Lehrstuhl für Netzarchitekturen und Netzdienste
School of Computation, Information and Technology
Technische Universität München

Grundlagen Rechnernetze und Verteilte Systeme (IN0010)

Übungsblatt 10

29. Juni – 03. Juli 2026

Aufgabe 1 TCP Sequenznummern

Im Folgenden betrachten wir eine Datenübertragung zwischen einem PC und einem Webserver um das Konzept der TCP Sequenz- und Acknowledgementnummern nachzuvollziehen.

a)* Was versteht man unter der *Maximum Segment Size* (MSS) und was ist der Unterschied zur *Maximum Transmission Unit* (MTU)?

b)* Berechnen Sie die MSS wie sie bei Verwendung von IPv4 und IPv6 über Ethernet (MTU = 1500 B) auftritt. Gehen Sie davon aus, dass keine Optionen oder Extension Header verwendet werden.

Der PC fordert nun mittels HTTP 1.1 über IPv6 ein Textdokument an, welches auf dem Webserver gehostet wird. Dazu stellt er ein Request mit einer Länge von $L_{\text{Request}} = 430\,\text{B}$. Der Webserver beantwortet die Anfrage zunächst direkt mit dem Response HTTP Header in einem eigenen Segment mit $L_{\text{Response}} = 160\,\text{B}$. Daraufhin sendet er die Textdatei mit einer Länge von $L_{\text{Data}} = 3080\,\text{B}$, welche er aufgrund der Größe auf mehrere Segmente aufteilt.

c)* Welches HTTP Kommando wird benutzt um das Textdokument vom Server anzufragen?

d) Wie viele Daten-Segmente wird der Webserver zum Client zurücksenden?

e) Bestimmen Sie die Größe der Payload in den Daten-Segmenten.

Im Folgenden wollen wir den Verlauf aller gesendeten Segmente nachvollziehen und betrachten jeweils die verwendeten Sequenz- und Acknowledgementnummern, die gesetzten Flags und die Payloadlänge. PC und Server wählen dabei `8999` und `1839` als initiale Sequenznummern. Beachten Sie, dass sich die Sequenznummern in Segmenten immer auf die eigene initiale Sequenznummer beziehen, während die sich die Acknowledgementnummern immer auf die initiale Sequenznummer des Gegenübers beziehen.

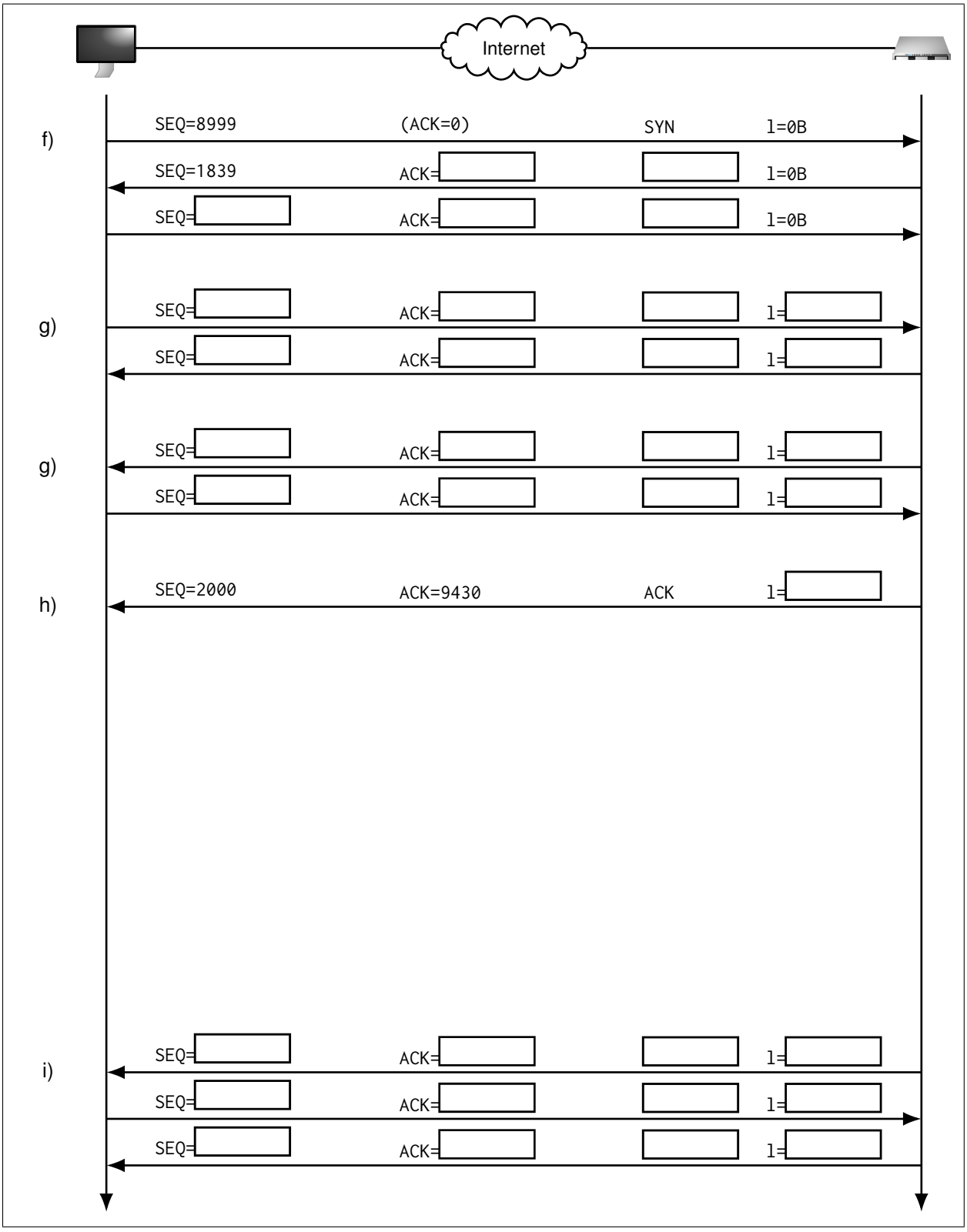
Zunächst initiiert der PC den Verbindungsaufbau und stellt die HTTP Anfrage. Der Server antwortet dann mit der HTTP Response welche nur den Header enthält und sendet danach die Datensegmente. Nachdem er alle Segmente gesendet hat initiiert der Server auch den Verbindungsabbau. Gehen Sie davon aus, dass jedes Datensegment individuell und umgehend mit einem eigenen Acknowledgement bestätigt wird.

f) Vervollständigen Sie den Verbindungsaufbau im Diagramm.

g) Vervollständigen Sie die Segmente zur Übermittlung des HTTP-Requests und der HTTP-Response.

h) Ergänzen Sie die Segmente für die Übermittlung der Daten und die jeweiligen Bestätigungen.

i) Vervollständigen Sie den Verbindungsabbau im Diagramm.



Aufgabe 2 Fluss- und Staukontrolle bei TCP

Das im Internet am weitesten verbreitete Transportprotokoll ist TCP. Dieses implementiert Mechanismen zur Fluss- und Staukontrolle.

a)* Diskutieren Sie die Unterschiede zwischen Fluss- und Staukontrolle. Welche Ziele werden mit dem jeweiligen Mechanismus verfolgt?

b) Ordnen Sie die folgenden Begriffe jeweils der TCP-Fluss- bzw. Staukontrolle zu:

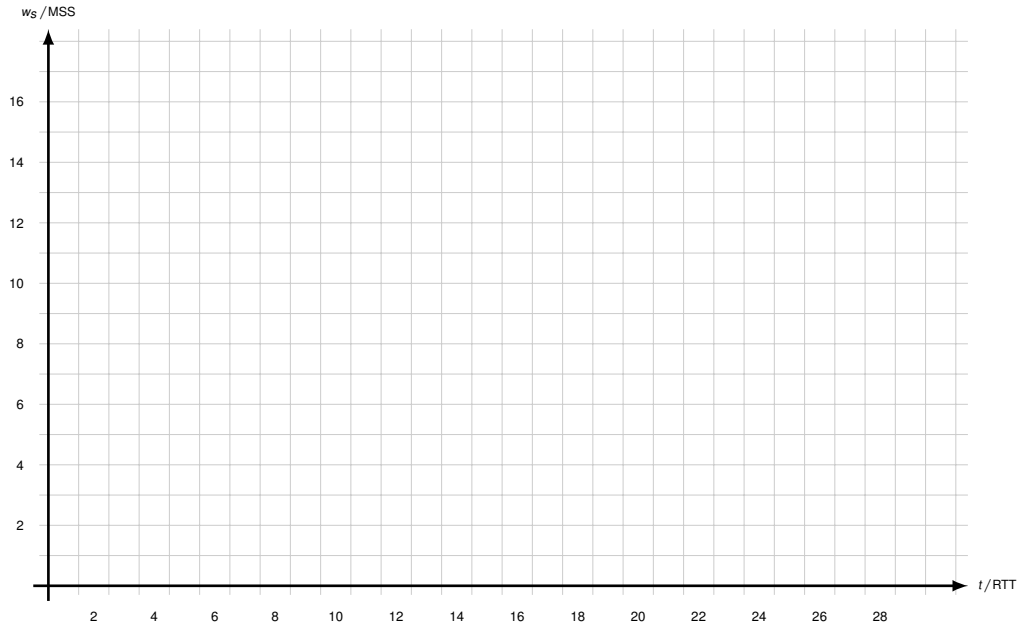
- Slow-Start
- Empfangsfenster
- Congestion-Avoidance
- Multiplicative-Decrease

Zur Analyse der mit TCP erzielbaren Datenrate betrachten wir den Verlauf einer zusammenhängenden Datenübertragung, bei der die Slow-Start-Phase bereits abgeschlossen ist. TCP befinde sich also in der Congestion-Avoidance-Phase. Wir bezeichnen die einzelnen Fenster wie folgt:

- Sendefenster W_s, $|W_s| = w_s$
- Empfangsfenster W_r, $|W_r| = w_r$
- Staukontrollfenster W_c, $|W_c| = w_c$

Wir gehen davon aus, dass das Empfangsfenster beliebig groß ist, so dass das Sendefenster allein durch das Staukontrollfenster bestimmt wird, d. h. $W_s = W_c$. Es treten keinerlei Verluste auf, solange das Sendefenster kleiner als ein Maximalwert x ist, also $w_s < x$.
Wird ein vollständiges Sendefenster bestätigt, so vergrößert sich das aktuell genutzte Fenster um genau 1 MSS.
Hat das Sendefenster den Wert x erreicht, so geht genau eines der versendeten TCP-Segmente verloren. Den Verlust erkennt der Sender durch mehrfachen Erhalt derselben ACK-Nummer. Daraufhin halbiert der Sender das Staukontrollfenster, bleibt aber nach wie vor in der Congestion-Avoidance-Phase, d. h. es findet kein erneuter Slow-Start statt. Diese Vorgehensweise entspricht einer vereinfachten Variante von TCP-Reno (vgl. Vorlesung).
Als konkrete Zahlenwerte nehmen wir an, dass die maximale TCP-Segmentgröße (MSS) 1460 B und die RTT 200 ms beträgt. Die Serialisierungszeit von Segmenten sei gegenüber der Ausbreitungsverzögerung vernachlässigbar klein. Segmentverlust trete ab einer Sendefenstergröße von $w_s \geq x = 16$ MSS auf.

c)* Erstellen Sie ein Schaubild, in dem die aktuelle Größe des Sendefenster w_s gemessen in MSS über der Zeitachse t gemessen in RTT aufgetragen ist. In Ihrem Diagramm soll zum Zeitpunkt $t_0 = 0\,\text{s}$ gerade die Sendefenstergröße halbiert worden sein, also $w_s = x/2$ gelten. Zeichnen Sie das Diagramm im Zeitintervall $t = \{0, \ldots, 27\}$.



d)* Wieviel Zeit vergeht, bis nach einem Segmentverlust das Staukontrollfenster infolge eines weiteren Segmentverlusts wieder reduziert wird?

e)* Bestimmen Sie allgemein die durchschnittliche Verlustrate θ. Hinweis: Da das Verhalten von TCP in diesem idealisierten Modell periodisch ist, reicht es aus, lediglich eine Periode zu betrachten. Setzen Sie die Gesamtzahl übertragener Segmente in Relation zur Anzahl verlorener Segmente (Angabe als gekürzter Bruch ist ausreichend).

f) Bestimmen Sie mit Hilfe der Ergebnisse aus den Teilaufgaben d) und e) die in der betrachteten TCP-Übertragungsphase durchschnittlich erzielbare Übertragungsrate in kB/s.
Hinweis: Verwenden Sie den exakten Wert (Bruch) aus Teilaufgabe e).

g)* Bis zu welcher Übertragungsrate könnten Sie mit UDP maximal über den Kanal senden, ohne einen Stau zu erzeugen? Berücksichtigen Sie, dass der UDP-Header 12 B kleiner als der TCP-Header ohne Optionen ist.

Aufgabe 3 TCP und Long Fat Networks (Zusatzaufgabe)

In dieser Aufgabe betrachten wir sog. *Long Fat Networks*. Darunter versteht man Verbindungen, welche zwar eine hohe Übertragungsrate aber insbesondere auch eine hohe Verzögerung aufweisen. Beispiele dafür sind u. a. Satellitenverbindungen in Folge der hohen Ausbreitungsverzögerungen. Wir wollen insbesondere die Auswirkungen auf die TCP-Staukontrolle untersuchen.

a)* Bei TCP wird das Sendefenster in Abhängigkeit des Empfangsfensters sowie des Staukontrollfensters gewählt. Wie lautet der genaue Zusammenhang?

Zwei Nutzer seien nun über einen geostationären Satelliten an das Internet mit hoher Übertragungsrate angebunden. Die RTT zwischen beiden Nutzern betrage 800 ms, die Übertragungsrate sei $r = 24\,\text{Mbit}/\text{s}$.

b)* Wie groß muss das Sendefenster (gemessen in Byte) gewählt werden, damit kontinuierlich gesendet werden kann?

c)* Warum ist die Situation in Teilaufgabe b) ein Problem für die TCP-Flusskontrolle?

d)* Lesen Sie Sektion 2 von RFC 1323 (`http://www.ietf.org/rfc/rfc1323.txt`, siehe Anhang). Beschreiben Sie die Lösung für das Problem aus Teilaufgabe c).

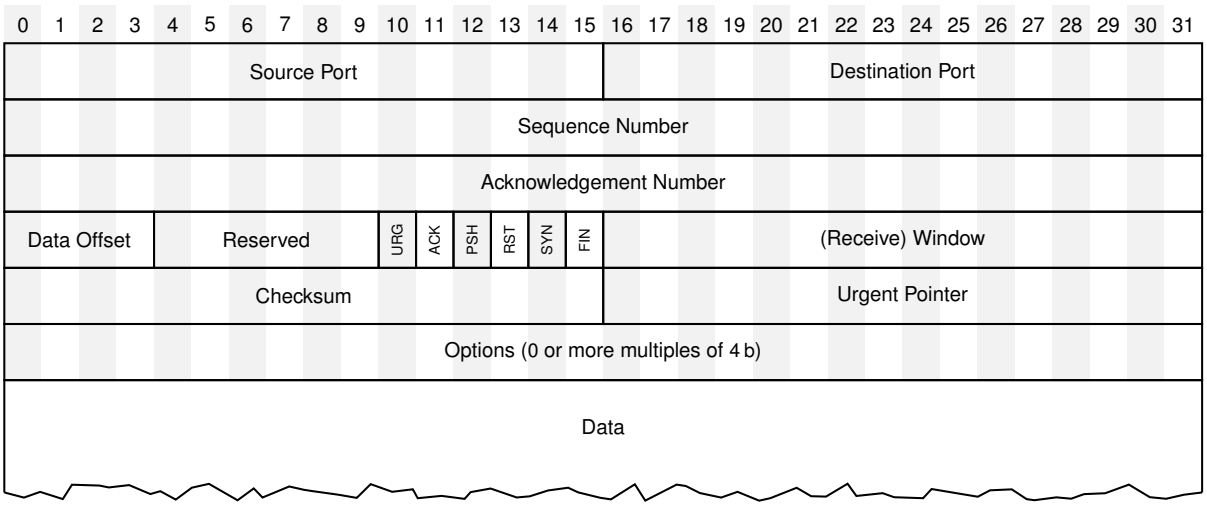
e) Bestimmen Sie den minimalen Wert für das `shift.cnt`-Feld der TCP-Window-Scaling-Option.

f) Geben Sie den Header des ersten TCP-SYN-Pakets an, welches die Verbindung aufbaut. Verwenden Sie dazu die konkreten Zahlenwerte aus der Angabe. Ein TCP-Header ist zur Erinnerung nochmals in Abbildung 3.1 dargestellt. Dort finden sich auch zwei Vordrucke zur Lösung.
Hinweis: Es ist nicht notwendig, den Header binär auszufüllen. Machen Sie aber bitte deutlich, ob es sich um hexadezimale, dezimale oder binäre Darstellung der Zahlen handelt.

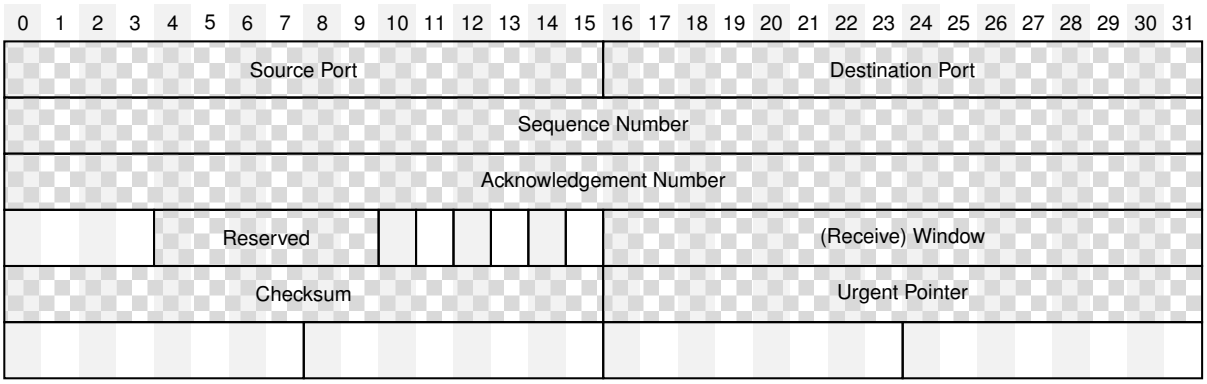
Angenommen die Größe des Staukontrollfensters betrage derzeit die Hälfte des in Teilaufgabe b) berechneten Werts. Die MSS betrage 1200 B und die TCP-Verbindung befinde sich derzeit in der Congestion-Avoidance-Phase.

g) Wie lange dauert es, bis das Fenster die Leitung komplett ausnutzen kann?
Hinweis: Das Staukontrollfenster wird durch TCP-Window-Scaling nicht beeinflusst.

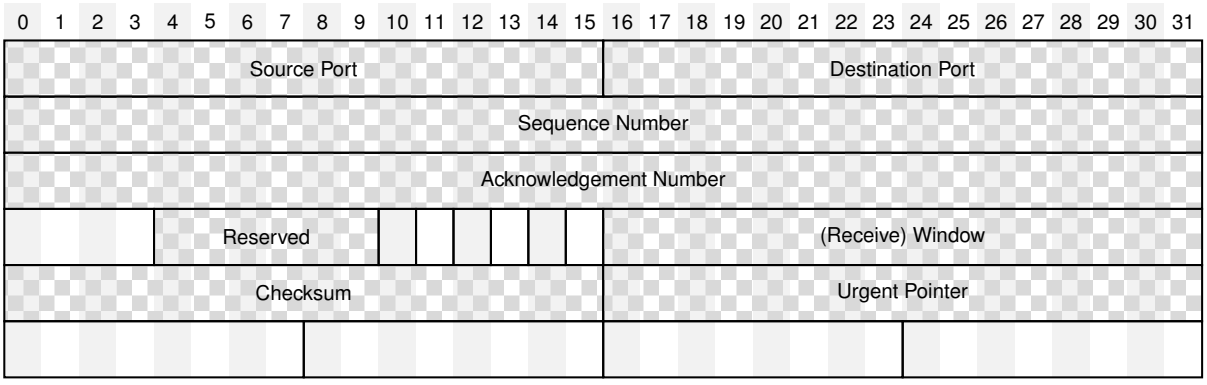
h) Ergibt sich aus dem Ergebnis von Teilaufgabe g) ein Problem?



(a) TCP-Header



(b) Vordruck



(c) Noch ein Vordruck, falls man sich vermalt hat

Abbildung 3.1: TCP-Header und Vordrucke zur Lösung von Aufgabe 3

2. TCP WINDOW SCALE OPTION

2.1 Introduction

The window scale extension expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32- bit value in the 16-bit Window field of the TCP header (SEG.WND in RFC-793). The scale factor is carried in a new TCP option, Window Scale. This option is sent only in a SYN segment (a segment with the SYN bit on), hence the window scale is fixed in each direction when a connection is opened. (Another design choice would be to specify the window scale in every TCP segment. It would be incorrect to send a window scale option only when the scale factor changed, since a TCP option in an acknowledgement segment will not be delivered reliably (unless the ACK happens to be piggy-backed on data in the other direction). Fixing the scale when the connection is opened has the advantage of lower overhead but the disadvantage that the scale factor cannot be changed during the connection.)

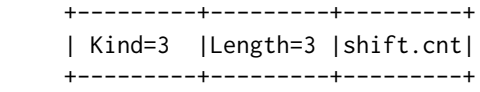
The maximum receive window, and therefore the scale factor, is determined by the maximum receive buffer space. In a typical modern implementation, this maximum buffer space is set by default but can be overridden by a user program before a TCP connection is opened. This determines the scale factor, and therefore no new user interface is needed for window scaling.

2.2 Window Scale Option

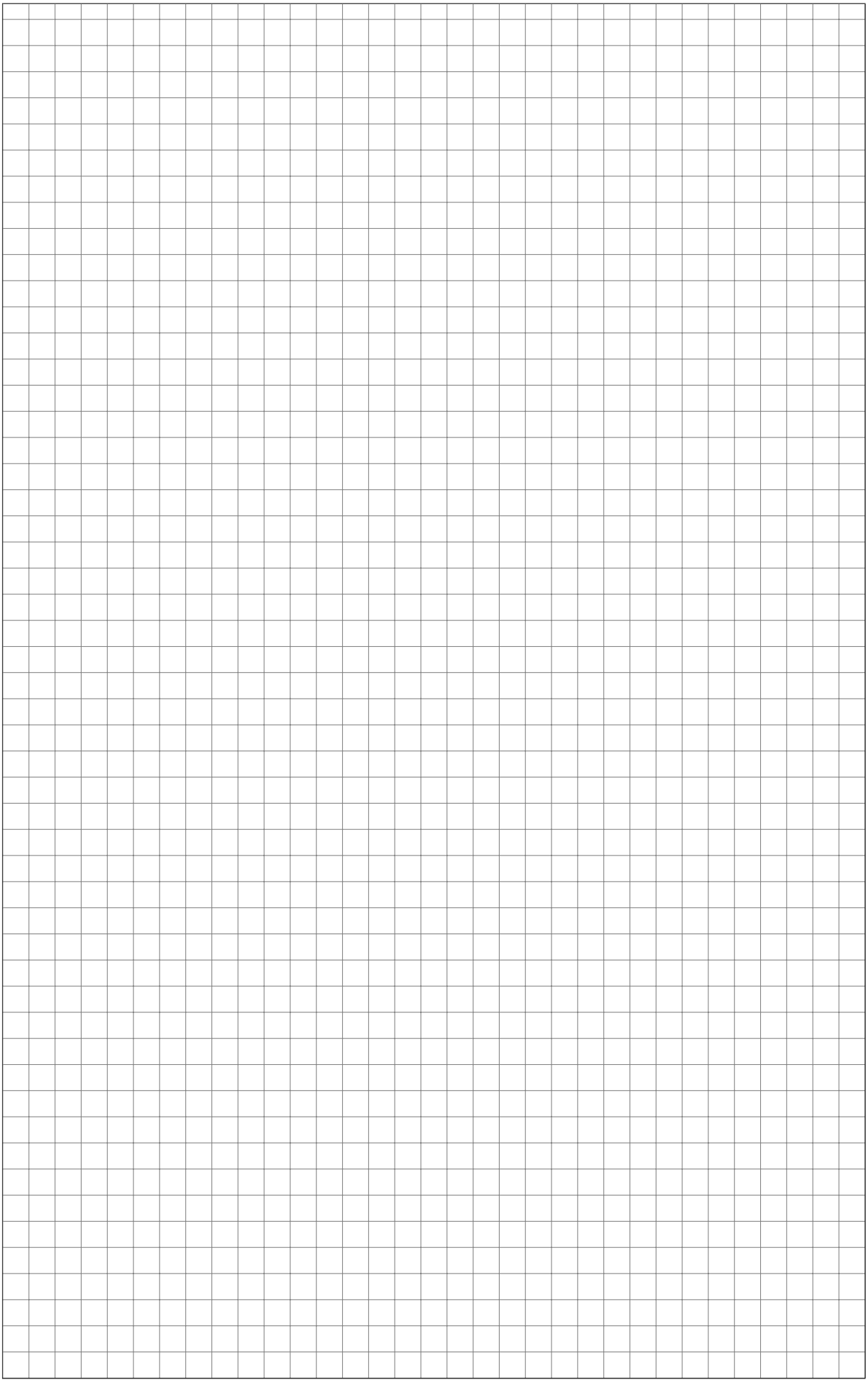
The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes: (1) indicate that the TCP is prepared to do both send and receive window scaling, and (2) communicate a scale factor to be applied to its receive window. Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.

TCP Window Scale Option (WSopt):

Kind: 3 Length: 3 bytes



This option is an offer, not a promise; both sides must send Window Scale options in their SYN segments to enable window scaling in either direction. If window scaling is enabled, then the TCP that sent this option will right-shift its true receive-window values by 'shift.cnt' bits for transmission in SEG.WND. The value 'shift.cnt' may be zero (offering to scale, while applying a scale factor of 1 to the receive window).

This option may be sent in an initial <SYN> segment (i.e., a segment with the SYN bit on and the ACK bit off). It may also be sent in a <SYN,ACK> segment, but only if a Window Scale op- tion was received in the initial <SYN> segment. A Window Scale option in a segment without a SYN bit should be ignored.

The Window field in a SYN (i.e., a <SYN> or <SYN,ACK>) segment itself is never scaled.