software Libero.

Il primo esempio mostra come lanciare due browser con indirizzi IP differenti. Un browser usa gli indirizzi della rete reale, mentre il secondo browser usa gli indirizzi di una rete virtuale. Occorre far notare che queste funzionalità necessitano di un certo numero di azioni perché IoTh non è prevista dai programmi (Firefox/Iceweasel in questo caso). In futuro la scelta della rete potrebbe es-

[2] La sintassi degli esempi è relativa all'ultima versione sperimentale.

sere effettuata in modo simile alla scelta delle stampanti. Un utilizzo pratico di questo esempio consiste nel poter usare un browser su una connessione protetta per poter svelare dati personali (ma non per esempio la propria locazione geografica), mentre un secondo browser serve per le connessioni locali, per ricevere informazioni ambientali quali una guida museale o la lista dei ristoranti del luogo dove ci troviamo, ma da quest'ultimo browser non sono accessibili informazioni personali. In questo esempio di applicazione quindi la privacy dell'utente viene mantenuta da una sorta di principio di indeterminazione: i servizi che possono conoscere la locazione non conoscono l'identità e viceversa.

Il risultato è quello mostrato in figura 2. Non si tratta di un servizio proxy. Un proxy opera su specifici protocolli, IoTh estende la rete remota. L'esperimento è stato fatto con un browser perché è un'applicazione comune, ma si sarebbe potuto utilizzare qualsiasi applicazione di rete, client, server o peer-to-peer. Mediante VDE si possono creare servizi distribuiti anche non basati su IPv4 o IPv6. VDE è compatibile con qualsiasi protocollo che possa operare su di una rete Ethernet.

Il secondo esempio mostra come un programmatore possa con semplicità implementare un server TCP secondo il paradigma Internet of Threads usando LWIPv6, sia che si tratti di un server Web, FTP o di un Mail Tran-

```
#include <lwipv6.h>
/* other header file includes */

void handler(void *arg)
{
  long fd = (long) arg;
  /* manage the connection using lwip_{read,write,recv,send,poll,select,...} */
  lwip_close(fd);
}

int main(int argc, char *argv[])
{
  struct stack *stack=lwip_add_stack(0);
  struct netif *vdeif=lwip_add_vdeif(stack, argv[1], NETIF_STD_FLAGS);
  struct ifreq ifr;
  int server;
  int port=80;
  struct sockaddr_in serv_addr;
  /* parse argv */

  lwip_ifup_flags(vdeif, NETIF_FLAG_DHCP);
  sleep(2);
  ifr.ifr_addr.sa_family = AF_INET;
  lwip_ioctl(fd, SIOCGIFADDR, &ifr);
  printf("%s\n", inet_ntoa(((struct sockaddr_in *)&ifr.ifr_addr)->sin_addr));

  server=lwip_msocket(stack, PF_INET, SOCK_STREAM, 0);
  serv_addr.sin_family      = AF_INET;
  serv_addr.sin_addr.s_addr = INADDR_ANY;
  serv_addr.sin_port        = htons(port);
  lwip_bind(server, (struct sockaddr *)(&serv_addr),sizeof(serv_addr));

  lwip_listen(server, 10);

  while (1) {
    long connected = lwip_accept(server, NULL, NULL);
    lwip_thread_new(handler,(void *)connected);
  }
}
```

Fig. 3 Struttura del codice sorgente di un server TCP con stack LWIPv6 embedded

sport Agent. Queste applicazioni con la rete *a bordo* (*embedded*) si possono chiamare *Net Appliance*. Sono autonome e, se hanno indirizzi preassegnati (MAC e IP), possono essere spente in un punto di una rete virtuale e attivate in un altro punto mantenendo il proprio indirizzo IP. In questo modo la migrazione di un *server deamon* diventa semplice come terminare un processo su un sistema e attivarlo su di un altro.

L'esempio in figura 3 mostra la struttura del codice di un server che acquisisce l'indirizzo via DHCP. Per migliorare la leggibilità delle parti rilevanti del codice sono state omesse la gestione degli errori e l'elaborazione relativa ad ogni connessione TCP. Anche la stampa dell'indirizzo assegnato via DHCP (utile per provare il server) viene ottenuto dopo una *sleep* di due secondi e non con un ciclo di attesa come sarebbe opportuno (ma meno lineare nel codice).

4. Conclusioni

Gli esempi trattati mostrano come il concetto di Internet of Threads apra nuove possibilità per creare nuove applicazioni o per semplificare l'implementazione di servizi esistenti. La nuova definizione di nodo di rete che ricomprende anche singoli programmi (processi o thread) consente di ridisegnare, ampliandolo, il concetto stesso di networking a ogni livello: al livello del supporto da parte di sistemi operativi, dell'interfaccia di programmazione, di routing, sicurezza, QoS, etc.

Riferimenti bibliografici

[1] Davoli R., Vde: Virtual distributed ethernet. In TRIDENTCOM'05, pages 213–220, 2005

[2] Davoli R., Goldweber M., editors. Virtual Square: Users, Programmers & Developers Guide. Lulu books, 2011.

[3] Davoli R., Goldweber M., msocket: Multiple stack support for the berkeley socket api. In ACM 27th Symposium On Applied Computingi (SAC), 2012. to appear.

[4] Davoli R., Goldweber M., et al. Virtualsquare international lab on virtuality wiki. http://wiki.virtualsquare.org

[5] Gershenfeld N., Krikorian R., Cohen D., The Internet of things the principles that run the Internet are now creating a new kind of network of everyday devices, an "Internet-0.". Scientific American, 291(4):76–81, 2004

[6] Marcus E., SternH., Blueprints for High Availability: Designing Resilient Distributed Systems. John Wiley & Sons, 2003

[7] McHardy P., et al. Netfilter: firewalling, nat and packet mangling for linux http://www.netfilter.org

[8] MenageP. B., Adding generic process containers to the linux kernel. In Proc. of the Ottawa Linux Symposium, 2007

[9] Schroth C., Janner T., Web 2.0 and soa: Converging concepts enabling the Internet of services. IT Professional, 9(3):36–41, 2007

[10] Smith J.E., Nair R., The architecture of virtual machines. IEEE Computer, 38(5):32–38, May 2005

Renzo Davoli
renzo@cs.unibo.it

Renzo Davoli è una persona fortunata. Da 35 anni studia informatica e da 20 è docente e ricercatore di sistemi operativi, reti, sistemi virtuali, didattica dell'informatica. È oggi retribuito come professore associato all'Università di Bologna per fare esattamente ciò che ha sempre desiderato fare.