

**UNIVERSITY OF ROMA TRE**

Department of Computer Science and Automation

# **Emulating Computer Networks with Netkit**

**Massimo Rimondini**

Computer Networks Research Group

<http://www.dia.uniroma3.it/~compunet>

4th International Workshop on Internet Performance, Simulation,  
Monitoring, and Measurement

# Copyright notice

- ◆ All the slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as "material") are protected by copyright
- ◆ This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and organizations appearing in the first slide
- ◆ This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes
- ◆ Information contained in this material cannot be used within network design projects or other products of any kind
- ◆ Any other use is prohibited, unless explicitly authorized by the authors on the basis of a written agreement
- ◆ Authors assume no responsibility for the contents of this material, which may be subject to changes
- ◆ This copyright notice must always be redistributed together with the material, or its portions

# Prerequisites

- ✦ Very basic knowledge of the Linux OS

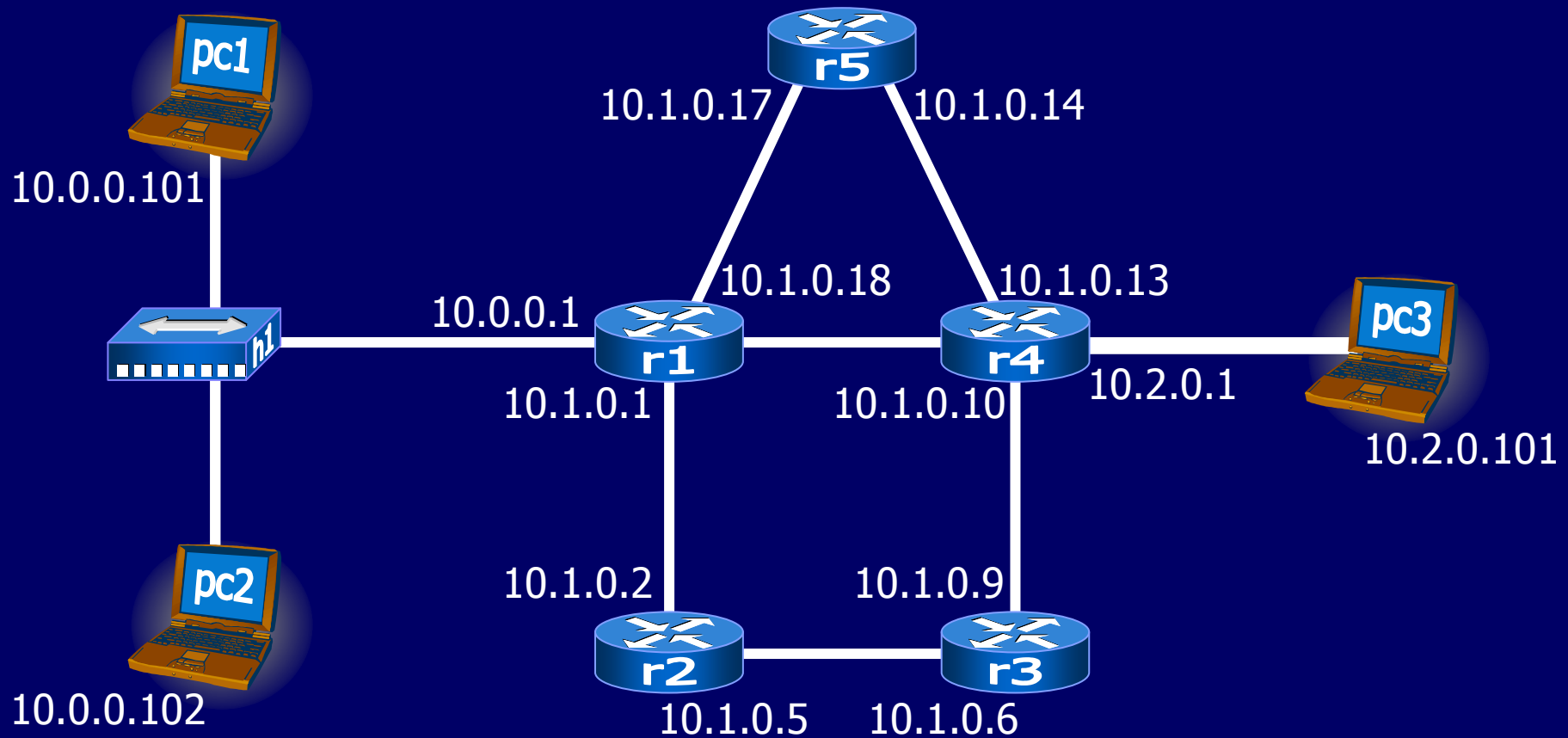


- ✦ Networking

- iso-osi stack
- routing protocols (rip, bgp)
- basic network tools



# What's this *Natkeet* thing?



This is *not* Netkit...

# What's this *Natkeet* thing?

```
-----
64 bytes from 10.1.0.10: icmp_seq=1 ttl=62 time=13.1 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=62 time=1.39 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=62 time=1.39 ms
64 bytes from 10.1.0.10: icmp_seq=4 ttl=62 time=1.39 ms
64 bytes from 10.1.0.10: icmp_seq=5 ttl=62 time=1.39 ms
--- 10.1.0.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 66.0ms
rtt min/avg/max/mdev = 1.363/3.824/13.171/4.000 ms
pc1:~# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) 56(84) bytes of data:
64 bytes from 10.2.0.1: icmp_seq=1 ttl=62 time=1.39 ms
64 bytes from 10.2.0.1: icmp_seq=2 ttl=62 time=1.39 ms
--- 10.2.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 27.0ms
rtt min/avg/max/mdev = 1.125/1.341/1.557/0.000 ms
pc1:~# traceroute 10.2.0.101
traceroute to 10.2.0.101 (10.2.0.101), 64 hops max, 40 byte packets
 1 10.0.0.1 (10.0.0.1)  2 ms  4 ms  2 ms
 2 10.1.0.17 (10.1.0.17)  2 ms  4 ms  3 ms
 3 10.1.0.13 (10.1.0.13)  3 ms  3 ms  3 ms
 4 10.2.0.101 (10.2.0.101)  3 ms  5 ms  3 ms
pc1:~#

-----
6 packets captured
6 packets received by filter
0 packets dropped by kernel
pc2:~# route
Kernel IP routing table
Destination:
10.0.0.0
default
pc2:~# ping 10.2.0.101
PING 10.2.0.101 (10.2.0.101) 56(84) bytes of data:
64 bytes from 10.2.0.101: icmp_seq=1 ttl=62 time=10.5 ms
64 bytes from 10.2.0.101: icmp_seq=2 ttl=62 time=10.5 ms
64 bytes from 10.2.0.101: icmp_seq=3 ttl=62 time=10.5 ms
--- 10.2.0.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 31.5ms
rtt min/avg/max/mdev = 10.599/10.599/10.599/0.000 ms
pc2:~# traceroute 10.2.0.101
traceroute to 10.2.0.101 (10.2.0.101), 64 hops max, 40 byte packets
 1 10.0.0.1 (10.0.0.1)  1 ms  2 ms  1 ms
 2 10.1.0.17 (10.1.0.17)  1 ms  4 ms  2 ms
 3 10.1.0.13 (10.1.0.13)  3 ms  4 ms  3 ms
 4 10.2.0.101 (10.2.0.101)  3 ms  4 ms  3 ms
pc2:~#

-----
r5
Connected to r5.
Escape character is '^['.
z
Hello, this is zebra (version 0.94).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification

Password:
r5> show ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

      Network        Next Hop        Metric From        Time
R(n) 10.0.0.0/24    10.1.0.18       2 10.1.0.18      02:43
O(s) 10.1.0.0/30   10.1.0.18       2 10.1.0.18      02:43

-----
r1
R(n) 10.1.0.4/30    10.1.0.2        2 10.1.0.2        02:15
R(n) 10.1.0.8/30    10.1.0.2        3 10.1.0.2        02:15
R(n) 10.1.0.12/30   10.1.0.17       2 10.1.0.17       02:57
C(i) 10.1.0.16/30   0.0.0.0         1 self
R(n) 10.2.0.0/24    10.1.0.17       3 10.1.0.17       02:57
r1>
r1> Connection closed by foreign host.
r1:~# topdump -i eth2 icmp
topdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes
17:59:38.594527 IP 10.1.0.18 > 10.2.0.101: icmp 48: time exceeded in-transit
17:59:38.599950 IP 10.1.0.18 > 10.2.0.101: icmp 48: time exceeded in-transit
17:59:38.602829 IP 10.1.0.18 > 10.2.0.101: icmp 48: time exceeded in-transit
17:59:38.615524 IP 10.0.0.102 > 10.2.0.101: icmp 48: 10.0.0.102 udp port 33444 unreachable
17:59:38.621018 IP 10.0.0.102 > 10.2.0.101: icmp 48: 10.0.0.102 udp port 33445 unreachable
17:59:38.623607 IP 10.0.0.102 > 10.2.0.101: icmp 48: 10.0.0.102 udp port 33446 unreachable
r1:~#

-----
r4 login: root (automatic login)
Linux pc1 2.6.11.7 #1 Tue Sep 13 18:38:01 UTC 2005; root@pc1:~#
Welcome to Netkit

r4:~# route
Kernel IP routing table
Destination:
10.1.0.4
10.1.0.0
10.1.0.0
10.1.0.12
10.1.0.8
-----
64 bytes from 10.0.0.101: icmp_seq=4 ttl=61 time=1.39 ms
--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3041ms
rtt min/avg/max/mdev = 1.315/1.552/2.055/0.295 ms
pc3:~# traceroute 10.0.0.101
traceroute to 10.0.0.101 (10.0.0.101), 64 hops max, 40 byte packets
 1 10.2.0.1 (10.2.0.1)  1 ms  1 ms  1 ms
 2 10.1.0.14 (10.1.0.14)  1 ms  1 ms  1 ms
 3 10.1.0.18 (10.1.0.18)  2 ms  2 ms  1 ms
 4 10.0.0.101 (10.0.0.101)  2 ms  2 ms  2 ms
pc3:~# traceroute 10.0.0.102
traceroute to 10.0.0.102 (10.0.0.102), 64 hops max, 40 byte packets
 1 10.2.0.1 (10.2.0.1)  1 ms  2 ms  1 ms
 2 10.1.0.14 (10.1.0.14)  3 ms  2 ms  1 ms
 3 10.1.0.18 (10.1.0.18)  2 ms  2 ms  1 ms
 4 10.0.0.102 (10.0.0.102)  2 ms  2 ms  2 ms
pc3:~# traceroute 10.0.0.102
traceroute to 10.0.0.102 (10.0.0.102), 64 hops max, 40 byte packets
 1 10.2.0.1 (10.2.0.1)  1 ms  1 ms  1 ms
 2 10.1.0.14 (10.1.0.14)  3 ms  2 ms  1 ms
 3 10.1.0.18 (10.1.0.18)  13 ms  2 ms  2 ms
 4 10.0.0.102 (10.0.0.102)  12 ms  2 ms  2 ms
pc3:~#

-----
r3:~# ip route show
10.1.0.16
10.1.0.4/30 dev eth0 proto 1
10.1.0.0/30 via 10.1.0.5 dev eth0
10.1.0.12/30 via 10.1.0.10 dev eth0
10.1.0.8/30 dev eth1 proto 1
10.1.0.16/30 via 10.1.0.5 dev eth0
10.2.0.0/24 via 10.1.0.10 dev eth0
10.0.0.0/24 via 10.1.0.5 dev eth0 proto zebra metric 3 equalize
r3:~# ping 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data:
64 bytes from 10.0.0.101: icmp_seq=1 ttl=62 time=10.5 ms
--- 10.0.0.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.599/10.599/10.599/0.000 ms
r3:~# traceroute 10.0.0.101
traceroute to 10.0.0.101 (10.0.0.101), 64 hops max, 40 byte packets
 1 10.1.0.5 (10.1.0.5)  1 ms  2 ms  1 ms
 2 10.1.0.1 (10.1.0.1)  1 ms  2 ms  1 ms
 3 10.0.0.101 (10.0.0.101)  2 ms  2 ms  2 ms
r3:~#
```

This *is* Netkit!

# Yet another boring emulator toy?

## ◆ From Wikipedia:

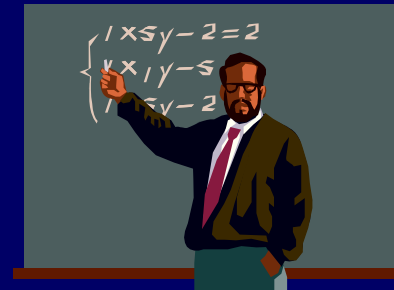
- Bochs, portable open source x86 and AMD64 PCs emulator
- FAUmachine
- Microsoft Virtual PC
- Microsoft Virtual Server
- OPEN COLINUX - Linux inside Windows
- Parallels
- QEMU
- SVISTA
- TRANGO Real-time Embedded Hypervisor
- twoOSTwo
- User-mode Linux
- Virtuozzo
- VM from IBM, apparently the first true virtual machine system and still in wide use today.
- VMWare
- Xen

# Yet another boring emulator toy?

Product	License	Pros	Cons
Xen	Open Source	◆ Performance	◆ Requires replacing the host kernel ◆ Requires porting the guest OS kernel
VMWare	Commercial	◆ Near native performance	◆ Complex configuration
Bochs	Open Source	◆ Good support for many guest OSES	◆ Performance
QEMU	(almost) Open Source	◆ Emulates multiple architectures ◆ Comes with an accelerator module	◆ Requires privileges to be installed ◆ Not lightweight
Plex86	Open Source	◆ Lightweight	◆ Development stalled

# Success stories involving Netkit

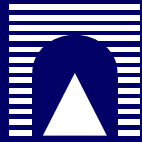
- ◆ Networking courses
  - routing protocols
  - application level services (dns, web, email, etc.)
- ◆ Emulation of the italian academic research network (GARR)
  - assignment of OSPF weights





# Outline

- ◆ **Understanding Netkit**
  - Architecture overview
- ◆ **Setting up Netkit**
  - Download and installation
- ◆ **Using Netkit**
  - Getting acquainted with Netkit commands
  - How to prepare Netkit labs
- ◆ **Sample scenarios**
  - A virtual network running BGP

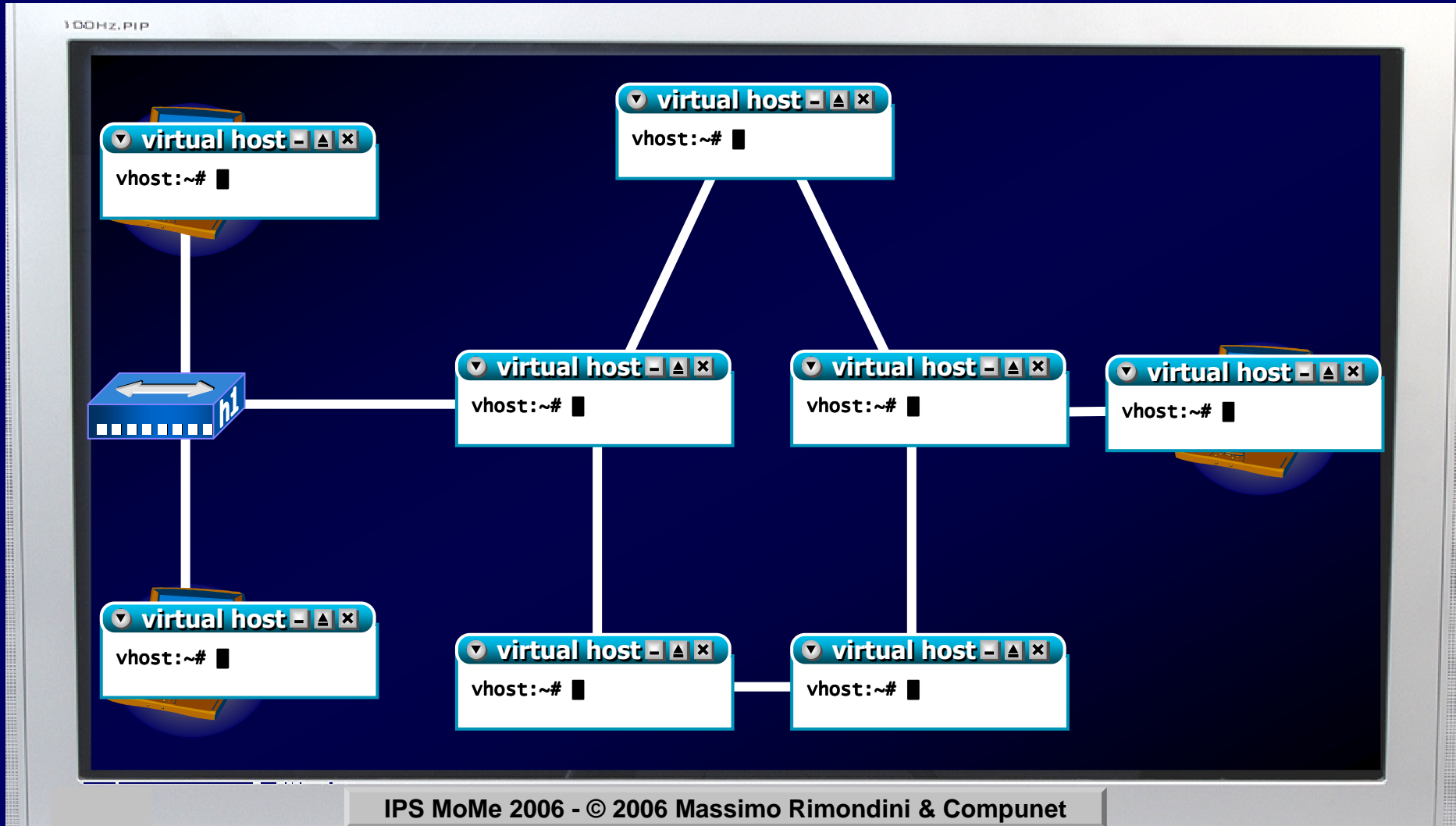


**UNIVERSITY OF ROMA TRE**

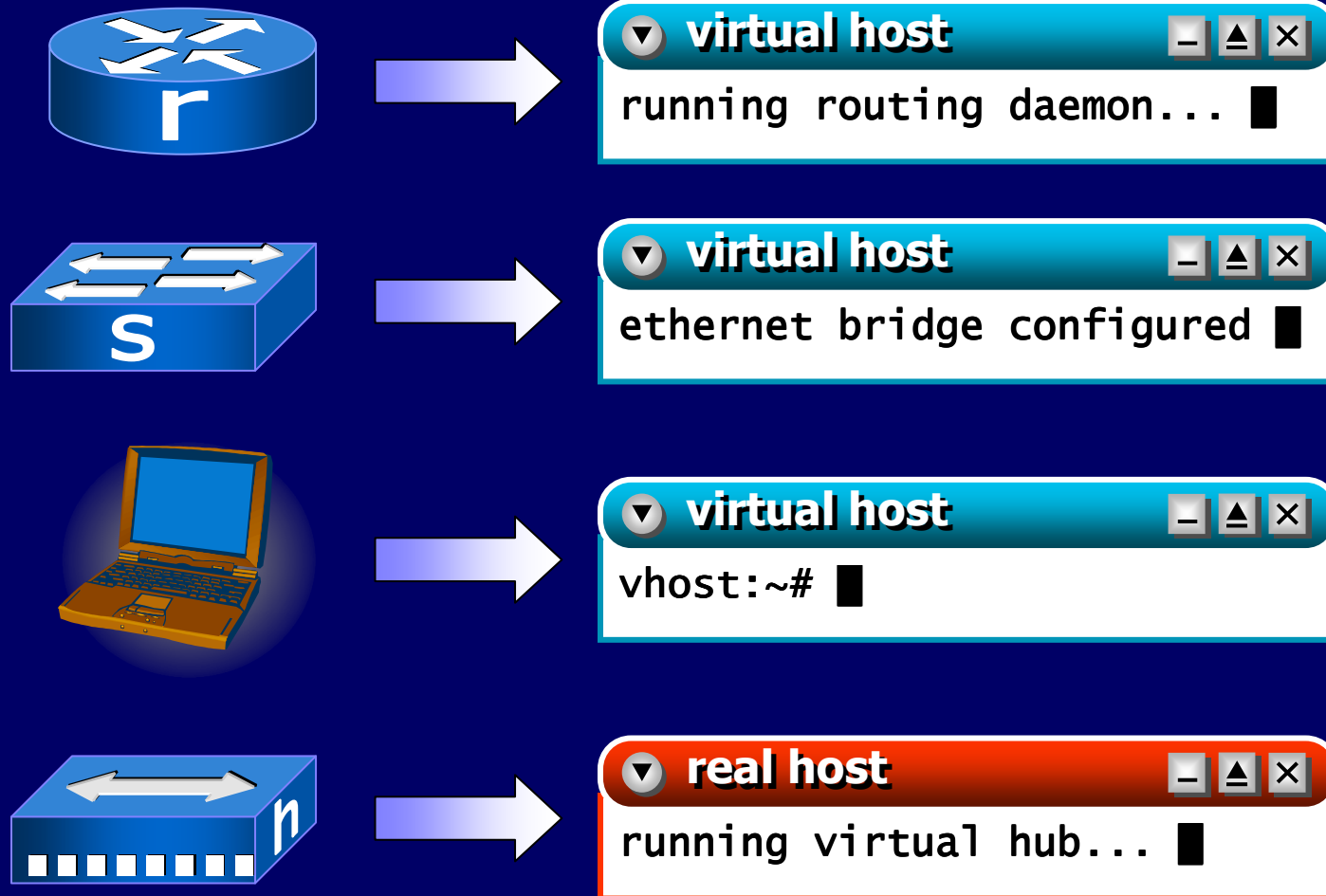
Department of Computer Science and Automation

# Understanding Netkit

# A Netkit network



# A Netkit network

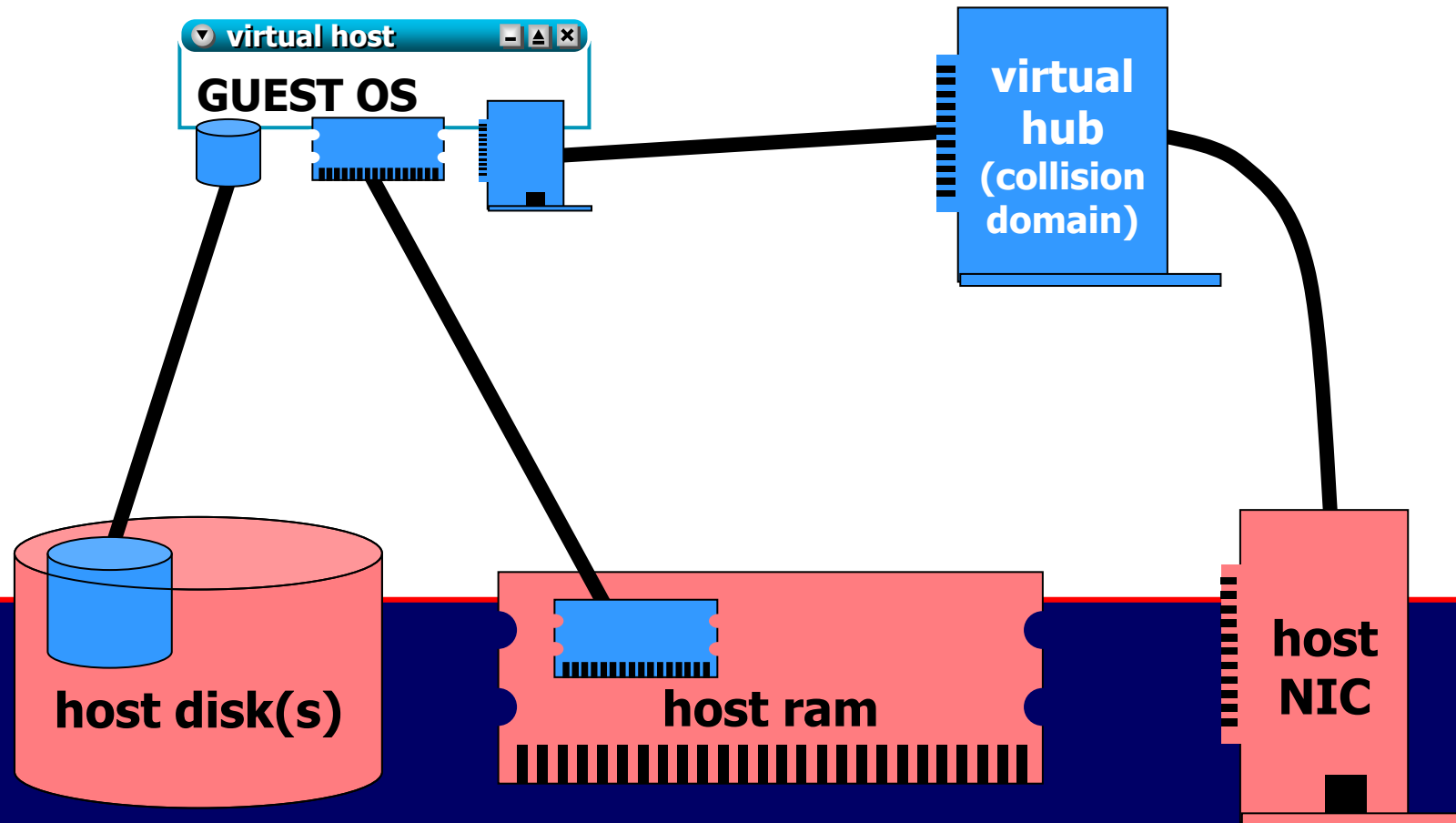


# A closer look at virtual machines

▼ real host



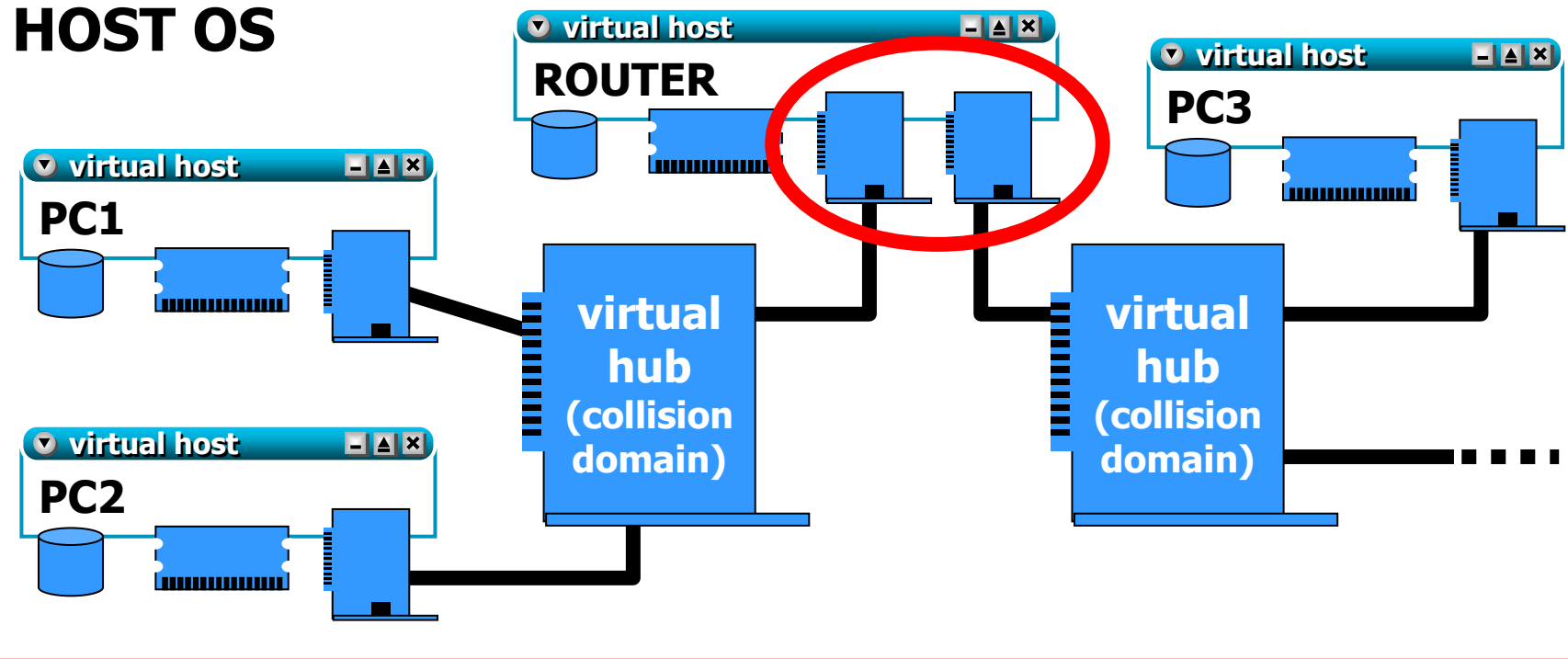
## HOST OS



# Interconnecting virtual machines

real host

HOST OS



# A few technicalities

- ◆ Virtual machines run a **user-mode-linux kernel**
  - Linux kernel compiled for running as a userspace process
  - Thin emulation layer ⇒ better performance
  - <http://user-mode-linux.sourceforge.net/>
- ◆ The guest OS is a **Debian GNU/Linux unstable**
  - Most popular networking tools are included
  - Copy-On-Write
    - “Damage-free” filesystem

# Cables & Interfaces

- ◆ Virtual machines can be equipped with an arbitrary number of network interfaces

## Network Layer

- Handled by the UML kernel
- IPv4/IPv6

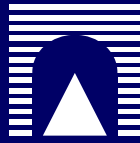
## Data-link Layer

- Ethernet
- No collisions

## Physical layer

- Virtual hub daemon (uml\_switch)
- Communication over unix sockets
- No delay, loss, reordering





**UNIVERSITY OF ROMA TRE**

Department of Computer Science and Automation

# Setting up Netkit

# System requirements

Host	i386 Linux host
CPU	>600 MHz ✦ lower freqs impact performance
RAM	~10MB for each virtual machine ✦ depends on the amount of emulated memory
Hard disk space	~650 MB + ~1-20 MB for each virtual machine ✦ depends on vm usage
Others	ext2/ext3 filesystems preferred

# A tiresome procedure 😊

◆ Just three steps

1. Download @ <http://www.netkit.org>

◆ `netkit-2.4.tar.bz2`

◆ `netkit-filesystem-F2.2.tar.bz2`

◆ `netkit-kernel-K2.2.tar.bz2`



2. `cd` to a directory of your choice and unpack the files

host machine

```
foo@host:~$ cd netkit
foo@host:~/netkit$ tar xjf netkit-2.4.tar.bz2
foo@host:~/netkit$ tar xjf netkit-filesystem-F2.2.tar.bz2
foo@host:~/netkit$ tar xjf netkit-kernel-K2.2.tar.bz2
foo@host:~$ █
```



# A tiresome procedure 😊

## 3. Set some environment variables

- ◆ set **NETKIT\_HOME** to the path where you installed Netkit
- ◆ set **PATH** to the string  
"\$PATH:\$NETKIT\_HOME/bin"
- ◆ set **MANPATH** to the string ":\$NETKIT\_HOME/man"

Example (using bash):

```
host machine
foo@host:~/netkit$ export NETKIT_HOME=/home/foo/netkit/netkit2
foo@host:~/netkit$ export PATH=$PATH:$NETKIT_HOME/bin
foo@host:~/netkit$ export MANPATH=:$NETKIT_HOME/man
foo@host:~/netkit$ █
```

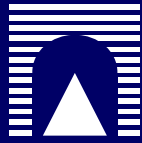
That's all folks!

# Test your installation

- ◆ cd to the Netkit directory
- ◆ Run `check_configuration.sh`

```
host machine
foo@host:~/netkit$ cd netkit2
foo@host:~/netkit/netkit2$ ./check_configuration.sh
.....
[ READY ] Congratulations! Your Netkit setup is now complete!
          Enjoy Netkit!
foo@host:~/netkit/netkit2$ █
```

- ◆ Test failures are accompanied by a short description of the fix



**UNIVERSITY OF ROMA TRE**

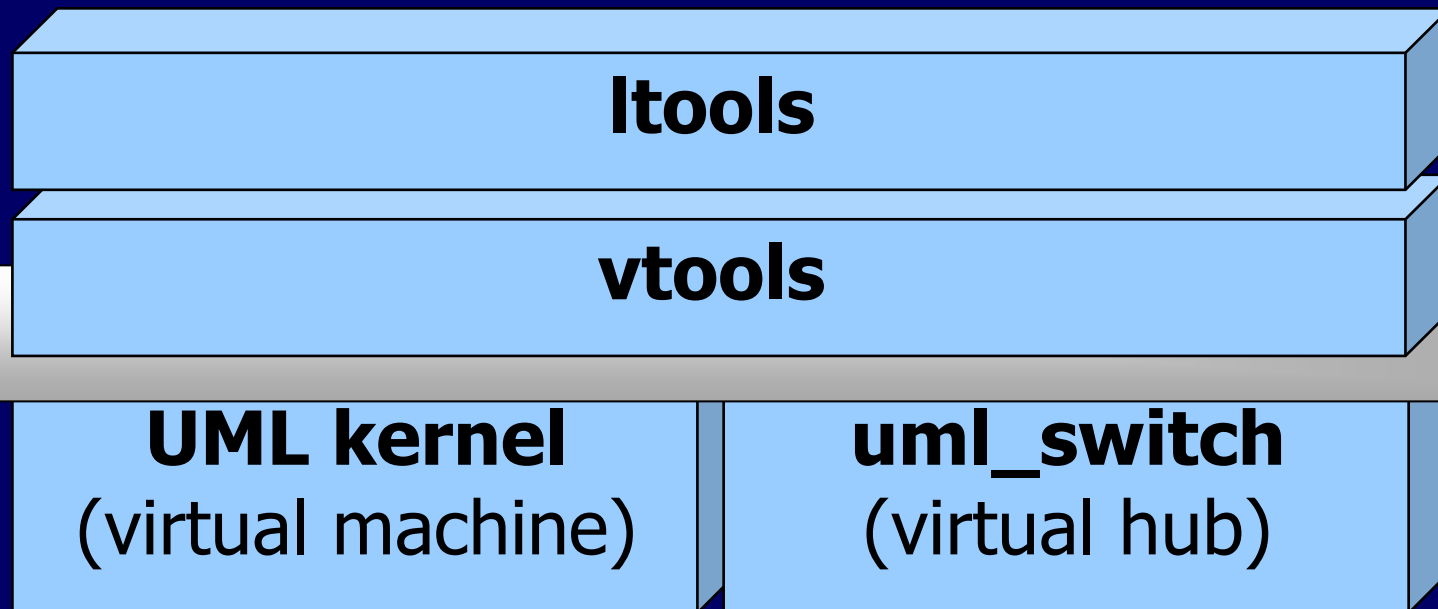
Department of Computer Science and Automation

# Using Netkit

(at last...)

# A toolkit made up of tools

- Virtual machines can be controlled by using two interfaces



# Vtools

<b>vstart</b>	Start a virtual machine with a given configuration
<b>vconfig</b>	Attach a network interface to a running virtual machine
<b>vlist</b>	List running virtual machines
<b>vhalt</b>	Gracefully shut down a virtual machine
<b>vcrash</b>	Kill a virtual machine
<b>vclean</b>	Panic button





# Booting a virtual machine

```
vstart [options] MACHINE_NAME
```

## ◆ Most common options:

- `--ethN=collision_domain`
  - Interfaces on the same collision domain can exchange traffic
- `-M memory_amount`
- `-p`
  - Just show what would be done

host machine

```
foo@host:~$ vstart --eth0=CD_A --eth1=CD_B -M 256 pc1
```

```
===== Starting virtual machine "pc1" =====
```

```
Kernel:      /home/max/netkit2/kernel/netkit-kernel
```

```
Modules:     /home/max/netkit2/kernel/modules █
```

```
.....
```

# Look: it's booting...

- The virtual machine terminal window automatically pops up

```
pc1
--- Starting Netkit phase 2 startup script

virtual host pc1 ready.


--- Netkit phase 2 init script terminated

pc1 login: root (automatic login)
Linux pc1 2.6.11.7 #1 Tue Sep 13 18:38:01 CEST 2005 i686
GNU/Linux
welcome to Netkit

pc1:~# █
```

# Where VMs write things

- ◆ **Copy-On-Write**: every change to the model filesystem is written to `pc1.disk`
- ◆ **Sparse files**: zeros do not consume disk space



```
host machine
foo@host:~$ ls -l pc1.*
-rw-r--r-- 1 foo foo 630358016 2006-02-18 19:53 pc1.disk
-rw-r--r-- 1 foo foo 4096 2006-02-18 19:57 pc1.log
foo@host:~$ du -h pc1.*
884K    pc1.disk
4.0K    pc1.log
foo@host:~$ █
```

# Getting rid of a virtual machine

- ◆ Gracefully shut down from the inside...

```
pc1
pc1:~# halt

Broadcast message from root (vc/0) (Sat Feb 18 19:46:13 2006):

The system is going down for system halt NOW!
INIT: Switching to runlevel: 0
INIT: Sending processes the TERM signal █
```

- ◆ ...or from the outside...

```
host machine
foo@host:~$ vhalt pc1
Halting virtual machine "pc1" (PID 8598) owned by foo [...]
foo@host:~$ █
```

# Getting rid of a virtual machine

- ◆ ...or harshly unplug its power cord

host machine

```
foo@host:~$ vcrash pc1
```

```
===== Crashing virtual machine "pc1" (PID 9741) =====
```

```
virtual machine owner: foo
```

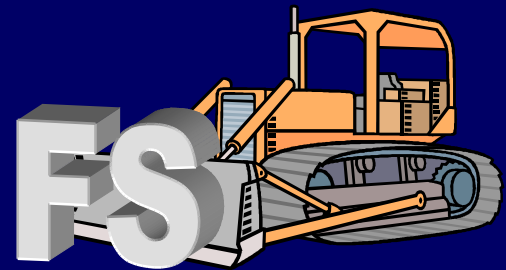
```
virtual machine mconsole socket:
```

```
/home/foo/.netkit/mconsole/pc1/mconsole
```

```
Crashing... done.
```

```
foo@host:~$ █
```

- ◆ Quick
- ◆ Screws virtual machine filesystem
- ◆ Remember: Copy-On-Write



# A Netkit lab...

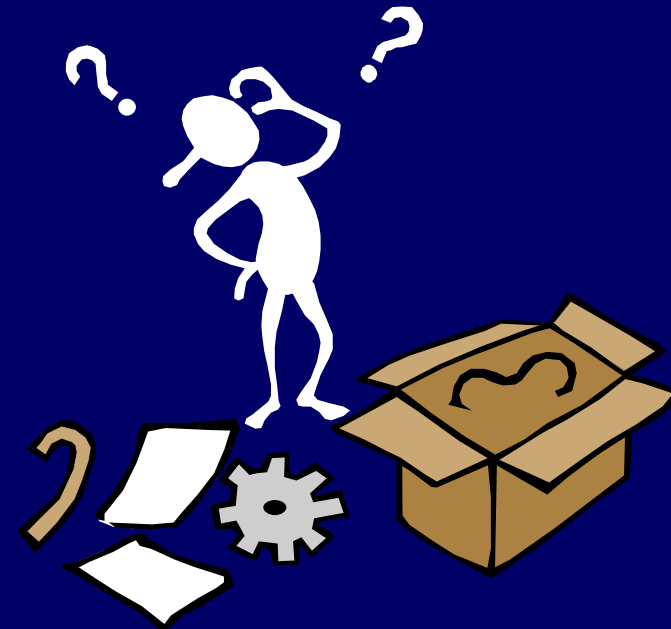
- ◆ ...*is* a set of pre-configured virtual machines that can be launched/stopped all together
- ◆ ...*consists* of a hierarchy of directories
- ◆ ...*allows* to set up complex network experiences
- ◆ ...*can be controlled* by using the ltools

# Ltools

<b>lstart</b>	Start a Netkit lab or just some of its machines
<b>ltest</b>	Start a Netkit lab in test mode
<b>lhalt</b>	Gracefully shut down (some of) the virtual machines of a lab
<b>lcrash</b>	Kill (some of) the virtual machines of a lab
<b>linfo</b>	Display info about a lab without starting it; sketch the network topology
<b>lclean</b>	Remove temporary files (no panic!)

# Mastering your own Netkit Lab

- ◆ How to prepare a lab:
  1. define topology
  2. assign addresses
  3. configure network services
  4. ...

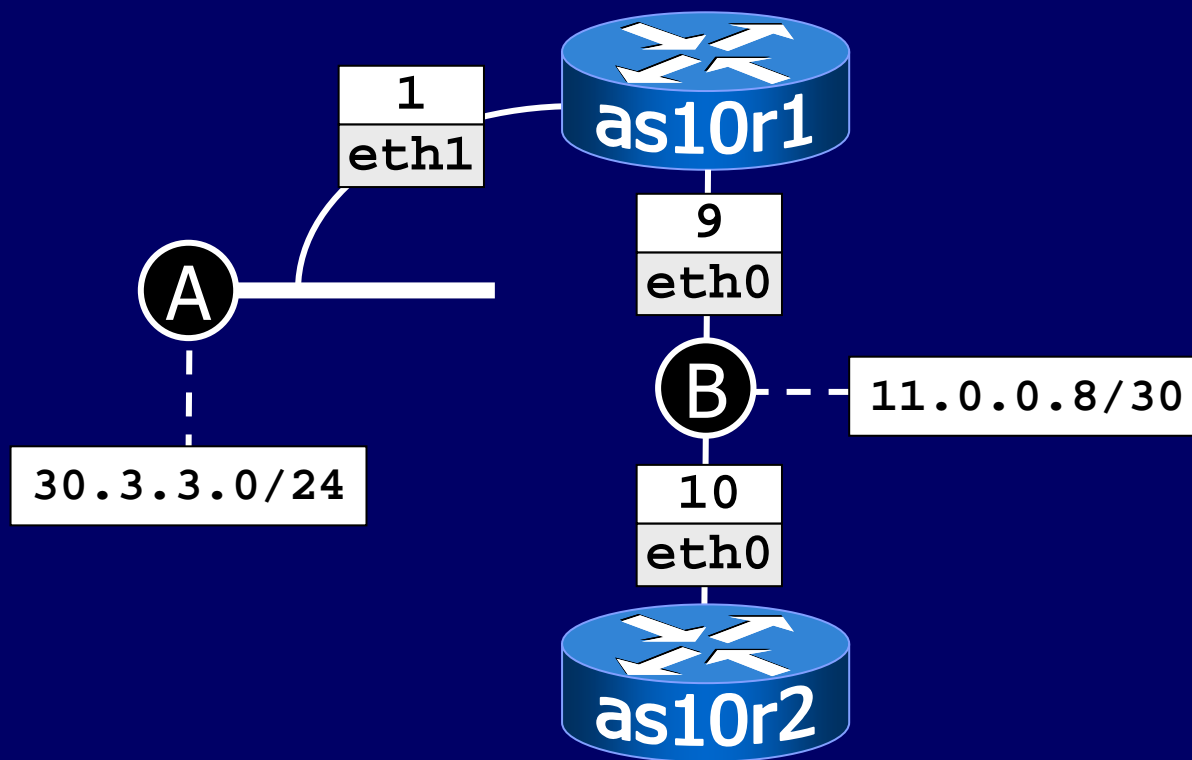




# My Own Lab, scene 1, take 1

## Network topology

- Sketch your planned topology before implementing it



### LEGEND

**(B)** collision domain name

11.0.0.8/30 network address

9 / eth0  
IP, last byte / Interface

**(A)** \*AN

# My Own Lab, scene 1, take 2

## Network topology

- ◆ A lab consists of a hierarchy of directories
- ◆ Each (even empty) directory represents a virtual machine

```
host machine
foo@host:~/lab$ ls
as10r1  as10r2  lab.conf
foo@host:~/lab$
```

- ◆ A lab consisting of two virtual machines (`as10r1`, `as10r2`)
- ◆ Check with `linfo`

- ◆ Link-level connections are described inside the file `lab.conf` (in the lab root)

# My Own Lab, scene 1, take 3

## Network topology

### ✦ lab.conf syntax

#### ■ vm[if]=cd

- vm: virtual machine name (e.g., as10r1)
- if: interface number (e.g., 0)
- cd: collision domain name (arbitrary string)

#### ■ vm[opt]=val

- opt: the name of a vstart option (e.g., mem)
- val: a value for that option

#### ■ Other optional items

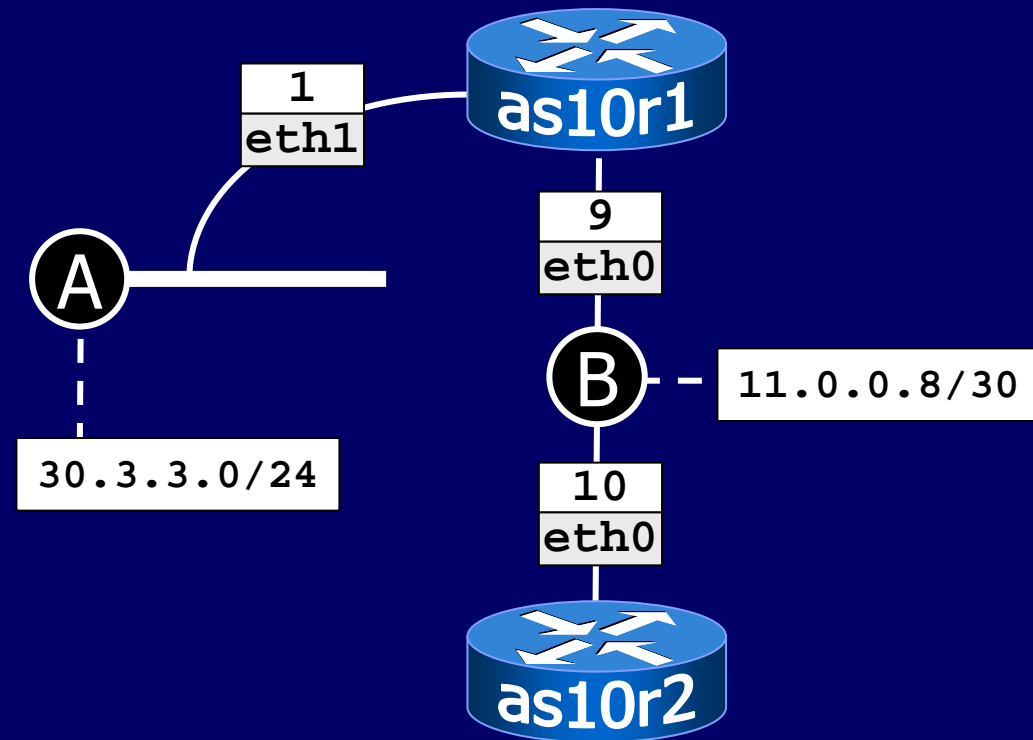
- Informational: LAB\_DESCRIPTION, LAB\_VERSION, LAB\_AUTHOR, LAB\_EMAIL, LAB\_WEB
- Explicit list of virtual machines (machines)

# My Own Lab, scene 1, take 4 Network topology

## ◆ Sample lab.conf

```
host machine
as10r1[0]=B
as10r1[1]=A

as10r2[0]=B
lab.conf
```



# My Own Lab, scene 2

## Assigning addresses, etc.

- ◆ On startup, virtual machine `vm` runs
  - `shared.startup`
  - `vm.startup`
- ◆ Interfaces may be configured inside `vm.startup`

### host machine

```
ifconfig eth0 11.0.0.9 netmask 255.255.255.252 broadcast 11.0.0.11 up  
/etc/init.d/zebra start  
as10r1.startup
```

- ◆ Network services may be started up inside `vm.startup` too...

# My Own Lab, scene 3

## Configuring network services

- On startup, Netkit copies the directory hierarchy in `vm/` (on the host) to `/` on virtual machine `vm`

```
host machine
foo@host:~/lab$ find .
.
./as10r1
./as10r1/root
./as10r1/root/this_is_a_file
./as10r2
./lab.conf
foo@host:~/lab$
```

```
as10r1
as10r1:~# pwd
/root
as10r1:~# ls
this_is_a_file
as10r1:~#
```

- Useful to alter routing software configuration files

# Ignition!

✦ To start up a lab:

```
host machine
foo@host:~$ cd lab
foo@host:~/lab$ lstart
```

or

```
host machine
foo@host:~$ lstart -d lab
```

✦ To crash a lab (use `lhalt` to halt):

```
host machine
foo@host:~$ cd lab
foo@host:~/lab$ lcrash
```

or

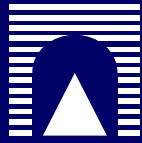
```
host machine
foo@host:~$ lcrash -d lab
```

✦ To get rid of temporary files (`.disk`, `.log`):

```
host machine
foo@host:~$ cd lab
foo@host:~/lab$ lclean
```

or

```
host machine
foo@host:~$ lclean -d lab
```



**UNIVERSITY OF ROMA TRE**

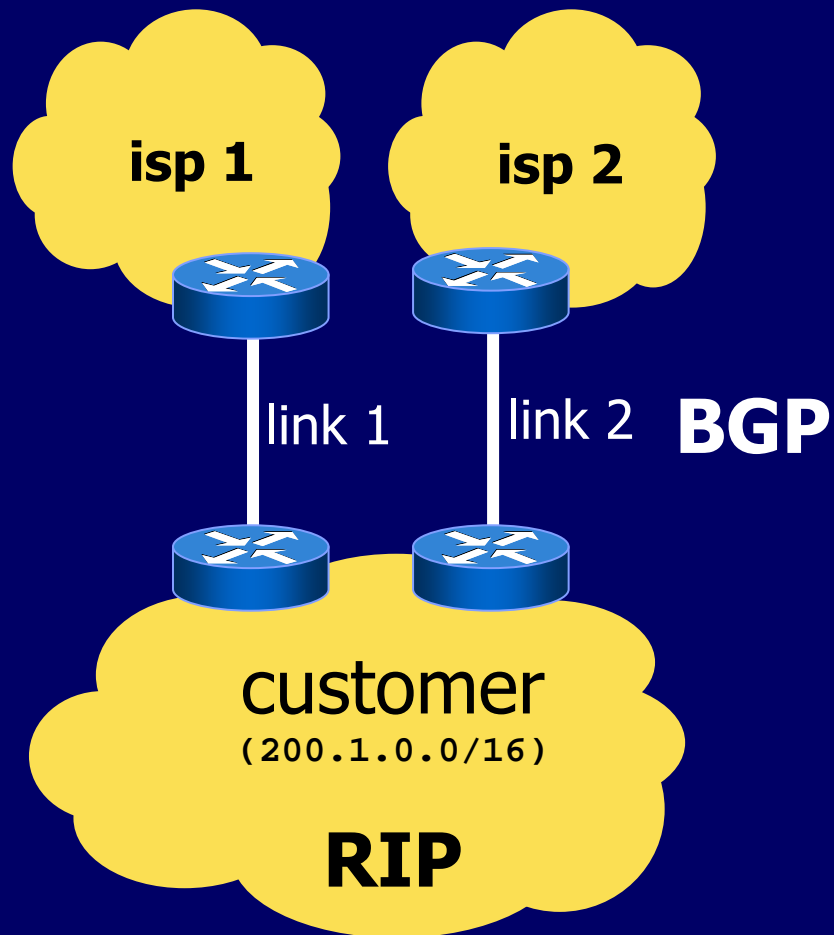
Department of Computer Science and Automation

# **Playing Around with Netkit**



# A sample lab

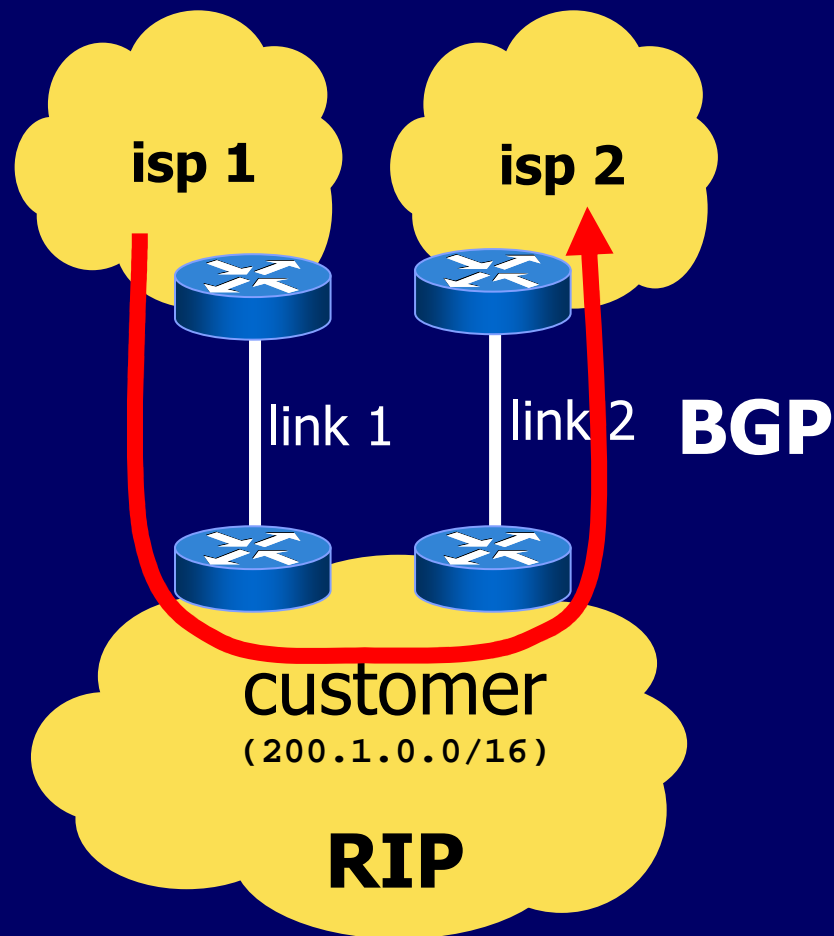
## • A multihomed network



# A sample lab

• A multihomed network, designed to:

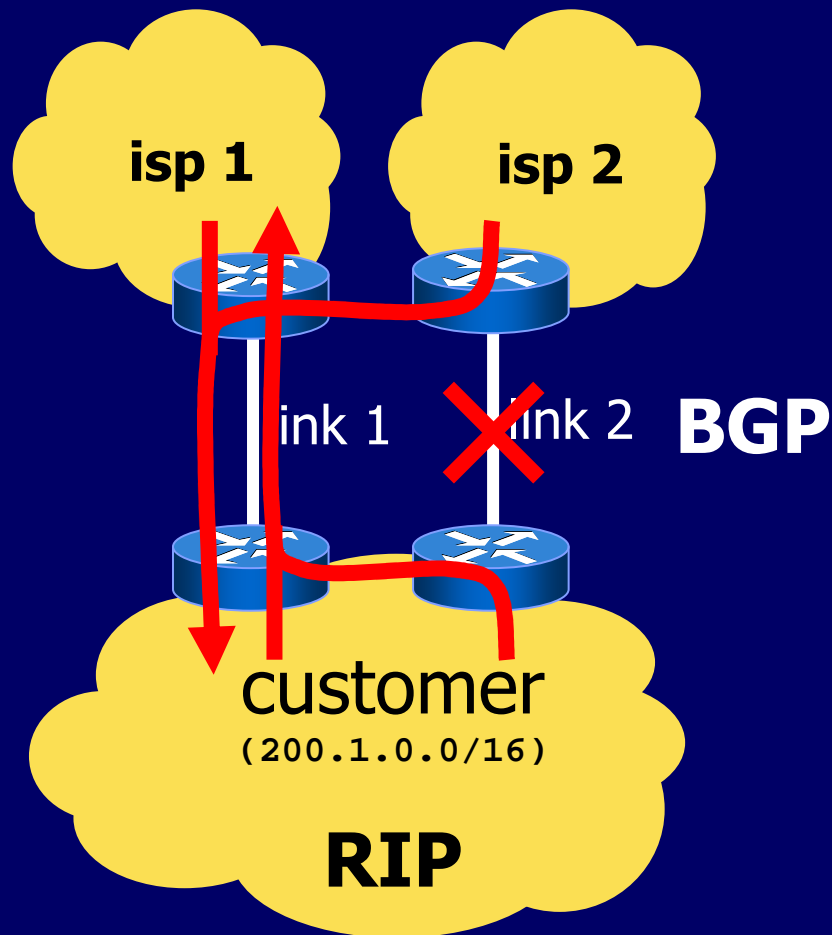
- Prohibit transit traffic



# A sample lab

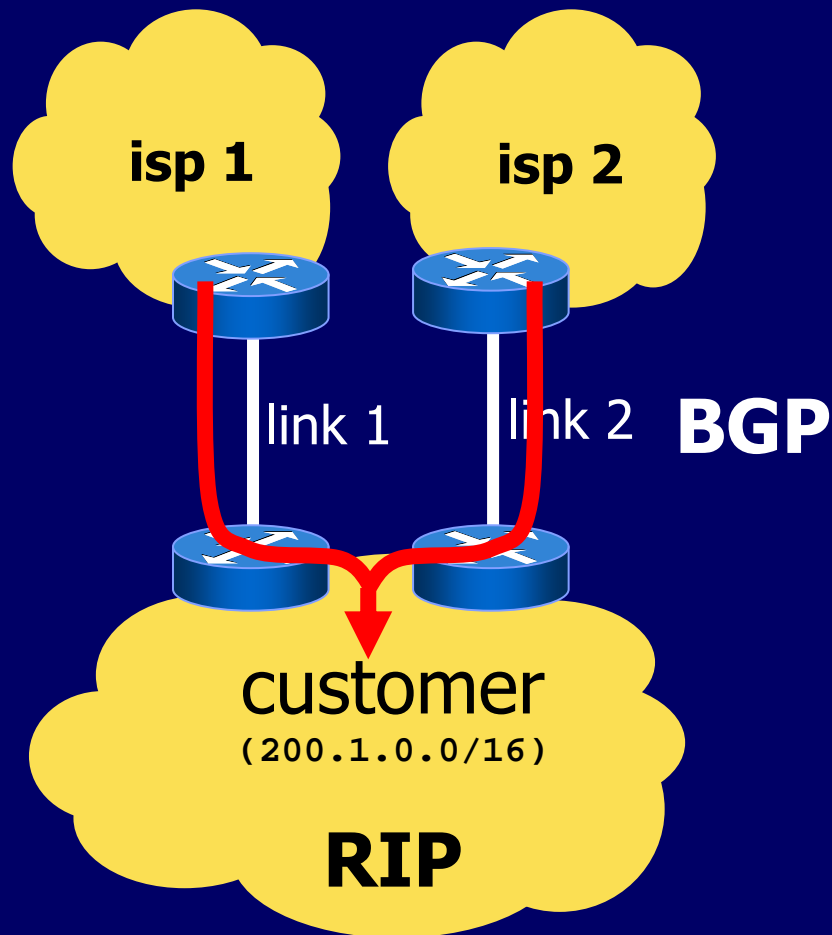
◆ A multihomed network, designed to:

- Prohibit transit traffic
- Be fault tolerant



# A sample lab

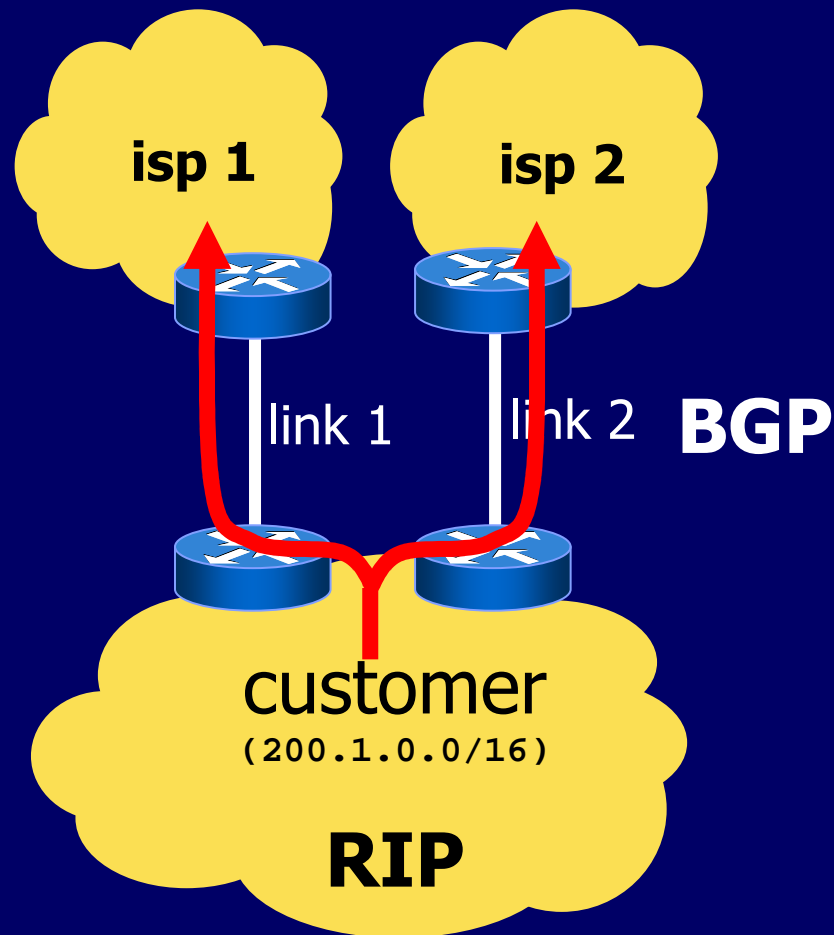
- A multihomed network, designed to:



- Prohibit transit traffic
- Be fault tolerant
- Perform loadsharing
  - inbound: by announcing /17s

# A sample lab

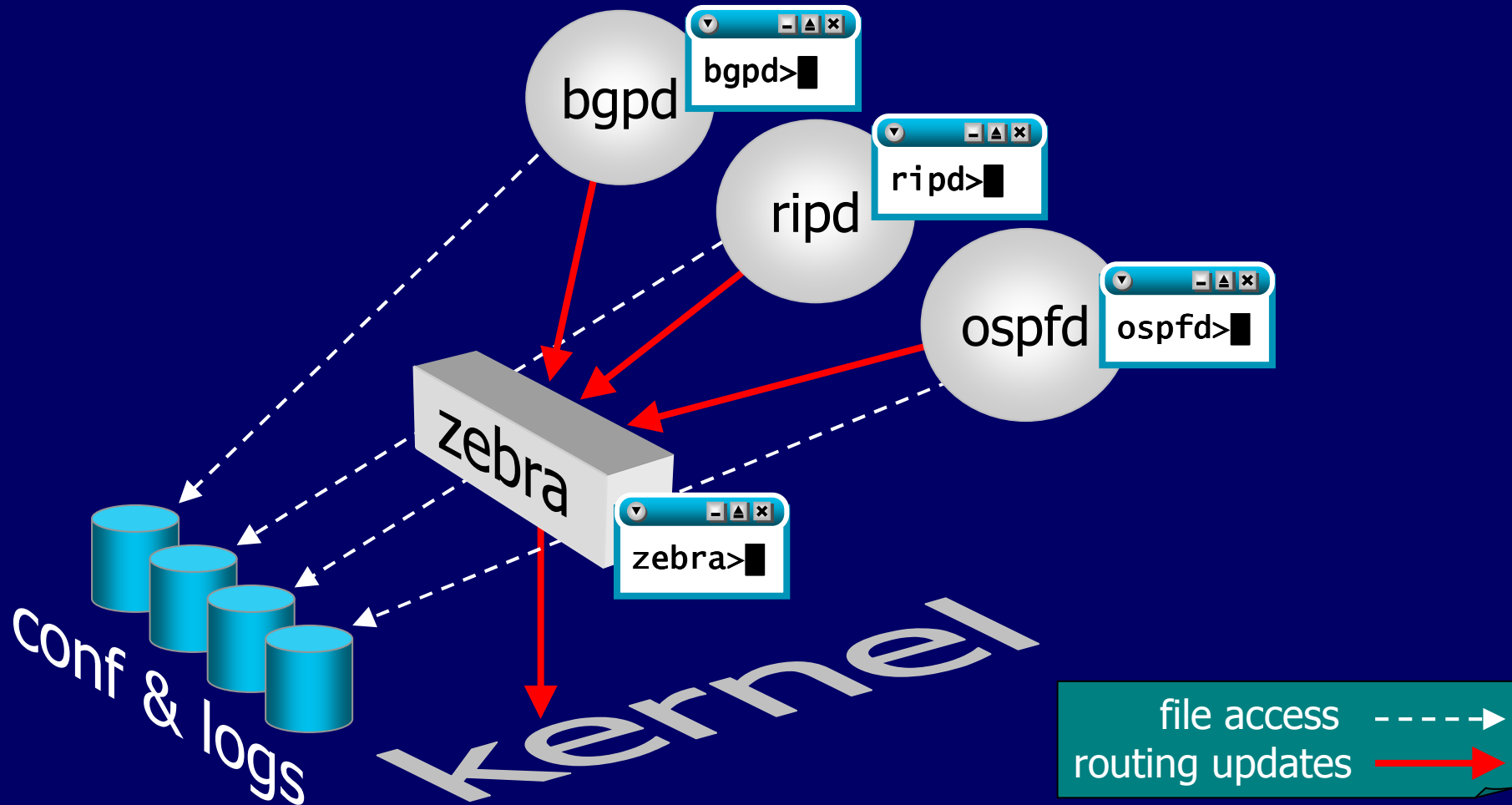
- ◆ A multihomed network, designed to:

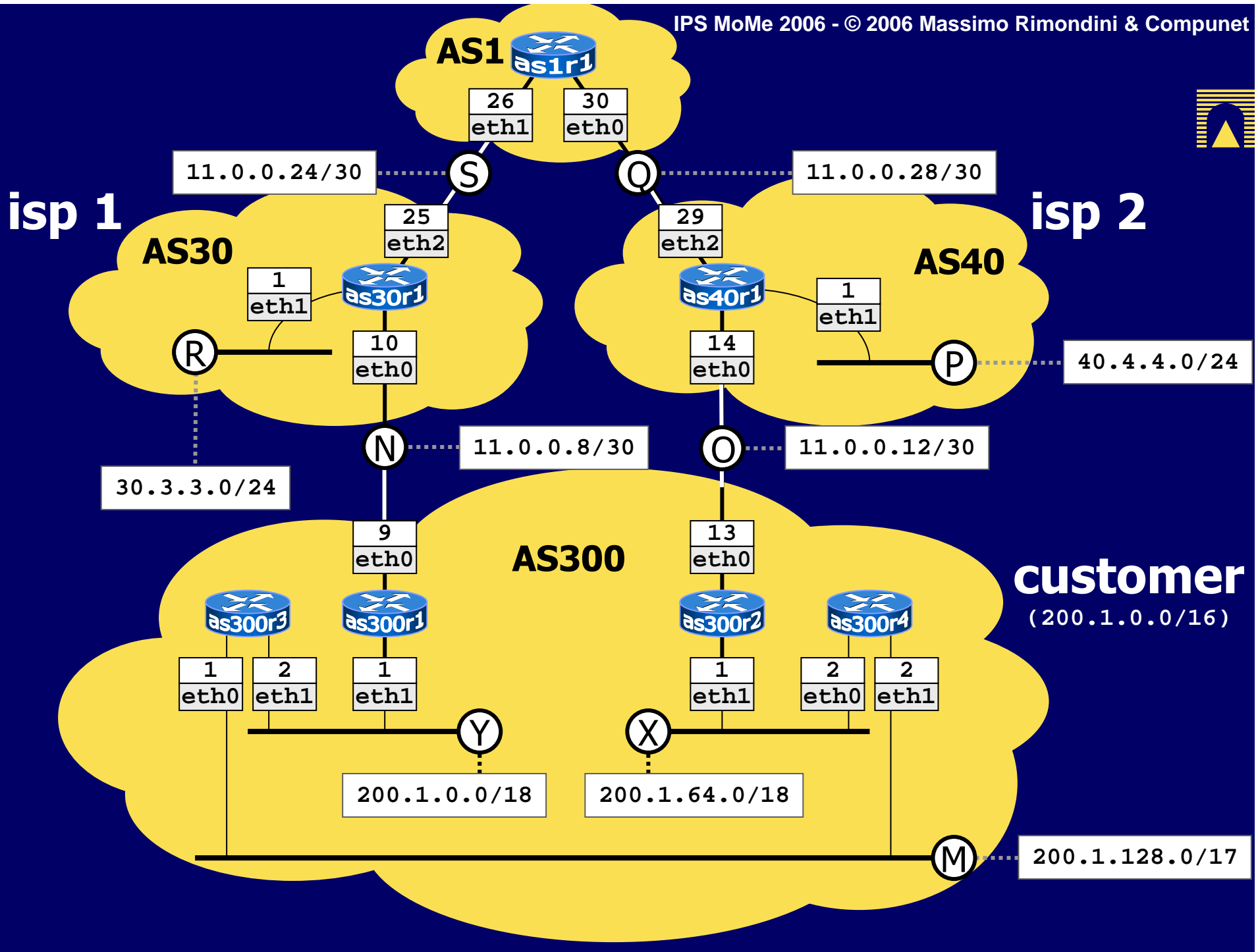


- Prohibit transit traffic
- Be fault tolerant
- Perform loadsharing
  - inbound: by announcing /17s
  - outbound: by nearest exit

# Making routers route traffic

- ◆ Routing is performed by the *zebra* software

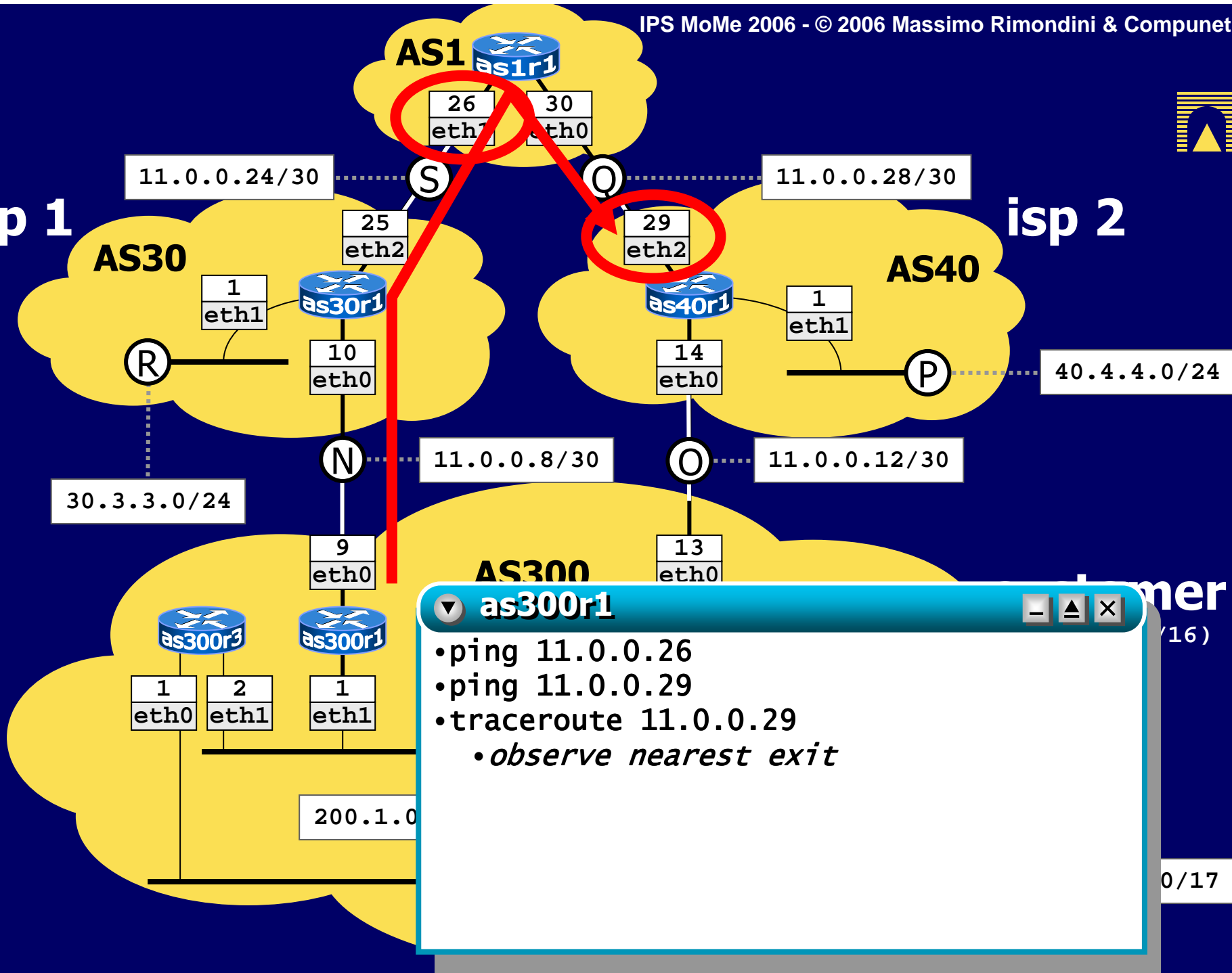






isp 1

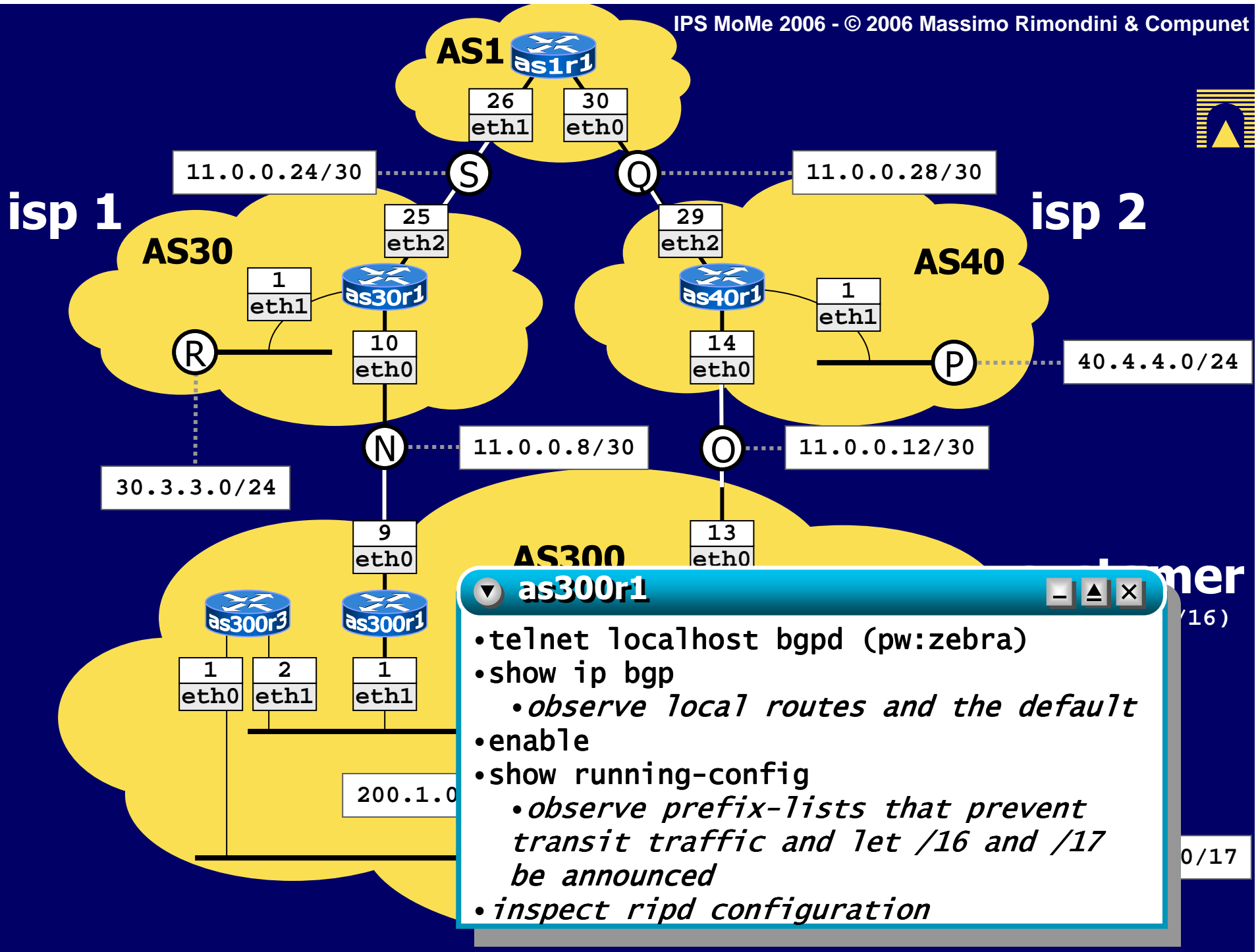
isp 2



```

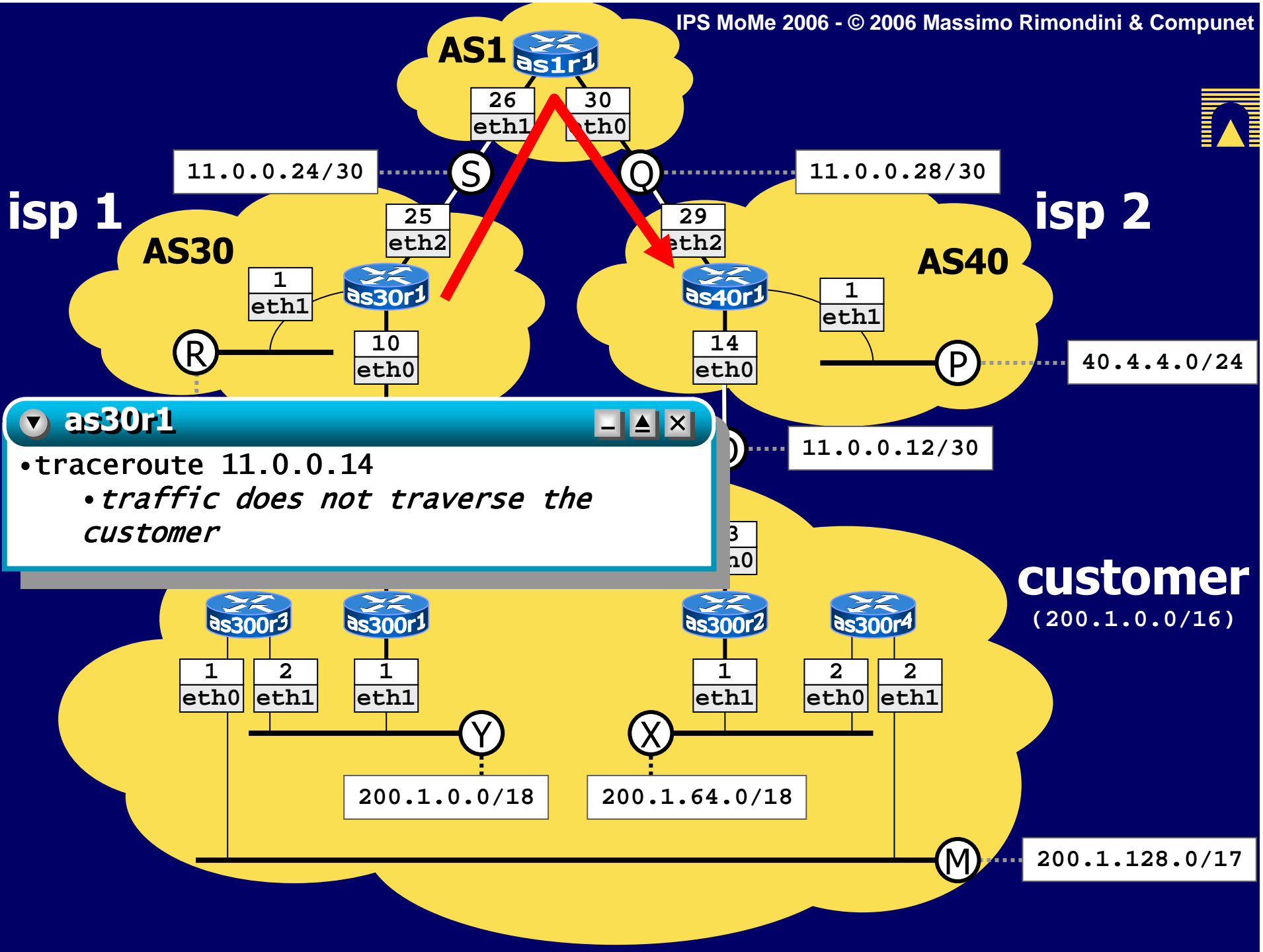
as300r1
•ping 11.0.0.26
•ping 11.0.0.29
•traceroute 11.0.0.29
  •observe nearest exit
  
```



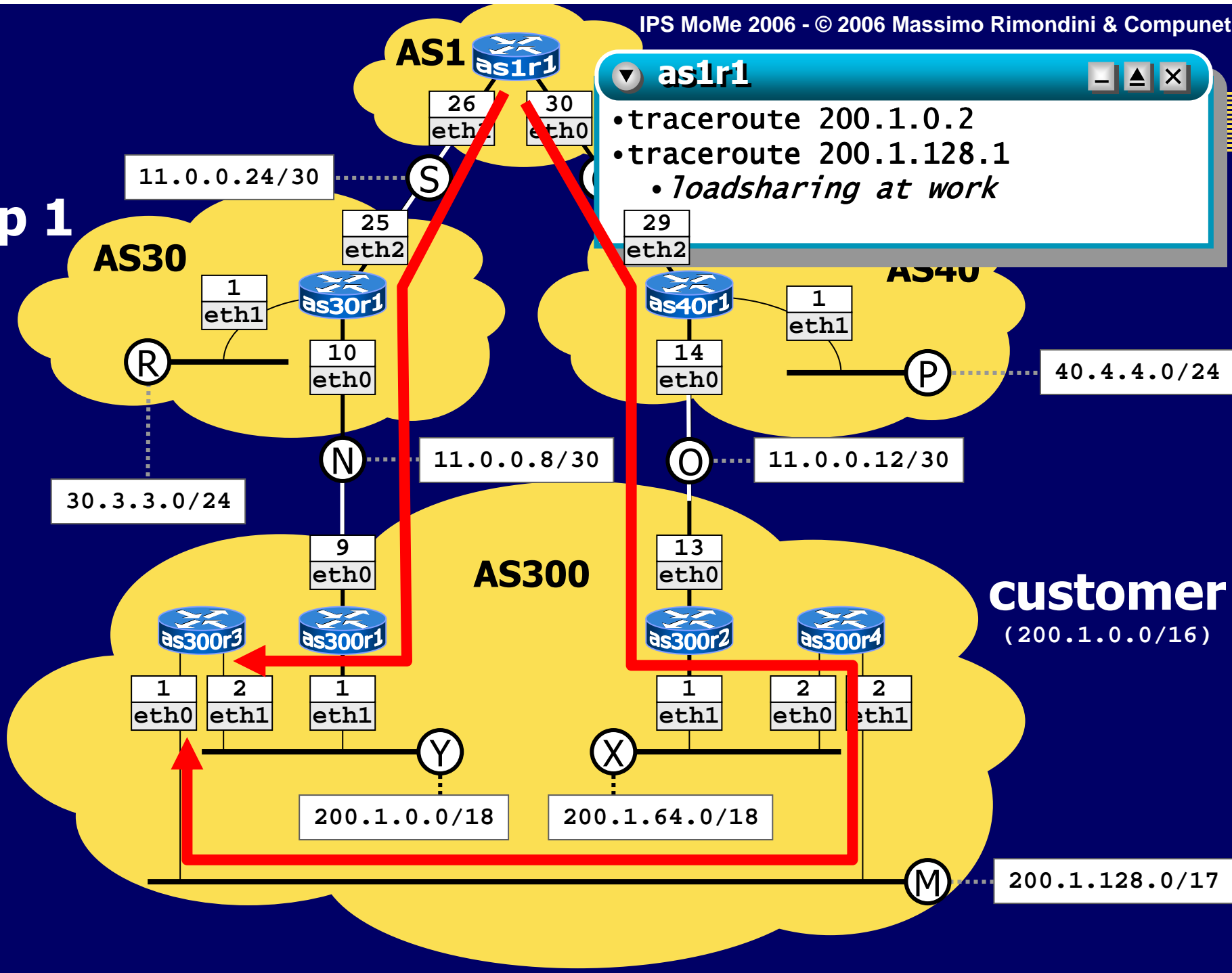


```

as300r1
•telnet localhost bgpd (pw:zebra)
•show ip bgp
  •observe local routes and the default
•enable
•show running-config
  •observe prefix-lists that prevent
  transit traffic and let /16 and /17
  be announced
•inspect ripd configuration
    
```



isp 1

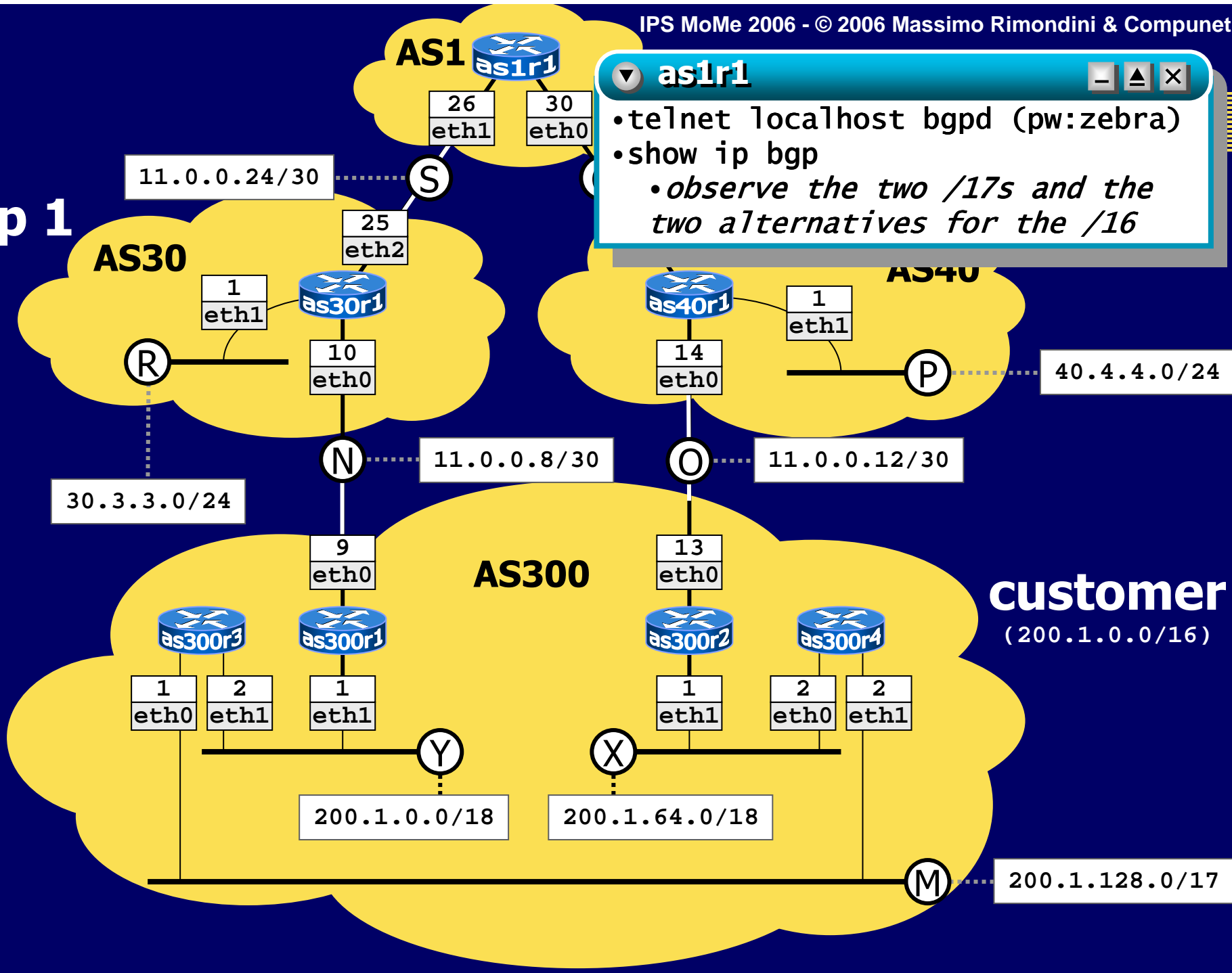


**as1r1**

- traceroute 200.1.0.2
- traceroute 200.1.128.1
- loadsharing at work

customer  
(200.1.0.0/16)

isp 1



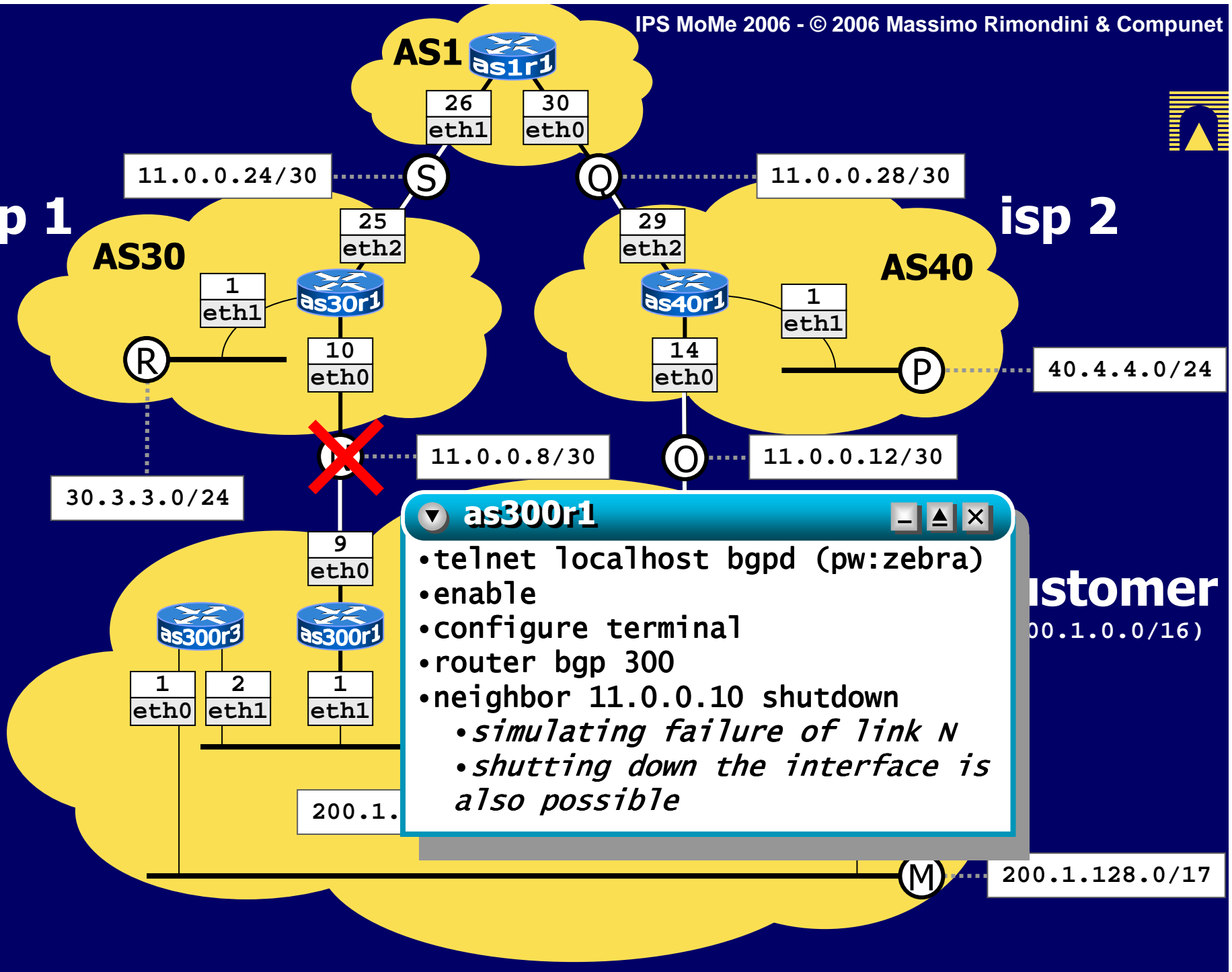
```
as1r1
•telnet localhost bgpd (pw:zebra)
•show ip bgp
  •observe the two /17s and the
  two alternatives for the /16
```

customer  
(200.1.0.0/16)



isp 1

isp 2



```

as30r1
•telnet localhost bgpd (pw:zebra)
•enable
•configure terminal
•router bgp 300
•neighbor 11.0.0.10 shutdown
  •simulating failure of link N
  •shutting down the interface is
  also possible
    
```

Customer  
00.1.0.0/16)

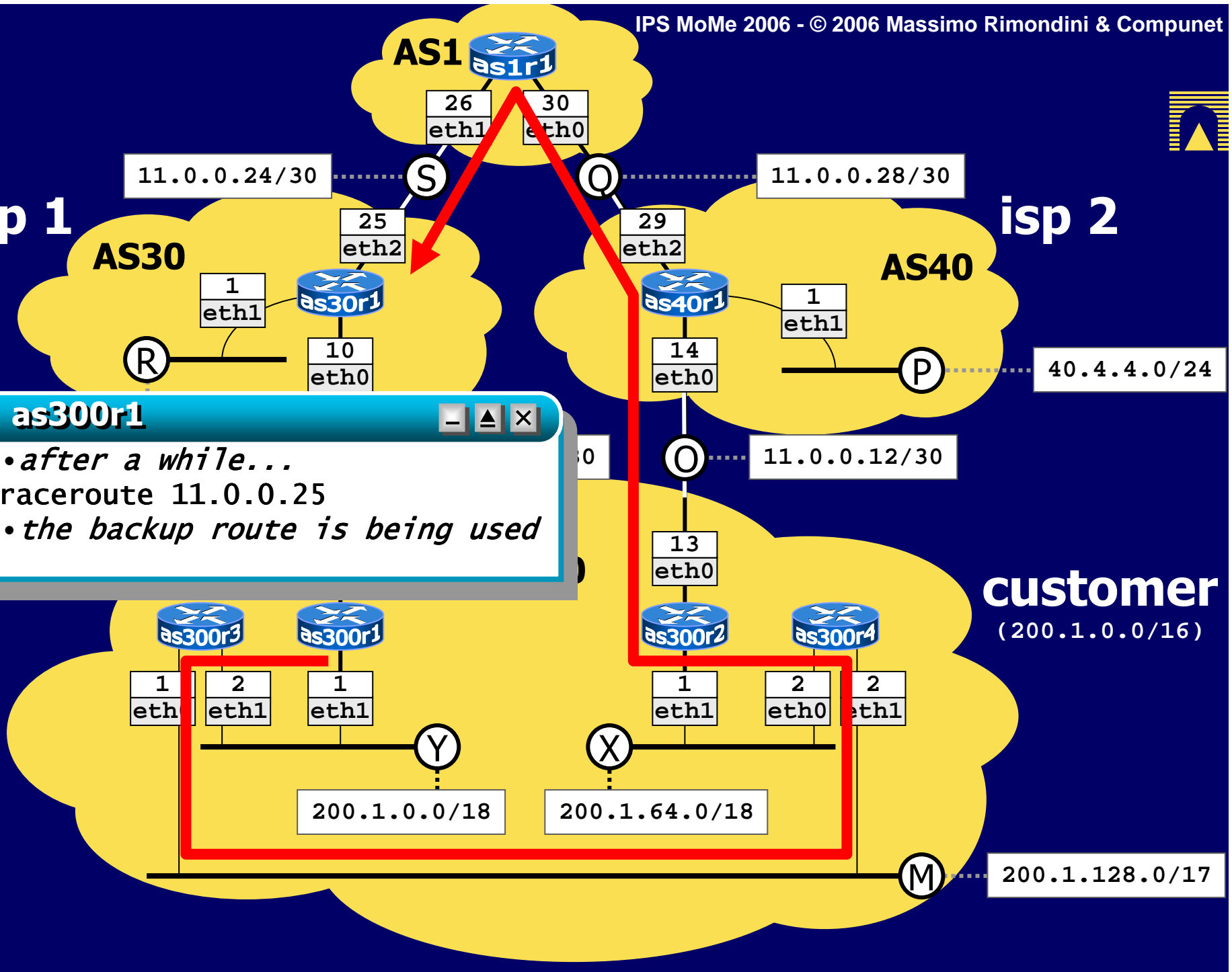


isp 1

isp 2

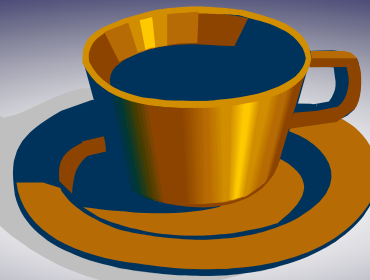
**as300r1**

- *after a while...*
- `traceroute 11.0.0.25`
- *the backup route is being used*



# Wanna play?

- ◆ Visit <http://www.netkit.org>
  - Other ready-to-use labs
  - Lecture slides
  - Netkit updates
  - Other resources (NetML)
- ◆ Forthcoming lab topology:



**Thank you!**