Digitale Kommunikation und Internetdienste 1

Wintersemester 2004/2005 – Teil 10

Belegnummer Vorlesung: 39 30 02

Übungen: 39 30 05

Jan E. Hennig

AG Rechnernetze und Verteilte Systeme (RVS)
Technische Fakultät
Universität Bielefeld

`jhennig@rvs.uni-bielefeld.de`

basierend auf den Arbeiten von Michael Blume, Heiko Holtkamp, Marcel Holtmann und I Made Wiryana

Übersicht

- TCP (Teil 1)
- Sliding Window-Algorithmus von TCP

Erinnerung (1)

- bis Internetebene einschließlich nur unzuverlässige Übertragung
- auf Transportebene ist zuverlässige Übertragung *eine* Möglichkeit
- diese wird durch TCP hergestellt
- auf Transportebene außerdem angesiedelt: Zuordnung einer Verbindung zu Prozessen
- geschieht über Portnummer

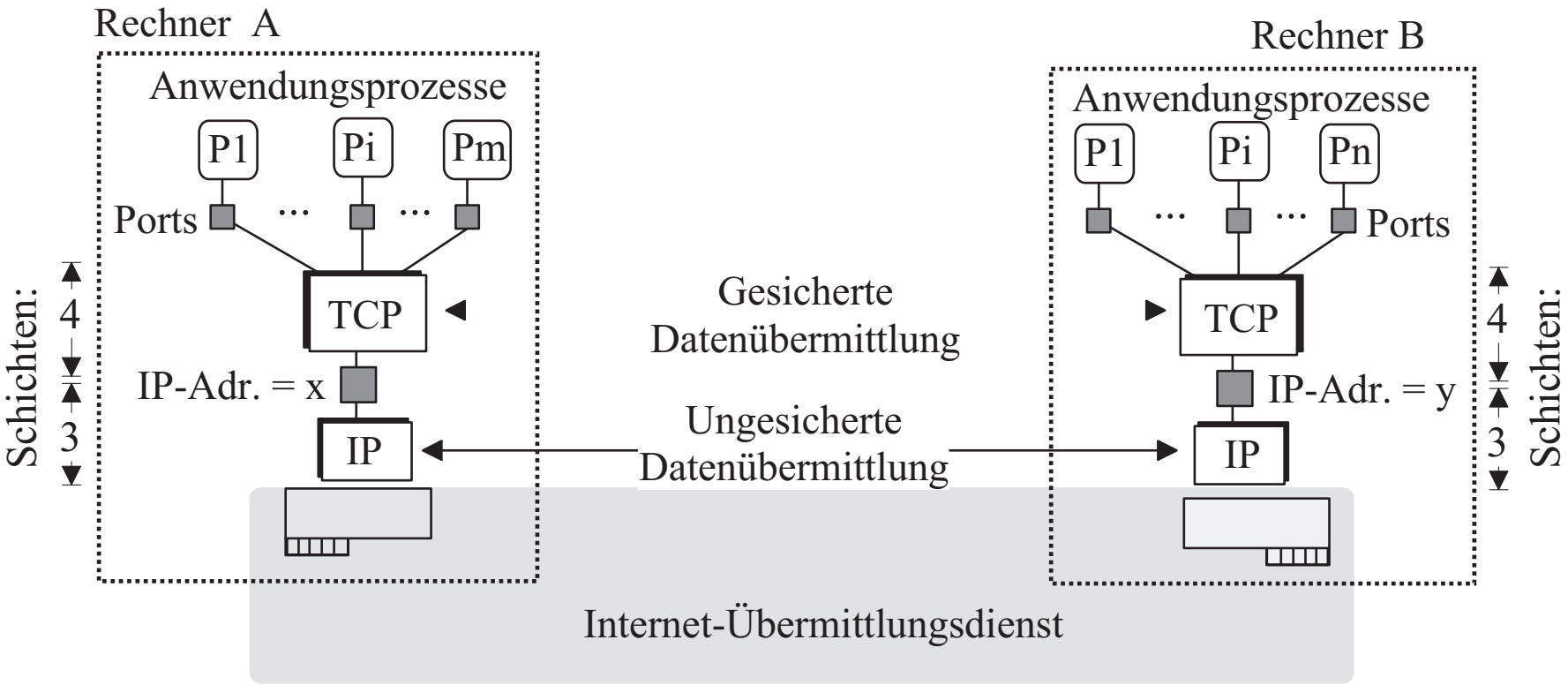
Erinnerung (2)

- zuverlässige Übertragung mit Hilfe von Bestätigungen (auch: Quittungen, ACKs) und Zeitablauf (timeouts)
- Bestätigungen informieren den Sender, daß ein Paket erfolgreich angekommen ist
- erhält der Sender keine Bestätigung innerhalb eines festgelegten Zeitraums
- so sendet er das Paket erneut (automatic repeat request nach timeout)

Erinnerung (3)

- *stop-and-wait*-Algorithmus ist einfachstes ARQ-Schema.
- Verbesserung durch *Concurrent Logical Channels*-Algorithmus (ARPANET)
- Flexibler und weitere Verbesserung durch *Sliding Window*-Algorithmus

Transmission Control Protocol (TCP)



Ports (1)

- auch bei TCP werden den aktiven Anwendungsprozessen Ports zugeordnet
- sie stellen die individuellen Kommunikationspuffer einzelner Anwendungensprozesse dar
- sie werden den Anwendungsprozessen nach Bedarf (auch dynamisch) zugeordnet
- bei TCP werden für eine Portnummer 16 Bit verwendet
- also bis zu 65535 TCP-Ports je Rechner möglich

Ports (2)

- wie bei UDP auch bei TCP einige Portnummern reserviert
- 0 bis 1023 sind weltweit eindeutig für Standarddienste (sog. *well known services*) reserviert
- sie werden auch *well known ports* genannt
- im Bereich von 1024 bis 65535 können sie im Rechner den Anwendungsprozessen frei zugeteilt werden
- darüber wird logische Verbindung zwischen Prozessen hergestellt

Auszug aus /etc/services


```
tcpmux          1/tcp                       # TCP port service multiplexer
echo            7/tcp
daytime         13/tcp
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                      # SSH Remote Login Protocol
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
domain          53/tcp          nameserver      # name-domain server
bootps          67/tcp                          # BOOTP server
bootpc          68/tcp                          # BOOTP client
gopher          70/tcp                          # Internet Gopher
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
kerberos        88/tcp          kerberos5 krb5 kerberos-sec     # Kerberos v5
pop2            109/tcp         postoffice pop-2        # POP version 2
pop3            110/tcp         pop-3           # POP version 3
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-ssn     139/tcp                         # NETBIOS session service
imap2           143/tcp         imap            # Interim Mail Access Proto v2
```

TCP (1)

- den Anwendungen bietet TCP Daten als *Datenstrom (data stream)* an
- Datenstrom prinzipiell unendlich lang
- Daten sind jedoch so nicht verschickbar
- tiefere Ebenen bieten nur das Verschicken von Datenpaketen begrenzter Größe an
- daher: einteilen in Datenblöcke (Segmente)

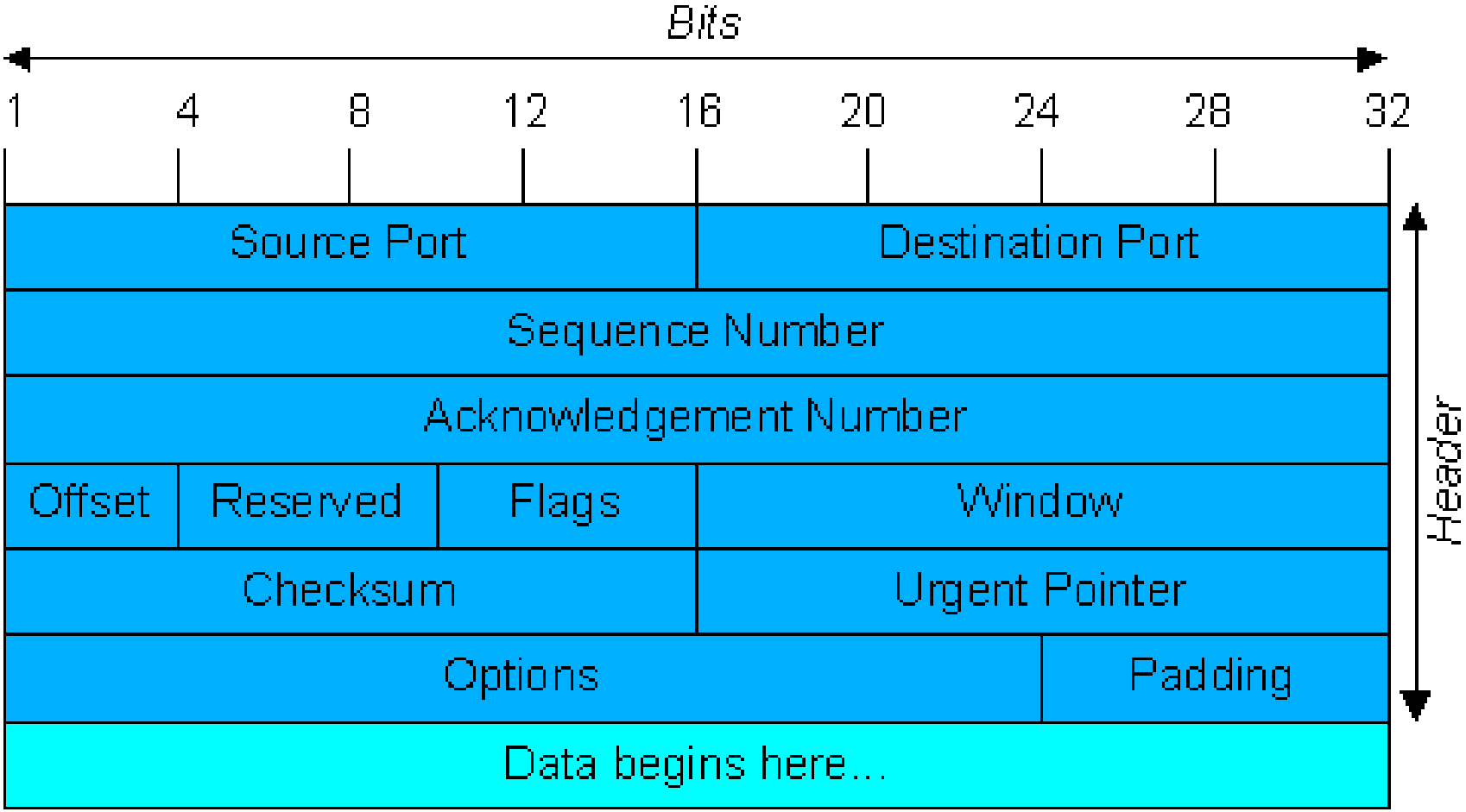
TCP (2)

- Segmentierung des Datenstroms in TCP-Datenpakete
- jedes Segment/Paket erhält TCP-Header
- Segmente werden durchnummeriert
- maximale Segmentgröße wird zwischen Quelle und Ziel vor dem Verschicken der ersten Daten ausgehandelt

TCP (3)

- die TCP-Pakete werden vom IP-Protokoll als zusammenhanglose IP-Pakete (Datengramme) übertragen
- TCP-Instanz des Zielrechners setzt die empfangenen IP-Pakete in der richtigen Reihenfolge in die ursprünglichen Daten (Nachricht) zurück
- erreicht ein IP-Paket den Zielrechner nicht, wird die Wiederholung der Übertragung eines entsprechenden Datensegments von TCP veranlaßt
- TCP ist im RFC 793 definiert
- diese Definitionen wurden im Laufe der Zeit von Fehlern und Inkonsistenzen befreit (RFC 1122)
- und um einige Anforderungen ergänzt (RFC 1323)

TCP-Header (1)


Bits
1
4
8
12
16
20
24
28
32
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Offset
Reserved
Flags
Window
Checksum
Urgent Pointer
Options
Padding
Data begins here...
Header

TCP-Header (2)

- *Source Port (Quell-Port)*:
- der Quell-Port enthält die Portnummer des Anwenderprozesses im Quellrechner, der die Verbindung initialisiert hat
- *Destination Port (Ziel-Port)*:
- der Ziel-Port enthält die Portnummer des Anwenderprozesses im Zielrechner, an den die Daten adressiert sind

TCP-Header (3)

- *Sequence Number (Sequenznummer)*:
- gilt in Senderichtung
- dient der Nummerierung von gesendeten Datensegmenten
- beim Aufbau einer virtuellen Ende-zu-Ende-Verbindung generiert jedes TCP-Modul eine Anfangs-Sequenznummer
- diese Nummern werden ausgetauscht und gegenseitig bestätigt

TCP-Header (4)

- *Sequence Number (Sequenznummer)* (cont'd):
- um die gesendeten TCP-Pakete eindeutig am Zielrechner zu identifizieren, muß der Fall ausgeschlossen werden, daß sich zu einem Zeitpunkt im Netz mehrere TCP-Pakete mit der gleichen Sequenznummer befinden
- aus diesem Grund darf sich die Sequenznummer innerhalb der festgelegten Lebenszeit für die IP-Pakete nicht wiederholen
- im Quellrechner wird die Sequenznummer immer jeweils um die Anzahl bereits gesendeter Bytes erhöht

TCP-Header (5)

- *Acknowledgement Number (Quittungsnummer)*:
- gilt in Empfangsrichtung
- dient der Bestätigung von empfangenen Datensegmenten
- diese Nummer wird vom Zielrechner gesetzt, um dem Quellrechner mitzuteilen, bis zu welchem Byte die Daten korrekt empfangen wurden

TCP-Header (6)

- *Data Offset (Datenabstand)*:
- vier Bit großes Feld
- gibt die Länge des TCP-Headers in 32-Bit-Worten an
- daher manchmal auch *Header Length* genannt
- bezeichnet somit die Stelle, ab der die Daten beginnen

TCP-Header (7)

- *Control Flags (Kontroll-Flags)*:
- legen fest, welche Felder im Header gültig sind
- steuern so die Verbindung
- Struktur sieht 6 Bits vor
- und wenn das entsprechende Bit gesetzt ist, gelten die folgenden Bedingungen:

TCP-Header (8)

- *URG*: Der Urgent Pointer (Zeiger im Urgent-Feld) ist gültig
- *ACK*: Die Quittungsnummer ist gültig
- *PSH*: Push-Funktion: Die Daten sollen sofort an die nächsthöhere Schicht weitergegeben werden z.B. UNIX-Implementierungen senden PSH immer dann, wenn sie mit dem Segment gleichzeitig alle Daten im Sendepuffer übergeben
- *RST*: Reset: Die Verbindung soll zurückgesetzt werden
- *SYN*: Verbindungsaufbauwunsch, muß quittiert werden
- *FIN*: Einseitiger Verbindungsabbau und das Ende des Datenstroms aus dieser Richtung, muß quittiert werden

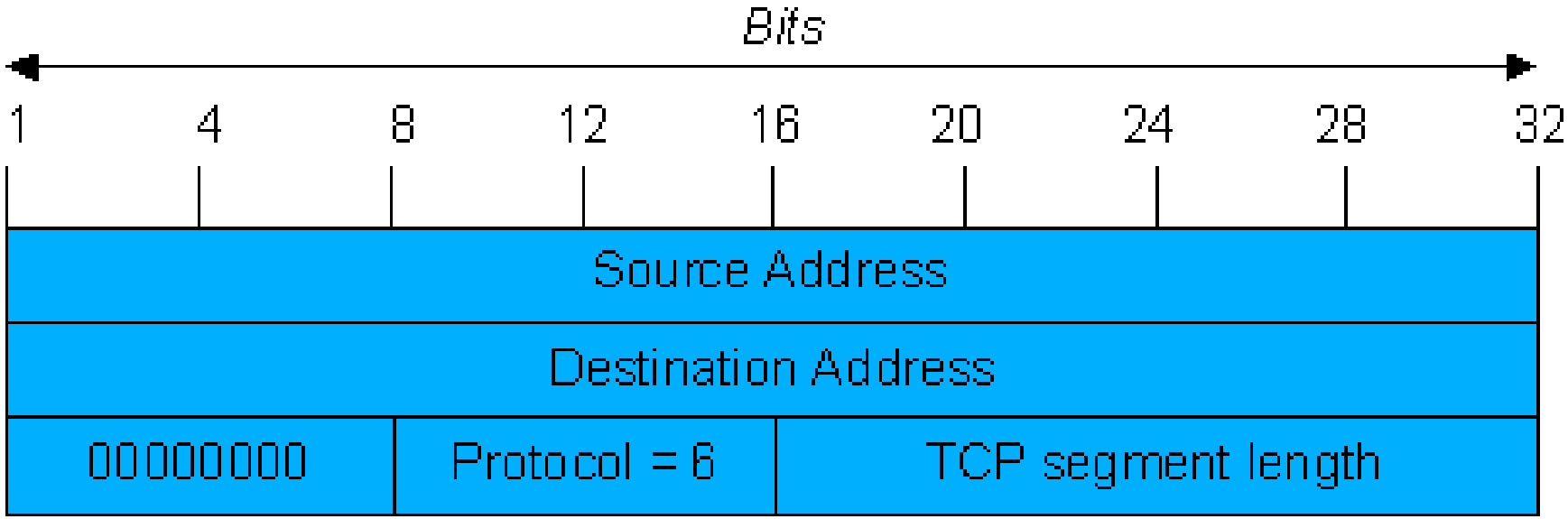
TCP-Header (9)

- *Window (Fenstergröße)*:
- diese Angabe dient der Flußkontrolle nach dem Sliding Window-Algorithmus
- hiermit steuert der Zielrechner den an ihn gesendeten Datenstrom im Quellrechner
- das Feld gibt die Fenstergröße an, d.h. wie viele Bytes beginnend ab der Quittungsnummer der Zielrechner in seinem Aufnahme-Puffer noch aufnehmen kann
- empfängt der Quellrechner ein TCP-Paket mit der Fenstergröße gleich 0, muß der Sendevorgang gestoppt werden
- die Ermittlung einer optimalen Fenstergröße gehört zu den wichtigsten Aufgaben bei der TCP/IP-Implementierung

TCP-Header (10)

- *Checksum (Prüfsumme)*:
- erlaubt es, den TCP-Header, die Daten und einen Auszug aus dem IP-Header (Pseudo-Header) auf Fehler zu prüfen
- bei Berechnung der Prüfsumme wird dieses Feld selbst als null angenommen
- Algorithmus für die Bildung der Prüfsumme: alle 16-Bit Wörter werden im 1er-Komplement addiert und die Summe ermittelt
- Pseudo-Header enthält Protokollnummer ($TCP = 6$)

TCP-Pseudoheader



TCP-Header (11)

- *Urgent Pointer (Urgent-Zeiger)*:
- TCP ermöglicht es, wichtige (dringliche) und meist kurze Nachrichten den gesendeten normalen Daten hinzuzufügen
- damit können außergewöhnliche Zustände signalisiert werden (z.B. Interrupts)
- derartige Daten werden hierbei als Urgent-Daten bezeichnet
- ist der Urgent-Zeiger gültig, d.h. $URG = 1$, so zeigt er auf das Ende der Urgent-Daten
- diese werden immer direkt nach dem TCP-Header übertragen, danach folgen „normale“ Daten.

TCP-Header (12)

- *Option*:
- TCP erlaubt es, Service-Optionen anzugeben
- das erste Byte im Optionsfeld legt den *Optionstyp (option kind)* fest
- mit den RFC 1323 und 2018 wurde das Optionsfeld in seiner Bedeutung wesentlich erweitert

TCP-Optionstypen

Optionstyp [kind]	Länge des Optionsfelds [Byte]	Bedeutung
0	nicht vorgesehen	Ende der Optionsliste
1	nicht vorgesehen	No-Operation
2	4	Maximum Segment Size (MSS)
3	3	Window Scale (WSopt)
4	2	SACK erlaubt
5	variabel	SACK
8	10	Times-Stamp-Abgleich
11	6	Connection Count CC (T/TCP)
12	6	CC.NEW (T/TCP)
12	6	CC.ECHO (T/TCP)

TCP-Optionen (1)

- *Maximum Segment Size (MSS)*:
- wird beim Verbindungsaufbau genutzt
- um dem Kommunikationspartner mitzuteilen,
- welche maximale Segmentgröße verarbeitet werden kann

TCP-Optionen (2)

- *Window Scale (WSopt)*:
- damit können die Kommunikationspartner optional während der Initialisierung festlegen ob die Größe des 16-Bit Window um einen konstanten Skalenfaktor multipliziert wird
- dieser Wert kann unabhängig für den Empfang (R) und das Versenden (S) von Daten ausgehandelt werden
- als Konsequenz dieses Verfahrens wird nun die Fenstergröße von der TCP-Instanz nicht mehr als 16 Bit sondern als 32 Bit-Wert aufgefaßt
- der maximale Wert für den Skalenfaktor von WSopt beträgt 14, was einer neuen oberen Grenze für Window von 1 GByte entspricht
- auf Grundlage des Skalenfaktors von WSopt wird auch der übertragene Wert von Window im TCPSegment neu berechnet

TCP-Optionen (3)

- *Timestamps Option (TSopt)*:
- besteht aus den Teilen Timestamp Wert (TSval) und Timestamp Echo Reply (TSecr)
- letzteres Feld ist nur bei ACK-Segment erlaubt
- mit diesen Informationen informieren sich die TCP-Instanzen über die Round Trip Zeit (RTT)
- diese dient für verschiedene Abschätzungen und automatische Anpassungen

TCP-Optionen (4)

- weitere Optionen:
- Selective Acknowledgement (SACK)
- Connection Count CC, sowie CC.NEW und CC-ECHO
  für TCP-Erweiterung Transaction-TCP (T/TCP)
- außerdem *Padding (Füllzeichen)*:
- die Füllzeichen ergänzen die Optionsangaben auf die Länge von 32 Bits

TCP Timeouts

- TCP verhindert den gleichzeitigen Verbindungsaufbau zwischen zwei Stationen
- d.h. nur eine Station kann den Aufbau initiieren
- außerdem ist es nicht möglich, einen mehrfachen Aufbau einer Verbindung durch den Sender aufgrund eines Timeouts des ersten Verbindungsaufbauwunsches zu generieren
- der Datenaustausch zwischen zwei Stationen erfolgt nach dem Verbindungsaufbau
- gehen Daten bei der Übertragung verloren, wird nach Ablauf eines Timeouts die Wiederholung der fehlerhaften Segmente gestartet
- durch die Sequenznummer werden doppelt übertragene Pakete erkannt
- aufgrund der Sequenznummer ist es möglich, $2^{32} - 1$ Daten (8 Gigabyte) pro bestehender Verbindung zu übertragen bis die Zählung wieder neu beginnt

Sliding Window

- die Flußkontrolle nach dem Sliding-Window-Algorithmus mit Window-Feld erlaubt
- einem Empfänger, dem Sender mitzuteilen, wieviel Pufferplatz zum Empfang von Daten zur Verfügung stehen
- ist der Empfänger zu einem bestimmten Zeitpunkt der Übertragung einer höheren Belastung ausgesetzt
- kann er dies dem Sender über das Window-Feld bekanntgeben
- über das Window-Feld kann also eine Anpassung im Betrieb erfolgen

Retransmission Time

- jedes übertragene TCP-Paket unterliegt einer Zeitüberwachung, der *(Retransmission Time)*
- ein Empfänger muß dazu nach einer bestimmten Zeitdauer eine Quittung über die erhaltenen Frames aussenden
- da diese Zeitdauer stark von der aktuellen Belastung des Netzes abhängt
- muß der Retransmission-Timer für jedes TCP-Paket neu berechnet und eingestellt werden

TCP-Verbindung

- eine TCP-Verbindung wird mit dem Ziel aufgebaut, einen sicheren Datenaustausch zwischen den kommunizierenden Anwendungsprozessen in entfernten Rechnern zu gewährleisten
- Beispiel: Kommunikation zwischen zwei FTP-Anwendungen (File Transfer Protocol)
- FTP ist eine TCP/IP-Standardanwendung und hat den Well Known Port Nummer 20
- aus Sicht der Kommunikation kann die Nummer des Well Known Ports auch als die weltweit eindeutige Identifikation einer TCP/IP-Standardanwendung gesehen werden

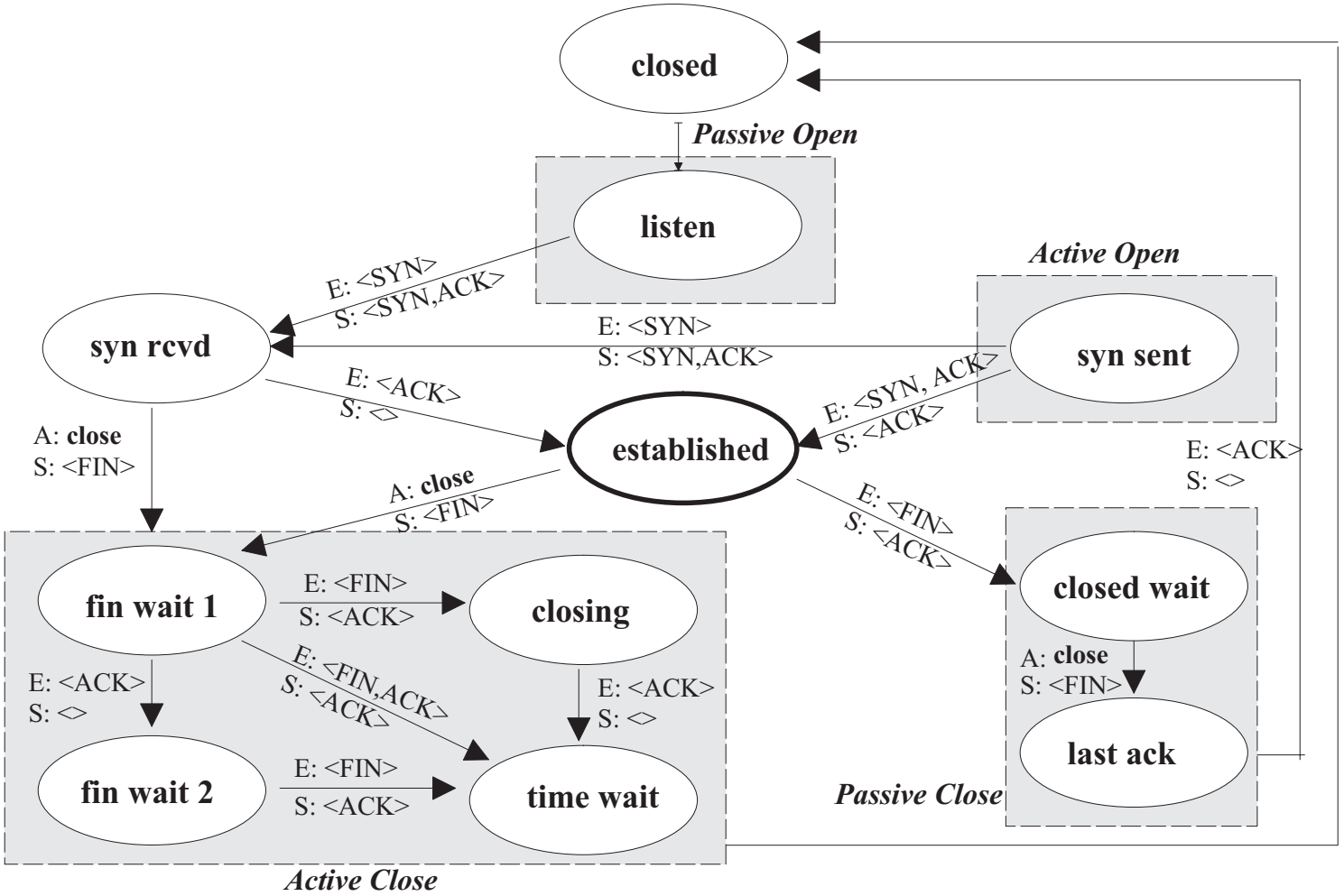
TCP-Handshake

- TCP-Verbindungen sind vollduplex, d.h. in beide Richtungen gleichzeitig verwendbar
- man kann eine TCP-Verbindung als ein Paar von gegenseitig gerichteten unidirektionalen Verbindungen interpretieren
- der Verbindungsaufbau zwischen zwei Rechnern erfolgt mit Hilfe des *Three Way Handshake (3WHS)*-Verfahrens
- das sorgt für eine Synchronisation der Kommunikationspartner
- und es stellt sicher, daß die TCP-Verbindung in jede Richtung korrekt initialisiert wird
- der Anwendungsprozeß im Quellrechner kommuniziert über einen wahlfreien Port, der dynamisch (aber nur im Quellrechner!) zugewiesen wird

TCP-Zustandsmodell (1)

- TCP basiert auf einer Zustandsmaschine
- eine TCP-Instanz befindet sich immer in einem wohldefinierten Zustand
- die Hauptzustände sind hierbei *listen* und *established*
- zwischen diesen stabilen Zuständen gibt es eine Vielzahl zeitlich befristeter Zwischenzustände
- mittels der TCP-Kontroll-Flags ACK, FIN, SYN und ggf. auch RST wird der Wechsel bzw. der Verbleib in einem Zustand signalisiert
- zwei Parameter dienen dabei dem Sliding-Window-Algorithmus

TCP-Zustandsmodell (2)



TCP-Fenstergröße

- Windowsize (WSIZE) gibt die Größe des TCP-Empfangspuffers in Byte an
- maximal $2^{16} - 1$ entspricht Wert von 65 535 Byte, d.h. rund 64 KByte
- moderne TCP-Implementierungen nutzen allerdings die Option des WSopt
- damit nun Werte bis 2^{30}, also rund 1 GByte möglich
- WSIZE ist ein Konfigurationsparameter der TCP-Implementierung
- üblicherweise auf einen Wert von 4, 8, 16 oder 32 KByte initialisiert
- dient als Initialwert: die *advertised Windowsize (advWind)*

TCP-Segmentgröße

- die *Maximum Segment Size MSS* stellt das Gegenstück zu WSIZE dar
- sie ist der maximale Wert des TCP-Sendepuffers
- für übliche TCP-Implementierungen gilt die Ungleichung $MSS < WSIZE$
- ein gutes Übertragungsverhalten von TCP auf sehr unterschiedlichen Trägernetzen ergibt sich durch das dynamische Aushandeln dieser Parameter
- zusammen mit der Bestimmung der RTT

TCP-Verbindungsaufbau (1)

- Empfänger startet im Zustand *passives Öffnen (passive open)*:
- eine Verbindung tritt in den Abhörstatus ein, wenn eine Anwendung TCP mitteilt, daß sie Verbindungen für eine bestimmte Portnummer annehmen würde
- Sender startet im Zustand *aktives Öffnen (active open)*:
- eine Anwendung teilt TCP mit, daß sie eine Verbindung mit einer bestimmten IP-Adresse und Portnummer eingehen möchte

TCP-Verbindungsaufbau (2)

- Beispiel: FTP (Ziel-TCP-Port 20 steht fest)
- Server-Rechner B mit IP-Adresse y öffnet Port 20 passiv
- Client-Rechner A mit IP-Adresse x startet Verbindungsaufbau von Quell-Port 3028 (zufällig frei)
- Rechner A generiert *Initial Sequence Number (ISN)* k
- Paket mit SYN-Flag, SEQ=k, Q-Port=3028, Z-Port=20 wird an Adresse y verschickt

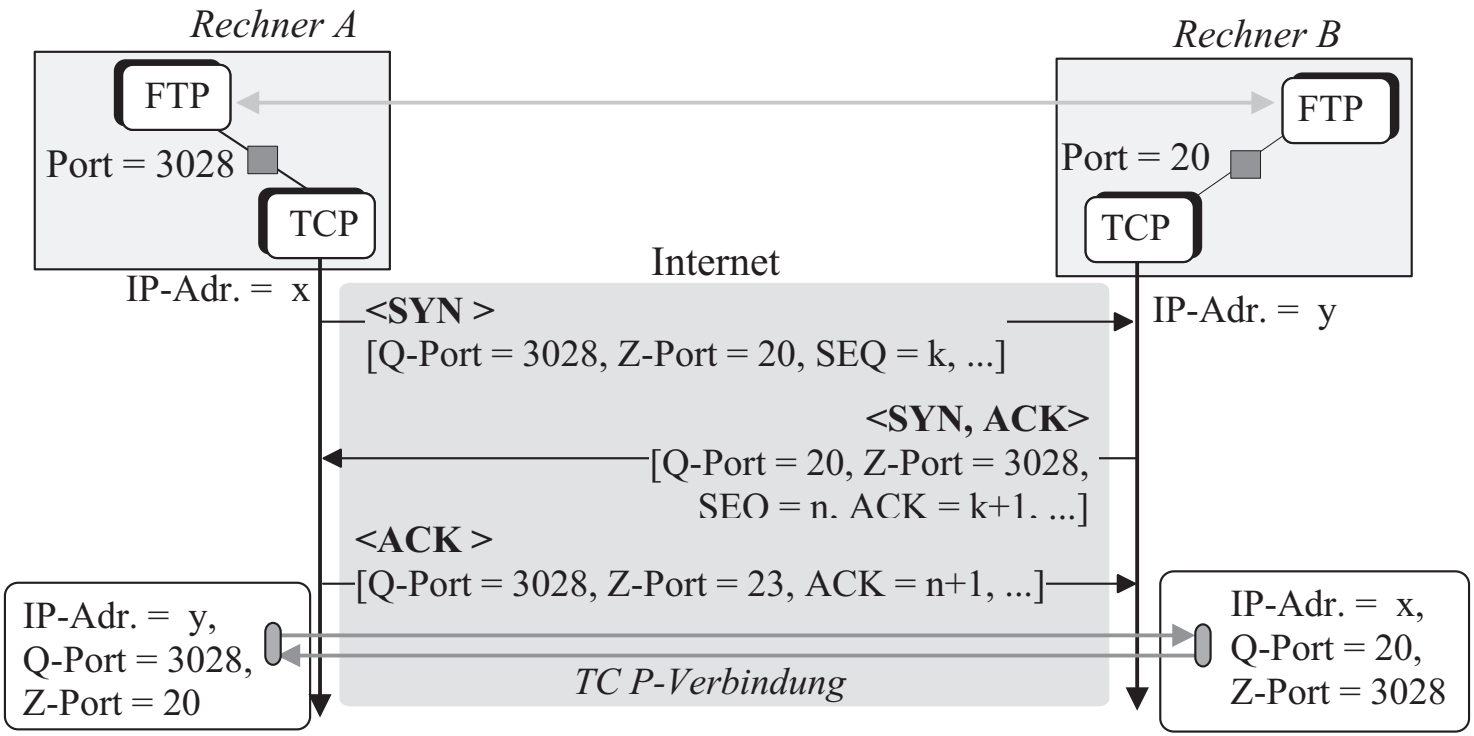
TCP-Verbindungsaufbau (3)

- Rechner B wartet an Port 20 auf Verbindungsaufbaupakete (SYN-Flag)
- leitet beim Empfang die Anfrage an die Anwendung weiter
- generiert selbst eine *Initial Sequence Number (ISN)* n
- Paket mit SYN+ACK-Flags, SEQ=n, ACK=$k+1$, Q-Port=20, Z-Port=3028 wird an Adresse x verschickt

TCP-Verbindungsaufbau (3)

- Rechner A wartet an Port 3028 auf ein Verbindungsaufbau- und Quittungspaket
- bestätigt Verbindungsaufbau durch weiteres Paket:
- ACK-Flag gesetzt, ACK=$n + 1$, Q-Port=3028, Z-Port=20, . . . an Zieladresse y
- damit ist je eine Verbindung pro Richtung aufgebaut

TCP-Verbindungsaufbau (5)



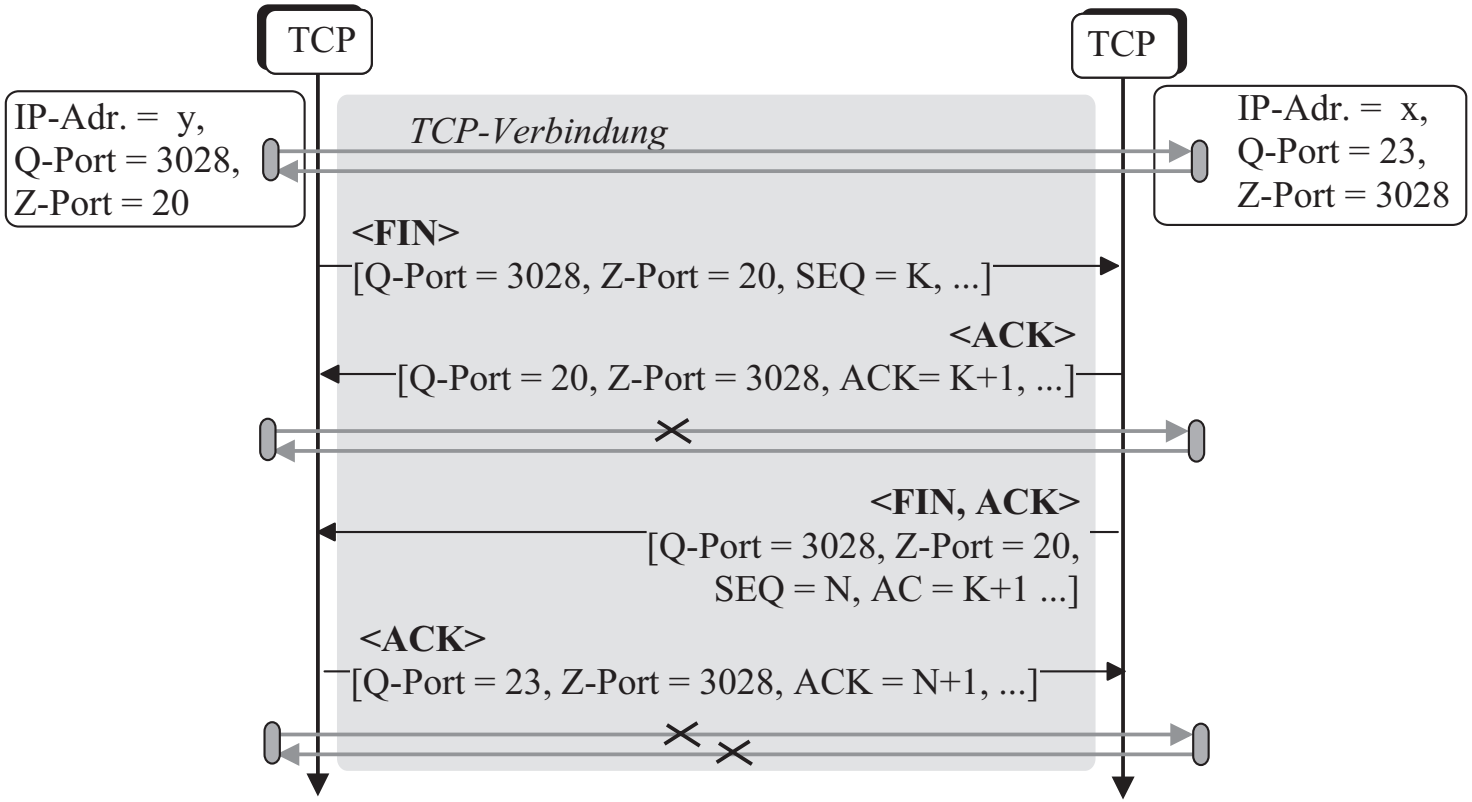
TCP-Verbindungsabbau (1)

- Abbau der Verbindung kann von jeder Seite eingeleitet werden
- die beiden logischen Verbindungen werden nacheinander abgebaut
- jede TCP-Instanz koordiniert den Abbau ihrer gerichteten Verbindung zur Partner-TCP-Instanz
- und verhindert hierbei den Verlust von noch unquittierten Daten

TCP-Verbindungsabbau (2)

- Der Abbau wird von einer Seite mit einem Paket mit gesetztem FIN-Flag initiiert (SEQ=K)
- dies wird von der Gegenseite durch das ACK-Segment mit dem gesetzten ACK-Flag positiv bestätigt
- die positive Bestätigung erfolgt hier durch die Angabe der Quittungsnummer ACK = $K + 1$
- damit wird *eine* gerichtete Verbindung abgebaut
- der Verbindungsabbau in der Gegenrichtung wird mit einem Segment, in dem die beiden FIN- und ACK-Flags gesetzt sind, begonnen
- nach der Bestätigung durch die Gegenseite wird der Abbauprozeß beendet

TCP-Verbindungsabbau (3)



TCP-Verbindungsabbau (4)

- beim Abbau einer Verbindung tritt u.U. ein zusätzlicher interner Time-Out-Mechanismus in Kraft
- die TCP-Instanz geht in den Zustand *aktive close*, versendet ein abschließendes ACK und befindet sich dann im Status Time-Wait
- dessen Zeitdauer beträgt zweimal die maximale Segment-Lebensdauer (*Maximum Segment Lifetime MSL*)
- nach Ablauf wird die TCP-Verbindung letztlich geschlossen
- TCP-Segmente, die länger als die MSL-Zeit im Netz unterwegs sind, werden verworfen
- der Wert von MSL beträgt in der Regel 120 Sekunden
- dann wird Port freigegeben und steht für spätere Verbindungen zur Verfügung

TCP-Flußkontrolle (1)

- Menge der Daten muß an die Aufnahmefähigkeit des Empfängers angepaßt werden
- die nötige Abstimmung bezeichnet man als *Flußkontrolle (flow control)*
- TCP verwendet dazu das Prinzip des Sliding Window
- wichtige Variablen:
  - *Sequence Number (Sequenznummer)*
  - *Acknowledgement Number (Quittungs- bzw. Bestätigungsnummer)*
  - *Window-Größe*

TCP-Flußkontrolle (2)

- mit der Sequenznummer werden die zu sendenden TCP-Segmente fortlaufend numeriert
- sie stellt die laufende Nummer in der gesendeten Segmentreihe dar
- mit der Quittungsnummer teilt der Empfänger dem Sender mit, welche Sequenznummer als nächste bei ihm erwartet wird
- seitens des Senders stellt die Window-Größe die maximale Anzahl der Datensegmente dar, die er absenden darf, ohne auf eine Quittung vom Empfänger warten zu müssen
- seitens des Empfängers kann die Window-Größe als die maximale Anzahl der Datensegmente gesehen werden, die beim Empfänger sicher aufgenommen werden können

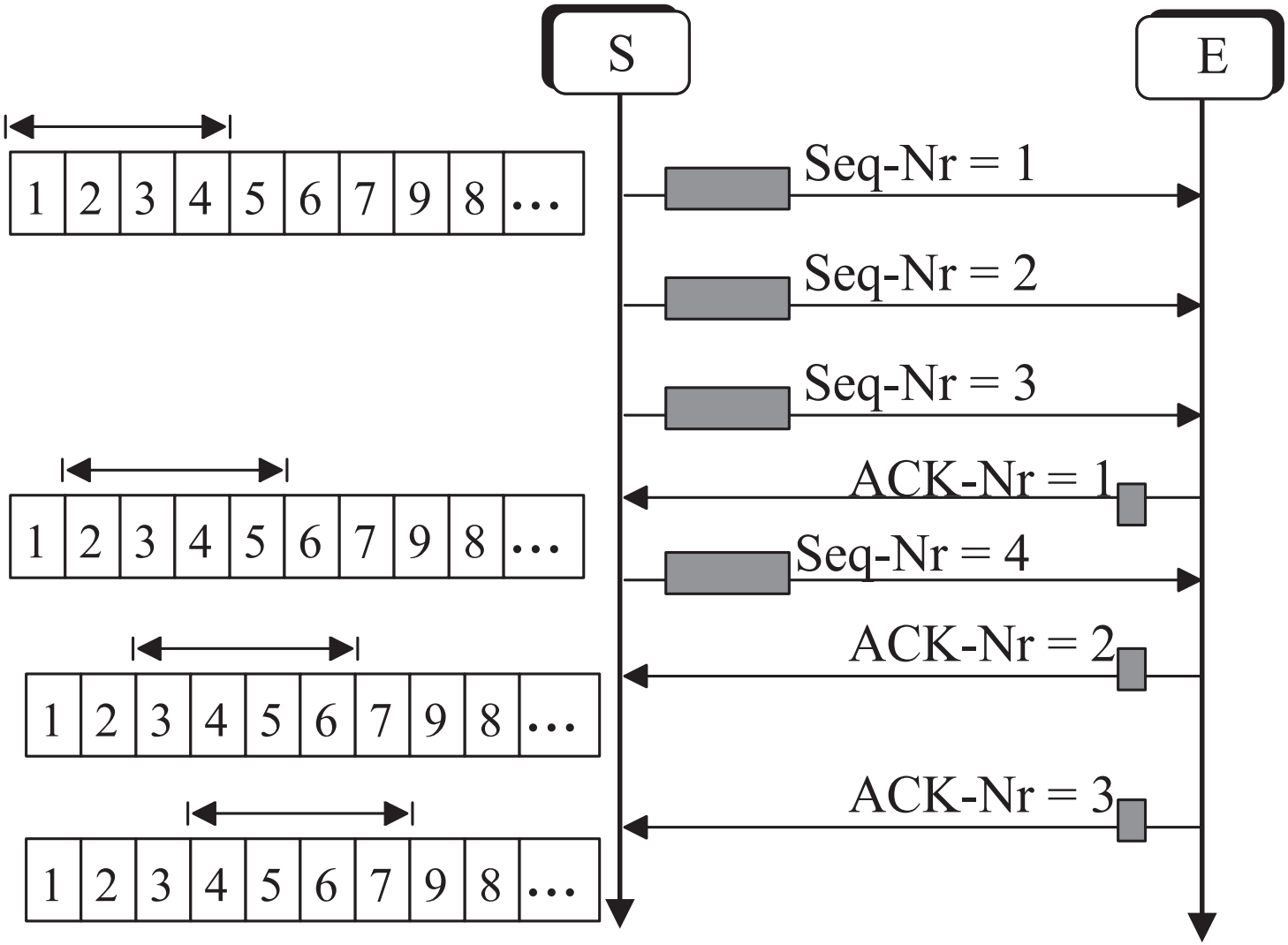
TCP-Flußkontrolle (3)

- wird die maximale Länge von TCP-Datensegmenten festgelegt, so kann die übertragene Datenmenge (die Daten, die „unterwegs“ sind) mit diesen drei Parametern immer kontrolliert werden
- Beispiel: Window-Größe = 4
- da die Window-Größe 4 beträgt, darf der Sender 4 Datensegmente absenden, ohne auf eine Quittung warten zu müssen
- dies bedeutet, daß er die Segmente mit den Nummern 1, 2, 3 und 4 absenden darf
- trifft nach dem Absenden der ersten drei Segmente nun eine Quittung für das erste Segment ein
- so verschiebt sich das Fenster mit den zulässigen Sequenznummern um eine Position nach rechts

TCP-Flußkontrolle (4)

- da maximal 4 Segmente unterwegs sein dürfen, kann der Sender nun die nächsten beiden Segmente mit den Nummern 4 und 5 senden
- trifft nun nach dem Absenden des Segments Nr. 5 die Quittung für SEQ=2 ein
- so verschiebt sich das Fenster wieder nach rechts um eine Position weiter
- . . .

TCP-Flußkontrolle (5)



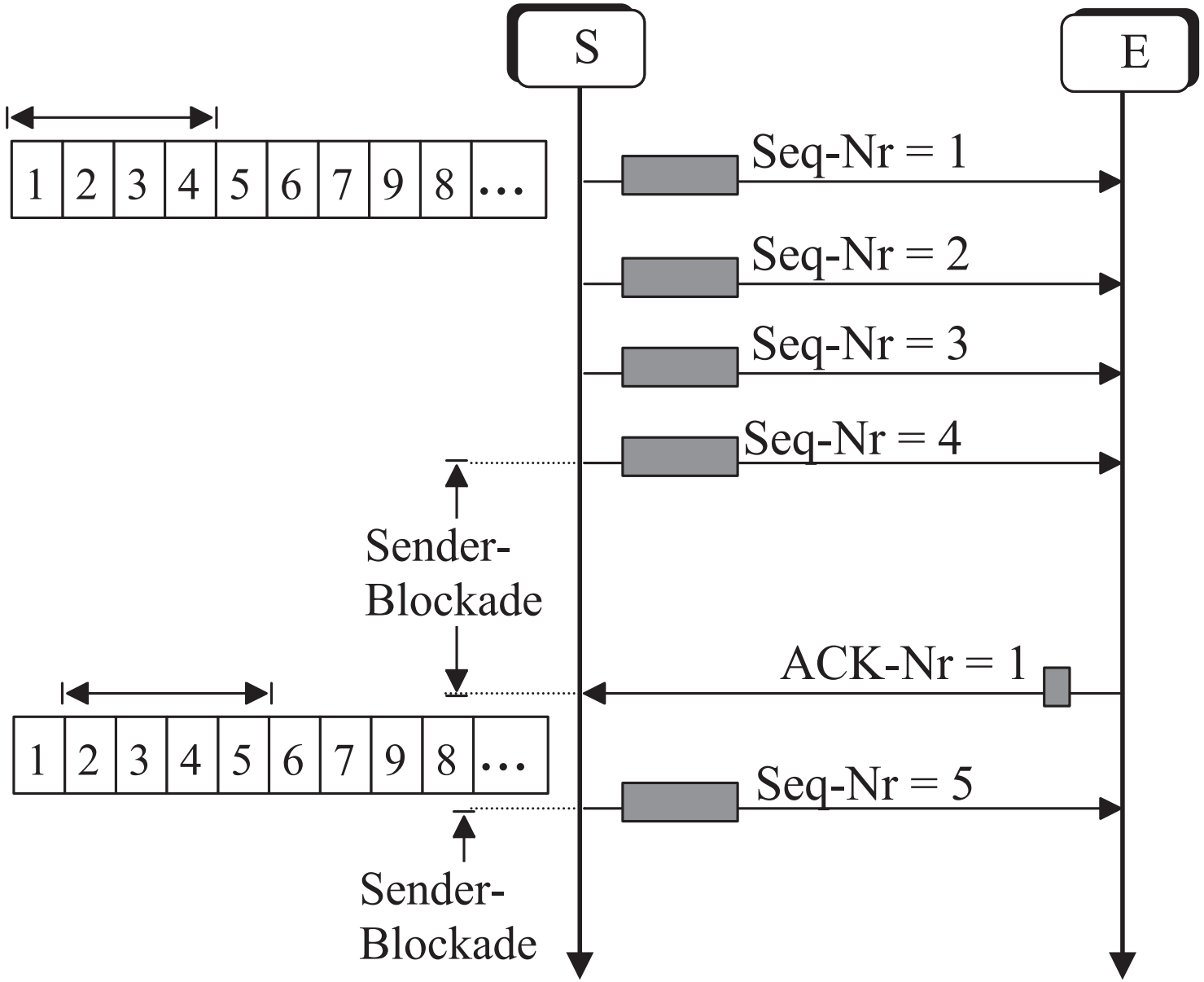
TCP-Flußkontrolle (6)

- nicht immer läuft der Prozeß so optimal
- so kann es zu einer *Sendeblockade* kommen
- diese kommt dann oft vor, wenn einerseits die Verzögerungszeit im Netz groß und andererseits die Window-Größe zu klein ist
- mit großen Verzögerungszeiten ist immer zu rechnen, wenn z.B. eine Satelliten-strecke als ein Übertragungsabschnitt eingesetzt wird
- muß der Sender nach dem Absenden der Segmente mit Sequenznummer 1, 2, 3 und 4 noch immer auf eine Quittung dazu warten
- so kann er das Fenster nicht weiterschieben und ist blockiert

TCP-Flußkontrolle (7)

- bevor einige Segmente quittiert werden, darf der Sender keine weiteren Segmente senden
- nach dem Eintreffen der Quittung für das Segment mit Sequenznummer 1 verschiebt sich das Sendefenster um eine Position nach rechts
- das Segment mit der Sequenznummer 5 darf nun gesendet werden
- anschließend muß der Sendevorgang wiederum bis zum Eintreffen der nächsten Quittung blockiert werden

TCP-Flußkontrolle (8)



TCP-Sliding-Window (1)

- bei TCP wird eine modifizierte Version des Sliding-Window-Prinzips verwendet:
- die Window-Größe wird in Bytes angegeben
- und nicht in der Anzahl der Segmente
- Segmente können somit unterschiedlich groß sein
- für den Empfangspuffer kann auch keine Segmentanzahl angegeben werden
- sondern nur eine Puffergröße

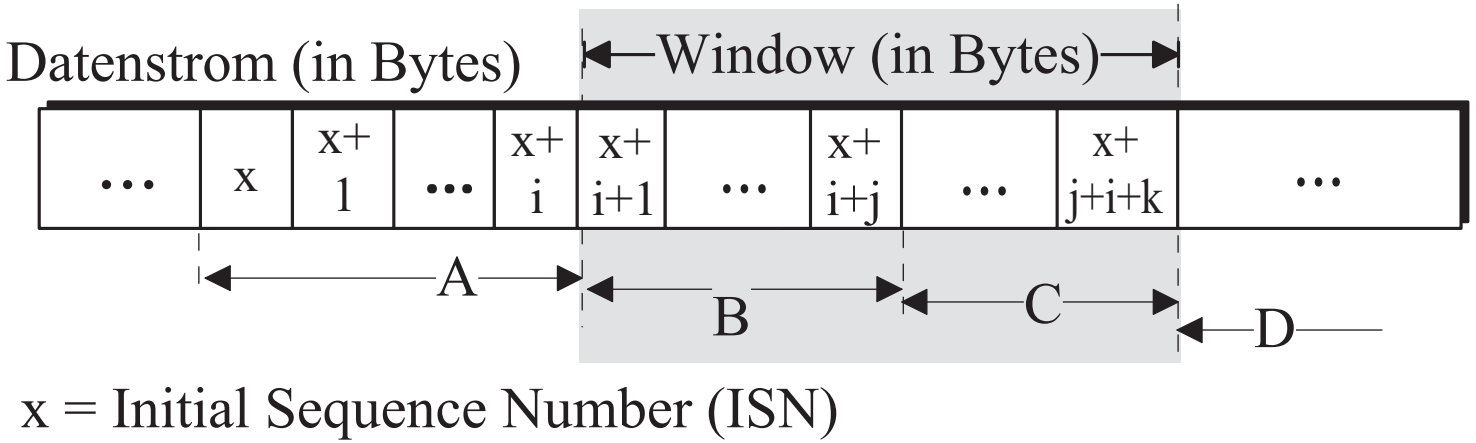
TCP-Sliding-Window (2)

- die Window-Größe legt die maximale Anzahl von Bytes fest
- die die Sendeseite absenden darf, ohne auf eine Quittung vom Ziel warten zu müssen
- mit dem Parameter Window wird ein Bereich von Nummern markiert, die den zu sendenden Datenbytes zuzuordnen sind
- dieser Bereich ist das *Sendefenster*
- vier Bereiche A, B, C und D im Strom von Datenbytes müssen unterschieden werden

TCP-Sliding-Window (3)

- A: i Datenbytes, die abgesendet und bereits vom Zielrechner positiv quittiert wurden
- B: j Datenbytes, die abgesendet und vom Zielrechner noch nicht quittiert wurden
- C: k Datenbytes, die noch abgesendet werden dürfen, ohne auf eine Quittung warten zu müssen
- D: Datenbytes außerhalb des Sendefensters. Diese Datenbytes dürfen erst dann abgesendet werden, wenn der Empfang von einigen vorher abgeschickten Daten bestätigt wird

TCP-Sliding-Window (4)



x = Initial Sequence Number (ISN)

Aussicht

Themenübersicht für die kommende Vorlesung:

- TCP (Teil 2)
- TCP-Erweiterungen und Verbesserungen
- T/TCP
- Domain Name Service (DNS)

Ende Teil 10. Danke für die Aufmerksamkeit.