Università degli Studi Roma Tre
Dipartimento di Informatica e Automazione
Computer Networks Research Group

Netkit

The poor man's system for experimenting computer networking

Version	2.2
Author(s)	G. Di Battista, M. Patrignani, M. Pizzonia, M. Rimondini
E-mail	contact@netkit.org
Web	http://www.netkit.org/
Description	an introduction to the architecture, setup, and usage of Netkit

copyright notice

about computer networks

- computer networks are (tipically) quite complex
  - several devices (computers, routers, etc.)
  - several interfaces
  - several protocols running
  - physical interconnections originate complex topologies

how to perform experiments?

- performing experiments may be unfeasible
- the currently used network cannot be exploited for experiments
  - it hosts services that are critical for the company
  - it would be necessary to coordinate different departments of the company
- network equipments are expensive
  - sometimes, even for performing simple experiments, several equipments should be available in the same test bed

simulation vs. emulation

- emulation and simulation systems put at user's disposal a virtual environment that can be exploited for tests, experiments, measures
- simulation systems aim at reproducing the performance of a real-life system (latency time, packet loss, etc.)
  - e.g.: ns, real, …
- emulation systems aim at accurately reproducing the functionalities of a real-life system (configurations, architectures, protocols), with limited attention to performance

netkit: a system for emulating computer networks

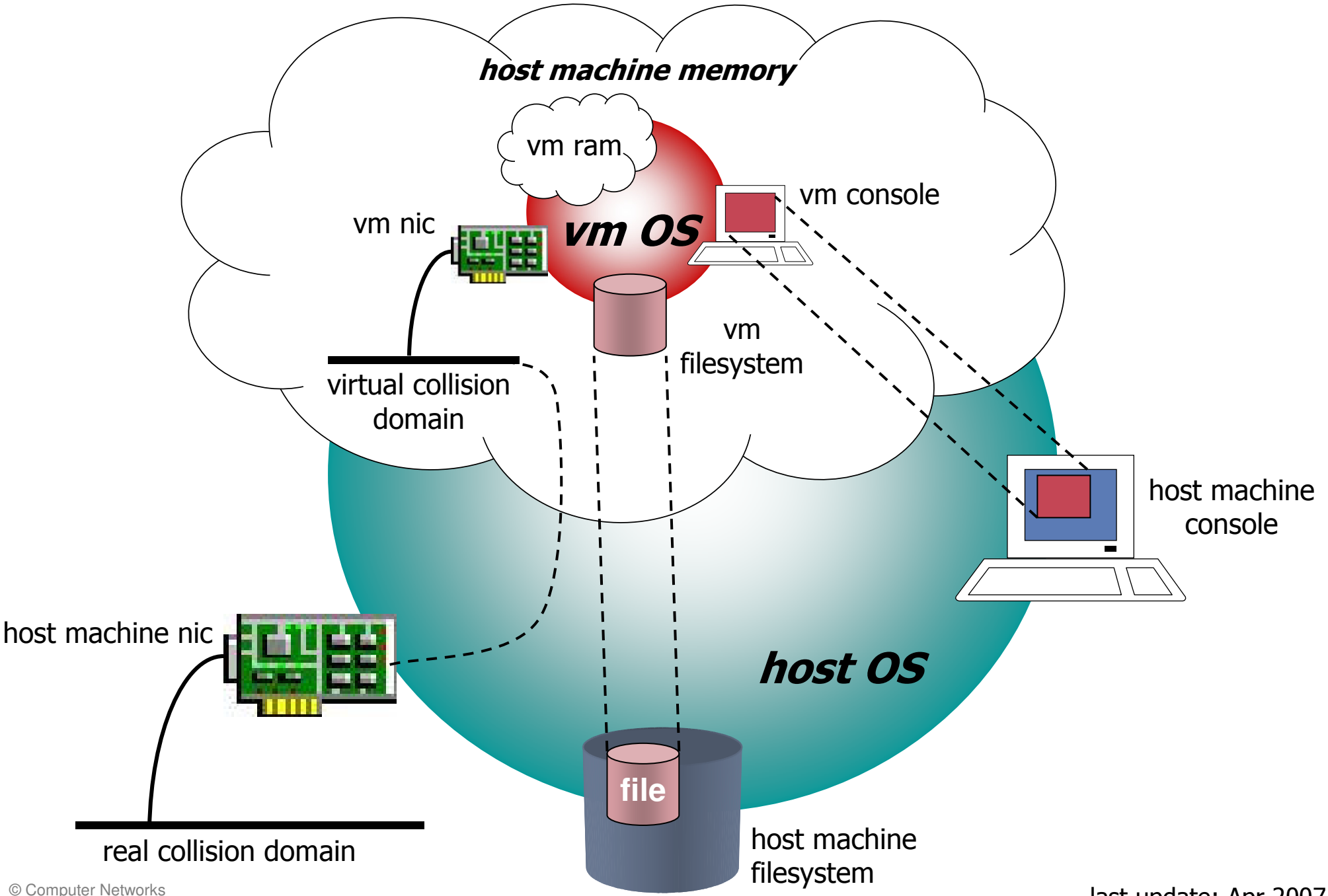
- based on uml (user-mode linux)
  - http://user-mode-linux.sourceforge.net/
- each emulated network device is a virtual linux box
  - a virtual linux box is one that is based on the uml kernel
- note: the linux os is shipped with software supporting most of the network protocols
  - hence, any linux machine can be configured to act as a bridge/switch or as a router

user-mode linux

- user-mode linux is a linux kernel (inner part of the linux os) that can be executed as a user process on a standard linux box
- a user-mode linux process is also called virtual machine (vm), while the linux box that hosts a virtual machine is called host machine (host)
- several virtual machines can be executed at the same time on the same host

uml virtual machines

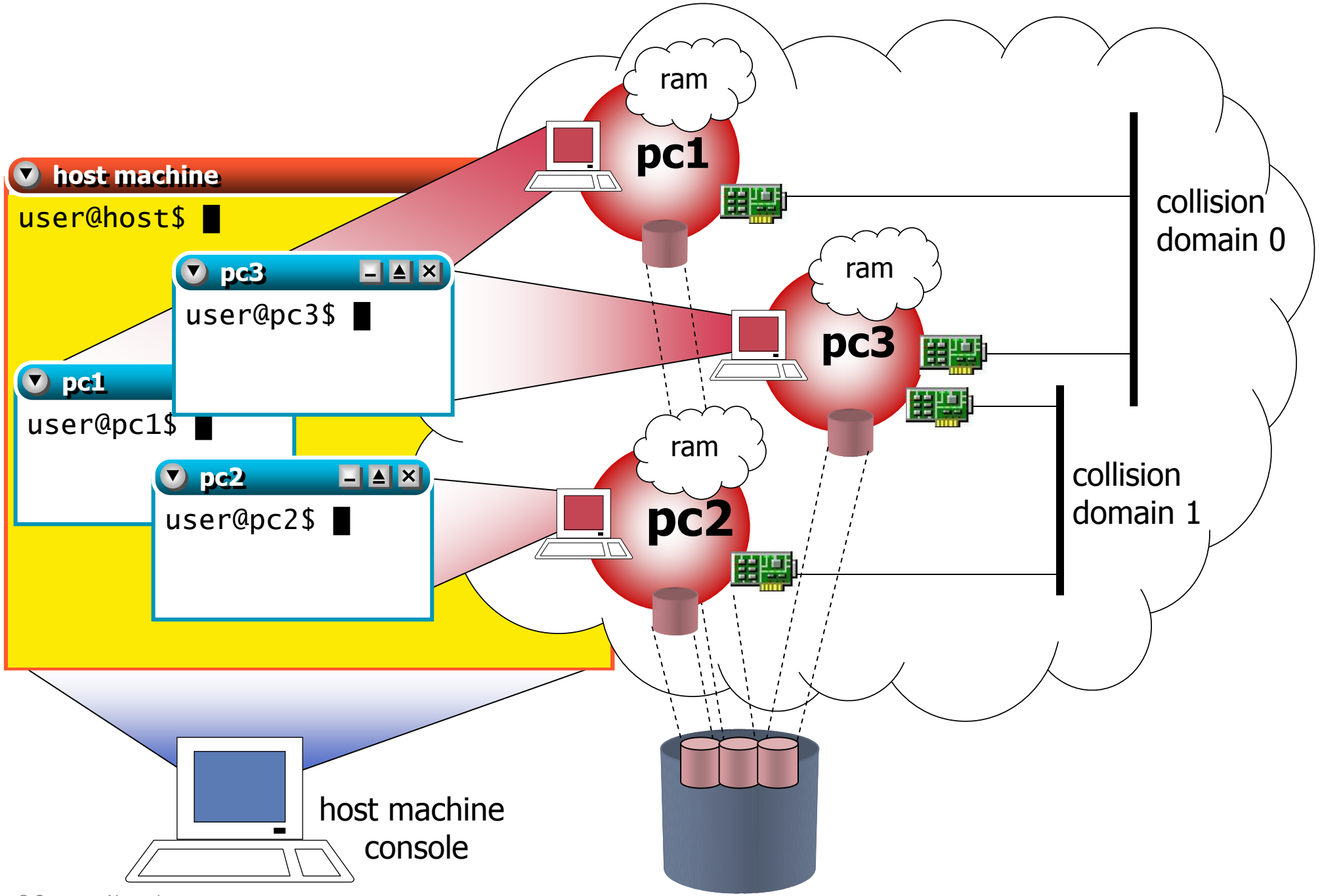
- each virtual machine has:
  - a console (a terminal window)
  - a memory ("cut" into the memory of the host)
  - a filesystem (stored in a single file of the host filesystem)
  - (one or more) network interfaces
- each network interface can be connected to a (virtual) collision domain
- each virtual collision domain can be connected to several interfaces



last update: Apr 2007

emulating a computer network using uml

- basic idea:
  - several virtual machines are created inside a single host machine
  - virtual machines are connected to virtual collision domains and thus can communicate with each other
- each virtual machine can be configured to play the role of a regular host, of a router, or even of a switch



what is netkit?

- a set of tools and commands that can be used to easily set up a virtual computer network
  - (most) commands are implemented as scripts
- a ready-to-use filesystem that is exploited as a pattern for creating the file system of each vm
  - most commonly used networking tools are already installed in this filesystem
- a uml kernel that is used as kernel for the virtual machines

Università degli Studi Roma Tre
Dipartimento di Informatica e Automazione
Computer Networks Research Group

setting up netkit

setting up netkit

- download at http://www.netkit.org/
- hw requirements:
  - i386 32 bit architecture
  - ≥ 600 MHz cpu
  - ~10 MB of memory for each vm (depending on the vm configuration)
  - ~600 MB of disk space + ~1-20 MB for each vm (depending on the usage of the vm)
- sw requirements
  - a linux box
  - works fine on many distributions, see http://www.netkit.org/status.html
  - standard, commonly available system tools (awk, lsof, etc.)

setting up netkit

- download the three files that make up the distribution
  - netkit-X.Y.tar.bz2
  - netkit-filesystem-FX.Y.tar.bz2 (warning: >100MB)
  - netkit-kernel-KX.Y.tar.bz2
- unpack them in the same location
  - tar xjf netkit-X.Y.tar.bz2
  - tar xjf netkit-filesystem-FX.Y.tar.bz2 (this may take a while; warning: decompressed size exceeds 600MB)
  - tar xjf netkit-kernel-KX.Y.tar.bz2

setting up netkit

- configure your shell to set the following environment variables
  - NETKIT_HOME must be set to the directory containing the decompressed version of netkit
  - "$NETKIT_HOME/bin" must be appended to the PATH
  - ":$NETKIT_HOME/man" must be appended to the MANPATH
  - for example (assuming bash is being used)
    - export NETKIT_HOME=~/netkit2
    - export PATH=$PATH:$NETKIT_HOME/bin
    - export MANPATH=:$NETKIT_HOME/man

setting up netkit

- you can check your configuration by entering the netkit directory…
  - cd $NETKIT_HOME
- …and running the check_configuration.sh script
  - ./check_configuration.sh
- if all the checks succeed, then you are ready to use netkit

Università degli Studi Roma Tre
Dipartimento di Informatica e Automazione
Computer Networks Research Group

using netkit

netkit commands

- netkit provides users with two sets of commands
  - v-prefixed commands (vcommands)
  - l-prefixed commands (lcommands)
- vcommands act as low level tools for configuring and starting up single virtual machines
- lcommands provide an easier-to-use environment to set up complex labs consisting of several virtual machines

netkit vcommands

- allow to startup virtual machines with arbitrary configurations (memory, network interfaces, etc.)
  - vstart: starts a new virtual machine
  - vlist: lists currently running virtual machines
  - vconfig: attaches network interfaces to running vms
  - vhalt: gracefully halts a virtual machine
  - vcrash: causes a virtual machine to crash
  - vclean: “panic command” to clean up all netkit processes (including vms) and configuration settings on the host machine

netkit lcommands

- ease setting up complex labs consisting of several virtual machines
  - lstart: starts a netkit lab
  - lhalt: gracefully halts all vms of a lab
  - lcrash: causes all the vms of a lab to crash
  - lclean: removes temporary files from a lab directory
  - linfo: provides information about a lab without starting it
  - ltest: allows to run tests to check that the lab is working properly

accessing the "external world" from a virtual machine

- two ways of doing this
  - the directory **`/hosthome`** inside a virtual machine directly points to the home directory of the current user on the real host
    - read/write access is allowed
  - vstart can automatically configure tunnels ("tap interfaces") by which a virtual machine can access an external network
    - see **`man vstart`** for more information

Università degli Studi Roma Tre
Dipartimento di Informatica e Automazione
Computer Networks Research Group

preparing a netkit lab

preparing a lab

- a netkit lab is a set of preconfigured virtual machines that can be started and halted together
- it may be implemented in (at least) two ways
  - by writing a single script **`lab-script`** that invokes **`vstart`** for each virtual machine to be started
  - by setting up a standard netkit lab that can be launched by using the lcommands (recommended)

a netkit lab as a single script

- a script (e.g., **`lab-script`**) invokes **`vstart`** with some options to start up each virtual machine
- by using the **`--exec`** option of **`vstart`**, the same script can be invoked inside vms (e.g., in order to automatically configure network interfaces)
- a check inside **`lab-script`** can be used to test if we are in the real host or inside a vm

a netkit lab as a single script

- example

```
vstart pc1 --eth0=0 --eth1=1 --exec=this_script
vstart pc2 --eth0=0 --exec=this_script
vstart pc3 --eth0=1 --exec=this_script
if [ `id -u` == "0" ]; then
        case "$HOSTNAME" in
                pc1)
                        ifconfig eth0 10.0.0.1 up
                        ifconfig eth1 10.0.0.2 up;;
                pc2)
                        ifconfig eth0 10.0.0.3 up;;
                pc3)
                        ifconfig eth0 10.0.0.4 up;;
        esac
fi
```

netkit labs using lcommands

- a standard netkit lab is a directory tree containing:
  - a lab.conf file describing the network topology
  - a set of subdirectories that contain the configuration settings for each virtual machine
  - .startup and .shutdown files that describe actions performed by virtual machines when they are started or halted
  - [optionally] a lab.dep file describing dependency relationships on the startup order of virtual machines
  - [optionally] a _test directory containing scripts for testing that the lab is working correctly

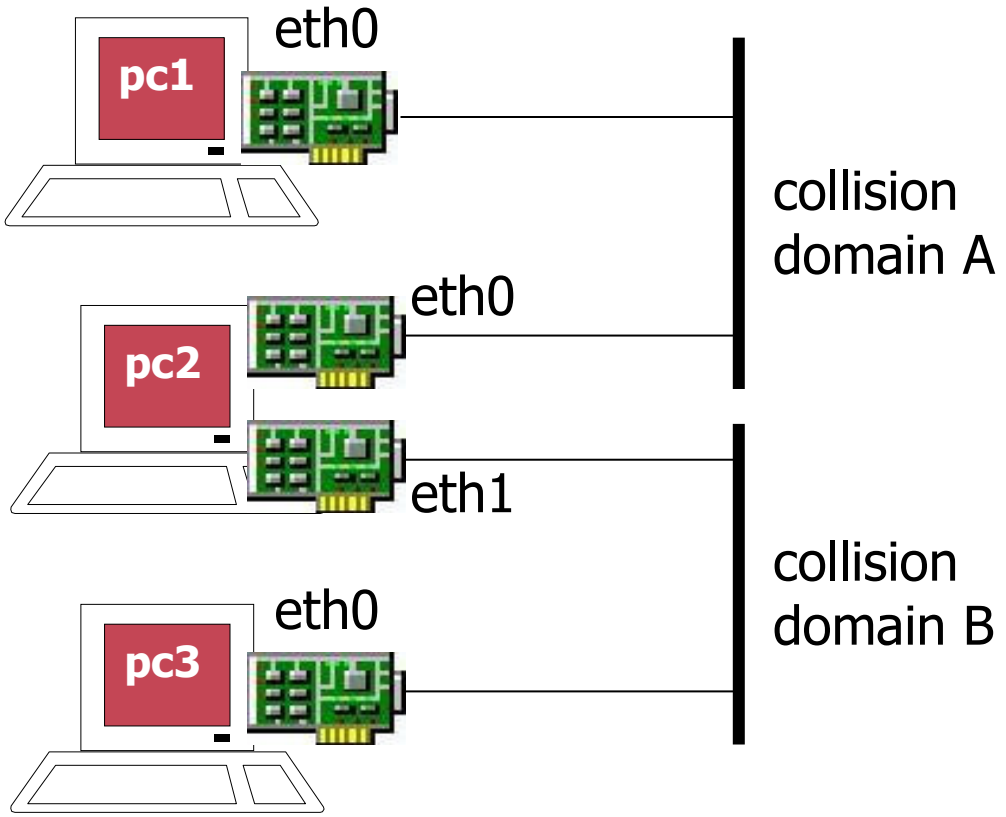
lab.conf

- this file describes
  - the settings of the vms that make up a lab
  - the topology of the network that interconnects the vms of the lab
- list of **`machine[arg]=value`** assignments
  - **`machine`** is the name of the vm (e.g., **`pc1`**)
  - if **`arg`** is an integral number (say ***`i`***), then **`value`** is the name of the collision domain to which interface **`eth`*`i`*** should be attached
  - if **`arg`** is a string, then it must be the name of a **`vstart`** option and **`value`** is the argument (if any) to that option

lab.conf

- example

```
pc1[0]=A

pc2[0]=A
pc2[1]=B
pc2[mem]=256

pc3[0]=B
```



`pc2` is equipped with 256MB of (virtual) memory

lab.conf

- other optional assignments
  - **`machines="pc1 pc2 pc3…"`**: explicitly declare the virtual machines that make up the lab
    - by default, the existence of a subdirectory **`vm_name`** in the lab directory implies that a virtual machine **`vm_name`** is started
  - **`LAB_DESCRIPTION`**
  - **`LAB_VERSION`**
  - **`LAB_AUTHOR`**
  - **`LAB_EMAIL`**
  - **`LAB_WEB`**

  } descriptive information displayed when the lab is started

lab subdirectories

- netkit starts a virtual machine for each subdirectory, with the same name of the subdirectory itself
  - unless **`lab.conf`** contains a **`machines=`** statement
- the contents of subdirectory **`vm`** are mapped (=copied) into the root (**`/`**) of **`vm`**'s filesystem
  - for example, **`vm/foo/file.txt`** is copied to **`/foo/file.txt`** inside virtual machine **`vm`**
  - this only happens the 1st time the vm is started; in order to force the mapping you have to remove the vm filesystem (**`.disk`** file)

startup and shutdown files

- shell scripts that tell virtual machines what to do when starting up or shutting down
- they are executed inside virtual machines
- **`shared.startup`** and **`shared.shutdown`** affect all the virtual machines
- upon startup, a vm named **`vm_name`** runs
  - **`shared.startup`**
  - **`vm_name.startup`**
- upon shutdown, a vm named **`vm_name`** runs
  - **`vm_name.shutdown`**
  - **`shared.shutdown`**

startup and shutdown files

- a typical usage of a **`.startup`** file is to configure network interfaces and/or start network services
- sample of **`vm_name.startup`**

```
ifconfig eth0 10.0.0.1 up
/etc/init.d/zebra start
```

lab.dep

- multiple virtual machines can boot at once (parallel startup)
  - `-p` option of lstart
- the startup order of virtual machines can be influenced by establishing dependencies
  - e.g., "pc3 can only boot after pc2 and pc1 are up and running"
- a lab.dep file inside the lab directory describes dependencies and automatically enables parallel startup
  - file format is similar to that of a Makefile
  - example

```
pc3: pc1 pc2
```

launching/stopping a lab

- ***lcommand*** **-d <lab_directory> [machine...]**
- or
  - enter the lab directory (**cd lab_directory**)
  - ***lcommand***
- where ***lcommand*** can be one of the following:
  - **lstart**, to start the lab
  - **lhalt**, to gracefully shut down the virtual machines of a lab
  - **lcrash**, to suddenly crash the virtual machines of a lab
- optionally, a list of **machine** names can be given on the command line, in which case only those machines will be affected by the command

removing temporary files

- a running lab creates some temporary files inside both the current directory and the lab directory
- to get rid of them all, use **`lclean`** after the lab has been halted/crashed
  - notice: **`lclean`** also removes virtual machine filesystems (**`.disk`** files); do not use it if you are going to launch your lab again using the same filesystems

ltest

- makes it easier to check that distributed labs work properly
- ltest starts a lab and dumps information about each virtual machine **`vm`**
  - the output goes into **`_test/results/vm.default`**
- [optionally] a script **`_test/vm.test`** may contain additional commands to be run inside **`vm`** in order to dump other information
  - the output goes into **`_test/results/vm.user`**

ltest

- sample of **`vm.default`** file

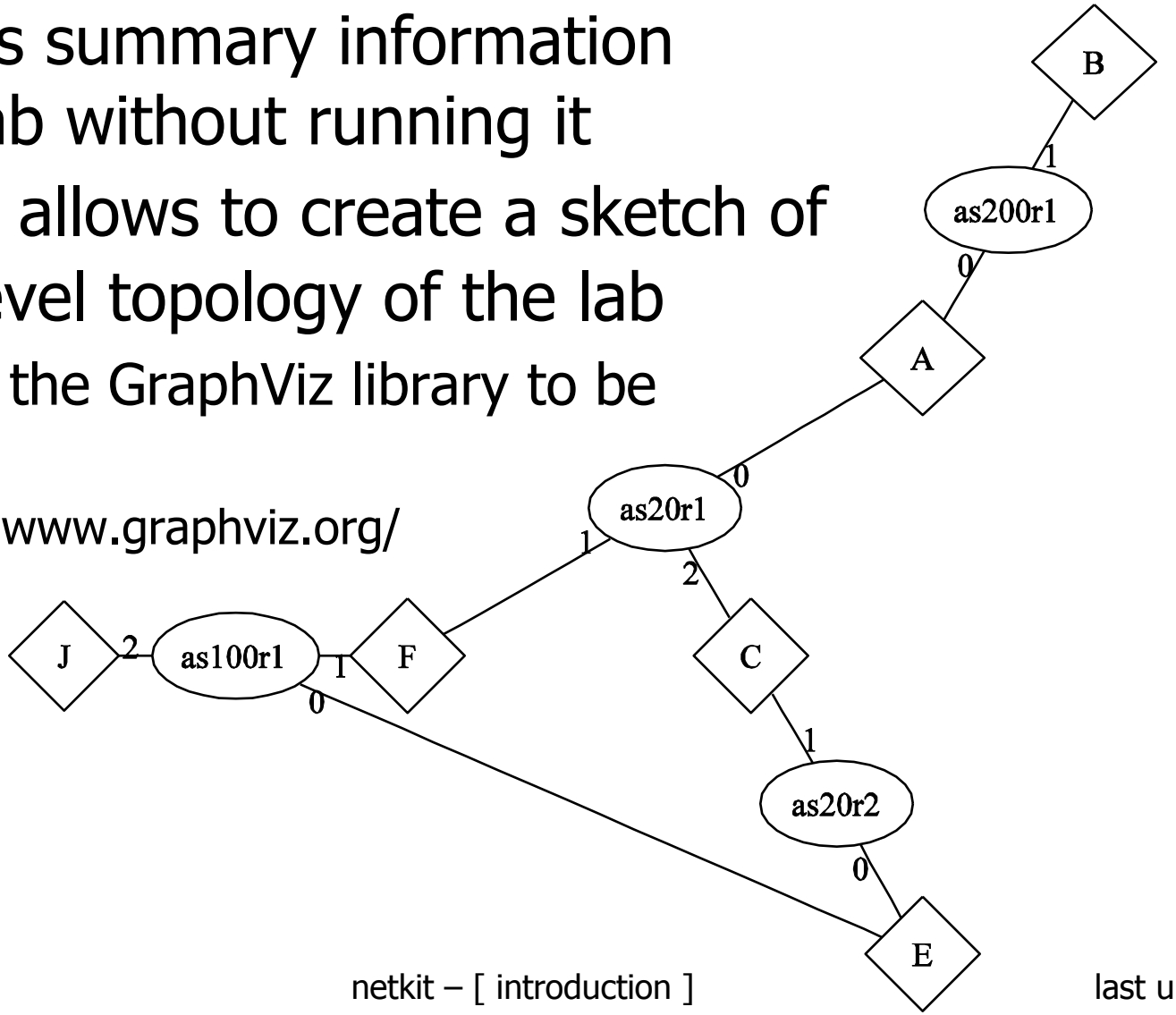
```
[INTERFACES]

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1


[ROUTE]

Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt
 Iface

[LISTENING PORTS]

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

[PROCESSES]

  UID COMMAND
    0 init [2]
    0 [ksoftirqd/0]
    0 [events/0]
.........
```

ltest

- when preparing a lab
  - launch ltest to dump lab information
  - move files **`_test/results/*`** to a subdirectory **`_test/results/good`**
- when testing a lab
  - launch ltest to dump lab information
  - compare (e.g., using **`diff`**) files **`_test/results/*`** with **`_test/results/good/*`**
  - check if they all match

getting information about a lab

- linfo prints summary information about a lab without running it
- option **-m** allows to create a sketch of the link-level topology of the lab
  - requires the GraphViz library to be installed
    - http://www.graphviz.org/



more information

- further information can be found…
  - …inside netkit man pages (you can start from **`man netkit`**)
  - …on the web site http://www.netkit.org/