Reti di Calcolatori

Esercizi Svolti su Kathará

Anno Accademico: 2024/25

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del file LaTeX disponibile al seguente link:
https://github.com/00Darxk/Reti-di-Calcolatori/

Indice

1 Introduzione

Le reti di calcolatori sono complicate, comprendono vari dispositivi, tra cui computer e router. Tante interfacce e protocolli diversi, collegati attraverso interconnessioni fisiche. Realizzano delle strutture topologiche molto vaste e complesse.

Si vuole sperimentare con reti anche molto complesse, senza utilizzare dispositivi fisici. Anche se si ha a disposizione una rete reale, sarebbe difficile convincere un provider a fornire a costi non eccessivi la loro rete per effettuare esperimenti su di essa. Realizzare una rete esclusivamente per effettuare questi esperimenti allo stesso modo si rivelerebbe estremamente costoso.

Kathará è un framework basato su container per effettuare esperimenti su reti di calcolatori, è un progetto open-source su github, per cui ognuno è in grado di contribuire allo sviluppo. Permette di effettuare emulazione di rete, differente da simulazione di rete. Negli strumenti di simulazione di un sistema si vogliono riprodurre le prestazioni di un sistema reale, la sua latenza, il ritardo, gli errori, la perdita di pacchetti, etc. Non si considerano le sue funzionalità, vengono analizzati solamente le prestazioni ed i parametri specifici della rete. Invece l'emulazione mira a riprodurre accuratamente le funzionalità offerte dal sistema, senza limitazioni di prestazioni. In questo caso le prestazioni rappresentano aspetti marginali della rete.

Per emulare una rete si utilizza una macchina in funzione da host, all'interno della quale vengono eseguiti container autonomi. Essenzialmente macchine isolate, queste vengono poi collegate da fili virtuali, in modo da simulare la presenza di una rete fisica, nonostante sia effettuata su una singola macchina. In Kathará i container sono collegati tra di loro in domini di collisione virtuali, non fili virtuali, per motivi di praticità. Questo consente di collegare l'host ad ogni dominio di collisione, per poter analizzare il comportamento su tutta la rete. Utilizzando fili virtuali invece non sarebbe stato così semplice da implementare.

Ciascuno dei container viene configurato come un dispositivo, in principio sono tutti uguali, ma possono rappresentare computer, router, switch, etc. Per rappresentare tutti gli elementi di una rete realisticamente.

I container rappresentano una virtualizzazione leggera, non sono macchine virtuali, quindi viene emulato un altro sistema operativo al loro interno, ma solamente un'applicazione o una piccola parte di un sistema operativo. Sono quindi molto leggeri, con un tempo di avvio ridotto rispetto alle macchine virtuali, utilizzate generalmente per realizzare micro-servizi che godono di vita autonoma.

Per Kathará viene utilizzato il sistema Docker, il più ampiamente utilizzato a livello globale per realizzare virtualizzazioni leggere. Per realizzare un container è necessaria un'immagine, ovvero un'insieme di software e le loro librerie e binari, necessari alla loro esecuzione, statici. Sono disponibili diverse immagini per realizzare dispositivi diversi, ma in questo corso verrà utilizzata solamente l'immagine di base presente in Kathará. Tramite quest'immagine è possibile realizzare un computer, un router, un bridge ed è possibile inserire al suo interno applicazioni di tipo server, servizi web, etc. Tutti gli elementi necessari per effettuare esperimenti di rete in questo corso. Ogni container poiché rappresenta un'esecuzione isolata di un'immagine, può essere costruiro su un'immagine diverse, oppure di versione diversa dagli altri container in esecuzione, senza influire sul funzionamento di Kathará.

Un dispositivo si presenta con un terminale, su cui possono essere eseguiti comandi specifici a quel dispositivo, una memoria dedicata, un filesystem e zero o più interfacce di rete. Ogni interfaccia

di rete è connessa ad un unico dominio di collisione. Tutte le interfacce trattate rispetteranno il protocollo IEEE 802.3 per comunicare tra di loro.

Kathará dispone di tre tipi di comandi:

- "v-commands": utilizzano il carattere `v` come prefisso, e sono comandi di basso livello per configurare ed avviare un singolo dispositivo;
- "l-commands": utilizzano il carattere `l` come prefisso, e permettono di gestire un intero "lab", configurandolo ed avviandolo;
- "Global commands": sono comandi principalmente di gestione.

I v-commands sono:

- `vstart`: comando di avvio di un dispositivo;
- `vconfig`: aggiunge un file di configurazione ad un dispositivo attualmente in esecuzione;
- `vclean`: termina l'esecuzione di un dispositivo.

Gli l-commands sono:

- `lstart`: comando di avvio di un lab;
- `lconfig`: permette di effettuare operazioni di configurazione su un dispositivo di un lab già attivo;
- `lclean`: termina l'esecuzione del lab;
- `lrestart`: termina e riavvia tutti i dispositivi del lab;
- `linfo`: fornisce informazioni sul lab.

I global commands sono:

- `check`: controlla l'ambiente del sistema, per controllare che l'installazione è andata a buon fine;
- `connect`: permette di collegarsi ad una macchina di Kathará già attiva;
- `list`: mostra tutti le macchine di Kathará in esecuzione per l'utente corrente;
- `settings`: mostra e configura le opzioni di Kathará;
- `wipe`: elimina tutte le macchine di Kathará, le loro connessioni ed eventuali opzioni.

Per testare il funzionamento di Kathará dopo l'installazione è consigliabile utilizzare i seguenti comandi:

```
> kathara check
```

Per controllare il suo corretto funzionamento si può testare la creazione di una singola macchina:

```
> kathara vstart -n pc1 --eth 0:A
```

Si crea una macchina di nome `pc1`, tramite l'opzione `-n` e si connette al dominio di collisione `A` con l'opzione `--eth 0:A`, che specifica anche il tipo di connessione virtuale, in questo caso ethernet. Se non vengono sollevati errori dopo questi comandi, si può interrompere la sua esecuzione con:

```
> kathara vclean -n pc1
```

Un lab di Kathará è un insieme di dispositivi che possono essere avviati e terminati contemporaneamente. La loro struttura consiste in una directory principale del lab, sempre contenente il file `lab.conf`, dove viene descritta la topologia della rete. Se sono è necessario, sono presenti subdirectories dove vengono specificate le configurazioni per i singoli dispositivi. Inoltre per ciascun dispositivo è ipotizzabile la presenza di un file chiamato con il nome del dispositivo di tipo `.startup` dove vengono descritte le operazioni da effettuare dal dispositivo all'avvio.

Negli esercizi ed in sede d'esame non sarà richiesto di realizzare un lab, ma si dovrà analizzare il comportamento di lab preesistenti, agendo sulle reti ed in caso correggendo eventuali errori o bug.

Il file `lab.conf` descrive la topologia della rete ed i dispositivi che devono essere avviati all'avvio del lab. Contiene istruzioni di sintassi:

```
<machine>[<arg>]=<value>
```

Dove `<machine>` è il nome della macchina su cui si vuole effettuare una certa operazione, `<arg>` è il tipo di operazione da effettuare su quella macchina. Se questo argomento è un numero indica a quale interfaccia della macchina si riferisce l'assegnazione, ovvero connessione ad un certo dominio di collisione, di valore specificato dal termine `value`.

Per avere un dispositivo nel lab, questo deve essere citato nel file di configurazione. Non è possibile riferirsi ad un'interfaccia successiva, se non si è già assegnata la sua precedente. Gli unici caratteri consentiti per definire il nome di una macchina o il nome di un dominio di collisione sono caratteri alfanumerici, dove lettere maiuscole e minuscole vengono considerate uguali.

I filesystem di tutte le macchine sono indipendenti ed isolati dal filesystem della macchina host, ma sarebbe conveniente il alcuni casi poter accedere e scrivere su file nella macchina host, per salvare una serie di dati, da analizzare dopo la terminazione delle macchine. Esistono due diverse modalità per permette un'interfaccia tra i filesystem della macchina host e dei vari dispositivi. Si possono condividere file tra i due filesystem direttamente, in modo che ogni cambiamento su uno dei due sia riflesso anche nell'altro. Oppure è possibile condividere una copia del file, in modo da avere due file indipendenti contenenti le stesse informazioni, quest'ultima è certamente la più semplice, ma la meno funzionale. Su Kathará sono presenti entrambi questi approcci. Utilizza una cartella `/shared` all'interno del lab, contenuta nei filesystem di ogni dispositivo in esecuzione in quel lab per condividere direttamente un file. Questa condivisione è abilitata di default, ma è possibile modificarlo nelle impostazioni. Invece per condividere una copia di un file si possono utilizzare le subdirectories di un dispositivo, direttamente collegate alle subdirectories del lab di quello specifico dispositivo. In queste stesse subdirectories sono contenuti i file di avvio `.startup` delle singole macchine. Contengono comandi shell da essere eseguiti all'avvio, per configurare le interfacce di rete o avviare certi servizi di rete.

Per avviare un lab di Kathará bisogna aprire una Powershell, su Windows, e navigare alla directory del lab. In questa directory vanno eseguiti i vari l-commands, per avviare o terminare l'esecuzione di un lab.

Per evitare eventuali complicazioni, in questi laboratori si disabilita l'opzione per il protocollo IPv6, poiché genera complicazioni di semantica di difficile interpretazione.

2 Laboratorio del 16 Ottobre: "Two Computers"

In questa esercitazione si vuole emulare la una connessione tra due calcolatori `PC1` e `PC2`. Vengono specificate le ultime due cifre del MAC address: `0:1` e `0:2`. Viene consigliato di rappresentare la topologia della rete, descritta nel file `lab.conf`, i file di estensione `.startup` sono i comandi eseguiti dalle ciascuna macchine all'accensione. In questo laboratorio, non ci sono comandi da eseguire all'avvio. Non sono presenti neanche cartelle, per cui la configurazione di queste macchine è estremamente basilare

Il file `lab.conf` contiene all'inizio i meta dati del file:

Seguono le dichiarazioni ed assegnazioni dei computer nel laboratorio. Dove si specifica il nome della macchina, e tra parentesi quadre l'opzione da assegnare. Si indica con 0 l'interfaccia `eth0`. La stringa assegnata consiste nel dominio di collisione specificato `A`.

Senza specificare l'indirizzo, Kathará utilizza un indirizzo casuale, per cui in molti di questi laboratori gli indirizzi MAC di queste macchine verranno assegnati per renderli facilmente leggibili:

```
pc1[0]="A/00:00:00:00:00:01"
```

Con `image` viene specificata l'immagine di docker utilizzata dal calcolatore, contiene tutti gli strumenti necessari per mandare, ricevere e analizzare i pacchetti:

```
pc1[image]="kathara/base"
```

La terza riga di assegnazione consiste nella configurazione del protocollo IPv6, protocollo molto invasivo, per cui si vederebbero dati non di interesse in questo esercizio in particolare:

```
pc1[ipv6]="false"
```

Quindi si disattiva questo protocollo. Analogamente si configura la macchina 2:

```
pc2[0]="A/00:00:00:00:00:02"
pc2[image]="kathara/base"
pc2[ipv6]="false"
```

Inoltre è possibile modificare l'indirizzo IP di un calcolatore nella rete tramite il comando `ip` si accedono a tutti i comandi sulle reti. Il termine `link` indica a quale livello appartiene il comando, in questo caso il livello due di link. Si specifica quale protocollo deve essere modificato con `set dev eth0`, e si specifica cosa viene modificato, in questo caso l'indirizzo `address`:

```
prompt> ip link set dev eth0 address 00:00:00:00:01
```

Per avviare Kathará si apre una powershell all'interno della cartella del laboratorio, dopo aver avviato docker, e si avvia con il comando:

```
prompt> kathara lstart
```

Se viene modificato il file in `lab.conf` bisogna riavviare Kathará con il comando `lrestart`. Si chiude invece con il comando `lclean`.

All'avvio apre due terminali per entrambe le macchine, e vengono specificati i domini di collisioni e le connessioni definite nel file `.conf`. Per determinare la configurazione di un certo livello si utilizza sempre il comando `ip`, seguito dal livello che s vuole analizzare:

```
prompt> ip link
```

Al'interno di ciascuna delle macchine viene installato uno strumento chiamato `scapy`, una libreria in python utilizzata per creare pacchetti, ed in particolare gestire e modificare pacchetti. In questo modo si avvia il prompt di `scapy`, e si esce con il comando `exit`.

Questo terminale permette di creare un pacchetto ed inviarlo. Si crea una variabile `p` a cui assegnare il pacchetto. Si indica che si tratta di un pacchetto ethernet con `Ethernet()`, dove bisogna specificare gli indirizzi MAC del mittente e del destinatario:

```
prompt> p=Ethernet(dst:'00:00:00:00:00:01', src='00:00:00:00:00:02')
```

In questo modo è possibile specificare indirizzi MAC arbitrari, anche non presenti nella rete, e Kathará permette di analizzare questi comportamenti anomali. Il resto dei campi non specificati vengono inizializzati ad informazioni di default.

Per inviare un pacchetto si utilizza la funzione `send()`, che prende come argomenti il nome della variabile a cui è stato assegnato il pacchetto `p` ed il protocollo a cui viene inviato assegnato ad `iface`:

```
prompt> send(p, iface='eth0')
```

Esistono delle tecnologie chiamate "packet sniffer", come Wireshark, per controllare il traffico di rete. Questo viene specificato nelle ultime righe del file di configurazione, questa macchina tuttavia non è collegata, si dice fluttuante e sarà utile per analizzare il traffico su queste reti emulate.

```
wireshark[bridged]=true
wireshark[port]="3000:3000"
wireshark[image]="lscr.io/linuxserver/wireshark"
wireshark[num_terms]=0
```

L'ultima configurazione determina quanti terminali di Wireshark aprire all'avvio di Kathará. Si specifica 0, poiché si vuole utilizzare l'interfaccia grafica, e poiché in questi laboratori non si utilizzeranno comandi sul terminale i Wireshark.

Contiene un'immagine di Wireshark, chiamata analogamente per semplicità. Quest'applicazione viene avviata all'avvio del laboratorio. Questa macchina ha un'interfaccia grafica disponibile. Con il comando `lconfig` è possibile aggiungere un'interfaccia ad una macchina ad uno specifico link. Si specifica il nome della macchina con `-n` e si può specificare di aggiungere o rimuovere l'interfaccia con `--add` o `--rm`:

```
prompt> kathara lconfig -n wireshark --add A
```

Questo comando viene eseguito all'interno dell'hub del laboratorio, si utilizza per collegare questa macchina al dominio di collisione A. Nello stesso terminale dove è stato eseguito il laboratorio.

Il comando `bridged` connette la macchina all'host, con il comando `port` si specifica la porta dove la macchina condivide l'interfaccia grafica. Si accede tramite l'indirizzo `localhost:xxxx` dove viene specificata la porta inserita nella configurazione. Cliccando due volte sul nome dell'interfaccia, si possono analizzare i pacchetti inviati su quella connessione. Bisogna analizzare la connessione

`eth1`, poiché la `eth0` viene inizializzata all'avvio di Kathará ed è connessa all'host. Il protocollo su vengono spediti i pacchetti creati, non essendo specificato.

Si può utilizzare la cartella `shared` costruita ogni volta che viene costruito il laboratorio, condivisa tra la macchina dispositivo e la macchina host. In questo modo è possibile utilizzare uno sniffer diverso da Wireshark per catturare i pacchetti. Si può effettuare quest'operazione tramite i comandi Linux `tcpdump`, con l'opzione `-tenny`, una composizione di tutte le flag necessarie per configurare il comando. Quando si manda un pacchetto, si può vedere il pacchetto sul terminale, direttamente. Aggiungendo l'opzione `-w` si può creare un file della cattura, di estensione `.pcap` "Packet Capture". Per salvare questo file nella cartella `shared`, condivisa, bisogna effettuare il comando in questa cartella. Ed è possibile aprirlo tramite Wireshark nella macchina host, per analizzare i pacchetti offline rispetto alla cattura.

3 Laboratorio del 25 Ottobre: "One Bridge"

In questo laboratorio si utilizza un bridge che collega quattro macchine su quattro domini di collisione differenti:

```
pc1[0]="A/00:00:00:00:00:01"
pc1[image]="kathara/base"
pc1[ipv6]="false"
pc2[0]="B/00:00:00:00:00:02"
pc2[image]="kathara/base"
pc2[ipv6]="false"
pc3[0]="C/00:00:00:00:00:03"
pc3[image]="kathara/base"
pc3[ipv6]="false"
pc4[0]="D/00:00:00:00:00:04"
pc4[image]="kathara/base"
pc4[ipv6]="false"
```

Ed una macchina che si comporta come bridge b1:

```
b1[0]="A/00:00:00:00:00:b1"
b1[1]="B/00:00:00:00:00:b2"
b1[2]="C/00:00:00:00:00:b3"
b1[3]="D/00:00:00:00:00:b4"
b1[image]="kathara/base"
b1[ipv6]="false"
```

Per permettere questo comportamento bisogna aggiungere un'interfaccia bridge con il seguente comando, che utilizza software per realizzare bridge, già presenti nei sistemi Linux. Si utilizza il software `ip link`, già utilizzato precedentemente:

```
root@b1:~$ ip link add name mainbridge type bridge
```

In questo modo si crea l'interfaccia di nome `mainbridge`, e di tipo `bridge`, all'interno della macchina b1.

Dopo aver creato il bridge bisogna connettere le diverse interfacce della macchina al bridge appena creato. Questo processo prende il nome di "enslaving", si realizza impostando il bridge come il master di quell'interfaccia, tramite il seguente comando:

```
root@b1:~$ ip link set dev eth0 master mainbridge
```

Questo va effettuato su ogni dominio di collisione a cui è connesso il bridge.

Quando viene realizzato il bridge, di default è spento e per attivarlo, o per fermarlo `set down`, si utilizza un ulteriore comando:

```
root@b1:~$ ip link set up dev mainbridge
```

Per controllare i bridge si utilizza un'altra serie di comandi già presenti in Linux, chiamata `brctl`, per "Bridge Control". Quando un bridge riceve un pacchetto, il MAC address del mittente viene salvato per un certo periodo di tempo nel suo filtering database. Dato che questo database è dinamico, il MAC address viene rimosso dopo un tempo di invecchiamento, di default di 5 minuti, o 300 secondi. Per modificare questo tempo si inserisce nel seguente comando come parametro, in secondi:

```
root@b1:~$ brctl setageing mainbridge 600
```

Bisogna specificare il nome del bridge precedentemente creato `mainbridge` ed il tipo di operazione da effettuare `setageing`.

Considerando questi comandi, il file `b1.startup`, permette di avere un bridge funzionate ed attivo ad avvio del lab:

```
ip link add name mainbridge type bridge
ip link set dev eth0 master mainbridge
ip link set dev eth1 master mainbridge
ip link set dev eth2 master mainbridge
ip link set dev eth3 master mainbridge
ip link set up dev mainbridge
brctl setageing mainbridge 600
```

Durante il lab per osservare il filtering database del bridge si può utilizzare il comando `showmacs`, che restituisce una tabella, contenente la porta, l'indirizzo MAC corrispondente. Inoltre contiene un'indicazione se quell'indirizzo MAC è locale, all'avvio infatti conosce automaticamente gli indirizzi MAC delle sue interfacce locali; ed il tempo di invecchiamento. Se un MAC è locale, il suo tempo non viene aumentato:

```
root@b1:~$ brctl showmacs mainbridge
port no    mac addr            is local?   ageing timer
    1      00:00:00:00:00:b1   yes             0.00
    1      00:00:00:00:00:b1   yes             0.00
    2      00:00:00:00:00:b2   yes             0.00
    2      00:00:00:00:00:b2   yes             0.00
    3      00:00:00:00:00:b3   yes             0.00
    3      00:00:00:00:00:b3   yes             0.00
    4      00:00:00:00:00:b4   yes             0.00
    4      00:00:00:00:00:b4   yes             0.00
```

Il primo parametro indica il numero di porta del bridge, su un kernel Linux, il massimo numero di porte disponibili è di 1024, queste vengono assegnate sequenzialmente a partire da 1, nell'ordine in cui sono state connesse.

Se viene inviato un pacchetto da una delle stazioni, il bridge è in grado di imparare il suo MAC address:

```
root@pc1:~$ scapy
>>> p=Ether(dst='00:00:00:00:00:02', src='00:00:00:00:00:01')
```

```
>>> sendp(p, iface='eth0')
Sent 1 packets.
>>>
```

Dopo l'invio di questo pacchetto, il bridge conosce la posizione nella rete della stazione `pc1`:

```
root@b1:~$ brctl showmacs mainbridge
port no    mac addr            is local?   ageing timer
    1      00:00:00:00:00:01   no             18.54
    1      00:00:00:00:00:b1   yes             0.00
    1      00:00:00:00:00:b1   yes             0.00
    2      00:00:00:00:00:b2   yes             0.00
    2      00:00:00:00:00:b2   yes             0.00
    3      00:00:00:00:00:b3   yes             0.00
    3      00:00:00:00:00:b3   yes             0.00
    4      00:00:00:00:00:b4   yes             0.00
    4      00:00:00:00:00:b4   yes             0.00
```

Se si invia un pacchetto con l'indirizzo MAC sorgente errato, allora il bridge non è in grado di imparare la posizione delle stazioni e quindi invierà pacchetti nei domini errati, impedendo ai pacchetti di arrivare a destinazione.

4 Laboratorio del 15 Novembre: "Basic IPv4"

In questo laboratorio si studieranno i comandi di base per IPv4, i comandi di ping, di rotta di indirizzamento, del protocollo ARP ed il loro funzionamento. Su questo laboratorio verranno utilizzate cinque macchine, per configurare indirizzi IPv4 e le netmask, assegnarli agli host per raggiungere l'internet e LAN anche distanti tra di loro.

In questa rete una prima macchina `pc1` con indirizzo MAC `0:1` è connessa ad un dominio di collisione A, connesso al primo router `r1`, con indirizzo MAC `0:a1`. I due router sono connessi sul dominio di collisione B. Il router `r2` è connesso al domino B sull'indirizzo MAC `0:b2`, e connesso al domino di collisione C sull'indirizzo MAC `0:c1`. Sul dominio di collisione C sono presenti due macchine `pc2` sull'indirizzo MAC `0:2`, e `pc3` sull'indirizzo MAC `0:3`

Si utilizzano due /24 per le LAN con gli host per definire gli indirizzi. Per il dominio di collisione A si ha l'indirizzo 195.11.14/24, il dominio di collisione B ha l'indirizzo 100.0.0.8/30, per identificare solamente due indirizzi IP assegnati alle due macchine, l'indirizzo IP dell'ultimo domino di collisione è `200.1.1.0/24` Le macchine sono identificate `pc1` da `.5` e `r1` da `.1` sul dominio A. `r1` da `0.9` e `r2` da `.10` sul dominio di collisione B. Sul dominio di collisione C, la macchina `r2` su `.1`, `pc2` su `.7`, `pc3` su `.3`.

Per ogni macchina si vuole determinare un indirizzo IP per poter comunicare tramite il comando `ip address add`, seguito dall'indirizzo IP e dall'interfaccia `dev`. Inoltre bisogna specificare una rotta di default, il default gateway, tramite il comando `ip route add default via` seguito dall'indirizzo IP di default, seguito dalla specifica interfaccia dopo `dev`.

```
# pc1:
ip address add 195.11.14.5/24 dev eth0
ip route add default via 195.11.14.1
# pc2:
ip address add 200.1.1.7/24 dev eth0
ip route add default via 200.1.1.1 dev eth0
# pc3:
ip address add 200.1.1.3/24 dev eth0
ip route add default via 200.1.1.1 dev eth0
```

Per ogni router bisogna impostare due indirizzi IP per ogni dominio di collisione connesso. Quando si assegna un indirizzo IP su di un'interfaccia di un router, il sistema operativo assegna automaticamente quei prefissi nella tabella di instradamento per essere direttamente connessi. Ma bisogna aggiungere gli indirizzi che non sono direttamente connessi manualmente, per permettere al router di inoltrare pacchetti verso altri domini di collisione.

```
# r1:
ip address add 200.1.1.1/24 dev eth0
ip address add 100.0.0.10/30 dev eth1
ip route add 195.11.14.0/24 via 100.0.0.9 dev eth1
# r2:
ip address add 195.11.14.1/24 dev eth0
```

```
ip address add 100.0.0.9/30 dev eth1
ip route add 200.1.1.0/24 via 100.0.0.10 dev eth1
```

Per ottenere una descrizione della configurazione IPv4 di una macchina si utilizza il comando `ip address`. Il comando `routel` mostra tutte le destinazione che la macchina è in grado di raggiungere, mostrando la tabella di instradamento.

SI inserisce un link alla macchina wireshark sul dominio di collisione C, e si effettua un ping tra `pc3` e `pc2`, tramite i seguenti comandi:

```
> kathara lconfig -n wireshark --add C
```

Prima di effettuare una comunicazione tramite IPv4, la macchina deve individuare l'indirizzo MAC del destinatario, tramite il protocollo ARP. Per controllare gli indirizzi MAC memorizzati si può ursae il comando `arp`, si può inserire la flag `-n` per non risolvere gli indirizzi.

Wireshark fornisce anche l'informazione su quale pacchetto di risposta è legato a quale pacchetto di richiesta.

In questo modo si può analizzare il comportamento di una connessione diretta, per analizzare una connessione ad una LAN remota si può collegare la macchina wireshark al dominio B, ed effettuare un ping tra la macchina `pc3` e `pc1`. In questo modo si può osservare che i pacchetti "sniffati" da wireshark contegno solamente gli indirizzi MAC dello specifico dominio di collisione, non gli indirizzi MAC appartenenti al dominio A o C. Invece si utilizzano gli indirizzi IP reali del destinatario e del mittente.

Si esegue il comando `traceroute` con la flag `-z 1` per semplificare la visualizzazione. Su wireshark saranno visibili tutti questi pacchetti inviati ed il loro ttl progressivamente maggiore. Si esegue questo comando da `pc2` a `pc1`:

```
root@pc2:~# traceroute 195.11.15.5 -z 1
```

In caso si indicasse un indirizzo IPv4 inesistente effettuando un ping, vengono inviati i pacchetti di richiesta, ma la ARP request non riceve risposta quindi inoltra un errore al mittente. Il router che ha fallito la consegna inoltra il messaggio ICMP di errore. Se invece si prova a effettuare un ping su un indirizzo non presente su questa rete, viene sollevato un errore di tipo Destination Net Unreachable. Questi pacchetti di errore devono essere mandati da una macchina all'interno di questa rete che è in grado di determinare che l'indirizzo non appartiene alla rete. Questo indirizzo non è nella tabella di instradamento, quindi viene mandato al default gateway, ma non essendo nemmeno nella sua tabella di instradamento, viene scartato ed invia il messaggio ICMP al mittente.

Ora si prova ad effettuare un traceroute verso una macchina che non esiste su una LAN esistente, copia l'ultima riga del traceroute prima della macchina inesistente indicando il carattere `!H`, per specificare che manca la macchina host. Altrimenti se si specifica una LAN inesistente, produce solamente una riga con il carattere `!N`, per indicare che non è presente il network.

5 Laboratorio del 22 Novembre: "Basic IPv6"

6 Laboratorio del 6 Dicembre: "DNS"

In questo lab sono presenti 5 zone: uniroma3, it, "" (zona della root), net e startup. Quest zone sono gestite rispettivamente dai NS localuni e dnsuni, dnsit, dnsroot, dnsnet, dnsstart e localstart.

La topologia è piatta, tutte le macchine sono legate allo stesso dominio di collisione 192.168.0.0/24. Sono presenti altre due macchine pc1 e pc2, contenute rispettivamente in uniroma3 e startup.

Tutti questi NS sono configurati per poter gestire richieste iterative o ricorsive. L'immagine docker utilizzata per gestire i DNS si chiama docker bind 9.11. Non bisogna configurare alcuna rotta, poiché sono tutti direttamente connessi. In questa configurazione per la prima volta ogni macchina contiene una sua cartella, contenente un file `resolv.conf`, questo file indica alla macchina dove si trova il suo Local Name Server. Si utilizza il comando `search` per indicare per cercare un sotto-dominio di un dominio, poiché quando una ricerca fallisce, invia di nuovo la richiesta, appendendo questo parametro, in modo da cercare all'interno del suo dominio.

Kathará all'avvio copia nel percorso specificato i contenuti di queste cartelle. I DNS invece hanno una configurazione diversa, il nome del processo che parte e si occupa della gestione del DNS si chiama `named`, questo si trova all'interno della cartella bind, dal comando Linux utilizzato per generare questo processo DNS. Si indica nelle opzioni in questa cartella l'indirizzo dove salvare in cache gli indirizzi cercati. Le configurazione del DNS comprendono sempre un comando `include` passando coe parametro la posizione delle opzioni di questo DNS, nella cartella bind descritta precedentemente ed un comando per assegnare la zona `zona`, che indica il nome della zona, e determina se la macchina è root. Nelle opzioni la prima riga indica il tempo per cui i valori memorizzati in cache verranno mantenuti, indicato dal comando `$TTL`. In seguito ad una chiocciola, che utilizzato per non dover ripetere la zona del NS, si inseriscono le opzioni per la gestione della relazione master-slave e si indica con `SOA` che questo NS è l'inizio di una nuova zona. È presente la mail di posta del gestore del DNS, per operatori umani in caso qualcosa vada storto, in seguito tra parentesi è presente la configurazione tra slave-master del DNS, con un numero seriale, indicato da `serial`, un tempo di refresh, un tempo di retry per provare a ricontattare il master, in tempo secondo dopo il quale non prova più a contattare con il master, ed un tempo di mantenimento in cache delle informazioni negative. In ognuno di questi comandi si indica il tipo della connessione `IN`, in questi casi poiché si tratta sempre di connessioni internet.

In seguito si indicano le autorità per il dominio `.it` e `.next` indicando che questo NS conosce informazioni su queste due zone. Si indica che questi NS sono collegati allo stesso dominio di collisione A, con un indirizzo 192.68.0.5 per il NS della radice, 192.168.0.1 per dnsit e 192.168.0.2 per dnsnet.

Sono presenti configurazioni analoghe per lo SOA delle altre zone

7 Laboratorio del 13 Dicembre: "HTTP-TCP"