



第 3 章 数据链路层

课程名称：计算机网络

主讲教师：姚烨

课程代码：U10M11016. 01

E-MAIL : yaoye@nwpu.edu.cn

第 11-12 讲

2021 – 2022 学年第一学期



本节课程位置

1. 概述

2. 物理层

3. 数据链路层

4. 局域网

5. 网络层

6. 传输层

7. 应用层

8. 广域网

9. 网络新技术



第 1 章 概述

■ 学习目的

- 数据链路层功能、通信双方如何从比特流中区分不同数据帧、数据帧可靠传输技术

■ 阅读材料

- 教材+参考教材（p65-p72; p197-p202）

■ 引导要点

- 成帧技术
- 可靠通信机制：检错与纠错机制，流量控制技术

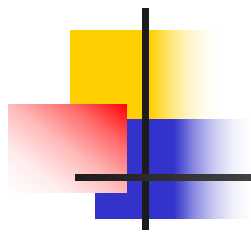
■ 编程任务

- 计算CRC校验码和简单校验和。
- 具体见PROJECT1（PROJECT1-1, PROJECT1-2）。
- 要求：设计、编程实现、技术报告。



第 3 章 数据链路层

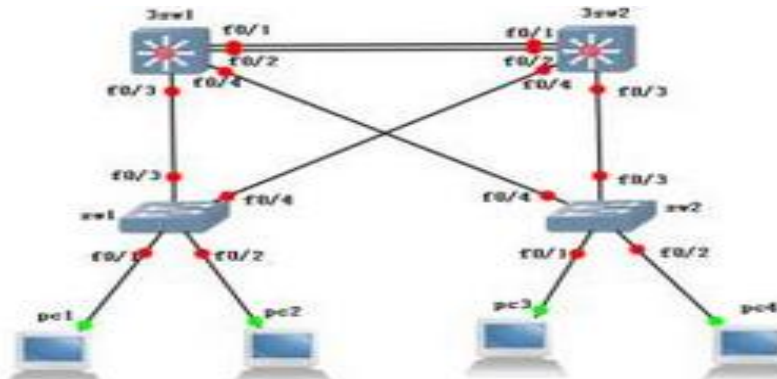
- 1. 成帧
- 2. 差错检测与纠错
- 3. 流量控制
 - 3.1 停止-等待协议
 - 3.2 连续ARQ协议



问题1：数据链路层研究什么问题，研究的目的是什么？

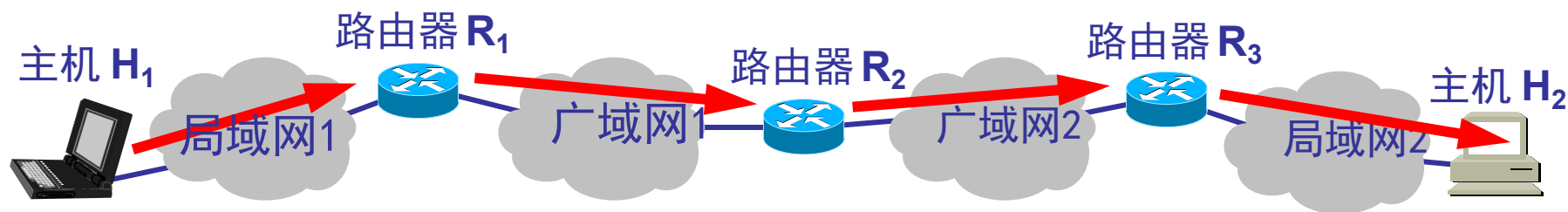
引言

- 物理链路(link)是一条**点到点物理线路**，中间没有任何其他交换节点。
 - 一条物理链路只是一条物理路径的组成部分。
- 数据链路(data link)：除物理链路外，还必须有通信协议控制物理链路的数据传输；若把实现这些协议的硬件和软件加到链路上，就构成了数据链路。
 - **目的：将不可靠的物理链路转变为可靠的数据链路。**
 - **数据链路(逻辑链路) = 物理链路 + 数据链路层通信协议。**
 - 使用适配器（即网卡, 或接口）实现这些协议的硬件和软件。
 - 一般适配器包括了物理层和数据链路层两层功能。

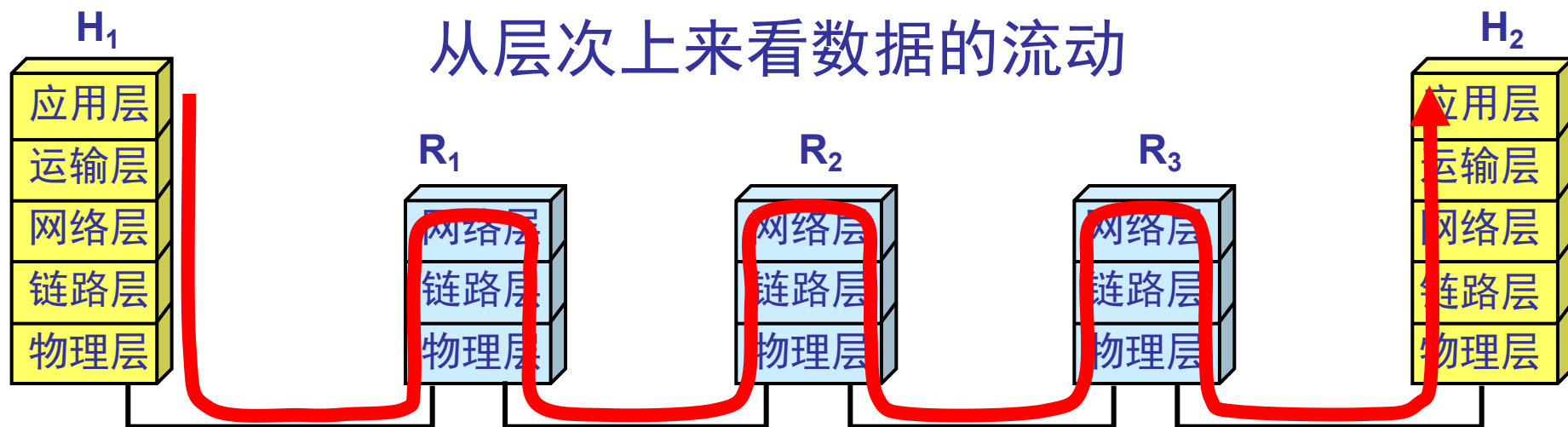


数据链路层简单模型

主机 H_1 向 H_2 发送数据



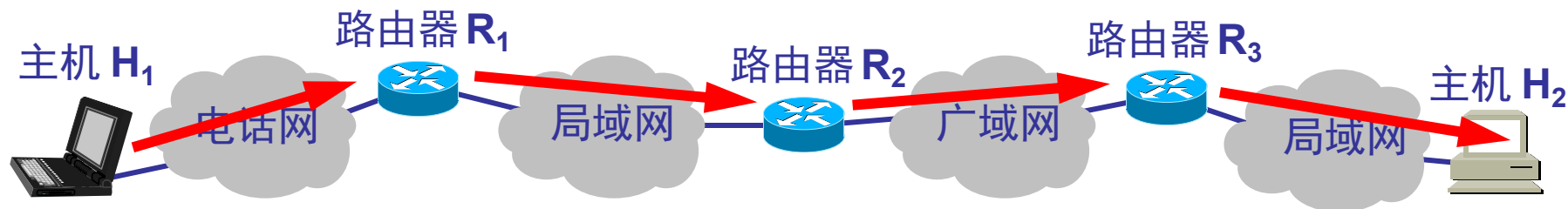
从层次上来看数据的流动



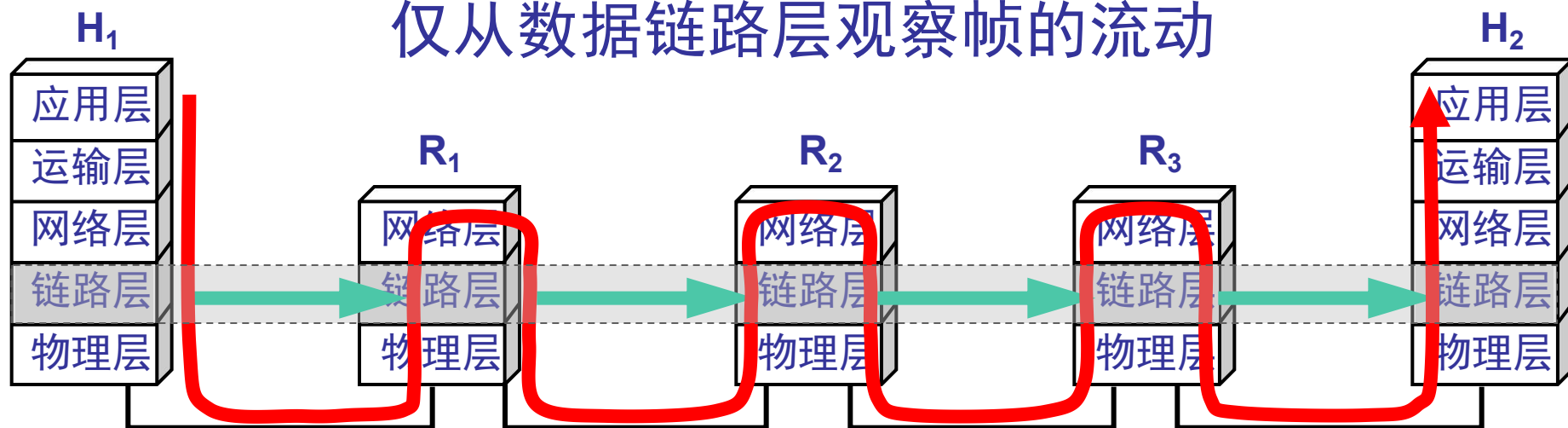
数据链路层的简单模型

(续)

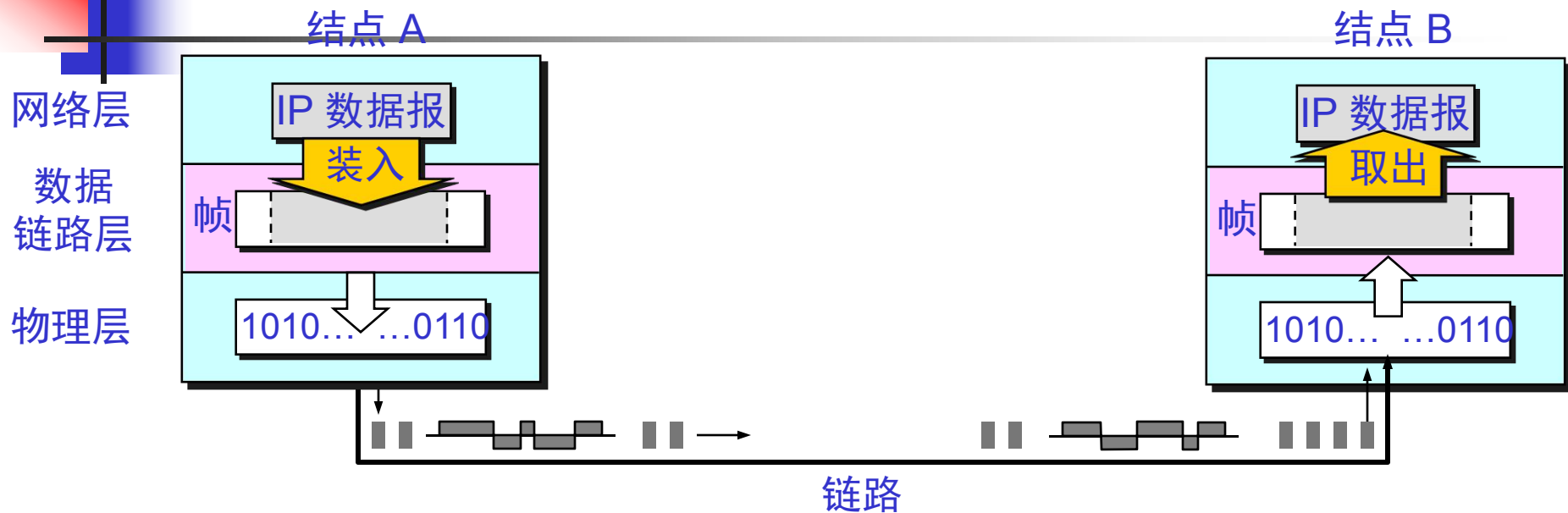
主机 H_1 向 H_2 发送数据



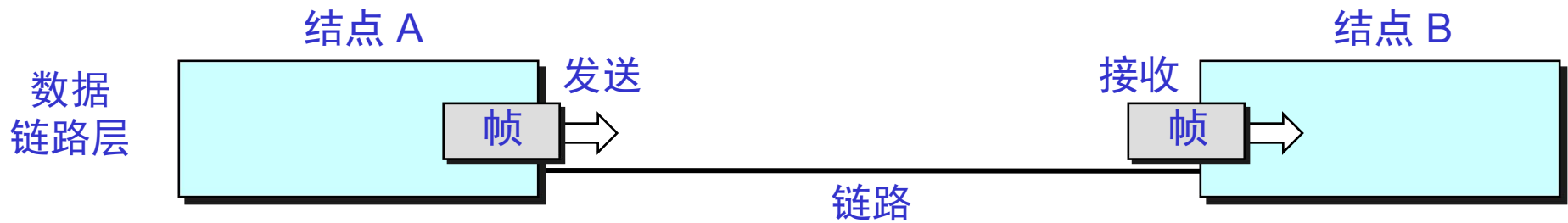
仅从数据链路层观察帧的流动



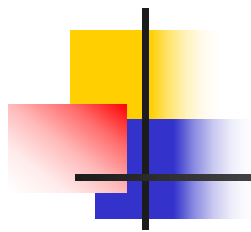
数据链路层传送PDU是帧



(a)



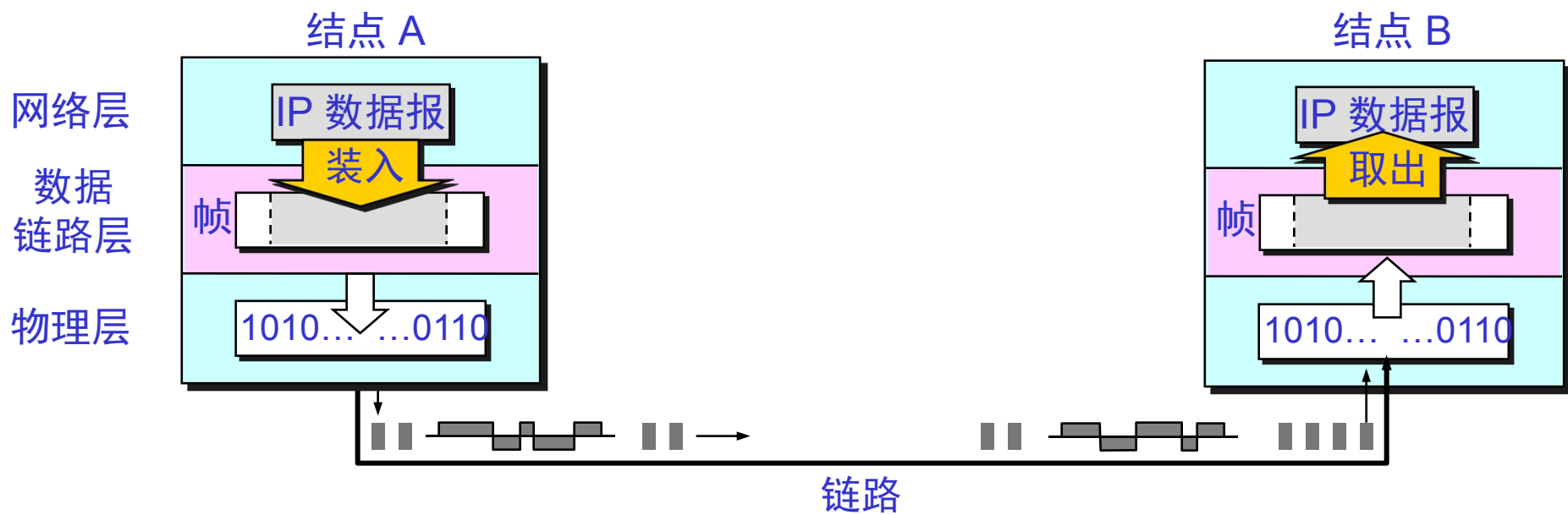
(b)



问题2：数据链路层要实现数据帧可靠传输，需要解决的三个基本问题是什么？

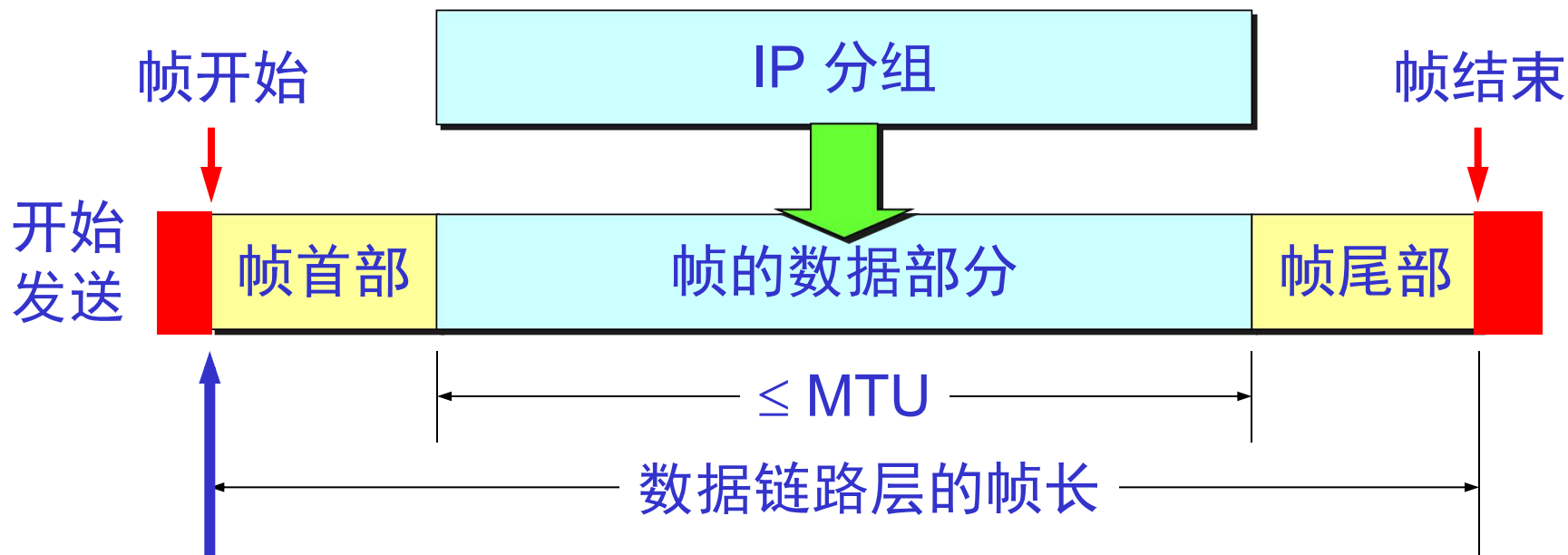
数据链路三个基本问题

- 1. 成帧规则
- 2. 差错控制（检错+纠错）
- 3. 流量控制



一、成帧

- 封装成帧 (framing)：在IP分组的前后分别添加首部和尾部，然后就构成了一个帧；
- 确定帧的界限？首部和尾部进行帧定界。





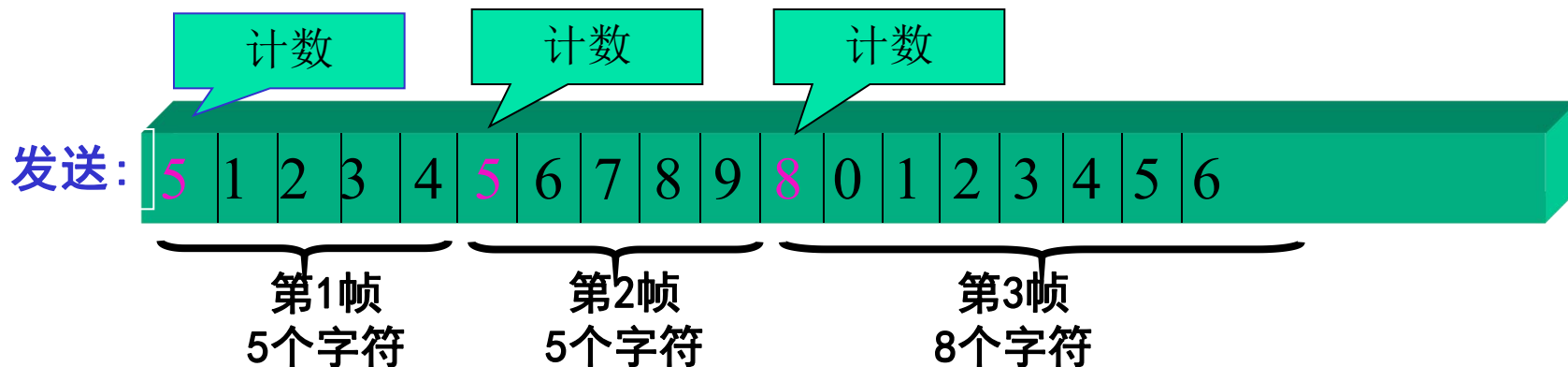
成帧定界四种方法

- (1) 字符计数法**
- (2) 字符填充分界符法** (面向字符同步传输)
- (3) 零比特填充分界符法** (面向比特流同步传输)
- (4) 物理层编码违例法**

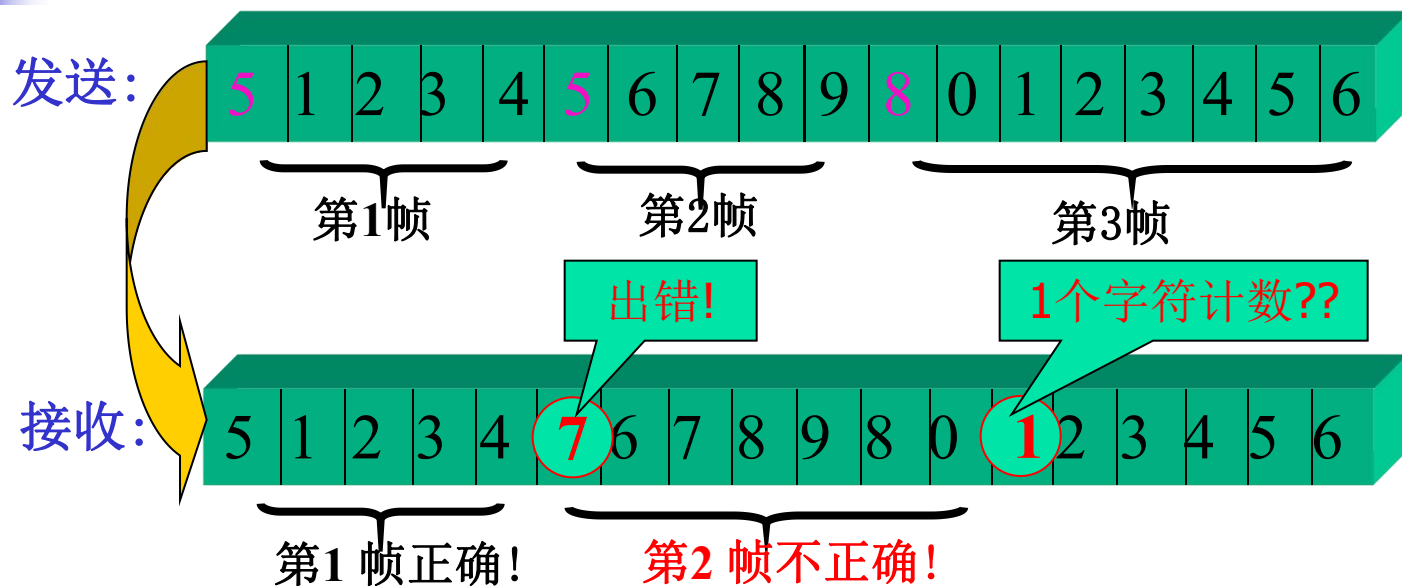
(1) 字符计数法

■ 具体方法

- 发送方利用帧首部一个字段作为帧长字段，按字符计数；
- 帧长字段：**帧包含的字符数+1**；
- 接收方利用帧长字段进行接收：利用帧长字段得知后面跟着多少个字符，知道了帧结束地方。



(1) 字符计数法

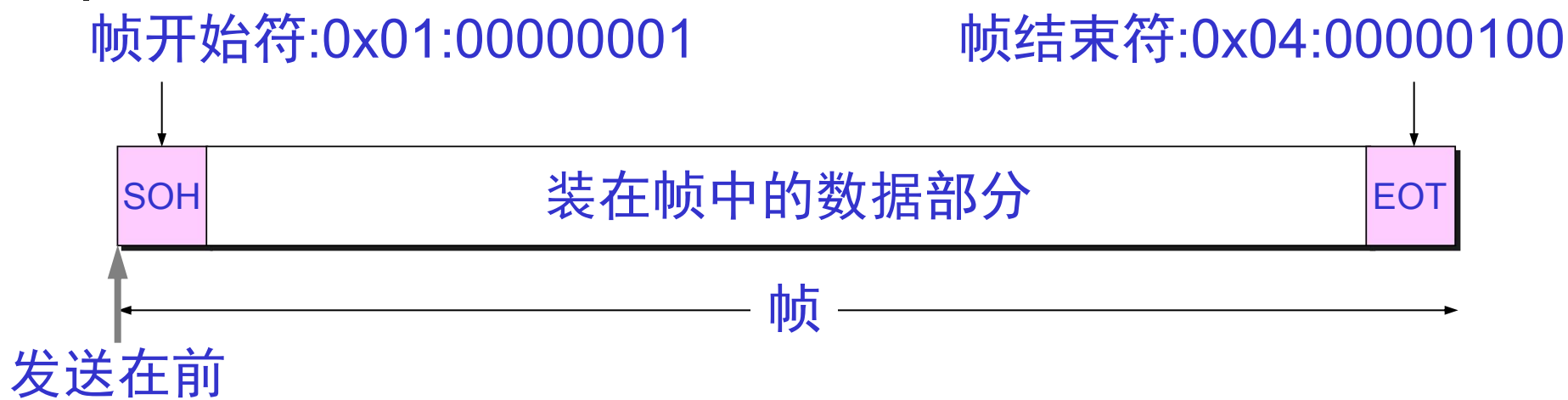


■ 问题:

- 如果帧长度字段值错误, 造成后面所有数据接收错误;
- 无法检错: 接收方还不知道出错;
- 无法纠错: 发送方无法利用重发机制进行纠错。

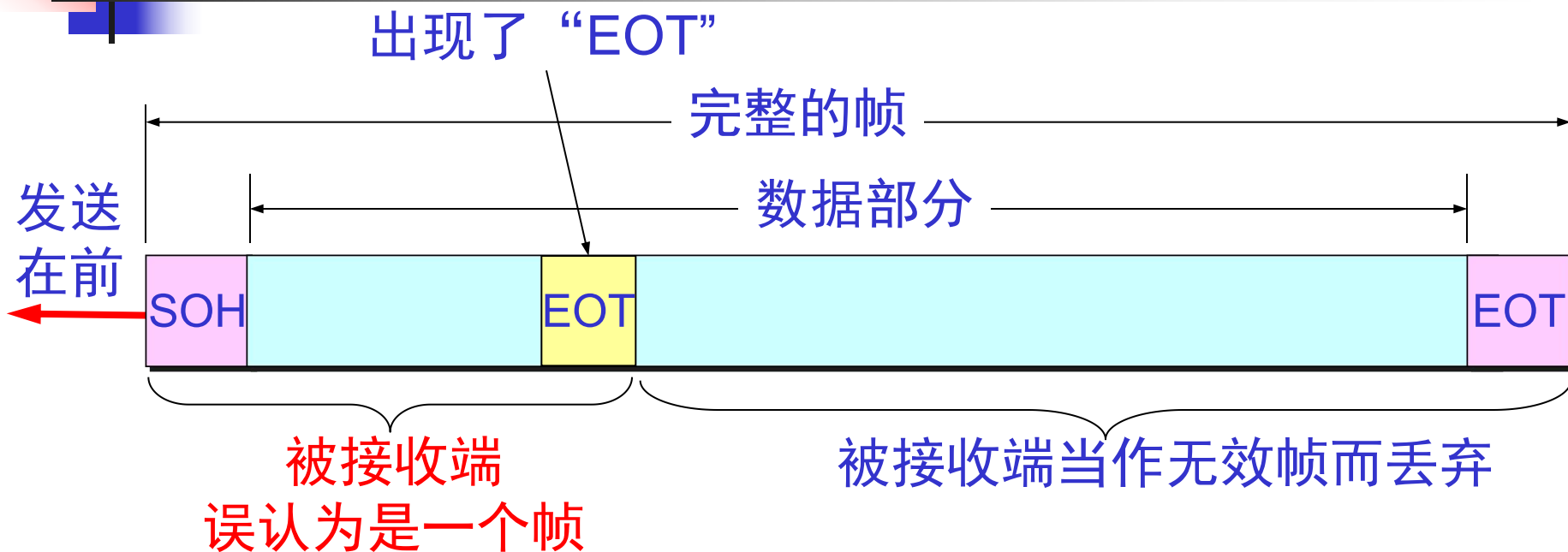
(American Standard Code for Information Interchange 美国标准信息交换代码)																										
ASCII控制字符													ASCII可打印字符													
十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符
0	00	NUL	1	01	SOH	2	02	STX	3	03	ETX	4	04	TX	5	05	LF	6	06	VT	7	07	FF	8	08	
9	09	HT	10	0A	NL	11	0B	SH	12	0C	DC1	13	0D	DC2	14	0E	DC3	15	0F	DC4	16	10	DC5	17	11	
18	12	DC6	19	13	DC7	20	14	DC8	21	15	DC9	22	16	DC10	23	17	DC11	24	18	DC12	25	19	DC13	26	1A	
27	1B	DC14	28	1C	DC15	29	1D	DC16	30	1E	DC17	31	1F	DC18	32	20	SP	33	21	!	34	22	"	35	23	
36	24	#	37	25	\$	38	26	%	39	27	&	40	28	'	41	29	(42	2A	*	43	2B	+	44	2C	
45	2D	,	46	2E	.	47	2F	:	48	30	0	49	31	1	50	32	2	51	33	3	52	34	4	53	35	
54	36	5	55	37	6	56	38	7	57	39	8	58	3A	:	59	3B	;	60	3C	<	61	3D	=	62	3E	
63	3F	>	64	40	@	65	41	A	66	42	B	67	43	C	68	44	D	69	45	E	70	46	F	71	47	
72	48	G	73	49	H	74	4A	I	75	4B	J	76	4C	K	77	4D	L	78	4E	M	79	4F	N	80	50	
81	51	O	82	52	P	83	53	Q	84	54	R	85	55	S	86	56	T	87	57	U	88	58	V	89	59	
90	5A	W	91	5B	X	92	5C	Y	93	5D	Z	94	5E	[95	5F	\	96	60	`	97	61	a	98	62	
99	63	b	100	64	c	101	65	d	102	66	e	103	67	f	104	68	g	105	69	h	106	6A	i	107	6B	
108	6C	j	109	6D	k	110	6E	l	111	6F	m	112	70	n	113	71	o	114	72	p	115	73	q	116	74	
117	75	r	118	76	s	119	77	t	120	78	u	121	79	v	122	7A	w	123	7B	x	124	7C	y	125	7D	
126	7E	~	127	7F	DEL	128	80		129	81		130	82		131	83		132	84		133	85		134	86	
135	87		136	88		137	89		138	8A		139	8B		140	8C		141	8D		142	8E		143	8F	
144	90		145	91		146	92		147	93		148	94		149	95		150	96		151	97		152	98	
153	99		154	9A		155	9B		156	9C		157	9D		158	9E		159	9F		160	A0		161	A1	
162	A2		163	A3		164	A4		165	A5		166	A6		167	A7		168	A8		169	A9		170	AA	
171	AB		172	AC		173	AD		174	AE		175	AF		176	B0		177	B1		178	B2		179	B3	
180	B4		181	B5		182	B6		183	B7		184	B8		185	B9		186	BA		187	BB		188	BC	
189	BD		190	BE		191	BF		192	C0		193	C1		194	C2		195	C3		196	C4		197	C5	
198	C6		199	C7		200	C8		201	C9		202	CA		203	CB		204	CC		205	CD		206	CE	
207	CF		208	D0		209	D1		210	D2		211	D3		212	D4		213	D5		214	D6		215	D7	
216	D8		217	D9		218	DA		219	DB		220	DC		221	DD		222	DE		223	DF		224	DE	
225	E0		226	E1		227	E2		228	E3		229	E4		230	E5		231	E6		232	E7		233	E8	
234	E9		235	EA		236	EB		237	EC		238	ED		239	EE		240	EF		241	EF		242	EF	
243	EF		244	EF		245	EF		246	EF		247	EF		248	EF		249	EF		250	EF		251	EF	
252	EF		253	EF		254	EF		255	EF		256	EF		257	EF		258	EF		259	EF		260	EF	
261	EF		262	EF		263	EF		264	EF		265	EF		266	EF		267	EF		268	EF		269	EF	
270	EF		271	EF		272	EF		273	EF		274	EF		275	EF		276	EF		277	EF		278	EF	
279	EF		280	EF		281	EF		282	EF		283	EF		284	EF		285	EF		286	EF		287	EF	
288	EF		289	EF		290	EF		291	EF		292	EF		293	EF		294	EF		295	EF		296	EF	
297	EF		298	EF		299	EF		300	EF		301	EF		302	EF		303	EF		304	EF		305	EF	
306	EF		307	EF		308	EF		309	EF		310	EF		311	EF		312	EF		313	EF		314	EF	
315	EF		316	EF		317	EF		318	EF		319	EF		320	EF		321	EF		322	EF		323	EF	
324	EF		325	EF		326	EF		327	EF		328	EF		329	EF		330	EF		331	EF		332	EF	
333	EF		334	EF		335	EF		336	EF		337	EF		338	EF		339	EF		340	EF		341	EF	
342	EF		343	EF		344	EF		345	EF		346	EF		347	EF		348	EF		349	EF		350	EF	
351	EF		352	EF		353	EF		354	EF		355	EF		356	EF		357	EF		358	EF		359	EF	
360	EF		361	EF		362	EF		363	EF		364	EF		365	EF		366	EF		367	EF		368	EF	
369	EF		370	EF		371	EF		372	EF		373	EF		374	EF		375	EF		376	EF		377	EF	
378	EF		379	EF		380	EF		381	EF		382	EF		383	EF		384	EF		385	EF		386	EF	
387	EF		388	EF		389	EF		390	EF		391	EF		392	EF		393	EF		394	EF		395	EF	
396	EF		397	EF		398	EF		399	EF		400	EF		401	EF		402	EF		403	EF		404	EF	
405	EF		406	EF		407	EF		408	EF		409	EF		410	EF		411	EF		412	EF		413	EF	
414	EF		415	EF		416	EF		417	EF		418	EF		419	EF		420	EF		421	EF		422	EF	
423	EF		424	EF		425	EF		426	EF		427	EF		428	EF		429	EF		430	EF		431	EF	
432	EF		433	EF		434	EF		435	EF		436	EF		437	EF		438	EF		439	EF		440	EF	
441	EF		442	EF		443	EF		444	EF		445	EF		446	EF		447	EF		448	EF		449	EF	
450	EF		451	EF		452	EF		453	EF		454	EF		455	EF		456	EF		457	EF		458	EF	
459	EF		460	EF		461	EF		462	EF		463	EF		464	EF		465	EF		466	EF		467	EF	
468	EF		469	EF		470	EF		471	EF		472	EF		473	EF		474	EF		475	EF		476	EF	
477	EF		478	EF		479	EF		480	EF		481	EF		482	EF		483	EF		484	EF		485	EF	
486	EF		487	EF		488	EF		489	EF		490	EF		491	EF		492	EF		493	EF		494	EF	
495	EF		496	EF		497	EF		498	EF		499	EF		500	EF		501	EF		502	EF		503	EF	
504	EF		505	EF		506	EF		507	EF		508	EF		509	EF		510	EF		511	EF		512	EF	
513	EF		514	EF		515	EF		516	EF		517	EF		518	EF		519	EF		520	EF		521	EF	
522	EF		523	EF		524	EF		525	EF		526	EF		527	EF		528	EF		529	EF		530	EF	
531	EF		532	EF		533	EF		534	EF		535	EF		536	EF		537	EF		538	EF		539	EF	
540	EF		541	EF		542	EF		543	EF		544	EF		545	EF		546	EF		547	EF		548	EF	
549	EF		550	EF		551	EF		552	EF		553	EF		554	EF		555	EF		556	EF		557	EF	
558	EF		559	EF		560	EF		561	EF		562	EF		563	EF		564	EF		565	EF		566	EF	
567	EF		568	EF		569	EF		570	EF		571	EF		572	EF		573	EF		574	EF		575	EF	
576	EF		577	EF		578	EF		579	EF		580	EF		581	EF		582	EF		583	EF		584	EF	
585	EF		586	EF		587	EF		588	EF		589	EF		590	EF		591	EF		592	EF		593	EF	
594	EF		595	EF		596	EF		597	EF		598	EF		599	EF										

用控制字符进行帧定界的方法举例



- 当传输的帧中数据是文本文件：所有字符从键盘可输入；
- 可实现透明传输：不管从键盘输入什么样字符均可放在帧中传输。
- 数据部分不会出现现象SOH，EOT等的帧定界控制字符。

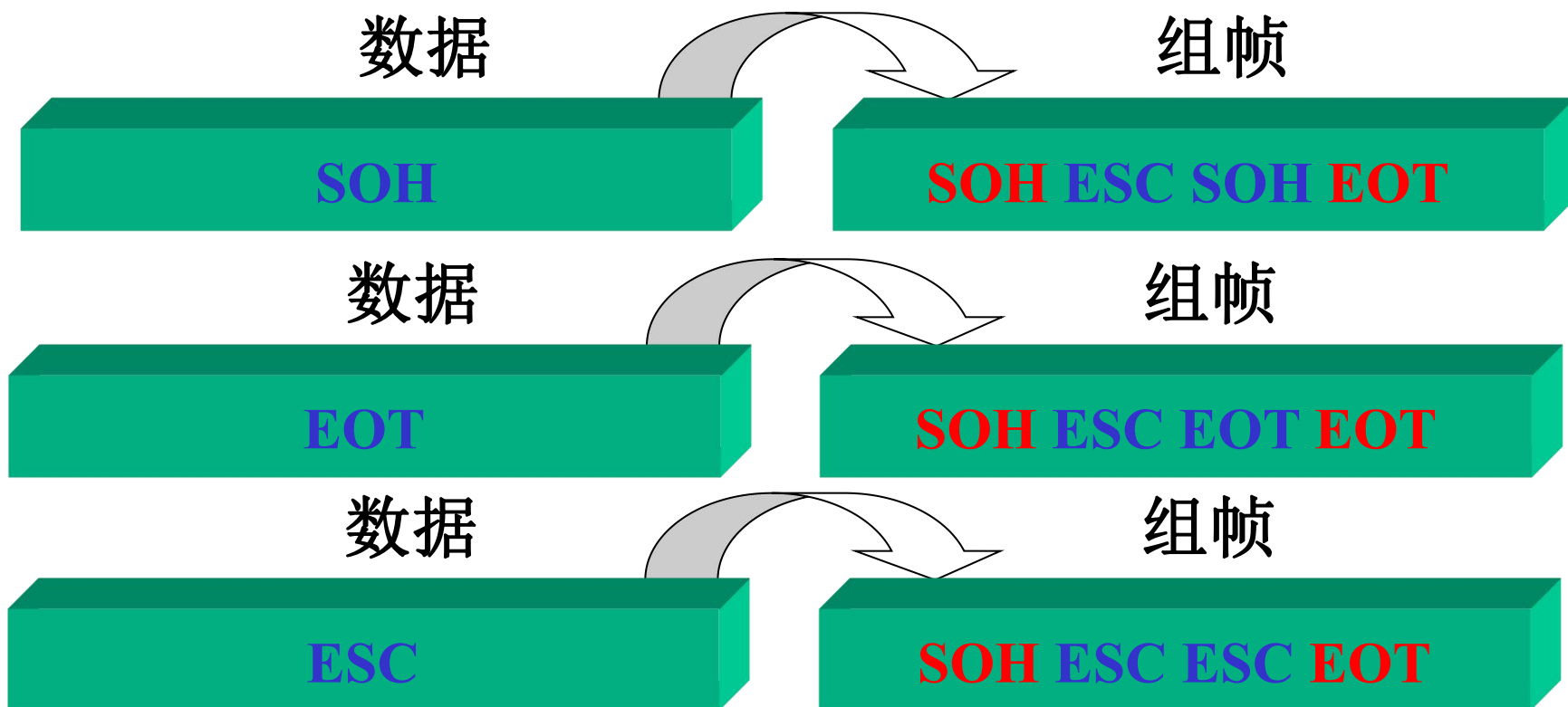
透明传输



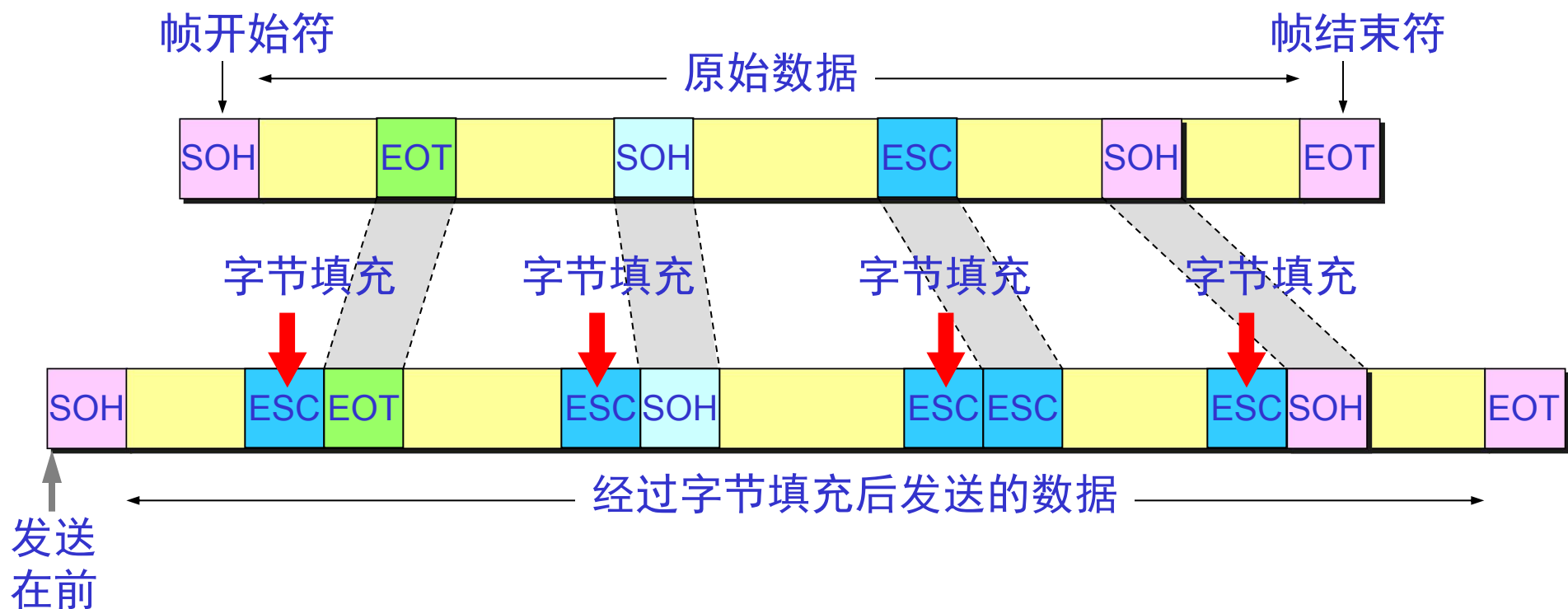
- 当数据是非文本文件（如可执行文件、或多媒体数据），则数据部分可能出现SOH或EOT的比特组合数据，造成接收方帧定界错误。

用字节填充法解决透明传输的问题

- 发送方数据中出现**标志字符**, 采用在该数据前插入一个特殊的转义字符 (ESC)。
- 接收方在数据链路层删掉转义字符即可。

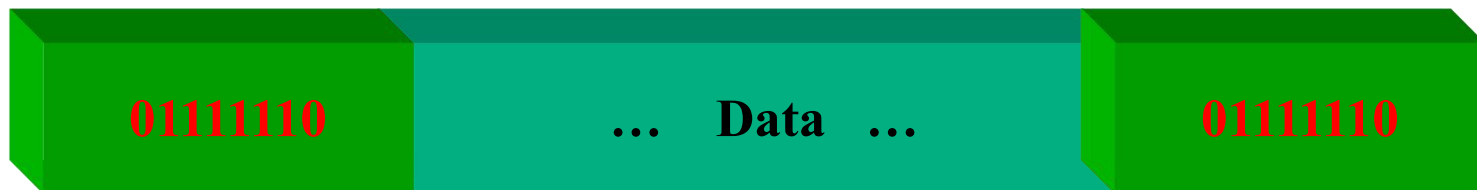


用字节填充法解决透明传输的问题



(3) 零比特填充充分界符法

- 面向位流的同步传输：数据块包含任意长度和组合的比特位。
- 具体方法
 - 每一帧的开始和结束都有一个特殊的位模式01111110。

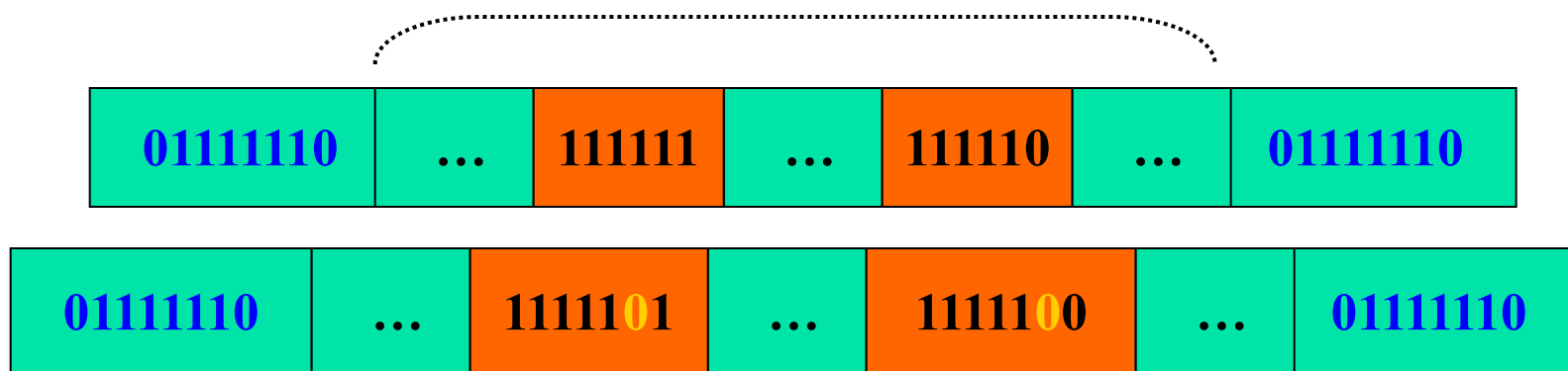


- 接收方通过位模式可识别出一个帧的边界。
- 连续的两个帧之间只需要一个位模式01111110。
- 如果数据帧丢失了位模式，接收方只需在数据流中扫描下一个位模式即可。

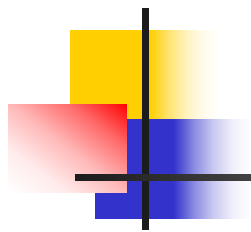


(3) 零比特填充充分界符法

Data



- 发送方：“逢五插0”，数据部分遇到五个1，在后面加一个0；
- 接收方：“逢五删0”，遇到五个1，如果后面是0，表示数据部分，删除0；如果是1，表示位模式。



问题3：利用编码技术，
可否在物理层标记不同
数据帧？

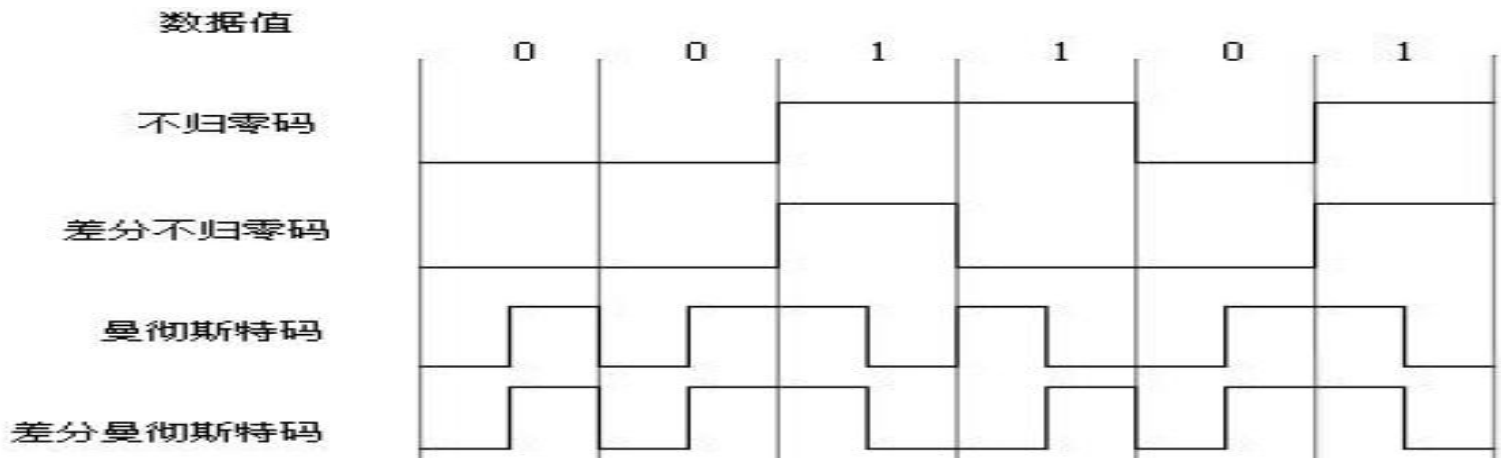
1.4 物理层编码违例法

■ 基本思路

- 数据位中不可能出现比特编码作为帧起始或结束定界符。

■ 举例

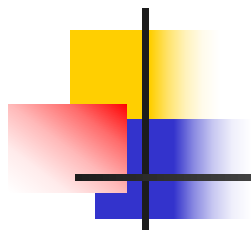
- 曼彻斯特编码中, 一个码元中间电平跳变表示0（正向跳变：低→高）和1（负向跳变：高→低）；
- 差分曼彻斯特编码：中间必跳变；0：开始有跳变；1：开始无
- 违例编码不用于表示数据，但可以在某些协议中用于表示数据帧的开始或结束边界。





二、差错控制（检错与纠错）

- 1. 基本概念
- 2. 差错检测（检错）：校验码
 - （1）奇偶校验
 - （2）CRC循环冗余校验
- 3. 差错纠正（纠错）



问题4：数据帧在网络中传输时，会出现差错，出现差错的原因有哪些？



1. 基本概念

- 通信系统基本任务：高效率而无差错地传送数据，但在任何一种通信线路上都不可避免地存在一定程度的噪声。
- 信道中存在两种差错：随机差错和突发差错, 均称为**比特差错**

种类	随机差错	突发差错
来源	随机热噪声（白噪声）	冲击噪声
特点1	信道固有的、持续存在的	外界的因素，持续时间短，突发性
特点2	码元的差错是独立的，和前后的码元无相关性	差错具有相关性，数据传输中产生差错主要原因

突发差错长度：差错发生第一个码元到有错最后一个码元间所有码元的数。

- 
- 误码率是衡量物理信道的通信质量的一个指标

$$P_e = \frac{\text{错误比特数}}{\text{总比特数}}$$

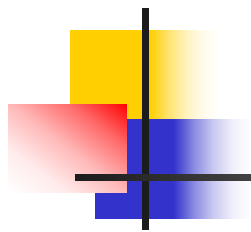
问题5：数据帧在网络中传输时，当出现比特差错，如何检错？



1. 基本概念

- 差错控制编码：数据位(k) + 校验码(r)，数据帧必有校验字段
 - 校验码：分为检错码和纠错码
 - 检错码：自动发现差错.
 - 纠错码：不仅能发现差错而且能够自动纠正.
 - 编码效率 R ：编码中有效信息位所占的比例： $R=k/(k+r)$

漏检率：某比特位出错但接收者无法检测到的概率。



问题6: 你知道的检错码
有哪些, 如何计算?



2. 差错检测

- 常用的两个检错码
 - 奇偶校验码；
 - CRC循环冗余校验码。
- 奇偶校验码
 - 增加一比特位校验位使得数据+校验位中 ‘1’的个数为奇数或者偶数。
 - 奇校验码：增加一比特位校验位（可为0或1）使得数据+校验位中 ‘1’的个数为奇数。
 - 偶校验码：增加一比特位校验位（可为0或1）使得数据+校验位中 ‘1’的个数为偶数。

奇校验例子

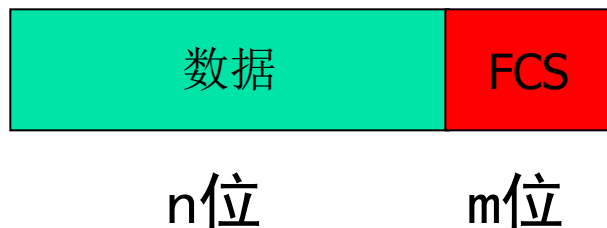
	字符1	字符2	字符3	字符4	字符5	字符6	字符7	字符8	校验 字符
b1	1	1	1	1	1	1	1	1	1
b2	0	1	1	0	0	0	0	1	0
b3	0	0	0	1	0	0	0	1	1
b4	0	0	1	0	0	0	0	0	0
b5	1	1	1	1	1	1	1	1	1
b6	0	0	0	0	0	1	0	0	0
b7	1	1	1	1	1	1	1	1	1
check	0	1	0	1	0	1	0	0	0

- 串行传输
 - 校验码
- 并行传输+线路
 - 纠错码
- 概率检测：可检测出信息传输过程中的部分误码（奇数位误码能检出，偶数位误码不能检出）；
- 不能纠错：发现错误后只能要求重发。
- 由于其实现简单，仍得到了广泛使用。



(2) 循环冗余校验: CRC

- 基本思想: (假设FCS校验码为: m 比特位)
 - 发送方: 选要发送的数据为被除数 (左移 m 位, 或补 m 个0), 选择一个预定的二进制数 ($m+1$ 位) 作为除数; 利用二进制模二除法运算, 计算得到一个余数, 作为CRC校验码 (FCS), 随数据帧发送给接收方。
 - 接收方: 以发送方发送的数据-CRC校验码为被除数, 选同样的预定的二进制码为除数, 如果余数为0, 表明没有差错 (接近概率1), 否则证明发生差错。



循环冗余校验码: CRC (Cycle Redundancy Check)

- 预定的二进制码（除数）

- 是一个标准的国际编码, 用户不能任意规定;
- 最高位和最低位为1;
- 以多项式形式表示, 称为**生成多项式G(X)**.
- LAN(ethernet, Toking Ring)采用CRC-32, HDLC协议采用CRC-CCIT, ATM采用CRC-8, CRC-10, 和CRC-12.
- 发送数据长度大于生成多项式表示的二进制码长度.
 - CRC-8: $G(x)=x^8 + x^2 + x + 1$ ----- (100000111)
 - CRC-10: $G(x)=x^{10} + x^9 + x^5 + x^4 + x + 1$
 - CRC-12: $G(x)=x^{12} + x^{11} + x^3 + x^2 + x + 1$
 - CRC-16: $G(x)=x^{16} + x^{15} + x^2 + 1$
 - CRC-CCIT: $G(x)=x^{16} + x^{12} + x^5 + 1$
 - CRC-32 $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

CRC的例子 (1)

- 要发送的二进制数序列为“1010001”，7位的数据序列对应6次多项式：

$$\text{数据多项式: } K(x) = x^6 + x^4 + 1$$

- 选定的生成多项式为：

$$G(x) = x^4 + x^2 + x + 1$$

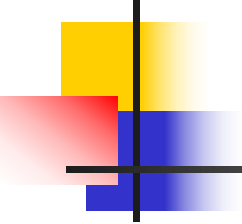
(最高次数为4, 除数 (5比特) : 10111)

- 被除数多项式为：

$$K(x) = x^4 K(x) = x^{10} + x^8 + x^4$$

(被除数: 10100010000) ----左移四位

CRC的例子


$$\begin{array}{r} 10111 \overline{) 10011111} \\ \underline{10111} \\ 11010 \\ \underline{10111} \\ 11010 \\ \underline{10111} \\ 11010 \\ \underline{10111} \\ 1101 \end{array}$$

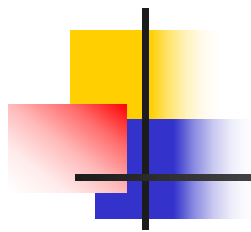
- 多项式除法后得到冗余码为: **1101**，所以相应的数据发送序列为: 1010001**1101**



CRC的检错能力讨论

- 可检测出全部单一位出现差错。
- 只要 $G(X)$ 中含有一个至少 3 项的因子, 可以检测出所有两位出错的情况.
- 只要 $G(X)$ 中含有因子 $(x+1)$ 时, 可检测出全部奇数个位出现差错.
- r 为生成多项式的最高幂次, 可检测出**突发差错长度**小于或等于 r 的所有突发性差错.
- 以 $1 - (1/2)^{r-1}$ 的概率检出突发差错长度大于 r 位的突发性差错.

举例: 如果 $r=16$, 则该CRC校验码能全部检查出**突发差错长度**小于或等于16 位的所有的突发差错, 并能以 $1 - (1/2)^{16-1} = 99.997\%$ 的概率检查出**突发差错长度**大于16位的突发性差错, 漏检概率为0.003%;



问题7：数据帧在网络传输过程中出现了差错，如何纠错？



(3) 纠错机制

■ 差错纠正的方法有两种：

□ (1) 反馈重发纠错法ARQ: Automatic Request for Repeat:

- 接收方检测错误，丢弃出错帧，发送方对出错帧重传；
- 前提条件：（1）**发送方知道**哪个帧错了，需要给每个数据帧编**序号**；（2）发送方**缓存**已发送的数据帧。
- **这是重点，具体流量控制部分讲**

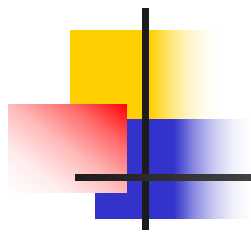
□ (2) 前向纠错(FEC)法: Forward Error Correction

- 接收方利用纠错码（**海明威码**）不仅可以检测差错，而且知道错误的位置，从而改正错误。
- 优点：无需重发；
- 缺点：编码效率低，算法比较复杂，实现比较困难，故很少使用。



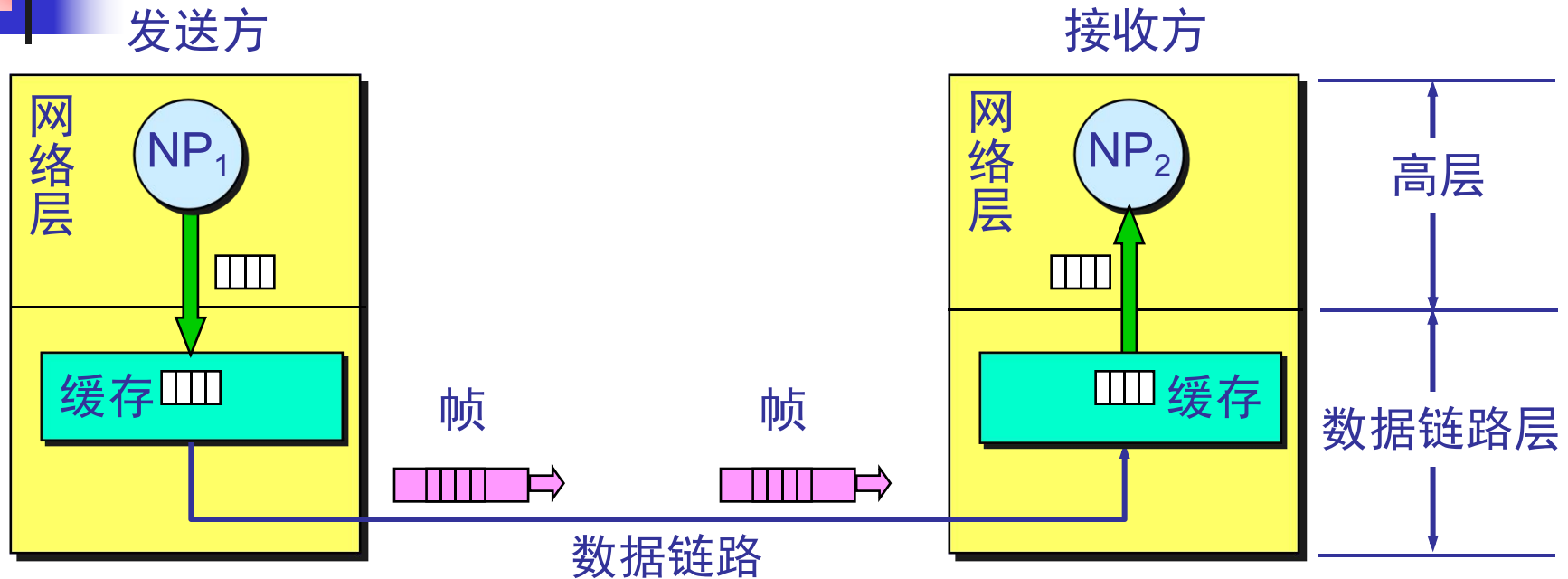
第 3 章 数据链路层

- 1. 成帧
- 2. 差错检测与纠错
- 3. 流量控制
 - 3.1 停止-等待协议
 - 3.2 连续ARQ协议



问题8：数据链路层流量
控制技术研究什么问题？
研究对象和目标是什么？

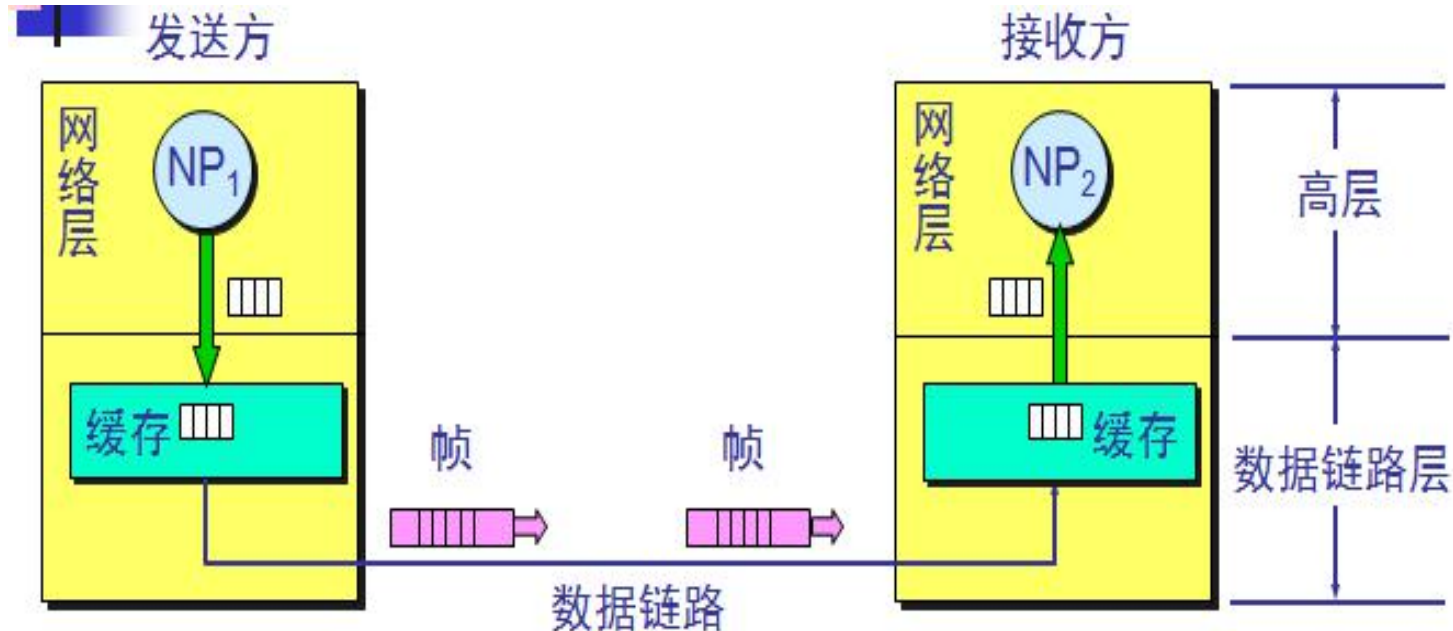
完全理想化的数据传输 信道基于的两个假定



- 假定 1：无噪声：数据帧既不会出差错（比特差错），**也不会丢失（传输差错）**？——差错控制问题
- 假定 2：大缓存：不管发送方以多快速率发送数据，接收方总是来得及收下，并及时向上层交付。——速率匹配问题
- 假定2相当于认为：接收方向上层交付分组的速率永远不会低于发送方发送数据帧速率。

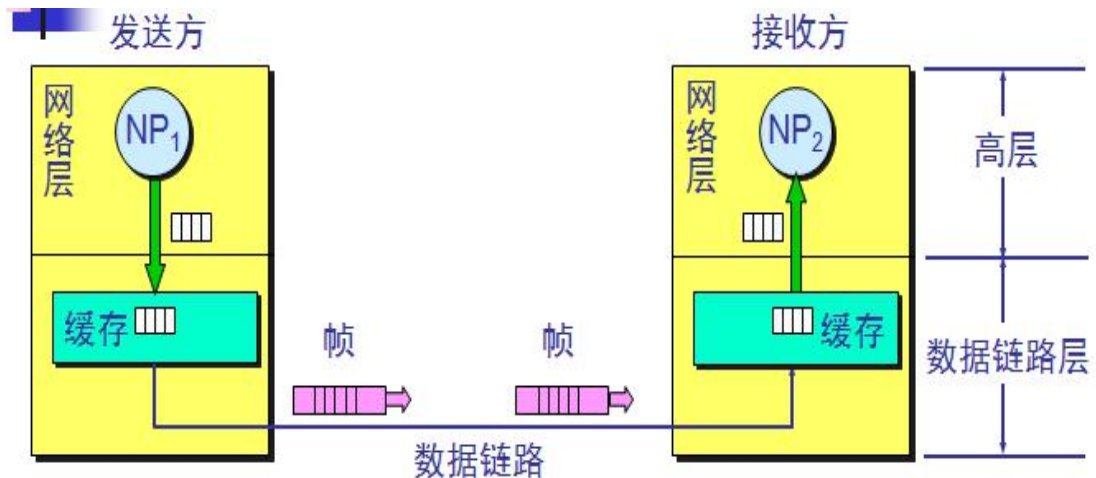
流量控制问题的提出

- 实际网络环境以上两个假设均不成立。
- 假设1：会出现无比特和传输差错——差错控制
 - 检错：校验码
 - 纠错：序号+确认反馈+超时重传机制（ARQ机制：反馈重发机制）。



流量控制问题的提出

- 假设2：大缓存不可能（流量控制）
 - 为了解决发送方发送速率和接收方交付速率匹配问题，一般采用方法：利用接收方交付速率控制发送方的发送速率，涉及到流量控制问题；在数据链路层一般采用**固定大小的滑动窗口技术**来实现。
- 现在的ARQ机制=差错控制+流量控制
 - 停止-等待仅仅实现流量控制，**如何改进实现差错控制？**
 - 连续ARQ协议：**后退N帧协议和选择重发协议**，在实现流量控制同时，实现了差错控制。





问题9：数据链路层一般都采用哪些流量控制技术？

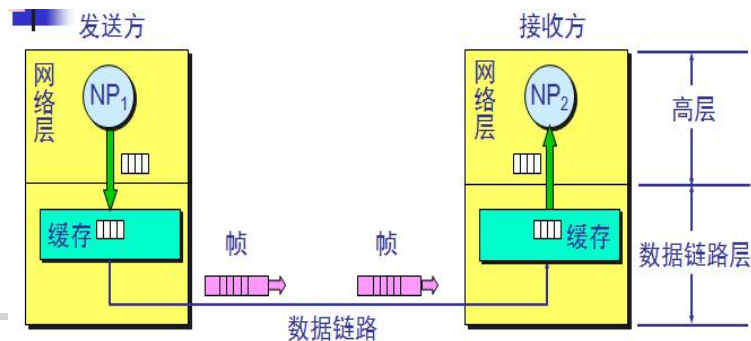


三、流量控制

固定大小滑动窗口技术

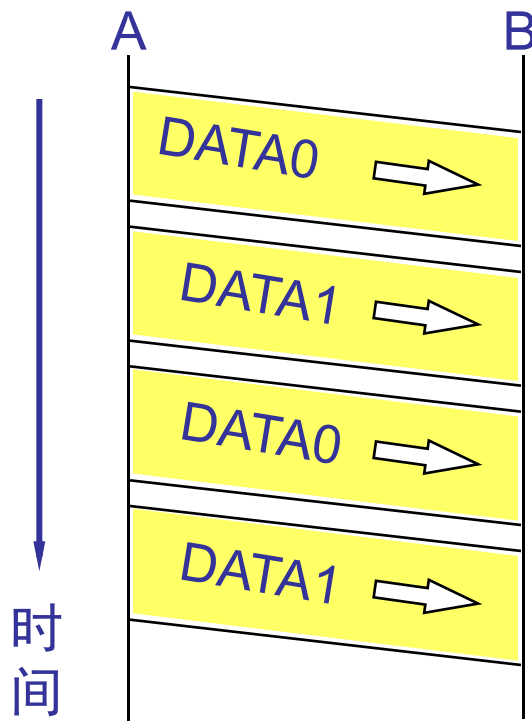
- (1) 停止-等待协议 $W_T=1$ $W_R=1$
- (2) 后退N帧协议 $W_T>1$ $W_R=1$
- (3) 选择重发协议 $W_T>1$ $W_R>1$

(1) 停止-等待协议

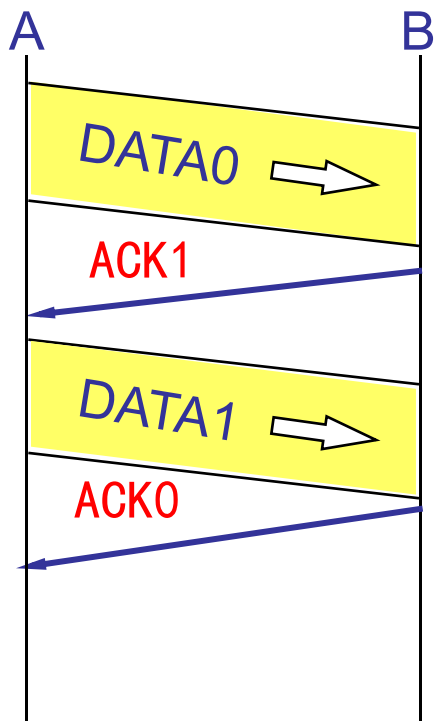


不需要流量控制

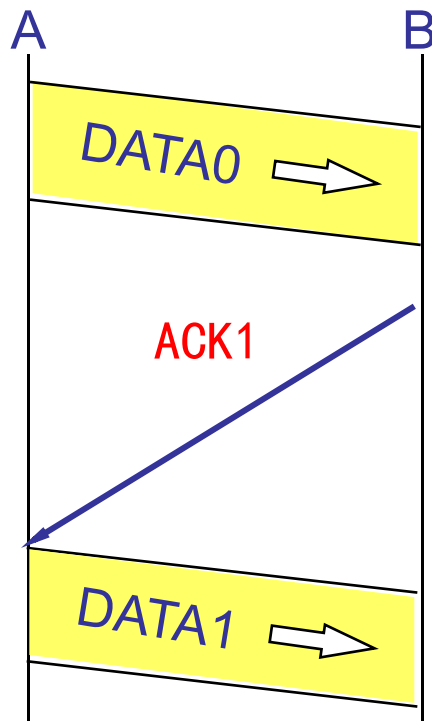
流量控制



接收队列足够大
(无差错)



停止-等待协议 (队列有限, 仅流量控制)



(1) 停止-等待协议（无差错控制）

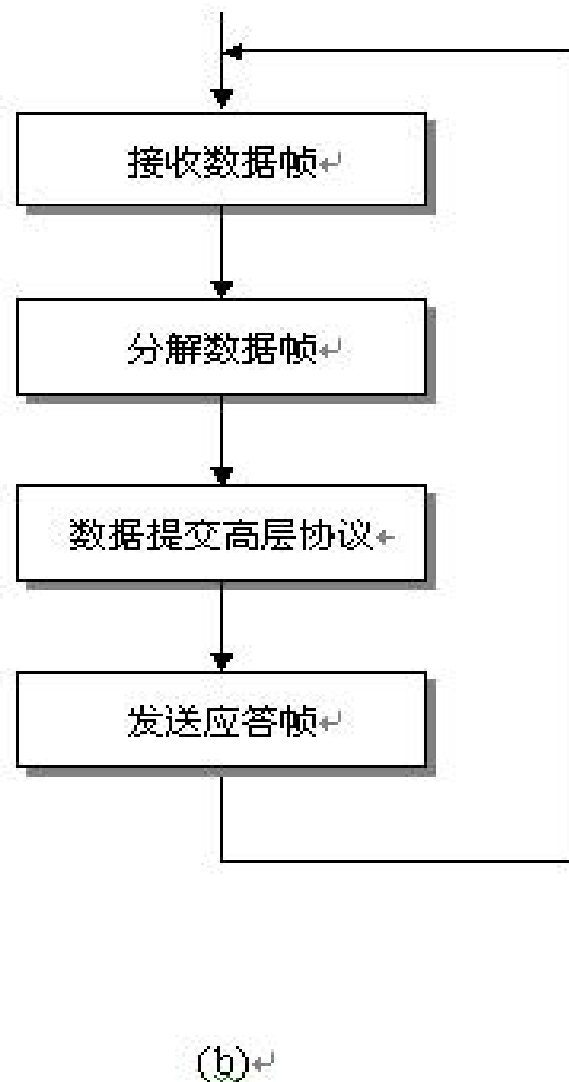
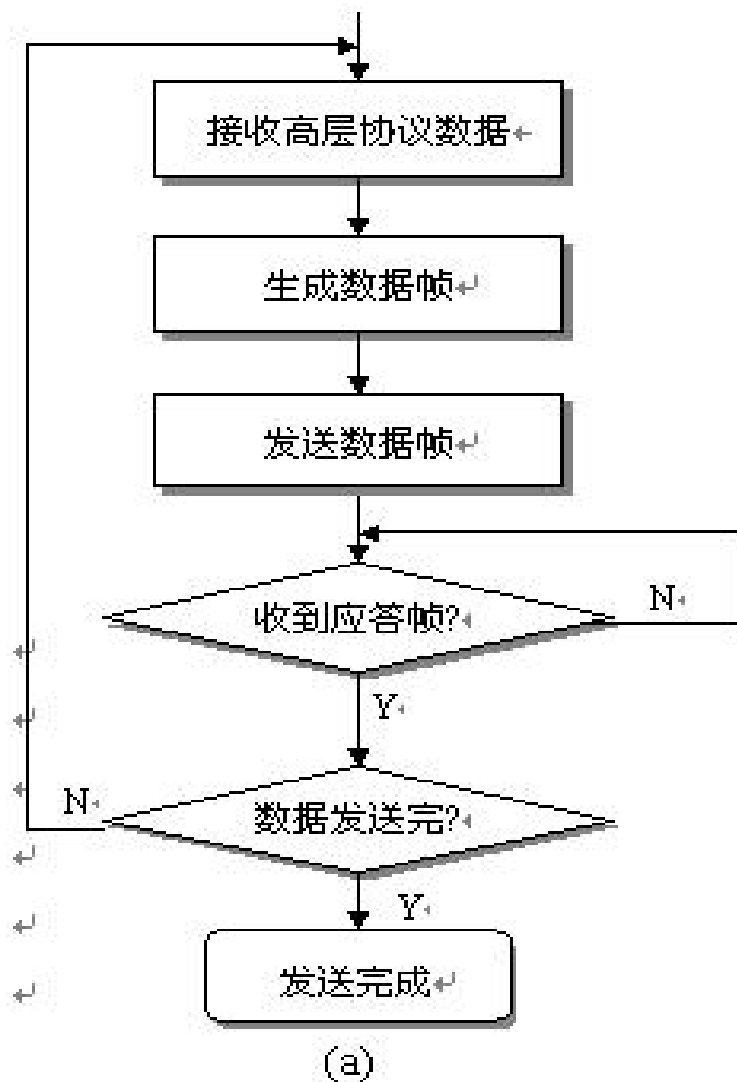


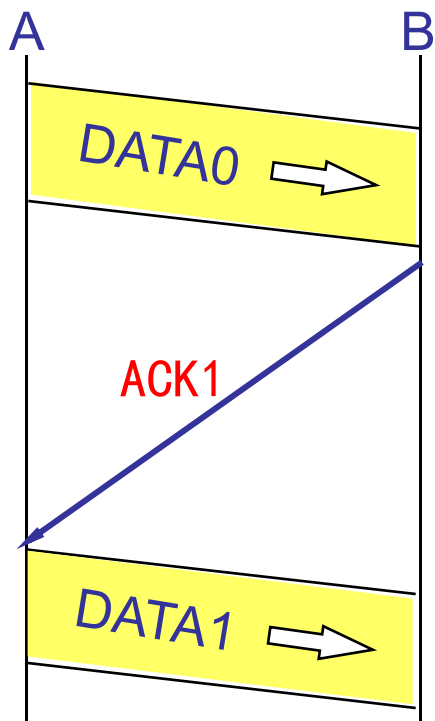
图 停止-等待协议工作流程

(a) 发送流程; (b)接收流程

可靠通信三大机制：序号 + 确认反馈 + 超时重传机制。

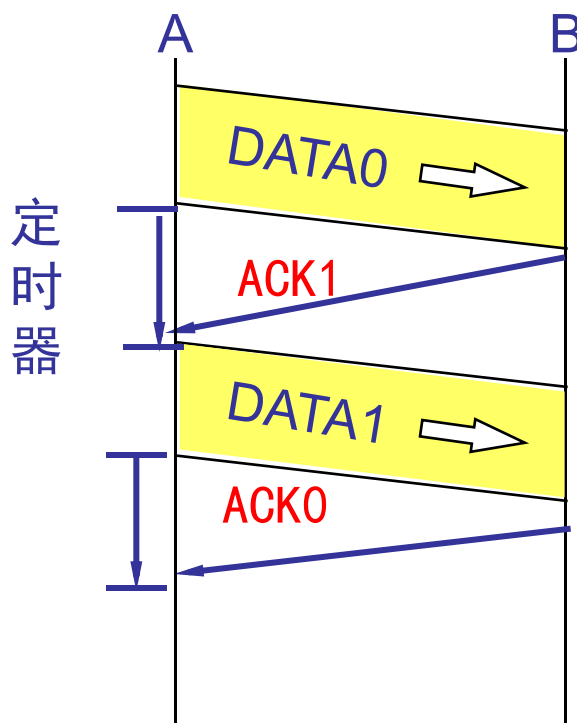
(1) 停止-等待协议

流量控制+差错控制（改进）（见教材流程图）



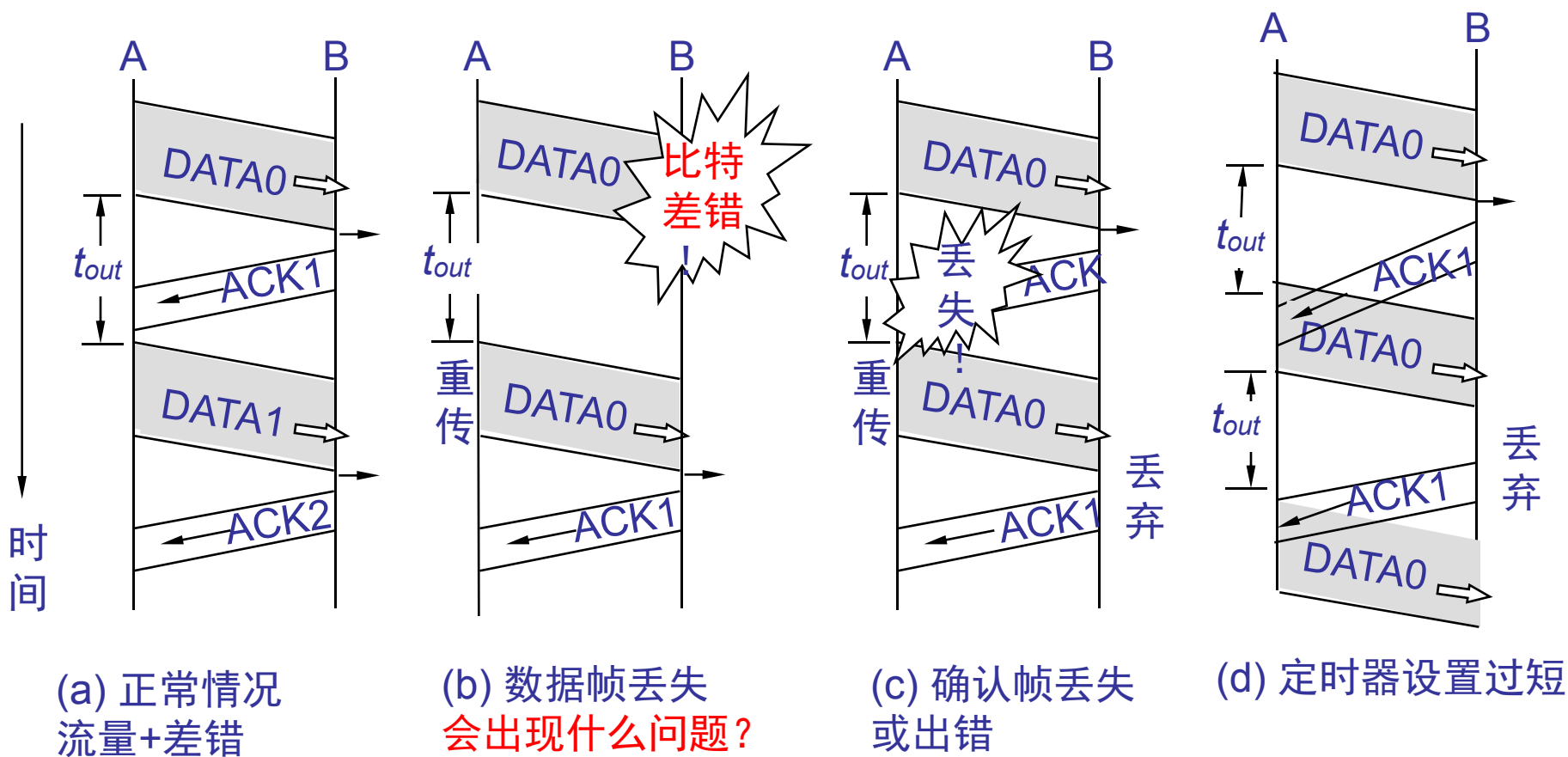
停止-等待协议（仅流量控制）

数据帧差错（丢失？），ACK差错，
出现死锁，如何解决？



流量控制+差错控制
（改进）

停止—等待协议特殊情况分析

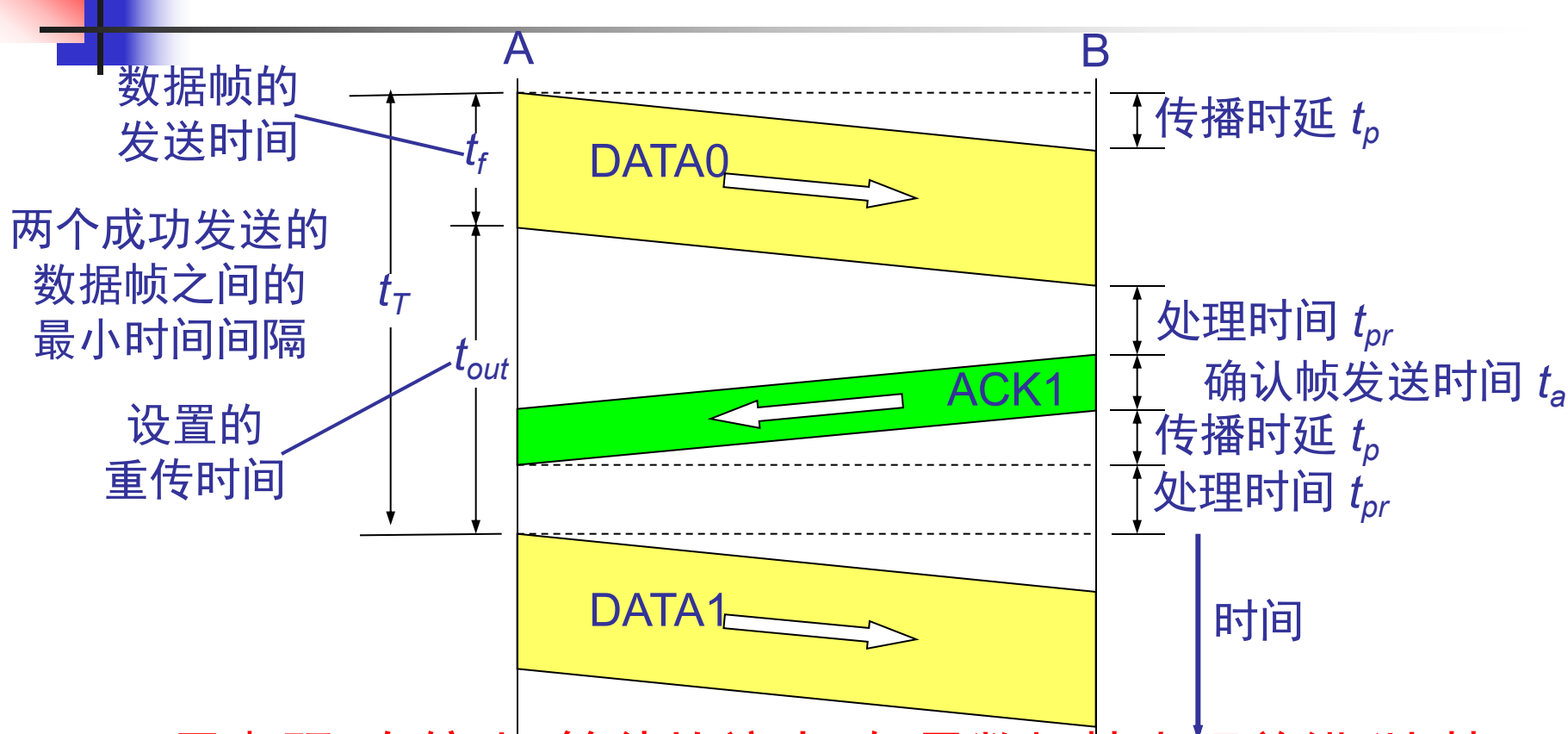




问题讨(1)

- 在上图(c)和(d)情况下,会导致接收方接收到重复帧问题。
- 重复帧是由于发送方**重发定时器超时**, **重发定时器超时原因?**:
 - **设定时间短**;
 - 数据帧出现比特差错,或网络**延迟时间长**;
 - 确认**差错**,或网络**延迟时间长**;
 - 注意:红色表示出现什么情况?
- 停止-等待协议引入1比特位的序号字段来解决。
 - 数据帧中的发送序号 $N(S)$ 以 0 和 1 交替的方式出现在数据帧中。
- 定时器时间设置:选为略大于“从发完数据帧到收到确认帧所需的平均时间RTT”。

停止-一等待协议中数据帧和确认帧的发送时间关系



思考题: 在停止-等待协议中, 如果数据帧出现差错 (比特差错) 的概率为 P , 请证明正确传输一个数据帧所需平均时间

$$T = t_T / (1 - P)$$



问题讨论(2)

- 数据帧出现比特差错, 接收方如何处理:
 - 方法一: 采用鸵鸟策略, 即不处理; 而发送方利用重发定时器超时重发来解决;
 - 方法二: 接收方成功接收到上一个数据帧后, 启动一个接收定时器, 如果超时或数据出现差错, 发送一个NACK确认, 要求发送丢失或差错数据帧, 一很少使用。
- 接收方连续接收到相同序号的数据帧, 表明发送端进行了超时重传。
- 发送方在发送完数据帧时, 必须在其发送缓存中暂时保留已发数据帧的副本, 才有条件在出差错时进行重传。
- 发送方只有接收到ACK确认, 才可清除该副本 (从内存中)。



停止-等待协议

- 优点：简单。

- 缺点：协议效率低, 物理链路的利用率比较低, 信道还远远没有被数据比特填满。

- 举例：一条线路带宽为1.5Mbps, RTT(Round Trip Time)=45ms, 如果采用停止-等待协议（发送方只能在一个RTT时间内发送一个数据帧），假设数据帧大小为1KB, 则发送方实际发送速率为182Kbps, 只相当于链路带宽的1/8。原因？

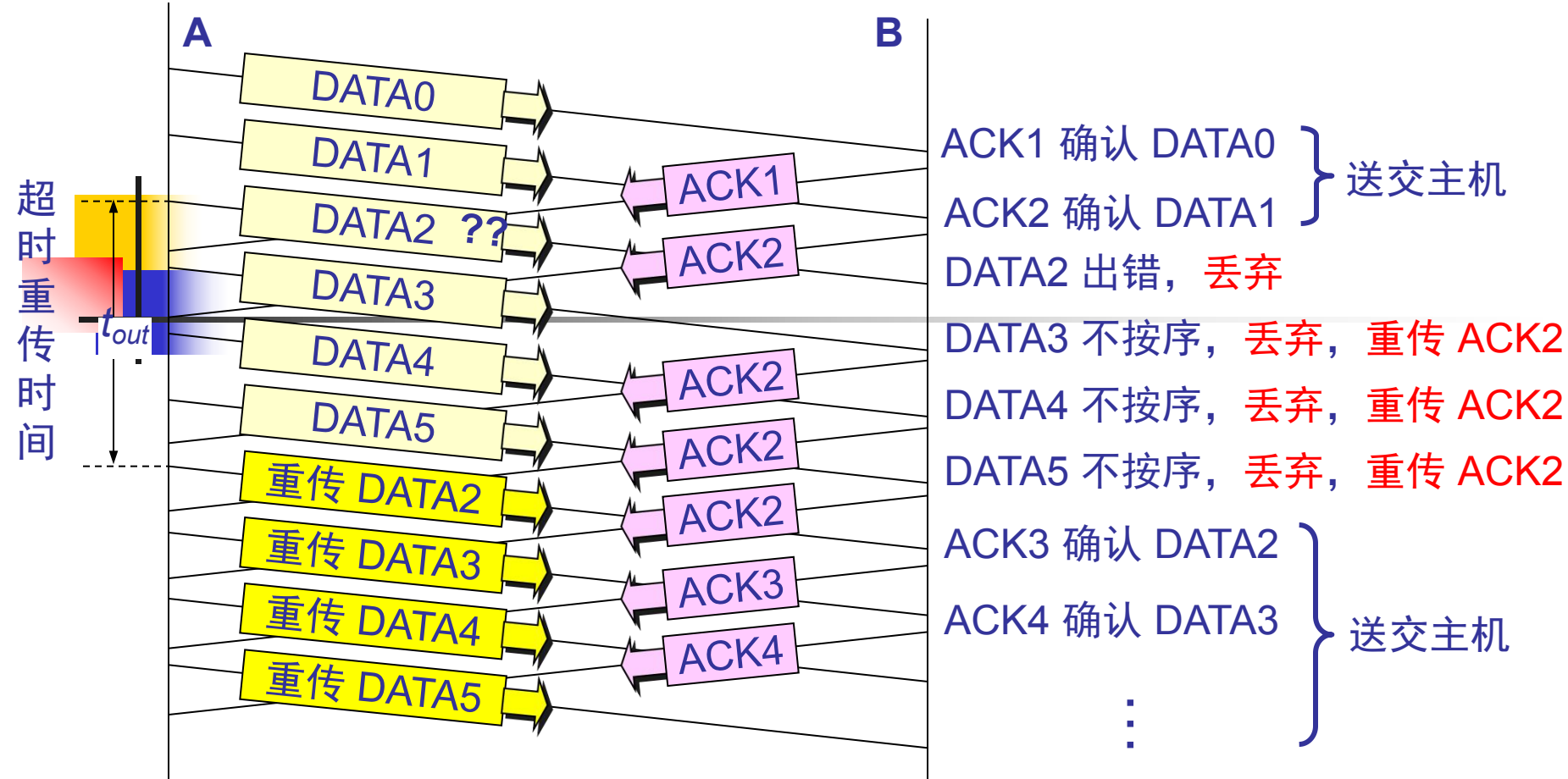
- 原因：发送方发送完一个数据帧，不能继续发送，需等待相应ACK应答；

- 改进：要提高链路利用率，发送方在等待接收方返回第一个ACK应答前，再连续发送7个数据帧，这也是连续ARQ协议思想。



(2) 后退N帧协议

- 连续ARQ协议 (automatic repeat request) : 发送方可一次连续发送多个数据帧, 同时等待ACK应答;
 - 后退N帧协议
 - 选择重发协议
- 后退N帧协议工作原理:
 - 发送方在发送完一个数据帧后, 不是停下来等待ACK确认帧, 而是可以连续再发送若干个数据帧, 同时启动重发定时器。
 - 如果发送方数据帧重发定时器超时前, 收到了接收方发来的确认帧, 继续接着发送后面数据帧。
 - 若发送方数据帧重发定时器超时, 还没有收到确认, 则从该帧开始的后继帧全部重发。



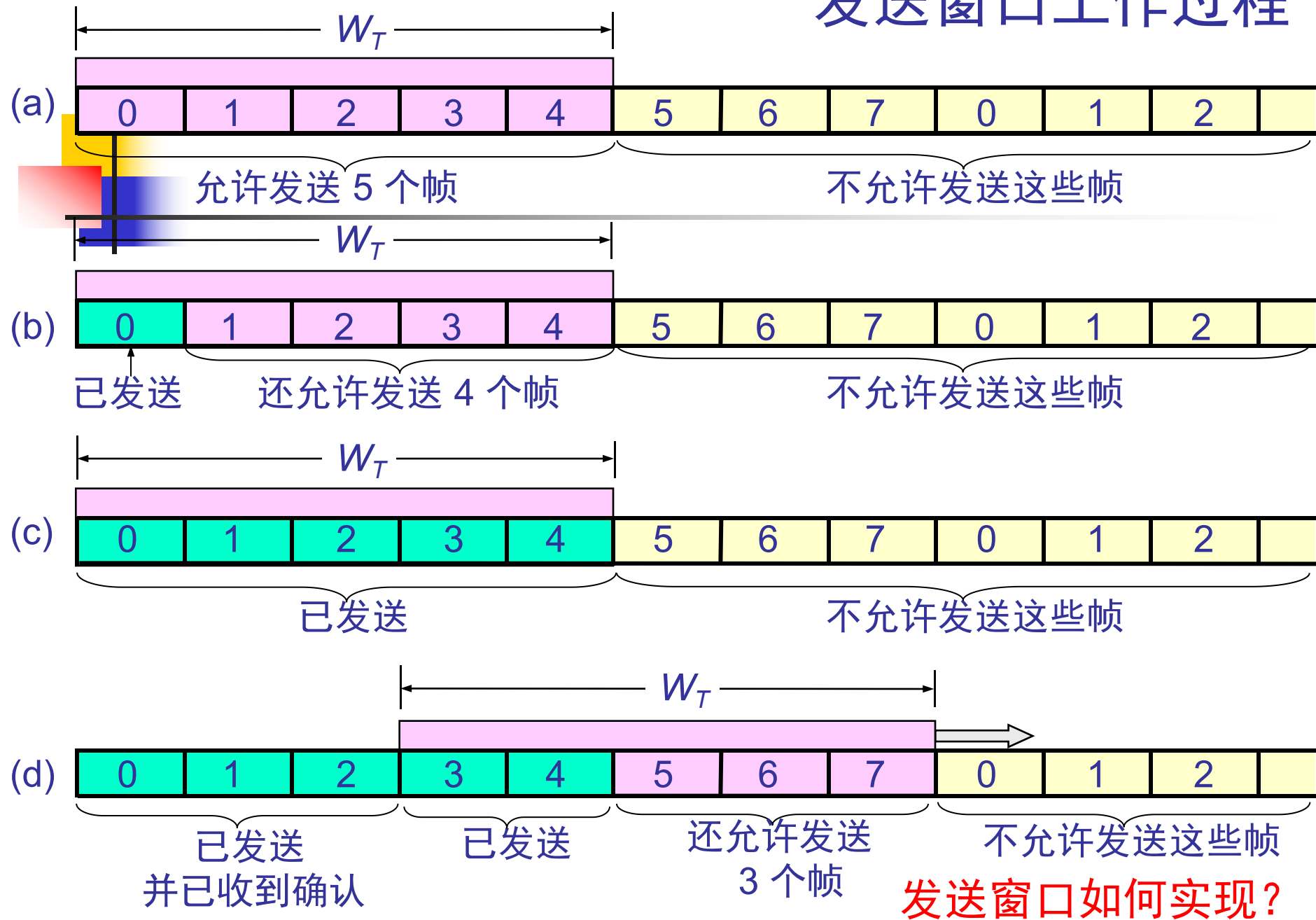
- 即使3、4、5数据帧没有差错，接收方按照乱序错误丢弃，因为接收方窗口大小为1，且**序号为2**。
- 2号帧重传法定时器超时，重发2、3、4、5号数据帧。
- 发送方窗口大于1，而**接收方窗口等于1**。
- 发送方接收到连续相同序号的ACK，说明接收方出现乱序。
 - 数据帧出现出错，但又收到后续数据帧。



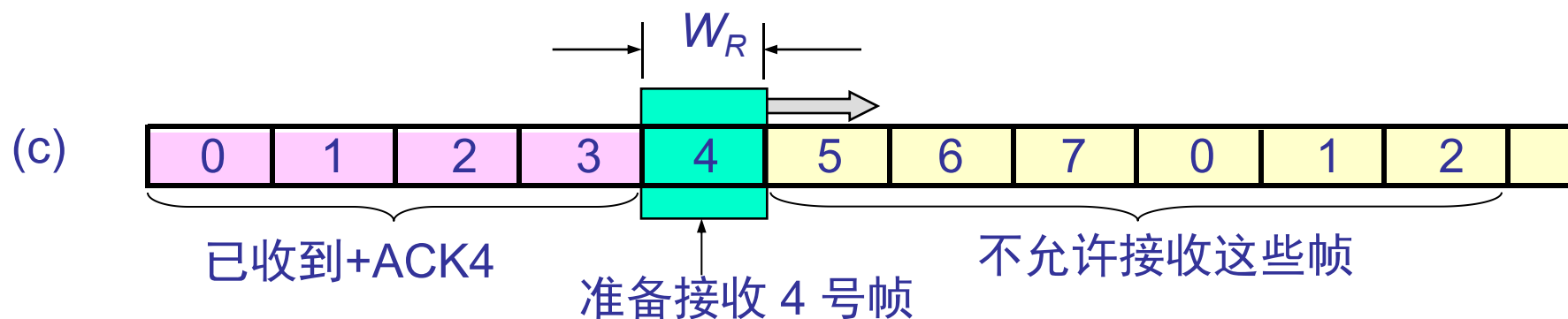
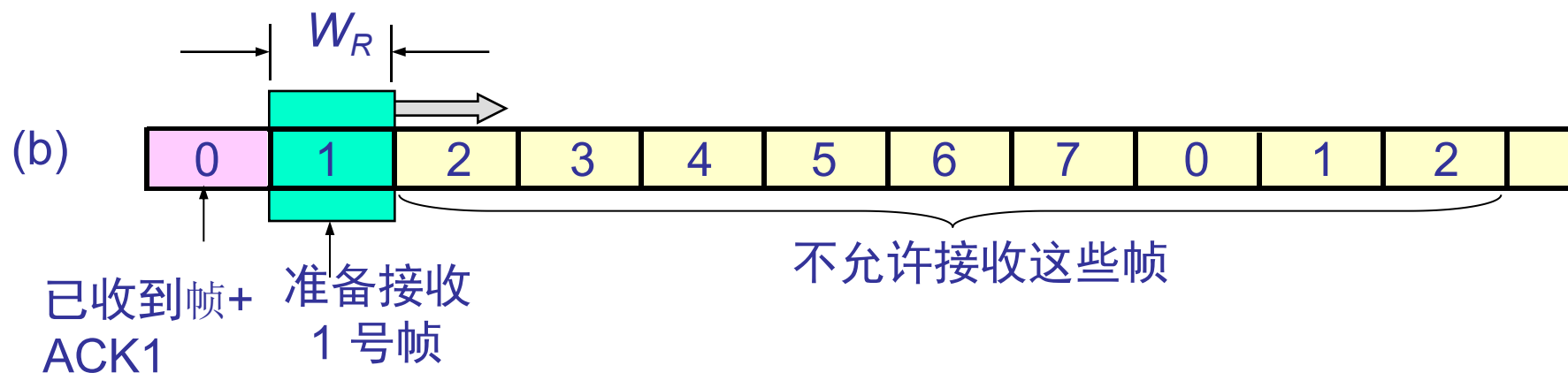
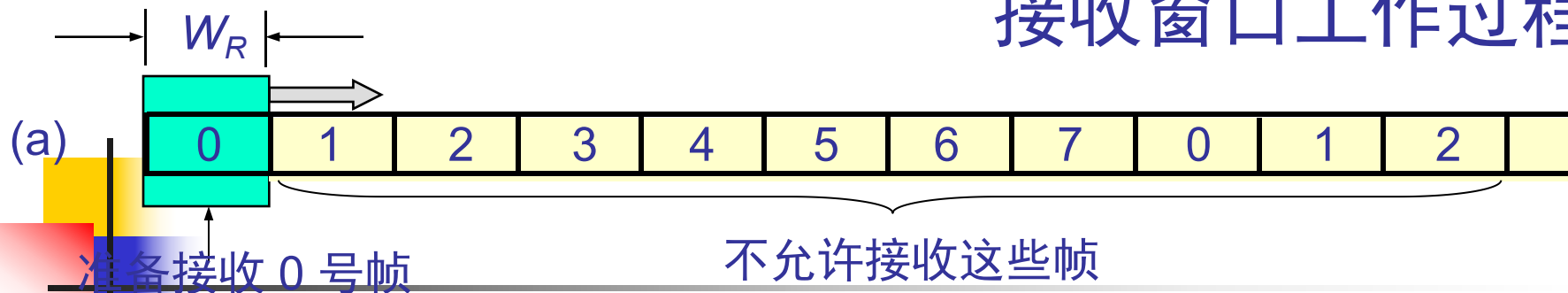
利用滑动窗口技术解释

- 发送方和接收方缓冲区分别设定一个**发送窗口和接收窗口**。
- 发送窗口
 - 发送方只能发送窗口内数据帧，对**发送端进行发送速率控制**。
 - 窗口大小：在**还没有收到接收方确认**情况下发送方最多**可以发送的数据帧个数**。
 - 发送方每接收到一个ACK确认，窗口向前滑动一个单位。
- 接收窗口
 - 接收窗口用来控制接收方可以接收哪个（**哪些？**）数据帧。
 - 只有**数据帧序号与接收窗口序号一致时接收**，否则丢弃。
 - 接收方正确接收并处理后（无差错），交付上层，窗口向前滑动一个单位，并发送一个ACK确认，准备接收下一个帧。

发送窗口工作过程

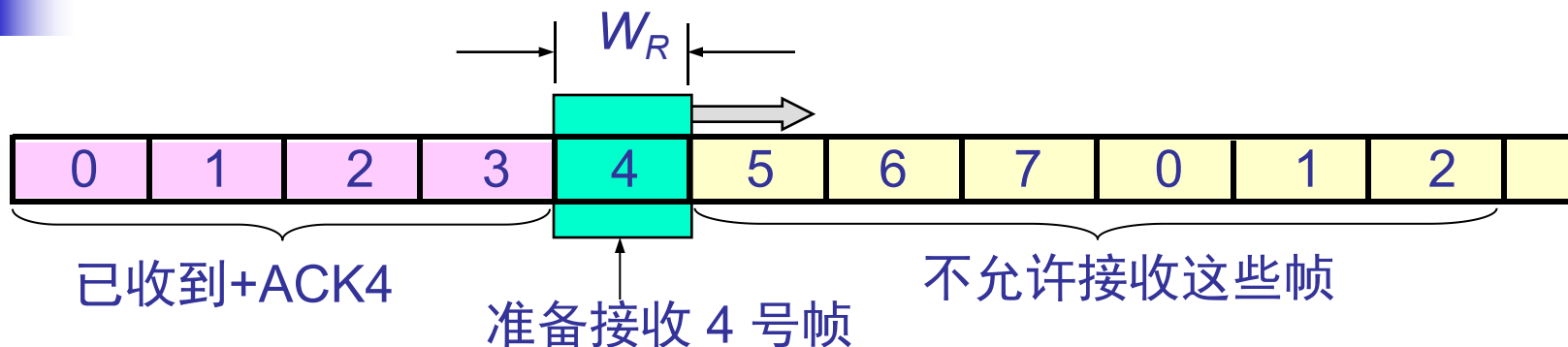


接收窗口工作过程



累计应答

乱序帧的处理



- 接收方准备接收4号帧。
- 但发送方可能连续发送了4, 5, 6, 7号帧。
- 如果4号帧出错该怎么办? **丢失呢?**
 - 4号出错帧采用鸵鸟策略, 丢弃;
 - 将5, 6, 7号帧即使正确接收, 作为乱序帧, 丢弃, 发送ACK4;
 - 并发送三个ACK4确认。



发送窗口最大值

- 后退N帧协议中, 如果序号字段为K比特位, 则帧序号空间为 $[0, 2^k - 1]$
- 定理1: 后退N帧协议中, 如果序号字段为K比特位, 则发送窗口最大值为 $2^k - 1$, 可保证该协议在任何情况下不会出现差错。
- 如果序号为3比特位, 序号空间为: 0, 1, ..., 7; 发送窗口最大值为7, 即一次可最多连续发送7个数据帧。
- 为什么发送窗口要减1?

(3) 后退N帧协议 (总结)

- 发送节点按照发送窗口大小一次连续发送多个数据帧。
- 发送方每发送完一个数据帧都要设置该帧的**重发**定时器。
 - 如果定时器未超时收到确认帧，则立即将定时器清零，窗口向前滑动。
 - 如果定时器超时未收到确认帧，重传该帧后所有数据帧。
 - **问题：发送方发送完多少个数据帧，就启动多少个定时器？如何进行优化？**
- 接收节点只按序（序号）接收数据帧：
 - 接收到一个数据帧，检测正确（序号和校验位），排队缓存处理，并交付上层协议后，发送一个ACK应答，接收窗口向前滑动一个单位。
 - 累计确认（cumulative acknowledgement）：ACK n 表示确认 $n-1$ 号前所有帧，并期望下次收到 n 号帧；
 - 捎带确认：对于全双工通信，接收方给发送方发送数据时捎带确认。



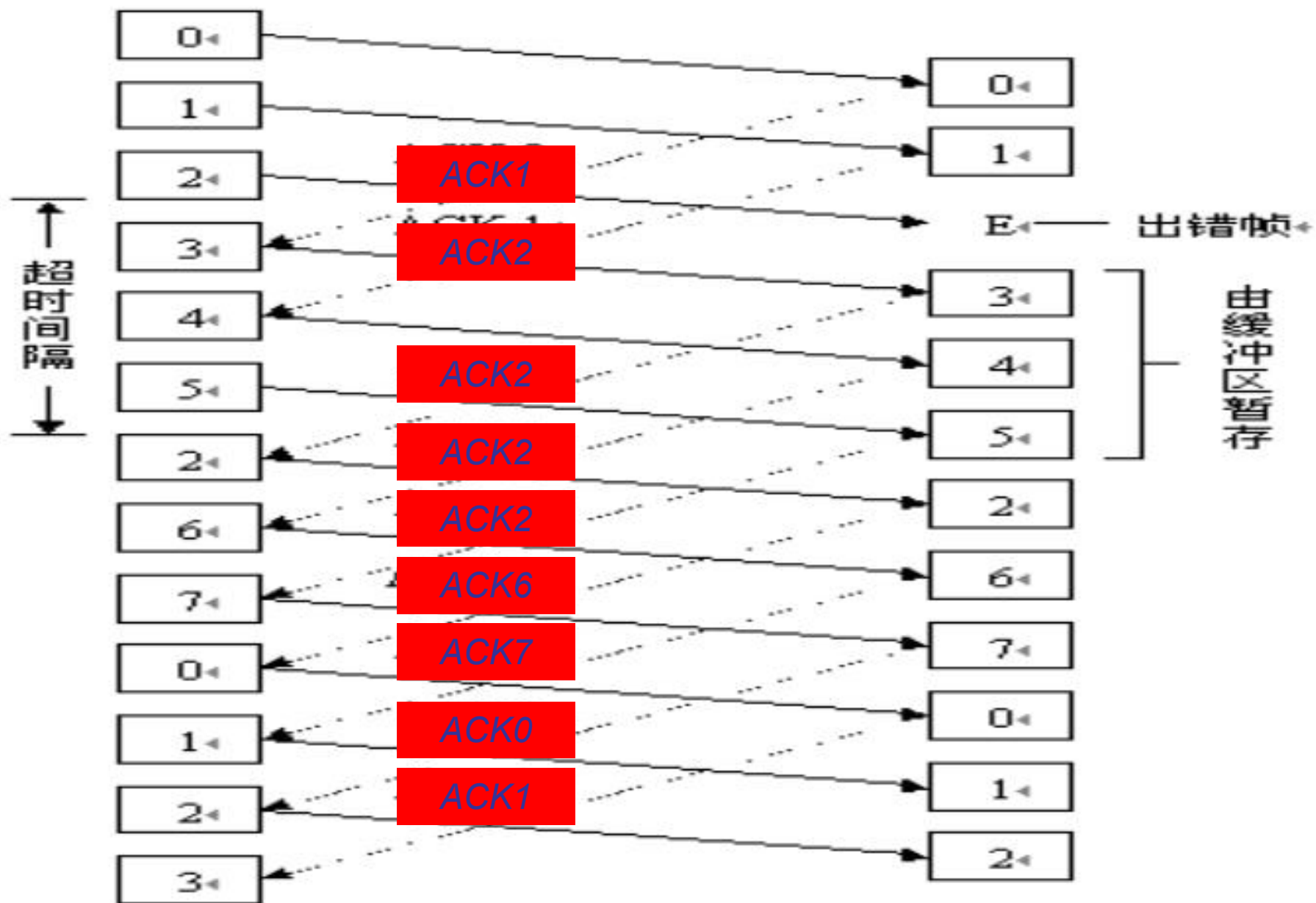
后退N帧协议存在的问题

- 后退N帧协议中, 将出错帧后所有的数据帧重传, 造成线路带宽的浪费, 原因是什么?
 - 接收窗口等于1, 无法缓存后面正确接收数据帧;
 - 过多的重复的数据帧在网络上传输。
- 在如何改进?
 - 增加接收窗口大小, 用于缓存已正确接收数据帧。
 - 只重传出现差错的数据帧。

选择重发协议

发送站

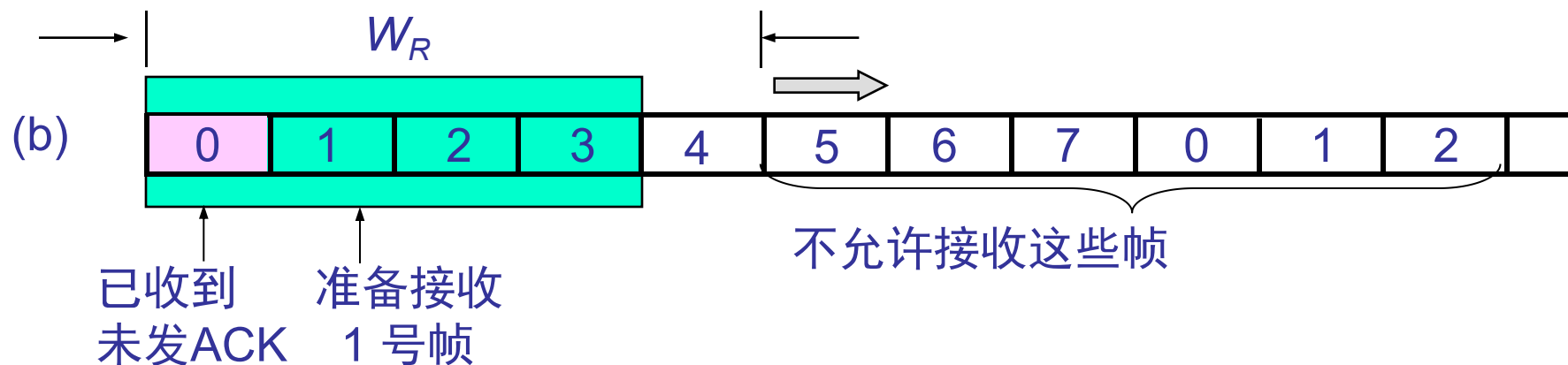
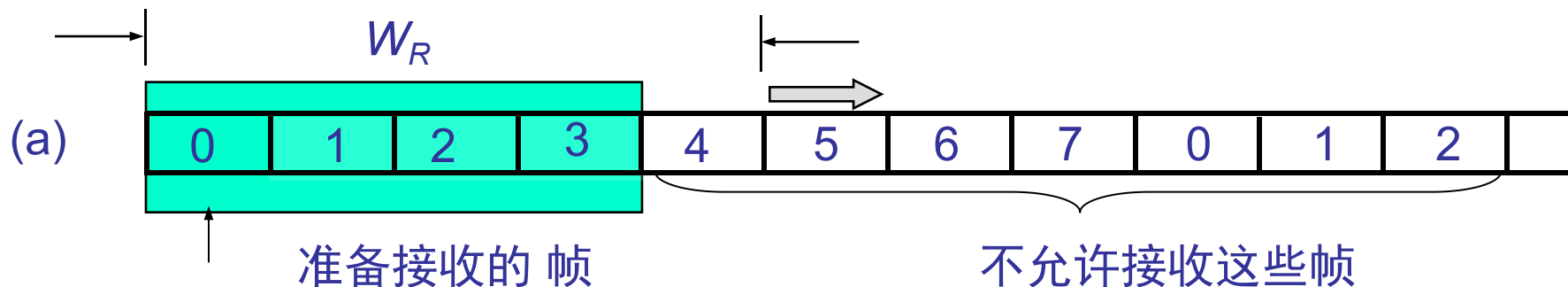
接收站



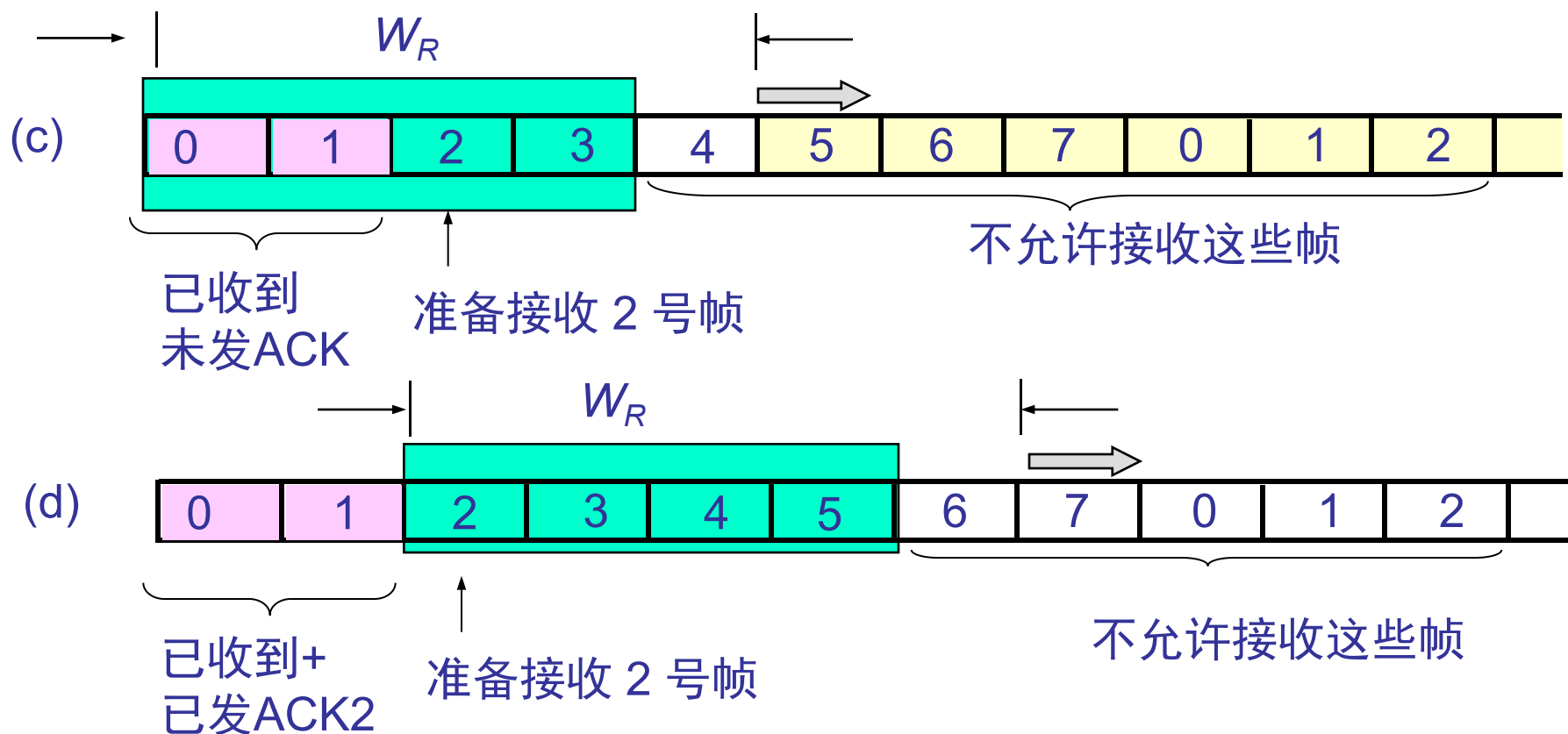
3.4 选择重发协议

接收窗口的工作过程

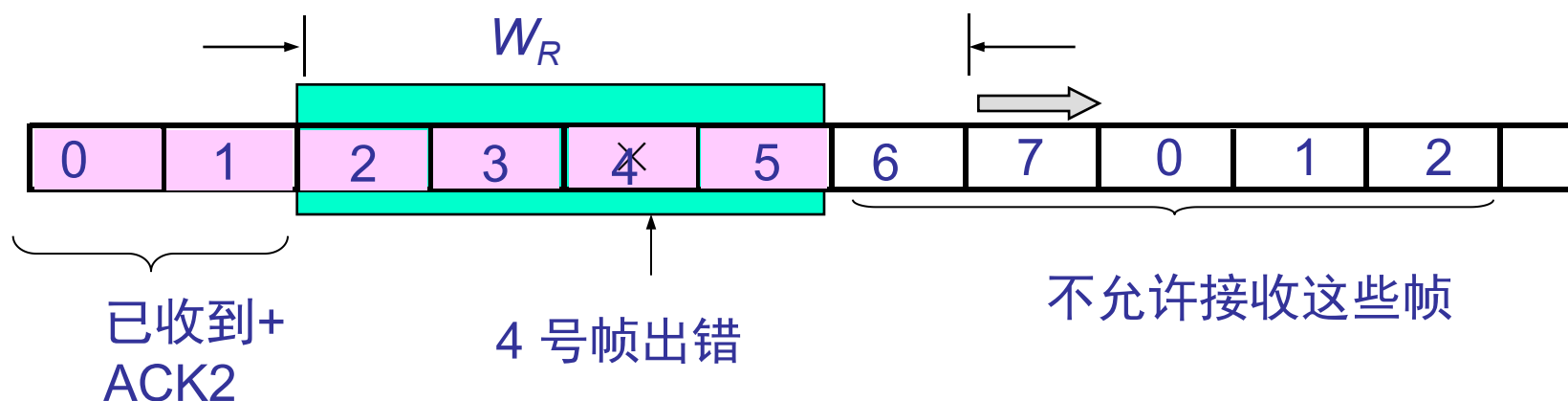
- 发送窗口的工作过程和后退N帧一样, 但接收窗口的大小 $W_R > 1$



3.4 选择重发协议



3.4 选择重发协议



- 如果4号帧出错, 仅仅对4号帧进行重传.
- 请问: 在选择重发协议中, 为什么发送窗口的最大值不为: $2^k - 1$

$$w_T \leq 2^{k-1}$$



流量控制总结

协议	序号比特位	序号空间	发送窗口大小	接收窗口大小
停止-等待协议	1	[0 , 1]	1	1
后退N帧协议	K	$[0, 2^k - 1]$	$2^k - 1$	1
选择重发协议	k			

数据链路层本节小结

- 成帧四种方法(对应概念:拆帧)
 - 字符计数法
 - 字符填充分界符法(面向字符的同步传输)
 - 零比特填充分界标志法(面向位流的同步传输)
 - 物理层编码违例法(传统以太网)
- 流量控制(+差错控制)
 - 停止—等待协议(带差错控制)
 - 流量控制: 序号 + ACK确认
 - 流量控制+差错控制: 序号 + ACK确认+超时重发
 - 连续ARQ协议: 后退N帧协议, 选择重发协议
 - 流量控制+差错控制: 序号 + ACK确认+超时重发

数据链路层本节小结

■ 差错检测与纠错

■ 比特差错

- 检错：利用校验码(奇偶校验码, CRC冗余校验码, 海明威码)，检测数据传输正确性（保持0与1不变，比特差错）；
- 检错码：只能检错；纠错码：检错+纠错。
- 纠错（差错控制）：**序号+确认+超时重发**

■ 流控控制：对象接收缓存，目的两个速率匹配

- 停止-等待；后退N帧和选择重发
- 流量控制+差错控制同时实现；

■ 数据帧重复处理

- 根据接收窗口序号判断，直接丢弃重复数据帧。

■ 乱序

- 根据接收窗口序号判断，丢弃（后退N帧），没有办法缓存
- 根据接收窗口序号判断，只要在接收窗口内缓存（选择重发）



复习&预习&作业

- 本节内容

教材:PP41-PP47

- 下节预习内容:

教材:HDLC协议 (PP36-PP41)



复习&预习&作业

■ 思考题

- 1. 流量控制机制有哪些方式, 各有什么优缺点;
- 2. 连续ARQ协议有哪两种方式, 各有什么特点;
- 3. 连续ARQ协议对发送窗口和接收窗口有什么限制?