



# Redundancy dei servizi con ucarp

Alessio Pennasilico  
a.pennasilico@alba.st  
<http://www.alba.st>

# Finalità

Capire come realizzare una infrastruttura di servizi ridondanti senza eccedere nel numero delle macchine, ottimizzandone l'utilizzo.

# Prerequisiti

Conoscenza di base del TCP/IP e della gestione delle interfacce con Linux.

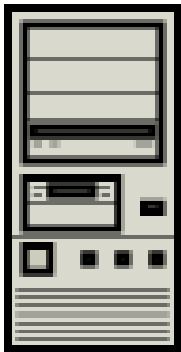
# Sommario

Obiettivi  
Servizi in modalità “standard”  
Configurare ucarp  
Verifica della nuova topologia

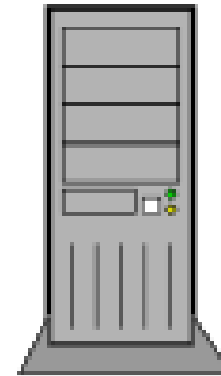
# Situazione di partenza

Immaginiamo due server, un WEB ed un Mail server, collegati in una rete senza accorgimento alcuno.

# Topologia di partenza



WEB Server  
172.16.90.31



Mail Server  
172.16.90.32

# PRO/CONS

Una simile topologia di tipo legacy risulta facilmente gestibile ma in caso di failure di uno dei due apparati il relativo servizio risulta non più disponibile.

# Obiettivi

Evitare che in caso di failure di un server il servizio da lui gestito diventi inutilizzabile.

# Strategia

Installare una copia identica del servizio web sul mail server.

Creare un indirizzo IP virtuale al quale attribuire la gestione del servizio.

# Modalità

L'indirizzo IP dell'host virtuale di default verrà sempre reindirizzato sul web server.

In caso di failure di questa macchina il mail server prenderà in gestione l'IP, e quindi il servizio.

# Considerazioni di design

In fase di dimensionamento delle macchine mi devo mettere in entrambi i casi nella situazione peggiore, quindi avere server in grado di gestire entrambi i servizi in caso di emergenza.

# Configurazione

```
# loading virtual interface
```

```
/usr/sbin/ucarp -i eth0  
-s 172.16.90.101 \  
-v 1 -P \  
-p password \  
-a 172.16.90.31 -B \  
-u /opt/bin/vip1-up.sh \  
-d /opt/bin/vip1-down.sh
```

# Scripts

**/opt/bin/vip1-up.sh**

```
#!/bin/sh  
/sbin/ifconfig eth0:1 172.16.90.31
```

**/opt/bin/vip1-down.sh**

```
#!/bin/sh  
/sbin/ifconfig eth0:1 down
```

# Verifica

```
web ~ # ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 00:11:25:AB:B9:0C
```

```
inet addr:172.16.90.101  Bcast:172.16.90.255
```

```
Mask:255.255.255.0
```

```
UP BROADCAST RUNNING MULTICAST  MTU:1500
```

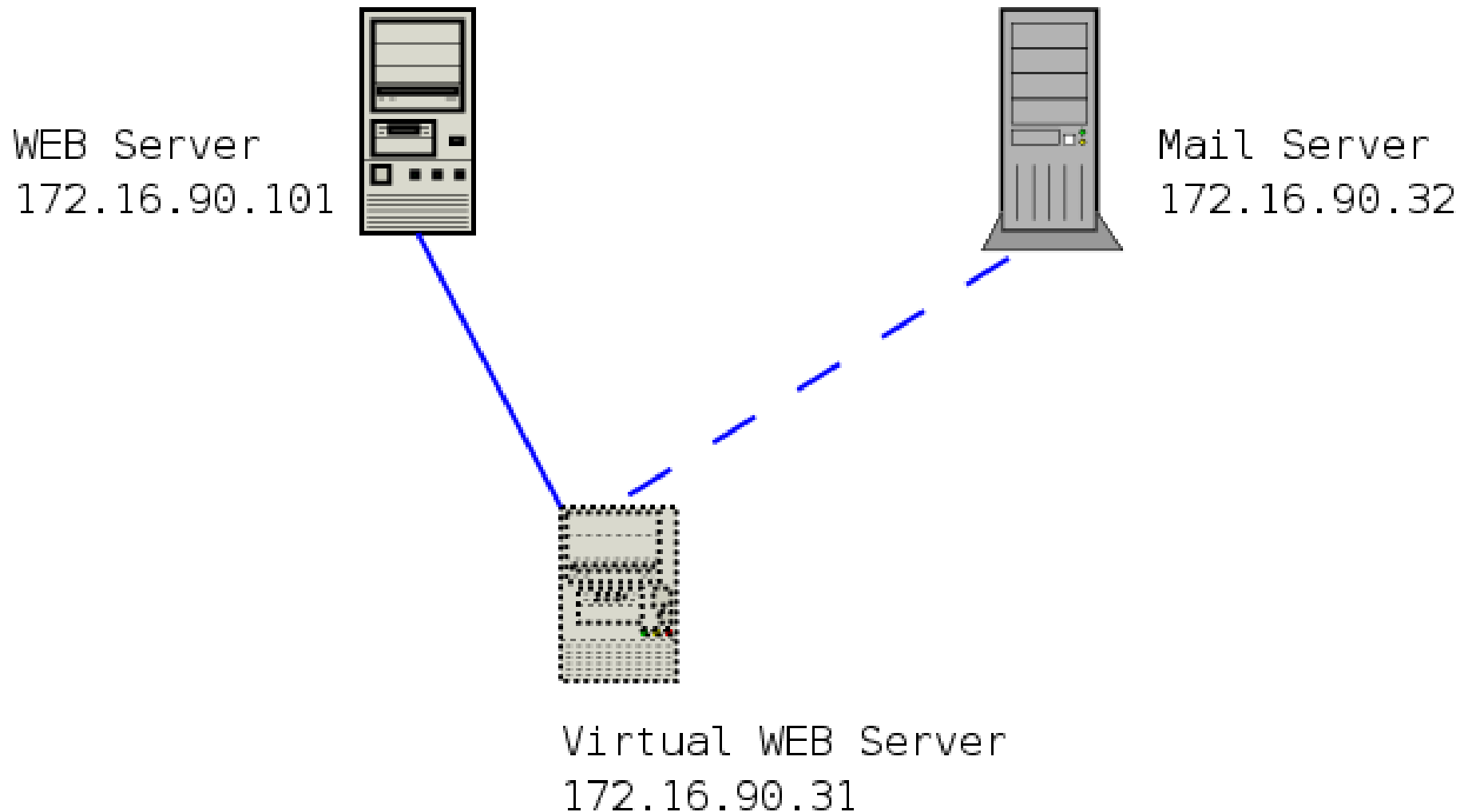
```
eth0:1    Link encap:Ethernet  HWaddr 00:11:25:AB:B9:0C
```

```
inet addr:172.16.90.31  Bcast:172.16.90.255
```

```
Mask:255.255.255.0
```

```
UP BROADCAST RUNNING MULTICAST  MTU:1500
```

# Risultato



# Test

Scollegando il cavo di rete dal web server potremo verificare che il mail server inizierà a processare le richieste.

# Attenzione

Ucarp previene i disservizi dovuti ad host failure. Se il server fosse funzionante a meno del servizio da lui gestito Ucarp non entrerebbe in funzione.

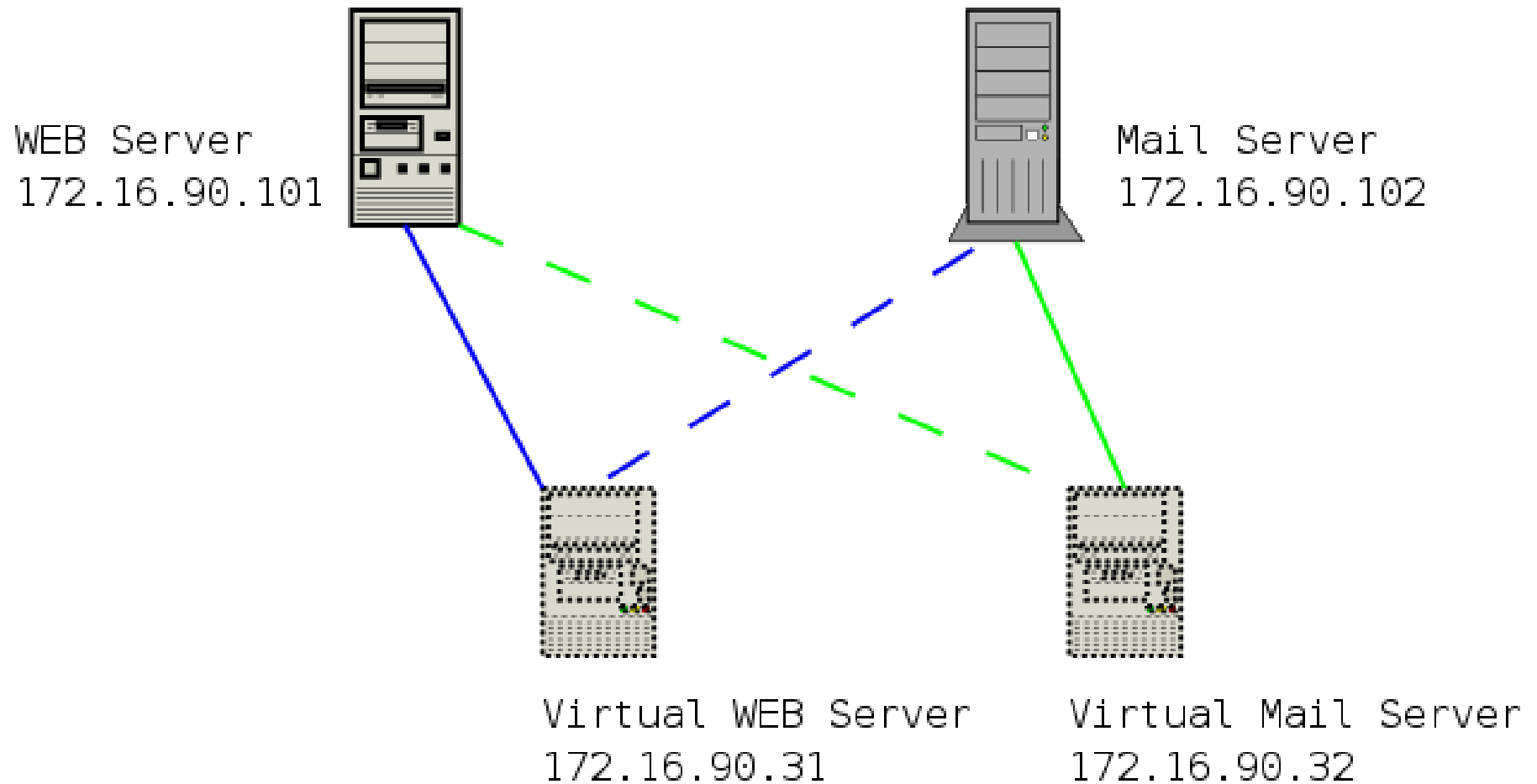
# Sviluppi...

E' tuttavia possibile verificare la disponibilit  dei servizi con degli script schedulati e scatenare Ucarp nel caso in cui un servizio non risponda.

# Aggiungo il secondo servizio

```
# loading virtual interface
/usr/sbin/ucarp -i eth0 \
  -s 172.16.90.102 \
  -v 1 -P \
  -p password \
  -a 172.16.90.32 -B \
  -u /opt/bin/vip1-up.sh \
  -d /opt/bin/vip1-down.sh
```

# Risultato





# tirando le somme...

Ho ottenuto la funzionalità di fail-over (non di load-balancing) per due servizi critici per la mia infrastruttura, senza dover investire nell'acquisto di ulteriori server che avrebbero lavorato pochissimo nel tempo.

# Bibliografia

- ✓ <http://www.ucarp.org>
- ✓ <http://www.alba.st>
- ✓ <http://www.recursiva.org>
- ✓ \$ apropos && man && google :)

# Licenza

Queste slides sono realizzate da Alessio “mayhem” Pennasilico e sono soggette alla licenza Creative Commons nella versione Attribution-ShareAlike 2.0; possono pertanto essere distribuite liberamente ed altrettanto liberamente modificate, a patto che se ne citi l'autore e la provenienza.