



# 网络安全原理与技术

李军，研究员  
陈震，助理研究员

[{zhenchen, jun1}@tsinghua.edu.cn](mailto:{zhenchen,jun1}@tsinghua.edu.cn)

网络安全实验室，信息技术研究院，  
清华大学

<http://security.riit.tsinghua.edu.cn>

# 实验室介绍



- <http://security.riit.tsinghua.edu.cn>
- 主要研究方向
  - 网络安全的基本算法
    - ◆ 网包分类
    - ◆ 特征匹配
  - 覆盖网络Overlay Network
  - 可信计算

# IT行业背景



- 信息和网络安全是IT工业不可缺失的部分
- 活跃期1993---至今
- 计算机科学与技术的重要内容
  - ACM Turing Award grant to RSA
  - 基本分类和特征匹配算法，数据结构
- 综合学科，实践性很强

# 课程内容



- Chapter 1 网络世界
- Chapter 2 安全威胁和网络安全
- Chapter 3 防火墙
- Chapter 4 网络处理器
- Chapter 5 网包分类
- Chapter 6 特征匹配
- Chapter 7 IP网络安全
- Chapter 8 基于模型的安全分析



# 课程意义

- 拓展视野，培养工程认识
- 消除对网络攻击的恐惧感？
  - 害怕病毒？害怕蠕虫？
  - 了解攻击，防范攻击
  - 安全是没有止境
- 消除对大系统的恐惧感？
  - 系统出了问题，不知所措？
  - 备份数据，进行重装？
- 创造财富
  - Juniper captured NetScreen with 3.5Billion
  - Symantec, Trend Micro etc.

# 课程演示设备



- 实验演示设备
  - SimpleNet网络教学设备
  - Juniper NS-5GT 防火墙
  - ServGate安全网关
  - Intel IXDP2800 Network Processor Platform
- 访问FIT 3-421, 网络安全实验室

# 讲课思路



- 覆盖计算机安全基本理论和网络安全工程实践
- 分析网络安全攻击
- 结合具体的实验设备和实验讲解
  - 防火墙
  - 入侵检测系统
  - 入侵保护系统
  - 网络处理器
  - 等等...

# 对同学们的建议



- 锻炼自己读写英文的能力(very important)
  - 对于技术研究领域，英文读写能力非常重要
- 尽早使用Linux系统(Unix)，熟悉了解OS
  - Windows注册表----- Linux Service.conf
  - Windows Services -----Linux /etc/services
- 加入”Open Source”活动，加入国际开源团队
- Linux系统
  - 安装 Ubuntu Linux, Fedora Core 5, FreeBSD, Gentoo Linux, Debian Linux, Gnoppix

# 期刊杂志会议



## ● 杂志

- 《信息网络安全》
- 《计算机安全》
- 《信息安全和通信保密》
- IEEE Magazine Security and Privacy

## ● 会议

- ACM Computer Communication and Security
- IEEE Symposium on Security and Privacy

## ● 网上电子期刊

- Google Network Computing, InfoWorld, ZDnet, linuxsecurity

# 学校课程



- 李军, 网络安全
- 任丰原, 计算机网络基础
- 段海新, 网络安全
- 吴建平, 计算机网络原理
- 尹霞, 计算机网络安全技术
- 罗平, 数据与网络安全
- ...

# 从病毒开始



- 一块USB
- 4个病毒
  - InfoStealer.QQRob.A
  - InfoStealer.Lineage
  - W32.Reper.A
  - W32.SilyDC
- 解开AutoRun.inf

*Any Question?*

*Send me your comments to*

*[zhenchen@csnet1.cs.tsinghua.edu.cn](mailto:zhenchen@csnet1.cs.tsinghua.edu.cn)*

网络安全原理与技术

# 第一章 网络世界

## Chapter 1 Hello Network!

李军，研究员  
陈震，助理研究员

网络安全实验室，信息技术研究院，  
清华大学

<http://security.riit.tsinghua.edu.cn>

# 目录



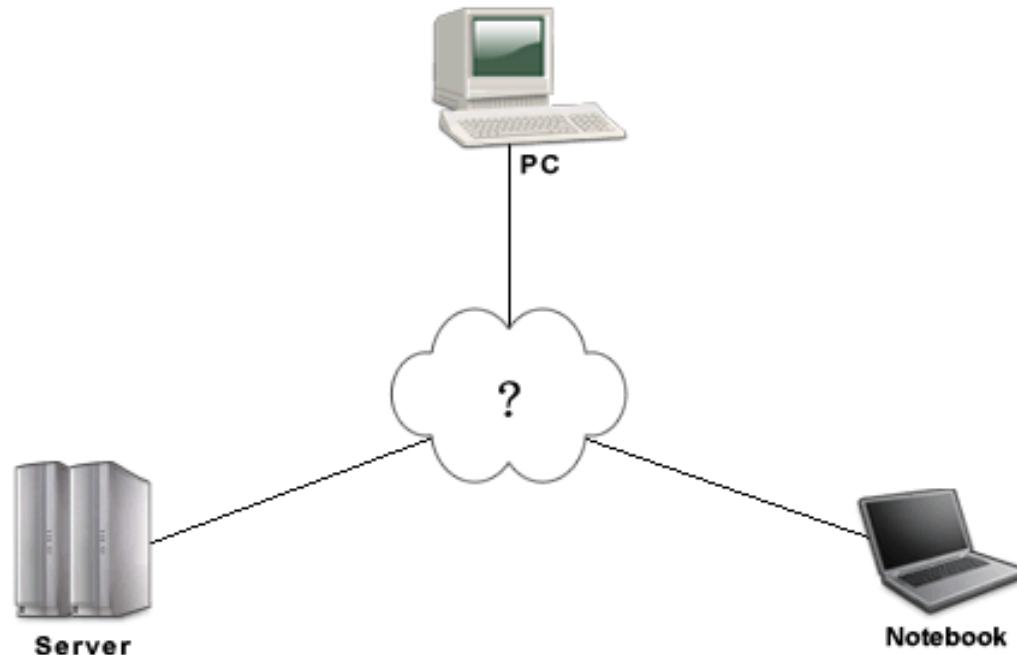
- 网络简介
- 网络拓扑 (Topology)
- 网络架构设计示范
- 网络抽象模型
- TCP/IP 协议
- 以太网络(Ethernet)

# 网络简介



# 何谓网络

- 经由一组的计算机或处理器单元，通过彼此共同的物理介质（电缆或无线传输介质）互相连接在一起，以达到资源(打印机、存储器、CPU 、内存及文件…等等)共享的目的。



# 网络的连接性(Connectivity)



## ● 基本元素

- 链路( link ):

- ◆ 电缆或无线传输介质

- 节点( node ):

- ◆ 一般计算机或特定功能的计算机(设备)

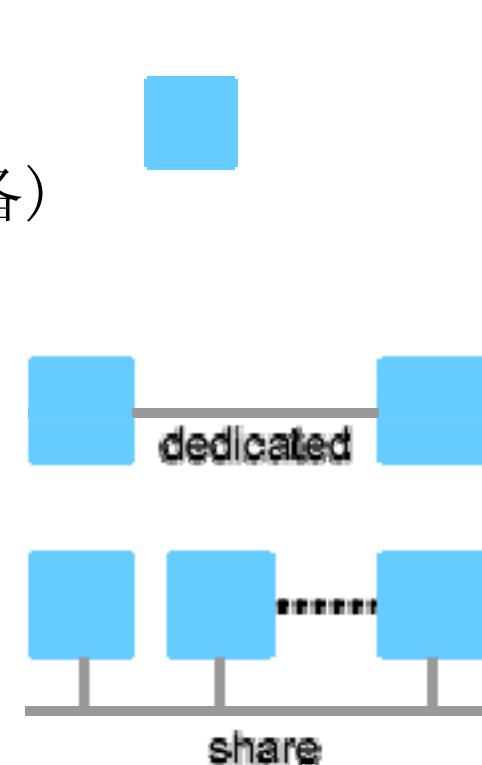
## ● 存取方式

- 点对点( point-to-point ):

- ◆ 独享带宽

- 多重接入( multiple-access):

- ◆ 共用带宽





## ● 连接方式

- 点对点 ( point-to-point ): 直接相连(实体相连)



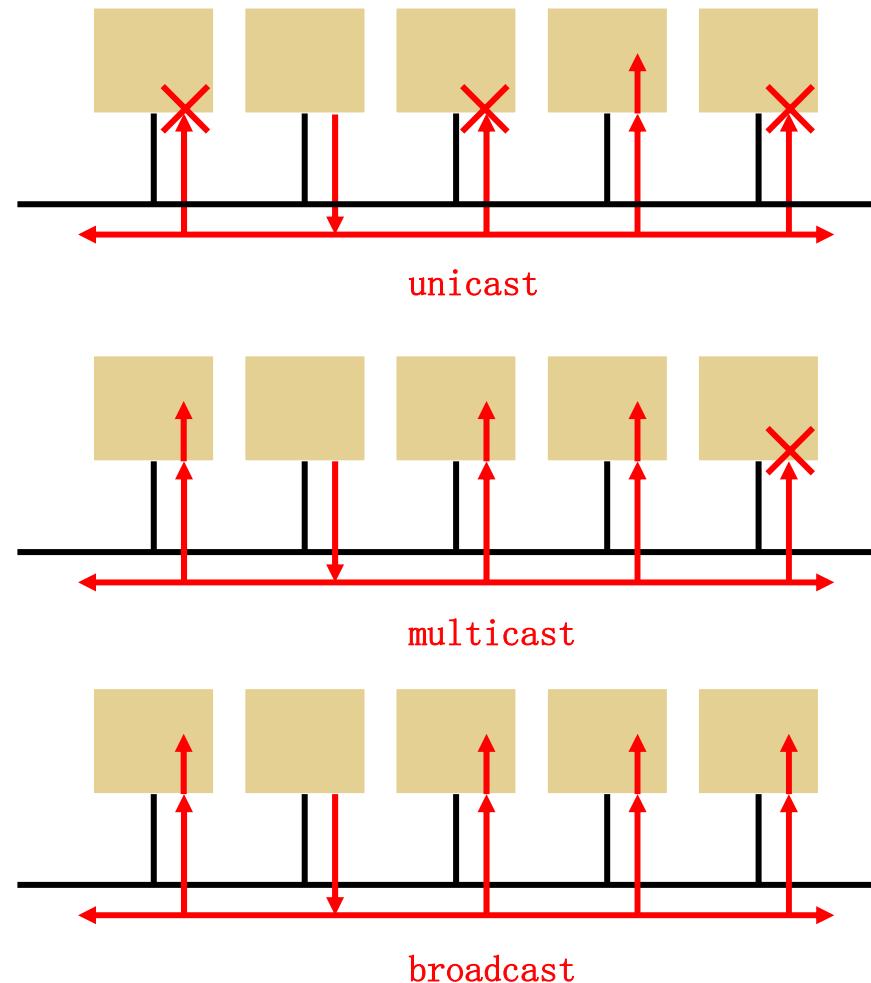
- 端对端 ( End-to-End ), 间接相连(跨网络相连)



# 传输模式



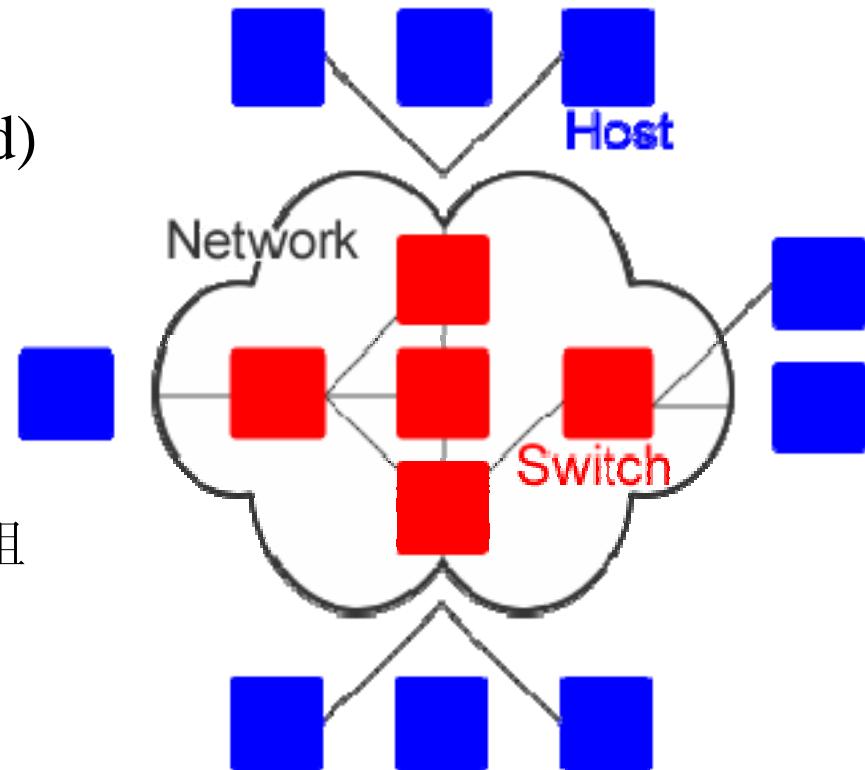
- Unicast (单点传输)
- Multicast (群播)
- Broadcast (广播)



# 交换网络(Switched Network)



- 电路交换 (circuit-switched)
  - 电话系统、专属电路
- 数据包交换 (packet-switched)
  - 计算机网络
  - 将待送数据切成许多数据包 (packet), 又叫网包  
送出, 到目的地再进行数据重组
  - 存储转发 (store-and-forward)



# 多路复用(Multiplexing)



- 时分多路复用

(Time—Division Multiplexing, TDM)

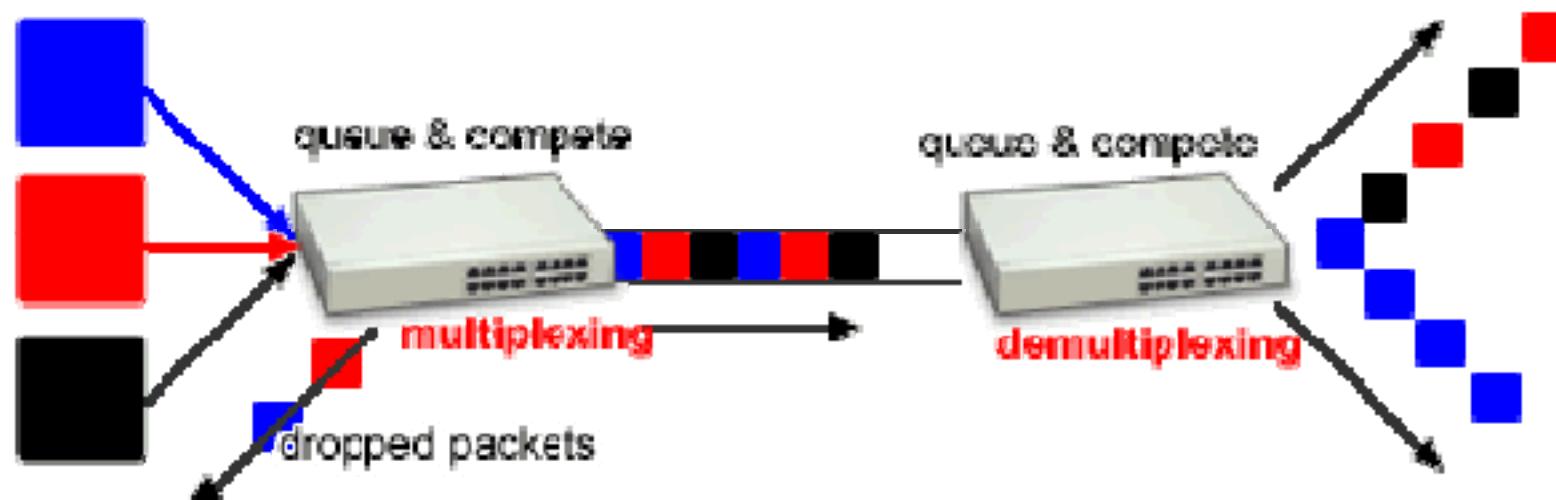
- 频分多路复用

(Frequency—Division Multiplexing, FDM)

# 多路复用与多路分解示意图



清华大学  
Tsinghua University



# internetwork & Internet



## ● 网络互连( internetwork )

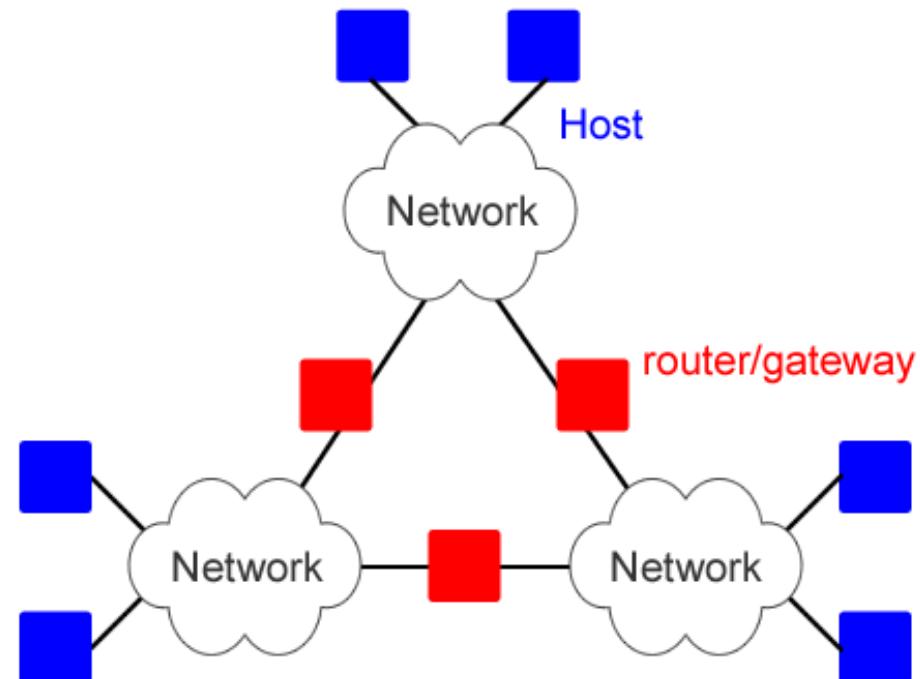
- 各组独立的网络(云状物)

互相连接而形成互联网

## ● 因特网( Internet )

- 采用TCP/IP协议进行互联

- 世界范围的网络

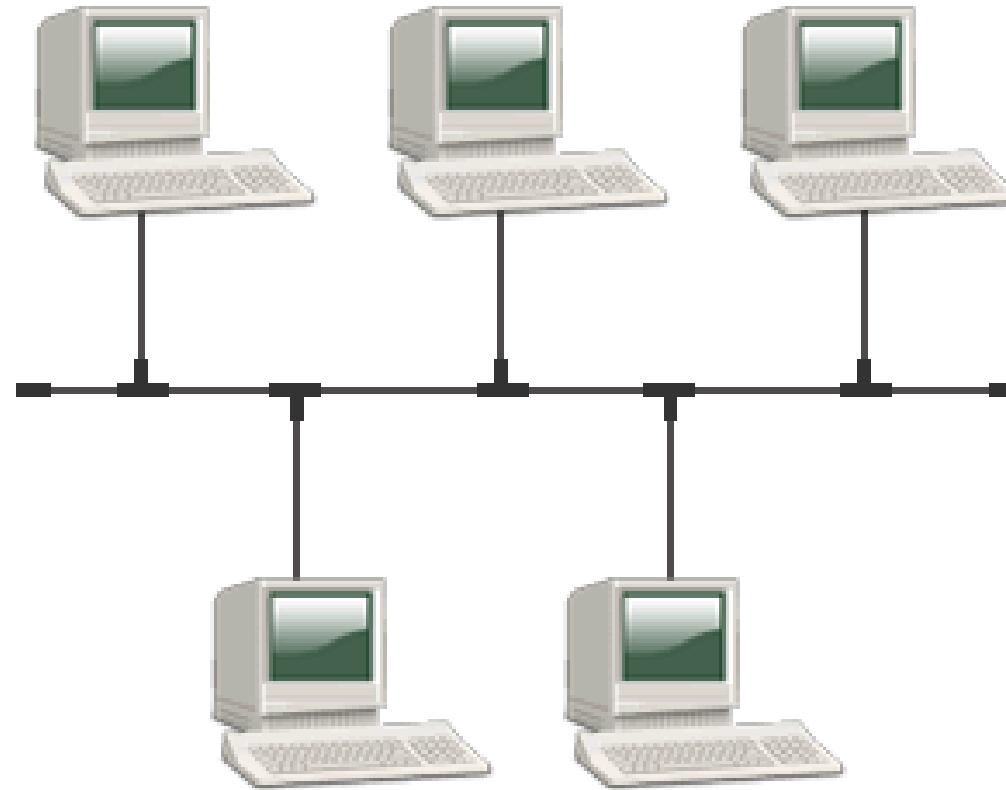


# 网络拓扑 (Topology)

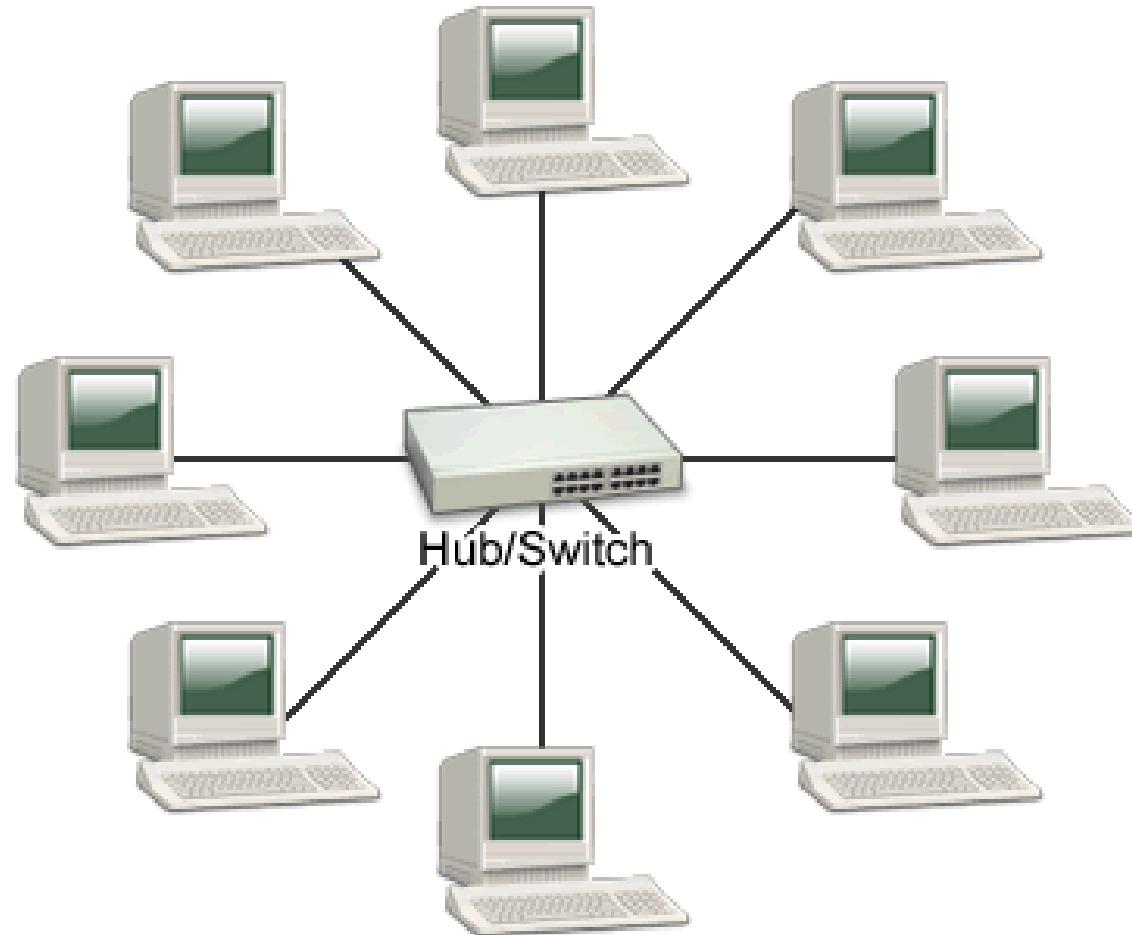


- 总线拓扑( Bus )
- 星型拓扑( Star )
- 环型拓扑( Ring )

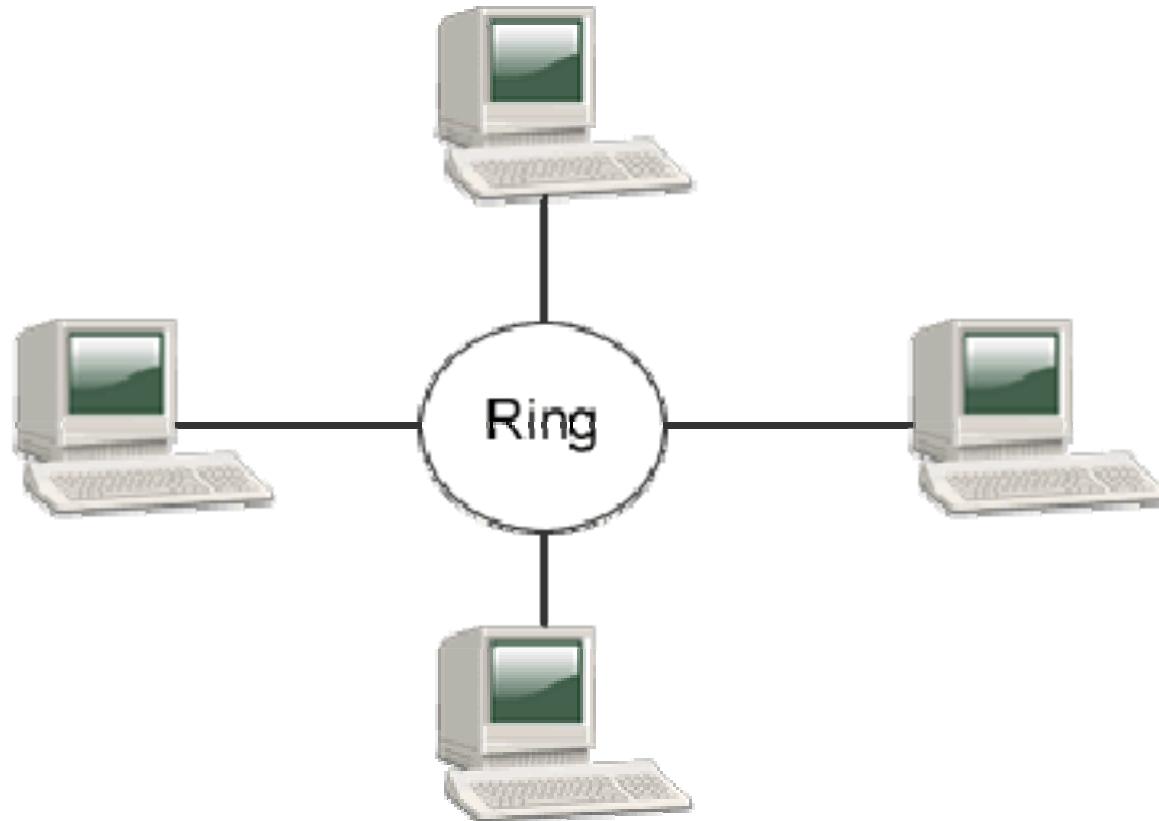
# 总线拓扑(Bus Topology)



# 星型拓扑(Star Topology)



# 环型拓扑(Ring Topology)

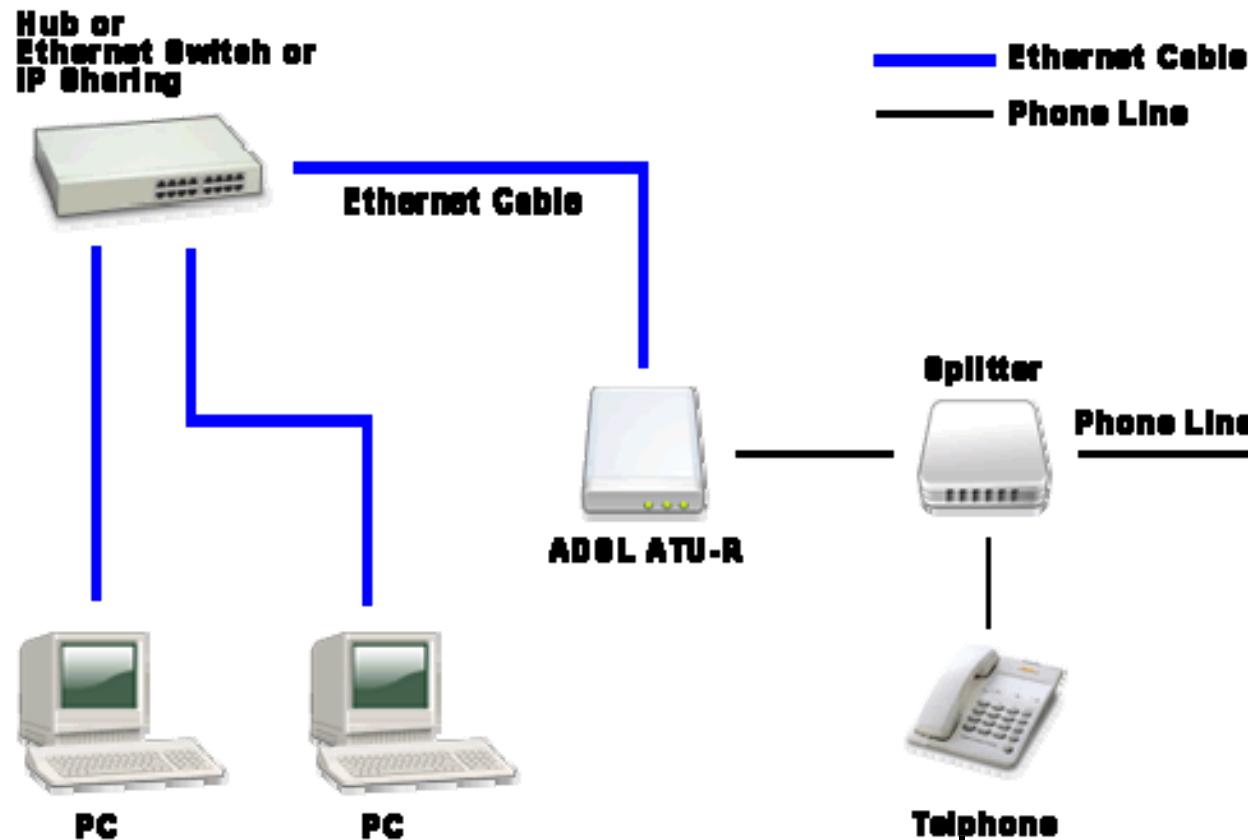


# 网络架构设计示范

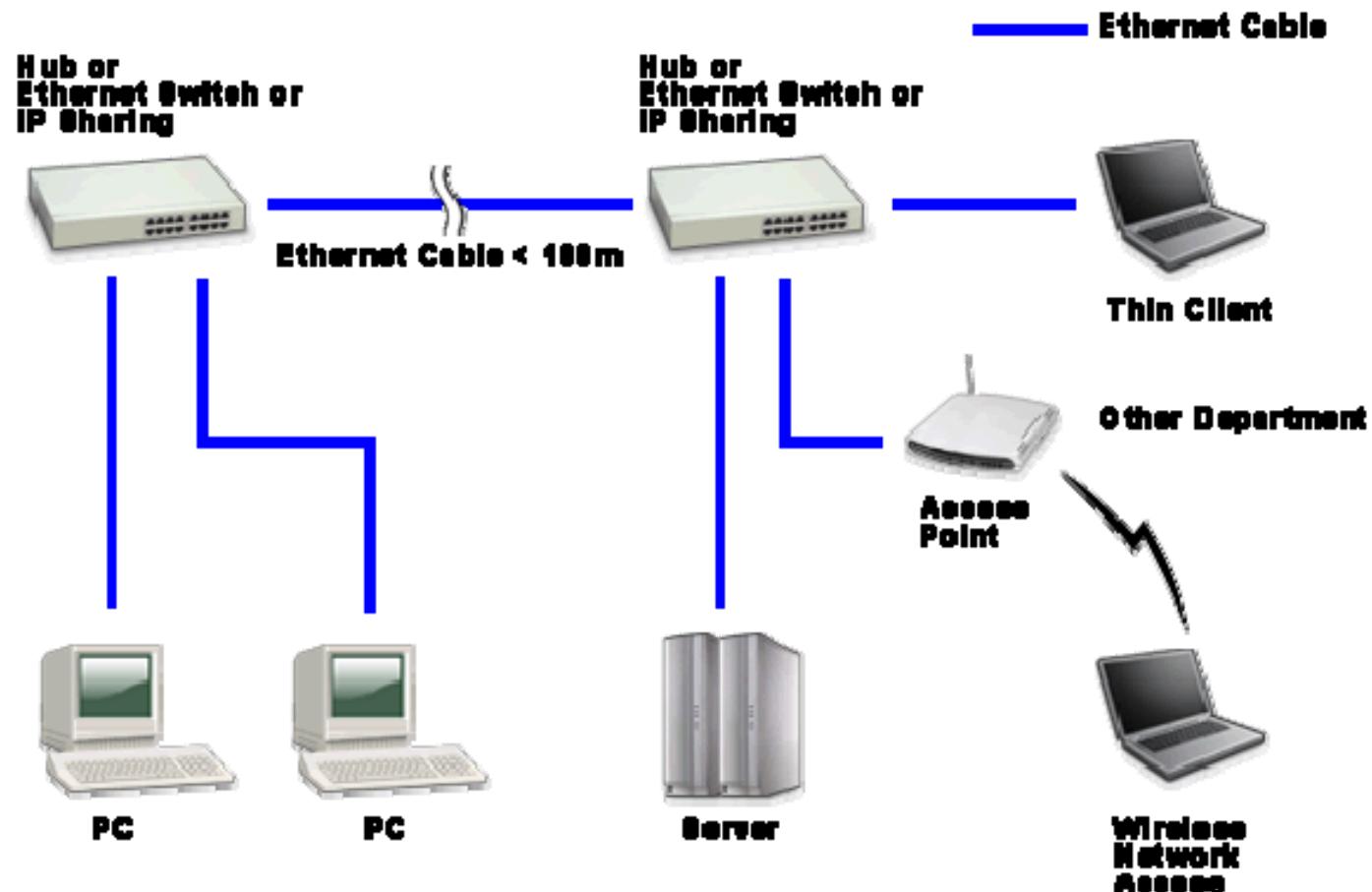
# 家用多台PC，通过 ADSL连接 Internet



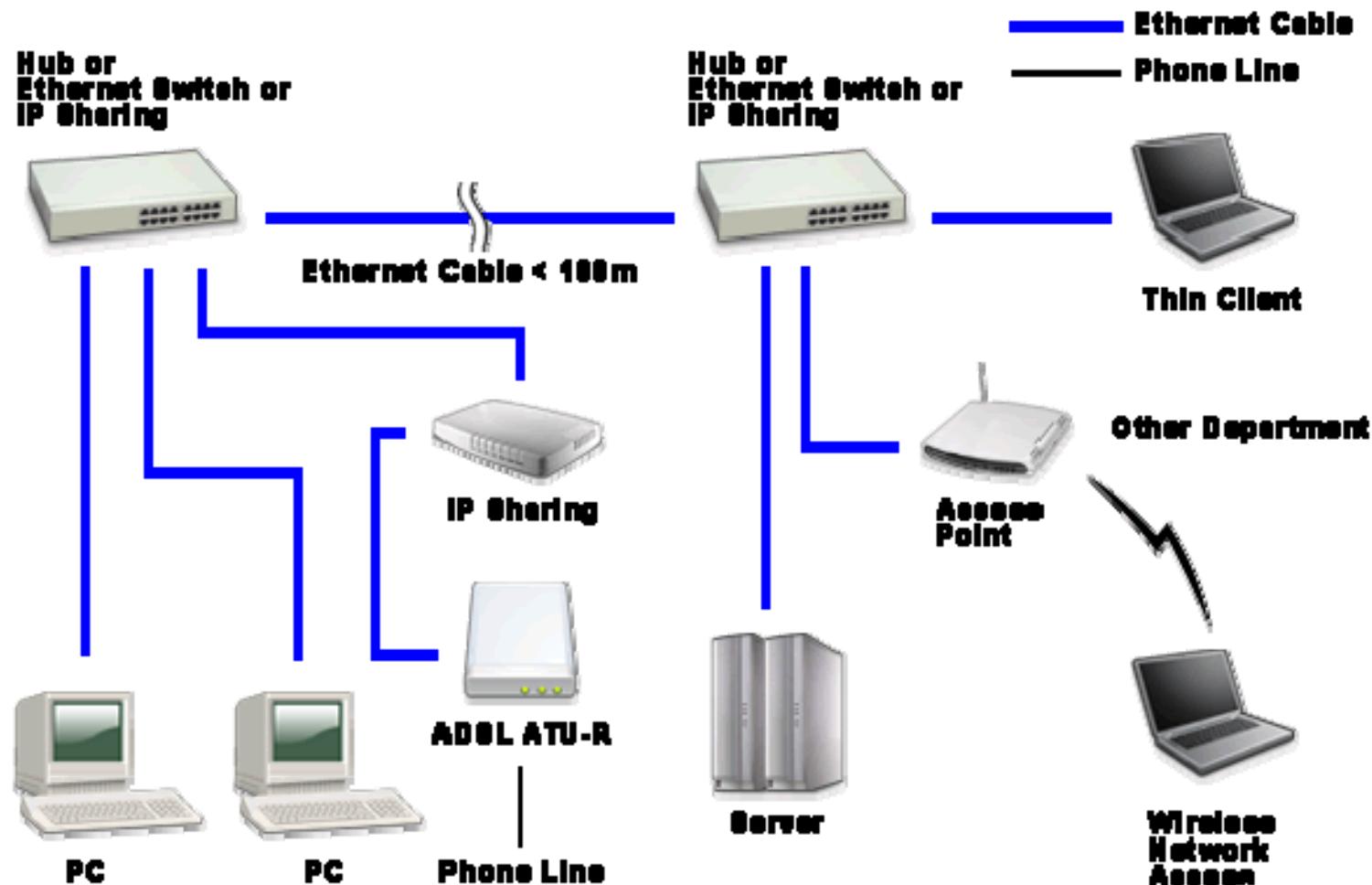
清华大学  
Tsinghua University



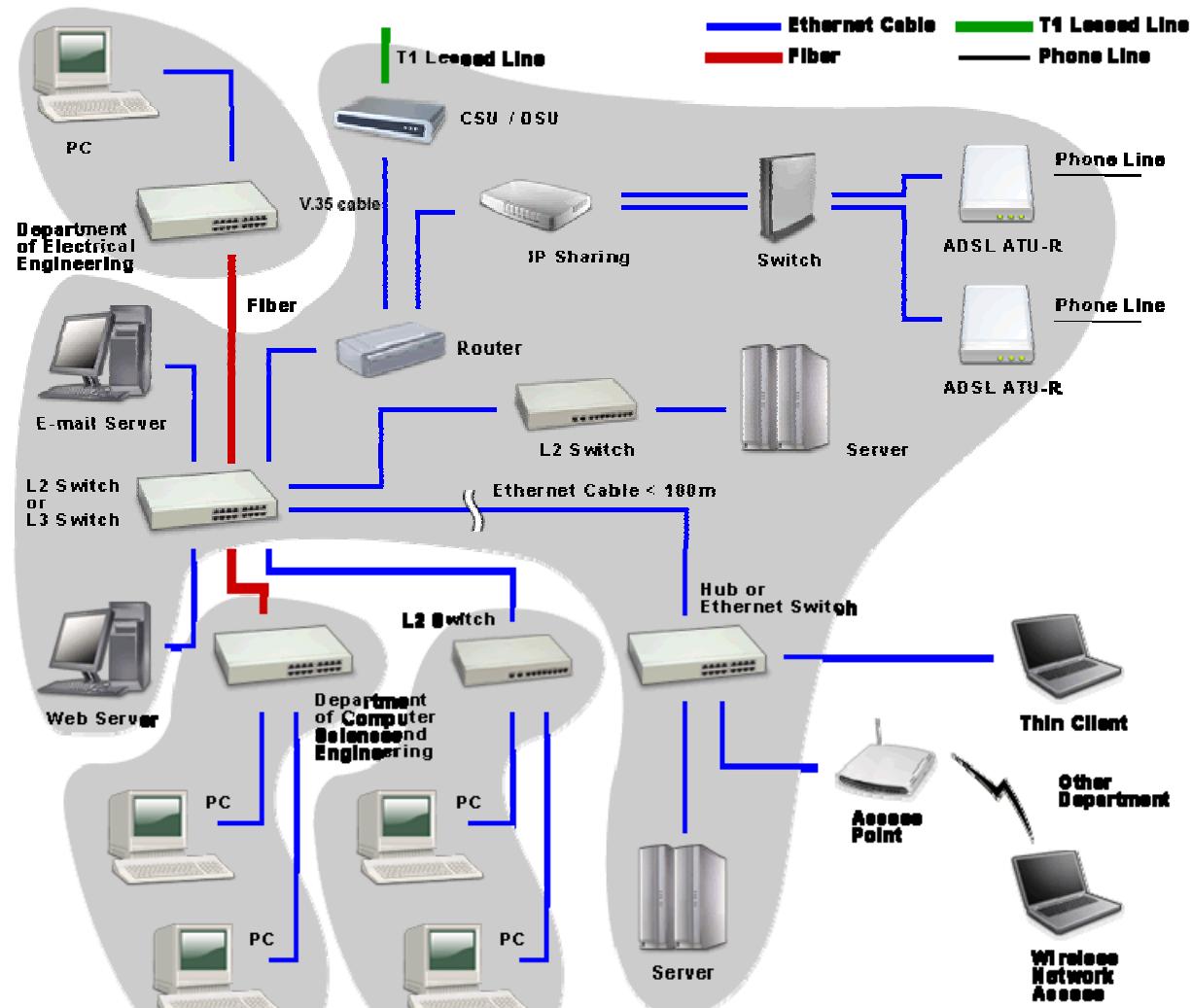
# 无上网功能小型无线网络架构



# 具有上网功能的小型网络架构



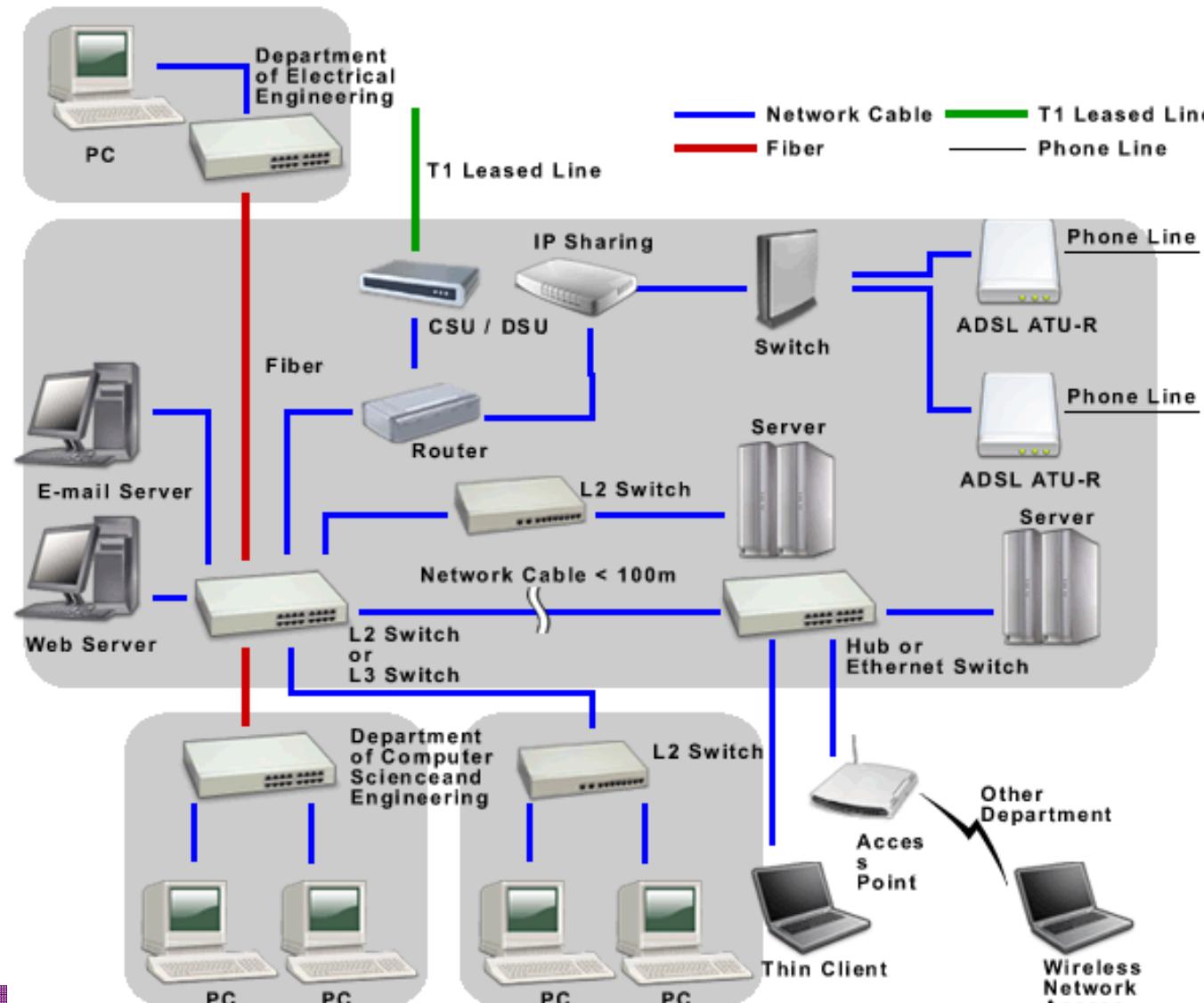
# 具有上网功能的中型网络架构 (租用T1线路)



# 具有上网功能的中型网络架构 (租用T1线路)



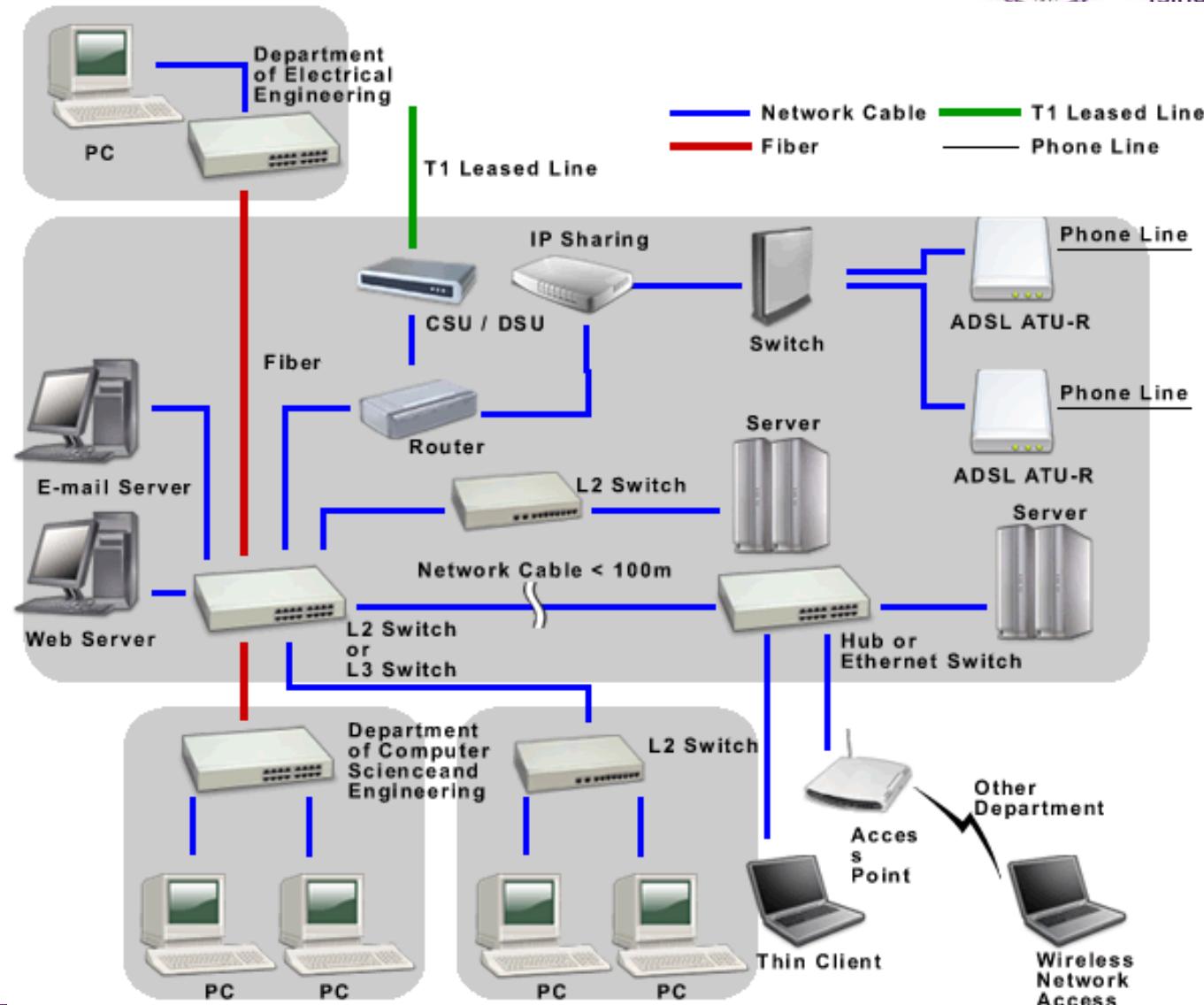
清华大学  
Tsinghua University



# 校园网络架构



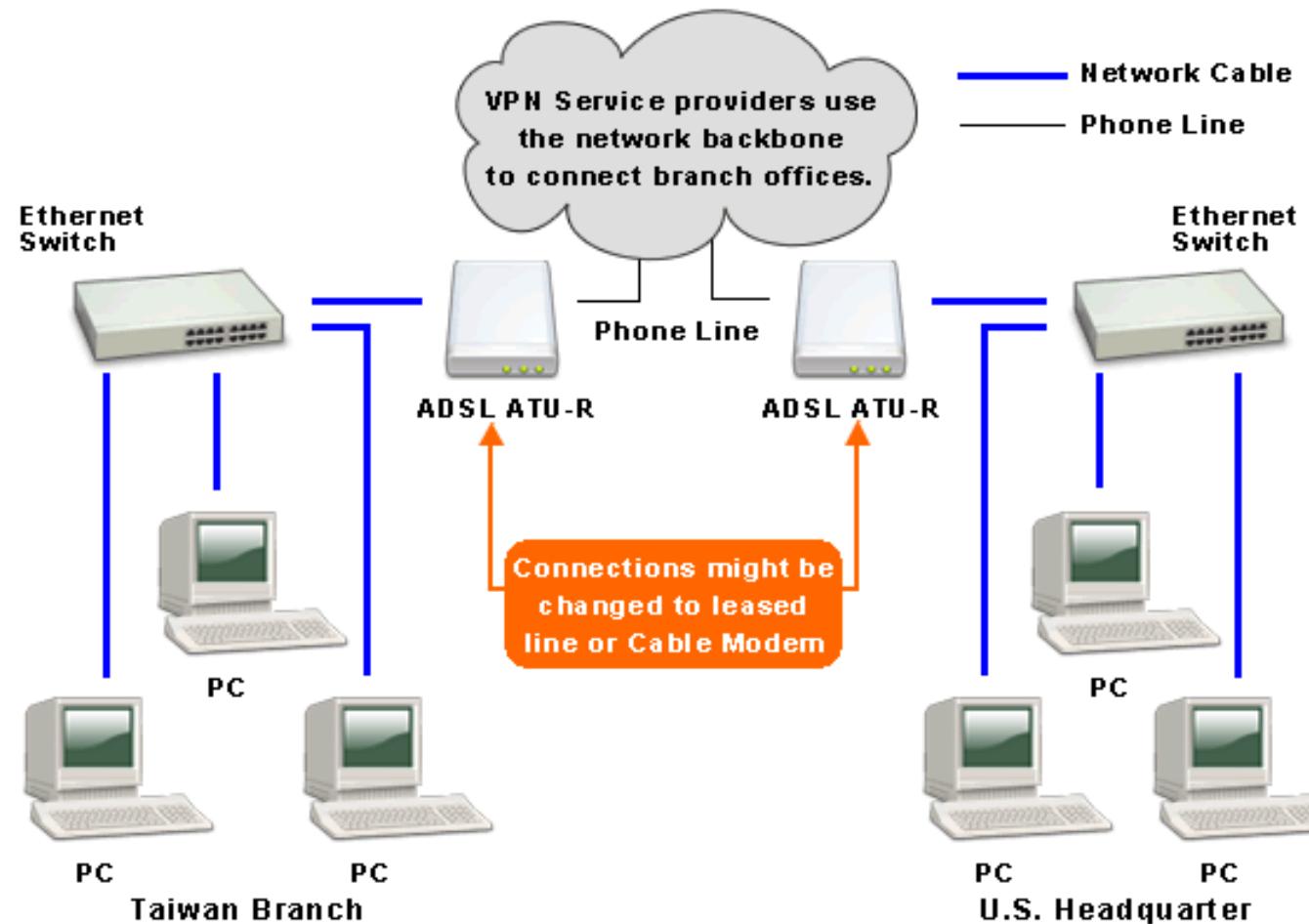
清华大学  
Tsinghua University



# 企业VPN 架构示范



清华大学  
Tsinghua University



# 网络模型与TCP/IP 协议

# OSI 7 Layer Reference Model



- ISO 的 OSI 7 Layer Reference Model 是当前共同的网络设计参考模型
  - ISO 指 International Organization for Standardization 国际标准组织
  - OSI 表示 Open System Interconnection, OSI 7 Layer 将网络运作划分为七层, 然而 TCP/IP 的网络实际上并没有将分层区划分得如此详细
- OSI Model (开放系统互连模型) 与 TCP/IP 的差异
  - 参考模型与实际环境

# OSI 7 Layer 与 TCP/IP Protocol Stack



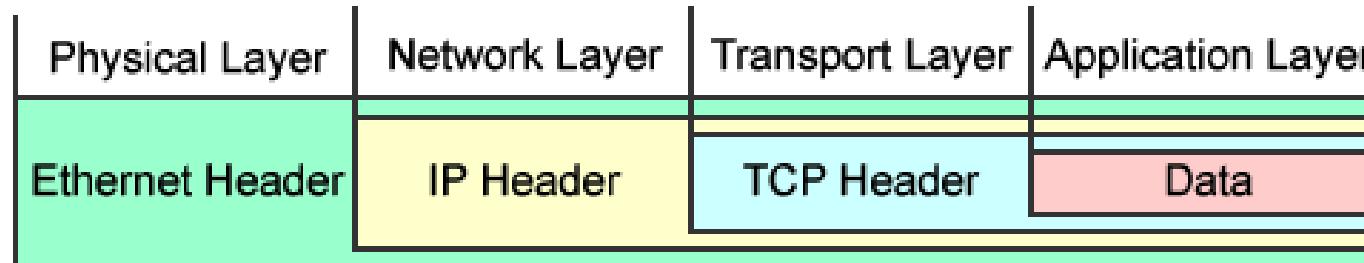
清华大学  
Tsinghua University

	OSI 7 Layer		TCP / IP
1	Application	7	<p>Application Example : FTP, WWW, BBS, MSN</p> <p>Transport Example : TCP, UDP</p> <p>Network Example : IP, Routing, ICMP</p> <p>Link Example : 100 Mbps Ethernet</p>
2	Presentation	6	
3	Session	5	
4	Transport	4	
5	Network	3	
6	Data Link	2	
7	Physical	1	

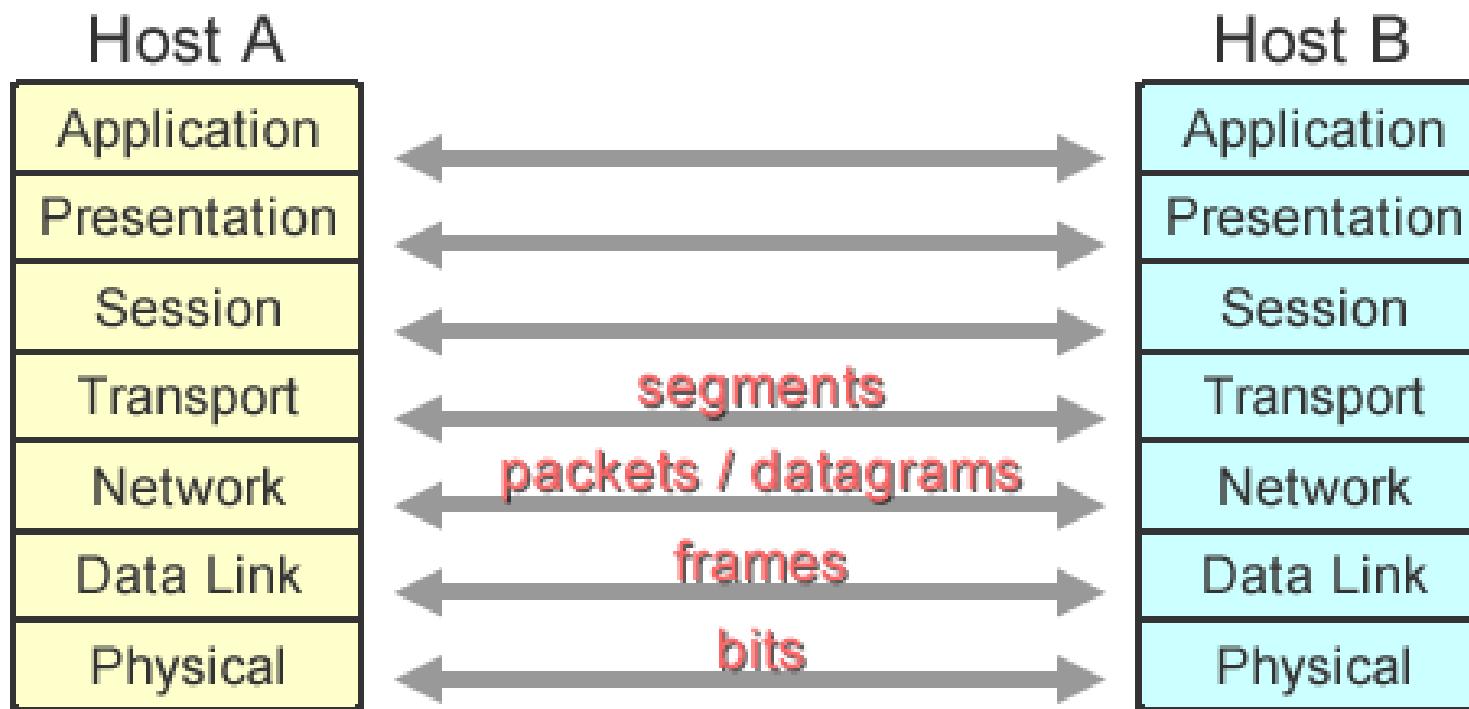
# 分层 (Layering) 的观念



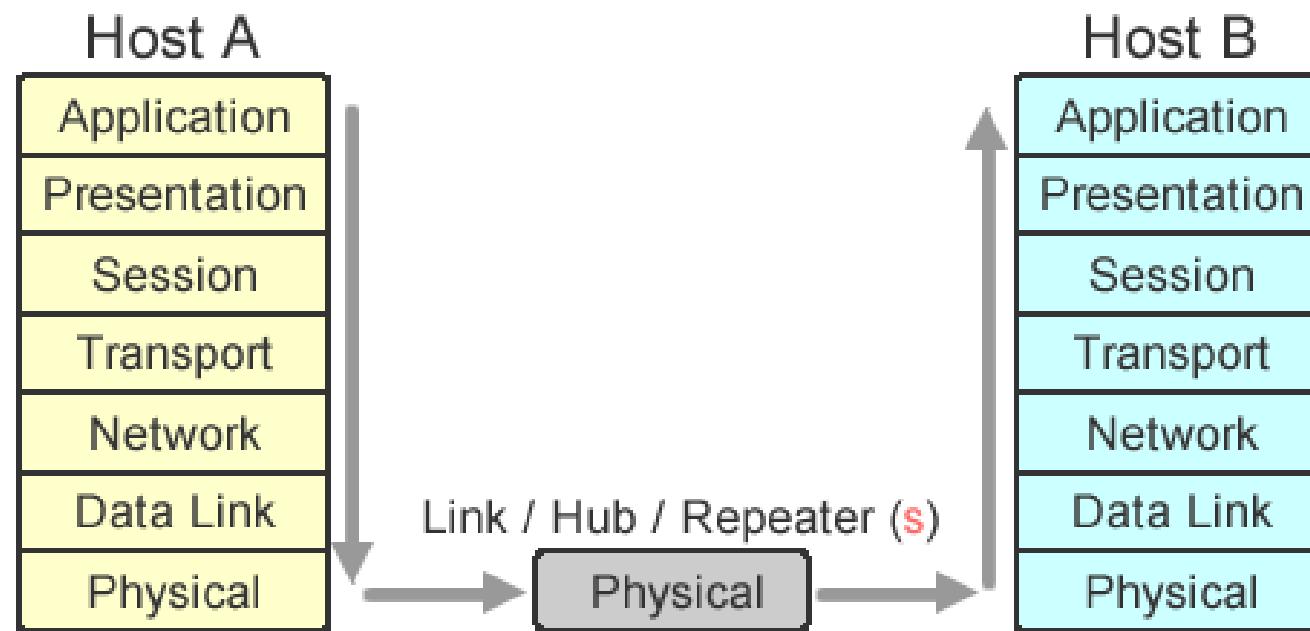
- peer-to-peer 通信 (同层的对等关系)
  - 分层负责，各司所职
- service interface (上下层关系)
  - 往下层传送时会添加各层所需的 header (Protocol)，称为数据封装



# Peer-to-Peer 对等通信



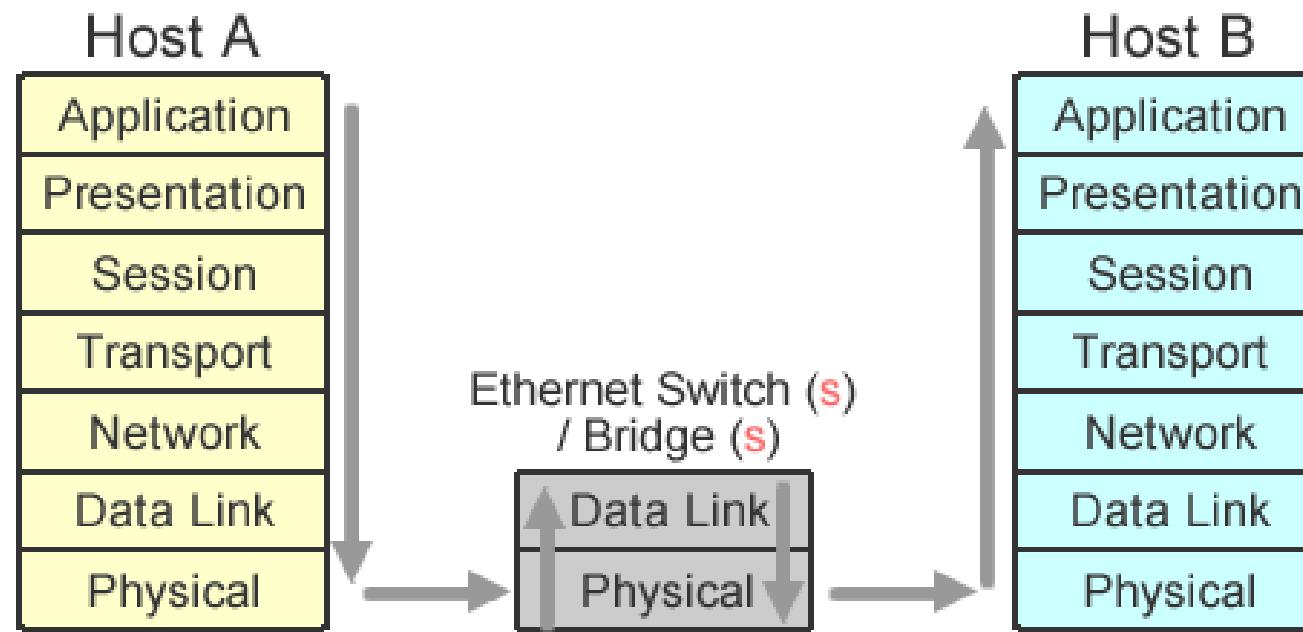
# 连接不同 Layer 1 的网络



# 连接不同 Layer 2 以下的网 络



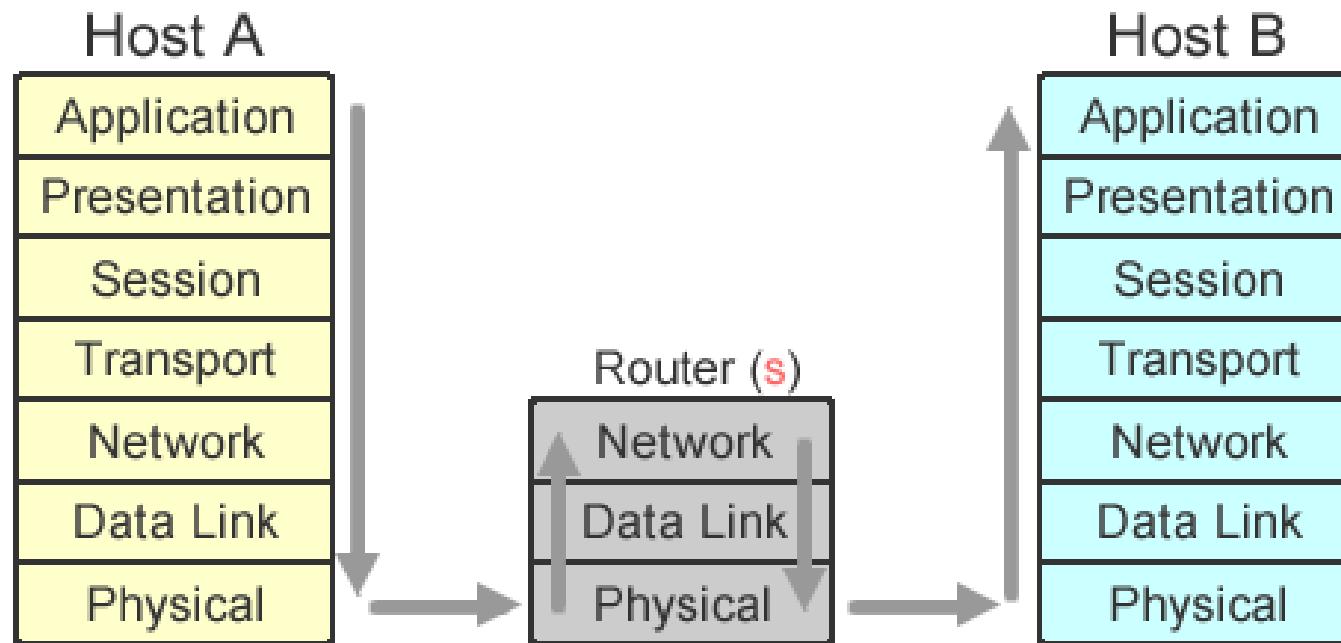
清华大学  
Tsinghua University



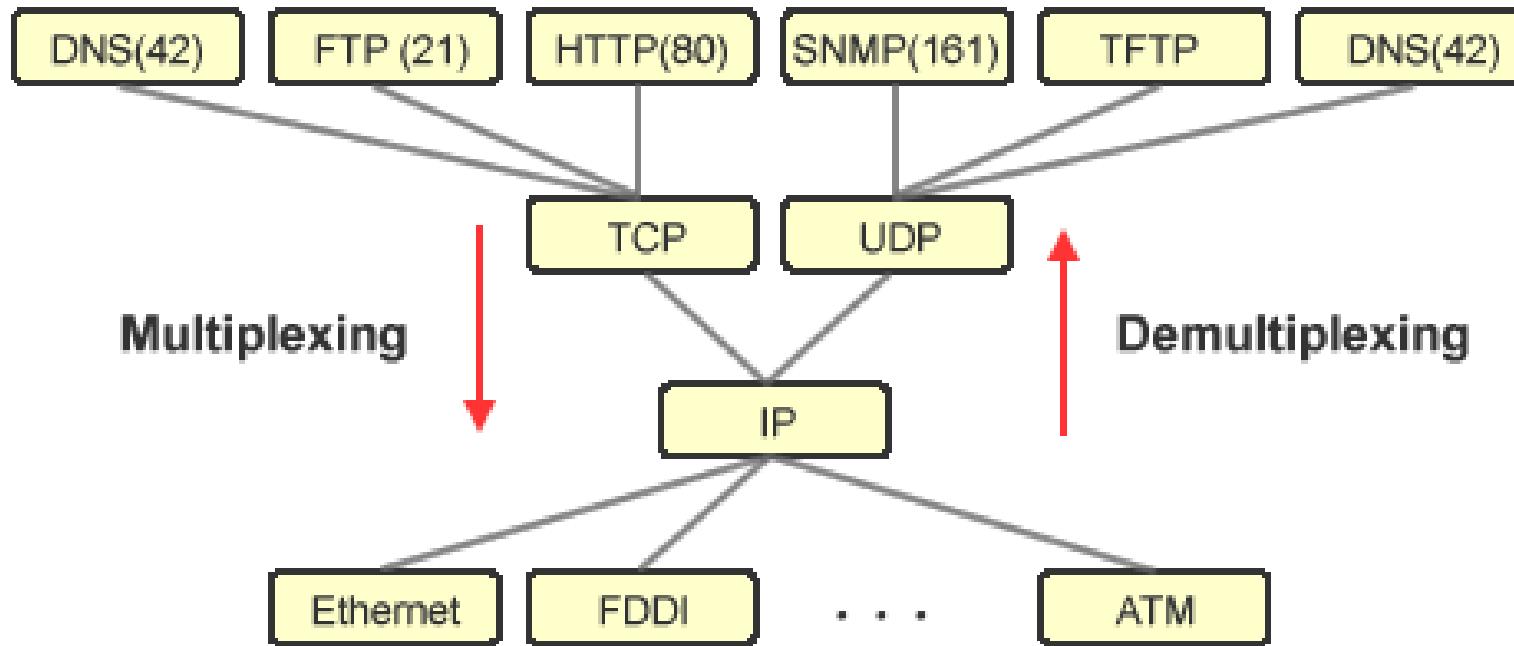
# 连接不同 Layer 3 以下的网 络



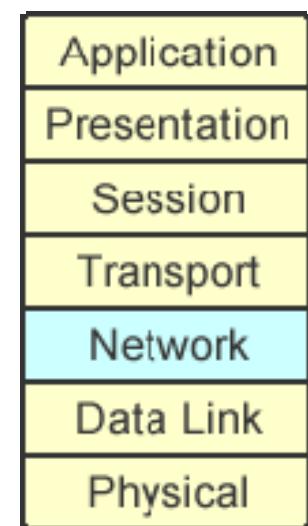
清华大学  
Tsinghua University



# TCP/IP Suite 示意图



# Internet Protocol



# 概述



- RFC 791定义IP (Internet Protocol) 协议
- Internet IP网络层协议，尽力而为传送数据到达指定地址，不确认数据是否正确到达，是一种无连接(Connectionless)的协议
- 主要功能
  - 网包 (Packet) 路由选择 (Routing)
  - Packet分段 (Fragmentation) 及重组 (Re-assembley)

# 分级 IP 地址 (Address)



- Internet Protocol Version 4 (IPv4)
- 辨别设备或主机唯一的识别码，适用于不同网络间寻址
- 由 32 Bits 组成，可有  $2^{32} = 4,294,967,296$  个地址
- 由网络地址和主机地址两部分组成
- 依照网络地址和主机地址的分配，分成五种 Class:
- 清华大学59. 66. 0. 1/16(A类), 166. 111. 0. 1/16(B类)

	1st Byte	2nd Byte	3rd Byte	4th Byte	hosts
Class A	0	Network	Host Addr		16,777,216
Class B	10	NetworkAddr		Host Addr	65,536
Class C	110	Network Addr		Host Addr	256
Class D	111	Multicast Multicast			Multicast
Class E	1111	Reserved			Reserved



Class	First Bits	First Byte Values	MAX of Hosts in its Subnet
A	0	0~127	$2^{24}-2=16,777,214$
B	1 0	128~191	$2^{16}-2=65,534$
C	1 1 0	192~223	$2^8-2=254$
D	1 1 1	224~239	Reversed for multicast
E	1 1 1 1	240~255	Reversed

# 子网掩码 Netmask



- 目的：对 IP 地址分类，以区分网络上的主机是否在同一网段内 netid
- 例：192.168.1.96 / 27

IP Address	11000000 10101000 00000001 01110100	
↓ subnet mask AND	27 bits	→
netid	11111111 11111111 11111111 11100000	
	192 . 168 . 1 . 96	

# 名词介绍



## ● netid

- 每一组IP地址的第一个IP地址，即为该组IP地址的网络号码 (netid)
- 例：192.168.1.0 / 24 → 192.168.1.0

## ● broadcast

- 每一组IP地址的最后一个IP，即为该组IP地址的广播地址 (broadcast)
- 例：192.168.1.0 / 24 → 192.168.1.255

# 不分级 IP 地址 - CIDR



- 无类型域间路由 —

CIDR (Classless Inter-Domain Routing)

- 使用 CIDR 即可代表数个 Class C 替换以往需要使用一个 Class B 的地址浪费，可节省 IP 地址
- 例： 222.77.235.0/21 及 222.77.237.0/21

222.77.235.0 → 11011110.1001101.11101011.0

222.77.237.0 → 11011110.1001101.11101101.0

由左至右比对到第 21 个bit 皆相同，视为同一网段

# 分割子网 Subnetting



- 将一较大的网络区段切割成几组较小的网络
- 使用于内部路由选择协议 (Interior Routing Protocol)
- 例如：172. 16. 0. 0 / 16  
可将一个 B 级网络分成 256 个较小网络

172. 16. 0. 0 / 24

172. 16. 1. 0 / 24

172. 16. 2. 0 / 24

•

•

•

172. 16. 255. 0 / 24

# 超网化 Supernetting



- 由数个较小的网络区段组合成单一较大网络
- 使用于外部路由选择协议 (Exterior Routing Protocol)
- 可减少 internet 路由表数量
- 例如: 203.100.64.0 / 20  
( $203.100.64.0 \sim 203.100.79.255$ )  
将 16个C 级网络合成一个 203.100.64.0/20 较大网络
  - netid: 203.100.64.0
  - range:  $203.100.64.0 \sim 203.100.79.255$
  - netmask: 255.255.240.0
  - broadcast: 203.100.79.255

# Private 专用地址



- 不需申请，可直接使用
- 内部Routing Information只在内部有效
- 需透过地址转换 NAT（网络地址转换）才能对外连接
- 可用的 Private 地址范围：
  - $10.0.0.0 \sim 10.255.255.255$  ( $10.0.0.0/8$ )
  - $172.16.0.0 \sim 172.31.255.255$  ( $172.16.0.0/12$ )
  - $192.168.0.0 \sim 192.168.255.255$  ( $192.168.0.0/24$ )

# IP Protocol Header



0	4	8	12	16	20	24	28	32											
4-bit Version	4-bit Header Len	8-bit Type of Service			16-bit Total Length ( Bytes )														
16 Bits Identification				3-bit Flags	13-bit Fragment Offset														
8-bit Time to Live ( TTL )		8-bit Protocol		16-bit Header Checksum															
32-bit Source IP Address																			
32-bit Destination IP Address																			
Options																			
Data																			



# 协议域说明1

- Version(4 bits)
  - IP Protocol 的版本，当前为 IPv4，下一代为 IPv6
- Header Len(4 bits)
  - IP Header 的长度(5~15)，默认为 5(即为  $5 \times 4 = 20$  Bytes)  
[注: 5 rows, 4 Bytes/row]
- Type of Service(8 bits) (服务类型)  
(当前作为 Differentiated Services, RFC 3317, RFC 2474)
  - Precedence (3 bits): 代表数据包的重要性，值愈高表示愈重要
  - D (1 bit): 设置0为一般延迟，设置1为低延迟(Low delay)
  - T (1 bit): 设置0为一般传送量，设置1为高的传送量(Throughput)
  - R (1 bit): 设置0为一般 Reliability，设置1为高的 Reliability
  - Unused (2 bits): 未使用

# 协议域说明2



- Total Length(16 bits)

- 总长度 (IP header + data , 576 ~ 65535 Bytes)
  - 单位为 Byte

- Identification (认证)

用来识别数据报 Datagram 使用，分段与依序重组

- Flags (标志)

- Bit 0: reserved, must be zero
  - Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
  - Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments. Fragment Offset

# 协议域说明3



## ● Fragment Offset (段偏移量)

- $0 \sim 8191$  ( $= 2^{13} - 1$ )
- 分段后数据包的位移量
- 以 8 bytes 为基本位移单位
- 最大总数据量 =  $8192 \times 8 = 65536$  bytes

## ● Time to Live (TTL) 生命期

- 数据包在网络的存活时间
- 每经过一个 Router 计数器自动减一，直至0为止，便将数据包丢弃(Discard)

# 协议域说明4



清华大学  
Tsinghua University

## ● Protocol (协议)

- 0: 保留
- 1: Internet控制报文协议 ICMP (Internet Control Message Protocol)
- 2: Internet组报文协议 IGMP (Internet Group Management Protocol)
- 5: 数据流 ST (Stream)
- 6: 传输控制协议 TCP (Transmission Control Protocol )
- 8: 外部关联协议 EGP (Exterior Gateway Protocol )
- 9: 内部关联协议 IGP (Interior Gateway Protocol )
- 17: 用户数据报协议 UDP (User Datagram Protocol )
- 其他请参考 rfc1700 Assigned Numbers

<http://www.ietf.org/rfc/rfc1700.txt>



# 协议域说明5

- Header Checksum (校验和)
  - IP Header 的错误检查码，将IP Header中所有16-bit的word转为1的补数相加后，再取一的补数，此值即为 Checksum 值
- Source IP Address
  - 来源IP地址
- Destination IP Address
  - 目的IP地址
- Options
  - 选择性字段
- Data
  - 真正传送的数据

# IP Header Checksum 计算示 范



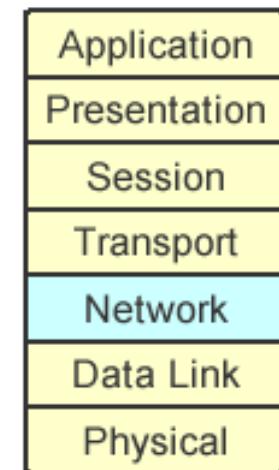
清华大学  
Tsinghua University

4	5	0	0
0	0	3	0
9	e	a	4
4	0	0	0
8	0	0	6
3	d	d	d
0	3	8	5
d	3	4	8
f	e	0	6

$$4500 + 0030 + 9ea4 + 4000 + 8006 + 3ddd + 0385 + d348 + fe06 = 3b68a$$

进位的3加回 b68a 中， $3+b68a=b68d$ ，换成二进位为 1011 0110 1000 1101  
再取其反码 0100 1001 0111 0010

# ICMP 因特网控制报文协议



# ICMP 因特网控制报文协议



- RFC792 定义 ICMP 协议 (Internet Control Message Protocol)
- 设计用来处理网络设备之间的错误的，报告网络环境中错误状态的发生
- 无法确保能把消息确实送达或是转回发送地
- ICMP 报文基本上是用数据包 Datagram 来传送错误消息，为了避免无限制地传送错误消息，ICMP 消息传送的错误并不会生成任何其他的错误消息
- 消息被分成几个分段 (Fragmentation) 时，ICMP 消息只会送第一个 Fragmentation 的错误消息

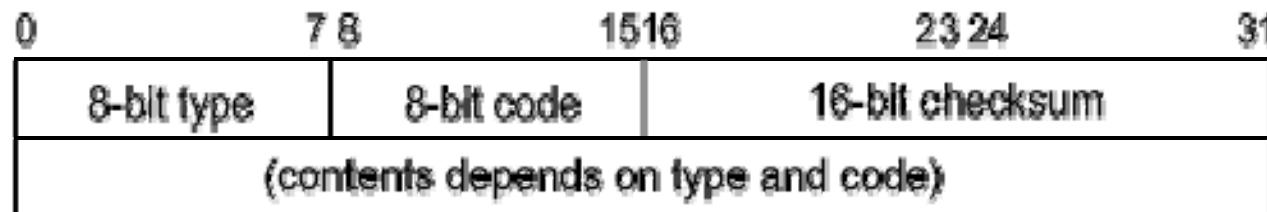
# ICMP 消息



- 消息格式是依照消息种类不同而异，相同部分

- TOS 字段为 0, Protocol 的字段为 1

- ICMP 相同消息部分



- Type : ICMP 消息的类别
    - Code 和下面的部分则是依各个消息类别不同
    - Checksum 的计算方式则是和 IP Checksum 相同
- 参考 <http://www.iana.org/assignments/icmp-parameters>

# ICMP 应用 — ping



- 利用 ICMP 消息里的 Type 0 和 8 ( Echo Reply/Request )
- 测试本机到网络某一端的机器是否有连接，或是测试远程机器是否有开启 ping 的回应

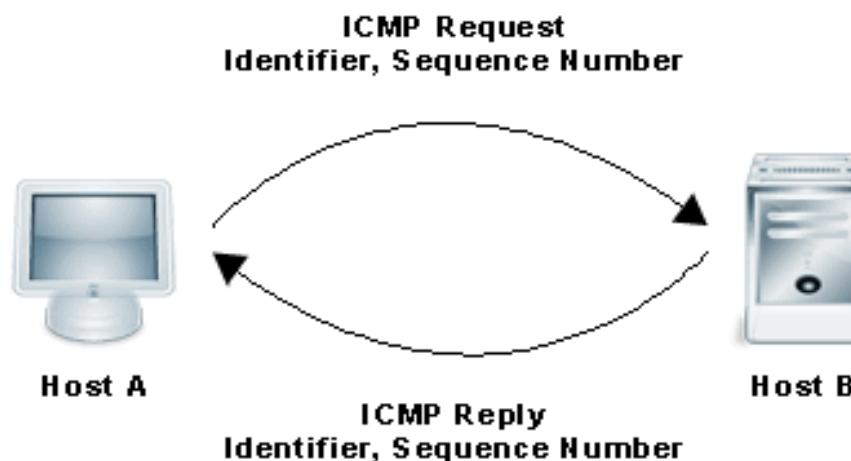
# Type 0/8:Echo Reply/Request



清华大学  
Tsinghua University

- Echo Reply 消息传回去的Identifier和Sequence Number 会和收到 Echo Request 的相同。

0	7 8	1516	23 24	31
Type	Code	Checksum		
Identifier		Sequence Number		
ICMP Echo Reply/Request Datagram				



# 举一个 ping 例子说明



## 1. 192.168.0.80 向192.168.0.10 发出 ICMP Type 8 Request Packet

```
⊕ Frame 8 (74 bytes on wire, 74 bytes captured)
⊕ Ethernet II, Src: 00:40:f4:4c:1d:e5, Dst: 00:00:00:00:00:00
⊕ Internet Protocol, Src Addr: 192.168.0.80 (192.168.0.80), Dst Addr: 192.168.0.10 (192.168.0.10)
⊖ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x465c (correct)
    Identifier: 0x0200
    Sequence number: 05:00
    Data (32 bytes)
```

## 2. 192.168.0.10 向192.168.0.80 发出 ICMP Type 8 Request Packet

```
⊕ Frame 9 (74 bytes on wire, 74 bytes captured)
⊕ Ethernet II, Src: 00:e0:4c:00:00:36, Dst: 00:40:f4:4c:1d:e5
⊕ Internet Protocol, Src Addr: 192.168.0.10 (192.168.0.10), Dst Addr: 192.168.0.80 (192.168.0.80)
⊖ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4e5c (correct)
    Identifier: 0x0200
    Sequence number: 05:00
    Data (32 bytes)
```

# ICMP 应用 — traceroute



## ● 利用 IP Packet 的 TTL 字段来进行测试网络

Packet 经过哪些路由

- router 会在 IP Packet 的 TTL 字段为 0 的时候，回送 ICMP Type 11 Time-to-Live Exceed 消息的功能

## ● 在实作上有两种方法

- 送 UDP 数据包制造 TTL = 0 而回应的状况
- 送 ICMP Echo Request 数据包制造 TTL = 0 回应的状况

# Traceroute 示范



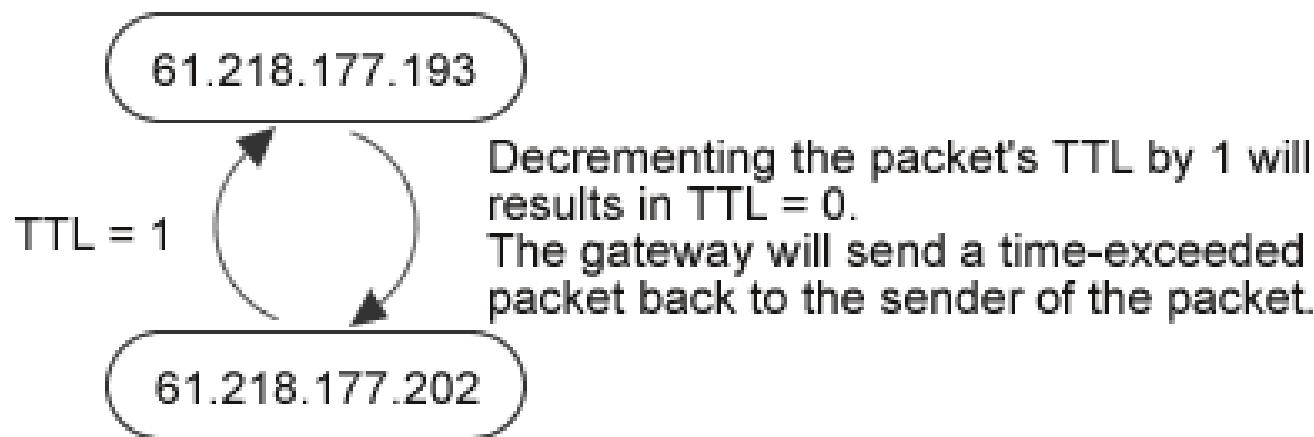
```
# traceroute 168.95.1.1  
traceroute to 168.95.1.1 (168.95.1.1), 30 hops max, 38 byte packets  
1 61-218-177-193.HINET-IP.hinet.net (61.218.177.193) 0.917 ms 0.885 ms 0.844 ms  
2 10.218.177.254 (10.218.177.254) 32.017 ms 34.033 ms 33.376 ms  
3 tc-c6r1.router.hinet.net (168.95.144.202) 31.748 ms 32.338 ms 31.771 ms  
4 tc-b-c12r1.router.hinet.net (168.95.254.129) 31.740 ms 32.338 ms 32.927 ms  
5 210.65.2.22 (210.65.2.22) 36.282 ms 35.720 ms 35.018 ms  
6 tp-s2-c6r9.router.hinet.net (211.22.35.1) 33.380 ms 34.851 ms 35.006 ms  
7 tp-b-c6r2.router.hinet.net (168.95.1.61) 33.369 ms 33.940 ms 33.387 ms
```

# 承上例 — 解释第一个数据包



清华大学  
Tsinghua University

- 第一个 Packet 的 TTL 设为 1，往外传送
- 第一个 Router 是 61.218.177.193，它收到 Packet 后将 TTL 减 1 后(变成 0)
- 于是 61.218.177.193 送一个 ICMP Time Exceed Packet 给 61.218.177.202

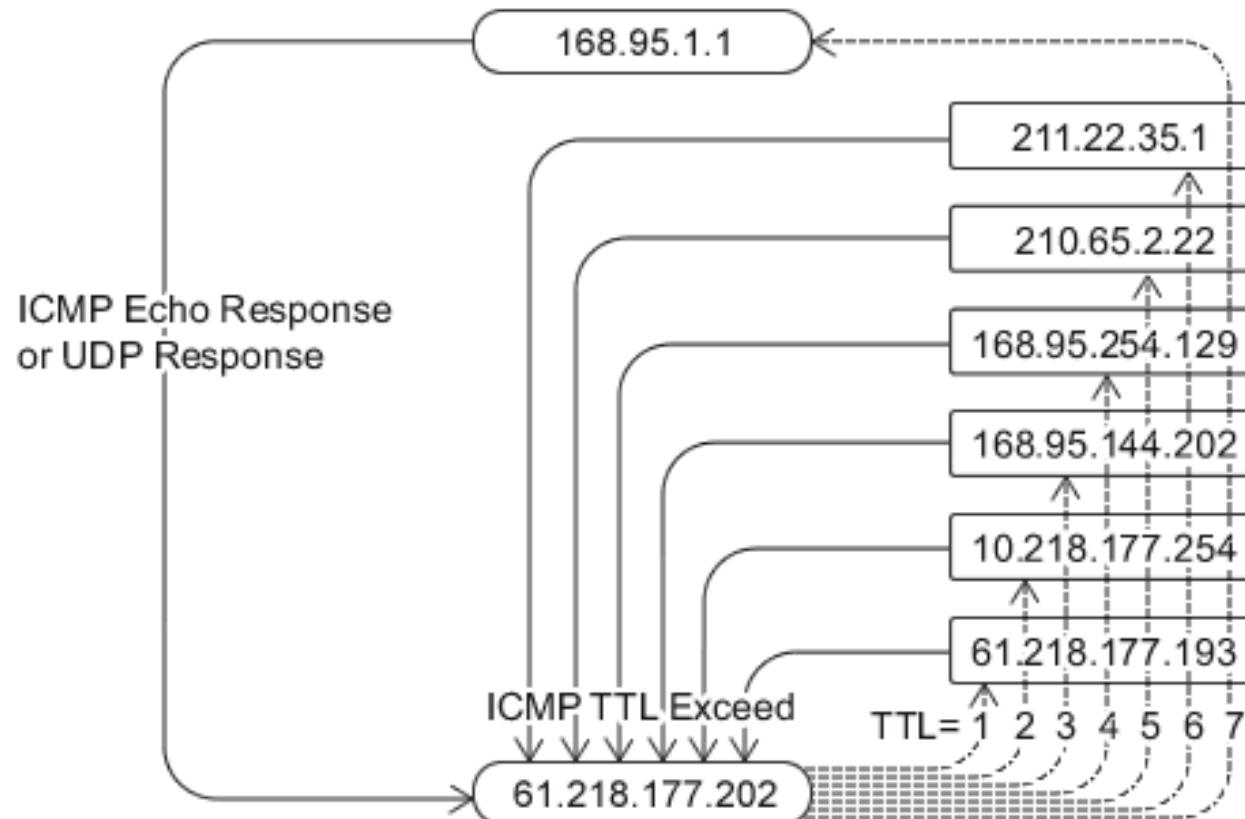


# Traceroute 的数据包流程图

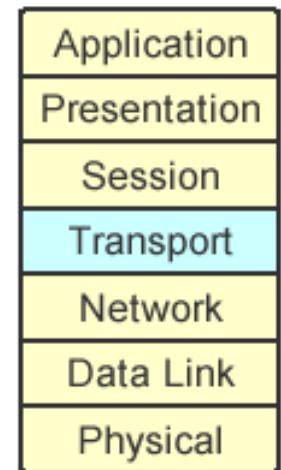


清华大学  
Tsinghua University

- 每个 Packet 的 TTL 会累加，直到抵达 168.



# UDP 用户数据报协议



# UDP 协议

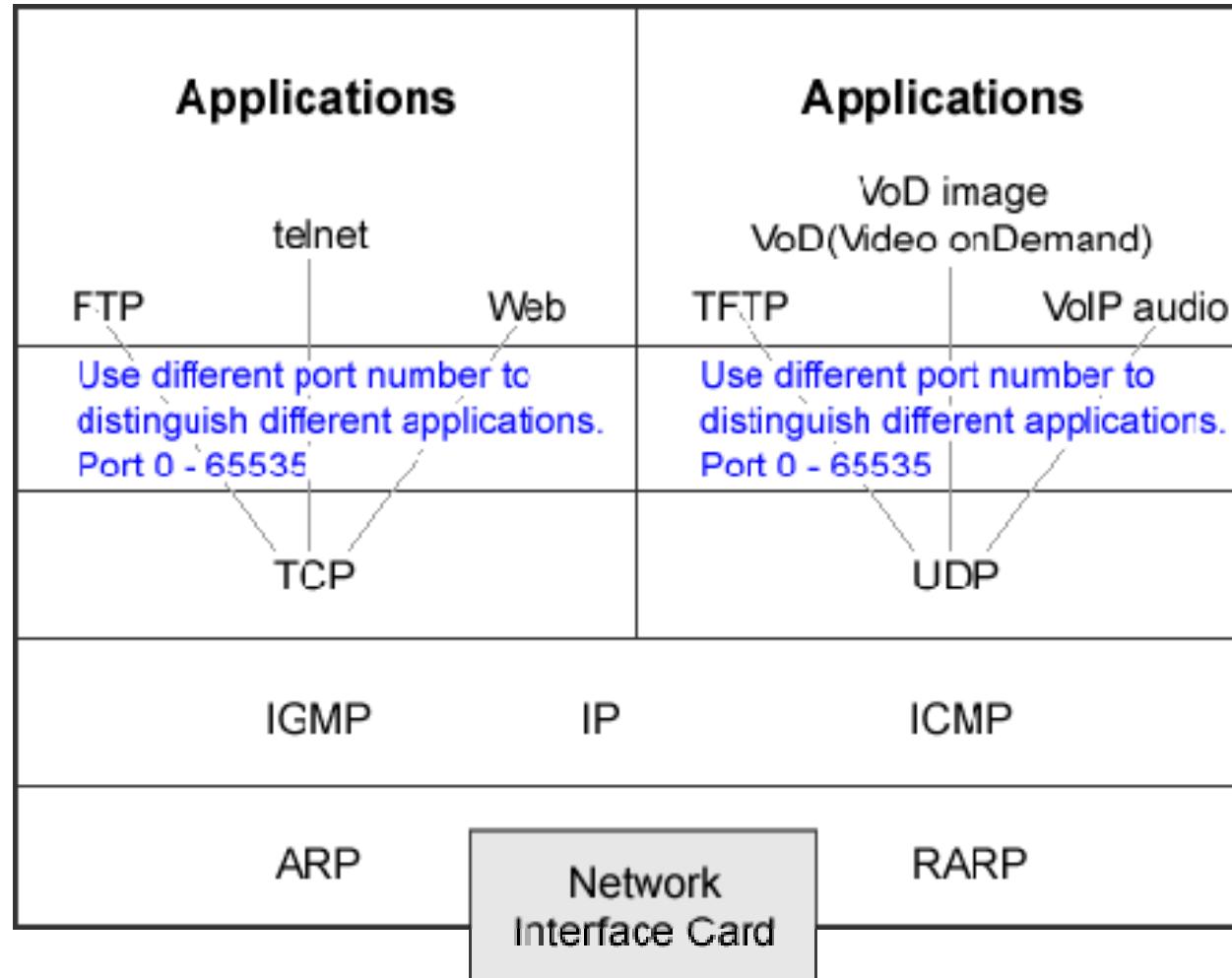


- 定义于 RFC 768 (User Datagram Protocol)
- 提供应用程序能够在最低的协议机制下送消息给其他应用程序。
- 无法保证是不是传送得到或是会被重制或有没有依照传送顺序到达。
- 讲究传输速度且允许一点数据流失 (Data Loss) 或是数据包遗失 (Packet Loss) 的应用程序常用 UDP 协议。例如 NetMeeting、VoIP、MSN 使用语音等。

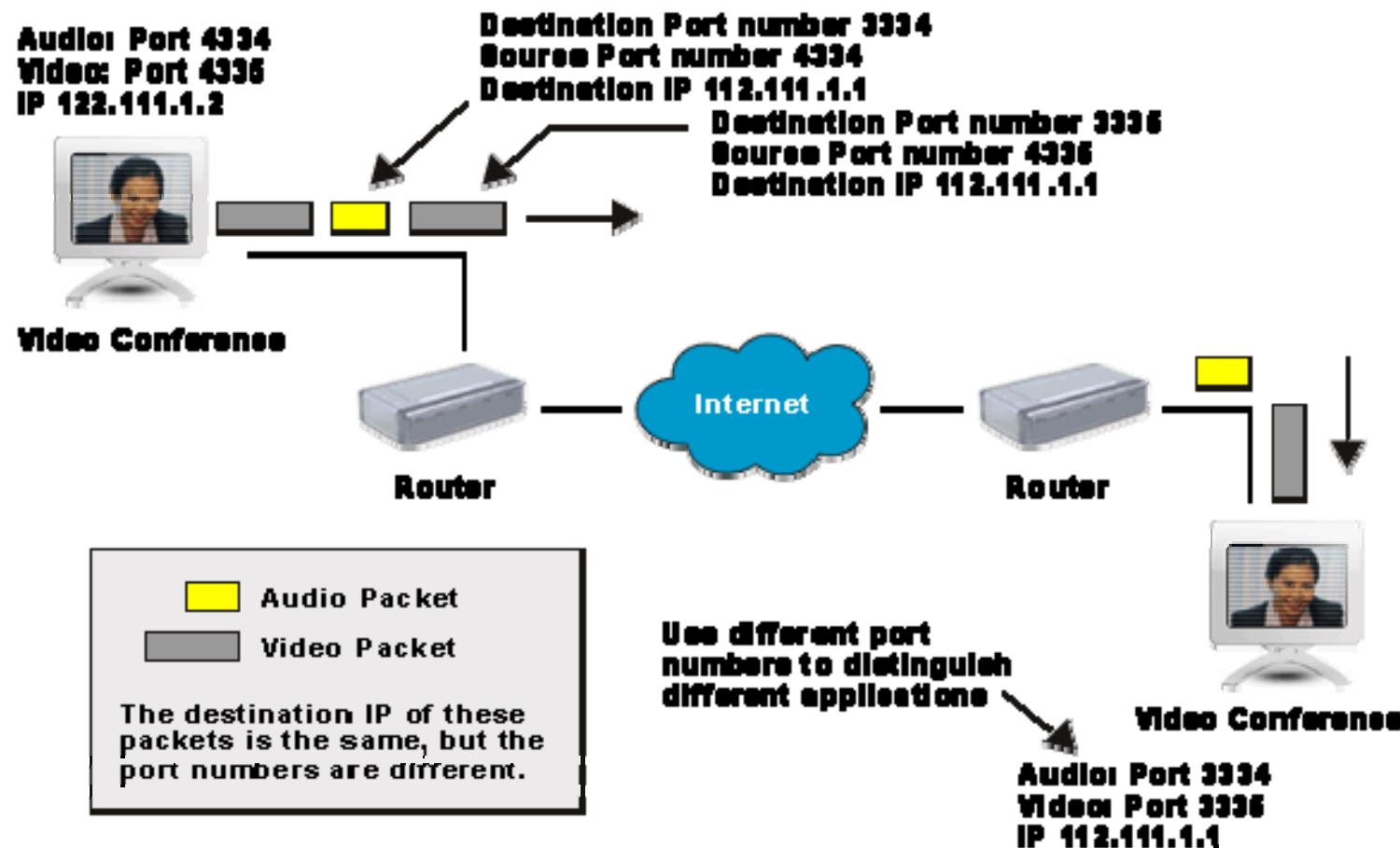
# 端口号(Port Number)与 多路复用(Multiplexing)的概念



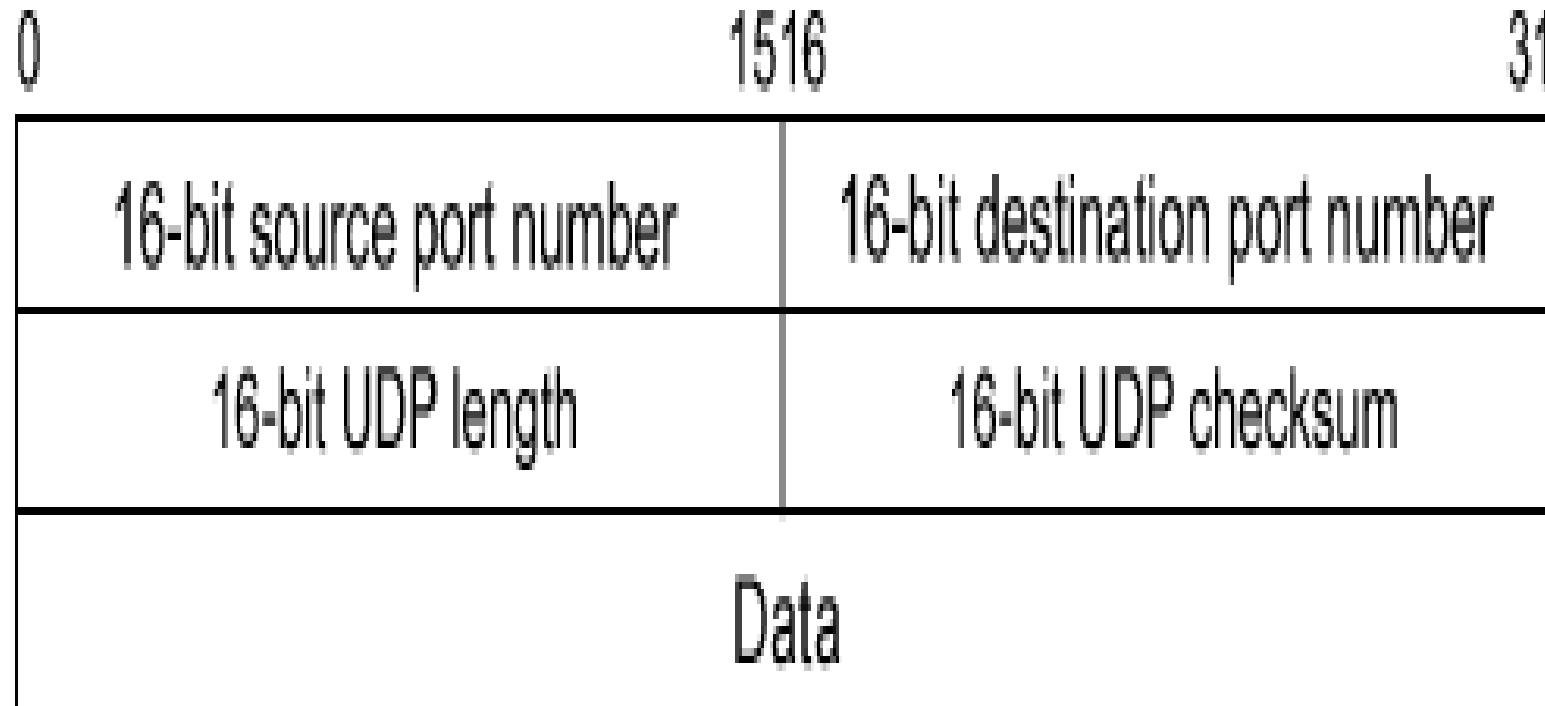
清华大学  
Tsinghua University



# 举例说明 — 视频会议



# UDP 数据格式



# UDP Header



清华大学  
Tsinghua University

capture - Ethereal

File Edit Capture Display Tools Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	00:30:c7:81:48:60	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.0.2? Tell 192.168.0.1
2	0.000129	00:30:c7:81:47:f7	00:30:c7:81:48:60	ARP	192.168.0.2 is at 00:30:c7:81:47:f7
3	0.000151	192.168.0.1	192.168.0.2	UDP	Source port: 32769 Destination port: 9090

+ Frame 3 (47 bytes on wire, 47 bytes captured)  
+ Ethernet II, Src: 00:30:c7:81:48:60, Dst: 00:30:c7:81:47:f7  
+ Internet Protocol, Src Addr: 192.168.0.1 (192.168.0.1), Dst Addr: 192.168.0.2 (192.168.0.2)  
- User Datagram Protocol, Src Port: 32769 (32769), Dst Port: 9090 (9090)  
    Source port: 32769 (32769)  
    Destination port: 9090 (9090)  
    Length: 13  
    Checksum: 0xb72a (correct)  
    Data (5 bytes)  
0000 00 30 c7 81 47 f7 00 30 c7 81 48 60 08 00 45 00 .0..G..0 ..H`..E.  
0010 00 21 00 00 40 00 40 11 b9 78 c0 a8 00 01 c0 a8 .!..@.0. .x.....  
0020 00 02 80 01 23 82 00 0d b7 2a 48 65 6c 6c 6f ....#.... .\*Hello

Filter: / Reset Apply User Datagram Protocol (udp), 8 bytes



# UDP 数据格式 — 详细说明



## ● Source Port :

- 非必须的字段，不使用时填零

## ● Destination Port :

- 表示这个数据报 Datagram 将送达的位置端口

## ● UDP Length :

- 以八进位表示数据报 Datagram 的长度，包括报头和数据。

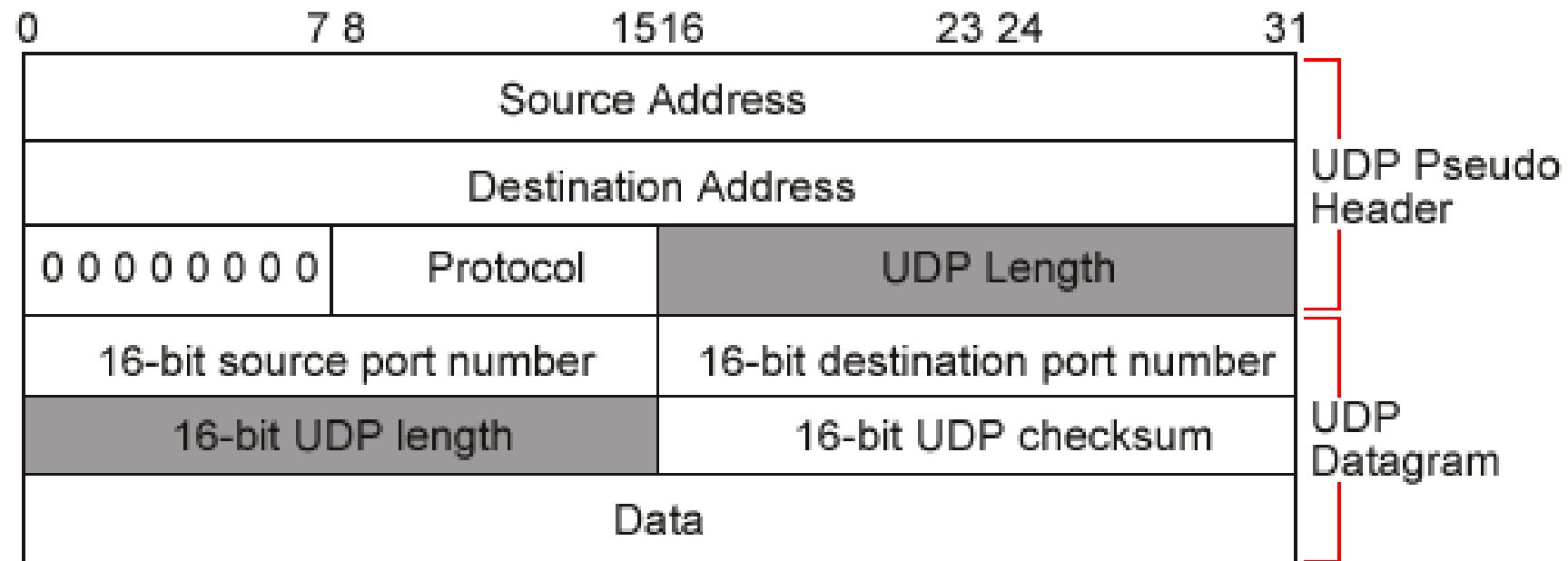
## ● UDP Checksum :

- 将 UDP 的 Pseudo Header、UDP Header 以及 UDP Data的所有数据加总之后，取一的补数，再取 16 bit的一补数，若必要时会在后面加上一个 0，以确保是两个八进位数的数据。

# UDP Pseudo Header 伪报头



清华大学  
Tsinghua University



# UDP Pseudo Header—说明



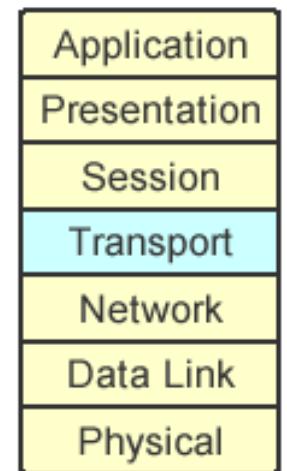
## ● UDP Pseudo Header :

- 将部分 IP header 、 UDP header 集合成一个header

## ● UDP Datagram 的最大长度 :

- 理论上 :  $65535 - 20(\text{IP header}) - 8(\text{UDP header}) = 65507$  Bytes
- 一般 socket 在实作上访问的缓冲区 (Buffer) 有限制 8192 Bytes
- 因此一些使用 UDP 的通信协议诸如 DNS、TFTP、SNMP 等多将其 Datagram 大小限制在 512 Bytes

# TCP 传输控制协议



# TCP 传输控制协议

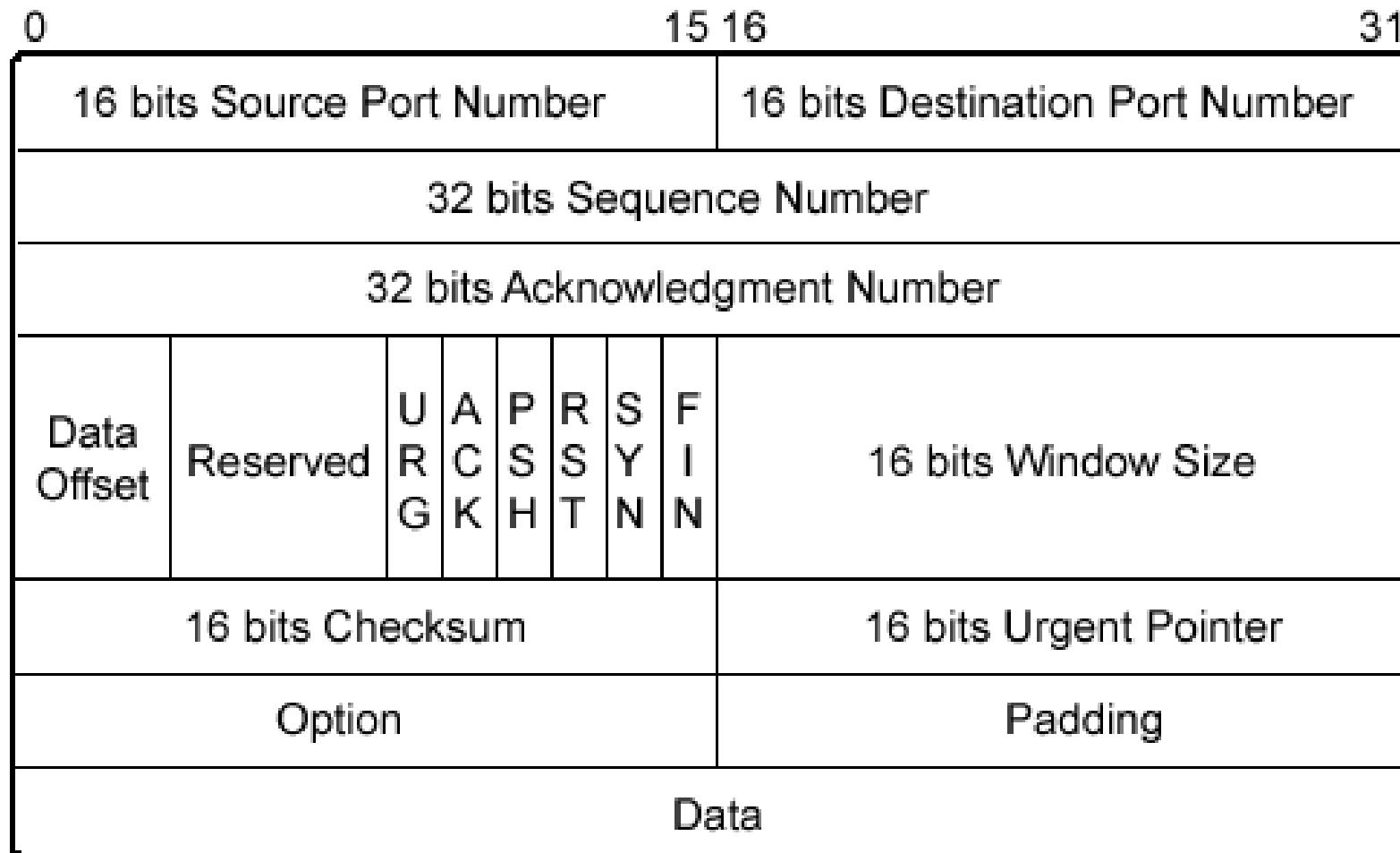


- 定义于 RFC 793 (Transmission Control Protocol)
  - 提供 Connection Oriented (连接导向) 并达成 End-to-End (端对端) Process-to-Process (程序对程序)、Reliable Data Delivery (可信赖的数据递送)。
- TCP 提供以下三个基本功能：
  - Reliability: 克服 Packet Loss, 传送的数据依照顺序交给程序
  - Multiplexing: 藉由不同的端口可使相同的IP 地址可以提供上层不同的服务, 如 FTP, Telnet, HTTP 等
  - Flow Control 或 Congestion Avoidance and Control: 避免网络或接收端拥塞, 以便提供有效率的传输

## TCP 数据格式



清华大学  
Tsinghua University





# TCP Header

No.	Time	Source	Destination	Protocol	Info
1	0.000000	00:30:c7:81:48:60	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.0.2? Tell 192.168.0.1
2	0.000127	00:30:c7:81:47:f7	00:30:c7:81:48:60	ARP	192.168.0.2 is at 00:30:c7:81:47:f7
3	0.000147	192.168.0.1	192.168.0.2	TCP	32883 > 23 [SYN, ECN, CWR] Seq=2942178646 Ack=0 Win=5840 Len=0
4	0.000381	192.168.0.2	192.168.0.1	TCP	23 > 32883 [SYN, ACK, ECN] Seq=3147591265 Ack=2942178647 Win=5792
5	0.000455	192.168.0.1	192.168.0.2	TCP	32883 > 23 [ACK] Seq=2942178647 Ack=3147591266 Win=5840 Len=0

Frame 3 (74 bytes on wire, 74 bytes captured)  
Ethernet II, Src: 00:30:c7:81:48:60, Dst: 00:30:c7:81:47:f7  
Internet Protocol, Src Addr: 192.168.0.1 (192.168.0.1), Dst Addr: 192.168.0.2 (192.168.0.2)  
Transmission Control Protocol, Src Port: 32883 (32883), Dst Port: 23 (23), Seq: 2942178646, Ack: 0, Len: 0

Source port: 32883 (32883)  
Destination port: 23 (23)  
Sequence number: 2942178646  
Header length: 40 bytes  
Flags: 0x00c2 (SYN, ECN, CWR)  
Window size: 5840  
Checksum: 0x3fbf (correct)  
Options: (20 bytes)

0000 00 30 c7 81 47 f7 00 30 c7 81 48 60 08 00 45 10 .0..G..0 ..H`..E, 0010 00 3c 3c 86 40 00 40 06 7c d2 c0 a8 00 01 c0 a8 .<<.0.0.  ..... 0020 00 02 80 73 00 17 af 5e 15 56 00 00 00 00 a0 c2 ..s... ^ .V..... 0030 16 d0 3f bf 00 00 02 04 05 b4 04 02 08 0a 00 55 ..?..... .....U 0040 29 d0 00 00 00 00 01 03 03 00 )..... *
--

# TCP flag (标志)



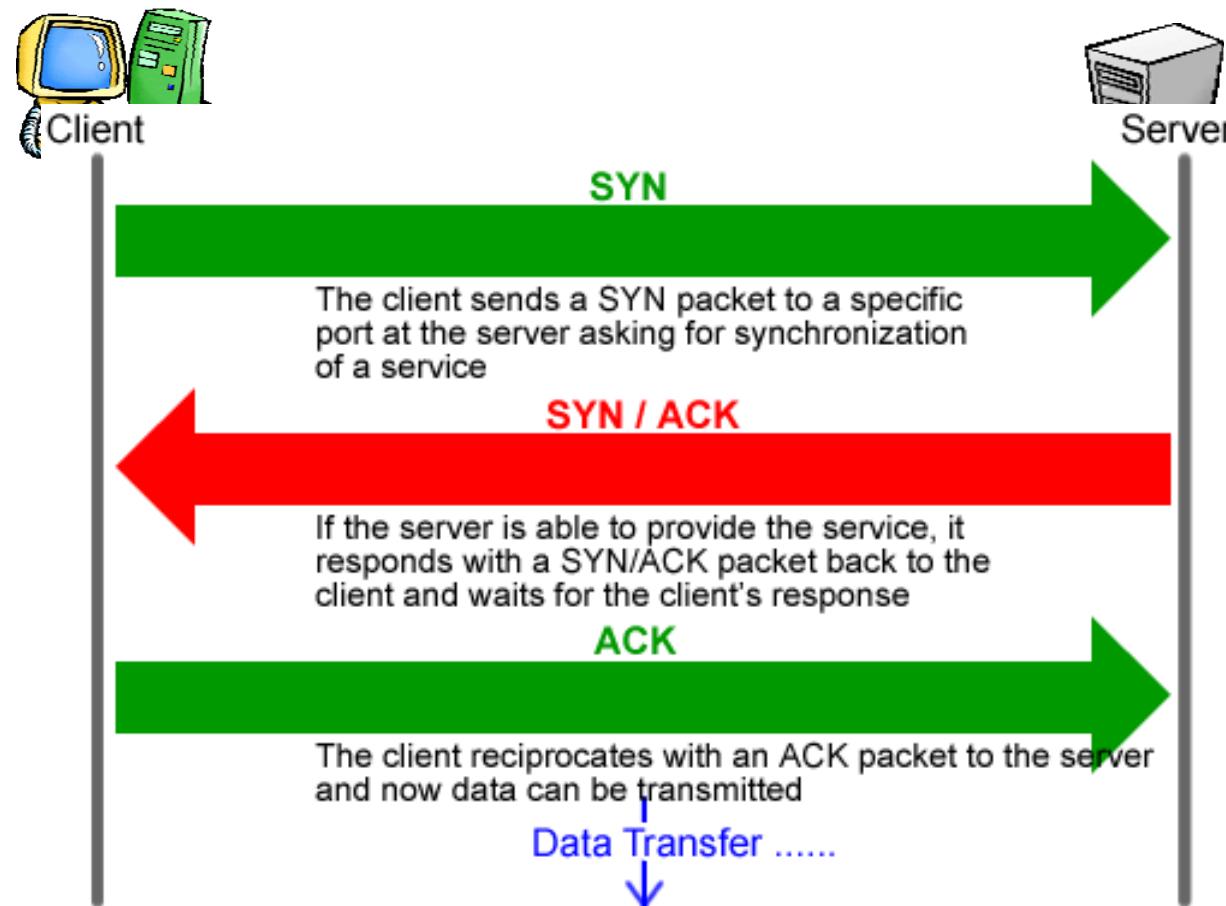
- TCP 连接都有一定的程序，并且分成几个不同的状态(State)，它是透过 TCP Header 字段的 flag 字段来实作

Abbr.	Name	Explanation
ACK	Acknowledge	To verify Acknowledgement Number
SYN	Synchronize	To synchronize Sequence Number
FIN	Finish	To finish sending data
RST	Reset	To reset connection
PSH	Push	To send data out



# TCP 的连接创建

## ● 三次握手 ( three-way handshaking)

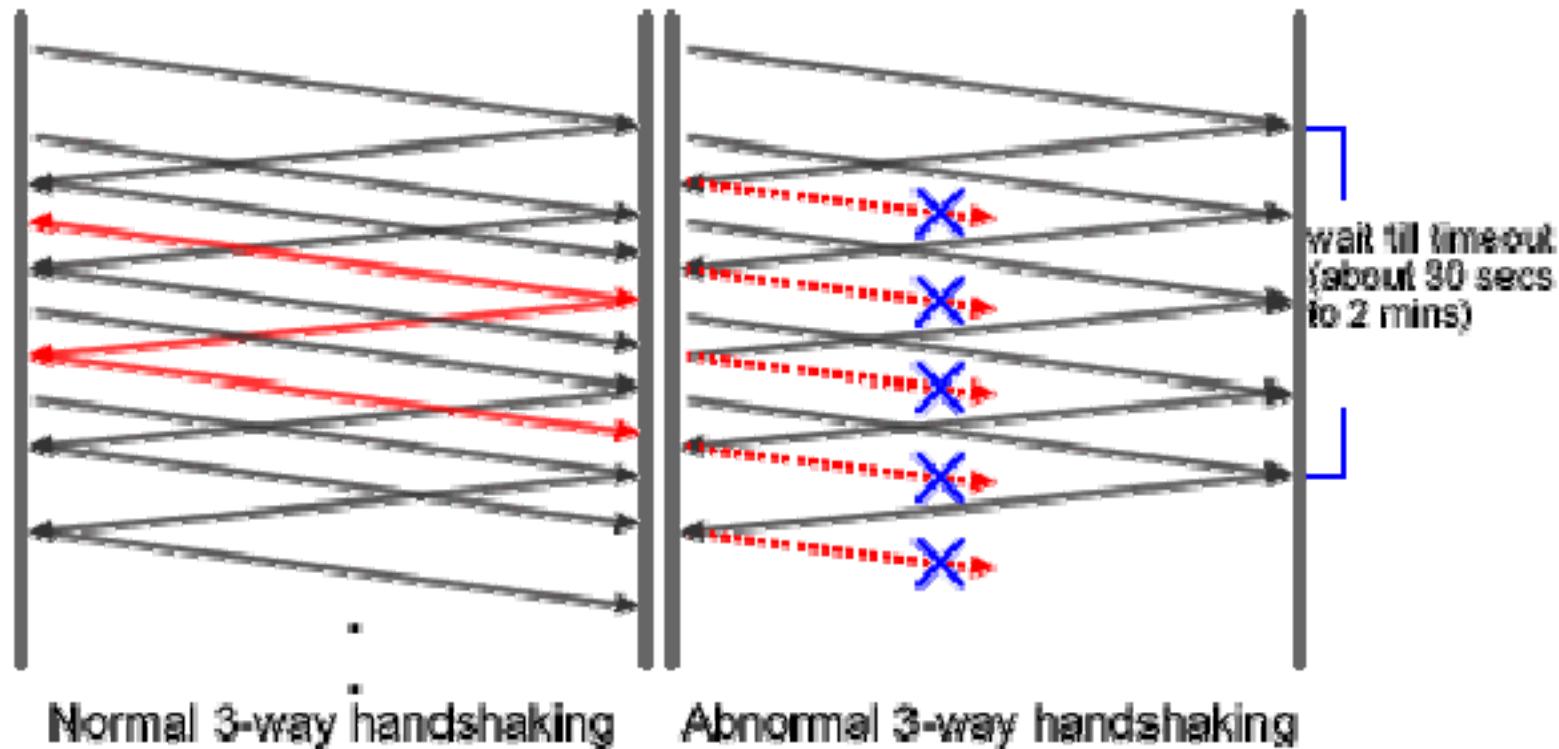


# SYN Flooding (同步广播泛滥)



清华大学  
Tsinghua University

- SYN Flooding 是针对 TCP 三次握手的一种拒绝服务 DoS (Denial of Service) / DDoS (Distributed Denial of Service) 的攻击法



# TCP 的连接退出

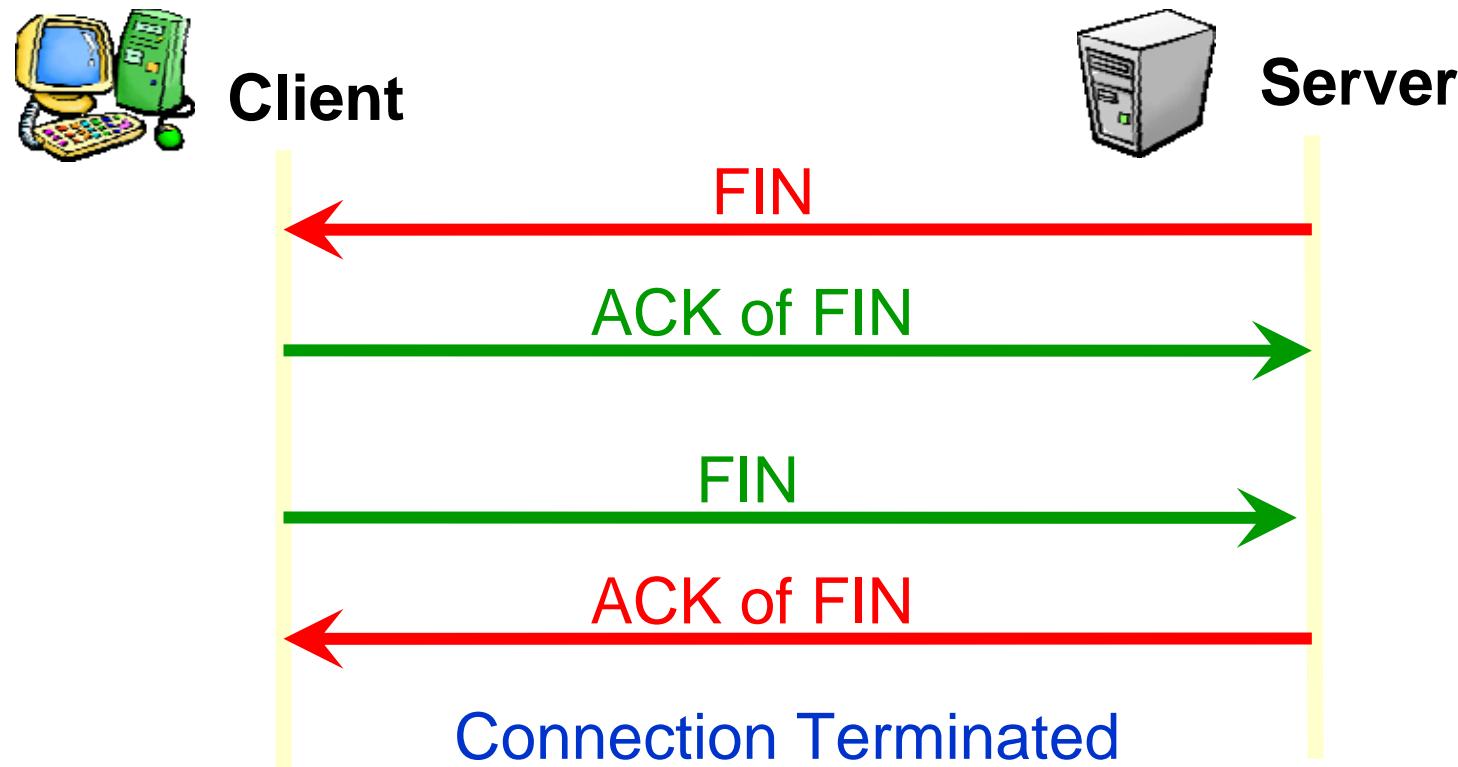


- 由于 TCP 为全双工 (Full-duplex) 的协议，两边的连接都可以独立进行退出连接
- 如果只有单方面退出连接称之为 Half-Close，在 Half-Close 之后，TCP 会要求另外一个方向的连接也退出

# TCP 的连接关闭



- 以下为客户端对服务器主动关闭连接过程的响应



# 用一个完整的例子来说明 TCP



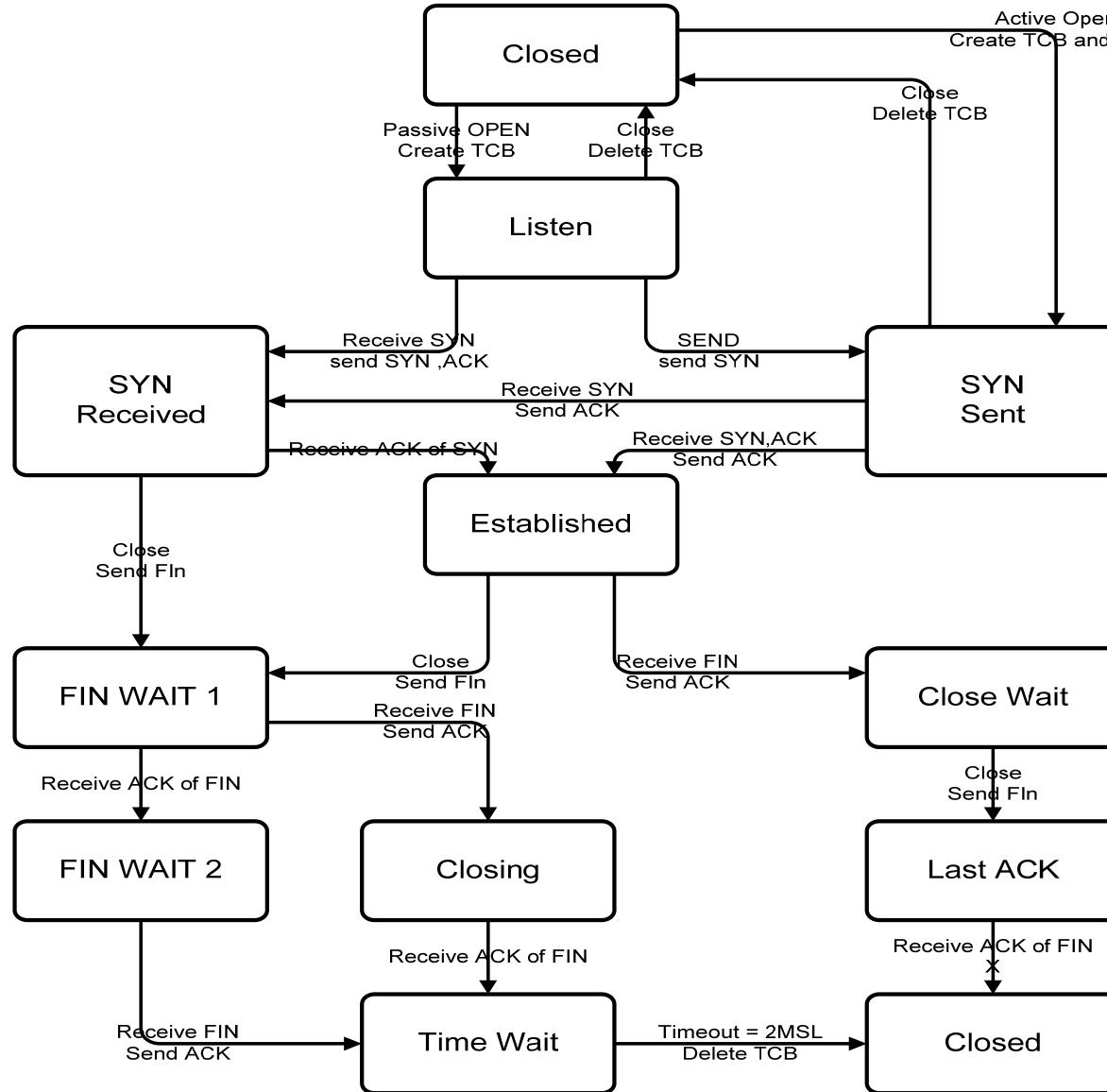
清华大学  
Tsinghua University

No	Source	Destination	Protocol	Info
1	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [SYN] Seq=514208937 Ack=0 Win=5840 Len=0
2	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [SYN,ACK] Seq=1256563805 Ack=514208938 Win=5792 Len=0
3	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [ACK] Seq=514208938 Ack=1256563806 Win=5840 Len=0
4	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [PSH,ACK] Seq=514208938 Ack=1256563806 Win=5840 Len=13
5	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [ACK] Seq=1256563806 Ack=514208951 Win=5792 Len=0
6	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [PSH,ACK] Seq=1256563806 Ack=514208951 Win=5792 Len=12
7	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [ACK] Seq=514208951 Ack=1256563818 Win=5840 Len=0
8	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [FIN,ACK] Seq=1256563818 Ack=514208951 Win=5792 Len=0
9	192.168.0.1	192.168.0.2	TCP	32784 > 9999 [FIN,ACK] Seq=514208951 Ack=1256563819 Win=5840 Len=0
10	192.168.0.2	192.168.0.1	TCP	9999 > 32784 [ACK] Seq=1256563819 Ack=514208952 Win=5792 Len=0

# TCP Connection State Diagram



清华大学  
Tsinghua University

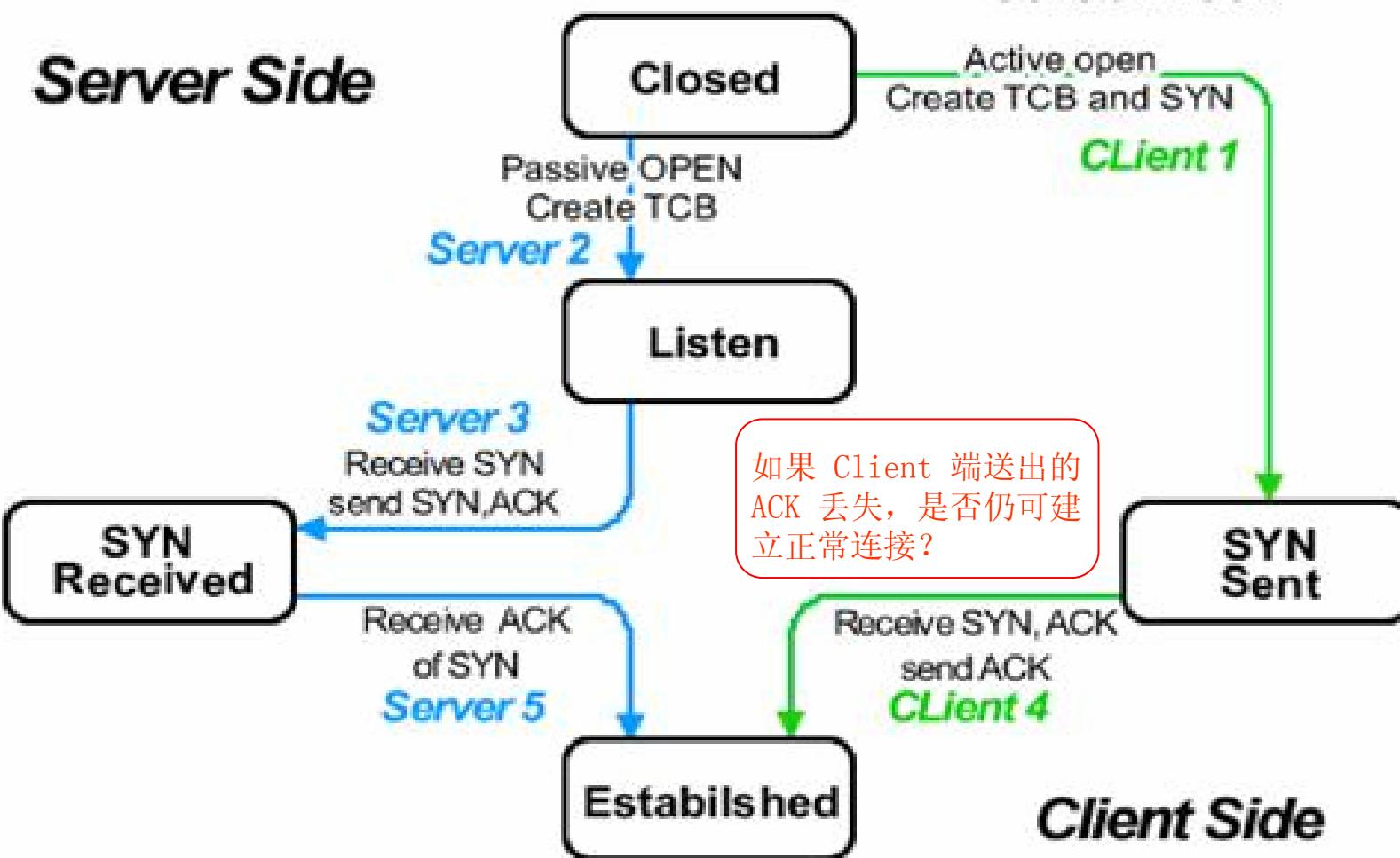


# TCP Connection State Diagram 的 连接创建部分



清华大学  
Tsinghua University

TCB : Transmission  
Control Block



# TCP Connection Establishment 步骤



Client Side		Server Side	
1	Client initiates a connection to server	1	Server listens on a fixed port
2	Client sends SYN to server and then transit to SYN Sent state		
		2	Server receives SYN and then sends SYN/ACK to client, and transit to SYN Received state
3	Client receives SYN/ACK and then sends ACK to server, and transit to Established state		
4	Client connection is established	3	Server receives ACK (response) of SYN and then transit to Established state
		4	Server connection is established

# TCP Connection State Diagram 的 Disconnection Establish 部分



清华大学  
Tsinghua University

## ● 此端先关闭

- Established → FIN\_WAIT\_1 → FIN\_WAIT\_2 → Time\_Wait → Closed

## ● 另一端先关闭

- Established → CLOSE\_WAIT → LAST\_ACK → Closed

## ● 两端同时关闭

- Established → FIN\_WAIT\_1 → Closing → Time\_Wait → Closed

# TCP 协议的可靠性



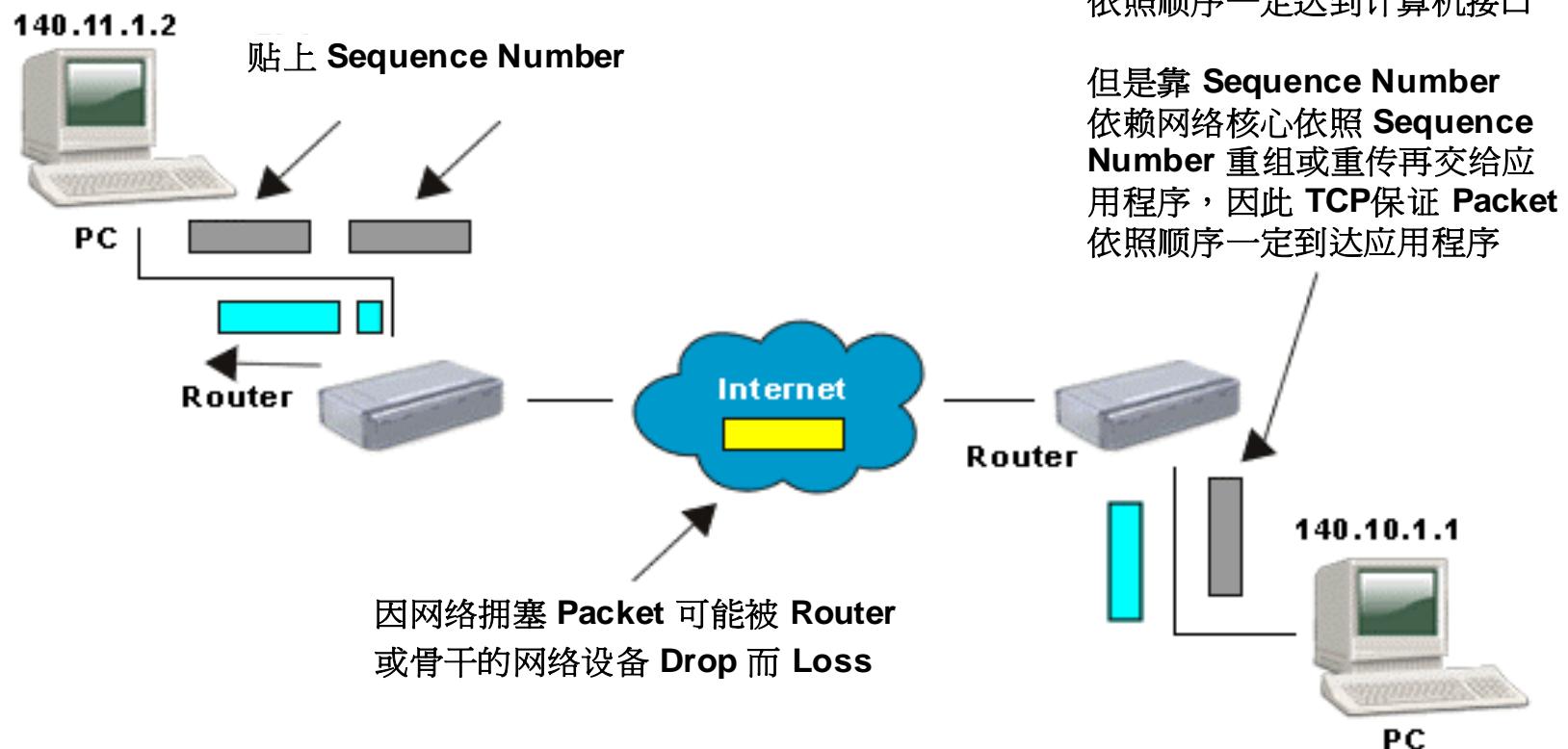
- Checksum : 确保 Packet 正确
- Sequence Number : 用来确保 Packet 依照顺序
- Positive Acknowledgment (ACK) : Packet Loss 或错误重传的策略
- Window Size: 作流量控制 (Flow Control)

# TCP协议的可靠性—Checksum



- 如同 UDP，TCP 的 Checksum 也是由 Pseudo Header、TCP Header 与 Data 一起计算而得
- 如果接收端计算结果与 Checksum 不同，就会认为这 TCP Packet 错误而要求传送端重传

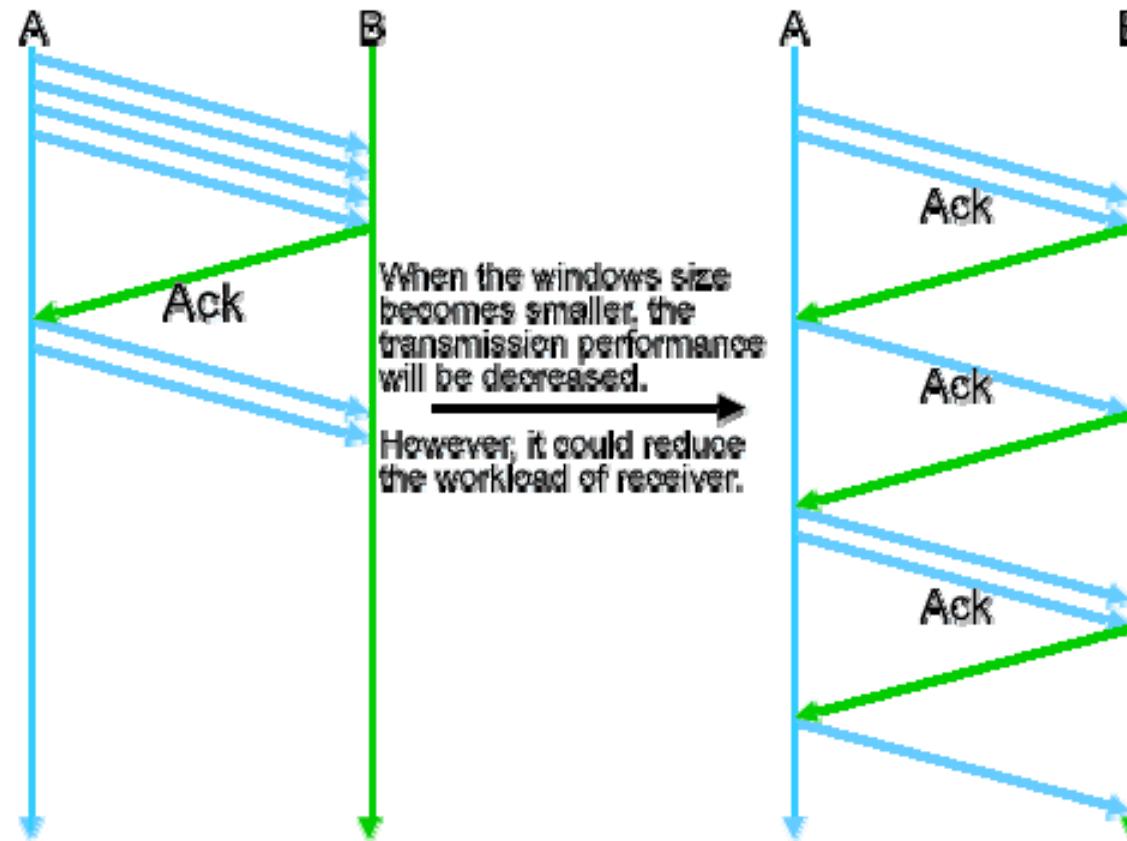
# TCP 协议的可靠性 – Sequence Number 的运用及 ACK



# TCP 协议的可靠性 – Window Size 流量控制



清华大学  
Tsinghua University



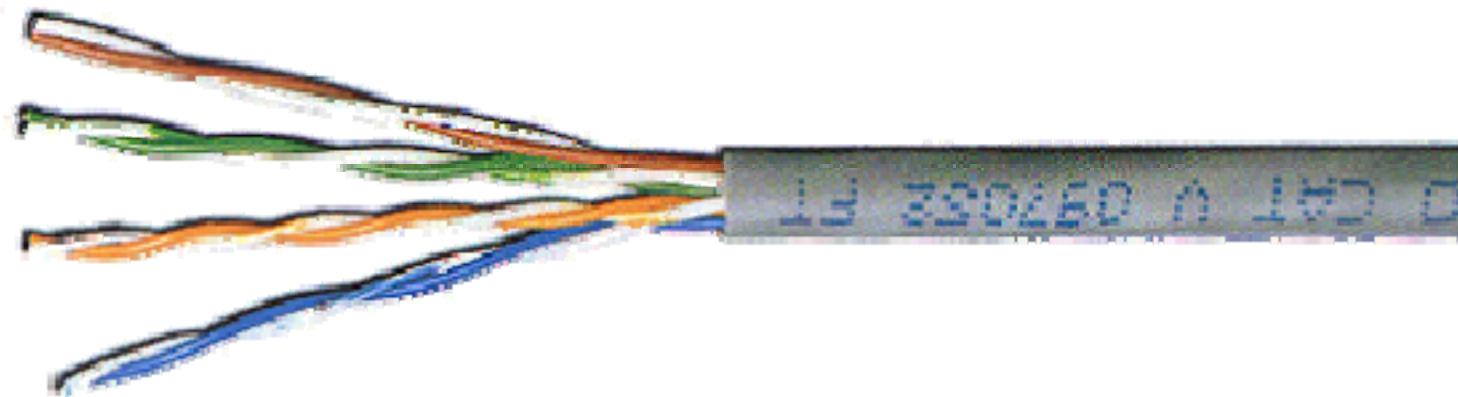
# 以太网 (Ethernet)

# 简介



- 使用最广泛的局域网类型
- 共享网络介质（Media）：
  - 10 Base T 及 100 Base T 使用 8 芯（4 对线）的 UTP (Unshielded Twisted Pair 无遮蔽双绞线) 网络线
  - 接头为 RJ-45，接法有平行接法和串接接法

# Category 5 以太网络线



上图为当前 Ethernet 所用的主要线材，每条导线是由两组双绞线所组成，一组做发送一组则做接收用途，故可做全双工传输；每段导线只能接一部计算机，导线两端各接一个RJ—45接头，一端接计算机网络卡插座，一端接集线器，最长可达100米

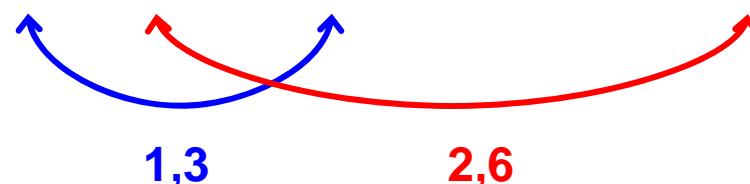


# Category 5 以太网络线

## ● EIA/TIA

**568A:** 白 绿、绿、白 橙、蓝、白 蓝、**橙**、白 棕、棕

**568B:** 白 橙、**橙**、白 绿、蓝、白 蓝、**绿**、白 棕、棕



## ● 功能

- 平行线: 568A - 568A / 568B - 568B
  - ◆ 计算机与 Hub / Switch 的连接
- 跳线(交叉线): 568A - 568B / 568B - 568A
  - ◆ 功能在于 Hub / Switch 的串接或两部计算机直接相连

# 传送必须与接收对应

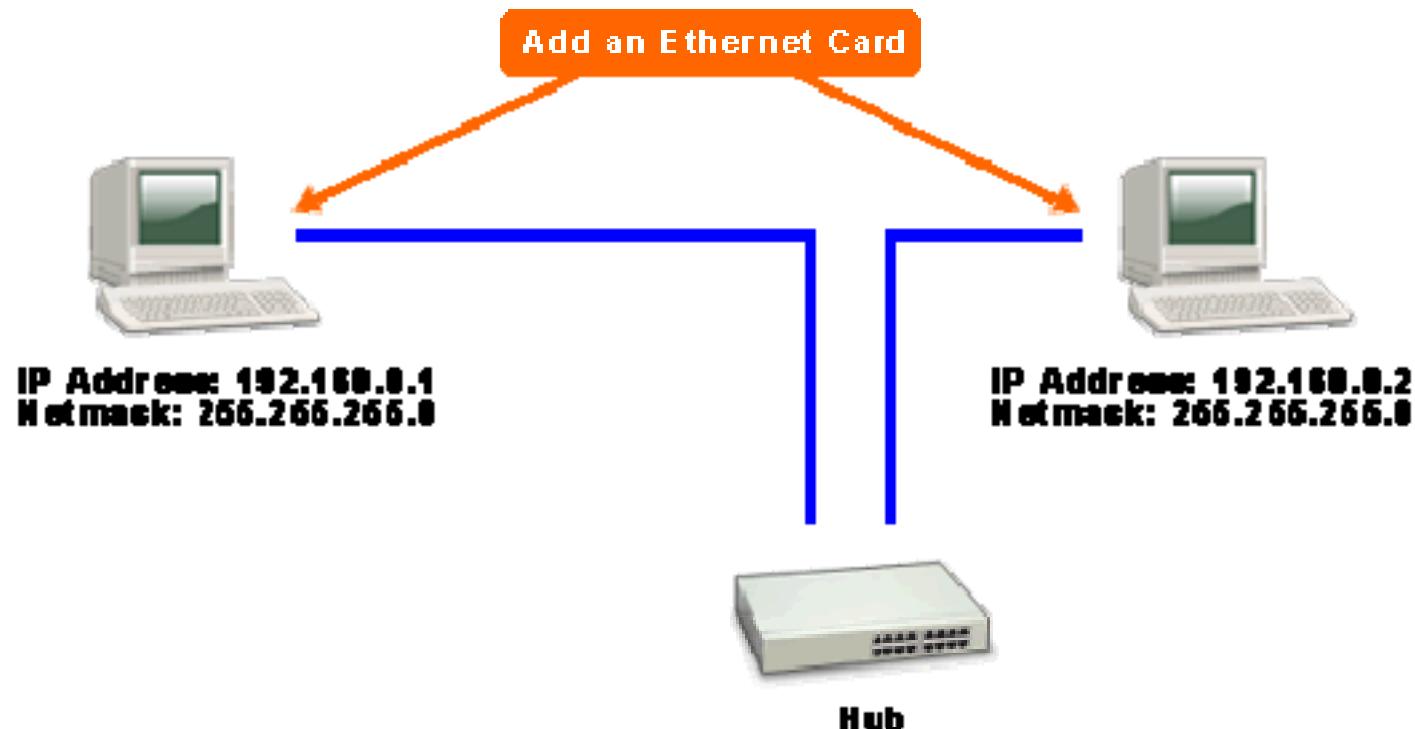


- 其中**White—Orange** 代表**Transmit+** (传送+)，**Orange** 代表**Transmit-**(传送-), 而 **White—Green** 代表**Receive+** (接收+), **Green** 代表**Receive-** (接收-)。
- 信号+、-是电路设计时差分电路(Differential Circuit) 的+ -，主要是为了抗共模噪声(Common Mode Noise)
- 基本规则：传送TX+ 连接接收RX+，传送TX- 连接接收RX-
- 网卡连接HUB的非上行口(Uplink Port)，或HUB连接HUB（一边是上行口）：使用EIA/TIA 568B 平行接法，两边都相同接法。因为HUB Port 上的1,2为RX+，RX-而 3,6 为 TX+，TX-
- 网卡连接网卡或网卡连接Uplink Port，或HUB连接HUB（不用上行口）：一边使用EIA/TIA 568A Crossover 接法
- 补充：当前很多HUB自动进行识别

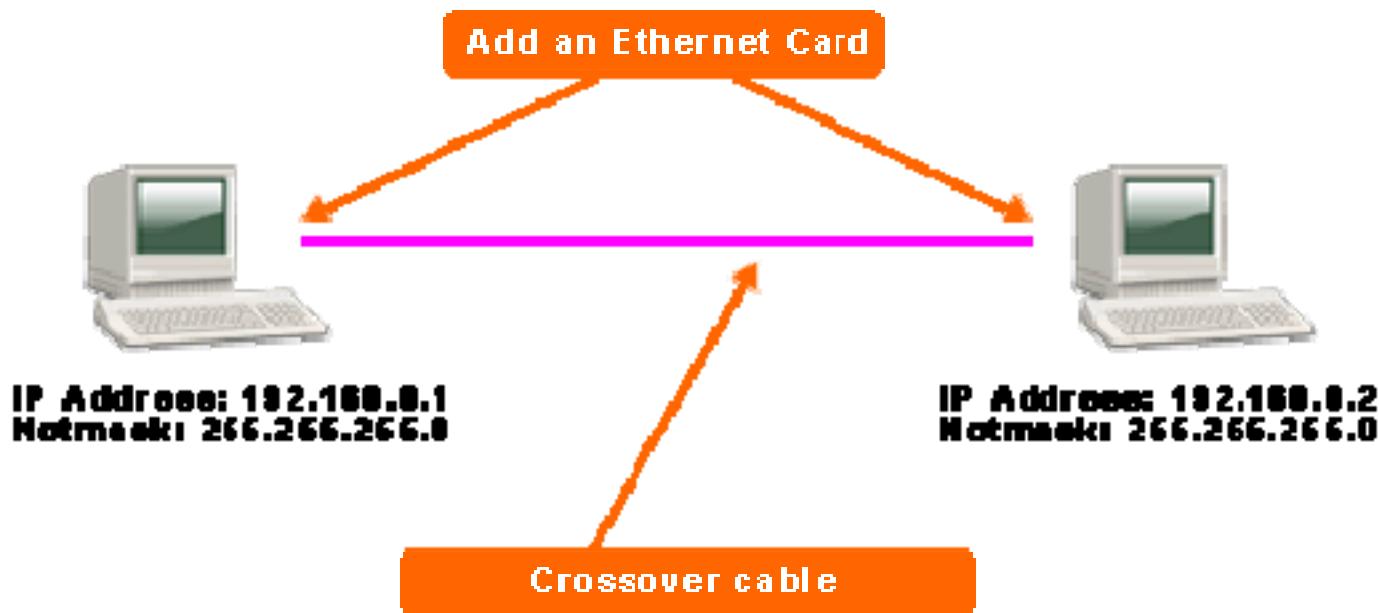
# 使用HUB与平行(Straight)网络线连接



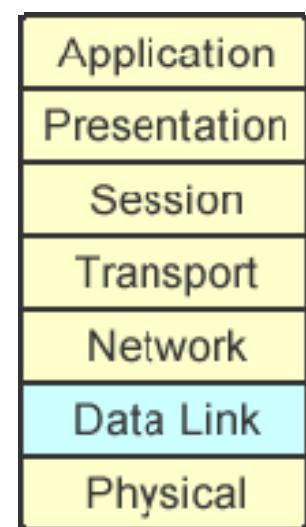
清华大学  
Tsinghua University



# 使用 Crossover 网络线连接



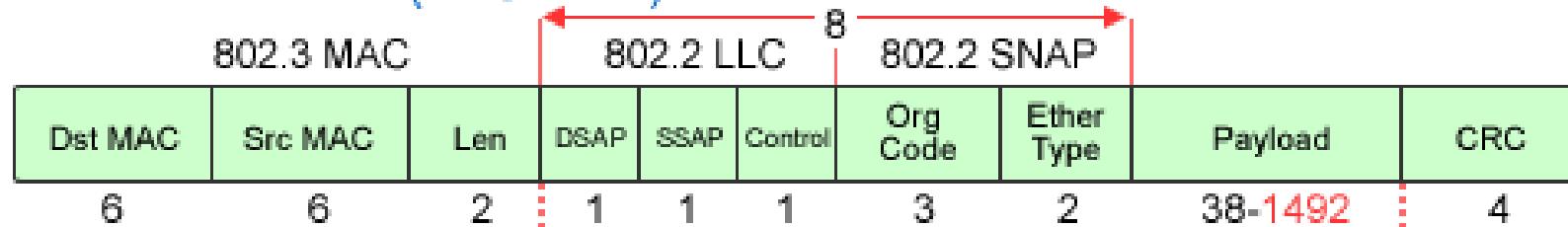
# IEEE 802 与 Ethernet



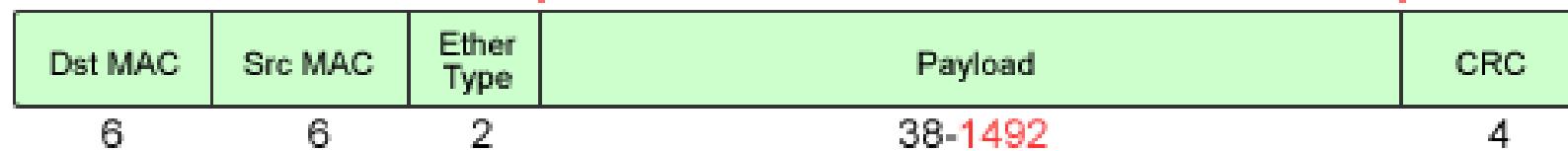
# IEEE 802 & Ethernet 比较



IEEE 802.2 / 802.3 (RFC 1042)



Ethernet (RFC 894)



# IEEE 802



802.3 MAC			802.2 LLC			802.2 SNAP				
Dst MAC	Src MAC	Len	DSAP	SSAP	Control	Org Code	Ether Type	Payload	CRC	
6	6	2	1	1	1	3	2	38-1492	4	

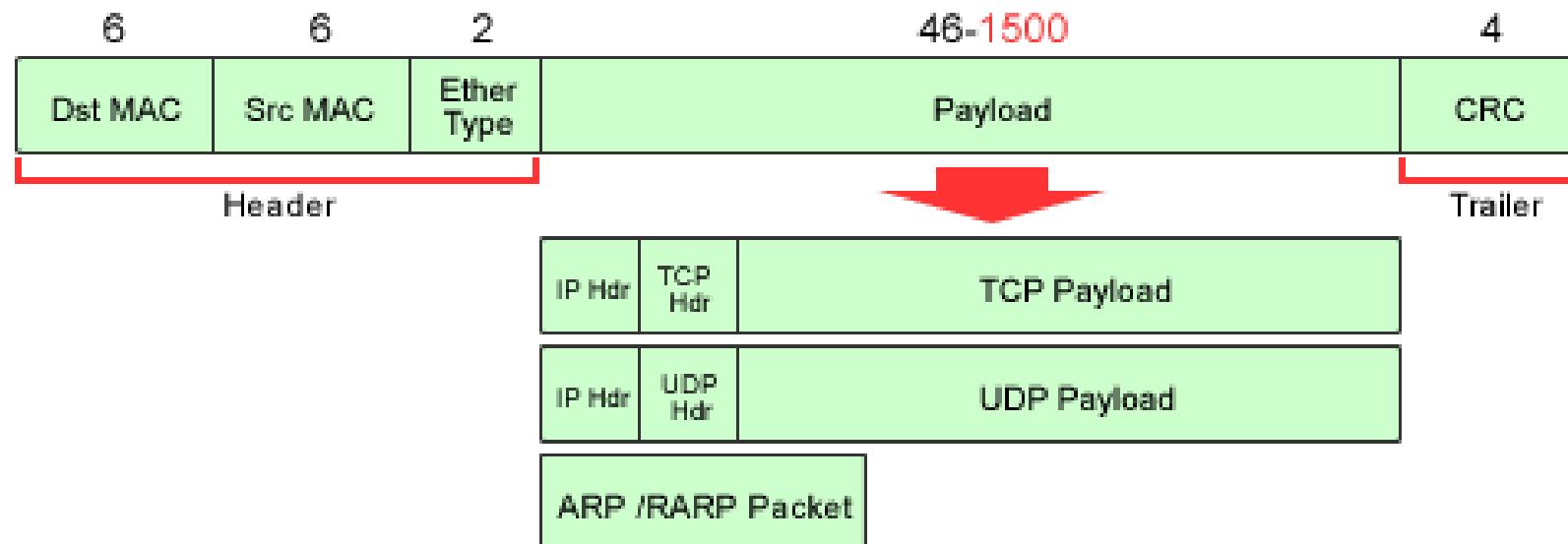
- Dst MAC ( Destination MAC Address)  
目的的物理地址
- Src MAC (Source MAC Address)  
来源的物理地址
- Len (Length)  
从头到尾的长度 (不含CRC)

# 协议域说明



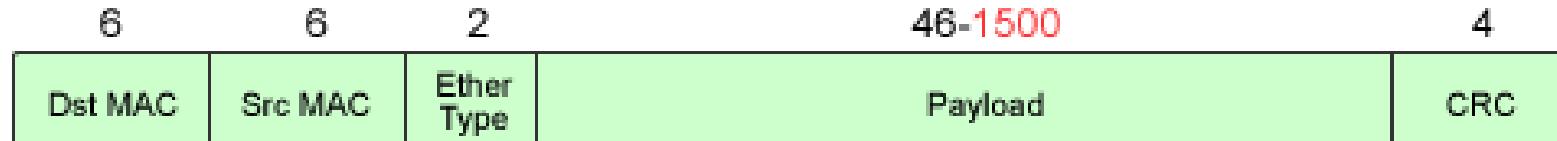
- 802.2 逻辑链路控制 LLC (Logical Link Control)
  - DSAP (Destination Service Access Point)
  - SSAP (Source Service Access Point)
  - Control
- 802.2 SNAP (Sub-network Access Protocol)
  - Org Code
  - Ether Type: 与以太网络的类别同
- CRC (Cyclic Redundancy Check)
  - 错误检查码
  - Trailer 报尾

# 以太封装 Ethernet Encapsulation





# 协议域说明



- Dst MAC (Destination MAC Address): 目的地物理位置
- Src MAC (Source MAC Address): 来源地物理位置
- Ether Type
  - 0x0800: IPv4
  - 0x08DD: IPv6
  - 0x0806: ARP
  - 0x8035: RARP
- Payload: 上层网络层的数据包 CRC (Cyclic Redundancy Check)
  - 错误检查码
  - Trailer报尾

# 最大传输单位 MTU

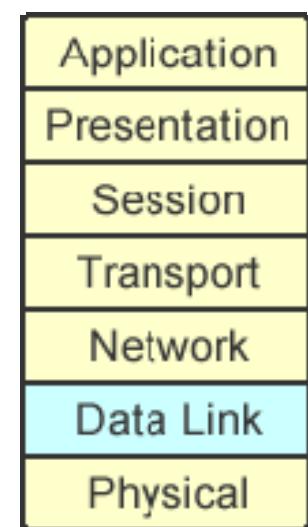


## ● Maximum Transmission Unit

- 数据链路层中对有效载荷(payload)传输的最大限制即通过TCP/IP 协议所传输的数据包最大有多少字节)
- 网络层会因数据链路层传输最大限制而做分段(fragment)

Network Category	MTU (Bytes)
FDDI	4352
Ethernet	1500
IEEE 802.3 / 802.2	1492

# MAC Address





# MAC 地址

- 每张网卡出厂时即拥有一个全世界独一无二的地址
- 目的在于直接相邻设备间的寻址，适用于同一网络中
- 由6个 bytes 组成，例：00: 10: F3: 03: 28: B0
- MAC Layer 的地址，负责和物理层沟通

6 Bytes MAC Address						Vendor Code
	00	10	F3	03	28	B0

MSB1 is the trailing bit on Ethernet  
MSB is the beginning bit in Token-Ring and FDDI

# MAC 地址



## ● 广播 Broadcast Address

- FF: FF: FF: FF: FF: FF

## ● 群播 Multicast Address

- 01: 00: 5E: xx: xx: xx

# Linux 指令： ifconfig



- ifconfig – 网络接口控制程序
  - 由命令及其输出结果可看出系统的设计以及相关设置的作用
  - 举例：ifconfig eth0
- 
- ```
● eth0 Link encap:Ethernet Hwaddr 52:54:AB:ED:6F:61
inet addr:210.34.6.89 Bcast:210.34.6.127 Mask:255.255.255.128
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:46299 errors:0 dropped:0 overruns:0 frame:189
      TX packets:3057 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      Interrupt:5 Base address:0xece0
```

# 网络卡接收数据包(Packet) 的依据



- 在以下四种情况中， 网络卡会接收数据包
  - Destination MAC 地址为该网络卡的 MAC 地址
  - 广播地址 (Broadcast Address)
  - 群播地址 (Multicast Address)
  - 当网络卡在混杂模式 Promiscuous Mode 时
    - ◆ Ethereal 预设即是 Promiscuous Mode

# MAC 地址与 IP 地址



## ● 为何不使用 MAC 地址为寻址方法？

- 各 MAC Protocol 的 MAC 地址 无法统一
- MAC 地址无法有任何的阶层观念

## ● 设计 IP 地址供寻址使用

## ● IP 地址的设计精神：

- 阶层式
- IPv4 使用 32 Bits 地址、IPv6 使用 128Bits 地址

# 第一章 附录

# 第一层 物理层(Physical Layer)



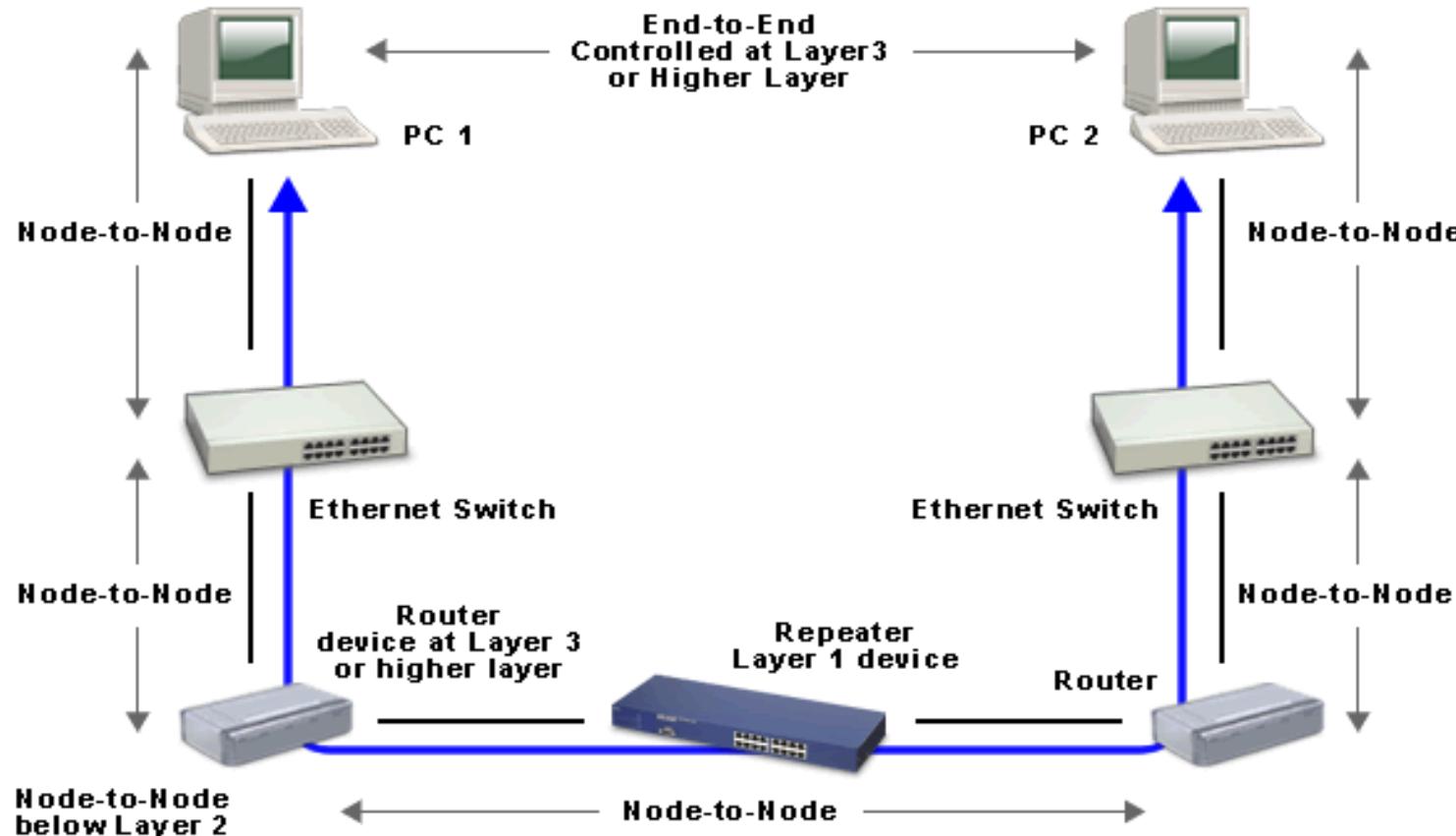
- 物理层主要是负责实体传输介质的规格制定
- 例如电缆(Cable)、光纤(Fiber)、双绞线 (Twisted Pair) 以及连接端的规格，其中亦包括了传输的信号种类及转换等



## 第二层 数据链路层(Data Link Layer)



- 确保节点与节点间(node-to-node)透过物理层能够正确有效的传输





- 数据链路层负责将原始数据编码封装为帧（Frame），并增加抗噪音干扰、线路不稳等传输错误控制与重送策略、或做流量控制（Flow Control）等
- Ethernet网卡中有唯一的 48 bits 的编号，称之为物理层地址（MAC Address 或是 Physical Address）
  - 其中 MAC 代表 Media Access Control（介质访问控制）。



# 第三层 网络层 (Network Layer)



清华大学  
Tsinghua University

- 主要功能为网包 (Packet) 的路由 (Routing) 与选择路由 (Route)，网包的分段 (Fragmentation) 等。
- 该层中最著名的代表，就是因特网不可或缺的 TCP/IP 中的 IP 协议 (Internet Protocol, 因特网协议)。
- 而在网络层中，有一最重要的地址观念：IP 地址。IP 地址即是一般常说 IP 为 123.132.211.1 这样的一串数字。
- 此层在传送中，由于是使用非连接为导向，所以仅能以尽力传输(best effort)方式传送出去，不保证数据会送达，以方便上层(传输层TCP/UDP)来控制



# 第四层 传输层 (Transport Layer)



清华大学  
Tsinghua University

- 连接创建与删除 (Connection Establishment /Tearing Down)
  - 如 TCP 传输控制协议 (Transmission Control Protocol)
- 端对端 (End-to-End)、流量控制 (Flow Control)
  - (不是Node-to-Node Flow Control)
- 拥塞控制 (Congestion Control)



# 第五层 会话层 (Session Layer)



- 设立Layer 5以上的装置与设备因上层应用的需求而创建的逻辑上的链接 (Logical Link)
  - 例如：Microsoft NetMeeting、MSN 等多人会谈的 Session 观念。
  - 然而切记这只是OSI 7 layer reference Model，事实上Microsoft NetMeeting、MSN 主要使用了 H. 323, SIP 会话启动协议(Session Initiation Protocol, RFC 3261)，而在这些标准中（例如 SIP），没有提到这是OSI Session Layer
- 建立Session 的目的在于：决定参与这Session的设备能够应用影音与文字通信做正常的沟通及决定数据的压缩与编码方法



# 第六层 表示层 (Presentation Layer)



清华大学  
Tsinghua University

- 主要使应用层能了解与翻译传输的数据内容
  - 例如不同的字符集 (Character Sets), 不同的文字编码方式例如 Base64、Uuencode 或 MIME type, 或不同的文件结构 (File Structure)、不同的加解密方法等。
  - XDR (External Data Representation) 外部数据表示法表示在应用层的应用程序 NFS (Network File System) 必须了解远程网络的文件结构, 才能将远程的网络文件经由表示层解析结构挂入本地的 (Local) 文件结构中。



# 第七层 应用层 (Application Layer)

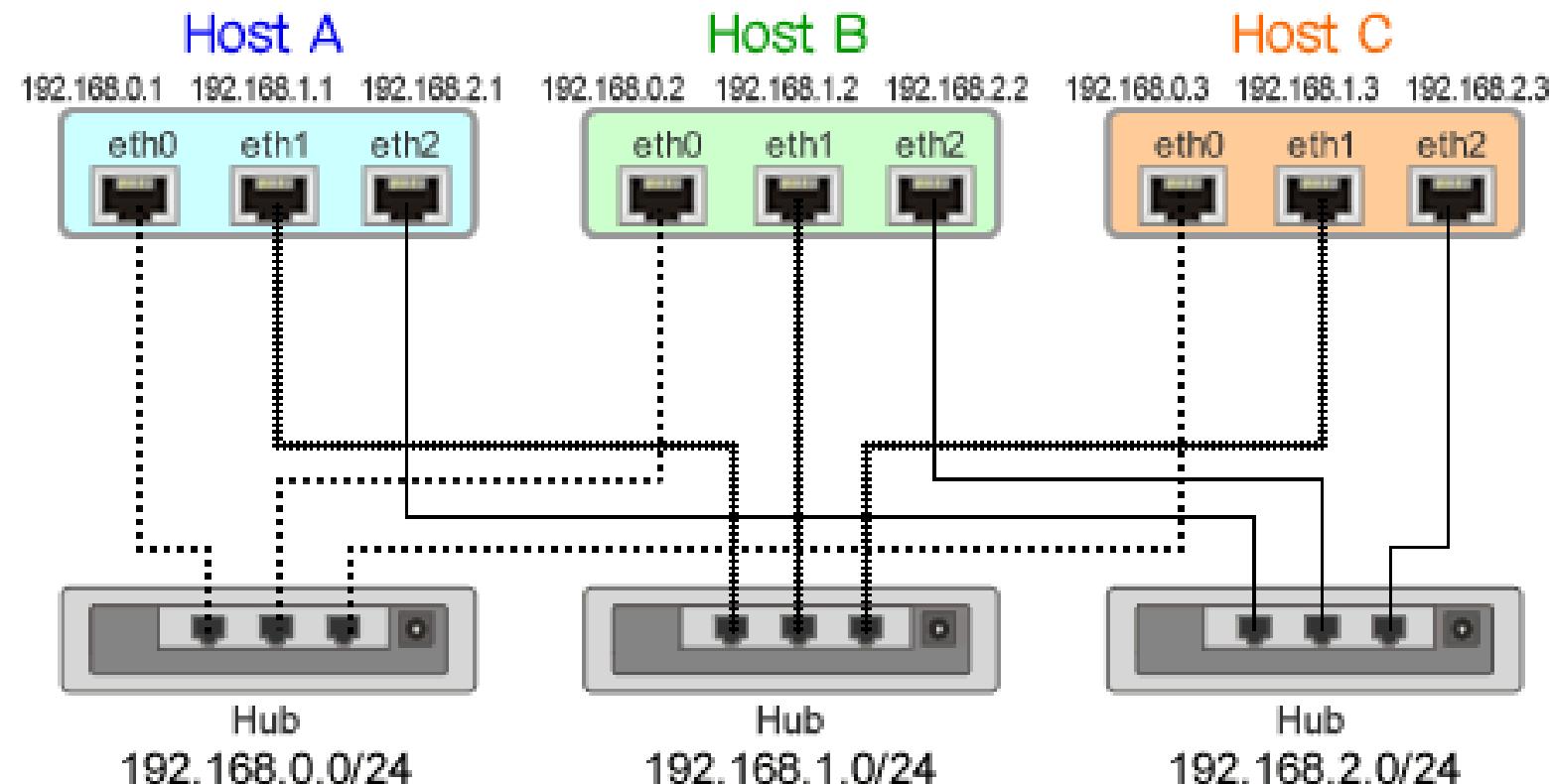


清华大学  
Tsinghua University

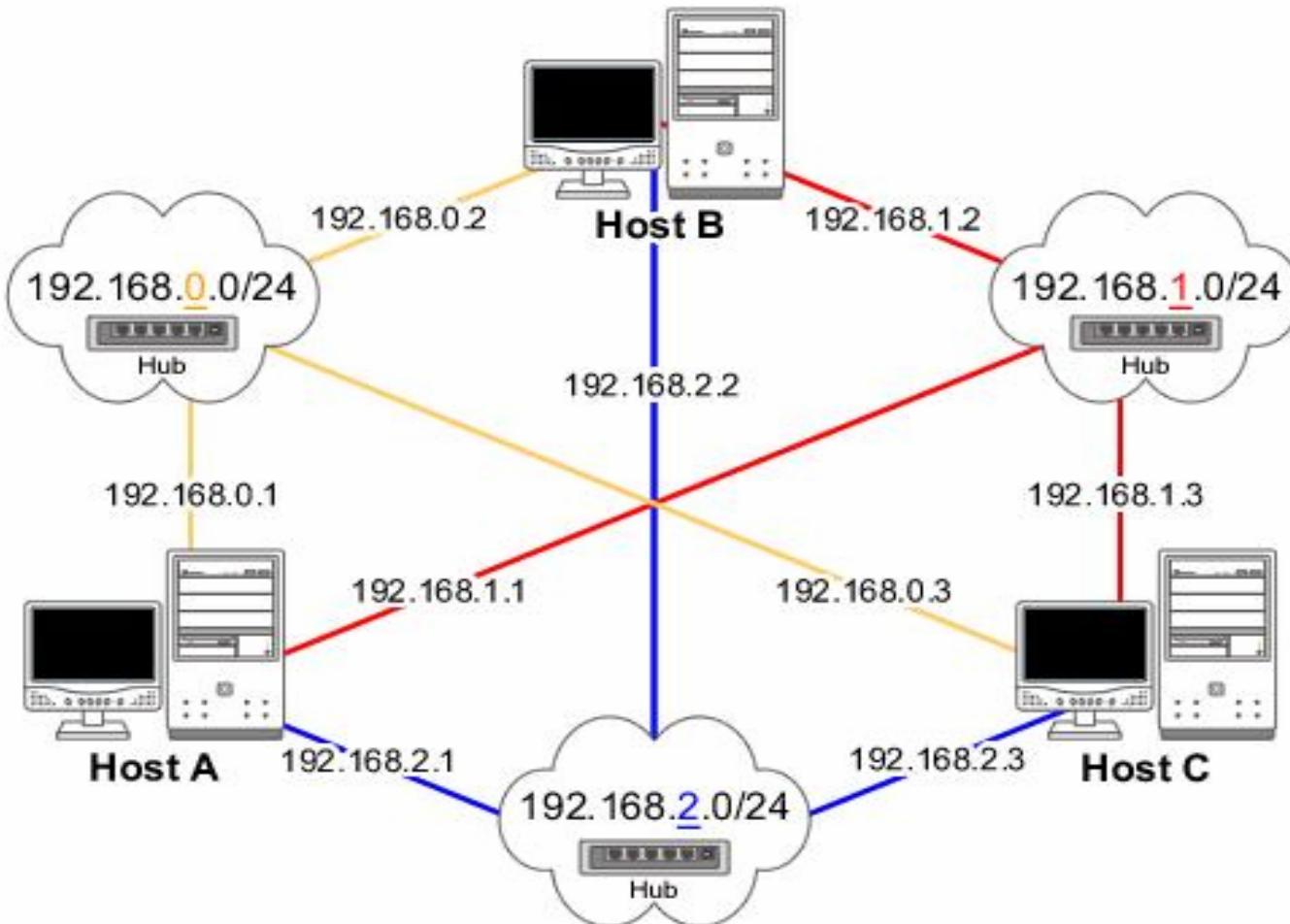
- 传输数据是为了满足应用需求。因此所有Layer 7 以下大都是为了 Layer 7 而设计
  - 例如E-mail (SMTP—Simple Mail Transport Protocol, POP 3 ) FTP (File Transfer Protocol)、VoIP (Voice over IP)等。
- 为了解 Application 如何编写，网络程序设计是网络研发者所必备的知识。
- 参见Richard Stevens的《 Unix Networking Programming》
- 参见Richard Stevens的《Advanced Programming in the UNIX Environment》



# Simple Net教学环境实体架构



# SimpleNet教学环境示意图





*Send me your comments to*  
[zhenchen@csnet1.cs.tsinghua.edu.cn](mailto:zhenchen@csnet1.cs.tsinghua.edu.cn)



## 第二章 计算机安全威胁和网络安全

Chapter 2 Computer Security Threats and Network Security

李军, 研究员

陈震, 助理研究员

[{zhenchen, junl}@tsinghua.edu.cn](mailto:{zhenchen, junl}@tsinghua.edu.cn)

网络安全实验室, 信息技术研究院,

清华大学

<http://security.riit.tsinghua.edu.cn>

# 计算机安全威胁



## ● 病毒

- 特洛伊木马 Trojans
- 蠕虫病毒 Worms

## ● 流氓软件（又叫间谍软件）

- Spyware
- Adware
- Malware

## ● 网络攻击

- 拒绝服务攻击 (DoS)
- 分布式拒绝服务攻击 (DoS Attack)

# 2006十大流行病毒



- Trojan.DL.Agent(木马代理)
- Phel(下载助手)
- Gpigon(灰鸽)
- Lmir/Lemir(传奇木马)
- QQHelper(QQ助手)
- Delf
- SDBot
- StarPage
- Lovgate(爱之门)
- Qqpass(QQ木马)



# Trojan木马

## ● 定义

- 神话中，表面上特洛伊木马是“礼物”，实际上却是藏匿袭击特洛伊城的希腊士兵。
- 现在，特洛伊木马是一些表面有用的软件程序，实际目的是危害并破坏计算机安全。

## ● 实质：一个client/server程序

## ● 目的：远程访问控制受害机器，获取控制权和密码，进行其它操作，如冰河木马

## ● 特点：不是合法的网络服务程序，木马必须隐藏自己

## ● 传播：一般为非主动传播，属种植型。最近的特洛伊木马都以电子邮件的形式传播。

# 木马隐身- Windows驱动程序/动态链接库木马



清华大学  
Tsinghua University

- 修改虚拟设备驱动程序（vxd）或动态连接库（DLL）
- 摆脱了原有的木马模式监听端口，采用替代系统功能的方法（改写vxd或DLL文件）
- 木马会将修改后的DLL替换系统已知的DLL，并对所有的函数调用进行过滤。
  - 对于常用的调用，使用函数转发器直接转发给被替换的系统DLL
  - 对于一些事先约定好的特种情况，DLL会执行一些相应的操作
- 实际上这样的木马多只是使用DLL进行监听，一旦发现控制端的连接请求就激活自身，绑在一个进程上进行正常的木马操作。
- 优势：
  - 没有增加新的文件，不需要打开新的端口，没有新的进程
  - 使用常规的方法监测不到它，在正常运行时木马几乎没有任何症状
  - 一旦木马的控制端向被控制端发出特定的信息后，隐藏的程序就立即开始运作
- 杀毒软件如IP armor采用了监视动态链接库的技术，可以监视所有调用Winsock的程序，并可以动态杀除进程，是对付驱动程序/动态链接库木马的一种工具



# 网银木马

## ● 危害

- 工商银行网银phishing
- 某证券公司客户端被种木马事件

## ● 盗取用户银行账号的木马

- Trojan/PSW.Soufan (证券盗手)
- Trojan/PSW.HidWebMon(网银盗手)
- TrojanSpy.Banker.s(t)(yy)

## ● 常用技术

- 虚假网站(phishing)和服务器攻击
- 键盘记录
- 嵌入浏览器执行
- 屏幕录像
- 窃取数字证书文件
- 伪装窗口

# 冰河木马



- 控制扫描端口(7626), 被控制端就成为一台服务器, 控制端则是一台客户机, C:\windows\G\_server.exe, G\_client是客户端应用程序
- 冰河会在注册表的 HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run和RUNSERVICE键值中加上了 \kernel32.exe,其次冰河的服务端会在:\windows\sysexplr.exe 文件
- 关联EXE文件或其它文件(关联是指无论运行什么exe文件, 冰河就开始加载)

# QAZ木马



- 具有蠕虫特性的木马，
  - 自动搜寻局域网中是否有完全共享的机器
  - 在共享机器上查找Notepad.exe
  - 替换Notepad.exe为Note.com
  - 将自身复制为Notepad.exe，并在注册表启动项中添加StartIE=notepad.exe \*\*\*

# DDos木马Tray.exe



- 分布式拒绝服务攻击
- 查注册表
  - HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run发现tray.exe
  - HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices发现一个键值为tray.exe

# Windows 检查木马小常识



- 检查启动组、注册表、Win.ini、System.ini、
  - 启动组对应的文件夹为：C:\windows\start menu\programs\startup，在注册表中的位置：HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellFolders Startup="C:\windows\start menu\programs\startup"。
- 通过regedit命令打开注册表编辑器
  - 检查HKEY-LOCAL-MACHINE-SOFTWARE-Microsoft-Windows-CurrentVersion-Run
  - 检查HKEY-LOCAL-MACHINE-CURRENTUSER-Software-Microsoft-Windows-CurrentVersion-Run
  - 检查HKEY-LOCAL-MACHINE-USERS-Software-Microsoft-Windows-CurrentVersion-Run
- 检查下面所列文件
  - C:\windows\winstart.bat、C:\windows\wininit.ini、Autoexec.bat
- 关闭常用网络连接，观察是否还存在网络数据传输

# 反木马软件



- 基本技术
  - 注册表监视
  - 远程线程监视
  - 反rootkit
  - 系统监控

# 蠕虫病毒(Worm)



- 蠕虫病毒可自动完成复制过程，控制计算机中传输文件或信息的功能
- 一旦计算机感染蠕虫病毒，蠕虫即可独自传播，并大量复制
- 特点：依靠网络主动传输，攻击受害机器，传播迅速，消耗极大的网络和计算资源

# CodeRed红码蠕虫



- 病毒一代名称: Code Red,
  - 别名: I-Worm.Bady, CodeRed, Bady
- 病毒二代名称: CodeRed II
  - 别名: CodeRed.C, CodeRedII, CodeRed v3, Code Red gen 3  
别名: IIS-Worm.CodeRed.c, Trojan.Win32.VirtualRoot, CodeRed B
  - 新蠕虫变种在WINDOWS系统中更改系统设置, 修改WINDOWS文件并放置**Trojan木马程序**, 最终导致受感染系统丧失安全策略
- 校园网很普遍



# CodeRed B传播原理

- CodeRed B利用WINDOWS2000, WINDOWS XP 的IIS系统漏洞，入侵系统
- 入侵系统后，蠕虫将CMD. EXE拷贝到C:盘及D:盘以下目录中:\inetpub\scripts\program files\common files\system\msadc
- 蠕虫扫描网络，寻找其它可被攻击的系统
- 蠕虫程序在C:盘和D:盘的根目录下生成一个大小为8,192字节的EXPLORER. EXE木马程序，然后重启系统，执行木马程序
- 该木马程序执行一下步骤
  - 运行WINDOWS的EXPLORER程序
  - 修改以下注册表键值使WINDOWS系统丧失对系统文件的保护能力：  
HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\SFCDisable=0xFFFFFFF9D. 这一设置的缺省值为 0
  - 修改注册表以下键值创建两个虚拟IIS目录C和D，分别映射到系统的C:盘和D:  
盘. HKLM\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots  
这些虚拟目录被赋予读写及可执行权限（这样木马程序通过IIS向所有黑客提供了对被感染服务器C:盘和D:盘的完全控制能力）
  - 以上木马程序完成了它的感染周期，并会每各10分钟重复进行以上提到的注册表项修改

# RedCode红码蠕虫清除



## ● CodeRed B 红码清除

- 运行相应补丁程序
- 删除c:\explorer.exe和d:\explorer.exe(这两个为隐藏, 只读文件)
- 改  
HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\SFCDisable键值为0
- 把  
HKLM\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots对于c:和d:的完全控制键值删除。
- 在  
\inetpub\scripts 和 \program files\common files\system\msadc中删掉root.exe
- 重启计算机

# CodeRed.F感染过程(1)



- 利用Windows XP的IIS服务漏洞，入侵系统，运行病毒程序
- 病毒程序调用初始化例程，确定 IIS Server服务的进程地址空间Kernel32. d11 的基本地址
- 查找 GetProcAddress 的地址，调用 GetProcAddress 来获取对一组 API 地址的访问权限，例如：LoadLibraryA, CreateThread, …, GetSystemTime
- 加载 WS2\_32. d11 以访问 socket、closesocket 和 WSAGetLastError等函数
- 从 User32. d11 获得 ExitWindowsEx，用于重新启动系统
- 主线程检查两个不同的标记符：
  1. 第一个标记符是“29A”，它控制 Trojan. VirtualRoot 的安装。
  2. 另一个标记符是名为“CodeRedII”的信号。如果存在该信号，此蠕虫会进入无限睡眠状态
- 主线程将检查默认语言，默认语言是中文（无论繁体还是简体），将创建 600 个新线程，否则，只创建 300 个
- 这些线程将产生一些随机的 IP 地址，用于搜索要感染的新 Web 服务器
- 这些线程运行时，主线程会将木马Cmd. exe文件从 Windows NT\System文件夹复制到下列文件夹（如果存在）：  
C:\Inetpub\Scripts\Root. exe  
D:\Inetpub\Scripts\Root. exe  
C:\Progra~1\Common~1\System\MSADC\Root. exe  
D:\Progra~1\Common~1\System\MSADC\Root. exe



# CodeRed.F感染过程(2)

- 如果该蠕虫放置的特洛伊木马程序已更改了注册表键：
  - HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\W3SVC\Parameters\VirtualRoots (通过添加几个新键并将用户组设置为 217)
  - 黑客发送HTTP GET 请求以在受感染的 Web 服务器上运行 scripts/root.exe，从而完全控制该 Web 服务器
- 否则，蠕虫将命令解释程序 (cmd.exe) 复制到IIS Web 服务器的默认可执行目录下，从而实现远程控制
- 在根驱动器C:\和D:\下放置Explorer.exe，属性为隐藏、系统和只读 (Norton AntiVirus标识为Trojan.VirtualRoot)。感染24或48小时后重新启动计算机
- 当计算机重新启动时，在系统试图执行Explorer.exe 时执行 Trojan.VirtualRoot (“Explore.exe”)。该特洛伊木马重新设置注册表键以确保其已被更改

# CodeRed.F清除-删除病毒文件



- 终止与所放置的特洛伊木马（NAV 标识为 Trojan.VirtualRoot）相关的当前进程：
  - 按 Ctrl+Alt+Delete，进入“任务管理器”。
  - 查找两个名为Explorer.exe 的进程，一个是正常的进程，另一个就是特洛伊木马。
  - 在两个 Explorer.exe 进程中，结束只有一个线程的进程。
- 删除在受感染系统上创建的 Explorer.exe 文件
  - 进入cmd, 键入下列命令：  
`cd c:\ ; attrib -h -s -r explorer.exe ; del explorer.exe`
- 删除以下四个文件(%Windir%\root.exe副本)：  
C:\Inetpub\Scripts\Root.exe  
D:\Inetpub\Scripts\Root.exe  
C:\Progra~1\Common~1\System\MSADC\Root.exe  
D:\Progra~1\Common~1\System\MSADC\Root.exe
- 打开“计算机管理器”(Compmgt.msc)  
删除\计算机管理(本地)\服务和应用程序\默认 Web 站点下所列出的其他所有驱动器重复该操作

# CodeRed.F 清除-恢复注册表



- 运行 regedit（将打开注册表编辑器）
- 修改以下键：  
HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots的键值，删除 CodeRed II 所创建的/C和/D。  
修改/MSADC为数字“201”和修改/Scripts为数字“201” 3.
- 恢复  
HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\WinLogon的键值 /SFCDisable为 0
- 重新启动系统以确保 CodeRed II 已被正确杀除

# 杀手13（Worm.KillOnce）



清华大学  
Tsinghua University

- 病毒运行后将自身写入系统目录及回收站，通过修改注册表进行自启动，
- 建立多个线程，在局域网内传播
- 关闭已知的反病毒软件，用NET命令打开局域网内工作站的后门
- 偷查感染特征：
  - 系统目录以及回收站中名为*Killonce.exe*文件
  - 运行时在注册表中  
*HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersionRun*项中加入键值为  
*Killonce* (*C:\WINNTSYSTEM32\KillOnce.exe*)
  - 无法运行*regedit.exe*或*msconfig.exe*时，出现“非法用户”和“无权执行”错误
  - 搜索DOC文件，在搜到的目录下生成*RICHED20.DLL*文件；搜索HTM文件，在搜到的目录下生成*SHDOCVW.DLL*文件，使病毒生成的文件可被WORD、IE等外部程序加载
  - 在管理组中建立一个GUEST用户，并将其访问权限提高到管理员，在系统中留下后门
  - 12月13日执行在*autoexec.bat*中加入*deltree /y c:\*.\**命令

# 流氓软件



- 又称恶意软件(Malware)和间谍软件(Spyware)
- 未经用户同意自动安装的各种软件,但卸载不易
- 不请自来, 送走很难
  - 3721, 中搜, Yahoo助手等
- 通常是访问Web页面时, 利用JSP, 或Java虚拟机漏洞而安装

# 拒绝服务攻击DoS



- DoS是拒绝服务(Denial of Service)的简称，造成DoS的攻击行为被称为DoS攻击
- 目的
  - 使计算机或网络无法提供正常的服务
- 常见DoS攻击有
  - 网络带宽攻击
    - ◆ 带宽攻击指以极大的通信量冲击网络，使得所有可用网络资源都被消耗殆尽，最后导致合法用户无法访问网络资源
  - 连通性攻击
    - ◆ 连通性攻击指用大量连接请求冲击服务器，使得所有可用的操作系统资源都被消耗殆尽，最终使服务器无法再处理合法用户请求

# 分布式拒绝服务攻击



- 分布式拒绝服务(DDoS, Distributed Denial of Service)攻击指借助于客户/服务器技术，将多个计算机联合起来作为攻击平台，对一个或多个目标发动DoS攻击，从而成倍地提高拒绝服务攻击的威力
- 攻击者使用一个偷窃帐号将DDoS主控程序安装在一个计算机上
- 攻击者秘密安装代理程序到Internet的大量计算机上，代理程序能都对目标机器发动DoS攻击
- 利用客户/服务器技术，主控程序几乎同时激活运行成百上千的主机上的代理程序，控制代理程序对一个过多个目标发动DoS攻击

# 其他常见攻击

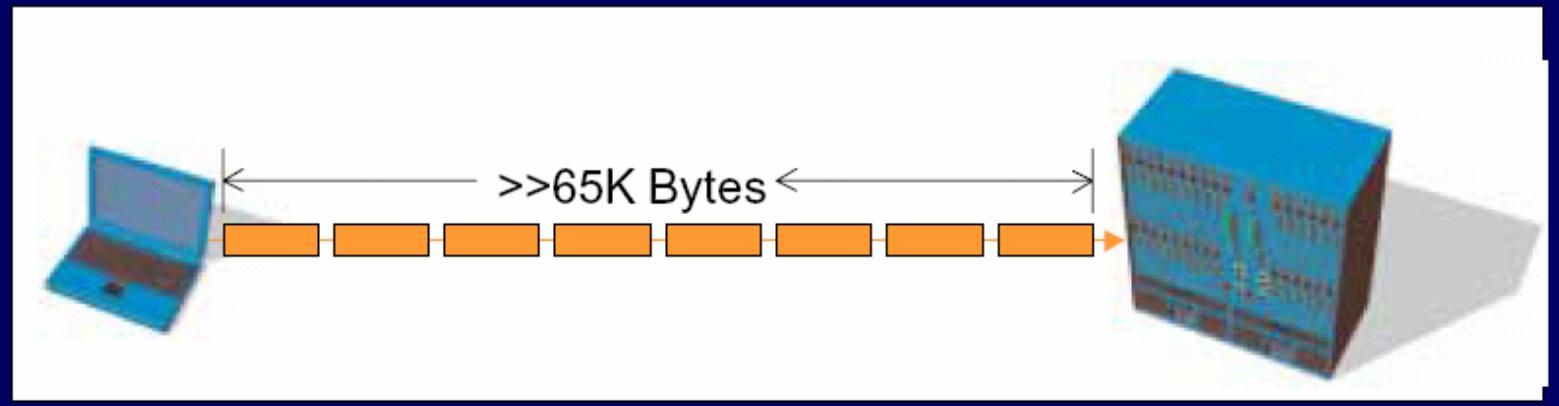


- Ping of Death
- Tear Drop
- Smurf 攻击
- Ping泛洪(Ping Flood)
- SYN攻击
- MAC攻击(交换机)
- ARP攻击
- VLAN跳攻击
- 虚假DHCP服务器攻击

# Ping of Death



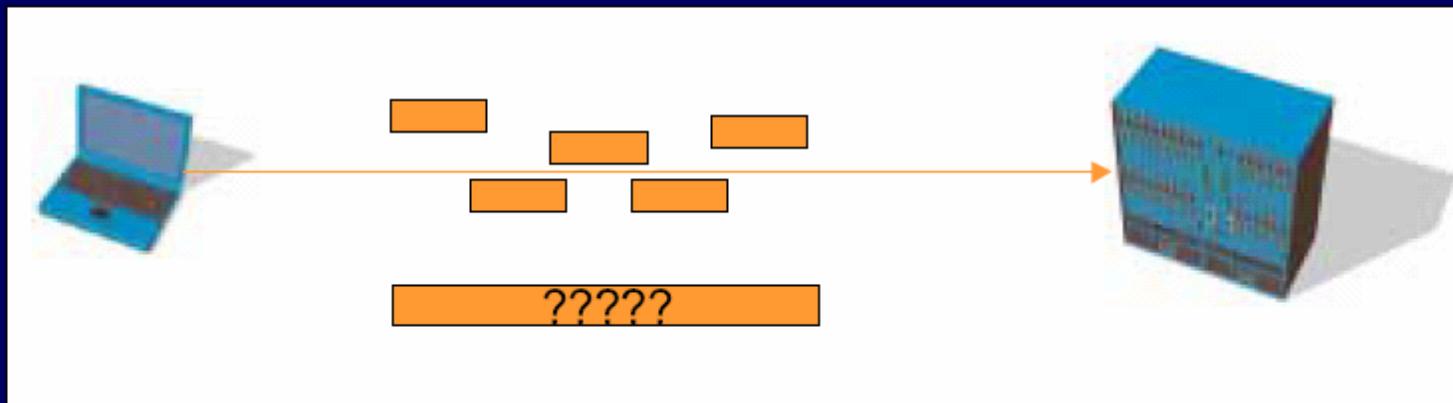
- Send multiple fragment packets to cause the system to believe that it is receiving a VERY large ping packet
- Operating systems only had buffers to handle reasonable size packets
- Typically caused the attacked system to crash



# Tear Drop



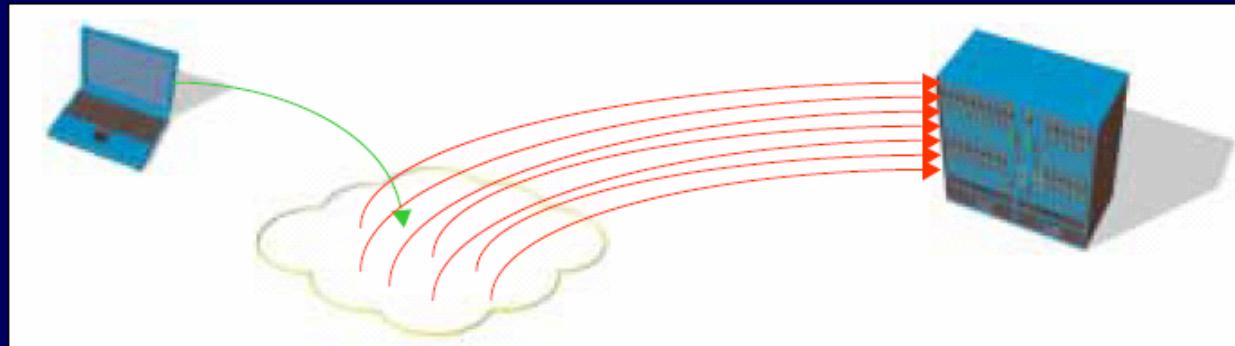
- Send multiple fragment packets that cause overlapping data segments
- Causes some systems to crash or run out of resources



# Smurf



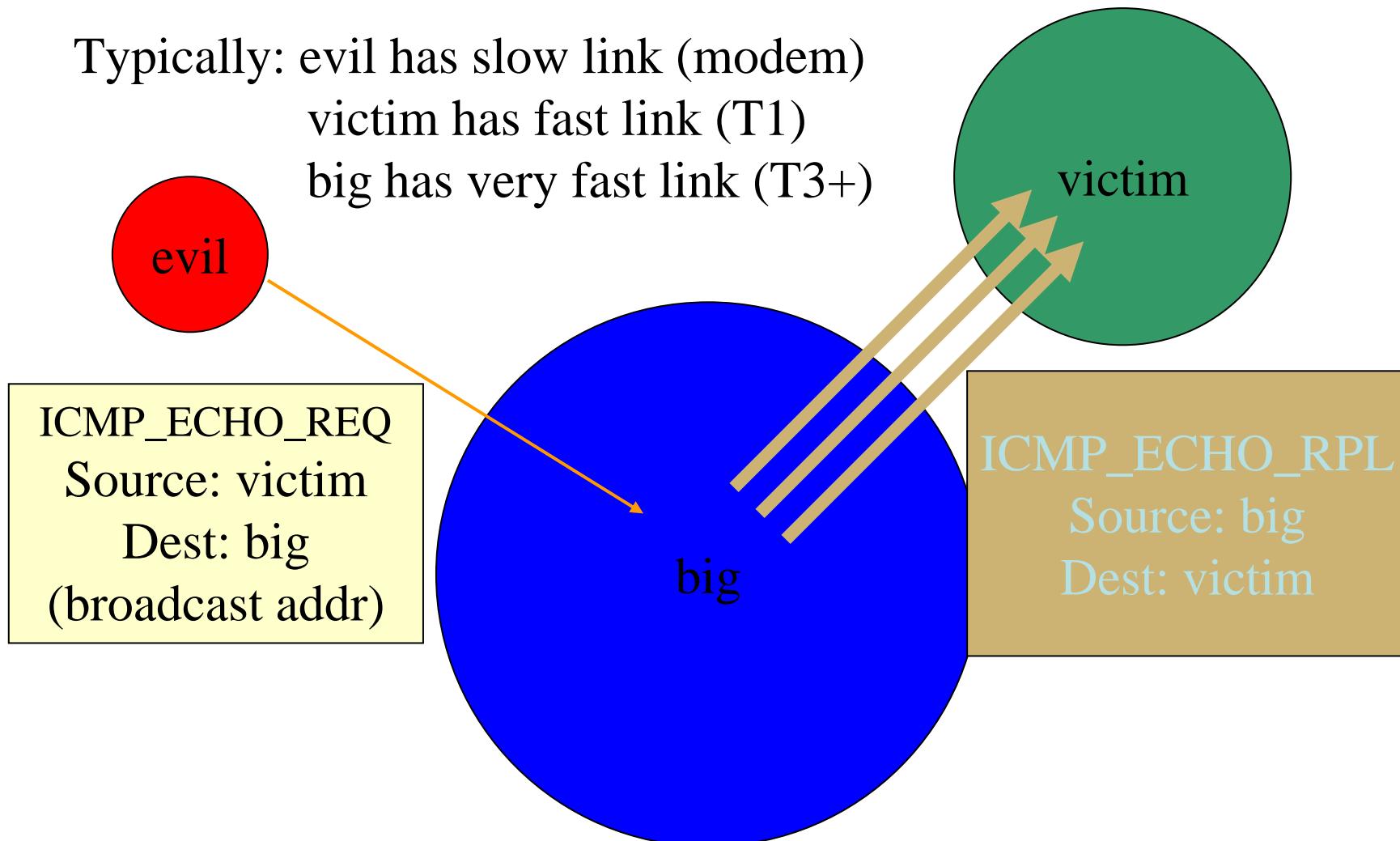
- Send a ping packet to the broadcast address on a network
- The source address of the ping is a spoofed address of the system or network being attacked
- All systems on the network send a ping response back to the spoofed address
- Creates large amounts of traffic to either flood a network or system



# Smurf Attack



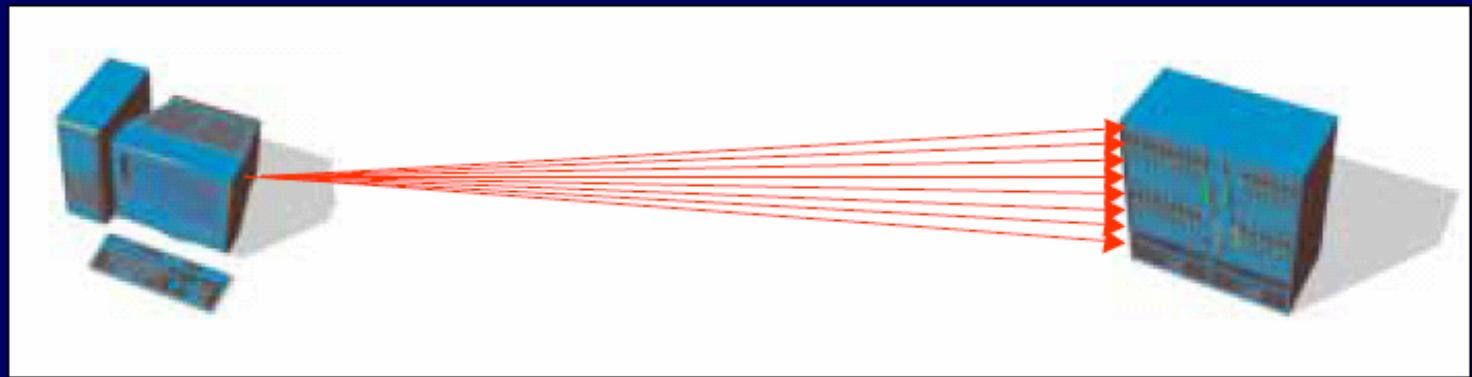
Typically: evil has slow link (modem)  
victim has fast link (T1)  
big has very fast link (T3+)



# Ping Flood



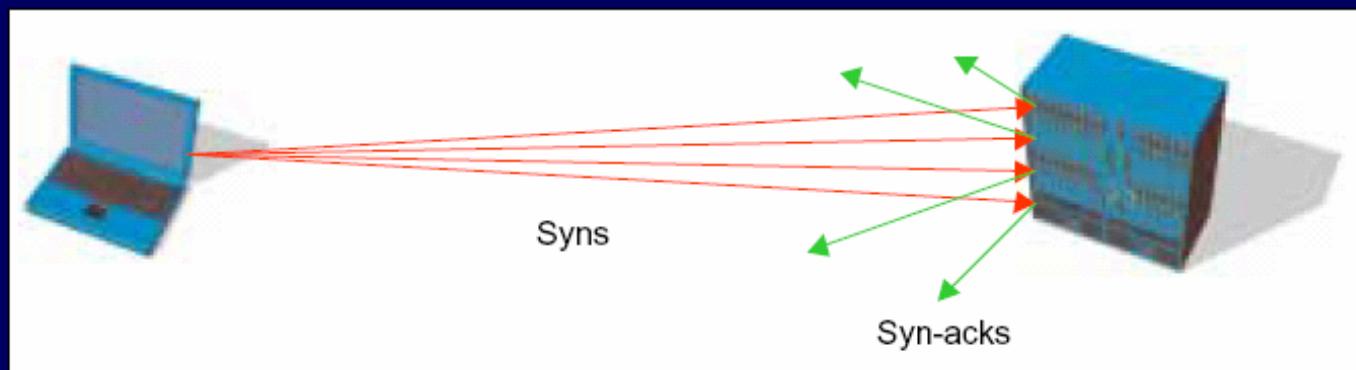
- Sends a large number of ping requests to a system
- Causes the system to slow down or fail



# SYN Attack



- Send TCP Syn packets to a system on an enabled service port (e.g. HTTP, SMTP, telnet, ftp)
- The source address is spoofed and the TCP source port is random
- Causes the system to allocate buffer space for the TCP connection and either crashes or does not allow for valid connections to be made



# Land Attack



清华大学  
Tsinghua University

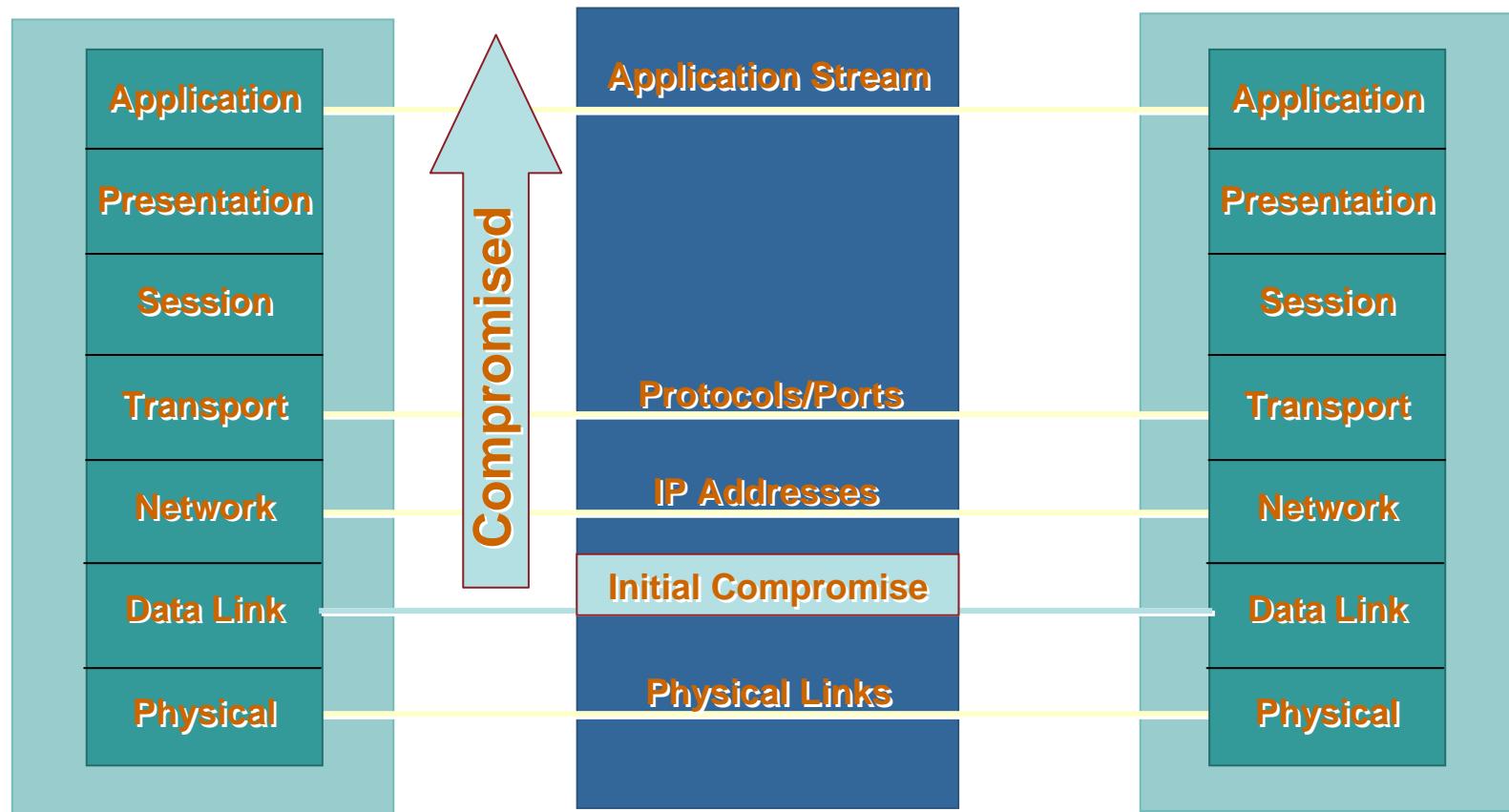
- Sends a TCP syn to the target with the source address spoofed as the target's address
- TCP source port is set to equal the destination port
- Causes the attacked host to infinite loop on trying to get correct sequence numbers
- Causes some systems to crash or slow down





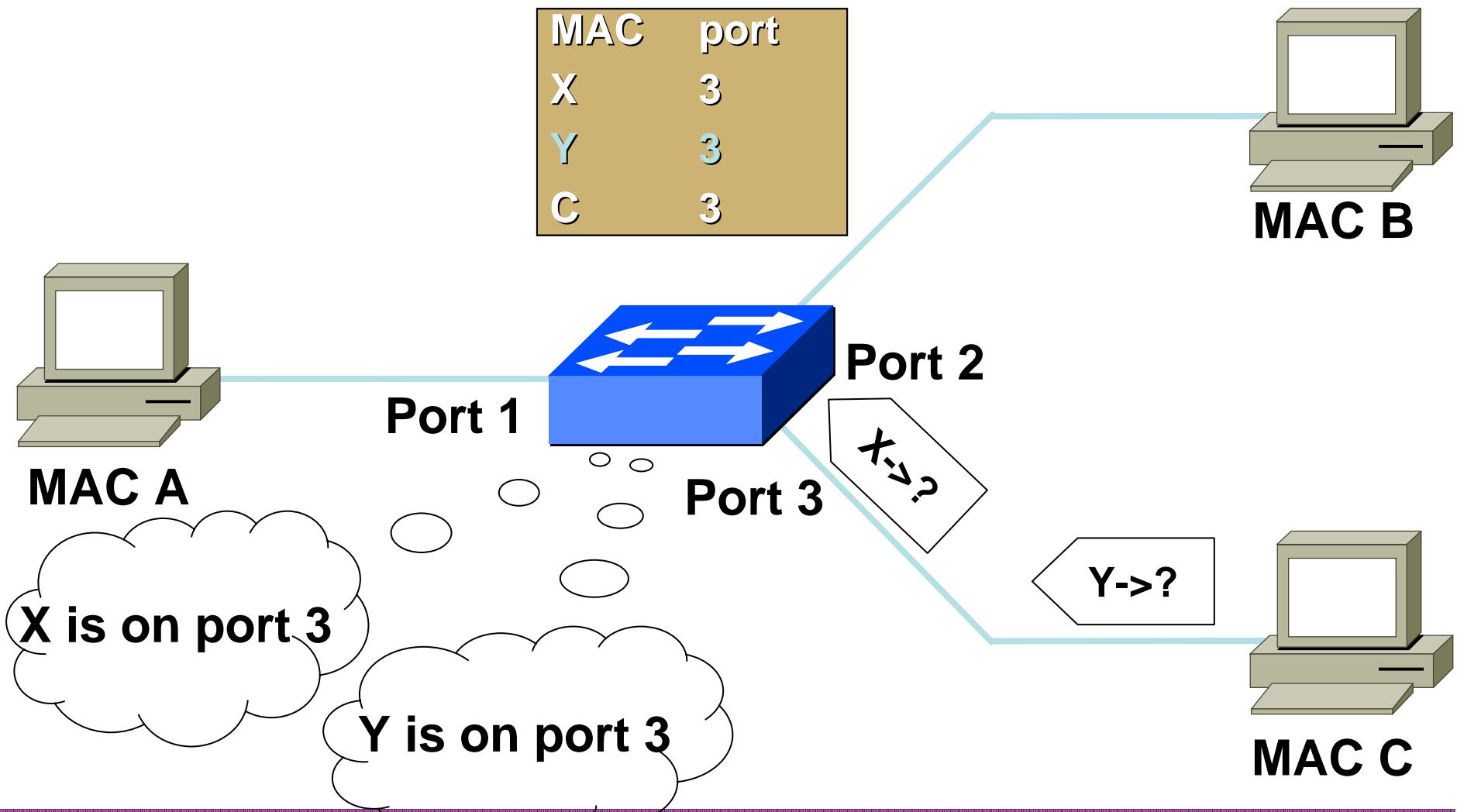
# The Domino Effect

- Unfortunately this means if one layer is hacked, communications are compromised without the other layers being aware of the problem
- Security is only as strong as your weakest link
- When it comes to networking, layer 2 can be a VERY weak link

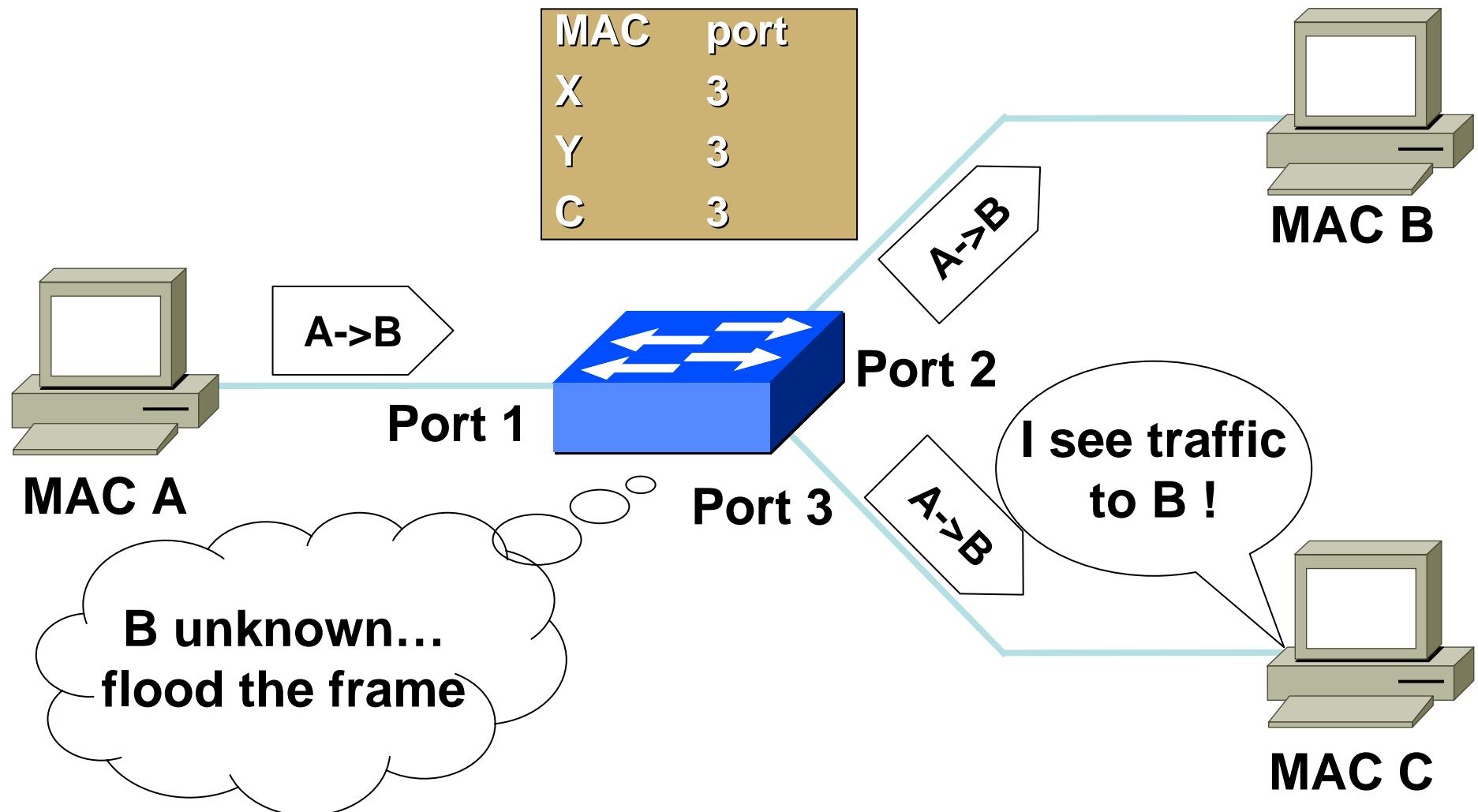


# MAC Attacks

# CAM Overflow 1/2



# CAM Overflow 2/2



# MAC Flooding Attack Mitigation



清华大学  
Tsinghua University

## ● Port Security

- Allows you to specify MAC addresses for each port, or to learn a certain number of MAC addresses per port
  - Upon detection of an invalid MAC block only the offending MAC or just shut down the port

## ● Smart CAM table

- Never overwrite existing entries
- Only time-out inactive entries
- Active hosts will never be overwritten

## ● Speak first

- Deviation from learning bridge: never flood
- Requires a host to send traffic first before receiving

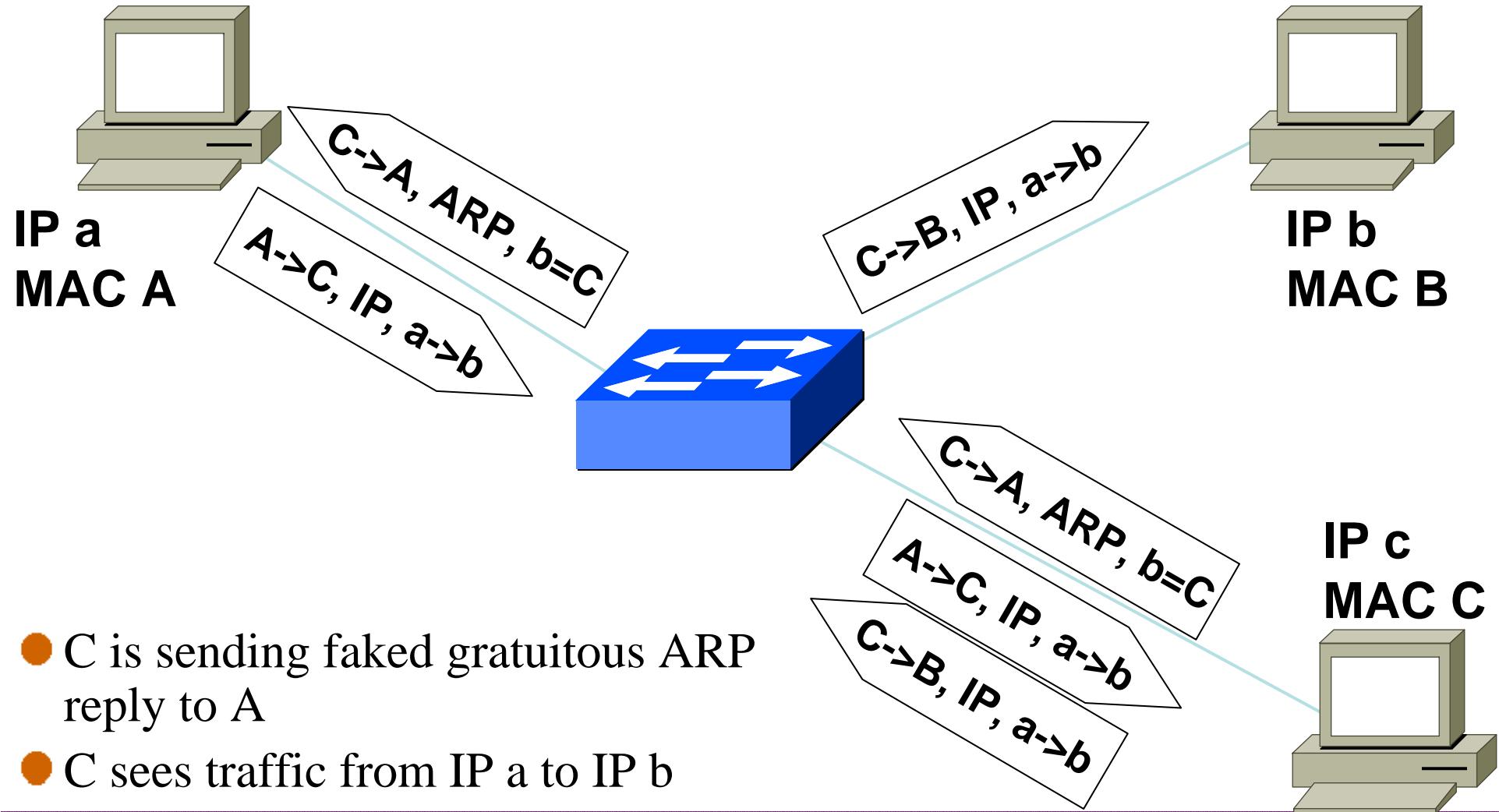
# ARP 攻击

# ARP Attacks

# ARP Spoofing



清华大学  
Tsinghua University



# Mitigating ARP Spoofing

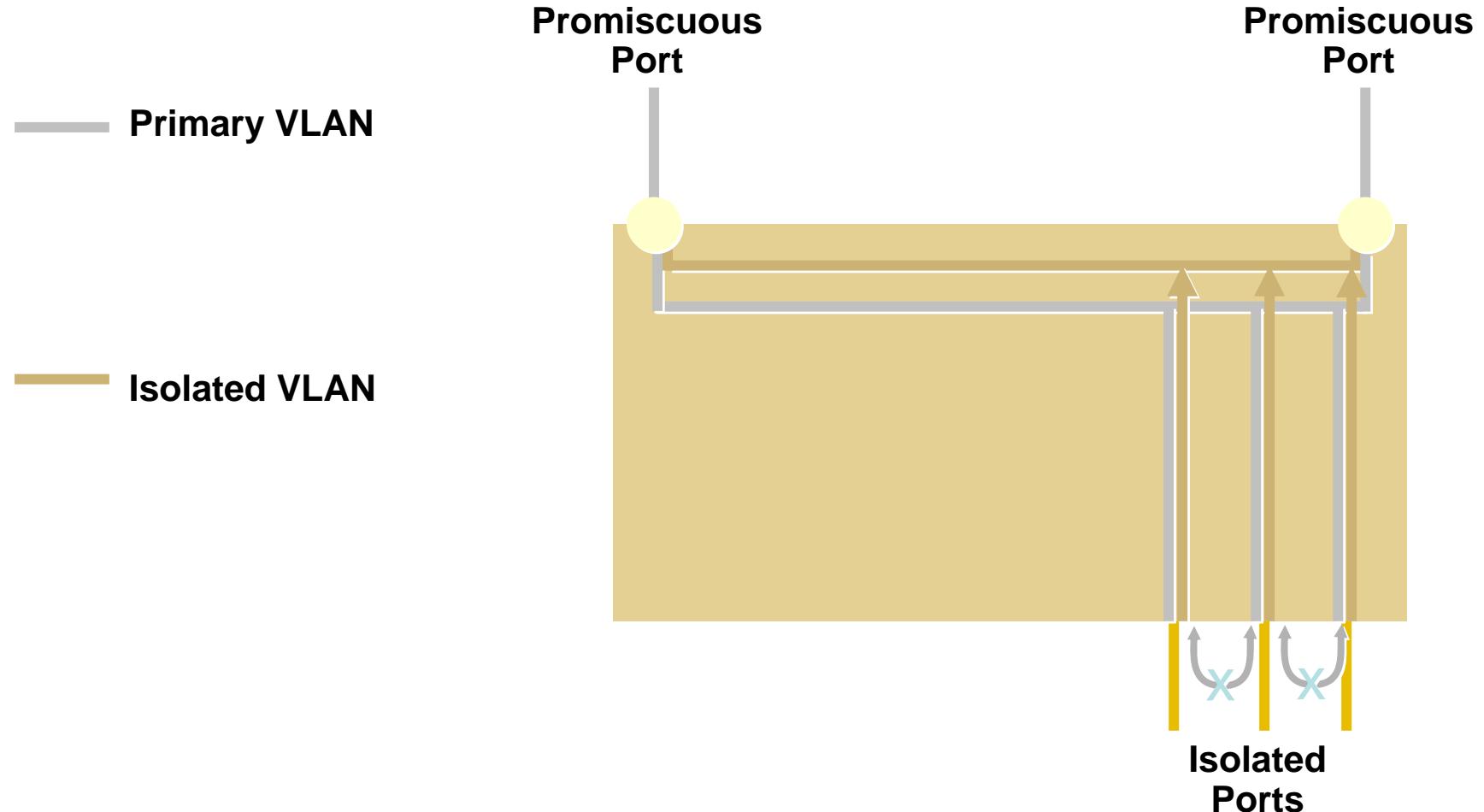


- ARP spoofing works only within one VLAN
- static ARP table on critical stations (but dynamic ARP override static ARP on most hosts!)
- ARP ACL: checking ARP packets within a VLAN
  - Either by static definition
  - Or by snooping DHCP for dynamic leases
- No direct communication among a VLAN: private VLAN
  - Spoofed ARP packet cannot reach other hosts

# ARP Spoof Mitigation: Private VLANs



清华大学  
Tsinghua University

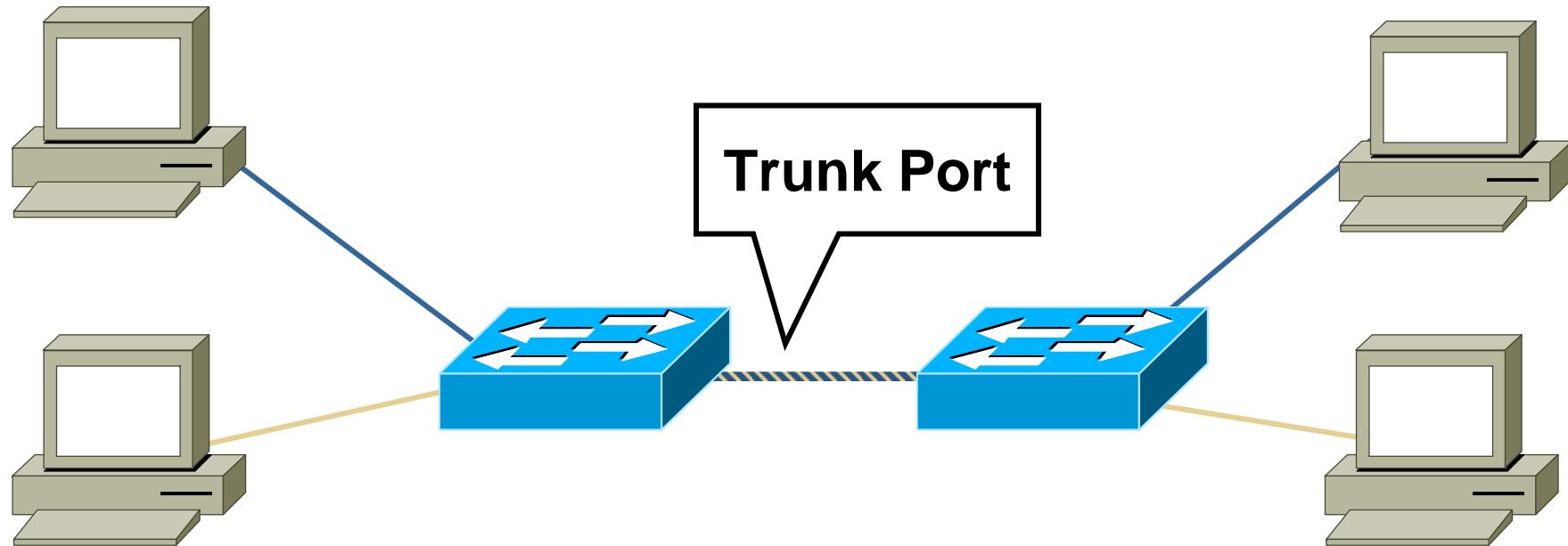


# VLAN跳攻击

# VLAN “Hopping”

# Attacks

# Trunk Port Refresher

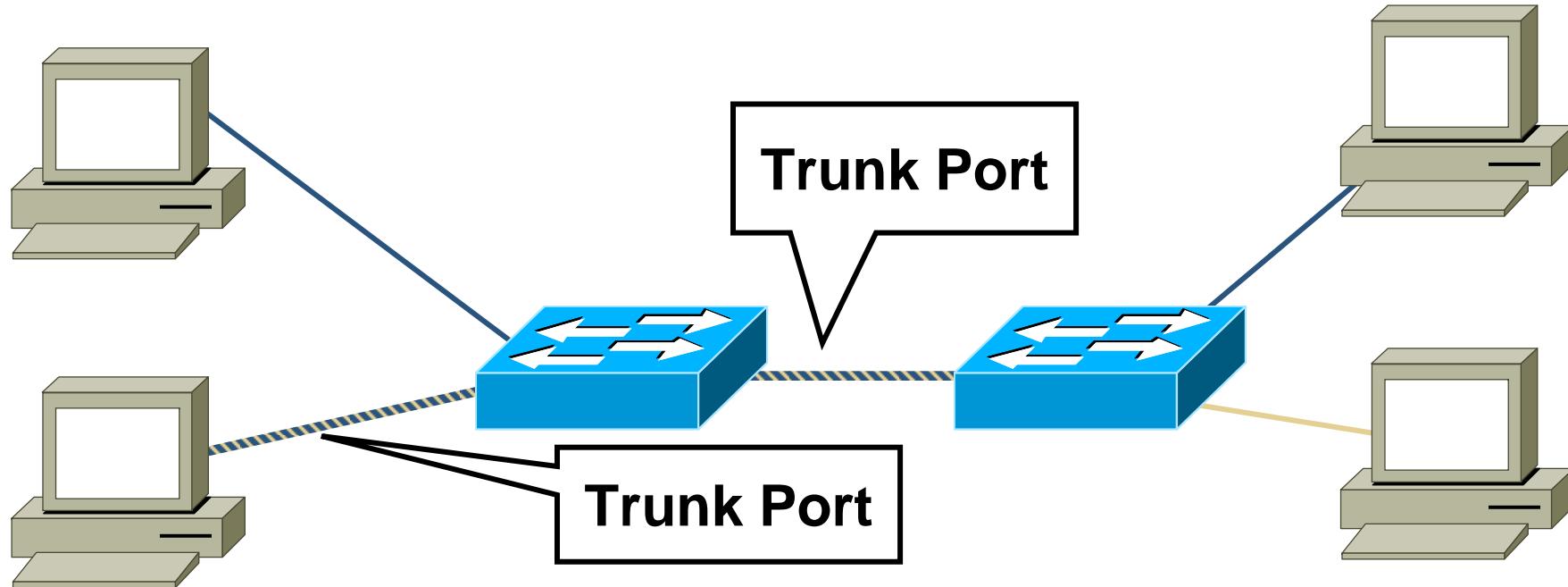


- Trunk ports have access to all VLANs by default
- Used to route traffic for multiple VLANs across the same physical link (generally used between switches)

# Basic VLAN Hopping Attack



清华大学  
Tsinghua University



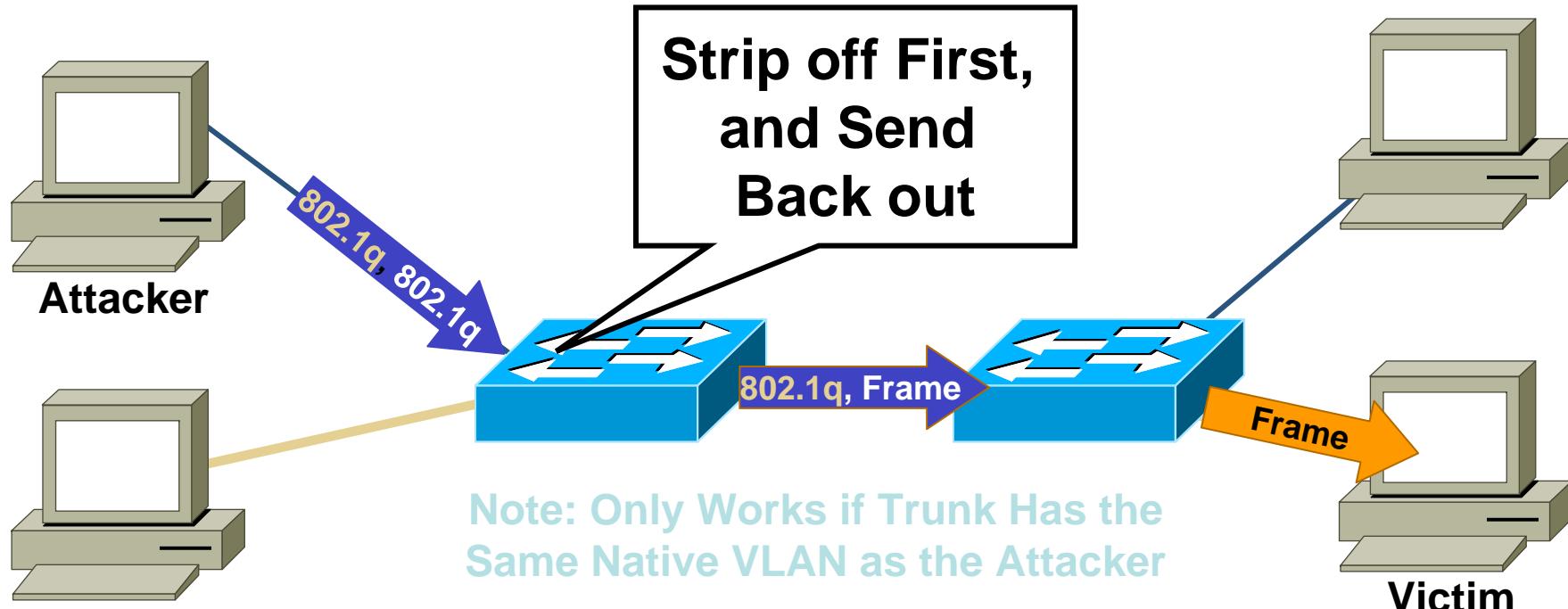
- A station can spoof as a switch with 802.1Q signaling
- The station is then member of all VLANs
- Requires a trunking favorable setting on the port (the SANS paper is three years old)

<http://www.sans.org/newlook/resources/IDFAQ/vlan.htm>

# Double Encapsulated 802.1Q VLAN Hopping Attack



清华大学  
Tsinghua University



- Send double encapsulated 802.1Q frames
- Switch performs only one level of decapsulation
- Unidirectional traffic only
- Works even if trunk ports are set to off

# Mitigation

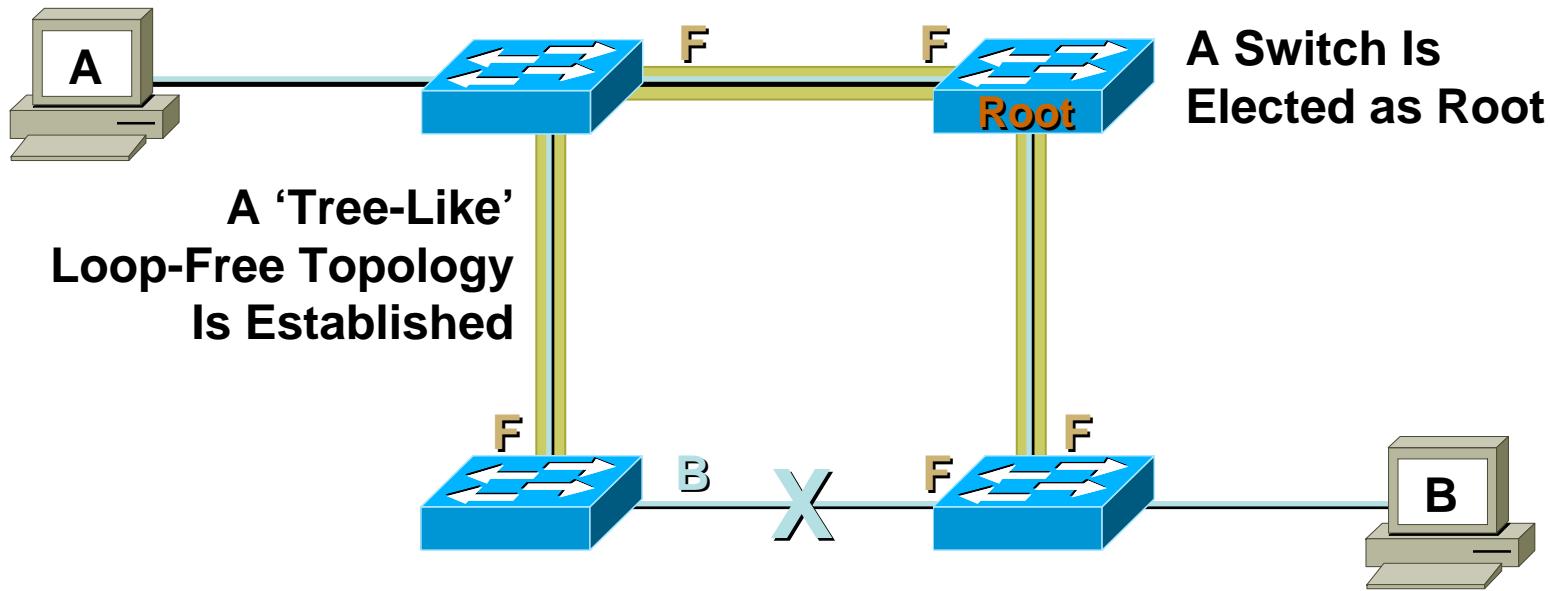


- Use recent switches
- Disable auto-trunking
- Never put host in the trunk native VLAN
- Put unused ports in an unused VLAN

# 生成树攻击

## Spanning Tree Protocol Attacks(STP)

# Spanning Tree Basics



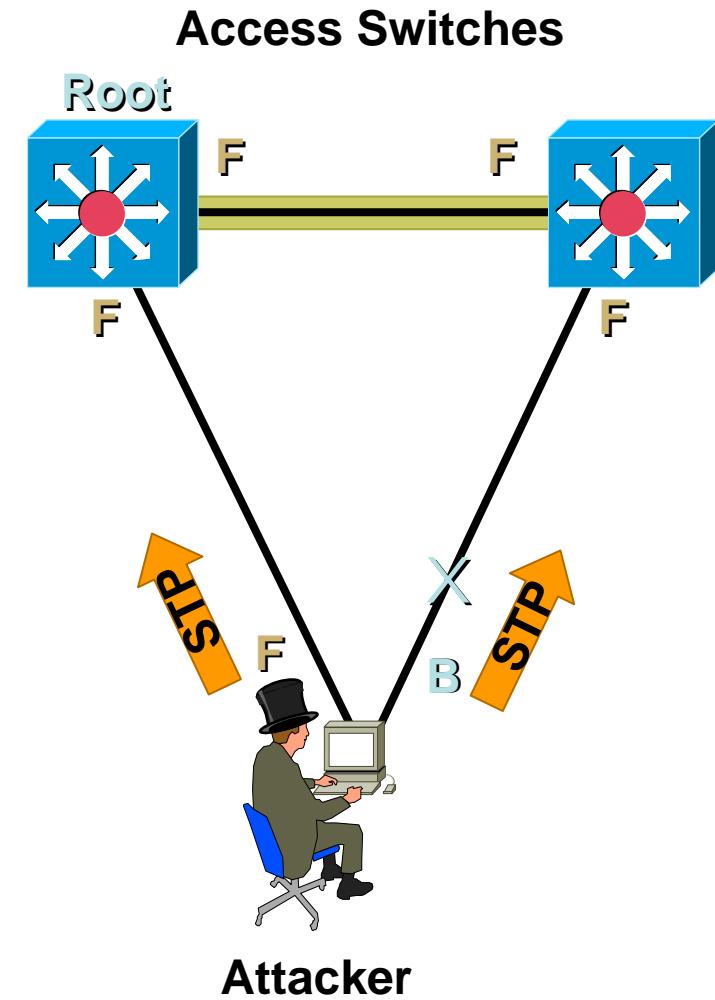
Loop-Free Connectivity

# Spanning Tree Attack

## Example 1/2



- Send **BPDU messages** from attacker to force spanning tree recalculations
  - Impact likely to be DoS
- Send BPDU messages to become root bridge

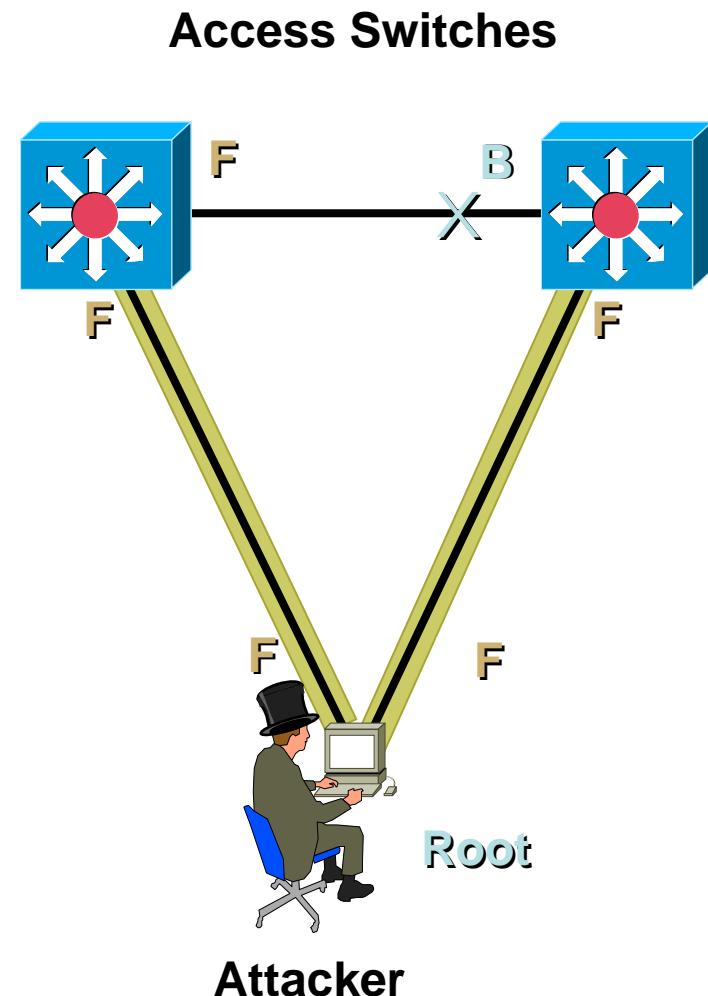


# Spanning Tree Attack

## Example 2/2



- Send BPDU messages from attacker to force spanning tree recalculations
  - Impact likely to be DoS
- Send BPDU messages to become root bridge
  - The hacker then sees frames he shouldn't
    - ◆ MITM, DoS, etc. all possible
    - ◆ Any attack is very sensitive to the original topology, trunking, PVST, etc.
    - ◆ Requires attacker to be dual homed to two different switches



# STP Attack Mitigation



- **Disable STP**

(It is not needed in loop free topologies)

- **BPDU Guard**

- Disables ports upon detection of a BPDU message on the port

- **Root Guard**

- Disables ports who would become the root bridge due to their BPDU advertisement

# 虚假DHCP服务器攻击

# DHCP Rogue Server

# Attack

# DHCP Rogue Server Attack



- Simply the installation of an unknown DHCP Server in the local subnet
- Other attack: exhaustion of DHCP pools
- RFC 3118 “Authentication for DHCP Messages” will help, but has yet to be implemented
- Mitigation:
  - Consider using multiple DHCP servers for the different security zones of your network
  - Use intra VLAN ACL to block DHCP traffic from unknown server

# 主机安全工具(Secure Tools)



- 为了确保终端主机的安全，需要安全工具的辅助
- 脆弱性分析工具（Vulnerability Analysis）
  - Retina (eEye Digital Security),
  - Microsoft Security Analyzer
  - Internet Security
- 反病毒软件（AntiVirus Tools）
  - Kaspersky, Symantec, Trend Micro, F-Prot, BitDefender, F-secure, Norton, MacAfee, Panda, NOD32
  - KingSoft/金山, KV/江民, Rising/瑞星, VRV/北信源
- 个人防火墙(Personal Firewalls)
  - Jetico, Kerio, Sygate, ZoneLabs, Outpost, ZoneAlarm,
  - 瑞星, 天网

# 网络攻击防范 Defense



- 线速访问控制列表
- 802.1x端口认证协议
- 2层安全防范

# Wire-Speed Access Control Lists



清华大学  
Tsinghua University

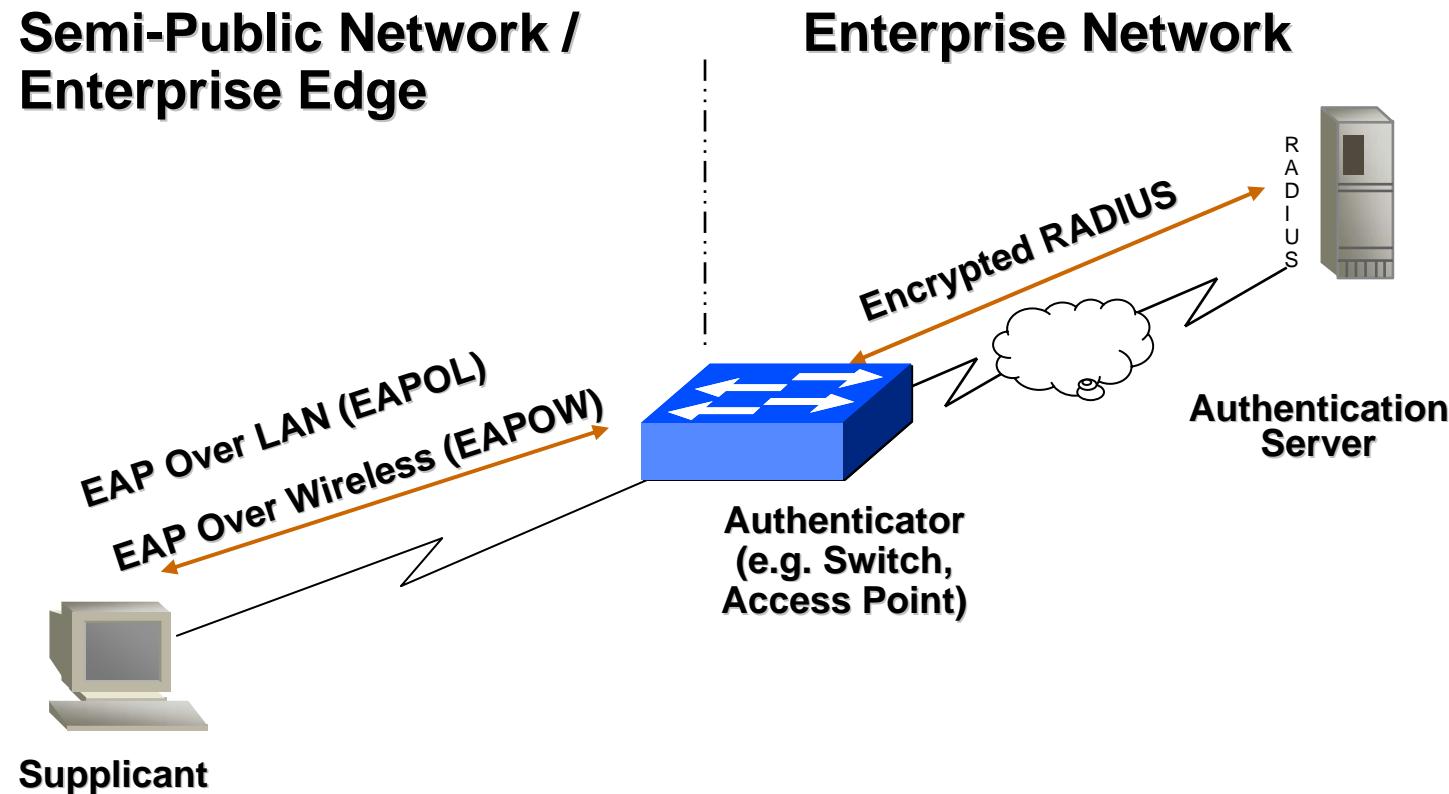
- Many current switches offer wire-speed ACLs to control traffic flows (with or without a router port)
- Allows implementation of **edge filtering** that might otherwise not be deployed due to performance concerns
- **VLAN ACLs** and **Router ACLs** are typically the two implementation methods

# 802.1x



- 802.1x is an IEEE Standard for *Port Based Network Access Control*
  - EAP based (Extensible Authentication Protocol)
  - Improved user authentication: username and password
  - Can work on plain 802.3 or 802.11

# IEEE 802.1X Terminology



# What Does it Do?



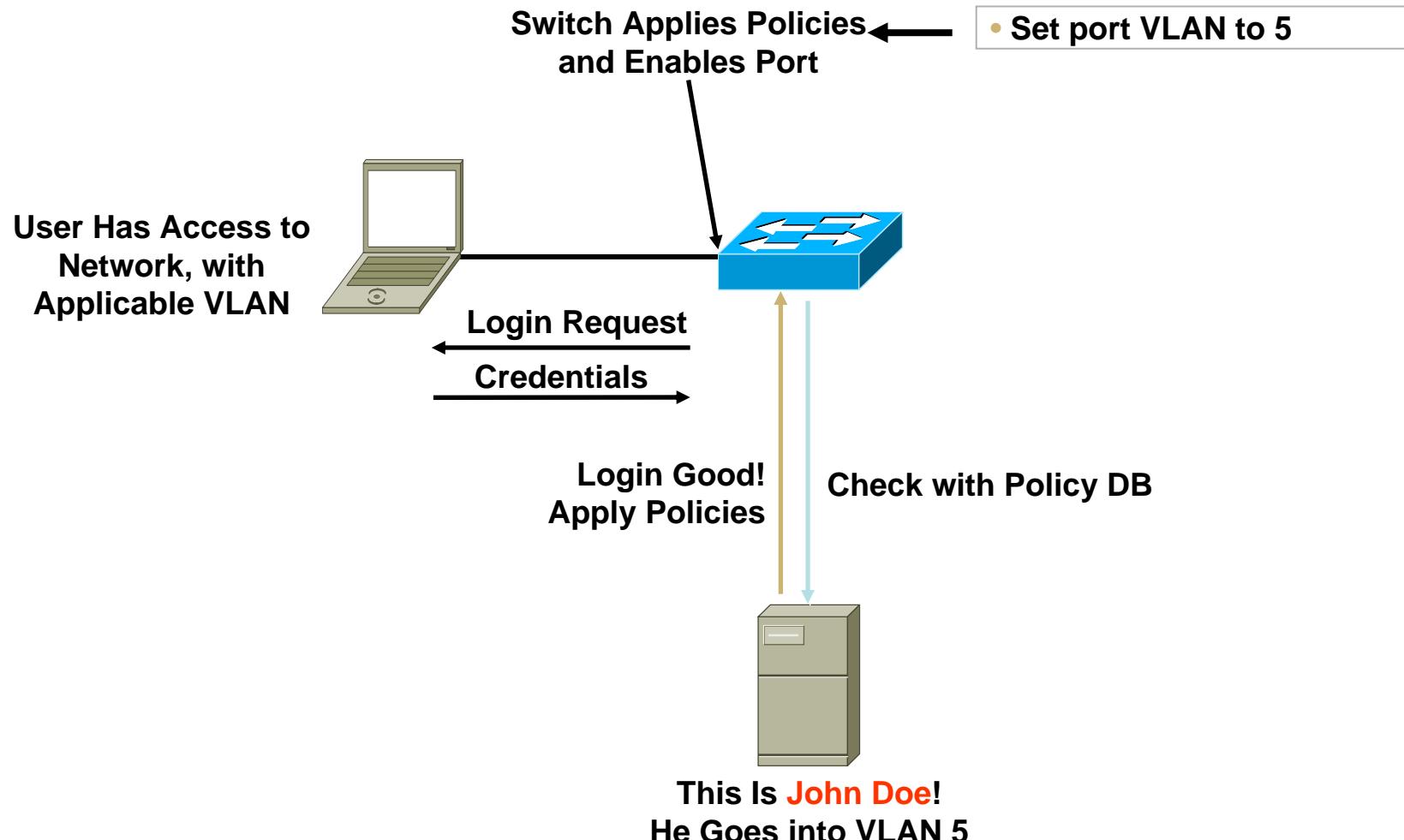
- Transport authentication information in the form of Extensible Authentication Protocol (EAP) payloads.
- The **authenticator (switch)** becomes the **middleman** for relaying EAP received in 802.1x packets to an authentication server by using RADIUS to carry the EAP information.
- Three forms of EAP are specified in the standard
  - EAP-MD5 – MD5 Hashed Username/Password
  - EAP-OTP – One-Time Passwords
  - EAP-TLS – Strong PKI Authenticated Transport Layer Security (SSL)  
- Preferred Method Of Authentication



# Example Solution “A”—Access Control and User Policy Enforcement



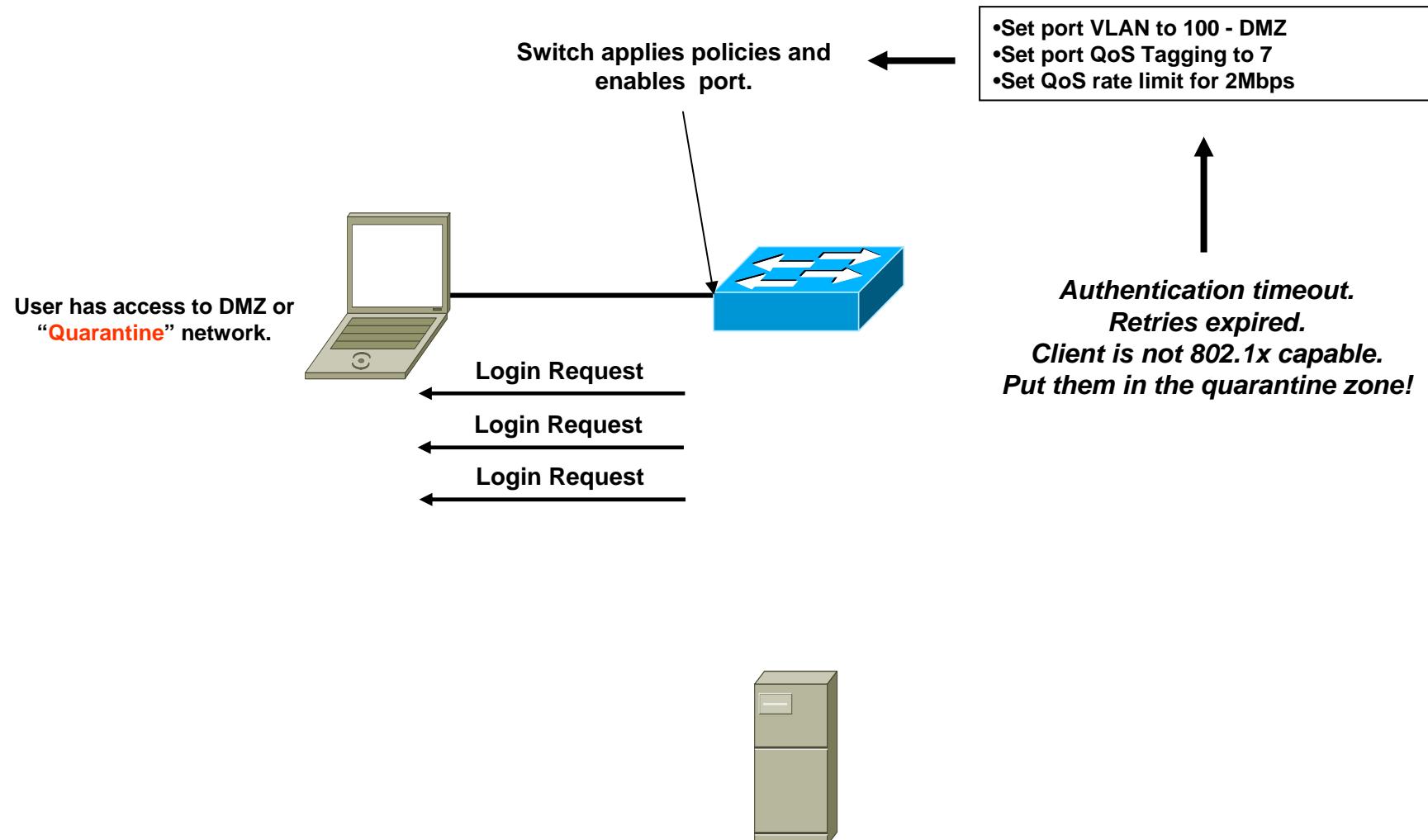
清华大学  
Tsinghua University



# Example Solution “B” – Access For Guest Users



清华大学  
Tsinghua University



# Layer 2 Security Best Practices 1/2



- Manage switches in as secure a manner as possible (SSH, OOB (out-of-bound), permit lists, etc.)
- *Always* use a dedicated VLAN ID for all trunk ports
- Be paranoid: do not use VLAN 1 for anything
- Set all user ports to non trunking
- Deploy port-security where possible for user ports
- Selectively use SNMP and treat community strings like root passwords
- Have a plan for the ARP security issues in your network

# Layer 2 Security Best Practices 2/2



- Enable STP(spanning tree ) attack mitigation (BPDU Guard, Root Guard)
- Use private VLANs where appropriate to further divide L2 networks
- Disable all unused ports and put them in an unused VLAN
- Consider 802.1X for middle term

All of the Preceding Features Are Dependant on  
Your Own Security Policy

# 网络安全工具（Network Security Tools）



- 安全威胁绝大多数是以网络传播，以下是网络安全产品
  - NextScreen, ServGate, Fortinet
  - 天融信，启明星辰，清网数安
- 防火墙Firewall
  - 配置规则
  - 控制访问
  - 阻隔攻击
- 入侵检测保护系统(Intrusion Detection/Protection System)
  - 主机IDS和网络IDS
  - 协议分析和追踪
  - 特征匹配
  - Snort 和Bro
- UTM统一威胁管理系统
  - 组合入侵检测保护和防火墙功能
- 蜜罐网络(Honeynet)

# 网络安全工具演化



- Firewall /SSL VPN
- IDS/IPS web/app proxies
- SEM (security event management)
- NBA (Network Behavior Analysis)

# Firewall



- Software Based
- Asic Based
- Network Processor based

# Network Intrusion Detection System



清华大学  
Tsinghua University

- Network IDS are now able to
  - Understand trunking protocols
  - Fast enough to handle 1 Gbps
  - *Including management of alerts !*
  - Understand layer 2 attacks

# Snort



- 采用Snort克制Code Red病毒的传播
  - 如果用户大量收到类似如下的HTTP请求应为是Code Red在传播。  
GET /default.ida?{之后有224个相同的字}%u9090%u6858%ucbd3%u7801%u9090 %u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00....
- 采用snort规则兼容系统的IDS用户，可以添加如下规则，来监控病毒的请求

```
alert tcp !$HOME_NET any -> $HOME_NET 80 (msg:" Code Red- IIS-Unchecked Buffer in Index Server ISAPI Extension";flags:PA; content:"/default.ida"; nocase;)
```

# Bro



清华大学  
Tsinghua University

- Vern Paxson, [www.bro-ids.org](http://www.bro-ids.org)
- 本质上是一种解释语言
- 与libpcap配合使用

# Honeynet(蜜罐网络)



- Honeypot（蜜罐）是指是伪装的不安全计算机系统，用于吸引攻击，观察攻击的网络工具
- Honeynet（蜜网）用多个Honeypot组和起来的网络系统，用来采集攻击者的信息
- Honeynet的关键技术
  - 网络欺骗
  - 端口重定向
  - 报警
  - 数据控制
  - 数据捕获
- 一般实验采用虚拟机（VM）虚拟出honeynet，其中有防火墙，入侵检测系统和多个服务器

# 结论



- 主机安全是关键
- 网络安全是防御性，通过网络访问控制，隔离病毒和其它攻击网包
- 通过网络安全防范对主机的安全攻击
- 当前主机安全与网络安全日益结合紧密，组成一套整合的防御系统



*Send me your comments to*

*[zhenchen@csnet1.cs.tsinghua.edu.cn](mailto:zhenchen@csnet1.cs.tsinghua.edu.cn)*

# 网络安全原理与技术

## 第三章 防火墙Firewall Chapter 3 Firewall

李军，研究员  
陈震，助理研究员

{zhenchen, jun1}@tsinghua. edu. cn

网络安全实验室，信息技术研究院，  
清华大学  
<http://security.riit.tsinghua.edu.cn>

# Overview



## ● Linux防火墙

- 技术简介
- netfilter 与 iptables 使用
- 简易防火墙
- 实验
  - ◆ 实验 1 设置本机防火墙
  - ◆ 实验 2 设置独立防火墙

## ● Juniper NS-5GT(NextScreen)

# 引子-从Fortinet 被起诉谈起



## ● Linux programmer wins legal victory

- German court supports effort to enforce the GPL, which governs countless projects in the free and open-source software realms.
- By Stephen Shankland
- Staff Writer, CNET News.com; Published: April 14, 2005, 12:43 PM PDT
- A Linux programmer has reported a legal victory in Germany in enforcing the General Public License, which governs countless projects in the free and open-source software realms.
- A Munich district court on Tuesday issued a preliminary injunction barring Fortinet, a maker of multipurpose security devices, from distributing products that include a Linux component called "initrd" to which Harald Welte holds the copyright.
- In addition to being a Linux programmer, Welte runs an operation called the GPL Violations project that attempts to encourage companies shipping products incorporating GPL software to abide by the license terms. The license lets anyone use GPL software in products without paying a fee, but it requires that they provide the underlying source code for the GPL components when they ship such a product.

# Linux防火墙技术简介

# 前言

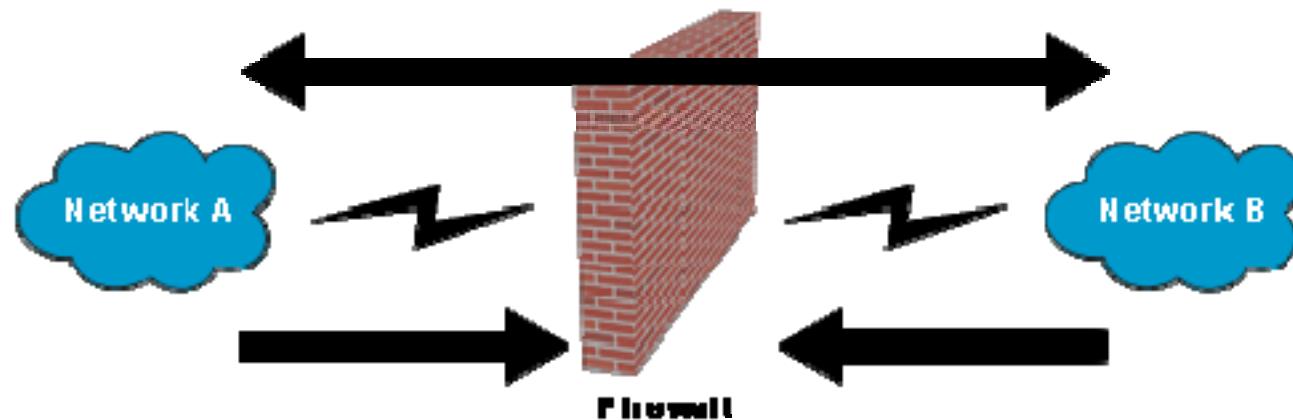


- IETF RFC 2979 定义了防火墙的行为和需求 (Behavior and Requirement)
- 基于以上规范的软件或是硬件，称为防火墙
- 防火墙能够防止未经允许的连接进入由防火墙所保护的网络
- 防火墙是介于其他网络和防火墙所保护的网络之间的中界点，能够让网络数据包从网络连到另一个网络
- 防火墙提供了穿透(Transparency)的功能，并提供了限制穿透的功能

# 基本防火墙



- 在防火墙上可以做任何过滤、伪装、限制或转换的工作



# NAT 也是 Firewall 的一种

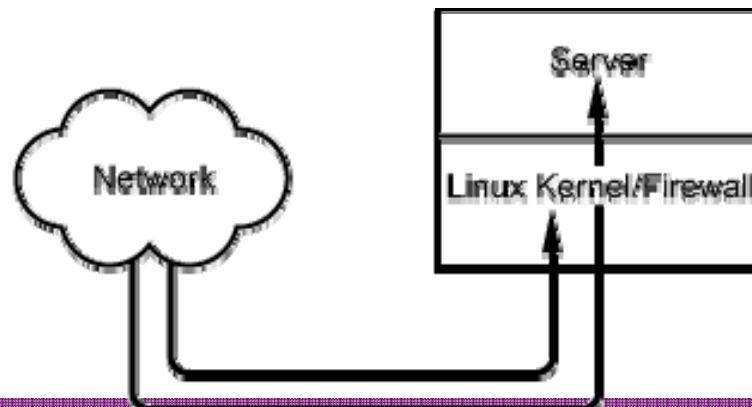


- NAT 能够接收局域网和外部网络的网包并转换地址、端口号后往目标地址传送出，也能够在这个架构上做过滤筛选的动作

# 主机防火墙



- 当前还有许多的防火墙是架设在主机自身，利用主机的网络接口和操作系统的网络核心架设防火墙，过滤不当的数据包进入主机或是阻挡攻击
- 提供了安全，也限制了主机通信自由



# Linux netfilter与iptables使用

# 简介



- netfilter/iptables 是 Linux 2.4.x - 2.8.x 的防火墙网络子系统的一部份
- 提供了 Stateful 的网包过滤、NAT 功能以及进入核心的数据包中添加一些额外的信息方便分类的功能 (Linux 中称之为 Mangling)
- 当网络数据包抵达 Linux 主机的网络卡接口的时候，网络卡会依照数据包的目的 MAC 地址判别是否要接收，在经过中断要求机制之后，数据包会抵达 Linux 网络核心判断是否传送，或本机处理并呼叫 netfilter 核心进行数据包状态的纪录与检查，最后由 netfilter 核心决定是否接收、丢弃或纪录等等

# Stateful Inspection 状态检查

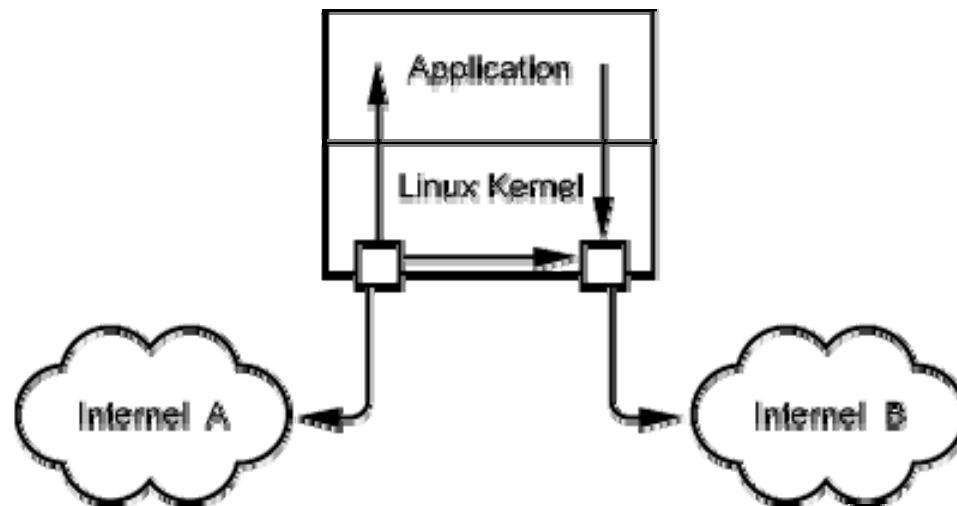


- Stateful Inspection 是现今防火墙的基本功能，表示防火墙本身的检查或过滤条件可以依照链接状态或协议进行分析，而不是只针对单一数据包进行辨识
- Packet Filtering 只针对单一 Packet 中的信息进行过滤，而 Stateful Inspection 则根据网包过去的信息或协议的结果进行过滤与检查的决策，因此实作上也比较复杂

# Linux 防火墙示意图



- 当网包抵达 Linux 防火墙的时候，依照它的目的地可以进入 Linux 主机由应用程序接收，或者是经由另外的接口传送出去



# netfilter



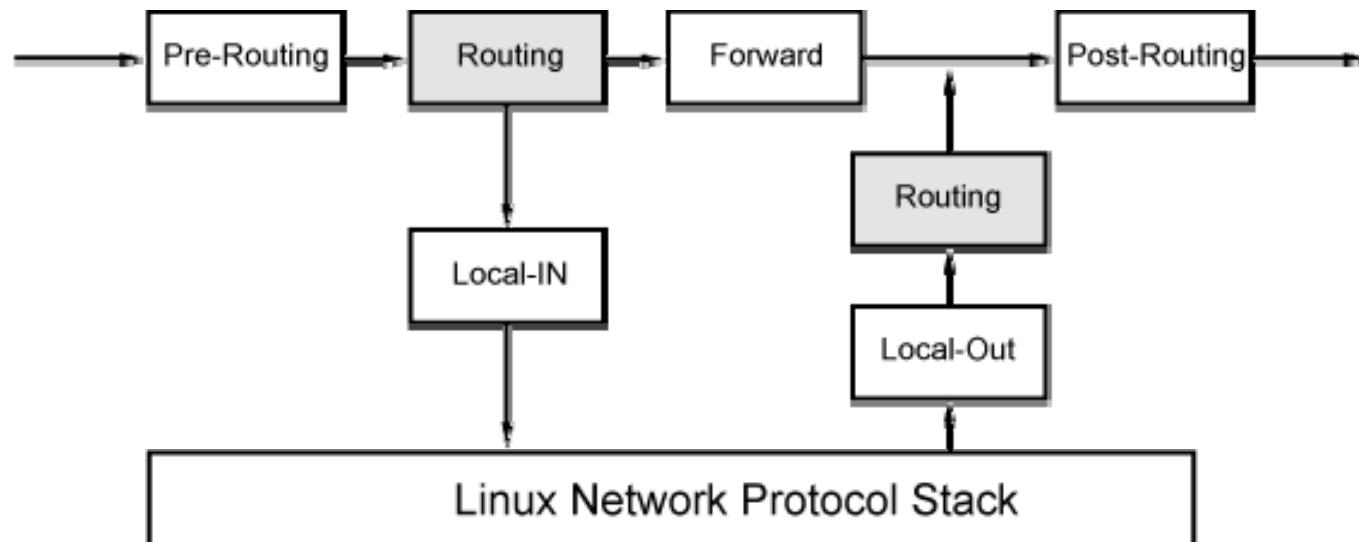
- netfilter 是在网包行进的路上设下一系列的挂入点(Hook)，并依照数据包所属的表格 (Table) 来决定如何处理数据包

# netfilter Hook

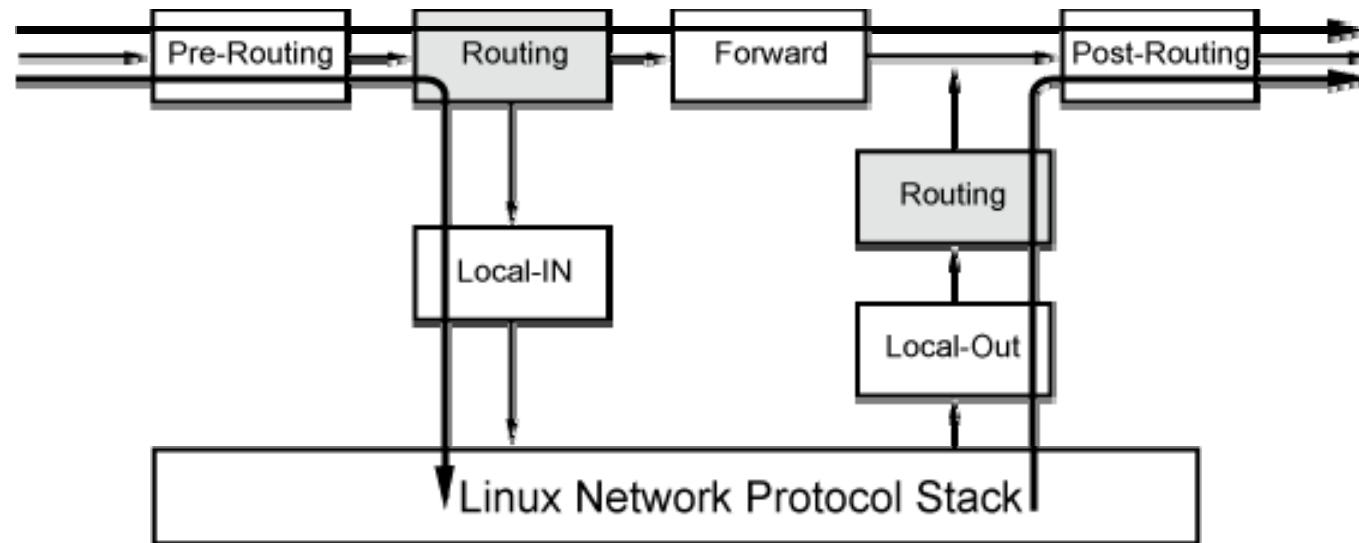


- netfilter 定义了五个挂入点 (Hook)，有 Pre-Routing、Local-IN、Forward、Local-Out 及 Post-Routing 这五个挂入点
- Routing 区块指依照网包的类型决定数据包的去向 (进入哪一处理程序的 Hook 点)
- 在数据包经过挂入点 (Hook) 的时候，可以依照特定的条件做特定的动作。常用的动作为 Accept 数据包、Drop 数据包与进行 NAT 的 Masquerade (伪装) 功能

# netfilter Hook 示意图



# 网包路由图



# 网包路由说明



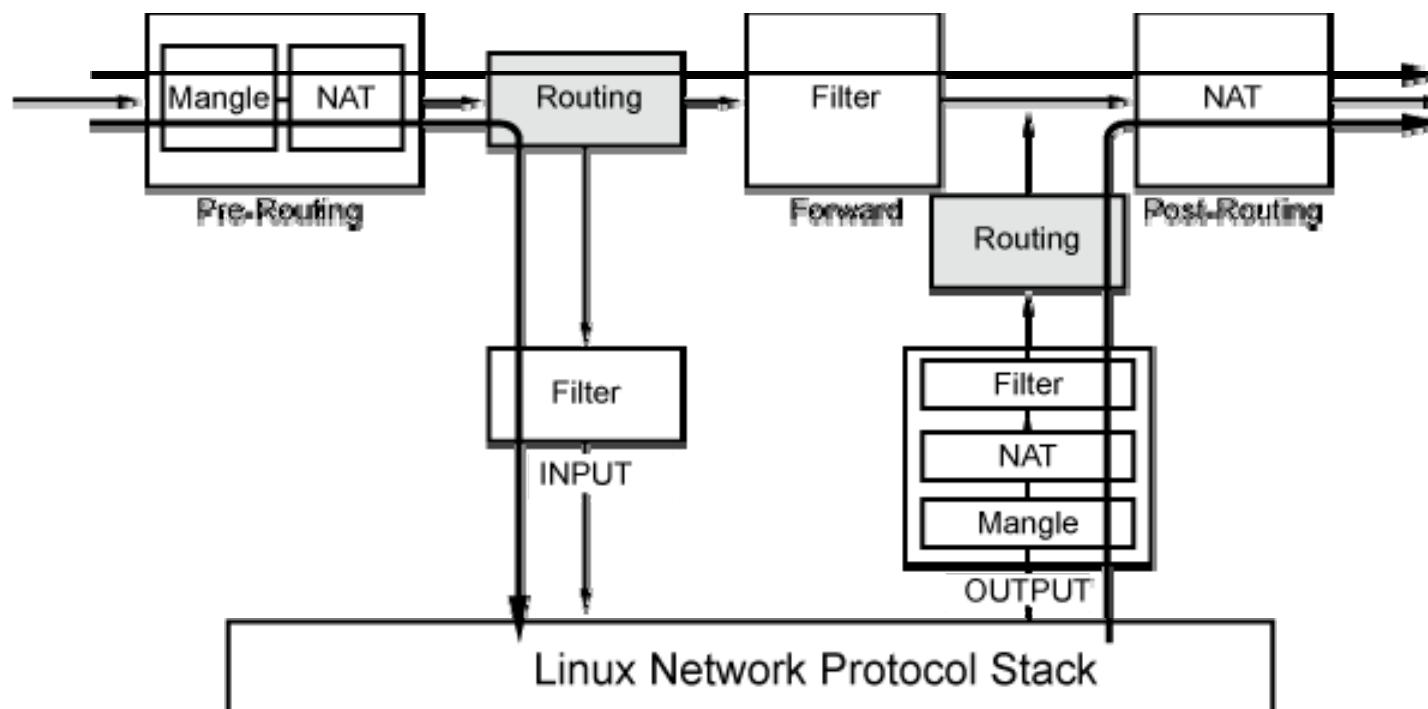
- 当网包进入 Linux 主机的网络接口之后，会先经过 Pre-Routing 这个挂入点，而后由 Routing 机制决定是要进入 Linux 主机（进入 Local-IN 这个挂入点）或是往另外一个网络接口过去（经由 Forward 和 Post-Routing 这两个挂入点出去）
- Linux 若有数据包需要送出的话，则会先经过 Local-Out 这个挂入点，再经过 Routing 和 Post-Routing 挂入点往外送

# netfilter Table

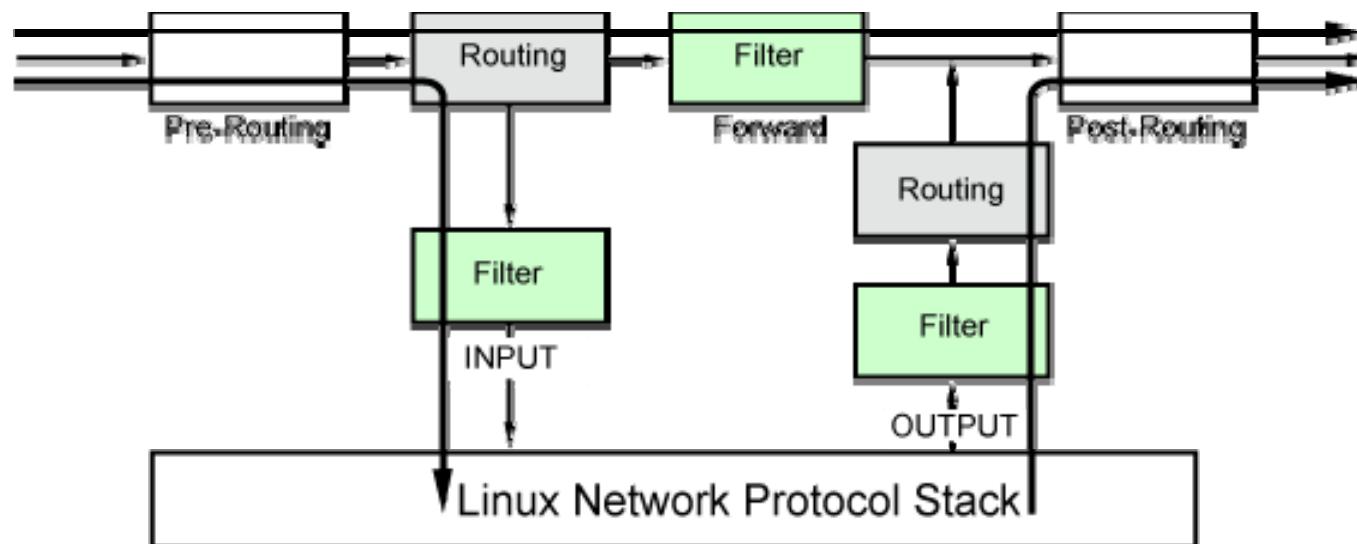


- netfilter 还定义了三个不同的表格(Table),  
NAT、Mangle 和 Filter
- 提供使用者能够藉由在不同的表格中设置动作来  
达到所需功能
- 如 NAT 表格让使用者利用它就可以方便地设计  
NAT 所需的功能, 而 Filter 表格则可以做到过滤  
所有经由 Linux 网络核心中所有数据包的功能

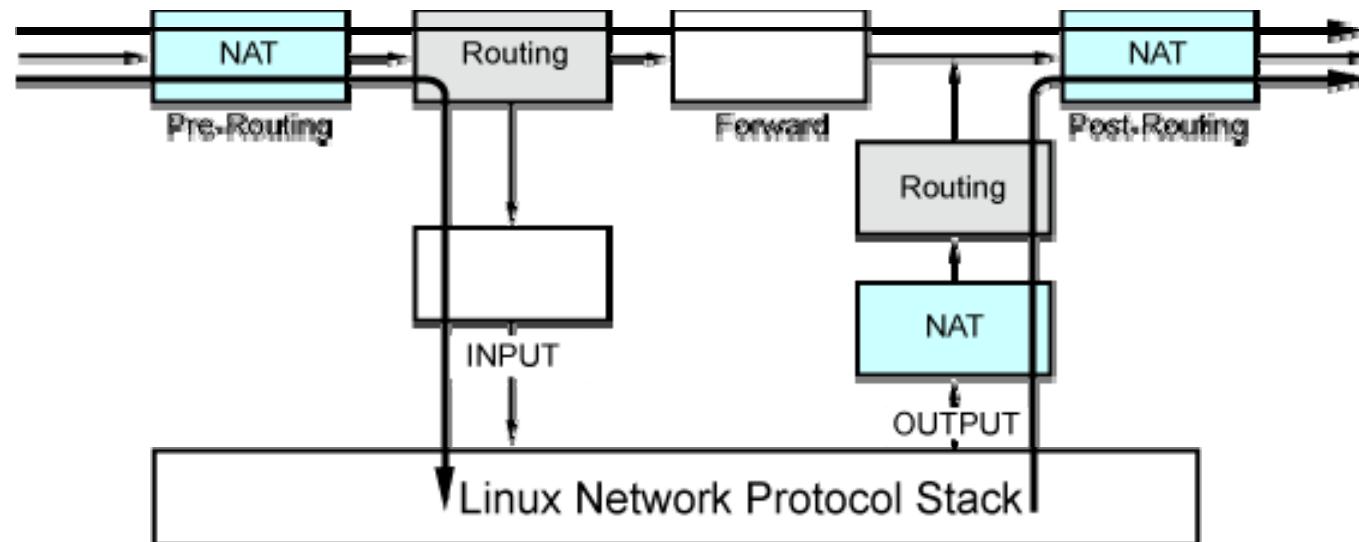
# netfilter 表格示意图



# Filter 表格示意图



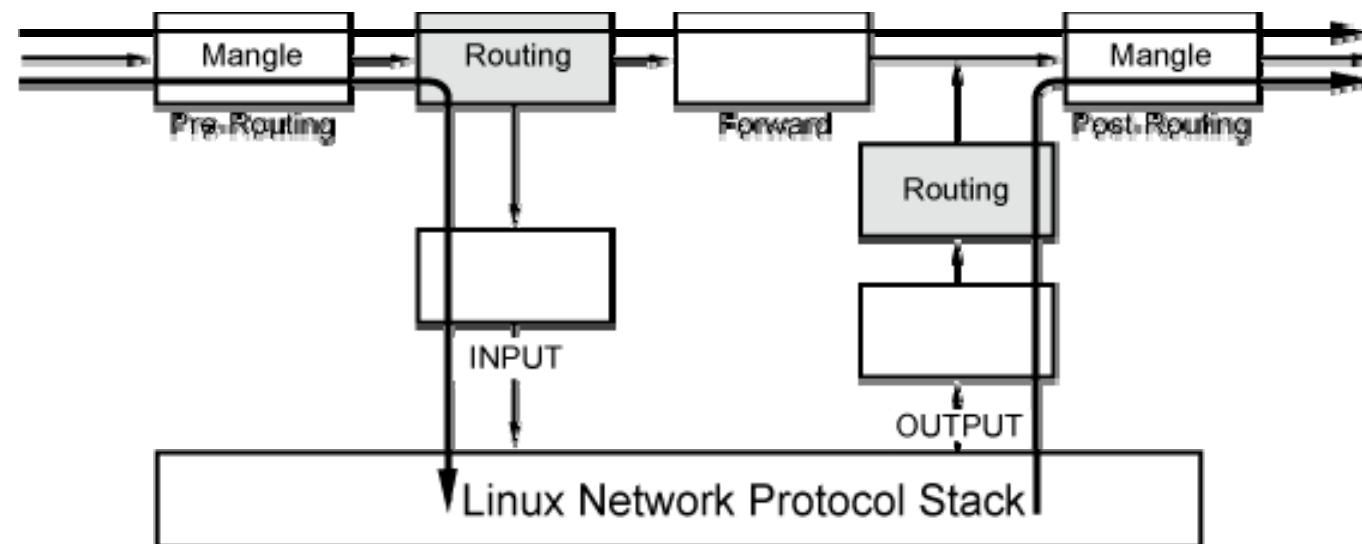
# NAT 表格示意图



# Mangle 表格示意图



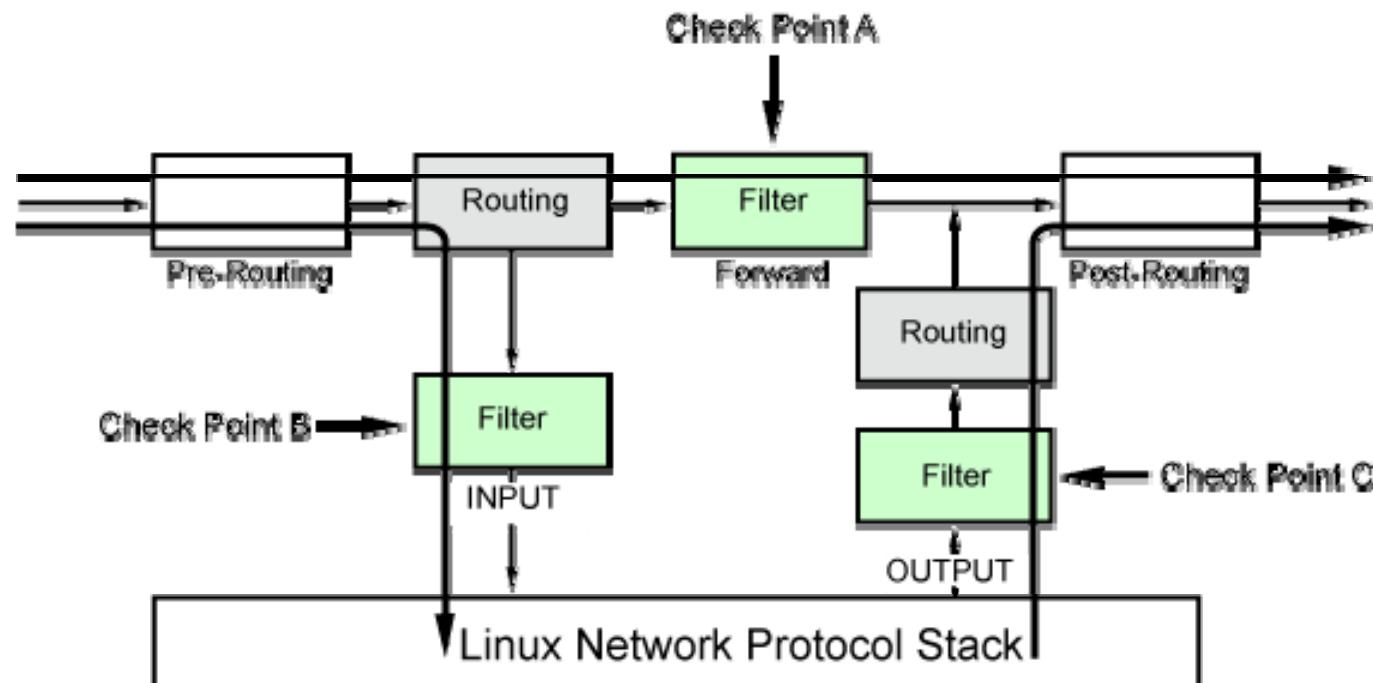
清华大学  
Tsinghua University



# 表格及挂入点概念



清华大学  
Tsinghua University

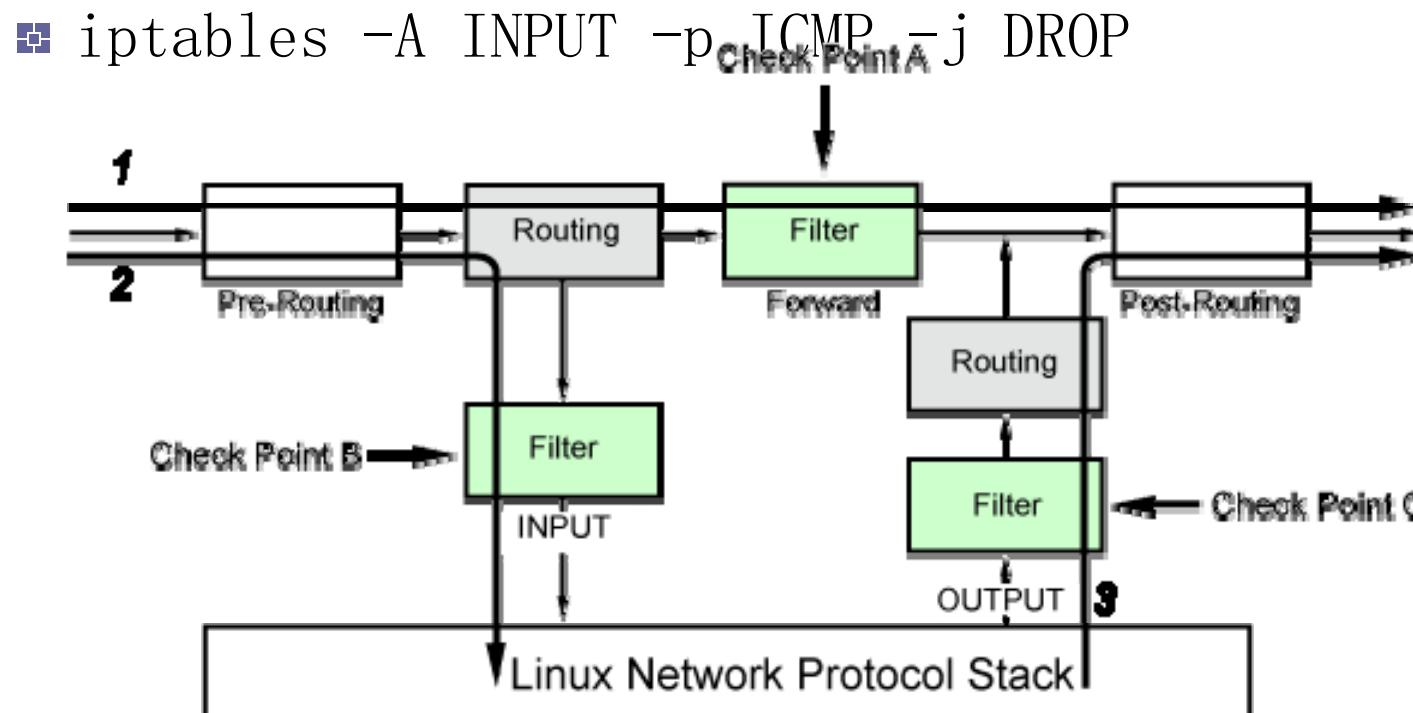


# 以一个示范来解释 iptables 的概念



清华大学  
Tsinghua University

- 假设利用 iptables 在 Filter 表格中设置网包的协议为 ICMP 时，作 DROP 的动作



# 由上一个示范说明 箭头1



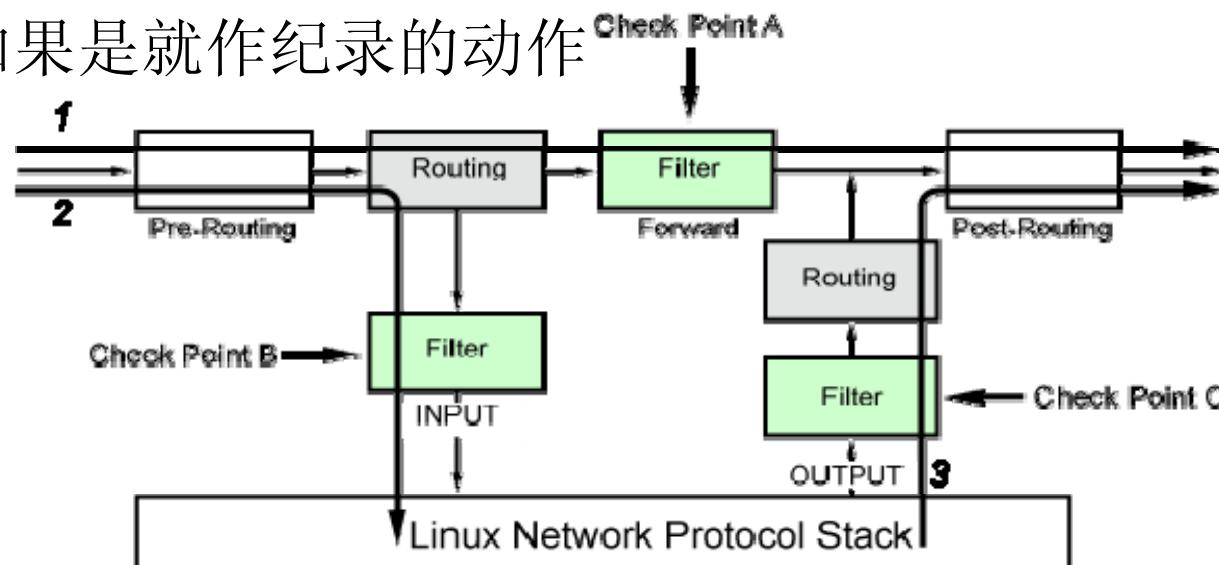
- 经由一个网络接口传送到另一个接口(注：必需启动 Linux 核心数据包传送的功能)，如箭头 1：
- 数据包会经由 Pre-Routing 进入，经过 Forward 网包到 Post-Routing 由另一个接口出去，因此数据包在经过 Forward 挂入点(检查点A)的时候会作检查是否为 ICMP 数据包，如果是就作纪录的动作

# 由上一个示范说明 箭头2



- 经由网络接口进到 Linux 本机的应用程序，如箭头2：

- 数据包会经由 Pre-Routing 进入，经过 INPUT网包到适当的应用程序，因此数据包在经过 INPUT 挂入点(检查点B)的时候会作检查是否为 ICMP 数据包，如果是就作纪录的动作

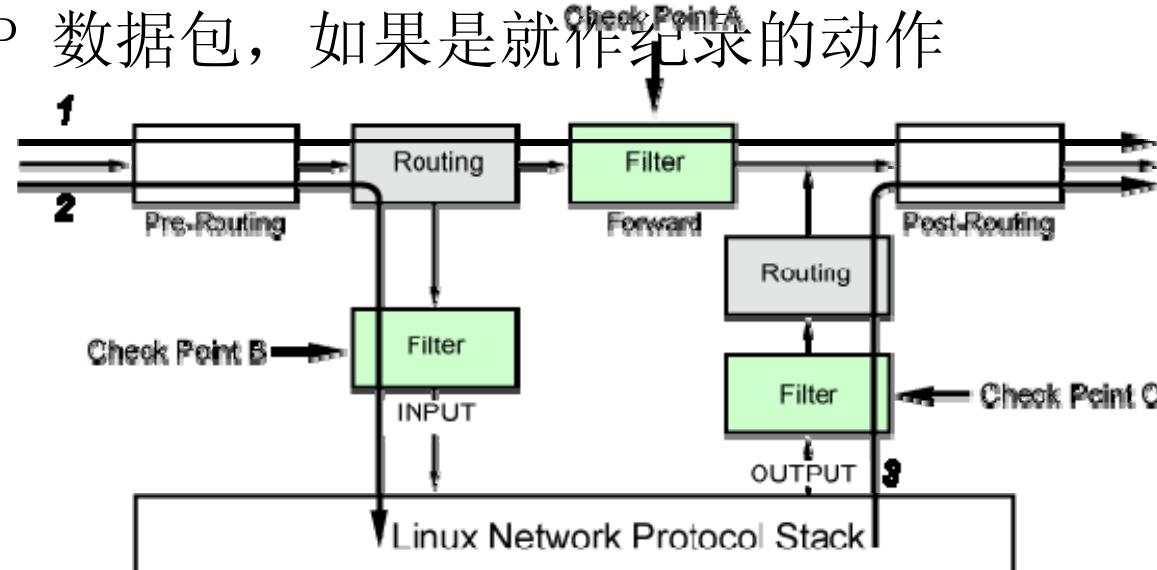


# 由上一个示范说明 箭头3



- 由 Linux 本机的应用程序往外传送的网包，如箭头3：

- 数据包由应用程序生成后，它会经进入OUTPUT，经过 OUT网包 之后到适当的应用程序，因此数据包经过 OUTPUT 挂入点(检查点C)的时候会作检查是否为 ICMP 数据包，如果是就作纪录的动作



# Connection Tracking 连接追踪

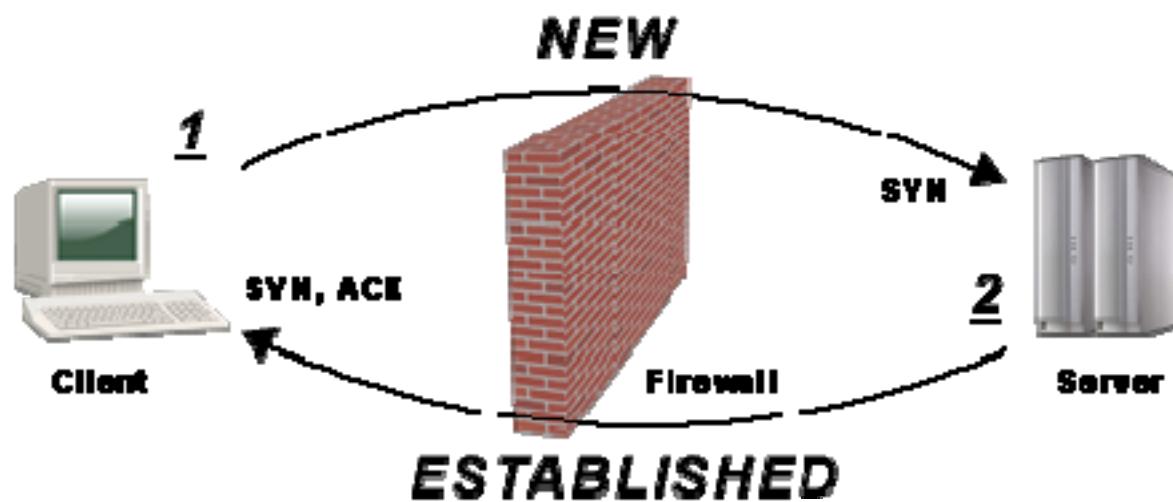


- Connection Tracking 就是让 netfilter 能够分辨并管控连接状态的机制
- Connection Tracking 也可由使用者自行添加新的协议分析
- 在 netfilter 里面，TCP Connection Tracking 可透过 iptables 控制
- TCP Connection Tracking可以分成四个 State：NEW、ESTABLISHED、RELATED和INVALID
- 除了本机生成的数据包是在 Output Chain以外，所有的 Connection Tracking 都是在 Pre-Routing Chain 里运作

# TCP 连接的 Connection Tracking 状态图



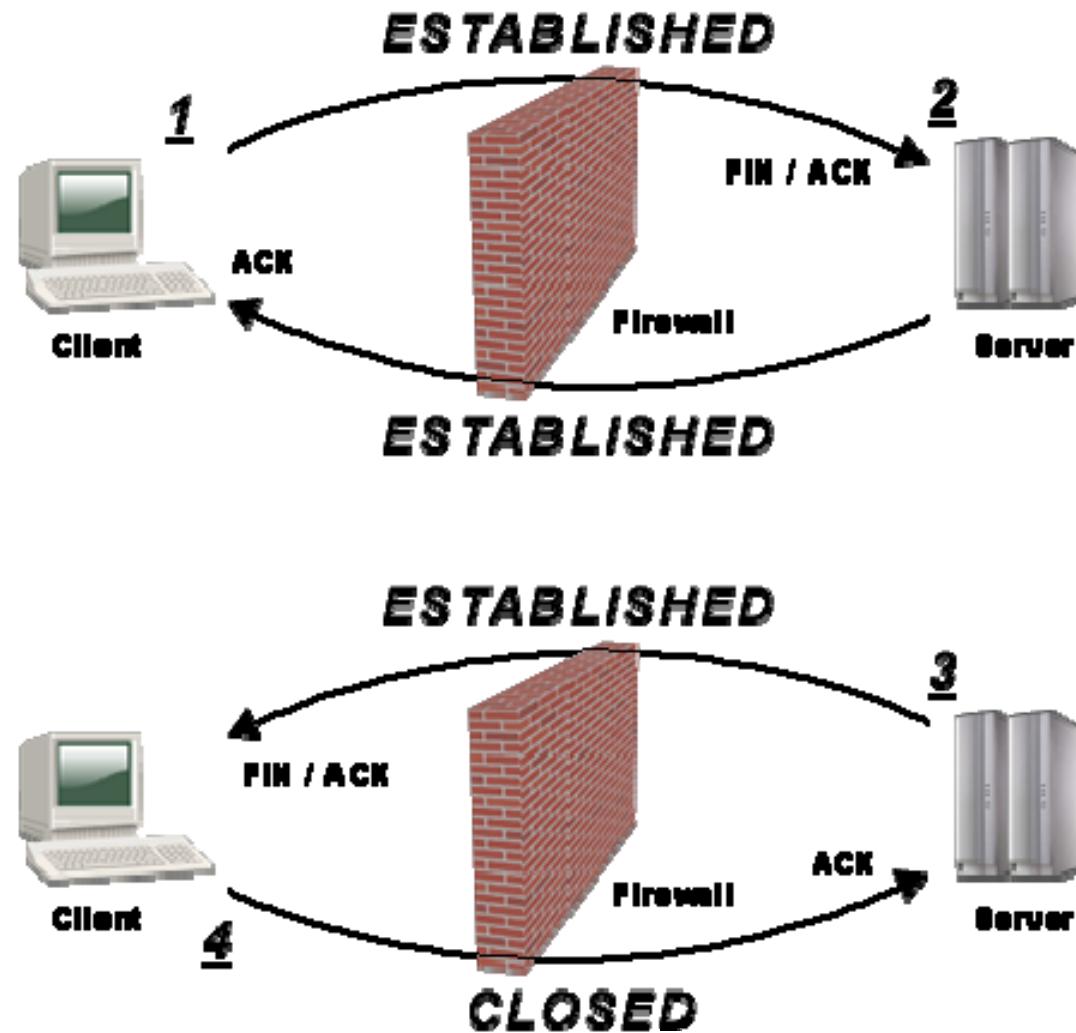
清华大学  
Tsinghua University



# TCP 关闭连接的 Connection Tracking 状态图



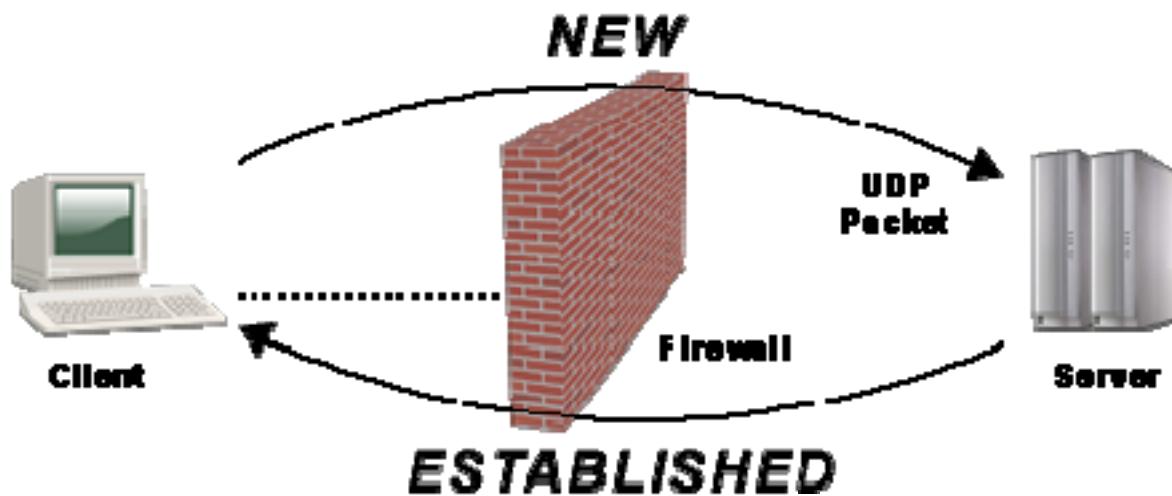
清华大学  
Tsinghua University



# UDP 连接的 Connection Tracking 状态图



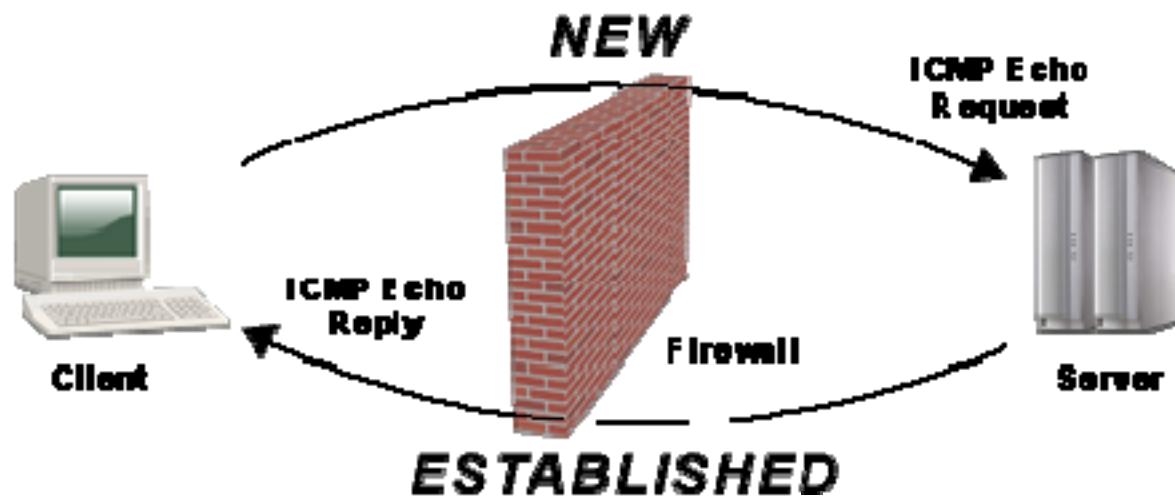
清华大学  
Tsinghua University



# ICMP 连接的 Connection Tracking 状态图



清华大学  
Tsinghua University





# iptables 命令

- iptables 是 userspace 的 netfilter 核心控制程序
- iptables 提供了表格(Table), 链式(Chain) , 以及规则(Rule) 这三种机制让使用者设置
- 表格分成三种: Filter、NAT及 Mangle,
  - Filter 表格里面有 INPUT、OUTPUT 和 Forward Chain
  - 在各个表格里面有不同的 Chain，而在各个 Chain 里面还可以添加各种不同的规则
- iptables -v -L 能将所有指定表格（不加表格则是 Filter 表格）的规则及其详细数据列出。用来解释表格、链式和规则的概念

# 以一个示范解释： iptables -v -L 运行结果



| # iptables -v -L                                     |       |        |          |    |     |               |
|------------------------------------------------------|-------|--------|----------|----|-----|---------------|
| Chain INPUT (policy ACCEPT 2043 packets, 138K bytes) |       |        |          |    |     |               |
| pkts                                                 | bytes | target | prot opt | in | out | source        |
| 0                                                    | 0     | DROP   | all      | -  | any | 192.168.0.100 |
| 52                                                   | 2828  | test   | all      | -- | any | anywhere      |

| Chain FORWARD (policy ACCEPT 0 packets, 0 bytes) |       |        |          |    |     |             |
|--------------------------------------------------|-------|--------|----------|----|-----|-------------|
| pkts                                             | bytes | target | prot opt | in | out | source      |
|                                                  |       |        |          |    |     | destination |

| Chain OUTPUT (policy ACCEPT 1023 packets, 64876 bytes) |       |        |          |    |     |          |
|--------------------------------------------------------|-------|--------|----------|----|-----|----------|
| pkts                                                   | bytes | target | prot opt | in | out | source   |
| 0                                                      | 0     | DROP   | all      | -  | any | anywhere |

| Chain test (1 references) |       |        |          |     |     |          |
|---------------------------|-------|--------|----------|-----|-----|----------|
| pkts                      | bytes | target | prot opt | in  | out | source   |
| 4                         | 400   | REJECT | icmp     | any | any | anywhere |

Chain

Rule

# 运行结果解说



- 所有由 192.168.0.100 企图送到主机的网包都会被 DROP 掉
- 无法对 192.168.0.100 连接，因为所有会往 192.168.0.100 的数据包也都会被 DROP 掉
- 所有送到本机的 ICMP 数据包都会被拒绝，并回送 icmp-port-unreachable 的数据包

# iptables 命令的格式



- iptables 命令的格式如下

```
# iptables [-t table] command [match] [-j  
target/jump]
```

- 举例来说

```
# iptables -t filter -A INPUT -s 192.168.0.1 -j  
LOG
```



# 针对各个参数作介绍 (1)

## ● table

- 在下 iptables 命令的时候，我们必须先指定其中一个表格来设置规则或是动作，不指定特定表格都是默认为 Filter 表格

## ● command

- command 指的是针对 iptables 的表格作指定的动作，主要是新增规则到某个 Chain 中或是对于 Chain 的处理
- match: match 指的是当遇到网包符合特定的状况或是有特定的连接状况发生的时候，iptables 的规则会生成作用



# 针对各个参数作介绍 (2)

## ● match

- generic match 是在一般状况的 match 比对
- implicit match 则是如 tcp、udp、icmp 等依照协议特殊选项
- explicit match, 这个 match 必须使用 -m 或 --match, 依照不同的 match 模块可以有不同的选项, 因此在这边只介绍一些较常见的 match, 也可以自行编写新的 iptables match, 或是找合用的 patch 来使用

# 针对各个参数作介绍 (3)



## ● target/jump

- target/jump 是指定对网包或是连接所做的动作

# iptables Table 列表



| Table                             | Description                                                          |
|-----------------------------------|----------------------------------------------------------------------|
| -t nat<br>or<br>--table nat       | The table has PREROUTING and POSTROUTING chains.                     |
| -t mangle<br>or<br>--table mangle | The table has PREROUTING, FORWARD and POSTROUTING chains.            |
| -t filter<br>or<br>--table filter | This is the default table. It has INPUT, FORWARD and FORWARD chains. |

# iptables command 列表



| Command           | Brief Description                                                          |
|-------------------|----------------------------------------------------------------------------|
| -A --append       | Appends a new rule to the end of selected chain.                           |
| -D --delete       | Deletes a rule from selected chain.                                        |
| -R --replace      | Replaces a rule in selected chain.                                         |
| -I --insert       | Inserts a new rule into specified position of selected chain.              |
| -L --list         | Lists all rules in selected chain.                                         |
| -F --flush        | Deletes all rules in selected chain.                                       |
| -Z --zero         | Resets packet counter of specified chain.                                  |
| -N --new-chain    | Creates a new user-defined chain.                                          |
| -X --delete-chain | Deletes a user-specified chain, and remove all rules defined in the chain. |
| -P --policy       | Sets policy for the chain.                                                 |
| -E --rename-chain | Renames the user specified chain.                                          |

# iptables 额外 command 列表



| Command           |                                                         | Brief Description                                                                                                    |
|-------------------|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| -v --verbose      | --list<br>--append<br>--insert<br>--delete<br>--replace | The --list parameter lists detailed information. Other parameters print detailed results of each respective command. |
| -x --exact        | --list                                                  | Displays the detailed information in exact numbers. Using -v only lists in K (thousands), M (millions).              |
| -n --numeric      | --list                                                  | Lists numeric values.                                                                                                |
| --line-numbers    | --list                                                  | Lists rules' line numbers.                                                                                           |
| -c --set-counters | --insert<br>--append<br>--replace                       | Initialize respective counters.                                                                                      |
| --modprobe        | All                                                     | Specifies position in iptables for modules.                                                                          |

# iptables Generic match 列表



清华大学  
Tsinghua University

| Generic Match          | Brief Description                                                                                                                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -p --protocol          | Specifies protocol, protocol name and number; details in file /etc/protocol. Use of "!" excludes specified protocol from matching. For example, protocol !tcp results in only comparing UDP/ICMP and GRE. |
| -s --src --source      | Specifies source address. Use of CIDR address allowed; use of "!" symbol also allowed as specified above.                                                                                                 |
| -d --dst --destination | Specifies destination address. Use of CIDR address allowed; use of "!" symbol also allowed as specified above.                                                                                            |
| -i --in-interface      | Specifies interface via which a packet is to be received. Use of + denotes inclusion of all interfaces of type specified. For example, eth+ refers to all ethN interfaces.                                |
| -o --out-interface     | Specifies interface via which a packet is to be sent out. Use of + denotes inclusion of all interfaces of type specified. For example, eth+ refers to all ethN interfaces.                                |
| -f --fragment          | Specifies fragmented packets for matching.                                                                                                                                                                |

# iptables implicit match 列表



清华大学  
Tsinghua University

| Implicit Match |                               | Brief Description                                                                                                          |
|----------------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| -p tcp         | --sport<br>--source-port      | Specifies source ports, use of : denotes a range of ports. For example, --sport 23:80 specifies source ports from 23 to 80 |
|                | --dport<br>--destination-port | Specifies destination ports. Same usage of symbol : allowed as in sport above.                                             |
|                | --tcp-flags                   | Specifies TCP flags, SYN, FIN, ACK, RST, URG, and PSH. Use of NONE denotes no flags, or ALL, all flags.                    |
|                | --syn                         | Specifies TCP connection state at time of initiation. Substitutions with tcp-flags SYN, RST, and ACK SYN allowed.          |
|                | --tcp-option                  | Sets special TCP option for matching.                                                                                      |
| -p udp         | --sport<br>--source-port      | Specifies source ports, use of : denotes range of ports. For example, --sport 23:80 denotes ports 23 to 80.                |
|                | --dport<br>--destination-port | Specifies destination ports, same usage of : allowed as in sport above.                                                    |
| -p icmp        | --icmp-type                   | Specifies ICMP data type.                                                                                                  |

# iptables explicit match 列表



清华大学  
Tsinghua University

| Explicit Match |                    | Brief Description                                                                                                                    |
|----------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| limit          | --limit X/Y        | Sets maximum match limit. For example, 3/hour refers to 3 matches per hour. Time unit used can be seconds, minutes, hours, and days. |
|                | --limit-burst      | Sets maximum number of impromptu packets allowed for matching within a specified time period.                                        |
| mac            | --mac-source       | Sets source MAC address.                                                                                                             |
| mark           | --mark             | Marks the packet.                                                                                                                    |
| multiport      | --source-port      | Defines various ports for matching.                                                                                                  |
|                | --destination-port |                                                                                                                                      |
|                | --port A,B,C,D     |                                                                                                                                      |
| owner          | --uid-owner        | Sets user ID of packet owner (originator) for matching.                                                                              |
|                | --gid-owner        | Sets group ID of packet owner (originator) for matching.                                                                             |
|                | --pid-owner        | Sets process ID of packet for matching.                                                                                              |
|                | --sid-owner        | Sets session ID of packet for matching                                                                                               |
| state          | --state            | Sets connection state for matching: INVALID, ESTABLISHED NEW and RELATED.                                                            |
| tos            | --tcs              | Sets parameter for matching with entry in Type Of Service field.                                                                     |
| ttl            | --ttl              | Sets parameter for matching with entry in Time to Live field.                                                                        |

# iptables target/jump 列表



| Target/Jump |                    | Brief Description                                                                                                                                                                                                                                                                                     |
|-------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCEPT      |                    | Accepts the packet, aborts the current table and proceeds to the next table.                                                                                                                                                                                                                          |
| DROP        |                    | Ignores and drops the packet, and aborts the iptables framework without proceeding to the next chain or table.                                                                                                                                                                                        |
| LOG         | --log-level        | Records level of logging.                                                                                                                                                                                                                                                                             |
|             | --log-prefix       | Adds prefix to logged messages with the specified string.                                                                                                                                                                                                                                             |
|             | --log-tcp-sequence | Logs sequence number from TCP packet header.                                                                                                                                                                                                                                                          |
|             | --log-tcp-options  | Logs options from TCP packet header.                                                                                                                                                                                                                                                                  |
|             | --log-ip-options   | Logs options from IP packet header.                                                                                                                                                                                                                                                                   |
| MARK        | --set-mark         | Sets mark.                                                                                                                                                                                                                                                                                            |
| MASQUERADE  | --to-ports         | Redirects to specified port after masquerading packet.                                                                                                                                                                                                                                                |
| QUEUE       |                    | Queues packet to userspace for the applications.                                                                                                                                                                                                                                                      |
| REDIRECT    | --to-ports         | Redirects packet to a port on the server.                                                                                                                                                                                                                                                             |
| REJECT      | --reject-with      | Rejects packet using method specified. For example, if tcp has tcp-reset, sends the RST packet to refuse connection. Using ICMP can specify the following responses: icmp-net-unreachable,icmp-host-unreachable, icmp-net-prohibited, and icmp-host-prohibited. The default is icmp-port-unreachable. |
| RETURN      |                    | Exits current chain, and returns to previous table and resume at next rule in previous chain.                                                                                                                                                                                                         |
| TOS         | --set-tos          | Sets the TOS field.                                                                                                                                                                                                                                                                                   |
| TTL         | --ttl-set          | Sets the TTL field, default value is 64.                                                                                                                                                                                                                                                              |
|             | --ttl-dec          | Decrement the packet's TTL field by 1.                                                                                                                                                                                                                                                                |
|             | --ttl-inc          | Increments the packet's TTL field by 1.                                                                                                                                                                                                                                                               |
| ULOG        | -ulog-nlgroup      | Sets log group in userspace, and allows use of netlink database to retrieve the logged content.                                                                                                                                                                                                       |
|             | -ulog-prefix       | Sets uLog's prefix.                                                                                                                                                                                                                                                                                   |
|             | -ulog-cprange      | Sets size of log to be copied to userspace.                                                                                                                                                                                                                                                           |
|             | -ulog-qthreshold   | Sets the upper limit for number of packets to accumulate before being sent on to queue in userspace.                                                                                                                                                                                                  |

# iptables 示范



- 本机 port 53 接受来自 1.2.3.4 的 UDP 网包

```
# iptables -A INPUT -p UDP -s 1.2.3.4 --dport 53  
-j ACCEPT
```

- 新增一个 Rule，下一行则是将其删除，注意要删除的命令必须与新增时的命令相同

```
# iptables -A test -p tcp -j ACCEPT
```

```
# iptables -D test -p tcp -j ACCEPT
```

# iptables 示范



- 接受 ICMP type 为 3 ( ping ) 的网包

```
# iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
```

- 若每秒内出现 20 个 tcp SYN 数据包，则记录下来，prefix 为 SYN Flood。

```
# iptables -A INPUT -p tcp --syn -m limit --limit  
20/second -j \ LOG --log-prefix "SYN Flood:"
```

# iptables 示范



- Connection Tracking 的 State 为 new 的 tcp 网包不是SYN，设置将这个数据包记录起来，prefix 设成“Error New State”

```
# iptables -A INPU网包p tcp ! --syn -m state --  
state NEW -j LOG \ --log-prefix "Error New  
State:"
```

# 简易防火墙

# 两个简单的 Linux 防火墙



- 接下来以设置两个简单的 Linux 防火墙为例：
- 本机防火墙。
- 独立防火墙。

# 本机防火墙



- 在不需要额外网络设备的状况下，可以使用 netfilter 架构直接在 Linux 的服务器上实作 防火墙

# 架设一台有两个网络装置的 Linux 主机



- 假设要架设一台有两个网络装置的 Linux 主机，一个网络端口提供因特网的 www 和 mail(POP3 ,SMTP) 服务，另外一个网络端口则是提供 telnet 服务供局域网地址为 192.168.0.0/24 的连接管理机制
- 假设局域网连接的端口是 eth0，而因特网的网络端口是 eth1。依照以上的需求可以知道，eth0 要开放 telnet (port 23) 使用，而 eth1 要开放 www (port 80)、mail (port 25) 及 POP3 (110)

# 依照上述的假设，应以下列的步骤 进行防火墙的设置，步骤 1



清华大学  
Tsinghua University

## 1. 拒绝所有的网包

- 由于要设置的是防火墙，也就是要禁止除了许可的连接之外的任何通信连接，因此我们先默认拒绝掉所有的数据包，尔后再依照需求新增规则来开放通行
- 命令：

```
# iptables -P INPUT DROP
```

```
# ip网包les -P OUTPUT DROP
```

```
# iptables -P FORWARD DROP
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置，步骤 2



清华大学  
Tsinghua University

## 2. 开启新的 Chain

- 开启两个不同的 Chain，命名为 myLAN 和 myWAN，  
供局域网以及因特网两个不同的接口使用。并且设置  
一个提供允许连接的 Chain: allowed
- 命令：

```
# iptables -N myLAN
# iptables -N myWAN
# iptables -N myLANback
# iptables -N myWANback
# iptables -N allowed
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置，步骤 3



清华大学  
Tsinghua University

## 3. 设置动作

- 先设置可以通过防火墙进入主机的连接类型。第一行允许 SYN数据包，也就是一开始创建连接的数据包，第二行允许的是 ESTABLISHED 和 RELATED 状态的数据包，第三行则是将所有的 TCP数据包 DROP 掉，也就是说，当前两行规则都不符合的时候，就将数据包 DROP 掉
- 命令：

```
# iptables -A all网包d -p TCP --syn -j ACCEPT
# iptables -A allowed -p TCP -m state --state
    ESTABLISHED,RELATED \ -j ACCEPT
# iptables -A allowed -p TCP -j DROP
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置，步骤 4



清华大学  
Tsinghua University

## 4. 增加通行规则

- 接下来分别对因特网和局域网设置规则，让服务器能提供服务。第一行是局域网 192.168.0.0/24 能连上 port 23。下面三行则是分别对因特网开放 port 25、80 及 110
- 命令：

```
# iptables -A myLAN -p TCP -s 192.168.0.0/24 --dport 23 -j allowed
# iptables -A myWAN -p TCP -s 0/0 --dport 25 -j allowed
# iptables -A myWAN -p TCP -s 0/0 --dport 80 -j allowed
# iptables -A myWAN -p TCP -s 0/0 --dport 110 -j allowed
# iptables -A myLANback -p TCP -d 192.168.0.0/24 --sport 23 -j allowed
# iptables -A myWANback -p TCP -d 0/0 --sport 25 -j allowed
# iptables -A myWANback -p TCP -d 0/0 --sport 80 -j allowed
# iptables -A myWANback -p TCP -d 0/0 --sport 110 -j allowed
```

依照上述的假设，应以下列的步骤  
进行防火墙的设置，步骤 5



清华大学  
Tsinghua University

## 5. 添加 filter 表格中

- 在分别设置好两个 Chain 之后，我们开始将它添加到 Filter 表格中。在这个示范中，防火墙的目的是为了过滤进出本机的数据包，因此，我们只需要将它添加到 Filter 表格的 INPUT 和 OUTPUT Chain 中即可，也就是说，在数据包进入本机或是由本机发出时做检查即可，FORWARD Chain 如果没什么特殊状况要设置就可以是空的
- 由于 WWW、SMTP、POP3 及 telnet 网包等服务都是 TCP 连接，因此对于 UDP 及 ICMP 数据包为避免意外状况发生，因此全都 DROP 掉

# 依照上述的假设，应以下列的步骤 进行防火墙的设置，步骤 5



清华大学  
Tsinghua University

## ■ 命令：

```
# iptables -A INPUT -p TCP -i eth0 -j myLAN  
  
# iptables -A INPUT -p TCP -i eth1 -j myWAN  
  
# iptables -A OUTPUT -p TCP -j myLANback  
  
# iptables -A OUTPUT -p TCP -j myWANback  
  
# iptables -A INPUT -p UDP -j DROP  
  
# iptables -A INPUT -p ICMP -j DROP  
  
# iptables -A OUTPUT -p UDP -j DROP  
  
# iptables -A OUTPUT -p ICMP -j DROP
```

# 独立的防火墙

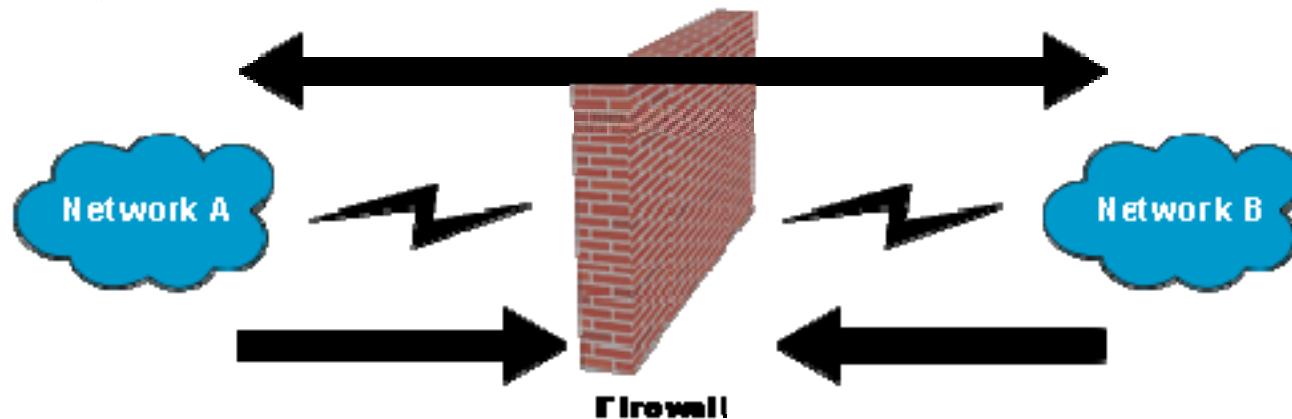


- 独立的防火墙可以分为两种，一种是因特网和局域网的防火墙，除了防火墙功能以外还有 NAT 的功能，另一种则是单纯就当防火墙使用，不做任何的地址转换工作
- 在建立独立的防火墙之前，为了要让 Linux 主机的两个网络接口之间能够互通，也就需要有 Bridge 的功能
- 在 Linux 核心版本 2.4.18 之后，核心内都有 Bridge 的选项，因此必须先将 Bridge 设置起来，并安装 userspace 命令

# 独立防火墙示范



- 假设网络 B 里面提供的服务主要是 port 2000~3000，并设置让 IP 地址为 10.50.100.1 的主机能够以 SSH 进到网络 B 中各主机的 port 22 做管理及设置的工作



# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (步骤1)



清华大学  
Tsinghua University

## ● 启动 Bridge 模式

- 新增一个 Bridge 接口，并且取消掉 STP

```
# brctl addbr br0
```

```
# brctl stp br0 off
```

- 接下来将两个要连接在一起的网络接口都接到 br0 上面

```
# brctl addif br0 eth1
```

```
# brctl addif br0 eth2
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (步骤1)



清华大学  
Tsinghua University

- 接下来将 eth1 和 eth2 原有的网络地址 down 下来，以避免不必要的麻烦。并且设置 0.0.0.0 为这两个网络的地址

```
# ifconfig eth1 down  
  
# ifconfig eth2 down  
  
# ifconfig eth1 0.0.0.0 up  
  
# ifconfig eth2 0.0.0.0 up  
  
# ifconfig br0 up
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (步骤2)



清华大学  
Tsinghua University

## ● 过滤器设置

- 依照前述默认的功能，我们要开启防火墙的 port 2000~3000 的连接，并且让 10.50.100.1 能够连接到各防火墙内的 port 22
- 这部分的设置和本机防火墙一样，都是要设置 Filter 表格

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (步骤2)



- 默认拒绝所有网包：

```
# iptables -P INPUT DROP  
# iptables -P OUTPUT DROP  
# iptables -P FORWARD DROP
```

- 开启新 Chain 并设置可进来作连接动作的类型

```
# iptables -N Wall  
# iptables -N allowed  
# iptables -A allowed -p TCP --syn -j ACCEPT  
# Iiptables -A allowed -p TCP -m state --state  
ESTABLISHED, RELATED \ -j ACCEPT  
# iptables -A allowed -p TCP -j DROP
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (步骤2)



清华大学  
Tsinghua University

## ■ 增加通行规则

```
# iptables -A Wall -p TCP -s 0.0.0.0/0 --dport  
2000:3000 -j allowed  
# iptables -A Wall -p TCP -s 10.50.100.1 --dport 22 -j  
allowed
```

## ■ 添加到 filter 表格中

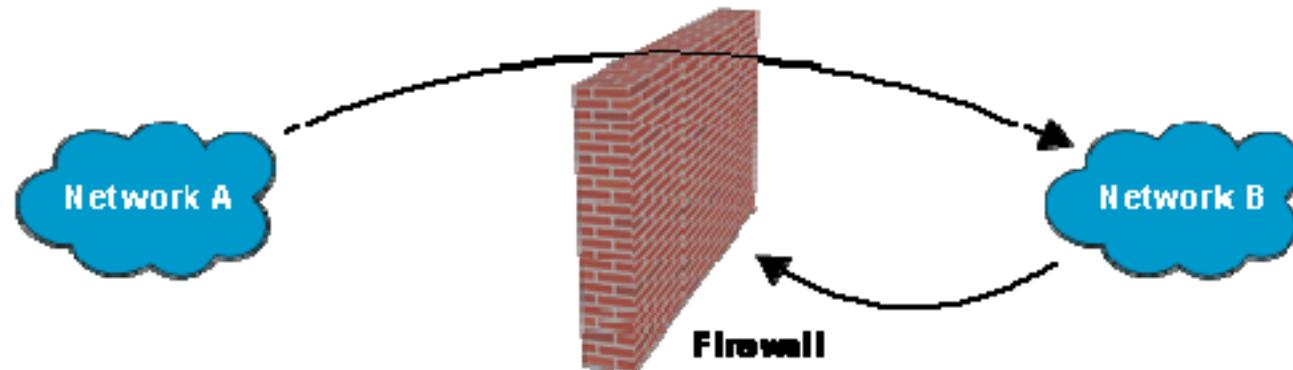
```
# iptables -A FORWARD -p TCP -i eth1 -j Wall  
# iptables -A FORWARD -p UDP -j DROP  
# iptables -A FORWARD -p ICMP -j DROP
```

# 举例说明：只进不出的防火墙



清华大学  
Tsinghua University

- 设置只提供网络 A 到网络 B 的连接，但没有往回走的路



# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (1)



清华大学  
Tsinghua University

## ● 增加新 Chain

```
# iptables -N inWall
```

```
# iptables -N backWall
```

## ● 从 A 到 B

```
# iptables -A inWall -p TCP -s 0.0.0.0/0 --dport 2000:3000  
-j \ allowed
```

```
# iptables -A inWall -p TCP -s 10.50.100.1 --dport 22 -j  
allowed
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (2)



清华大学  
Tsinghua University

## ● 从 B 回 A

```
# iptables -A backWall -p TCP -d 0.0.0.0/0 --sport  
2000:3000 \ -j allowed  
  
# iptables -A backWall -p TCP -d 10.50.100.1 --sport 22 -j  
\ allowed
```

# 依照上述的假设，应以下列的步骤 进行防火墙的设置 (3)



清华大学  
Tsinghua University

- 在 eth1 加上网络 A 的进入 Chain ,eth2 加上网络 B 的回去 Chain

```
# iptables -A FORWARD -p TCP -i eth1 -j inWall
```

```
# iptables -A FORWARD -p TCP -i eth2 -j backWall
```

- 拒绝掉 UDP 及 ICMP 数据包

```
# iptables -A FORWARD -p UDP -j DROP
```

```
# iptables -A FORWARD -p ICMP -j DROP
```

# 实验导引

实验 1 设置本机防火墙

实验 2 设置本机防火墙

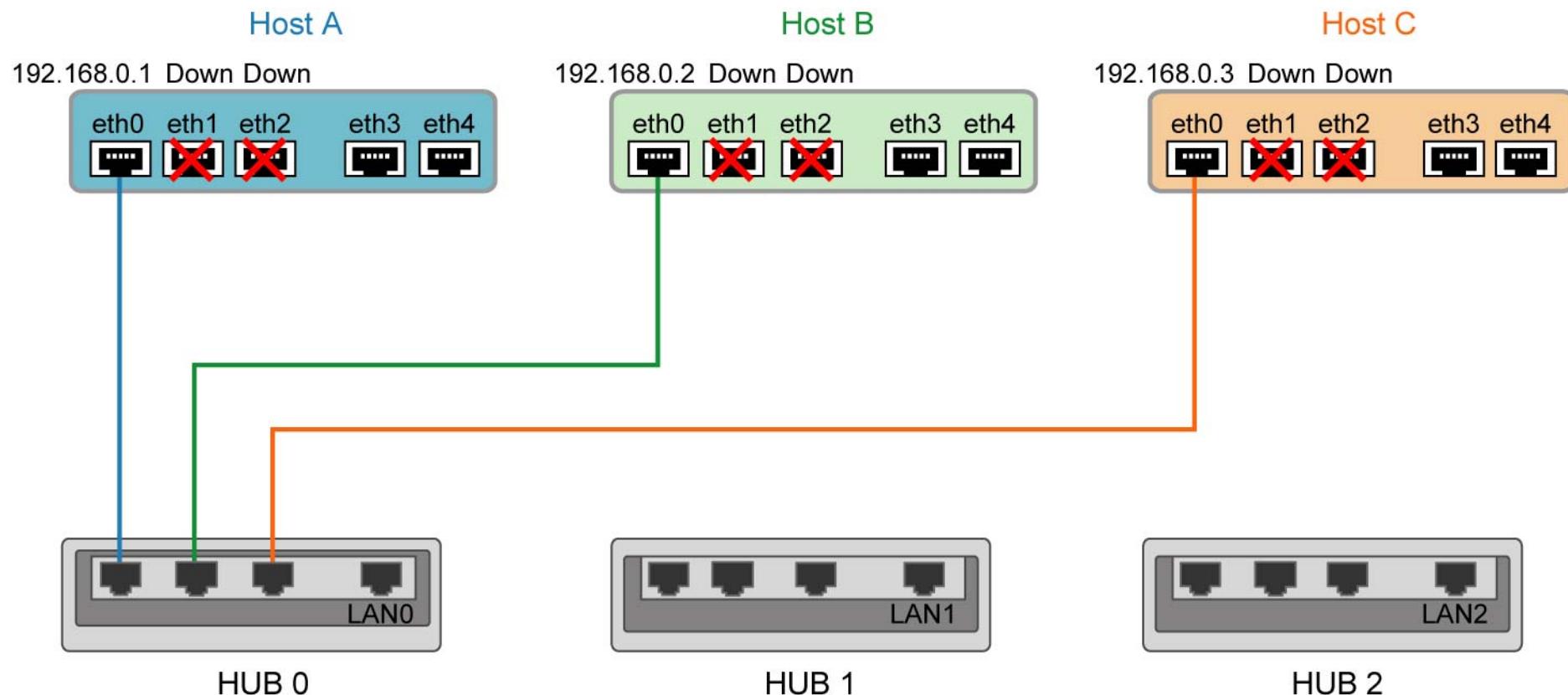
# 实验 1 设置本机防火墙



## ● 实验目的

- 了解如何设置 Linux 防火墙
- 熟悉 iptables 命令
  
- 注：本实验只适合在 SimpleNet 本机实操

# 实验架构图



本实验会变更网络默认值

Network Security Lab, Research Institute of Information Technology, Tsinghua University

# Step 1: 设定实验环境



- 设定 Host B 为本机防火墙，并以 Host A 和 Host C 分别尝试连接
- Host A:
  - ifconfig eth1 down
  - ifconfig eth2 down
- Host B:
  - ifconfig eth1 down
  - ifconfig eth2 down
- Host C:
  - ifconfig eth1 down

# Step 1: 设定实验环境（续）



- 开启 Ethereal，以观察下列步骤的网包
- Host A:
  - 开启 Ethereal，interface 选 eth0
- Host B:
  - 开启 Ethereal，interface 选 eth0
- Host C:
  - 开启 Ethereal，interface 选 eth0

# Step 1：设定实验环境（续）



- Host B: 在 NetGuru 本机操作以 Keyboard、Mouse、Monitor 连接，按 Ctrl+Alt+F1 切换到文字模式来设置防火墙，按 Ctrl+Alt+F2 切换回 X 画面
  - iptables -P INPUT DROP
  - iptables -P OUTPUT DROP
  - iptables -P FORWARD DROP
  - iptables -N allow\_tcp
  - iptables -N testChain
  - iptables -N rule\_tcp
  - iptables -N rule\_icmp

# Step 1: 设定实验环境（续）



清华大学  
Tsinghua University

- iptables -A allow\_tcp
- iptables -A allow\_tcp -p TCP --syn -j ACCEPT
- iptables -A allow\_tcp -p TCP -m state --state ESTABLISHED, RELATED -j ACCEPT
- iptables -A allow\_tcp -p TCP -j DROP
  
- iptables -A rule\_tcp -p TCP -s 0/0 --dport 23 -j allow\_tcp
- iptables -A rule\_tcp -p TCP -j RETURN
- iptables -A rule\_tcp -p TCP -s 0/0 --dport 80 -j allow\_tcp
- iptables -A rule\_icmp -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
- iptables -A INPUT -p TCP -j rule\_tcp
- iptables -A INPUT -p ICMP -j rule\_icmp
- iptables -A OUTPUT -p ALL -j ACCEPT

## Step 2: 进行连接测试



- Host B: 执行 iptables -v -L, 并观察其结果
  - iptables -v -L
- Host A: 测试防火墙的设置
  - 执行 telnet 192.168.0.2 是否能连上?
  - 执行 ping -c3 192.168.0.2 是否有响应?

## Step 2: 进行连接测试（续）



- Host B: 将rule\_icmp 清空加上新 rule
  - iptables -F rule\_icmp
  - iptables -A rule\_icmp -p ICMP -j REJECT --reject-with icmp-proto-unreachable
- Host A: 再测试防火墙的设置
  - 执行 ping -c3 192.168.0.2 是否有响应?
- 问题与讨论
  - 观察执行 ping -c3 192.168.0.2 的结果，新增 rule\_icmp 前后有何不同？
  - 观察 Ethereal 数据包，并解释以下规则的用意为何：
    - ◆ iptables -A rule\_icmp -p ICMP -j REJECT --reject-with icmp-proto-unreachable

## Step 2: 进行连接测试（续）



### ● Host A: 测试防火墙的 80 port (Web Server)

- 执行 telnet 192.168.0.2 80, 是否能连上?
- 观察 Ethereal 抓取的数据包

### ● 问题与讨论

- 为什么我们在防火墙中加入了以下规则之后仍然无法和 192.168.0.2 的 80 port 连接:
  - ◆ `iptables -A rule_tcp -p TCP -s 0/0 --dport 80 -j allow_tcp`

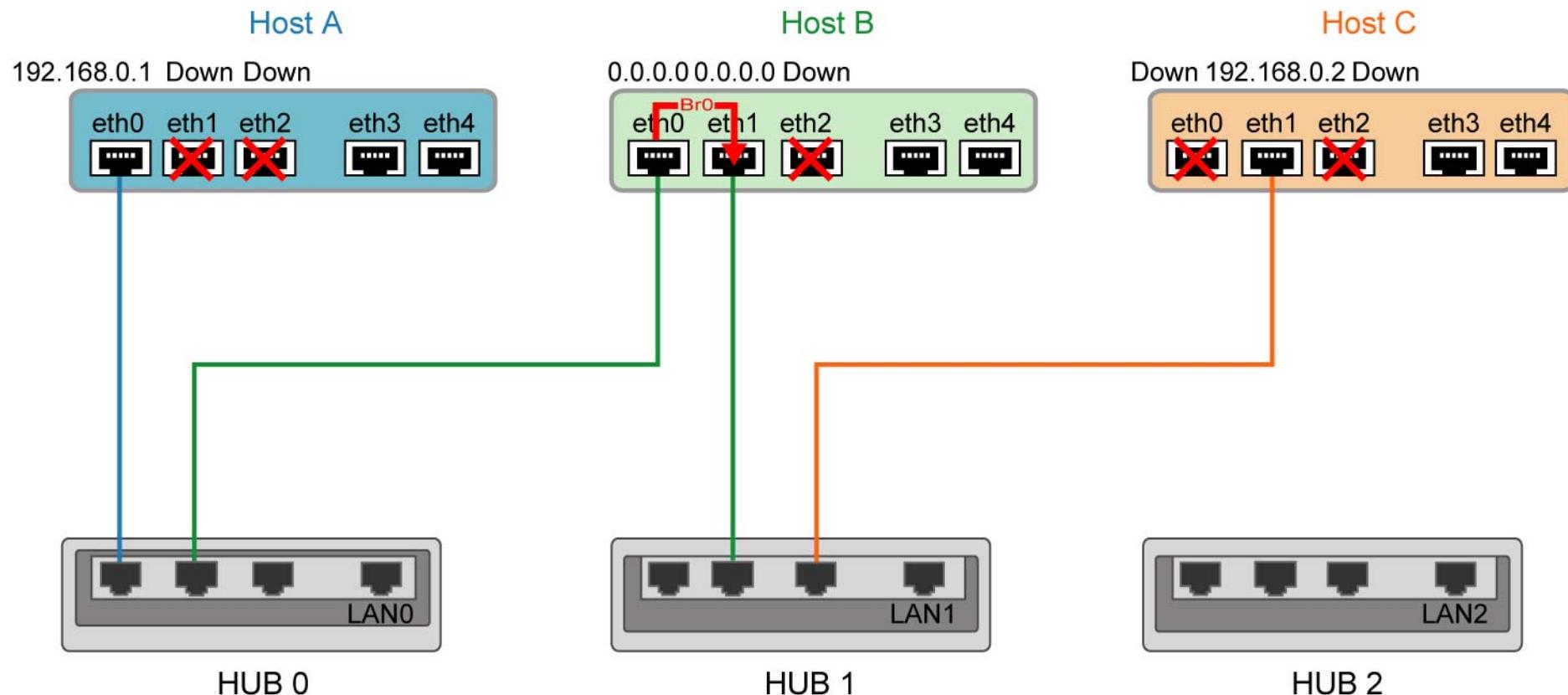
# 实验 2 设置独立防火墙



## ● 实验目的

- 了解如何设定 Linux 防火墙
- 熟悉 iptables 命令

# 实验架构图



本实验会变更网络默认值

Network Security Lab, Research Institute of Information Technology, Tsinghua University

# Step 1: 设定实验环境



- Host B 当成是独立防火墙，以硬件接口让 Host A 和 Host C 分隔为不同的网络
- Host A:
  - ifconfig eth1 down
  - ifconfig eth2 down
- Host B:
  - Ifconfig eth2 down
- Host C:
  - ifconfig eth0 down
  - ifconfig eth1 192.168.0.2 netmask 255.255.255.0
  - ifconfig eth2 down

# Step 1: 设定实验环境（续）



- 开启 Ethereal，以观察下列步骤的网包
- Host A:
  - 开启 Ethereal，interface 选 eth0
- Host B:
  - 开启 Ethereal，interface 选 eth0
  - 开启 Ethereal，interface 选 eth1
- Host C:
  - 开启 Ethereal，interface 选 eth1

# Step 1：设定实验环境（续）



- Host B: 在 NetGuru 本机操作以 Keyboard、Mouse、Monitor 连接，按 Ctrl+Alt+F1 切换到文字模式来设置防火墙，按 Ctrl+Alt+F2 切换回 X 画面
  - `iptables -P INPUT DROP`
  - `iptables -P FORWARD DROP`
  - `iptables -P OUTPUT DROP`

# Step 2: 进行连接测试



- Host A: 与 Host C 做 telnet 的连接
  - 执行 telnet 192.168.0.2 是否可连接?

## ● 问题与讨论

- 如果不能连接, 解释为什么?

# Step 3: 设定桥接



## ● Host B: 进行桥接并将原来的名称取消

- brctl addbr br0
- brctl stp br0 off
- brctl addif br0 eth0
- brctl addif br0 eth1
- ifconfig eth0 down
- ifconfig eth1 down
- ifconfig eth0 0.0.0.0 up
- ifconfig eth1 0.0.0.0 up
- ifconfig br0 up
- iptables -A FORWARD -p tcp --dport 23 -s 192.168.0.1 -j ACCEPT
- iptables -A FORWARD -p tcp --sport 23 -d 192.168.0.1 -j ACCEPT

# Step 4: 进行连接测试（续）



## ● Host A: 重复 Step 2

- 执行 telnet 192.168.0.2 是否可连接?
- 执行 ping -c3 192.168.0.2 是否可连接?

## ● 问题与讨论

- 执行 telnet 的结果与 Step 2 有何不同? 观察由 192.168.0.1 到 192.168.0.2 的数据包
- 执行 ping 的结果为何? 观察由 192.168.0.1 到 192.168.0.2 的数据包

# Step 5: 设置防火墙新规则



## ● Host B:

- `iptables -A FORWARD -p icmp -s 192.168.0.1 -i br0 -j ACCEPT`
- `iptables -A FORWARD -p icmp -d 192.168.0.1 -i br0 -j ACCEPT`
- `iptables -A FORWARD -p icmp -j DROP`

# Step 6: 进行连接测试（续）



清华大学  
Tsinghua University

- Host A: 与 Host C 做 ping 的连接
  - 执行 ping -c3 192.168.0.2 是否可连接?
- Host C: 与 Host A 做 ping 的连接
  - 执行 ping -c3 192.168.0.1 是否可连接?
- 问题与讨论



清华大学  
Tsinghua University

# Juniper NS-5GT

# Juniper 5GT



## ● NextScreen

# Juniper 5GT Wireless



- 基于IXP425网络处理器平台
- Juniper 5GT Wireless提供了防火墙/VPN 功能
- 采用高级别安全技术与WiFi相结合，融合企业级安全性和802.11 b/g 无线接入技术。
- 它运用无线安全区以及无线专用的验证和保密机制，能够为不同用户群设置适当级别的无线接入权限。
- 用户还能跨越企业和远程办事处站点维护一致的无线安全配置文件。
- 这些特性，再加上防火墙和VPN功能、管理简便性和优越的性价比，使该产品成为功能强大的集成解决方案。



## ● Network Processor 是什么？？？



*Send me your comments to*  
[zhenchen@csnet1.cs.tsinghua.edu.cn](mailto:zhenchen@csnet1.cs.tsinghua.edu.cn)

## 第四章 基于网络处理器的反蠕虫病毒系统

Chapter 4 AntiWorm NPU-based Parallel Bloom filters  
in Giga-Ethernet LAN

Dr. Jun Li

Dr. Zhen Chen

[{zhenchen, junl}@tsinghua.edu.cn](mailto:{zhenchen,junl}@tsinghua.edu.cn)

**Network Security Lab**  
**Research Institute of Information Technology**  
**Tsinghua University**

<http://security.riit.tsinghua.edu.cn>

# Outline



- Project Background
- Project Technique Characteristic
- Achievement and Contribution
- Project Proceeding Status
- Problems
- Future Work

# Project Overview



- Intel IXA University Program
- Basic idea
  - Using the flexibility and high performance of Network Processors to detect and scan IP packet (or TCP flow) to locate worm codes, and cut off their propagation.
- Objectives
  - Fast Content Scan Engine
  - An AntiWorm Prototype System
  - Implementation and performance evaluation based on IXP2400

# Brief Introduction of AntiWorm Project



清华大学  
Tsinghua University

## AntiWorm NPU-based Parallel Bloom filters in Giga-Ethernet LAN

- Scenarios : **Giga-Ethernet LAN**
- Purpose: **AntiWorm**
- Tool: **Intel IXP2400 NPU**
- Theory: **Parallel Bloom Filters**

# Worms



- Worm are malicious codes, which can be self-reproduced and propagate over a network, with or without human assistance.
  - RTM (Robert T. Morris) worm in 1988
  - CodeRed worm in 2001
  - Blaster worm in 2003
  - Sasser worm in 2004
  - SQL slammer worm recently
  - Witty worm recently

# Life Cycle of The Worm

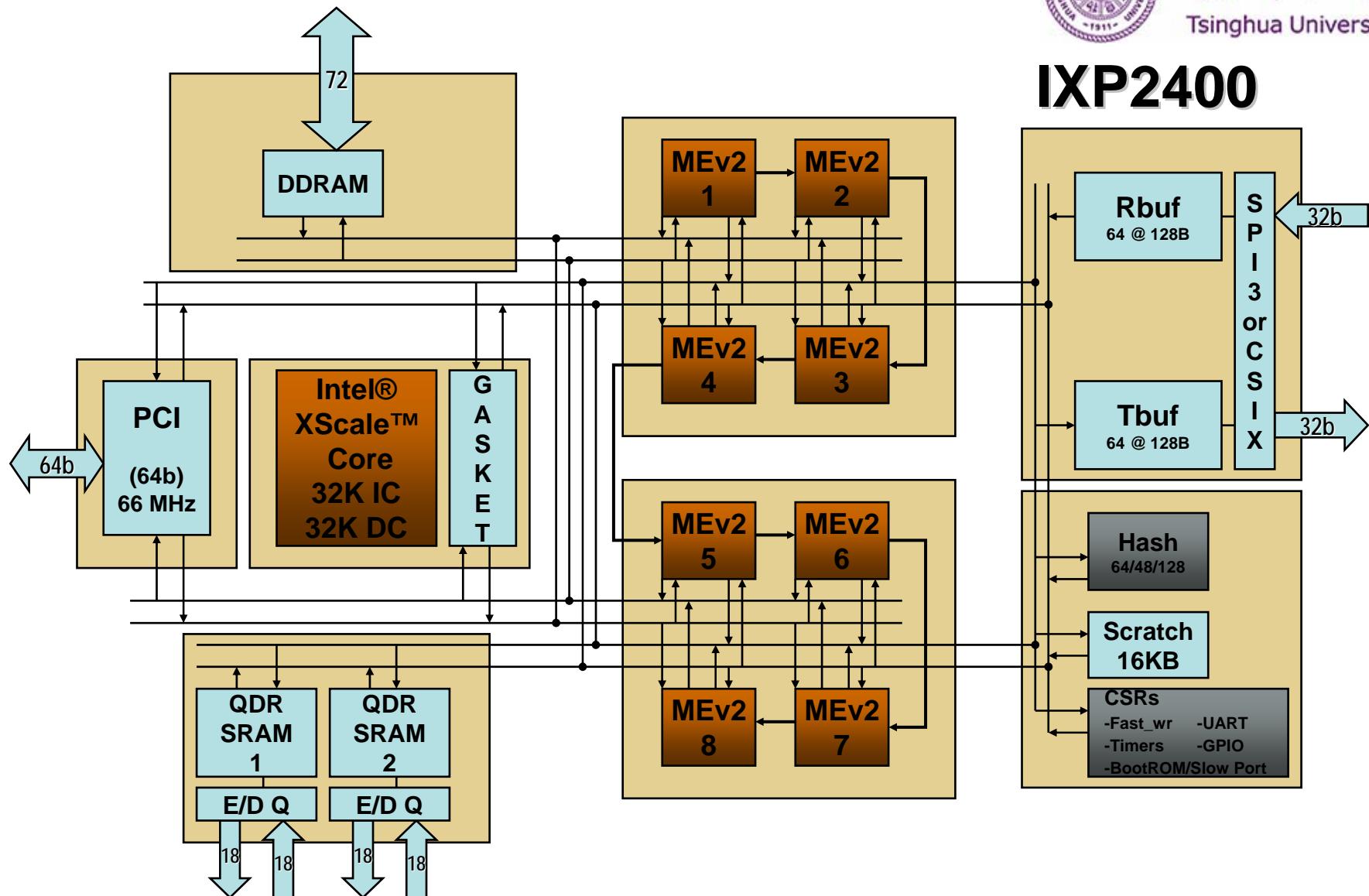


- Target discovery: the worm in the infected host tries to locate a vulnerable host
- Carrier: First, make the new host prepare to receive the worm code. Then, transfer the worm code to the new host
- Activation: change the new host's registry, and make it to reference the worm code. Then the process repeats.

# Brief introduction of IXP2400



**IXP2400**

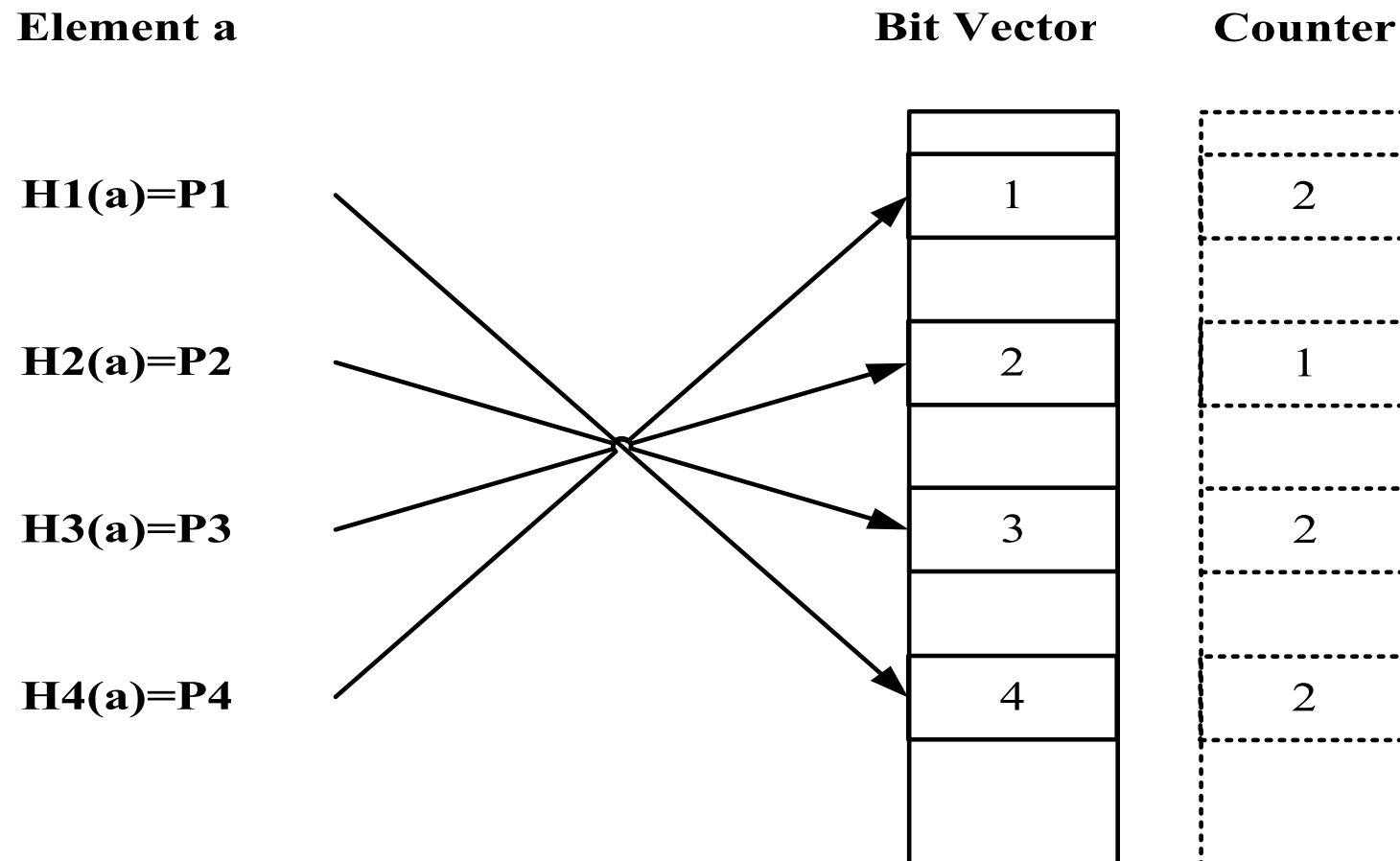


# Searching Method: Bloom filter



- A Bloom filter is a method for representing a set of elements to support membership queries.
- It was invented by Burton Bloom in 1970.
- For scalability reason, we will use Counting Bloom filters.

# (Counting) Bloom Filter



# Novelty in the Project



- Using the flexibility and high performance of Network Processors to detect and scan IP packet (or TCP flow) to locate worm codes, and cut off their propagation
- NPU-Based implementation can live update the worm's signature flexibly to keep pace with fast evolution of worms,
- Parallel Bloom filters is extensible to the amount of signature
- Parallel Bloom Filters technique based Content Scanning Engine for AntiWorm

# Outline



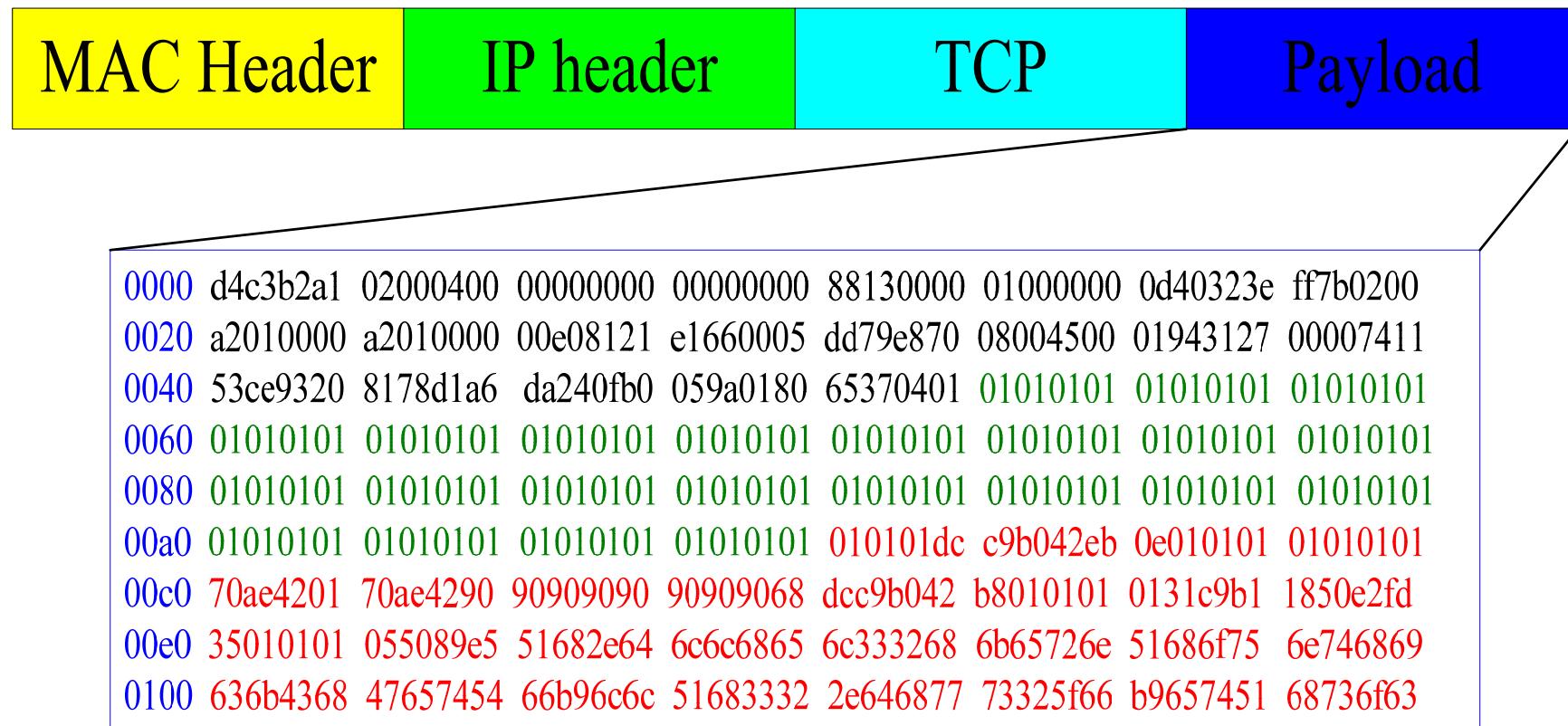
- Project Background
- Project Technique Characteristic
- Achievement and Contribution
- Project Proceeding Status
- Problems
- Future Work

# Worm Signature Generation



- Fixed length segment(16Bytes) can be extracted from the Binary of the worm code as Signature
- The method commonly used in other AV software, e.g. ClamAV .
- There are many other signature generated methods, e.g. behavior based detection.

# SQL Slammer Worm Packet Payload



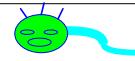
# Collect Network Traffic



清华大学  
Tsinghua University

TCP Header

<Http://qos.cs.tsinghua.edu.cn/~networkporcessor>



TCP Header

its propagation by using Network Processor Based Parallel

TCP Header



we are dedicated to scan worm and cutoff

TCP Header

Bloom Filters techniques, a prototype Is Implemented by our team

Figure 1. TCP flow packet fragment and out of order.

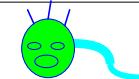
# Reorder Network Traffic



清华大学  
Tsinghua University

TCP Header

<Http://qos.cs.tsinghua.edu.cn/~networkporcessor>



TCP Header



we are dedicated to scan worm and cutoff

TCP Header

its propagation by using Intel IXP Network Processor Based Parallel

TCP Header

Bloom Filters techniques. Our team has Implemented a prototype.

Figure 2. Reorder the packets In the same TCP flow.

# Retrieval and Scanning

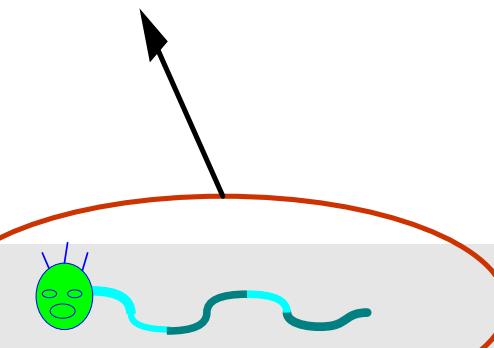


清华大学  
Tsinghua University

Now we capture It!!!!

[Http://qos.cs.tsinghua.edu.cn/~networkprocessor](http://qos.cs.tsinghua.edu.cn/~networkprocessor)

We are dedicated to scan worm and cutoff its propagation by using Intel IXP Network Processor based Parallel Bloom Filters techniques. Our team has implemented a prototype by using IXP2400 Network Processor. Some assault of real worms are simulated...



A small diagram showing a green worm-like creature with blue antennae moving along a wavy teal line, which represents a network path or propagation route. A red oval encircles the worm and the line, and a black arrow points upwards from the top right corner of the oval towards the text above.

Figure 3. Retrieval Flow Contents and Scan.

# Response to the Capture of Worm



清华大学  
Tsinghua University

- Passive response
  - Alarm and Log into file
  - Drop the packet related to the Worm
- Proactive response
  - TCP kill
  - Block all packet within the same flow

# Outline



- Project Background
- Project Principles
- Achievement and Contribution
- Project Proceeding Status
- Problems
- Future Work

# Achievement and Contribution



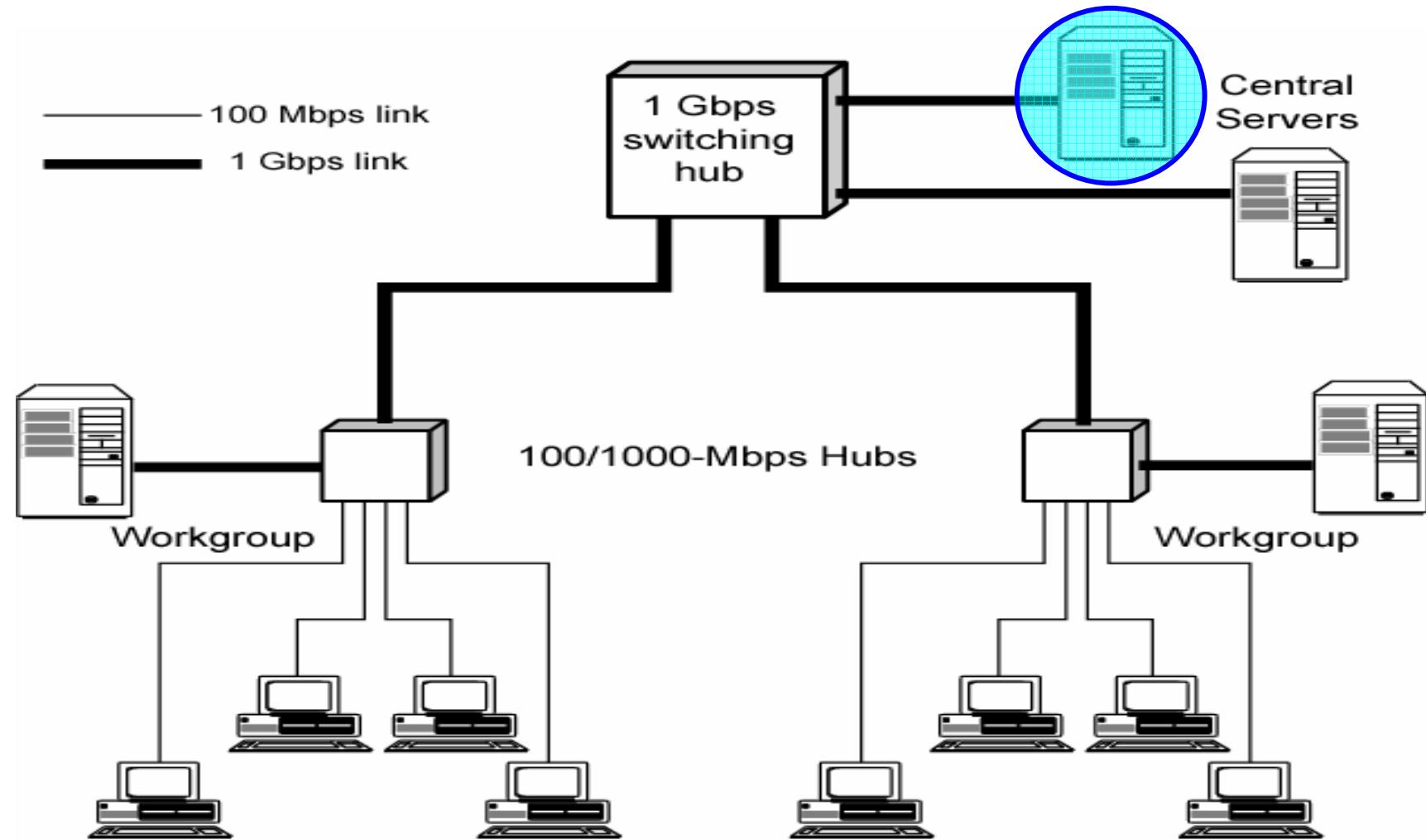
清华大学  
Tsinghua University

- Implemented two prototype AntiWorm systems
  - IP Packet Scanning system
    - ◆ Wide Range, not very precise
  - TCP flow Scanning system
    - ◆ TCP constrained, more precise
- Two software package:
  - WormDetector 1.0
  - TCPScanner 1.0
- Parallel Bloom Filters based Content Scanning Engine
- 2 Papers and Technique Reports

# AntiWorm system positioned in Gigabit LAN



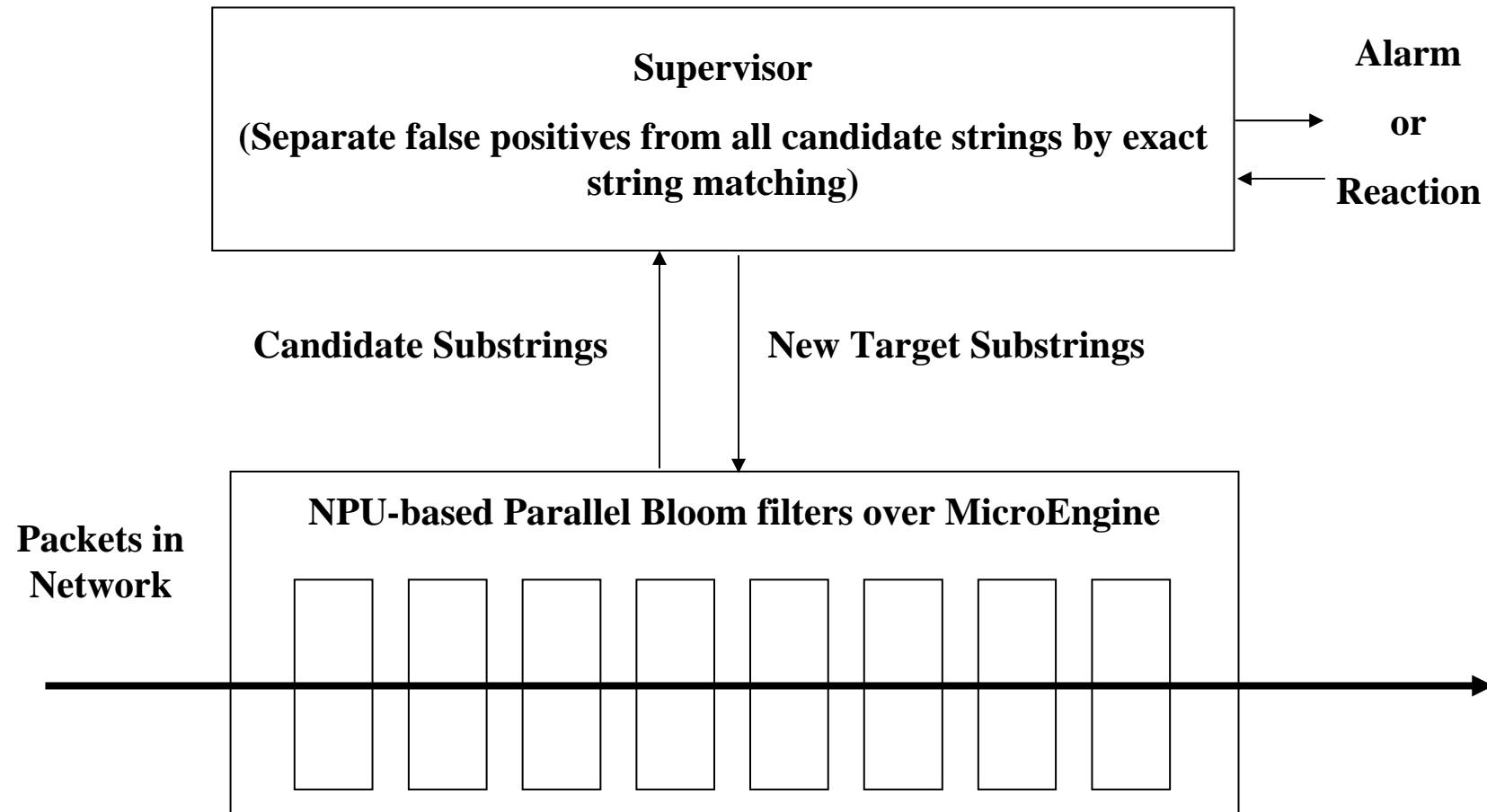
清华大学  
Tsinghua University



# The Structure of the Prototype AntiWorm System

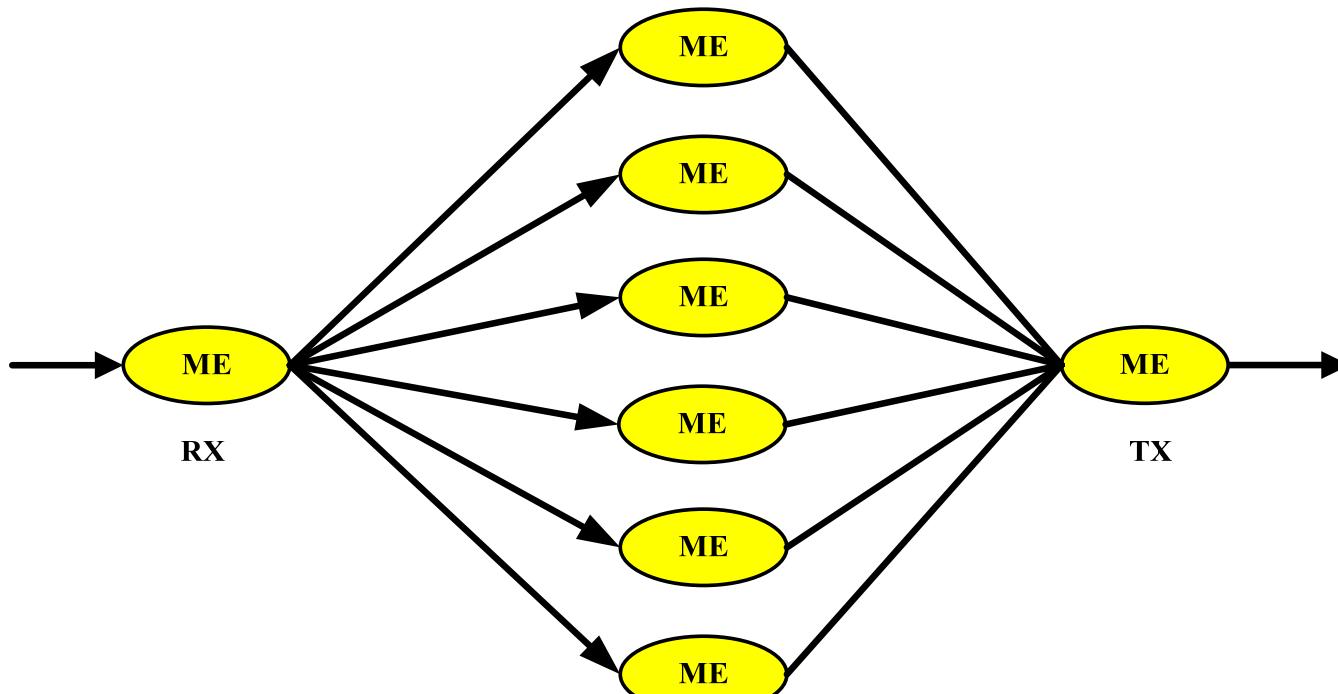


清华大学  
Tsinghua University



## IP Packet Scanning System

# Physical Configuration



Parallelize the MEs for Bloom Filters

# Working Picture



清华大学  
Tsinghua University

enp2611\_static\_fwd - Developer Workbench - [(0:1) Thread8 - PC: 321 (Active)]

File Edit View Project Build Debug Simulation Hardware Tools Window Help

Microengine 0:1 Thread8

```
local_csr_wr[same_me_signal, 10000!sig_next_val]
:    nop
:    nop
:    nop
w2#: :label for debug
ctx_arb[10000!sig_scr_get]
alu[--, $10000!rdata0, -, 0]
beq[ring_empty#]
:    modify[$rdata0]
m949#:
    bloomfilter1($10000!rdata0,$10000!rdata2)
    alu_shf[10000!port, 0xf, and, $10000!rdata4, >>16]
    alu[10000!port_out, 10000!port, +, 1]
    .if(port_out == 4)
    alu[--, 10000!port_out, -, 4]
    bne[1044_01#]
        alu[10000!port_out, --, b, 0]
```

FileView ThreadView InfoView

Add Watch... Refresh 17920931

| Name             | Value                    | Description |
|------------------|--------------------------|-------------|
| 10067!offset     | 304                      | GPR -       |
| 10067!dram_...   | 266240                   | GPR -       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| + local_mem[2... | Array of 4 32-bit values | Micro       |
| +\$10067!tem...  | dram :                   |             |
| + 10067!test     | Array of 4 32-bit values | GPR A       |
| ...              |                          |             |

Chip <unnamed> Options... Save Stats To File...

| Traffic Interface         | Rx buffer fullness | Ix buffer fullness | Packets received | Receive rate | Packets sent | Transmit rate |
|---------------------------|--------------------|--------------------|------------------|--------------|--------------|---------------|
| Device ID 0 (x32MPHY4 Rx) |                    |                    | 13224            | 685.2525     | n/a          | n/a           |
| Port 0                    | 65536 [100%]       | n/a                | 3306             | 171.3091     | n/a          | n/a           |
| Port 1                    | 65536 [100%]       | n/a                | 3306             | 171.3103     | n/a          | n/a           |
| Port 2                    | 65536 [100%]       | n/a                | 3306             | 171.3252     | n/a          | n/a           |
| Port 3                    | 65536 [100%]       | n/a                | 3306             | 171.3079     | n/a          | n/a           |
| Device ID 1 (x32MPHY4 Tx) |                    |                    | n/a              | n/a          | 13007        | 675.0683      |
| Port 0                    | n/a                | 0 [0%]             | n/a              | n/a          | 3251         | 168.7168      |
| Port 1                    | n/a                | 0 [0%]             | n/a              | n/a          | 3254         | 168.9116      |
| Port 2                    | n/a                | 0 [0%]             | n/a              | n/a          | 3251         | 168.7030      |
| Port 3                    | n/a                | 0 [0%]             | n/a              | n/a          | 3251         | 168.7369      |

For Help, select Help->Help Topics on the main menu IXP2400 BO uEng: 17920931

# Throughput (un optimized)



清华大学  
Tsinghua University

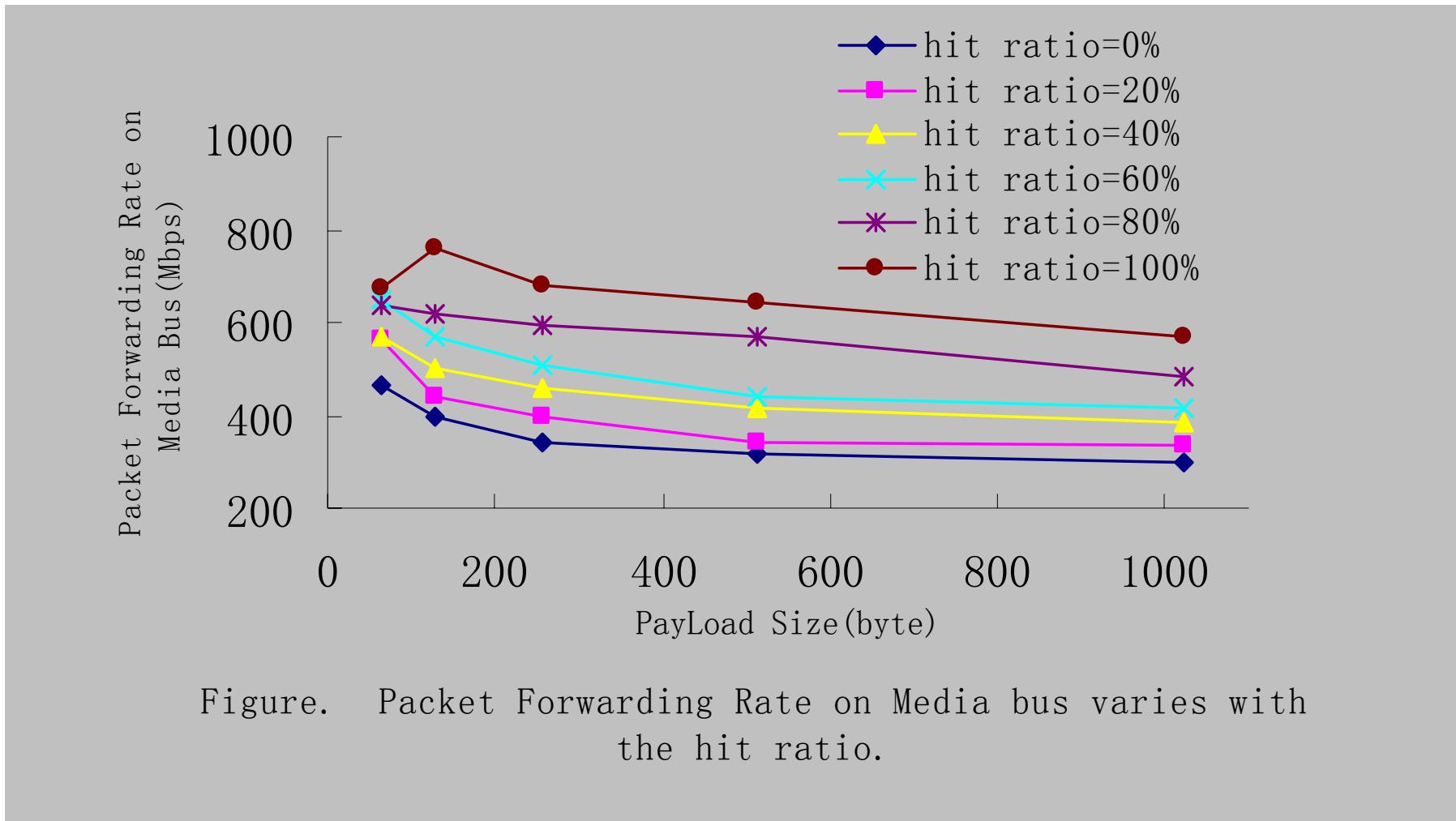


Figure. Packet Forwarding Rate on Media bus varies with the hit ratio.

# Delay in Worst Case

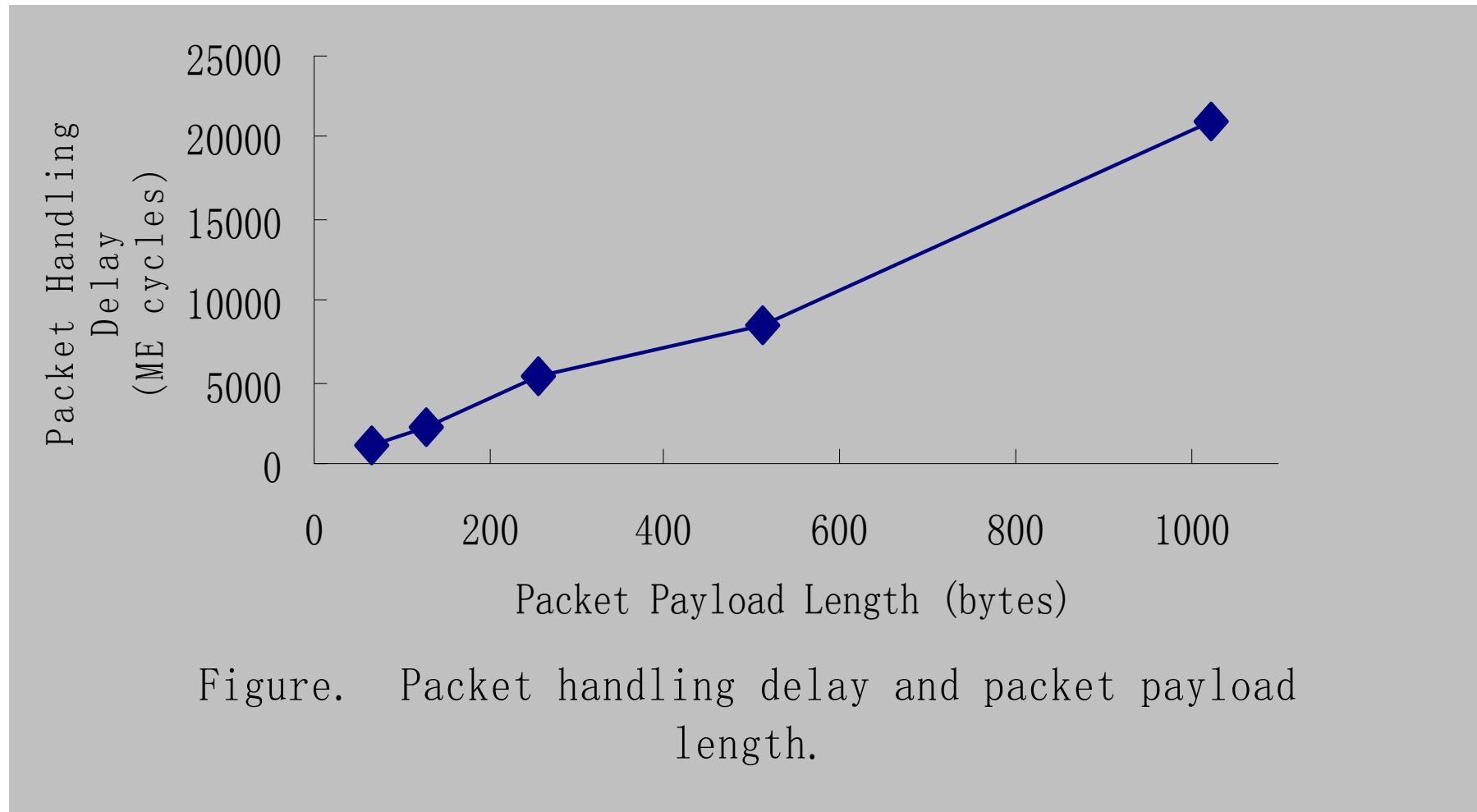
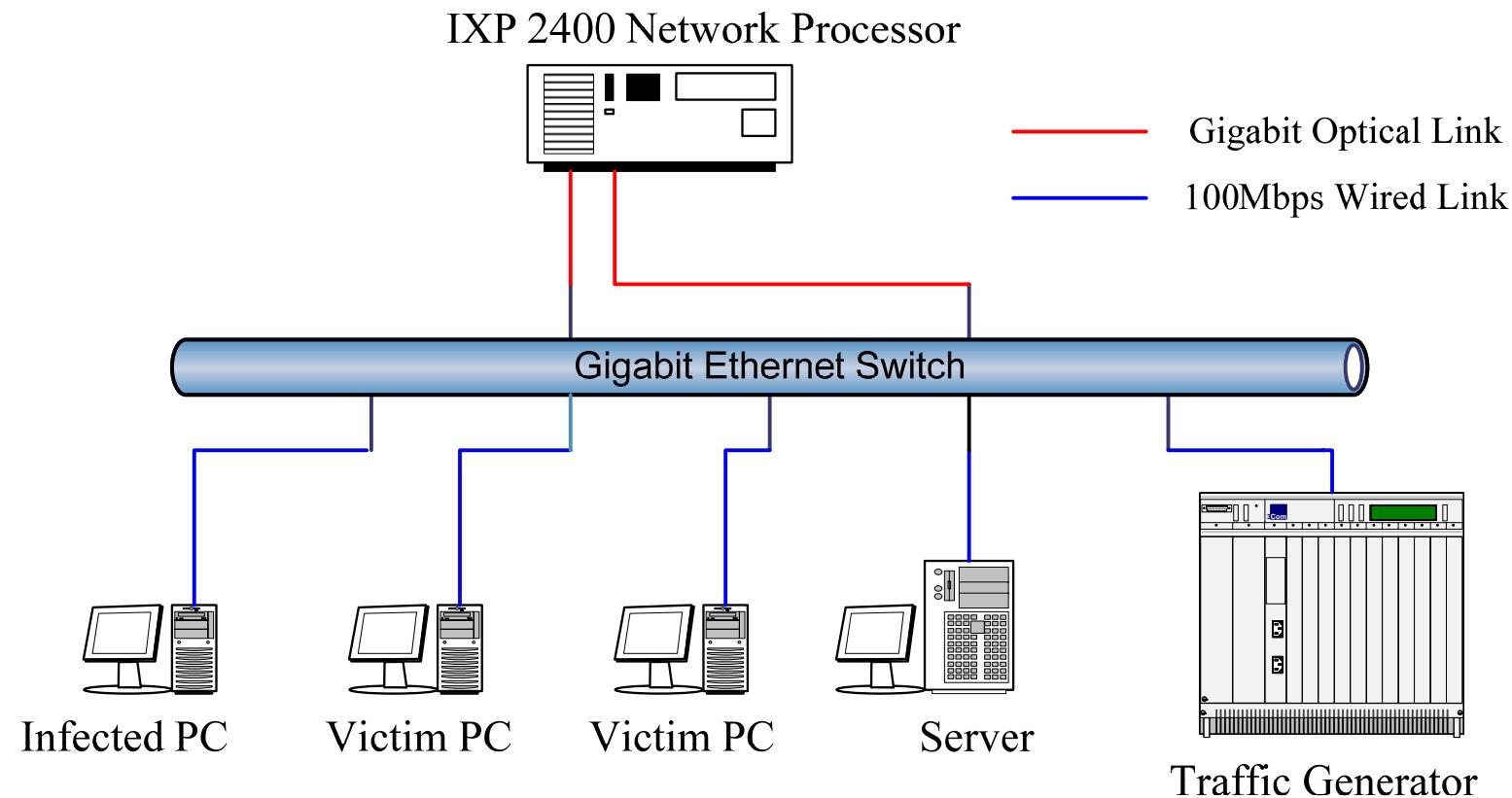


Figure. Packet handling delay and packet payload length.

# Gigabit Ethernet Test Environments



# Real Test Network Environment



# Real Worm Assault



- Benign executable of Blaster Worm in Isolated Environments
- Other Buffer Overflow Worms

# Problems and Solutions



- If a signature is broken, and carried by two consecutive packets, how to locate it?  
----😊 Flow based content processing
  - How to identify a Worm's outbreak?  
---- 😊 Abnormal Traffic Measurement and Alarm
  - The research continues, here we go
- AntiWorm NPU-based Parallel Bloom filters for TCP/IP Content Processing in Giga-Ethernet**

# TCP Flow Scanning System

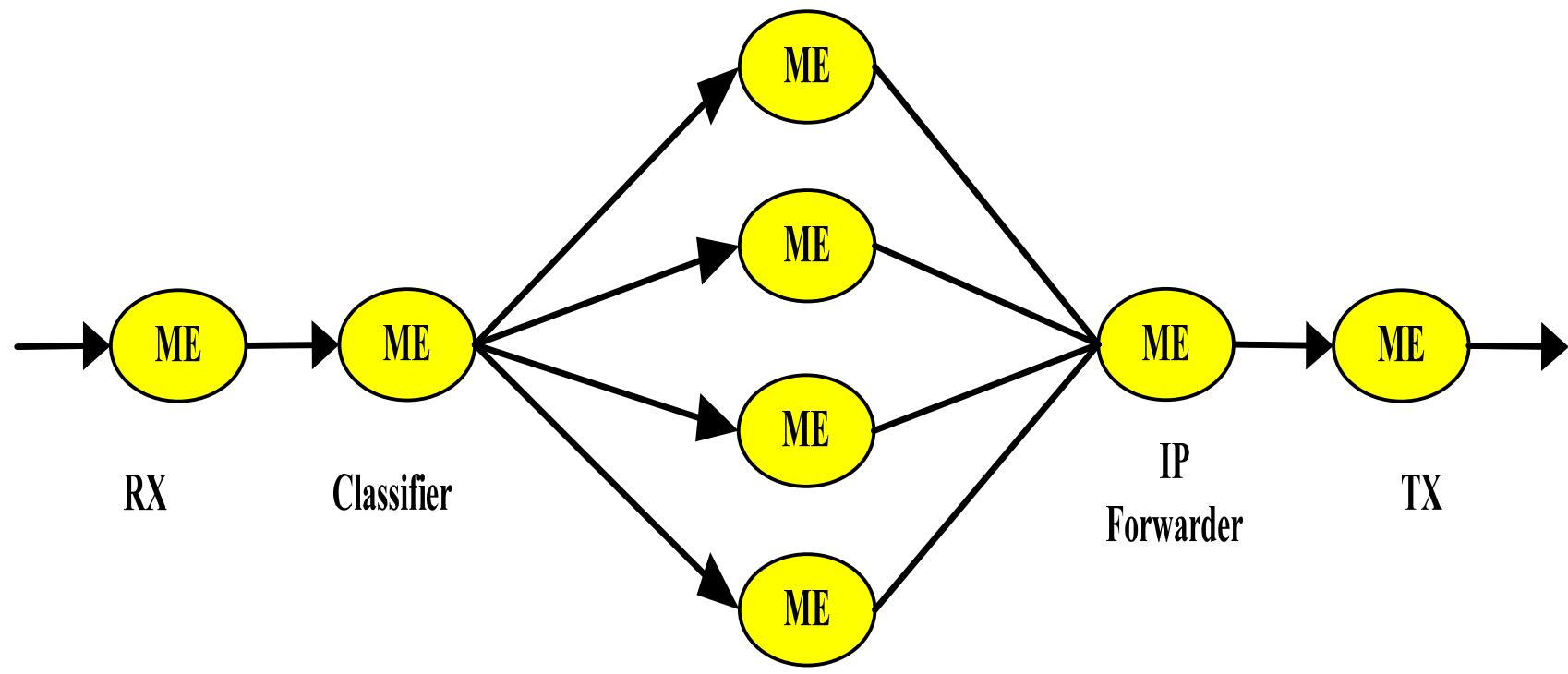
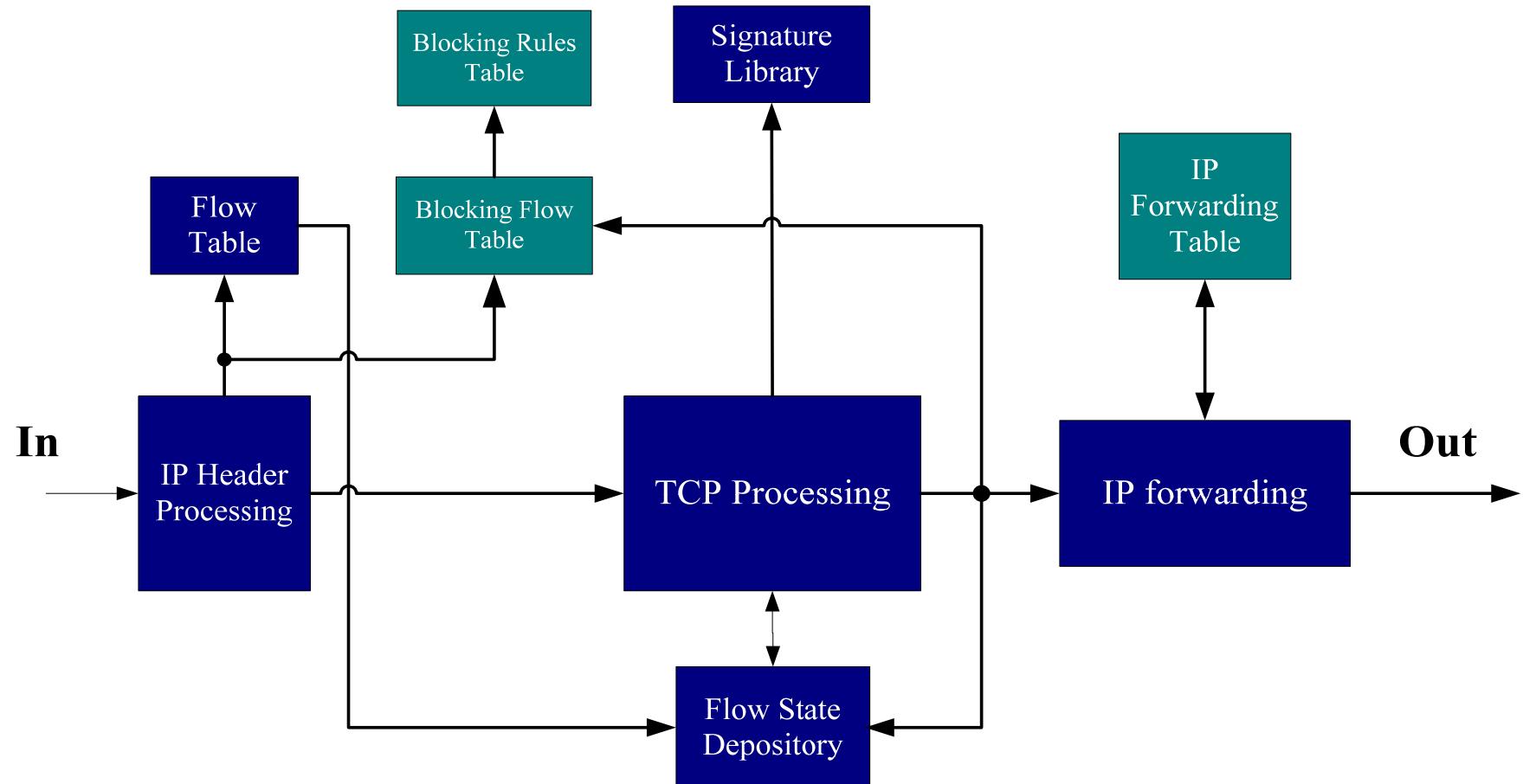
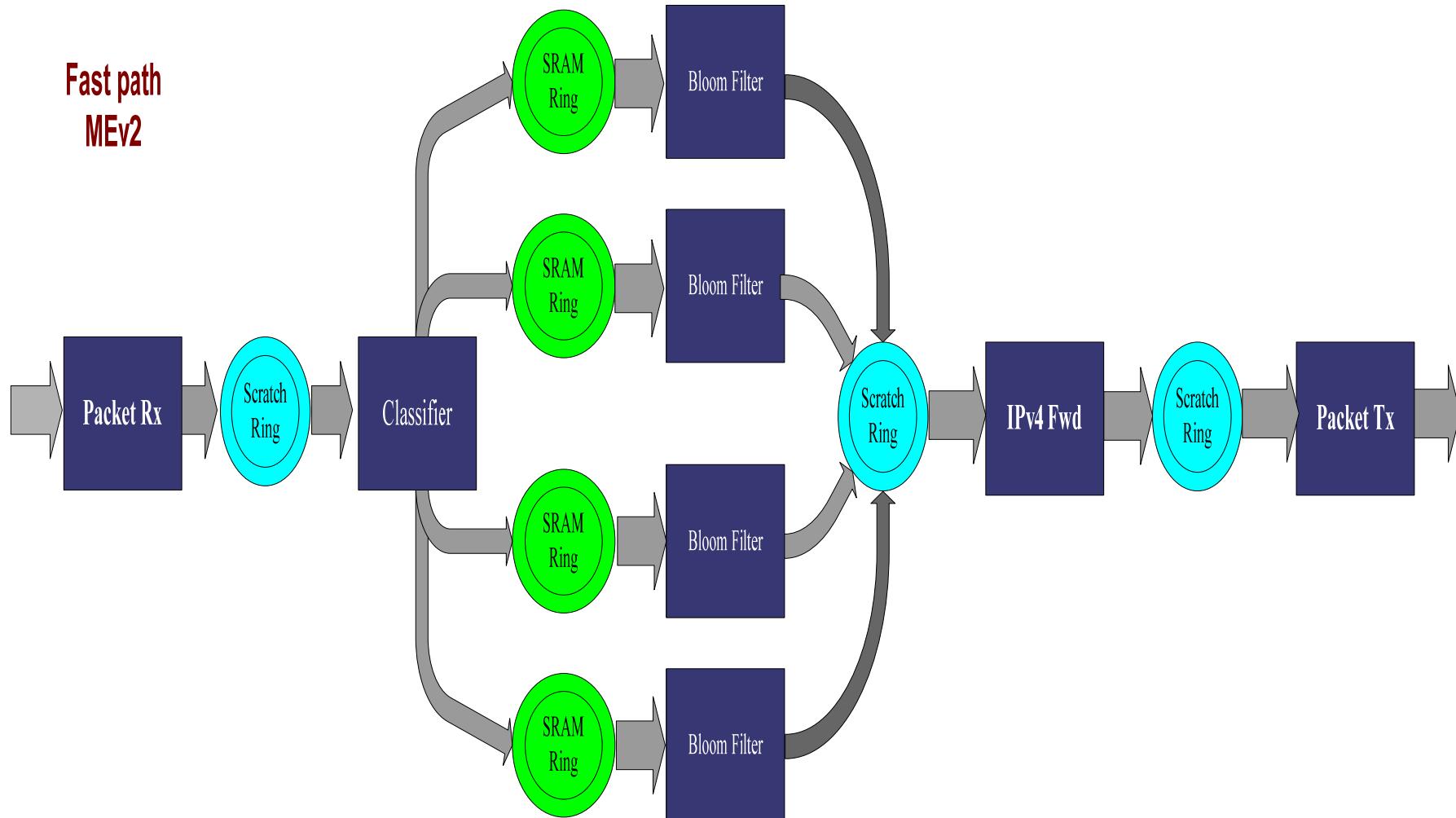


Figure . ME Allocation in TCP Flow Scanning System

# Logical Function Blocks in System



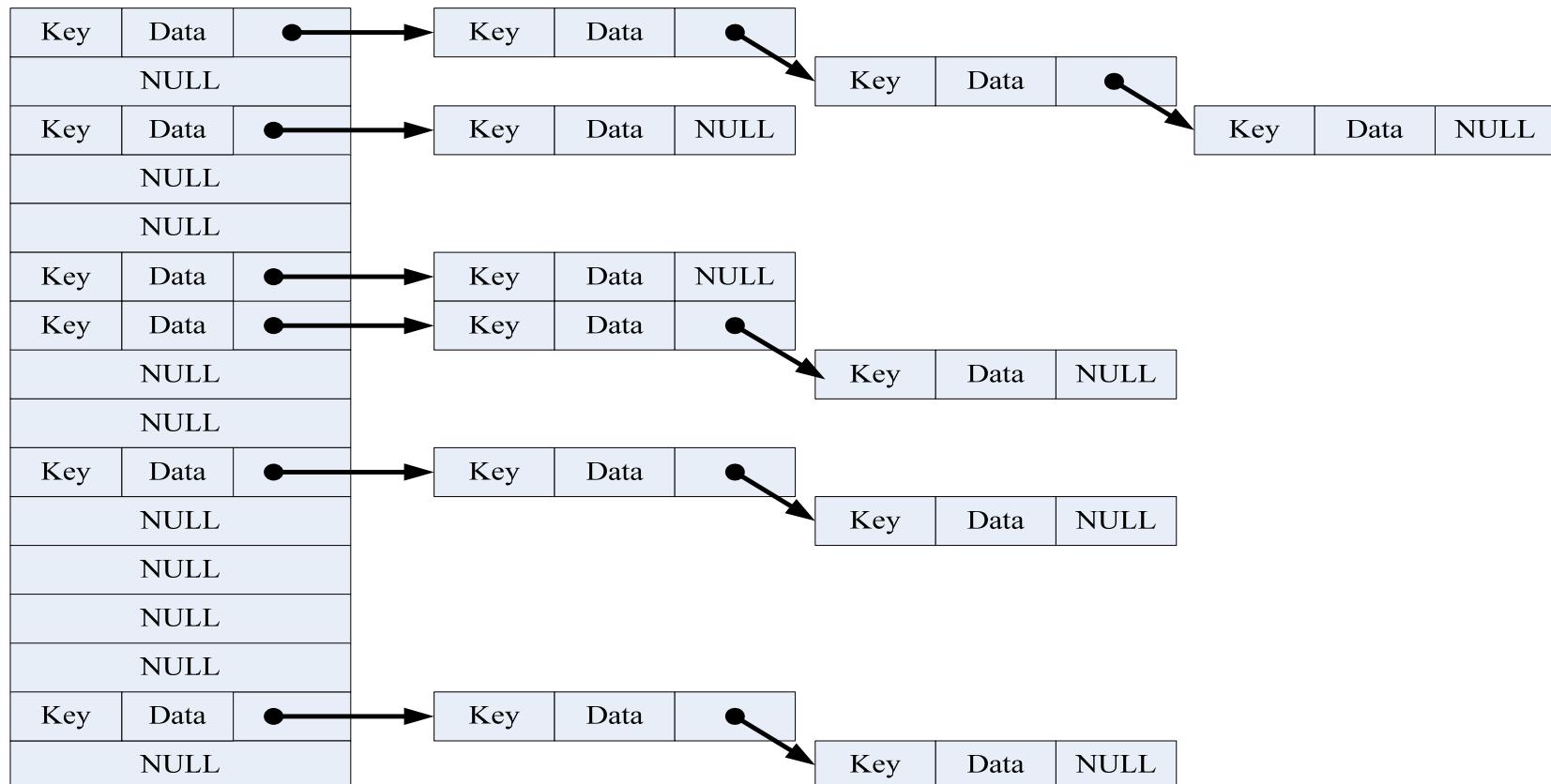
# Physical Function Blocks of TCP flow Content Processing System



# Flow Table Store (Hash Table)



清华大学  
Tsinghua University

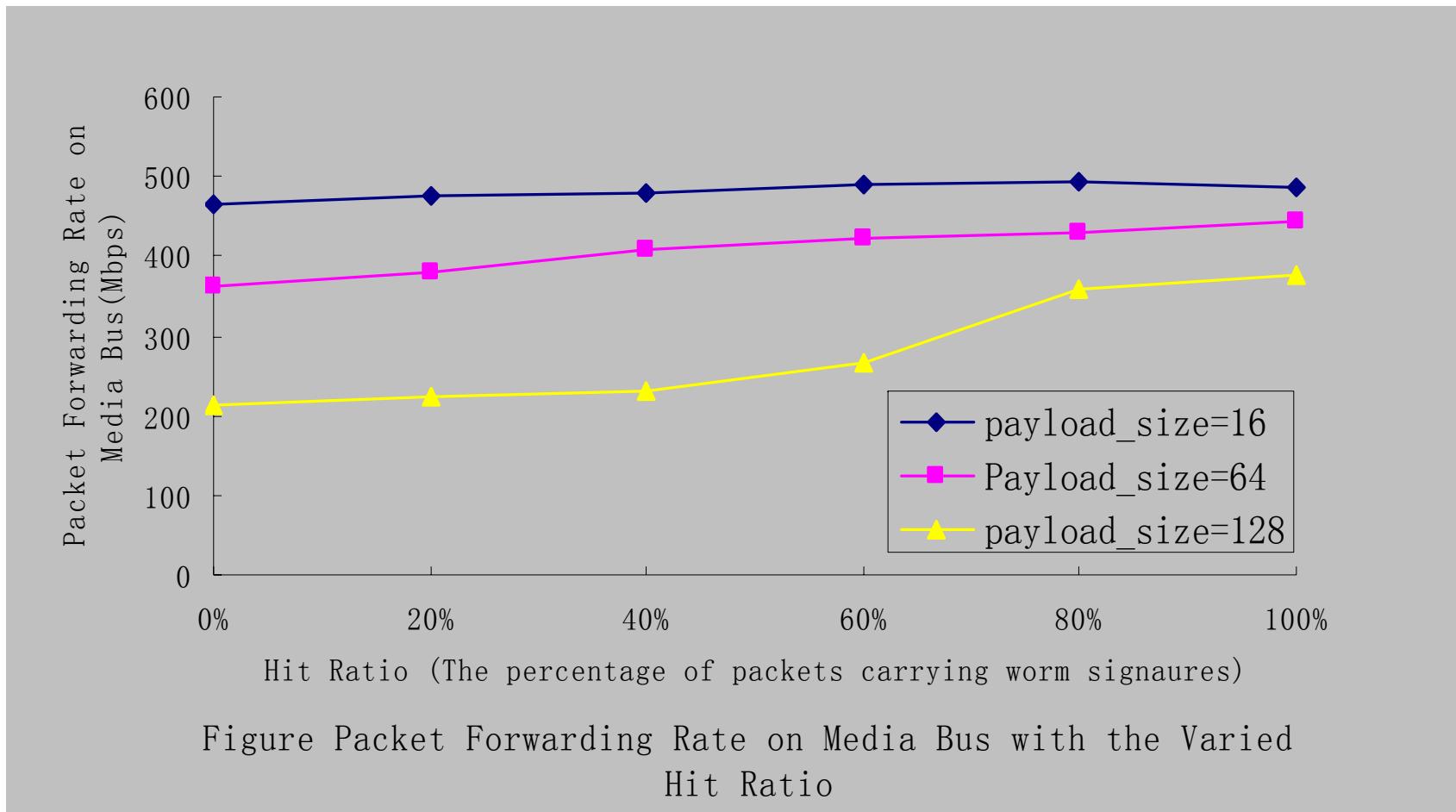


# The TCP Flow State Record (LSB-MSB)

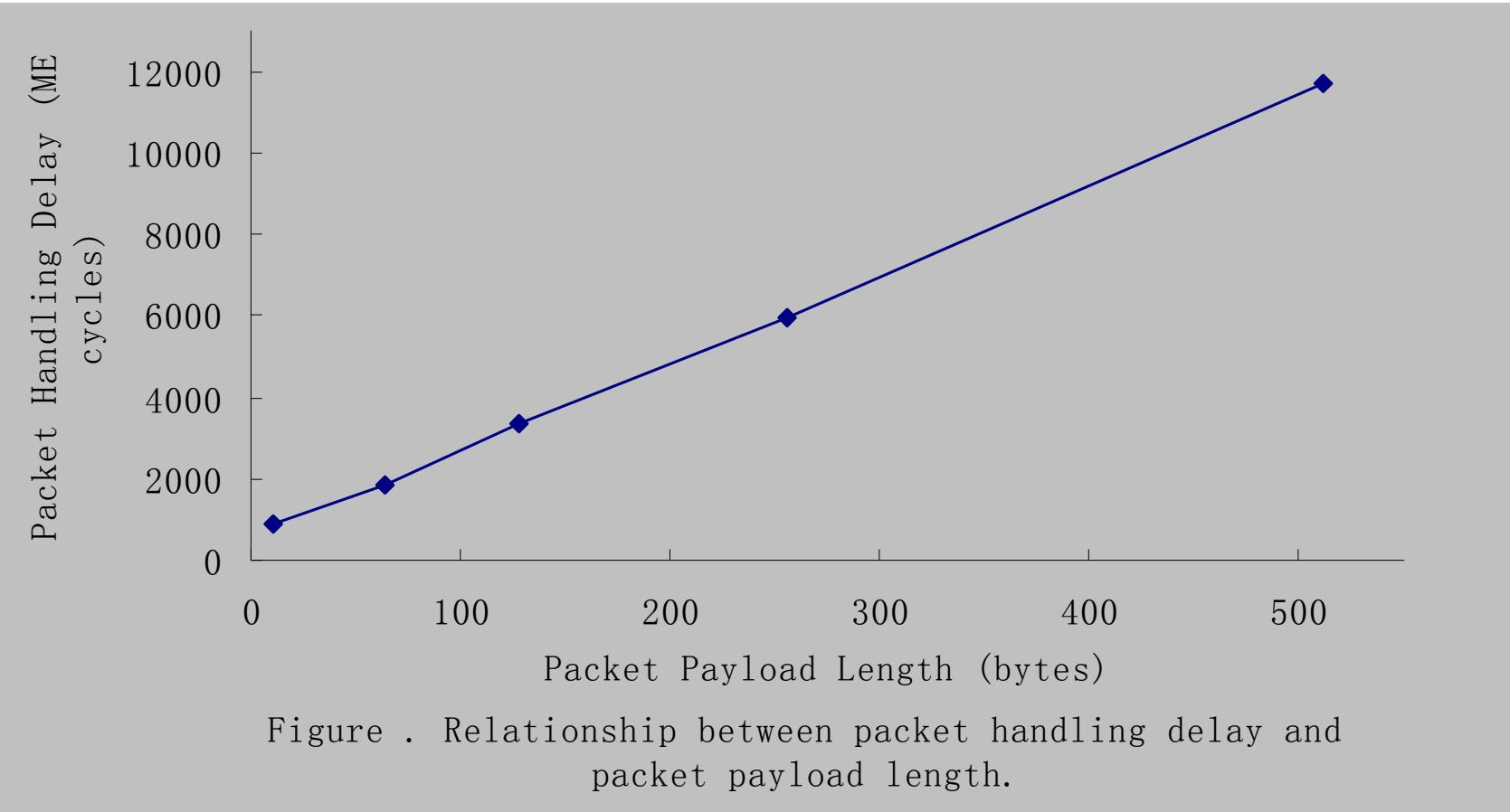


| LW | Bits | Size | Field             | Description                                   |
|----|------|------|-------------------|-----------------------------------------------|
| 0  | 31:0 | 32   | Hash Value        | The hash value of five tuples                 |
| 1  | 31:0 | 32   | Next Flow Pointer | Address of the next Flow state in DRAM        |
| 2  | 31:0 | 32   | Source IP address | Source IP address                             |
| 3  | 31:0 | 32   | Dest IP address   | Destination IP address                        |
| 4  | 31:0 | 32   | Port              | Source port and destination port              |
| 5  | 31:0 | 32   | Protocol          | Protocol                                      |
| 6  | 31:0 | 32   | Sequence Number   | TCP packet sequence number                    |
| 7  | 31:0 | 32   | Payload offset    | Source and Destination Port                   |
| 8  | 31:0 | 32   | CODE BITS         | Some Flags for Packet handling                |
| 9  | 31:0 | 32   | Time Stamp        | The time mark for the arrival packet (update) |
| 10 | 31:0 | 32   | Blocking Flag     | Blocking sequence number                      |
| 11 | 31:0 | 32   | Remaining length  | Index the last remaining byte length          |
| 12 | 31:0 | 32   | 1st word          | The first word of the last packet in flow     |
| 13 | 31:0 | 32   | 2nd word          | The second word of the last packet in flow    |
| 14 | 31:0 | 32   | 3rd word          | The third word of the last packet in flow     |
| 15 | 31:0 | 32   | 4th word          | The fourth word of the last packet in flow    |

# Throughput (un optimized)



# Delay in Worst Case



# Real Environments



# Defense In Depth



- Defense In Depth
  - Deep packet inspection

## ● Proactive Defense

- Without any knowledge of new breakup Worm
- How to automatically detects new worm in real time by monitoring all traffic on a network
- Some new techniques for Worm Detection, e.g., Traffic measurement based Abnormal traffic detection

# Abnormonitor



- Monitor the abnormal network activity, which is possibly a worm scan.
- For example:
  - A large number of flow from a single unknown source IP;
  - A large number of flow to the same destination port, especially those well-known dangerous ports;
  - A large number of ICMP Destination Unreachable Message, which possibly means failed scan connections

# Abnormonitor (cont.)



- Strongpoint:

- Detect worms at the Target Discovery stage;

- Detect any kind of fast-scan worm, even new worms;

- Light computation requirement.

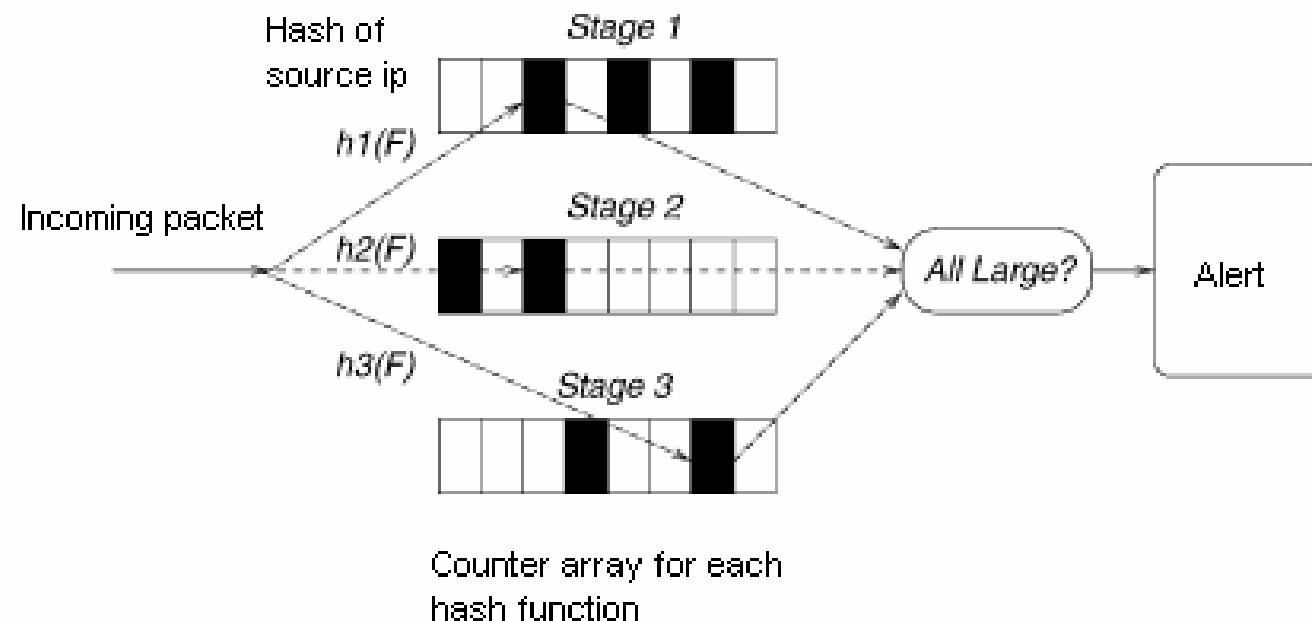
- Weak point:

- May cause many false alarms.

# Abnormonitor (cont.2)



## ● Example:



## ● Source IP Check

# Outline



- Project Background
- Project Principles
- Achievement and Contribution
- Project Proceeding Status
- Problems
- Future Work

# Research Progress 2004.10.1-2005.3.31



清华大学  
Tsinghua University

- A Scanning Engine based on Parallel Bloom Filters
- Worm Detection System based on IXP 2400 Network Processors
- A Novel Fast Hash Function Based on MD5
- Software package WormDetector 1.0 is deliverable, implemented in micro code

# Research Progress(2005.4.1-2005.9.31)



清华大学  
Tsinghua University

- TCP/IP Content Processing Engine
- Flow based scanning (stateful )
- Software package TCPScanner 1.0 is deliverable,  
implemented in micro code

# Outline



- Project Background
- Project Principles
- Project Proceeding Status
- Achievement and Contribution
- Problems
- Future Work

# Problems



- Some Performance Issues encountered in the implementation
- There are still many headroom left to lift the performance issues

# Outline



- Project Background
- Project Principles
- Project Proceeding Status
- Achievement and Contribution
- Problems
- Future Work

# Future Work



- Target to deliver a industry level AntiWorm function block.
- A more advanced string match algorithm, combined BM and Hash method to identify the string
- Hash-BM算法

# 可信计算(Trusted Computing)



- 1999年10月，Compaq、HP、IBM、Intel和Microsoft公司共同成立TCPA（Trusted Computing Platform Alliance，可信计算平台联盟）。2003年4月，TCPA中的AMD、HP、IBM、Intel和Microsoft对外宣布，将TCPA重新改组，更名为TCG（Trusted Computing Group，可信计算组织），目前该组织已有约90名全球性会员。
- TCPA首次提出了可信计算平台的概念，其基本思想是通过硬件和软件的标准化来抵御软件攻击和保护数据的安全性。它提出在当前的计算平台中加入新的硬件和软件来支持可信计算，计划首先在计算机中加入基于TPM（Trusted Platform Module，可信平台模块）的芯片，随后逐步将该芯片的功能直接集成到CPU、显卡、硬盘、BIOS等硬件设备中。可信平台模块具有安全存储和加密功能，使非法用户无法对其内部的数据进行更改，确保计算机始终遵守TCPA规范的要求。

# 可信网络连接(Trusted Network Connect)



- TCG提出的可信网络连接（Trusted Network Connect, TNC）主要针对当前网络的安全性问题，它是一套开放的网络安全解决方案架构。该架构指导网络管理员实施一定的安全策略，以确保连接到网络的终端主机是安全的。为此TCG成立了可信网络连接组TNCWG，专门负责制订可信网络连接规范。TNCG在2005年5月完成可信网络连接规范1.0版本，在2006年中推出了1.1版本，在2007年初又推出了1.2版本。目前参与制订该规范成员涵盖了绝大多数著名网络安全公司包括Juniper Networks、3Com公司、Symantec、趋势科技Trend、Verisign、Sygate、Zone Labs等，已经成为事实上的网络访问控制标准。

# Trusted Computing with NP



- IXP2350 based Trust Network Connect System

# About Trusted Network Connect (TNC)



清华大学  
Tsinghua University

## ➤ Definition

- ❖ Define and promote an open solution architecture that will enable network administrators to enforce security policies for endpoint host connections to their corporate networks.

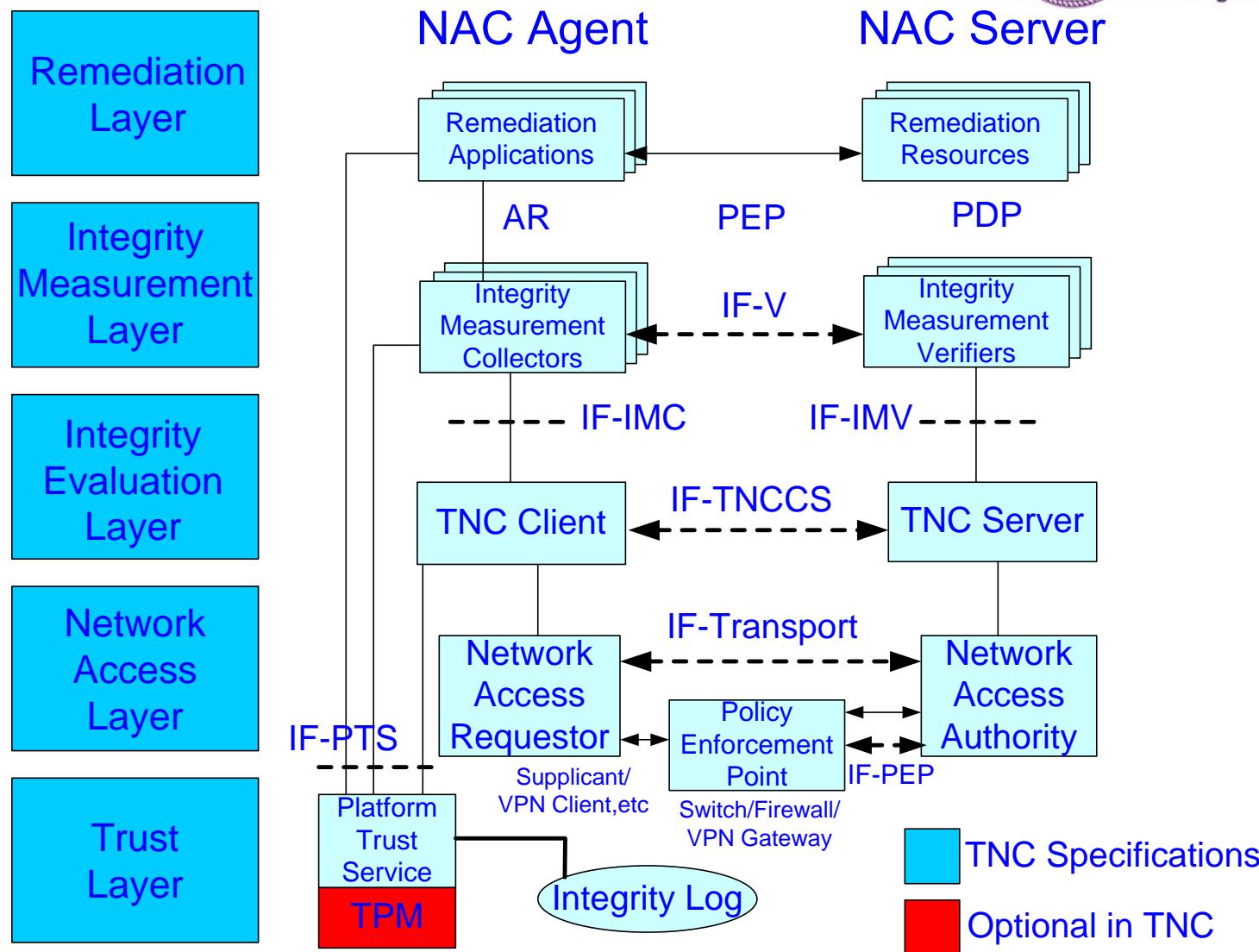
## ➤ Target and execution

- ❖ authenticating and enforcing security policies on client devices that connect to corporate networks.
- ❖ Determining the level of security and "safe-computing" policy compliance of users and devices seeking to attach to a network
- ❖ Providing an appropriate level of network access based on the detected level of policy compliance

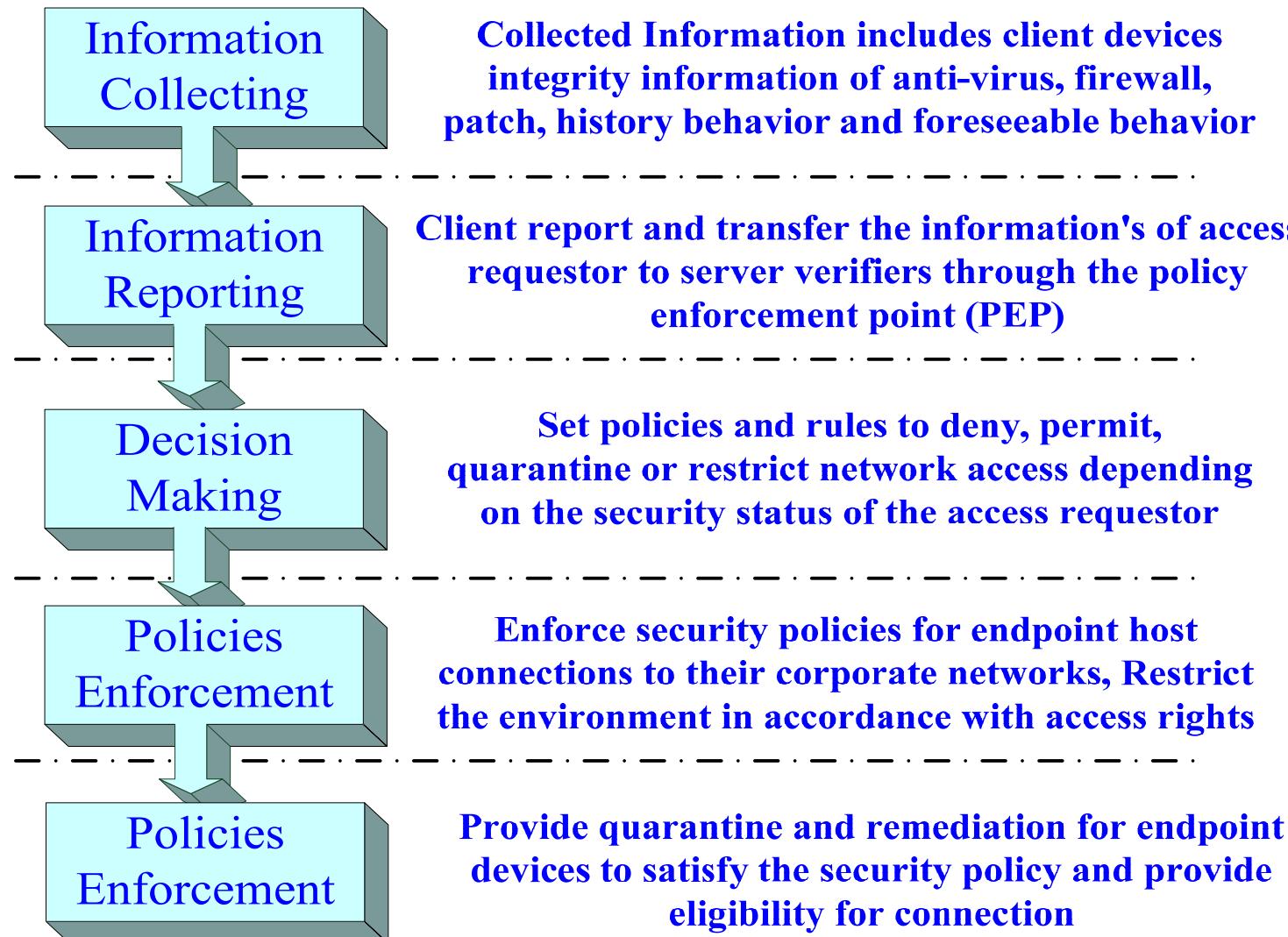
# TNC Architecture



清华大学  
Tsinghua University



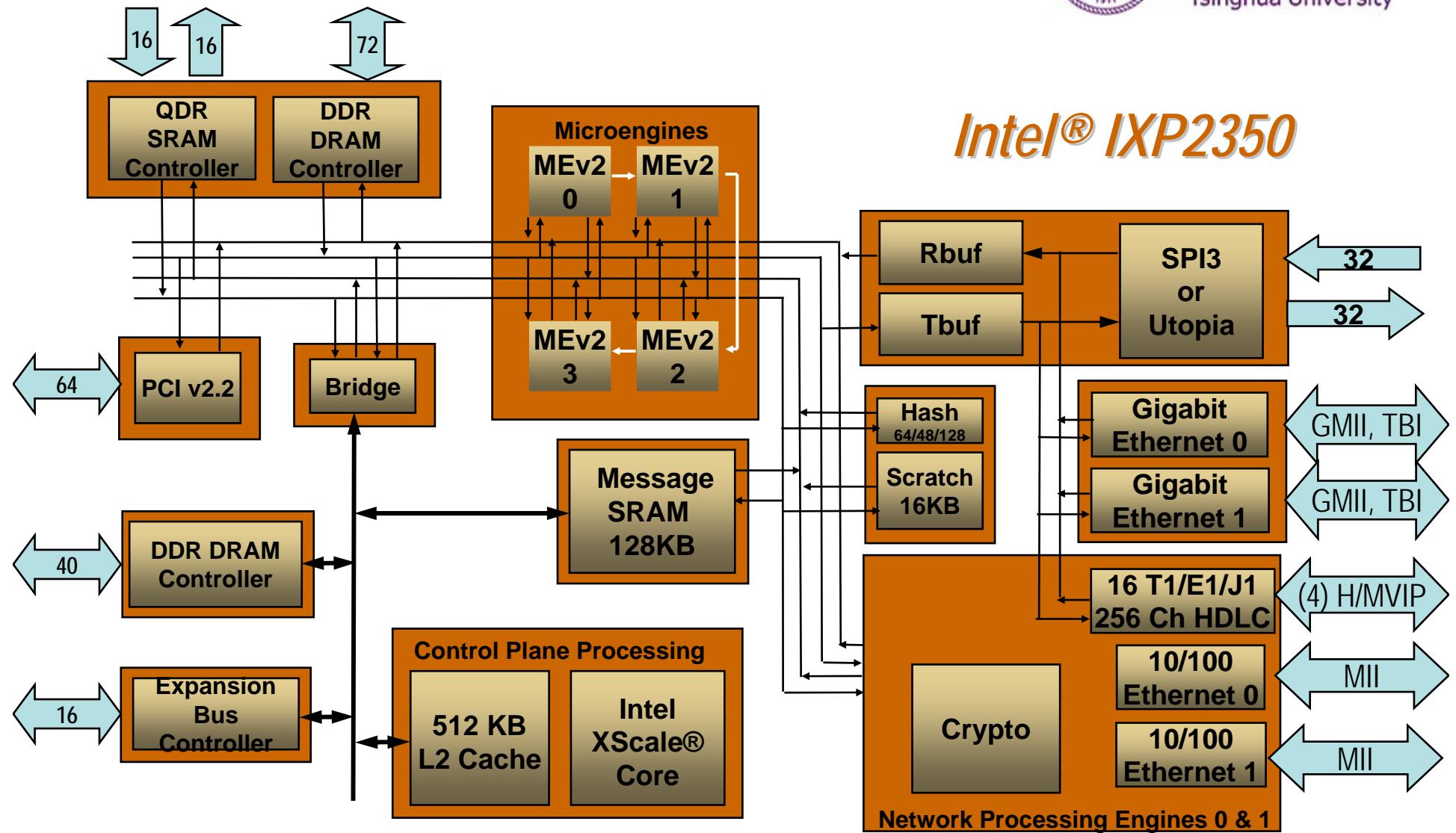
# Procedure



# IXP2350



清华大学  
Tsinghua University



# Our future plan



- Base on IXP2350, implement trusted network connection system
  - ❖ According to hardware condition of IXP2350, we use it as an authentication server, build PDP based on XScale and PEP based on ME
  - ❖ Try to collect trusted information and build reporting mechanism
  - ❖ Referring to the TNC architectural standard of TNG, enforcing security policies on client devices that connect to corporate networks.



*Send me your comments to*

*[zhenchen@csnet1.cs.tsinghua.edu.cn](mailto:zhenchen@csnet1.cs.tsinghua.edu.cn)*

# 留给大家的问题



- 为什么针对Windows系统的病毒特别多？
- 病毒编写者和网络安全商在IT世界丛林法则中的关系和如何互动的？
- 系统攻击者（如蠕虫病毒的编写者）的动机是什么？基本的底线是什么？
- TNC能解决网络安全问题吗？
- 如何彻底解决网络安全问题？