

# COMP 3721数据通信导论

11a - 第11周 - 第1部分

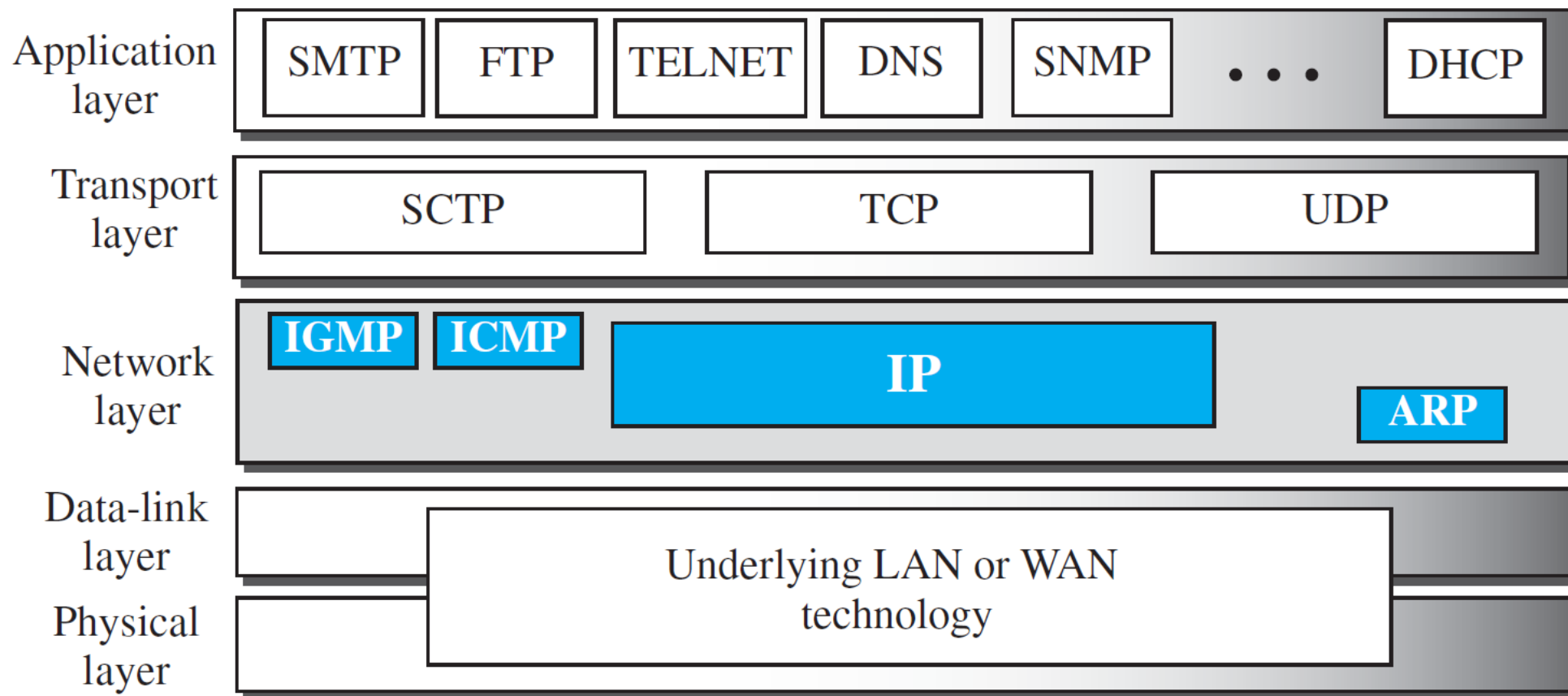
# 学习成果

- 在本讲座结束时，您将能够
  - 解释IPv4协议的工作原理。
  - 解释ICMP协议的行为和目的。

# 简介 – 网络层协议

- **IPv4**（主要协议）
  - 主要的网络层协议 — IPv4 — 负责在网络层进行 **分组**、**转发** 和 **数据包的投递**。
- 三种辅助协议：
  1. **地址解析协议（ARP）**
    - 将网络层地址（IP 地址）映射到链路层地址
  2. **互联网控制消息协议版本 4（ICMPv4）**
    - 帮助 IPv4 处理网络层传输过程中可能出现的一些错误
  3. **因特网组管理协议（IGMP）**
    - 帮助 IPv4 实现多播功能

# 简介 – 网络层协议



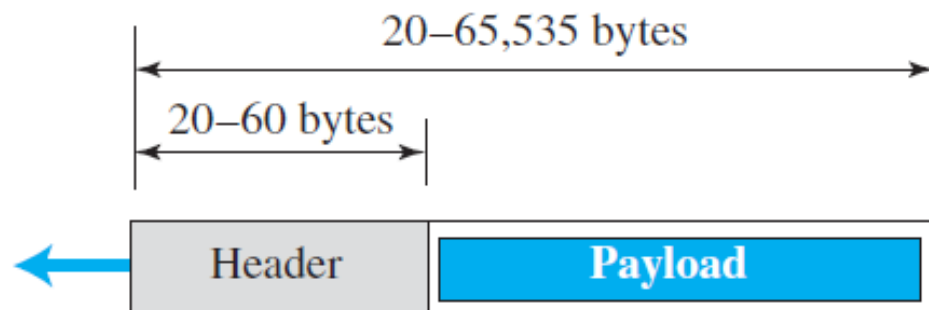
# IPv4

- 一种 **不可靠的** 数据报协议，提供 **尽力而为的交付服务**。
  - IPv4 数据包可能会被损坏、丢失、乱序到达或延迟，并可能造成网络拥塞
- 如果需要可靠性，则必须将 IPv4 与可靠的传输层协议（如 **TCP**）配合使用。
- 一种 **无连接** 协议，采用 **数据报方式**。
- IP 使用的数据包称为 **数据报**。
- **数据报**
  - 一个 **长度可变的数据包**，由两部分组成：**头部** 和 **有效载荷（数据）**
    - 头部包含路由和传输所必需的信息。
    - 有效载荷是来自其他使用IP服务协议的分组。

# 尽最大努力且无连接意味着什么？

- 设想一种邮局服务：
  - 邮局会 **尽力** 投递普通邮件，但并不总能成功。
    - 如果一封未注册的信件丢失或损坏，需由寄件人或预期收件人自行发现。
    - 邮局本身不会跟踪每封信件，也无法通知寄件人某封信件的丢失或损坏情况。
- 无连接协议：
  - 每个数据报被独立处理**独立地**。
  - 每个数据报可以沿**不同的路径**到达**目的地** → 数据报可能乱序到达。
- 为了解决这些问题，它依赖于**高层协议**（如 TCP）。

# IPv4 – 数据报格式



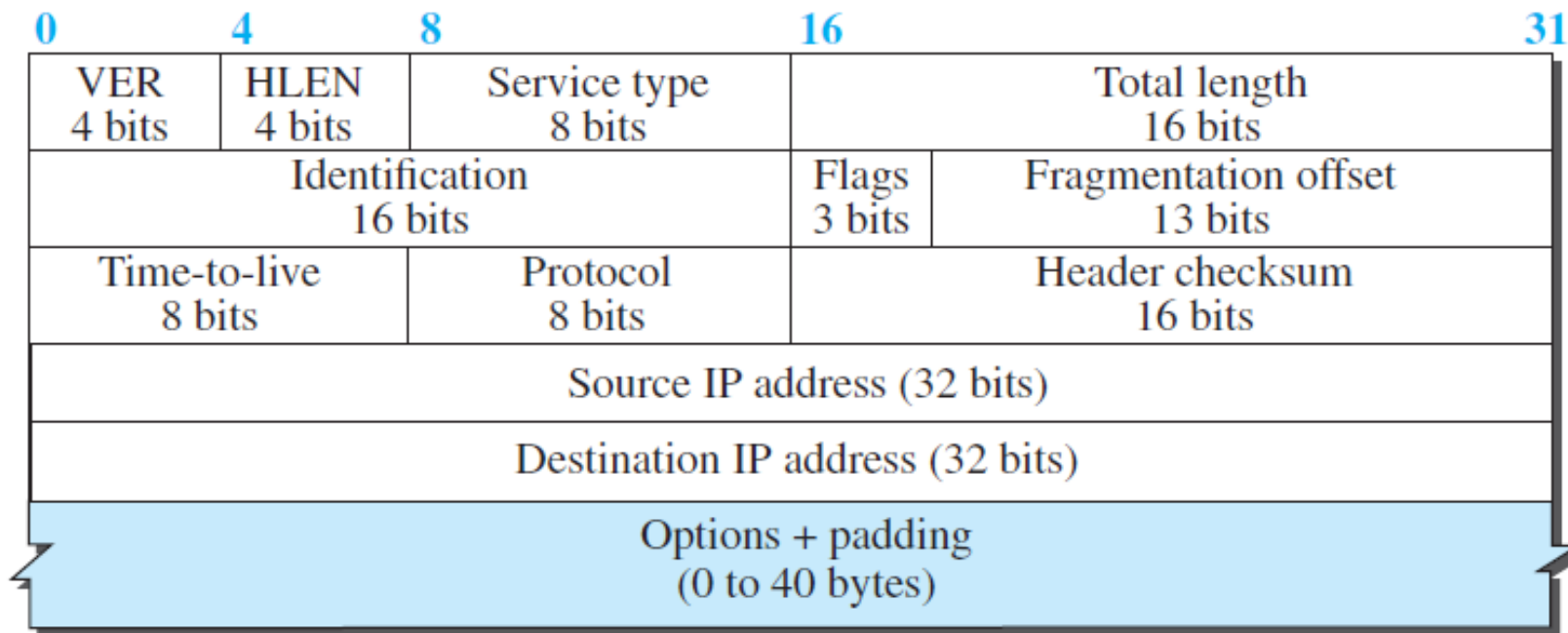
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

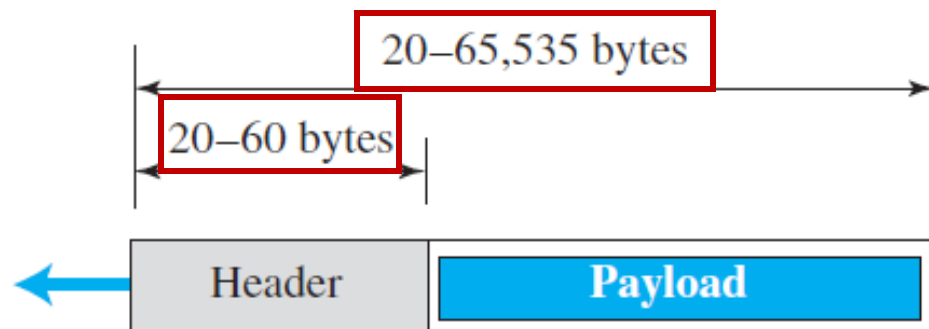
Flags 

	D	M
--	---	---



b. Header

# IPv4 – 数据报格式



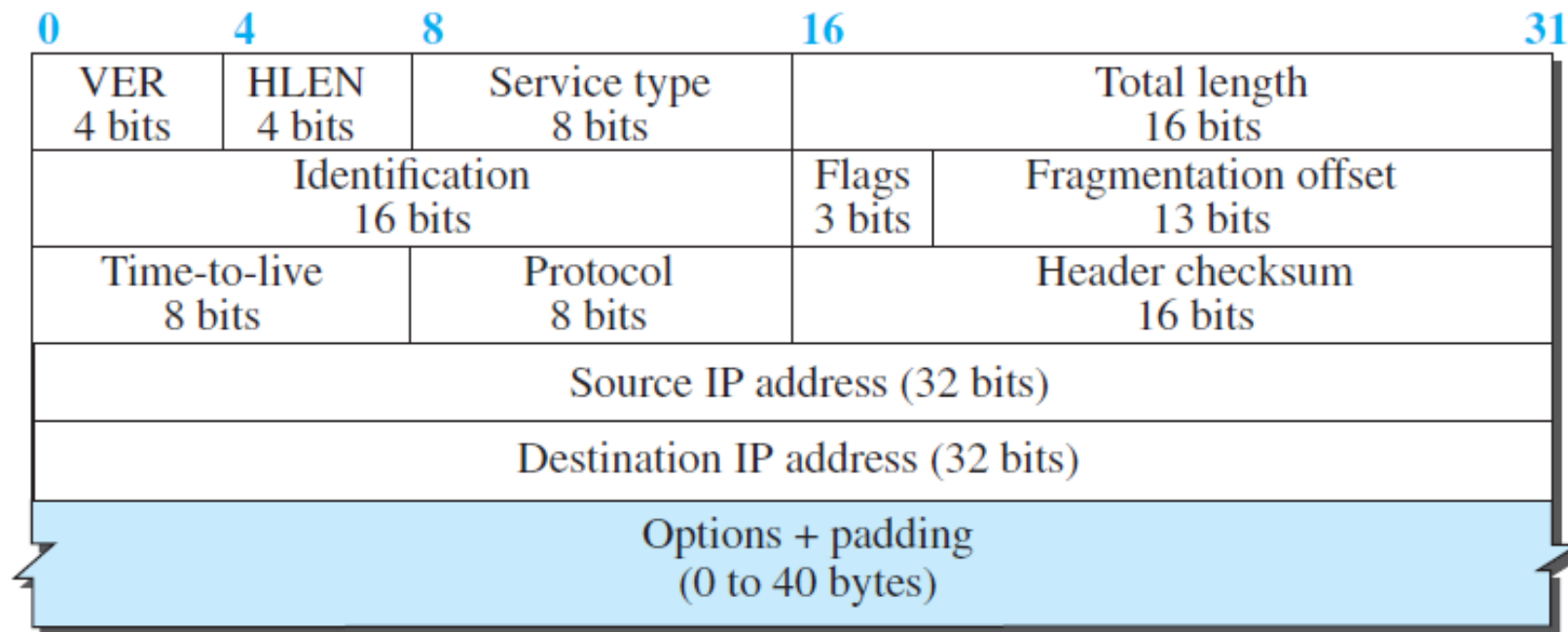
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

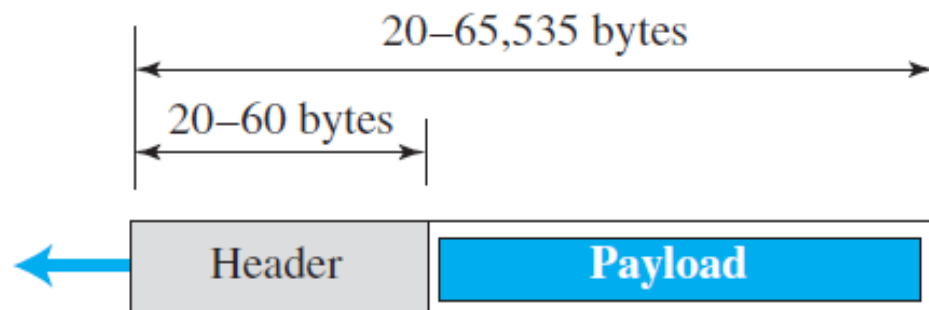
	D	M
--	---	---



b. Header



# IPv4 – 数据报格式



a. IP datagram

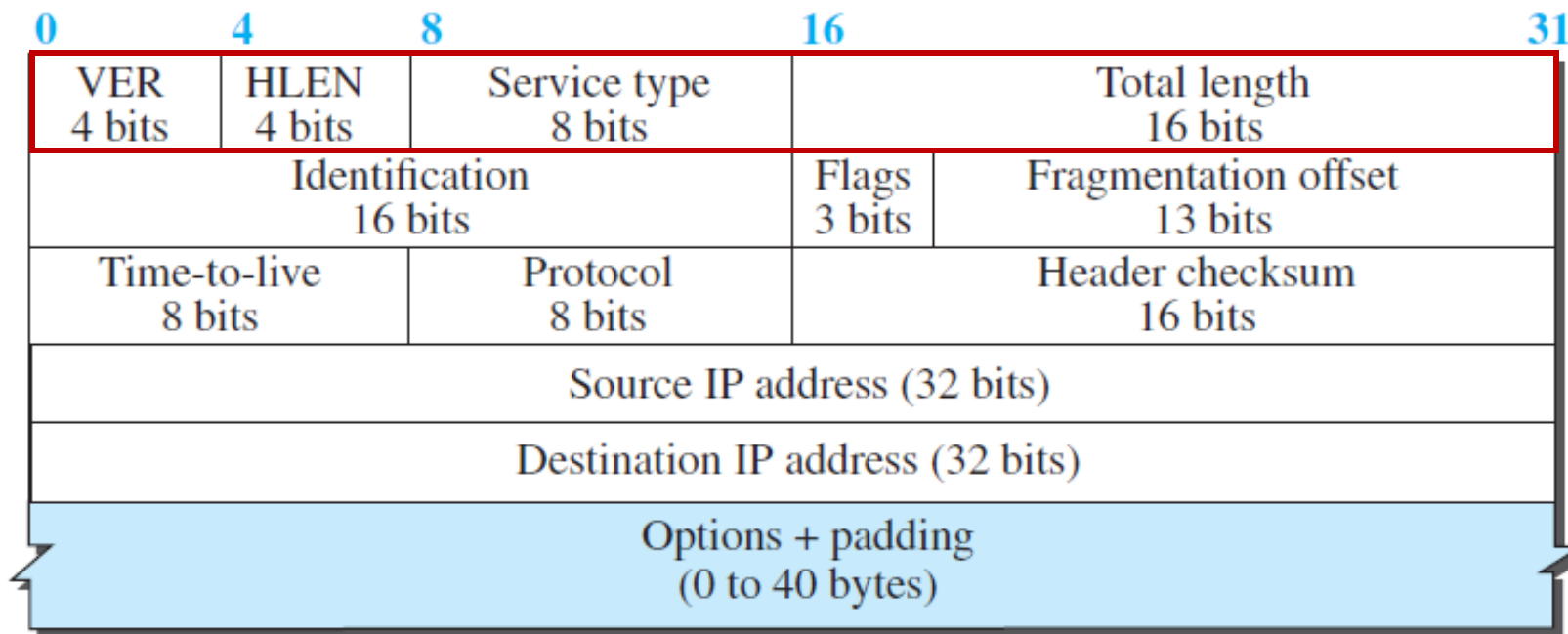
## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

	D	M
--	---	---

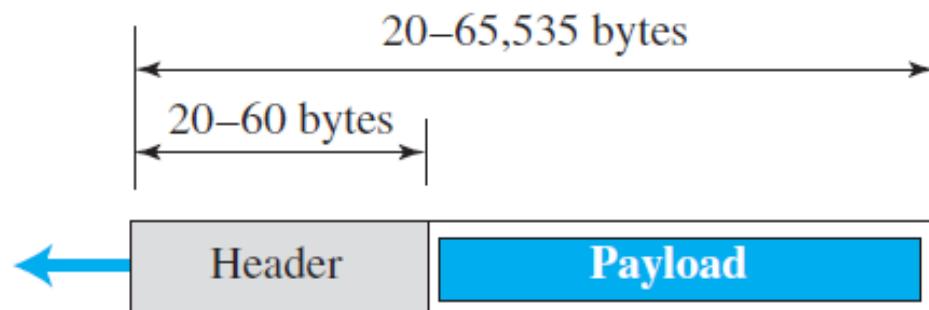
**首部长度的  
(HLEN)：**以  
4字节字为单位的  
IP首部总长度



b. Header

**总长度：**IP数  
据报的总字节长度  
(包括首部和有效  
载荷)。

# IPv4 – 数据报格式



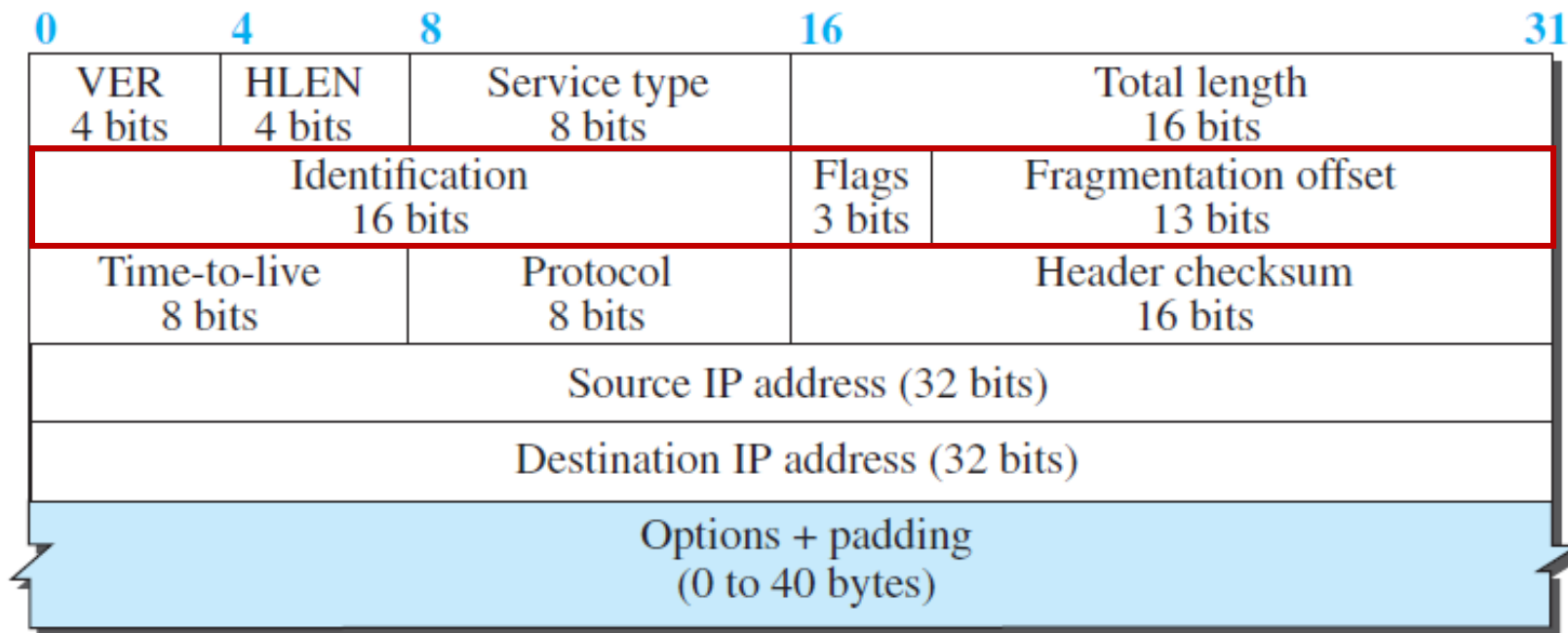
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

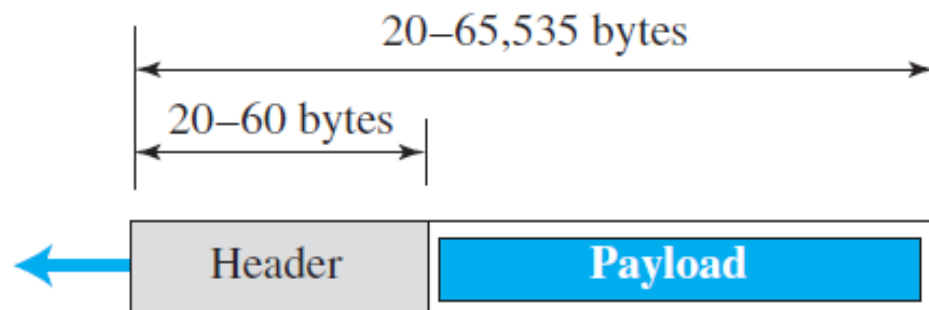
Flags 

	D	M
--	---	---



b. Header

# IPv4 – 数据报格式



a. IP datagram

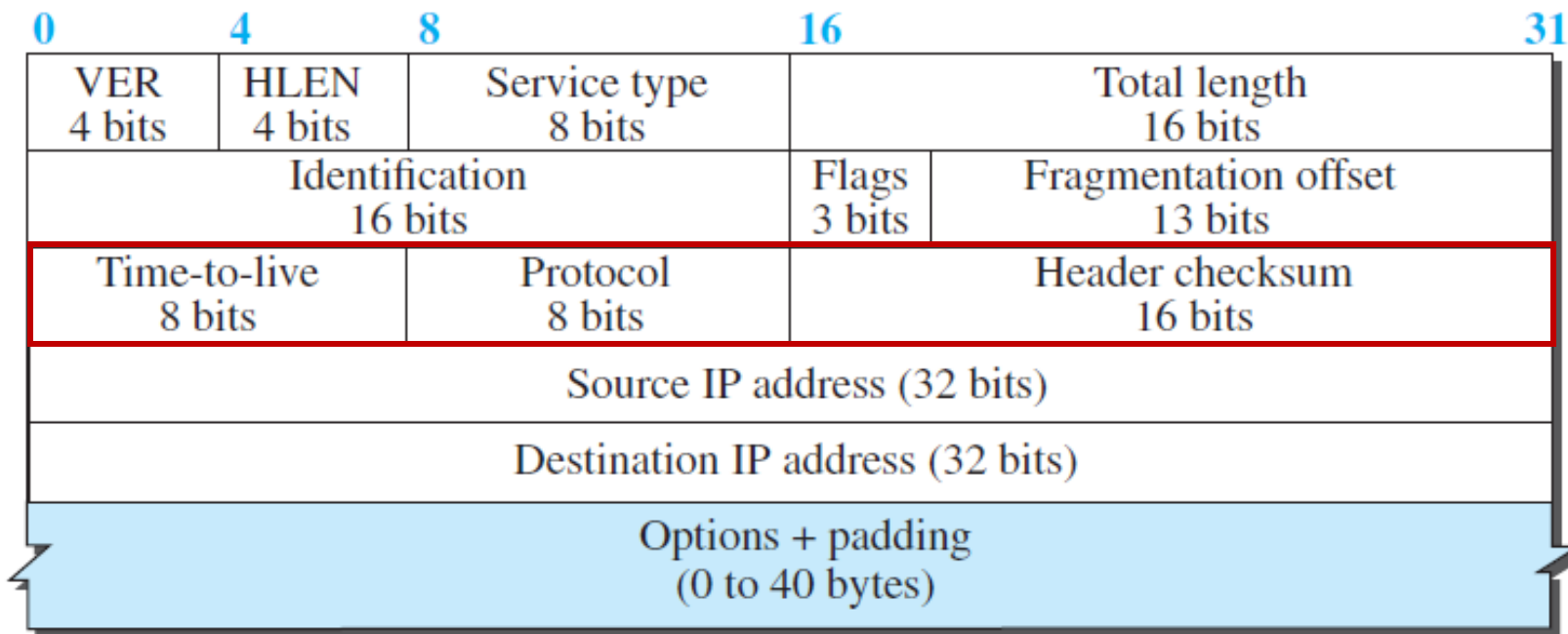
## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

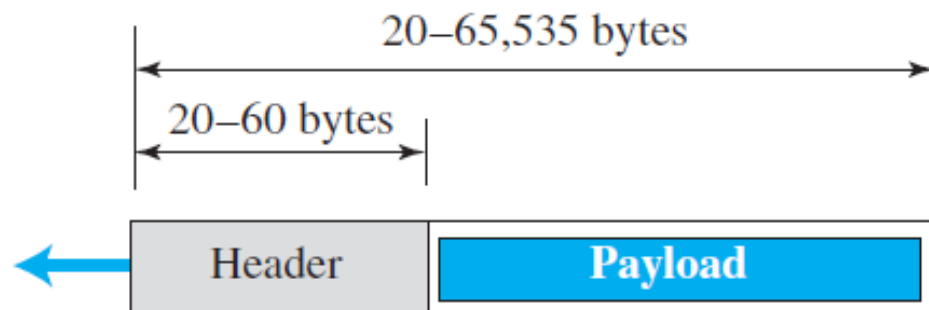
	D	M
--	---	---

生存时间:  
控制最大数量  
数据报经过的  
跳数 (路由器)



b. Header

# IPv4 – 数据报格式



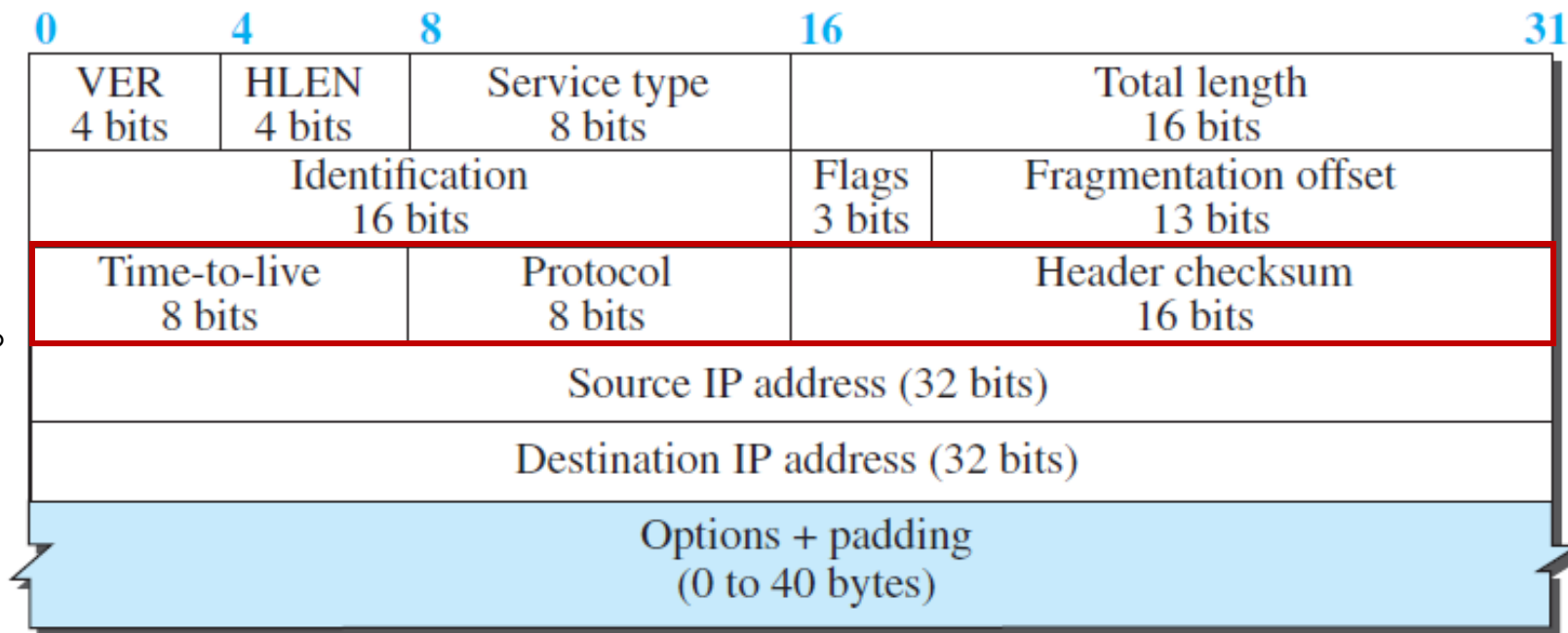
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

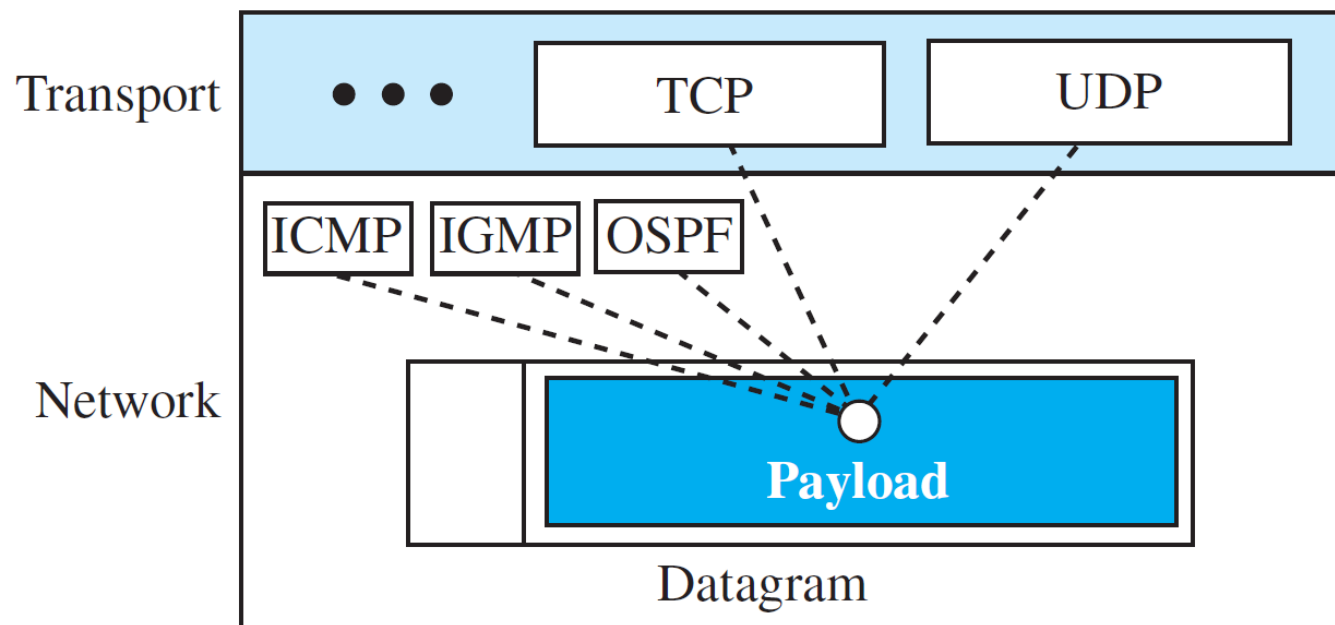
	D	M
--	---	---



b. Header

协议：  
其数据包被承载在有效载荷中的协议。

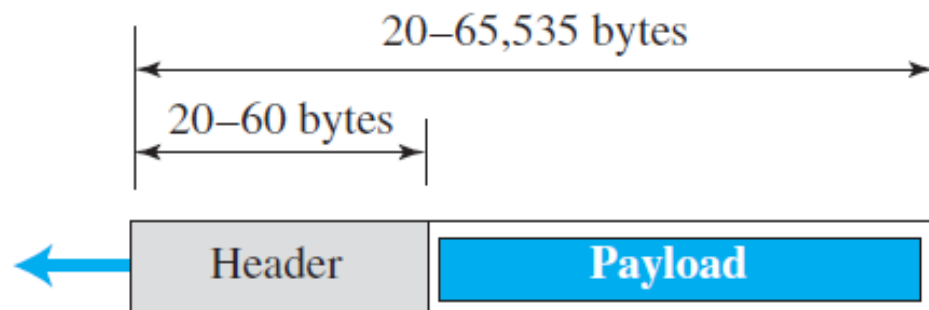
# IPv4 – 数据报协议字段



## Some protocol values

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

# IPv4 – 数据报格式



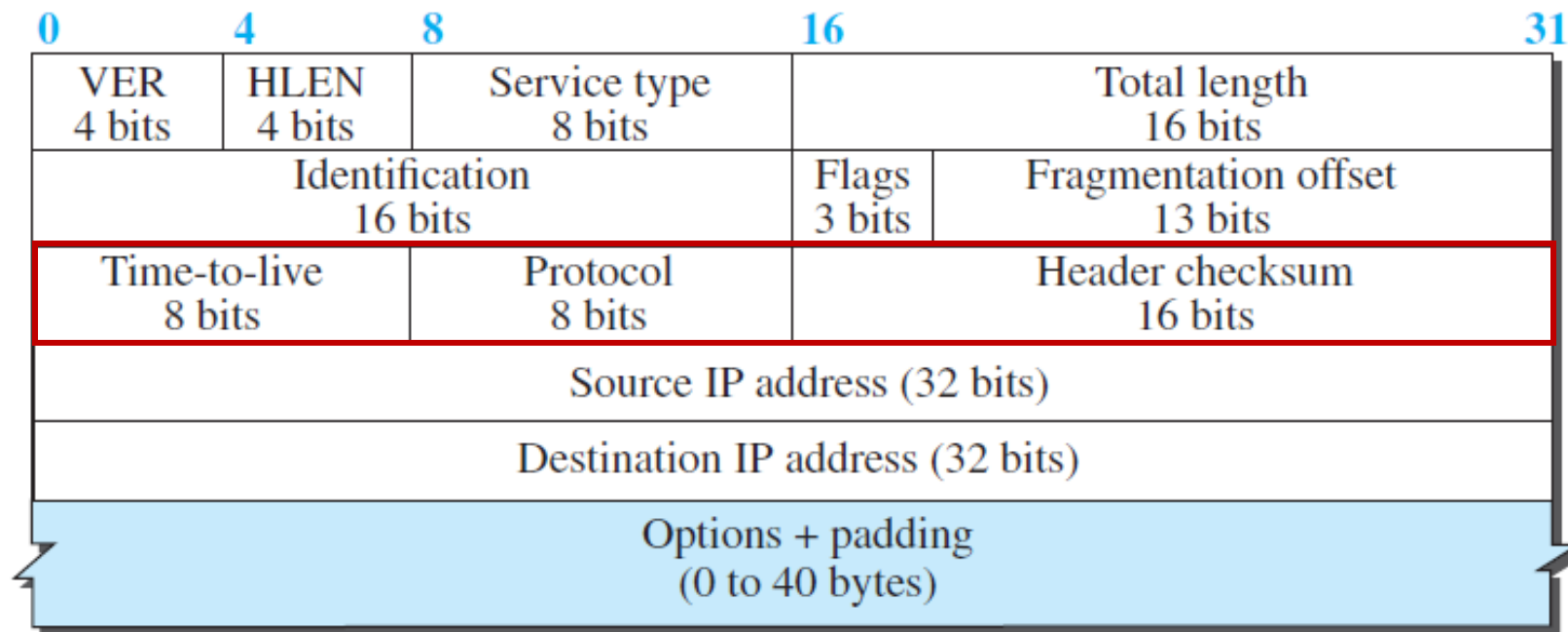
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

	D	M
--	---	---

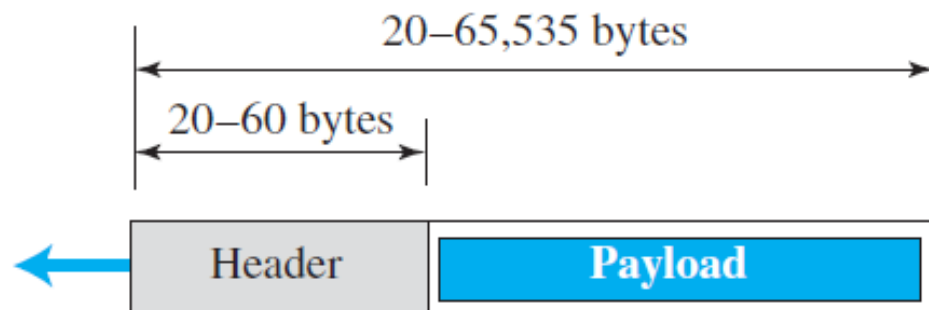


b. Header

为什么 IP 头部需要错误处理?

为什么在每个路由器上都要重新计算校验和?

# IPv4 – 数据报格式



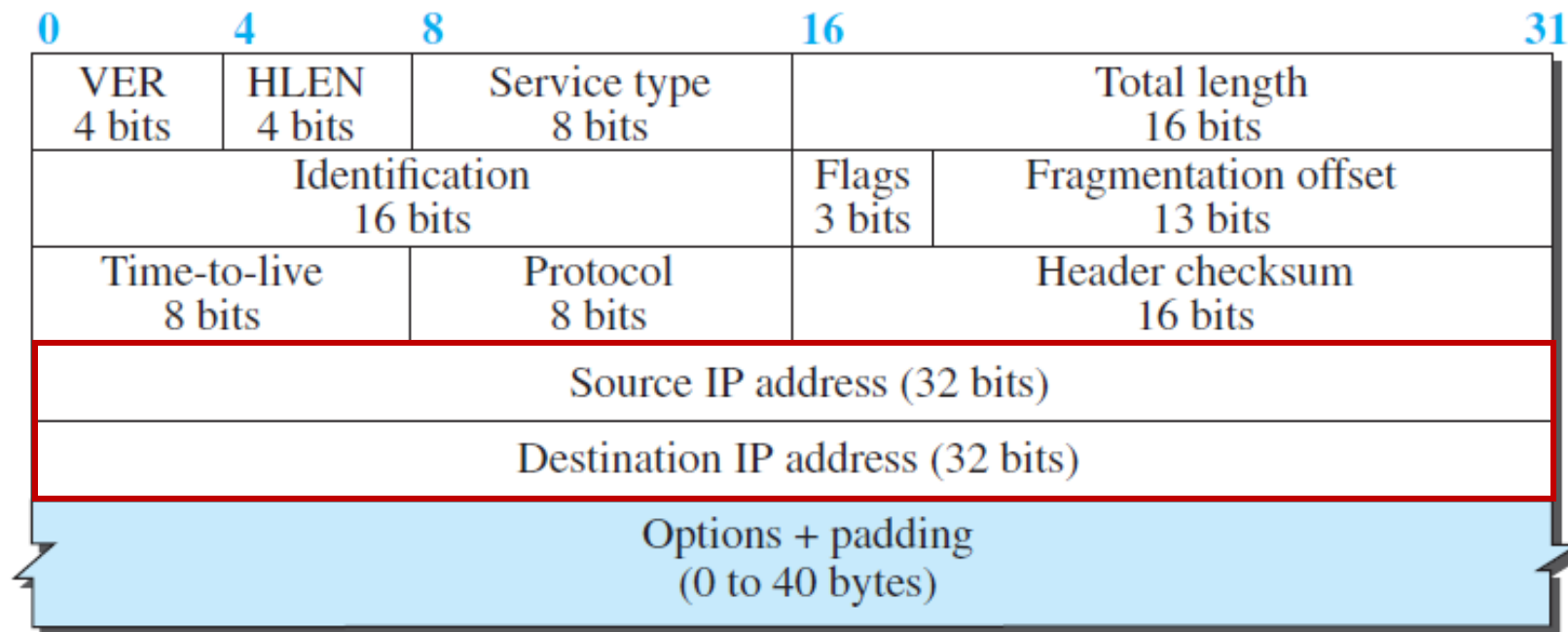
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

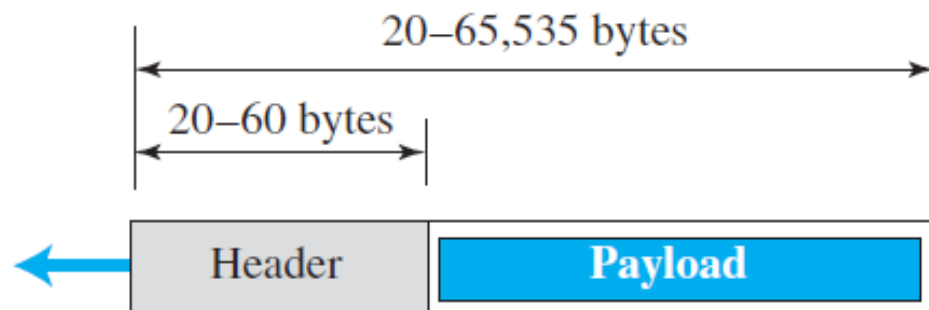
Flags 

	D	M
--	---	---



b. Header

# IPv4 – 数据报格式



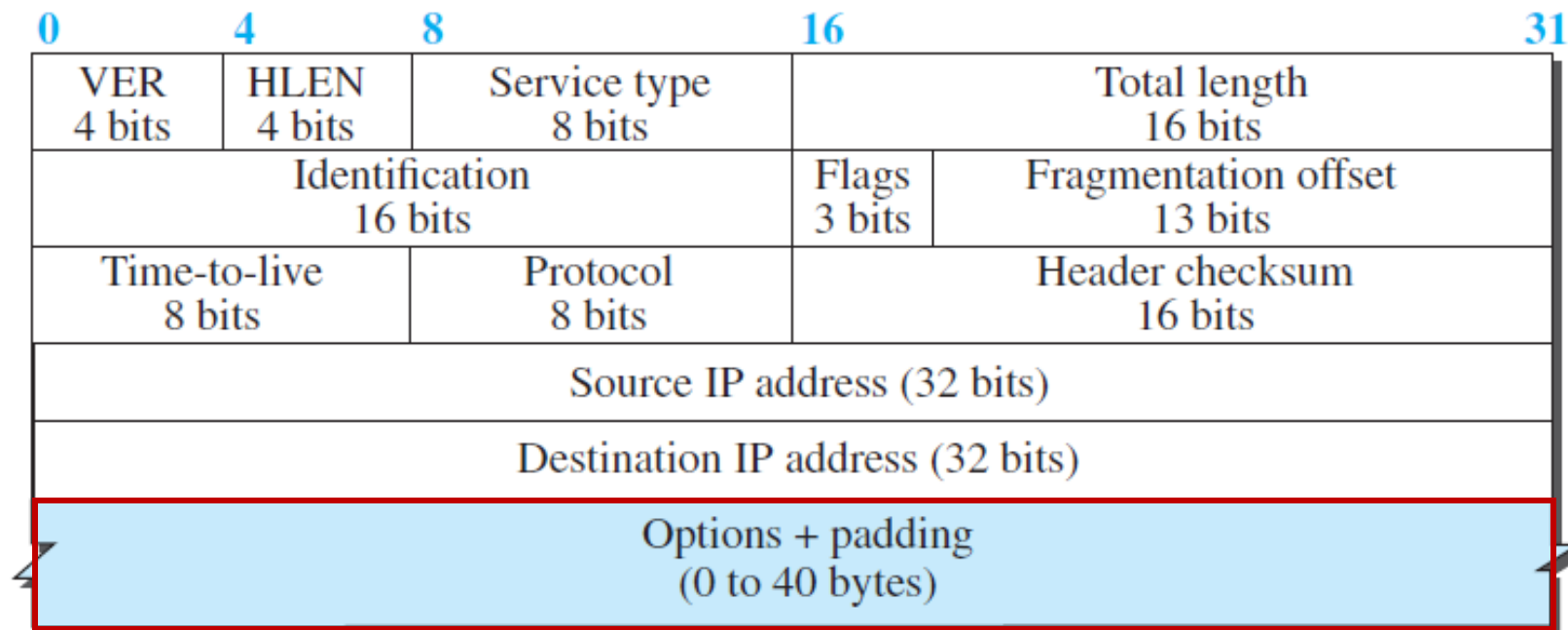
a. IP datagram

## Legend

VER: version number  
HLEN: header length  
byte: 8 bits

Flags 

	D	M
--	---	---



b. Header



# 示例 1

- 一个IPv4数据包已到达，其前8位为 $(01000010)_2$ 。该接收方丢弃了数据包。为什么？

# 示例 1

- 一个IPv4数据包已到达，其前8位为 $(01000010)_2$ 。该接收方丢弃了数据包。为什么？
- **答案：**
- 此数据包中存在错误。左边的4位 $(0100)_2$ 表示版本号，是**正确的**。
- 接下来的4位 $(0010)_2$ 显示了一个无效的头部长度的 $(2 \times 4 = 8)$ 。头部中最小字节数必须为20。该数据包在传输过程中已**损坏**。

# 示例 2 — IPv4 中的头部校验和计算

• 我们计算一个16位的校验和，并将其填入IP头部的校验和字段中。我们需要考虑头部中每一个16位（2字节）的部分。为了表示简便，我们使用十六进制格式，每个十六进制字符代表4位。假设以下内容表示一个IP数据包的IP头部字段，请计算其校验和。

4	5	0	28
49.153		0	0
4	17	0	
10.12.14.5			
12.6.7.9			

4, 5, and 0	→	4	5	0	0
28	→	0	0	1	C
49.153	→	C	0	0	1
0 and 0	→	0	0	0	0
4 and 17	→	0	4	1	1
0	→	0	0	0	0
10.12	→	0	A	0	C
14.5	→	0	E	0	5
12.6	→	0	C	0	6
7.9	→	0	7	0	9

# 示例 2 – 答案

		2	
4	5	0	0
0	0	1	C
C	0	0	1
0	0	0	0
0	4	1	1
0	0	0	0
0	A	0	C
0	E	0	5
0	C	0	6
0	7	0	9
<hr/>			E

$$0 + 12 + 1 + 0 + 1 + 0 + 12 + 5 + 6 + 9 \\ = 46 = 0x2E$$

# 示例 2 – 答案

		2	
4	5	0	0
0	0	1	C
C	0	0	1
0	0	0	0
0	4	1	1
0	0	0	0
0	A	0	C
0	E	0	5
0	C	0	6
0	7	0	9
		4	E

$$2 + 0 + 1 + 0 + 0 + 1 + 0 + 0 + 0 + 0 + 0 = 4 = 0$$

x4

# 示例 2 – 答案

	3		
4	5	0	0
0	0	1	C
C	0	0	1
0	0	0	0
0	4	1	1
0	0	0	0
0	A	0	C
0	E	0	5
0	C	0	6
0	7	0	9
<hr/>			
3	4	4	E

$$5 + 0 + 0 + 0 + 4 + 0 + 10 + 14 + 12 + 7 \\ = 52 = 0x34$$

# 示例 2 – 答案

	3			
	4	5	0	0
	0	0	1	C
	C	0	0	1
	0	0	0	0
	0	4	1	1
	0	0	0	0
	0	A	0	C
	0	E	0	5
	0	C	0	6
	0	7	0	9
1	3	4	4	E

$$3 + 4 + 0 + 12 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 \\ = 19 = 0x13$$

# 示例 2 – 答案

	4	5	0	0	
	0	0	1	C	
	C	0	0	1	
	0	0	0	0	
	0	4	1	1	
	0	0	0	0	
	0	A	0	C	
	0	E	0	5	
	0	C	0	6	
	0	7	0	9	
Sum →	1	3	4	4	E



# 示例 2 – 答案

	4	5	0	0	
	0	0	1	C	
	C	0	0	1	
	0	0	0	0	
	0	4	1	1	
	0	0	0	0	
	0	A	0	C	
	0	E	0	5	
	0	C	0	6	
	0	7	0	9	
Sum →	1	3	4	4	E
Wrapped sum →		3	4	4	F

$0x1 + 0xE = 0xF$

# 示例 2 – 答案

		4	5	0	0	
		0	0	1	C	
		C	0	0	1	
		0	0	0	0	
		0	4	1	1	
		0	0	0	0	
		0	A	0	C	
		0	E	0	5	
		0	C	0	6	
		0	7	0	9	
		<hr/>				
Sum	→	1	3	4	4	E
Wrapped sum	→		3	4	4	F
Checksum	→		C	B	B	0

Wrapped Sum = Sum mod FFFF  
Checksum = FFFF – Wrapped Sum

# 数据报与帧

- 数据报可以穿越不同的 **网络**。
- 每个路由器将其接收到的帧中的IP数据报解封装，**处理该数据报，然后**再将其封装到另一个帧中。封装它到另一个帧中。
  - 所接收帧的格式和大小取决于该帧刚刚经过的物理网络所使用的协议。
- 例如，如果一个路由器将**局域网**连接到**广域网**，它将以局域网格式接收帧，并以广域网格式发送帧。

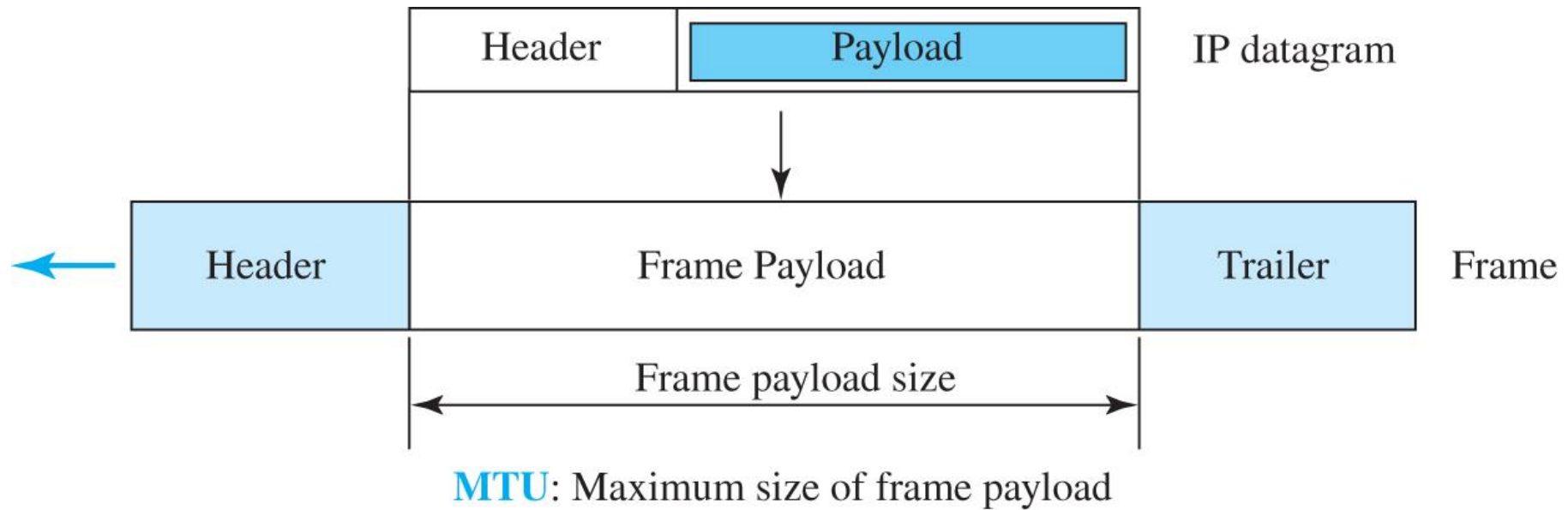
# 帧格式

- 每个 **链路层协议** 都有其自身的 **帧格式**。
  - 每种格式的一个特点是可被封装的有效载荷的 **最大大小**。
- 当数据报被封装在帧中时，数据报的总大小必须 **小于** 此最大大小。

# MTU

- **最大传输/传输单元 (MTU)** 。
  - 链路层帧**可承载的**最大数据量。
  - 局域网中的该值通常为1500字节。
- **问题**：发送方与目的地之间路径上的每条链路可能使用不同的链路层协议，而每种协议可能具有不同的MTU。
- 为了使IP协议与物理网络**无关**，设计者决定将IP数据报的**最大长度设为65,535字节**。

# MTU



# 分片

- 将数据报进行分割，以便其能够通过具有不同最大传输单元（MTU）的网络。
- 当一个数据报被分片时，**每个分片都有自己的头部** 其中大多数字段被重复，但**部分字段已被修改**。
  - 在到达最终目的地之前，一个数据报可能被**多次分片**（如果经过一个MTU更小的网络）。
- 数据报可以由源主机或路径中的任何路由器进行分片。
  - 数据报的**重新组装**仅由目的主机完成。
- 数据报的**有效载荷**被**分片**。
  - 除了某些选项外，头部的大部分部分必须由所有分片复制。

# 分片

- 对数据报进行分片的主机或路由器必须更改三个字段的值：**标志**、**分片偏移**和**总长度**。
  - 无论是否发生分片，都必须重新计算校验和的值。

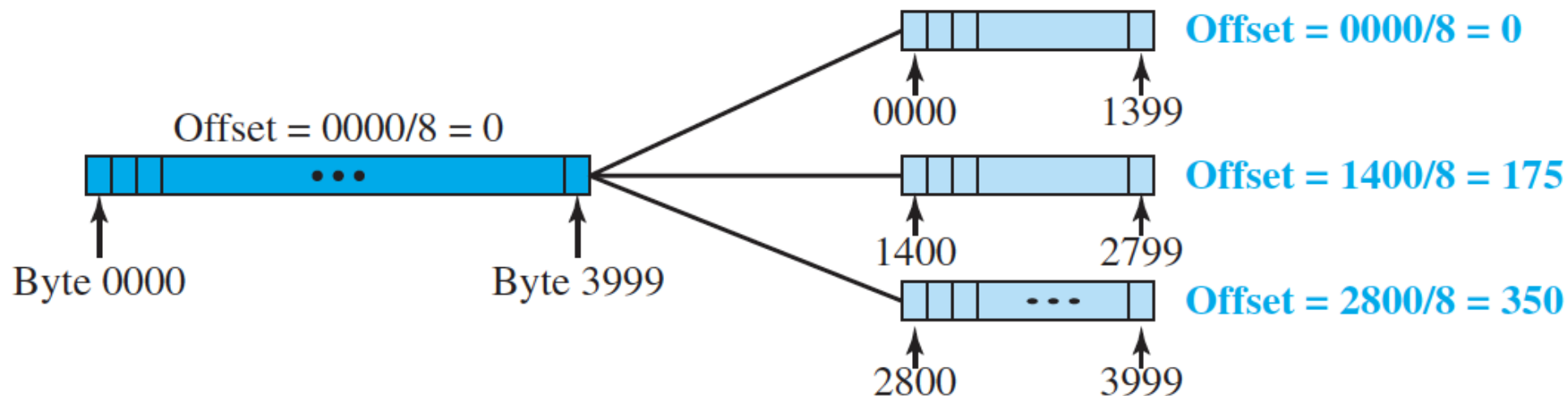


# 分片

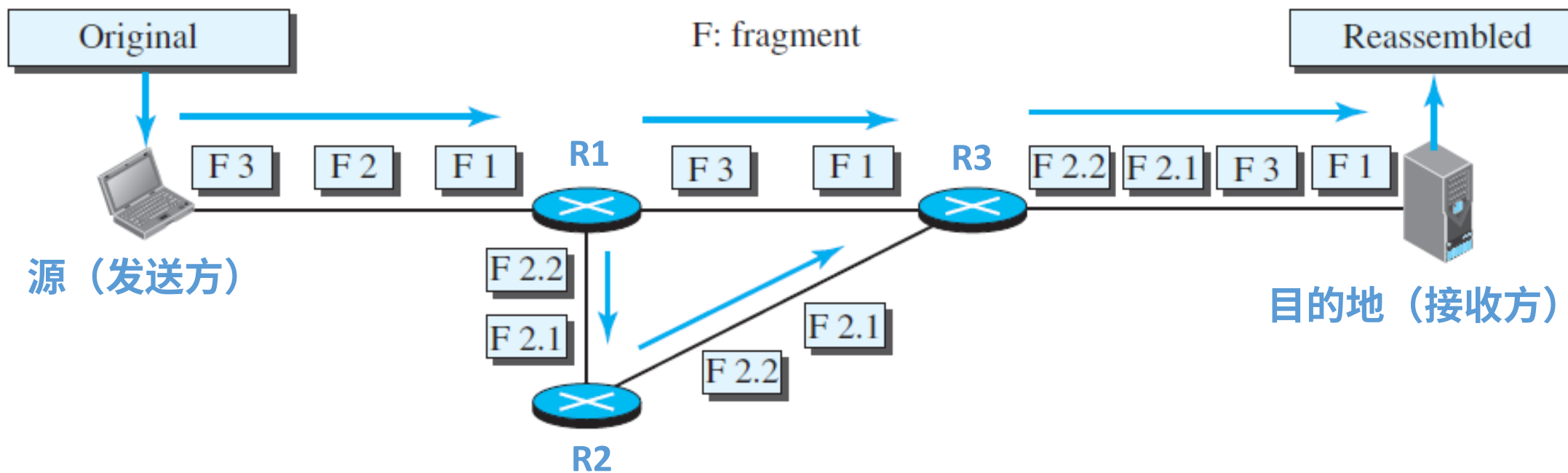
- 为了帮助目的主机**完成重组任务**，IP数据报头部中包含了三个字段。
  - **标识号**（16位字段）→ 与原始数据报及其所有分片相同
  - **标志位**（3位字段）
    - 最左边的比特未使用
    - 第二个比特（**不分片**比特，**D**）→ 若为1，则数据报**不得被分片**，即使其长度超过链路MTU（此时将丢弃该数据报，并向源端发送ICMP错误消息）。
    - 第三位（**更多片段**位，**M**）→ 用于确定最后一个片段（最后一个或唯一片段为0）
  - **分片偏移** → 用于指定该片段在原始IP数据报中的位置（原始数据报中数据的**偏移量**，以 8字节**为单位进行测量**）。

# 分片— 示例

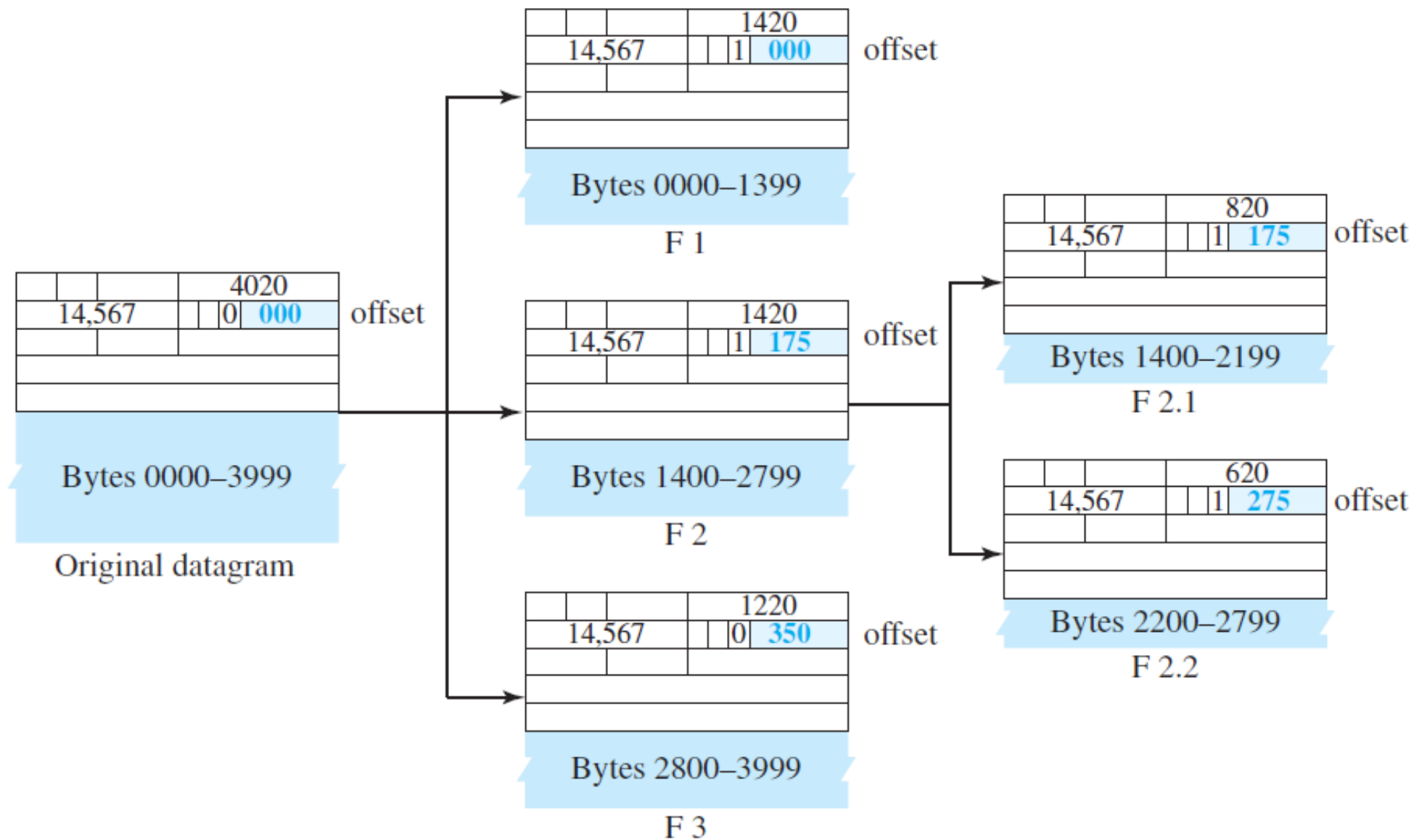
- 偏移量的值以**8字节为单位**进行度量，因为偏移量字段的长度仅为13位，无法表示大于8191 ( $2_{13}$ ) 的字节序列。这要求对数据报进行分片的主机或路由器必须选择每个分片的大小，使得**第一个字节的编号**能被**8整除**。



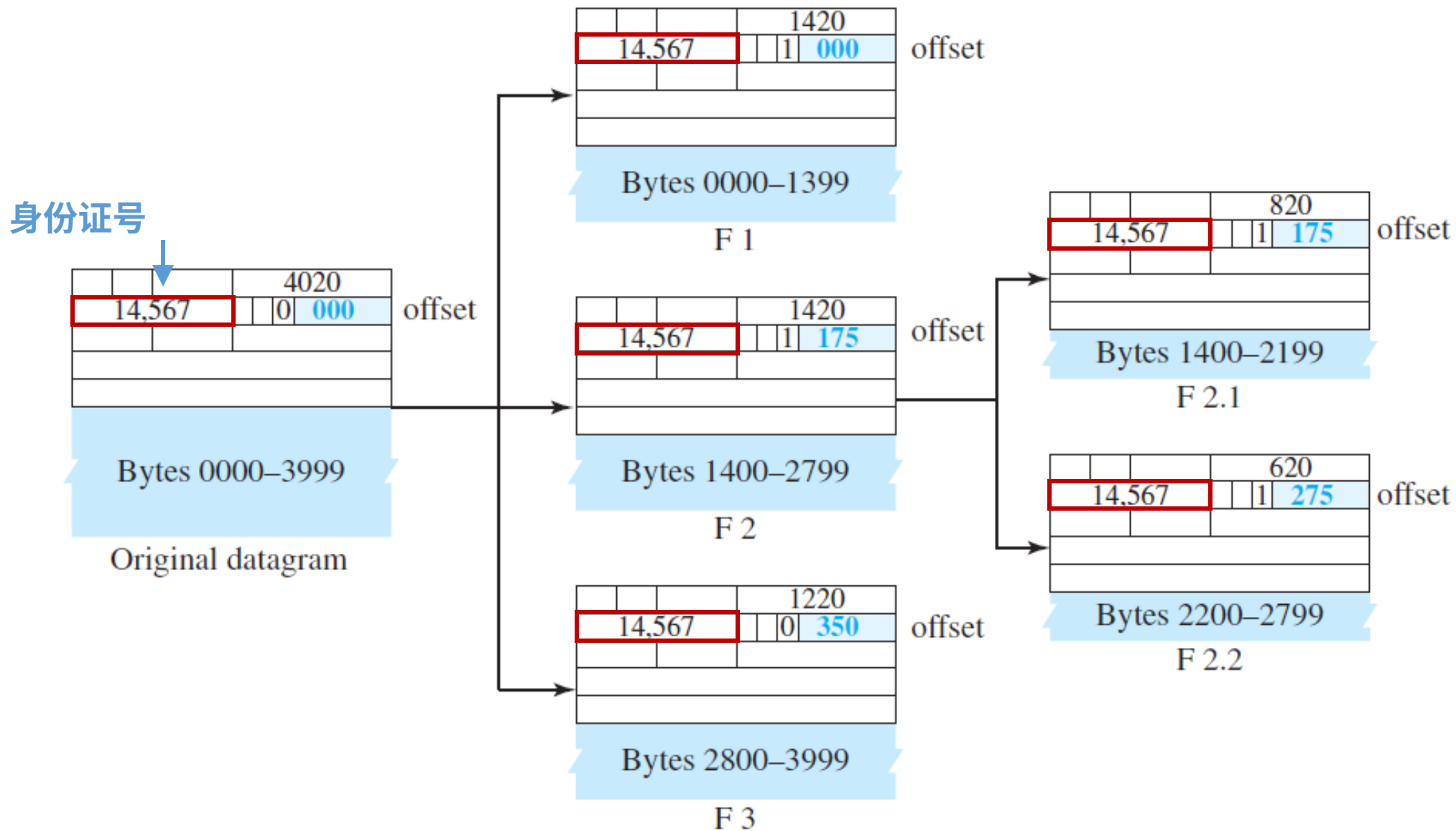
# 分片—示例



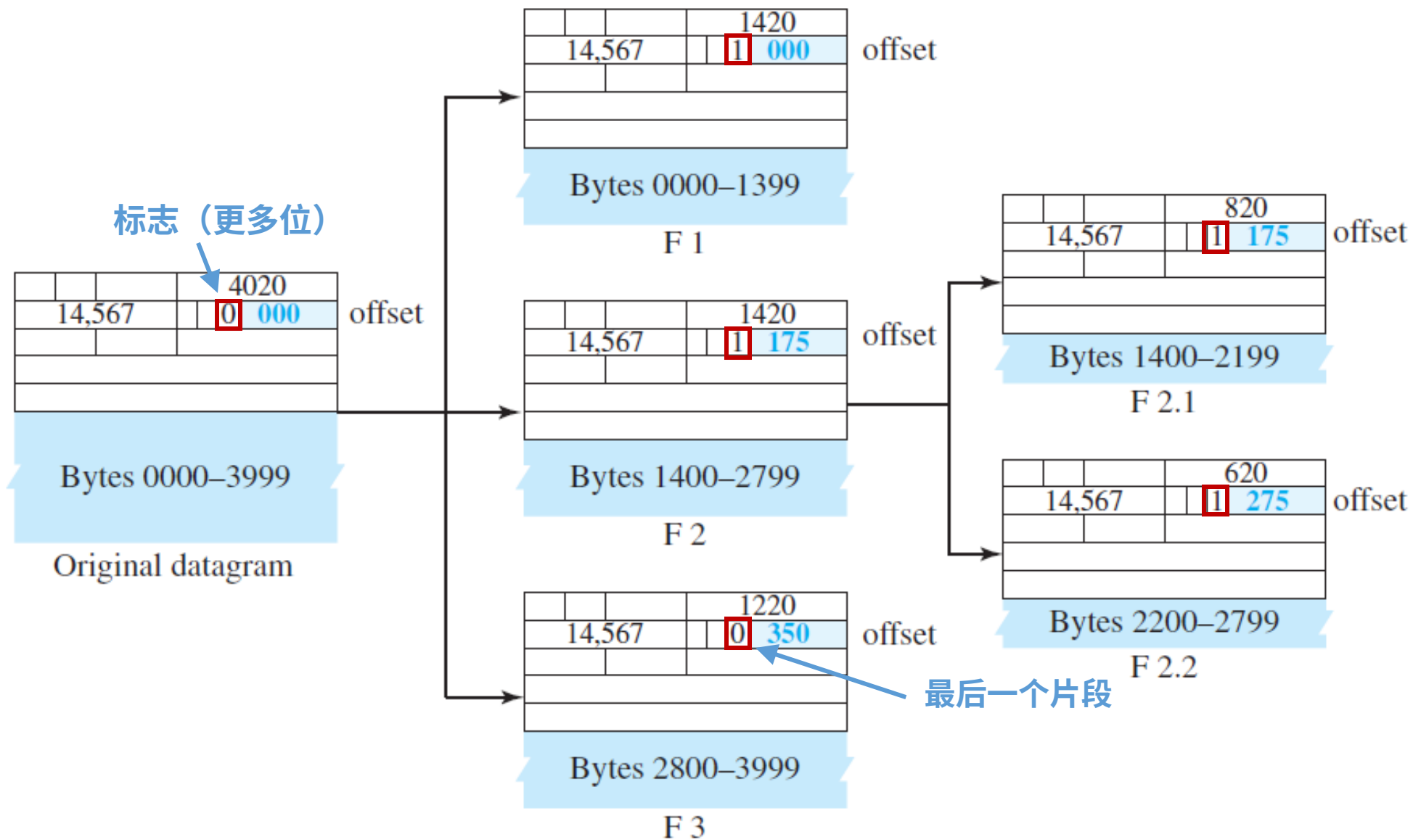
# 碎片化 – 示例



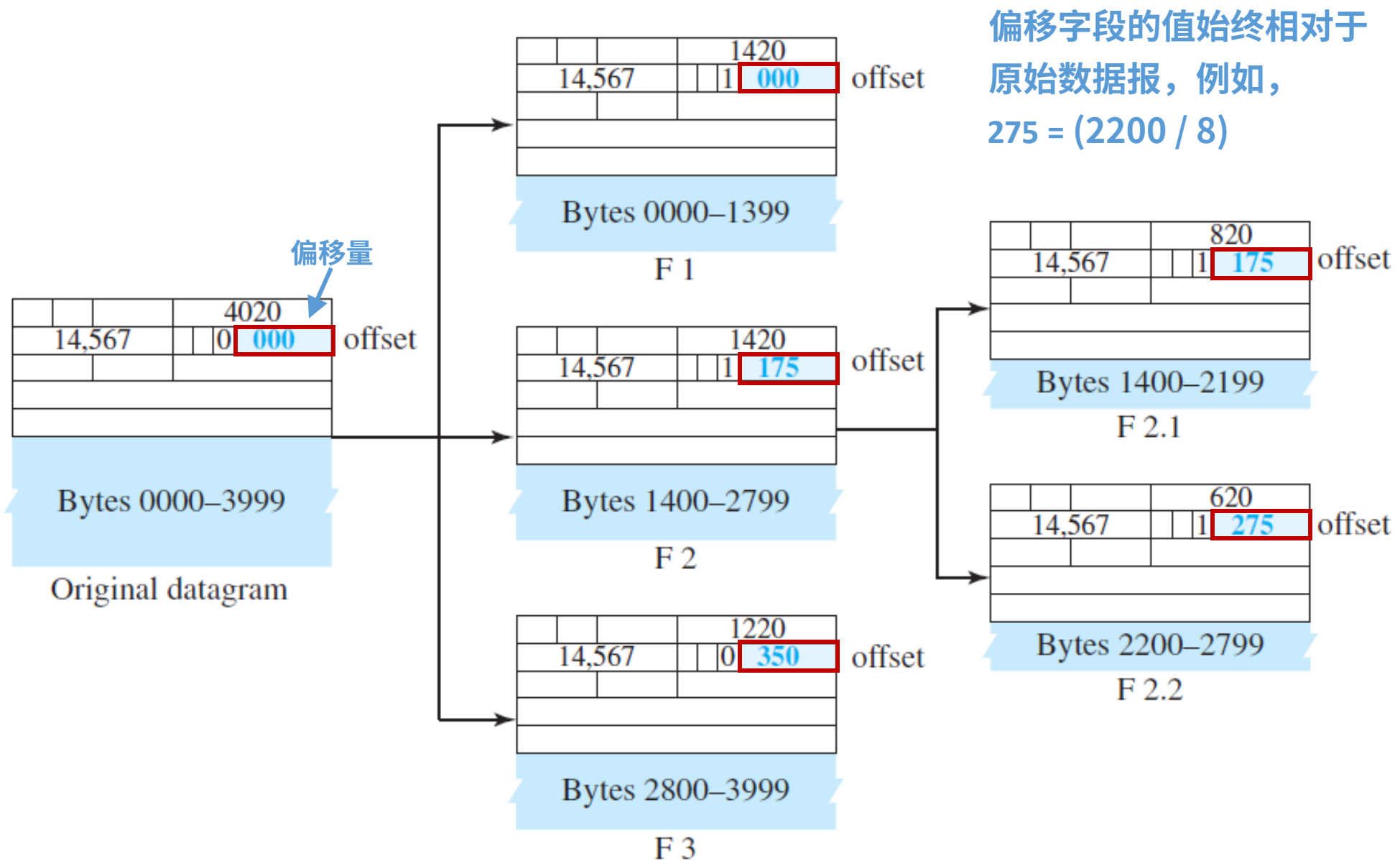
# 碎片化 – 示例



# 分片—示例



# 分片 - 示例



# ICMPv4

- IPv4 没有**错误报告**或**纠错**机制。
- 传输过程中可能发生哪些错误？
  - 路由器丢弃数据报，因为它无法找到路由。
  - 生存时间字段的值为零。
  - 最终目的地丢弃收到的数据报分片，因为它尚未接收到所有分片。
- 以下是发生错误的情况示例，其中 IP 协议**没有内置机制**来**通知原始主机**。



# ICMPv4

- IP 协议还缺乏主机和管理查询的机制。
  - 主机有时需要确定路由器或另一台主机是否处于活动状态。
  - 有时网络管理员需要从另一台主机或路由器获取信息。
- **互联网控制消息协议第4版（ICMPv4）** 正是为弥补上述缺陷而设计的。

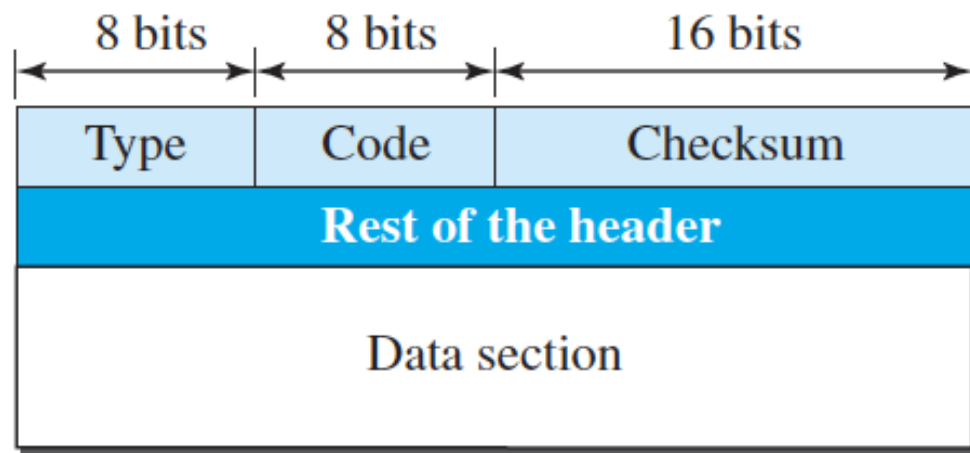
# ICMPv4

- 一种 **网络层协议**（RFC792）。
- ICMP 被 **主机** 和 **路由器** 用来相互传递网络层信息。
- ICMP 消息首先被封装在 IP 数据报中，然后再传送到下层。
  - IP 数据报中 **协议** 字段的值被 **设为 1**，以表示该 **IP 有效载荷** 是一个 **ICMP 消息**。

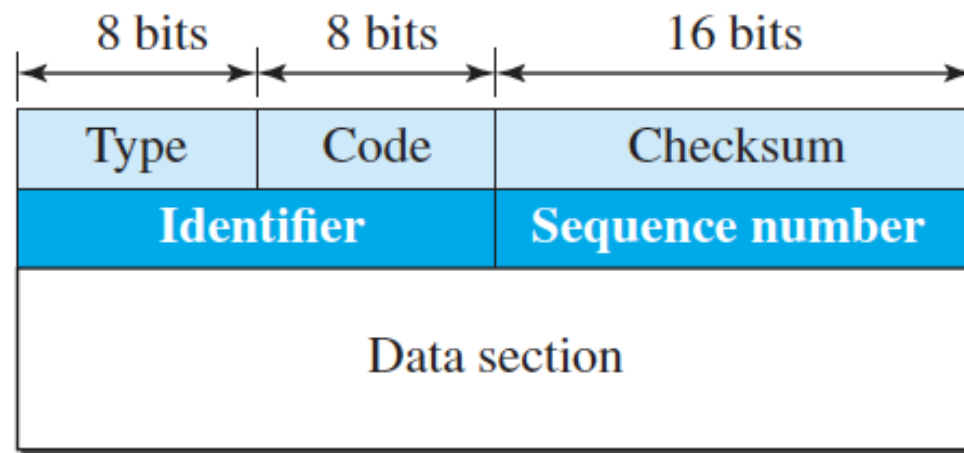
# ICMPv4 – 消息

- 两种类型的消息
  1. 错误报告消息
    - 路由器或主机（目标）在处理 IP 数据报时可能遇到的问题，并进行报告。
    - ICMP 使用源 IP 地址将错误消息发送回数据报的源端（发起者）。
  2. 查询消息
    - 这些消息成对出现（请求和应答），并帮助主机或网络管理员从特定信息获取路由器或另一台主机。
    - 用于探测或测试互联网上主机或路由器的活跃状态，测量两个设备之间 IP 数据报的单向或往返时间，甚至判断两台设备的时钟是否同步。
- ICMP 消息有一个 8 字节的头部和一个大小可变的数据部分
  - 头部中的两个重要字段：类型和代码

# ICMPv4 – 消息



Error-reporting messages



Query messages

## Type and code values

### Error-reporting messages

- 03: Destination unreachable (codes 0 to 15)
- 04: Source quench (only code 0)
- 05: Redirection (codes 0 to 3)
- 11: Time exceeded (codes 0 and 1)
- 12: Parameter problem (codes 0 and 1)

### Query messages

- 08 and 00: Echo request and reply (only code 0)
- 13 and 14: Timestamp request and reply (only code 0)

# ICMPv4 – 目的地不可达

- 使用最广泛的 **错误消息** 是目的地不可达（类型 3）。
- 示例： **类型 3**， **代码 0**：当我们使用 HTTP 协议访问网页时，如果服务器已关闭，则会生成“**目标主机不可到达**”的消息并发送回源地址。

# ICMPv4 — 超时消息，代码为 0

- 一条错误消息。
- **IP 数据报中生存时间**（**time-to-live**，**TTL**）字段的**用途**是防止数据报在互联网上无目的地循环转发。
- 当 TTL 值变为 0 时，访问路由器将丢弃该数据报，并向源端发送一条**超时消息**（**类型 11**，**代码 0**），以通知其当前情况。

# ICMPv4 – 回显请求与应答

- 一种查询消息。
- 回显请求（**类型 8**）和回显应答（**类型 0**）这一对消息被主机或路由器用来测试另一台主机或路由器的**活跃状态**。
- 一台主机或路由器向另一台主机或路由器发送回显请求消息；如果后者处于活动状态，则会以回显应答消息进行响应。
- 应用 → 诸如 **ping** 和 **traceroute** 之类的调试工具。

# 互联网上的ICMP与调试工具

## ping

- 用于检测主机是否在线并能够响应。
- 基于两种 **查询消息**（回显请求/回复）。

```
C:\Users>ping bcit.ca

Pinging bcit.ca [142.232.230.10] with 32 bytes of data:
Reply from 142.232.230.10: bytes=32 time=24ms TTL=245
Reply from 142.232.230.10: bytes=32 time=22ms TTL=245
Reply from 142.232.230.10: bytes=32 time=22ms TTL=245
Reply from 142.232.230.10: bytes=32 time=21ms TTL=245

Ping statistics for 142.232.230.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 21ms, Maximum = 24ms, Average = 22ms
```

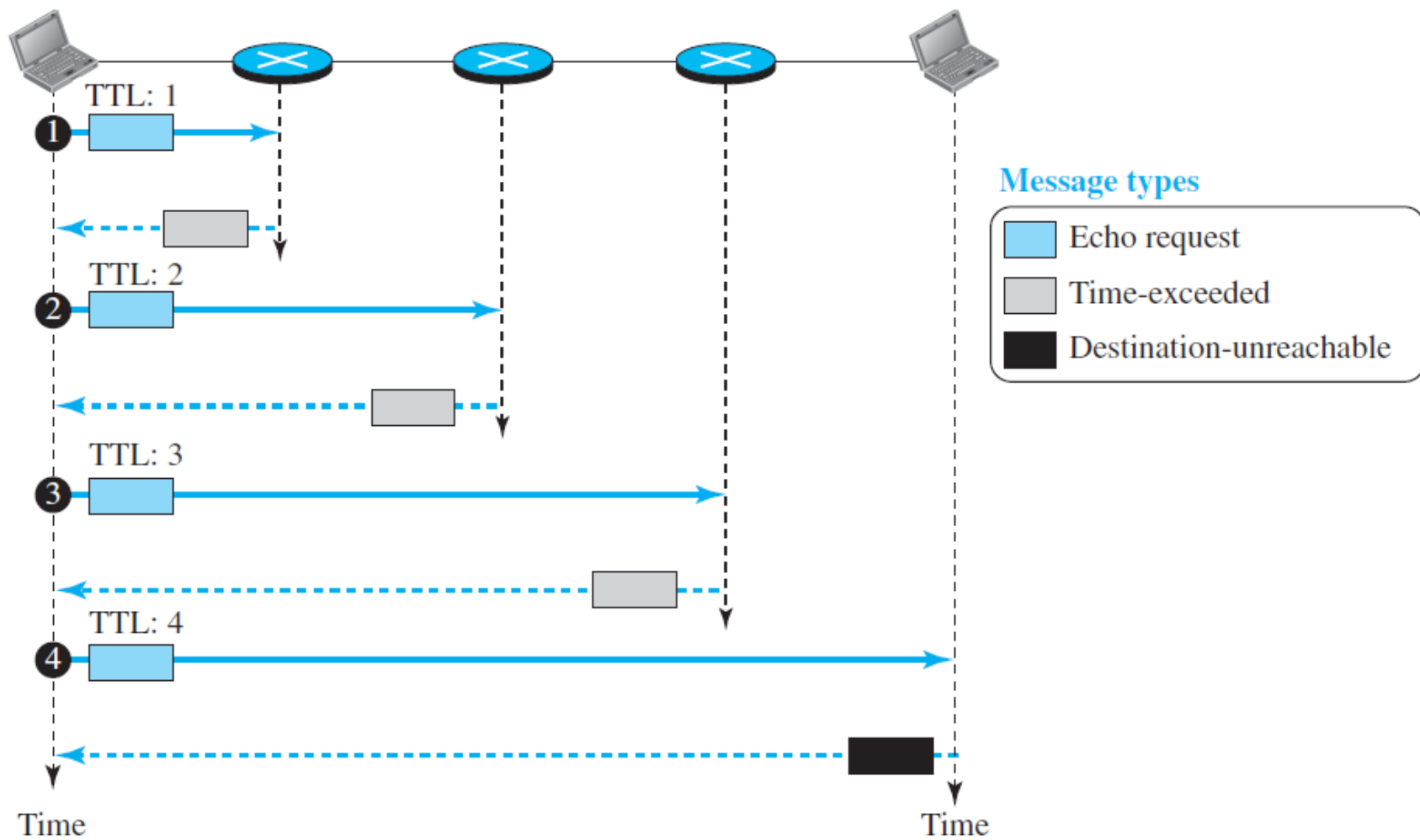


# 互联网上的ICMP与调试工具

- **traceroute/tracert**

- UNIX/Linux 中的 **traceroute** 程序或 **tracert** 命令（Windows系统中使用）。
- 基于两种 **错误报告消息**：**超时** 和 **目的地址-不可达**，以及 **回显请求查询消息**。
- 用于 **追踪** 数据包从源点到终点的 **路径**。
- 它可以找出沿路径访问的所有路由器的IP地址。
- 通常设置为最多检查 30 跳（路由器）的访问。

# 路由跟踪



# 路由跟踪

```
C:\Users>tracert bcit.ca
```

```
Tracing route to bcit.ca [142.232.230.10]  
over a maximum of 30 hops:
```

1	2 ms	1 ms	1 ms	192.168.1.254
2	5 ms	6 ms	5 ms	10.31.234.1
3	6 ms	5 ms	5 ms	154.11.10.159
4	6 ms	8 ms	5 ms	as6939.vanix.ca [206.41.104.27]
5	7 ms	6 ms	6 ms	bcnet.10gigabitethernet1-4.core1.yvr1.he.net [184.105.148.150]
6	21 ms	21 ms	21 ms	cr1-100g-bb3927ae8.srry1.bc.net [207.23.253.121]
7	21 ms	21 ms	21 ms	427-bcit-tx-cr1.vncv1.bc.net [207.23.240.25]
8	22 ms	22 ms	21 ms	ip-142-232-230-10.ptr.bcit.ca [142.232.230.10]

```
Trace complete.
```

# 摘要

- **IPv4** 是一种 **不可靠 无连接 协议**，负责从源到目的地的传输。
- IP 层（网络层）中的数据包称为 **数据报**（每个数据报包含一个头部和有效载荷）。
- IPv4 数据报在传输路径中可以被 **分片**。
- **校验和** 仅针对 **数据报头部** 进行计算。
- **ICMP** 消息被封装在 IP 数据报中。
- 两种ICMP消息类型： **差错报告** 和 **查询**。

# 参考文献

- [1] Behrouz A. Forouzan, Data Communications & Networking with TCP/IP Protocol Suite, 6th Ed, 2022, McGraw-Hill companies.
- [2] J.F. Kurose, K.W. Ross, Computer Networking: A Top-Down Approach, 7th Ed, 2017, Pearson Education, Inc.

# 阅读

- 教材第7章，7.4.2–7.4.4节。
- 教材第7章，第7.8节（练习测试）