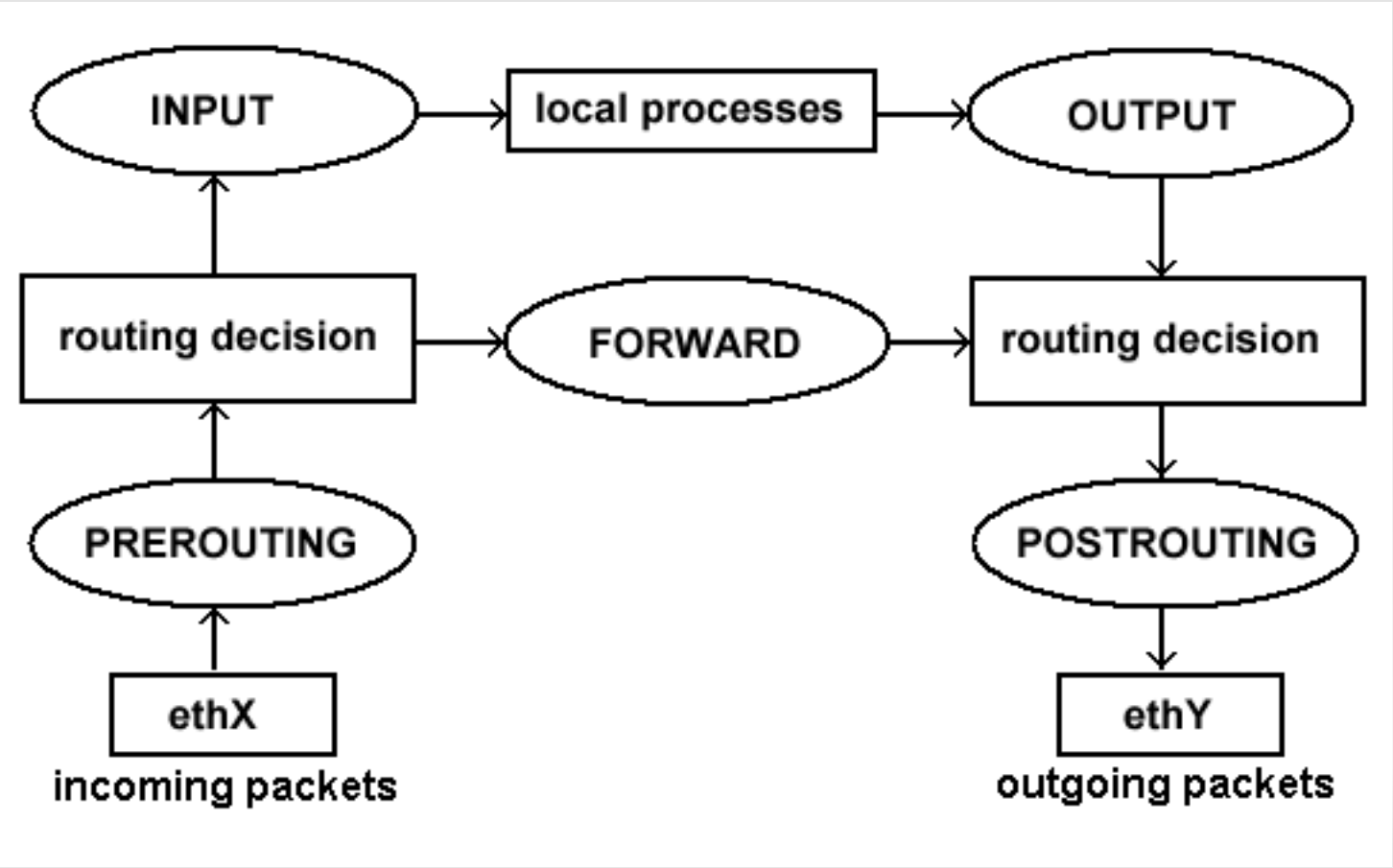
Netfilter & Packet Dropping

- Netfilter provides a set of hooks is several points of the kernel network stack.
- The hooks can be exploited to define custom functions for manipulating IP packets
  - Dropping
  - Manipulation of header fields
  - Etc.
- The hooks are triggered by the kernel after the execution of the functions that implement the network procedures

Netfilter Architecture

- An incoming IP packet travels in the kernel following a path



Netfilter Architecture

- Kernel path for incoming packets

1. Sanity checks (i.e., not truncated, IP checksum OK, etc)
2. Routing decision (it decides whether the packet is destined for another interface, or a local process)
   - Local process: the netfilter framework is called again for the NF_IP_LOCAL_IN hook
   - Another interface: the netfilter framework is called for the NF_IP_FORWARD hook
3. Final step (the packet passes a final netfilter hook the NF_IP_POST_ROUTING hook)

Netfilter Architecture

- When a hook is triggered, a customized function can manipulate the packet content
- Kernel modules can register to listen at any of the hooks described in the previous slide
- After manipulating a packet, the module returns a code to the claling function:
  - NF_ACCEPT: continue traversal as normal
  - NF_DROP: drop the packet; don't continue traversal
  - NF_STOLEN: stole the packet from the path
  - NF_QUEUE: queue the packet (for userspace handling)
  - NF_REPEAT: call this hook again

Netfilter & Iptables

- The iptables tool has been developed over the netfilter framework
- Kernel modules can register a new table, and ask for a packet to traverse a given table
- Hooks registered with netfilter

Prerouting	Forwarding	Postrouting
Conntrack	Mangle	Mangle
Mangle		Src NAT
Dst NAT	Filter	Conntrack
QDisc		

Registration of filtering functions

- Structure containing the function handle:

  static struct nf_hook_ops netfilter_ops_pre;

- Customized attributes of the hook:

  netfilter_ops_pre.hook = hook_pre_routing;

  netfilter_ops_pre.pf = PF_INET;

  netfilter_ops_pre.hooknum = NF_INET_PRE_ROUTING;

  netfilter_ops_pre.priority = NF_IP_PRI_FIRST;

  Registration of the hook

  ret = nf_register_hook(&netfilter_ops_pre);

Registration of filtering functions

- hook_pre_routing is the function implementing the packet filtering
- PF_INET: Internet Protocol Family
- NF_INET_PRE_ROUTING: the function is triggered before the routing decision
- NF_IP_PRI_FIRST: the registered function has the highest priority of execution
- The unregistration is performed using the following function:

```
nf_unregister_hook(&netfilter_ops_pre);
```

Kernel Modules

- The kernel module implementing the filtering function needs to be cross-compiled for the um architecture
- The Makefile is very similar to the Makefile used to compile kernel modules
- In addition to indicating the directory which contains the headers and the objects of the Netkit kernel, it is necessary to define the architectures of the host and target machines
  - ARCH=um
  - SUBARCH=i386

Kernel Modules

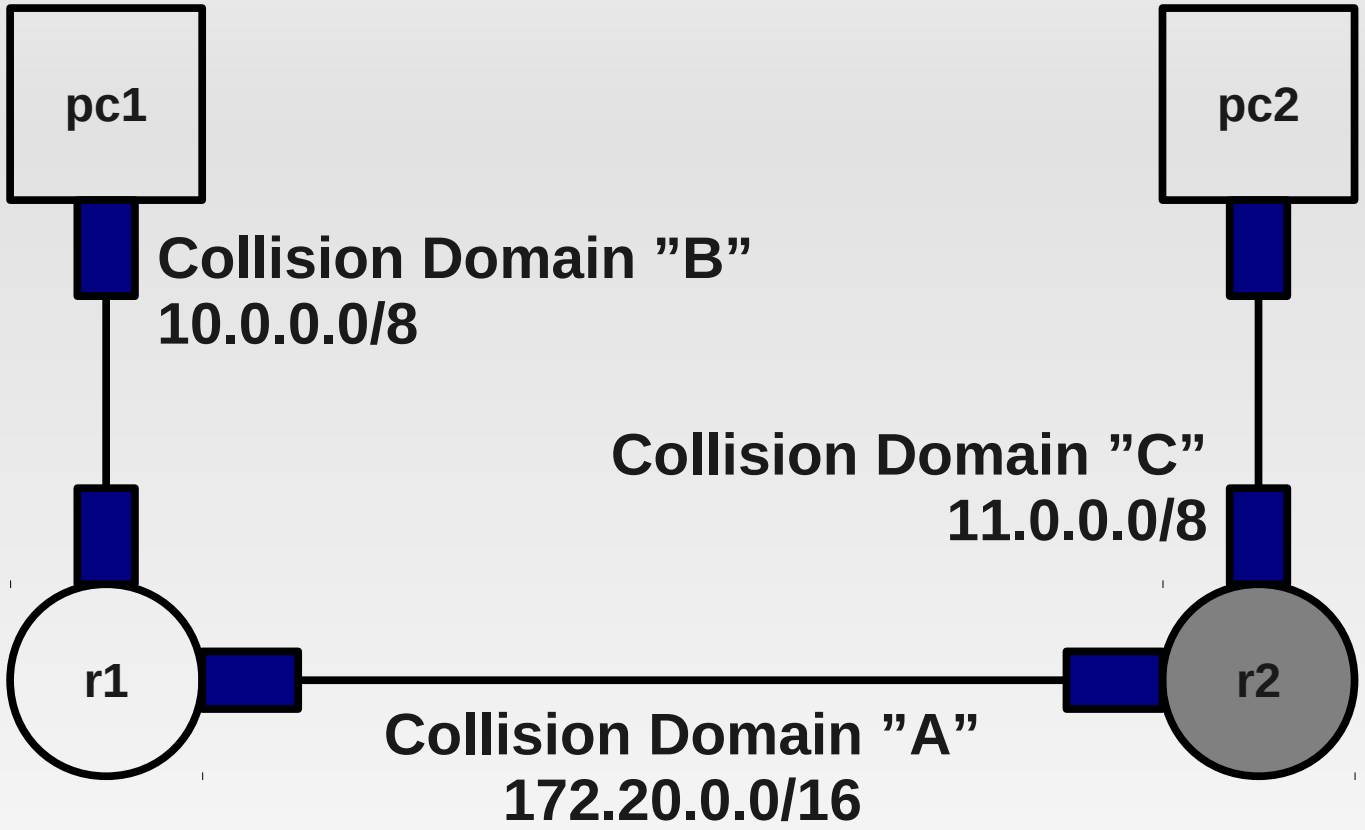
- Makefile for the pkt_drop module

```
obj-m += pkt_drop.o

KERNELPATH="path/to/kernel/src"

all:

    make -C $(KERNELPATH) M=$(shell pwd) ARCH=um
    SUBARCH=i386 modules

clean:

    make -C $(KERNELPATH) M=$(shell pwd) ARCH=um
    SUBARCH=i386 clean
```

Example

- Packet dropping:
  - A simple linux kernel module which defines a function that drops data packets before performing the routing decision.
  - The function is registerd as a PREROUTING hook
  - Note that some of the auxiliary functions defined by the kernel to access the header fields may NOT work
- See the code pkt_drop.c

Example


pc1
pc2
Collision Domain ”B”
10.0.0.0/8
Collision Domain ”C”
11.0.0.0/8
r1
r2
Collision Domain ”A”
172.20.0.0/16

Example

- The command **insmod** is usually used to load kernel modules

  insmod pkt_drop.ko drop_deg=5

  (5 out of 10 ICMP echo reqs will be discarded)

- The command **rmmod** is usually used to unload kernel modules

  rmmod pkt_drop

- modprobe is an alternative command to load and unload modules
  - modprobe -i pkt_drop.ko drop_deg=5
  - modprobe -r pkt_drop.ko