



ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ

Στρώμα μεταφοράς
στο Internet



- Αρχές λειτουργίας του στρώματος μεταφοράς και βασικές υπηρεσίες του
 - πολυπλεξία/αποπολυπλεξία
 - αξιόπιστη μεταφορά δεδομένων
 - έλεγχος ροής
 - έλεγχος σφαλμάτων
- Πρωτόκολλα στρώματος μεταφοράς στο Internet:
 - UDP: μεταφορά χωρίς σύνδεση
 - TCP: μεταφορά με σύνδεση

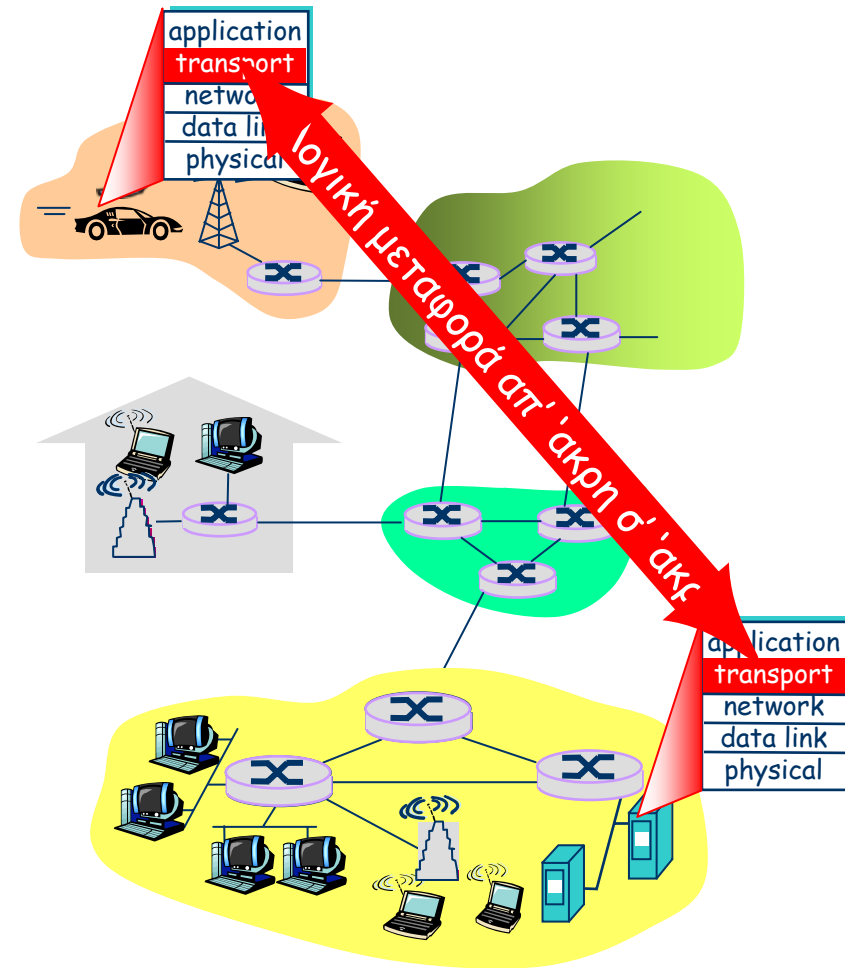


- Υπηρεσίες και πρωτόκολλα του στρώματος μεταφοράς
- Στρώμα μεταφοράς στο Internet
 - Πολυπλεξία/αποπολυπλεξία
- Μεταφορά χωρίς σύνδεση: UDP
- Αξιόπιστη μεταφορά δεδομένων
- Μεταφορά με σύνδεση: TCP
 - υπηρεσία συρμού byte
 - δομή τεμαχίου
 - διαχείριση συνδέσεων
 - μεταφορά δεδομένων
 - έλεγχος ροής
 - έλεγχος σφαλμάτων (επαληθεύσεις, επαναμεταδόσεις)
 - διαχείριση χρονομετρητών

Στρώμα μεταφοράς: Υπηρεσίες και πρωτόκολλα



- παρέχει **Λογική επικοινωνία** μεταξύ διαδικασιών εφαρμογής που τρέχουν σε διαφορετικούς host
- τα πρωτόκολλα μεταφοράς τρέχουν στις τερματικές διατάξεις
 - **Εκπομπή**: χωρίζει τα μηνύματα εφαρμογής σε **Τεμάχια** και τα διοχετεύει στο στρώμα δικτύου
 - **Λήψη**: επανασυναρμολογεί τα τεμάχια σε μηνύματα και τα διοχετεύει στο στρώμα εφαρμογής
- υπάρχουν περισσότερα από ένα πρωτόκολλα μεταφοράς διαθέσιμα στις εφαρμογές
 - Internet: TCP and UDP



Στρώμα μεταφοράς και στρώμα δικτύου



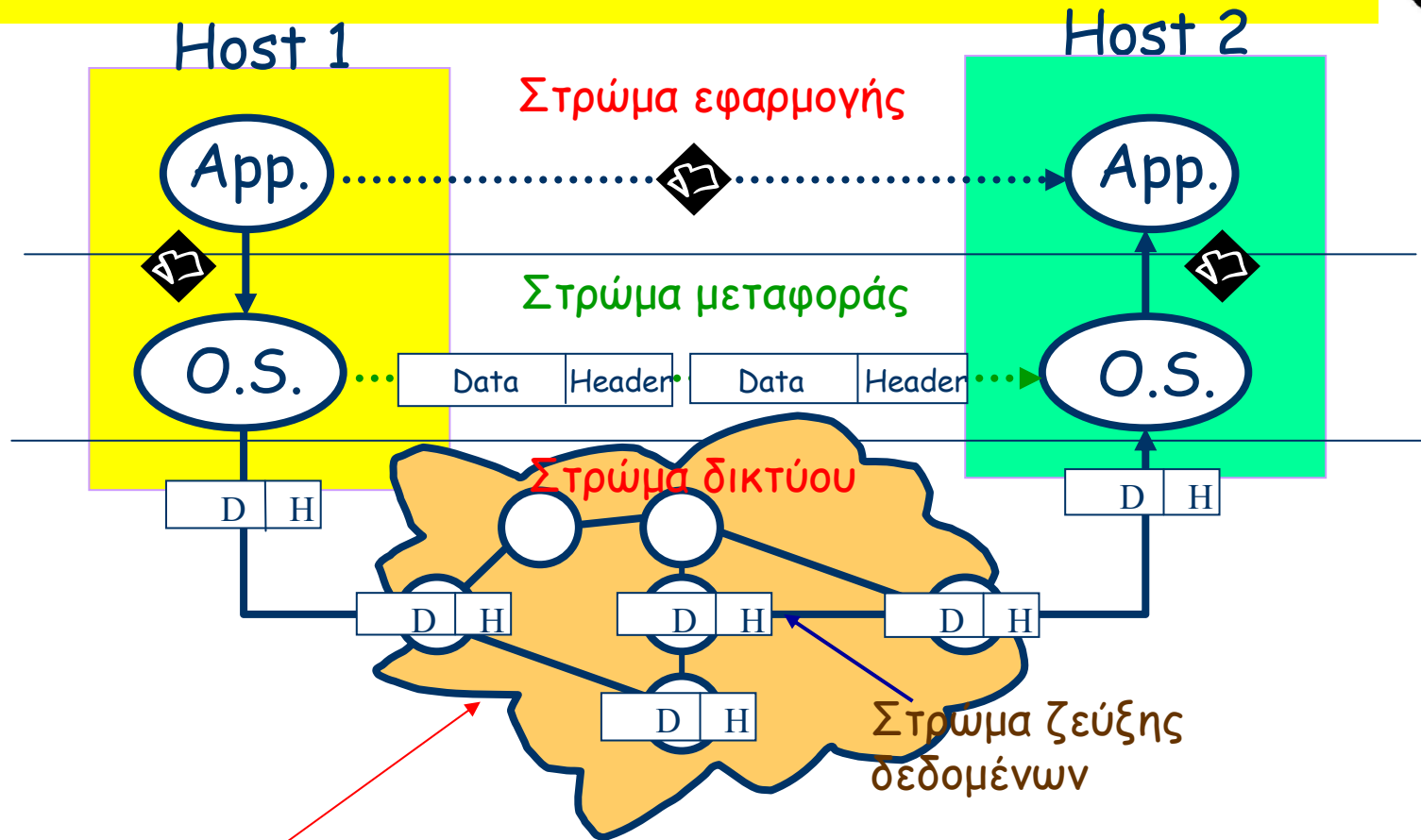
- *στρώμα δικτύου*: μεταφορά δεδομένων (λογική επικοινωνία) μεταξύ τερματικών διατάξεων
- *στρώμα μεταφοράς*: μεταφορά δεδομένων (λογική επικοινωνία) μεταξύ διαδικασιών εφαρμογής
 - βασίζεται στις υπηρεσίες δικτύου και τις βελτιώνει
- Αναλογία με αλληλογραφία:
 - Δύο σπίτια, σε διαφορετικές πόλεις, με αρκετά παιδιά στο κάθε σπίτι που αλληλογραφούν μεταξύ τους. Η Ελένη στο ένα σπίτι και ο Γιώργος στο άλλο μαζεύουν και μοιράζουν την αλληλογραφία.
 - διαδικασίες εφαρμογής = παιδιά
 - μηνύματα εφαρμογής = επιστολές σε φακέλους
 - hosts = σπίτια
 - πρωτόκολλο μεταφοράς = Ελένη και Γιώργος
 - πρωτόκολλο στρώματος δικτύου = ταχυδρομική υπηρεσία

Στρώμα μεταφοράς και στρώμα ζεύξης



- Τα πρωτόκολλα μεταφοράς έχουν κοινά χαρακτηριστικά με εκείνα της ζεύξης δεδομένων, αλλά υπάρχουν και σημαντικές διαφορές:
 - Στο στρώμα μεταφοράς απαιτείται η ρητή διευθυνσιοδότηση των προορισμών
 - Η αρχική εγκατάσταση της σύνδεσης είναι πιο πολύπλοκη
 - Ενδεχόμενη ύπαρξη αποθηκευτικής χωρητικότητας στο δίκτυο
 - Ποσοτική διαφορά (μεγάλος και δυναμικά μεταβαλλόμενος αριθμός συνδέσεων)
- Η προσωρινή αποθήκευση και ο έλεγχος ροής απαιτεί διαφορετική προσέγγιση στο στρώμα μεταφοράς απ' ότι στο στρώμα ζεύξης δεδομένων

Στρώμα μεταφοράς



- Υπηρεσία best-effort
- Πακέτα περιορισμένου μήκους
- Πακέτα καθυστερούν, χάνονται, ...
- Επικοινωνία μεταξύ host

- Ποια εφαρμογή λαμβάνει ποια πακέτα;
- Με ποιο ρυθμό πρέπει να στέλνουν οι host στο δίκτυο;

Στρώμα μεταφοράς

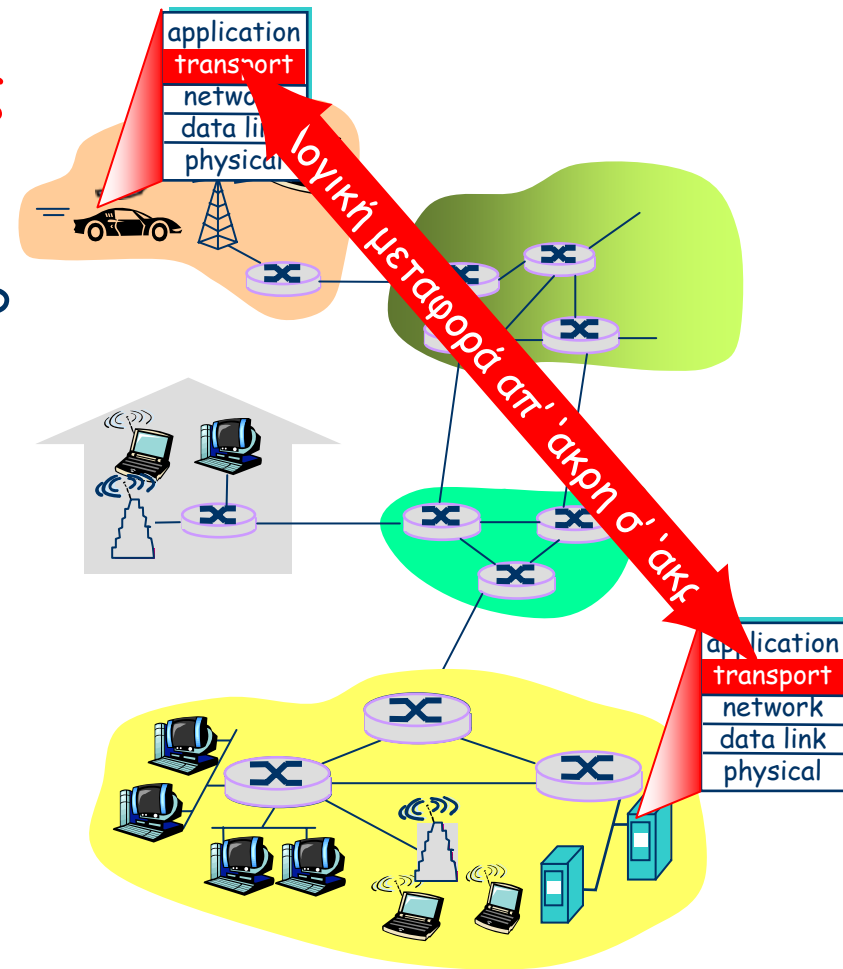


- Παρέχει αξιόπιστη ή μη αξιόπιστη μεταφορά δεδομένων μεταξύ διαδικασιών εφαρμογής.
- Υποστηρίζει μηνύματα αυθαίρετου μήκους
- Παρέχει τρόπο απόφασης για το ποια πακέτα πηγαίνουν σε ποιες εφαρμογές (πολυπλεξία/αποπολυπλεξία)
- Ρυθμίζει πότε οι host πρέπει να στέλνουν

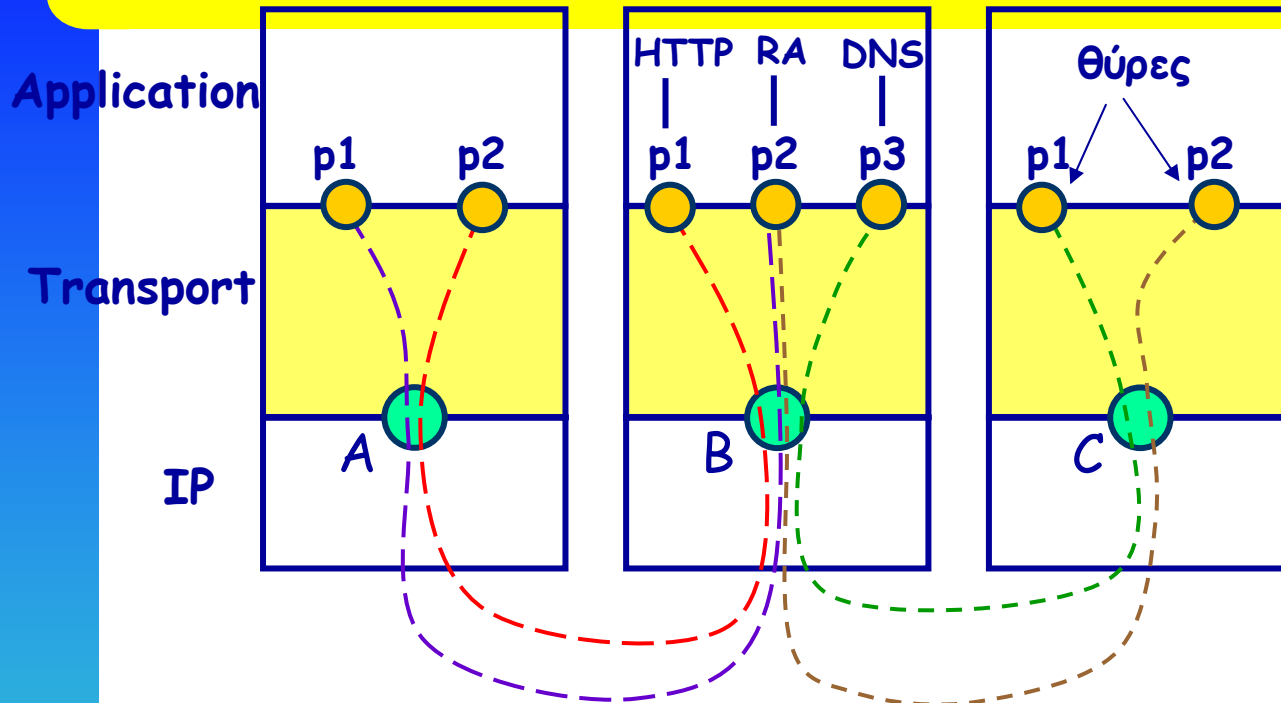
Στρώμα μεταφοράς στο Internet



- μη αξιόπιστη παράδοση χωρίς διατήρηση της σειράς: UDP
 - μινιμαλιστική επέκταση της υπηρεσίας "best-effort" του IP
- αξιόπιστη παράδοση με διατήρηση της σειράς: TCP
 - εγκατάσταση σύνδεσης
 - έλεγχος ροής
 - έλεγχος συμφόρησης
- υπηρεσίες που δεν προσφέρονται:
 - εξασφάλιση καθυστέρησης
 - εξασφάλιση εύρους ζώνης



Στρώμα μεταφοράς στο Internet



- UDP: αναξιόπιστη μεταφορά
- TCP: αξιόπιστη, με τη σειρά

- Και το TCP και το UDP επεκτείνουν την επικοινωνία μεταξύ host, του IP, σε επικοινωνία μεταξύ διαδικασιών εφαρμογής: **πολυπλεξία/αποπολυπλεξία** του στρώματος μεταφοράς
- Ποια πακέτα λαμβάνει η κάθε εφαρμογή;
 - **Λύση**: αντιστοίχιση κάθε υποδοχής σε μια **θύρα**
 - Κάθε τεμάχιο να έχει ειδικά πεδία για τους αριθμούς θυρών
 - Οι υποδοχές προσδιορίζονται μονοσήμαντα



- Χώρος διευθύνσεων θυρών των 16-bit για το UDP και το TCP
- Ο client πρέπει να ξέρει τη θύρα του server
- **Πασίγνωστες θύρες (0-1023)**: όλοι συμφωνούν ποιες υπηρεσίες τρέχουν σ' αυτές τις θύρες
 - π.χ., telnet:23, http:80, DHCP server:547, DNS:53
- **Εφήμερες θύρες (1024-65535)**: δίδονται στους client

Πολυπλεξία/αποπολυπλεξία



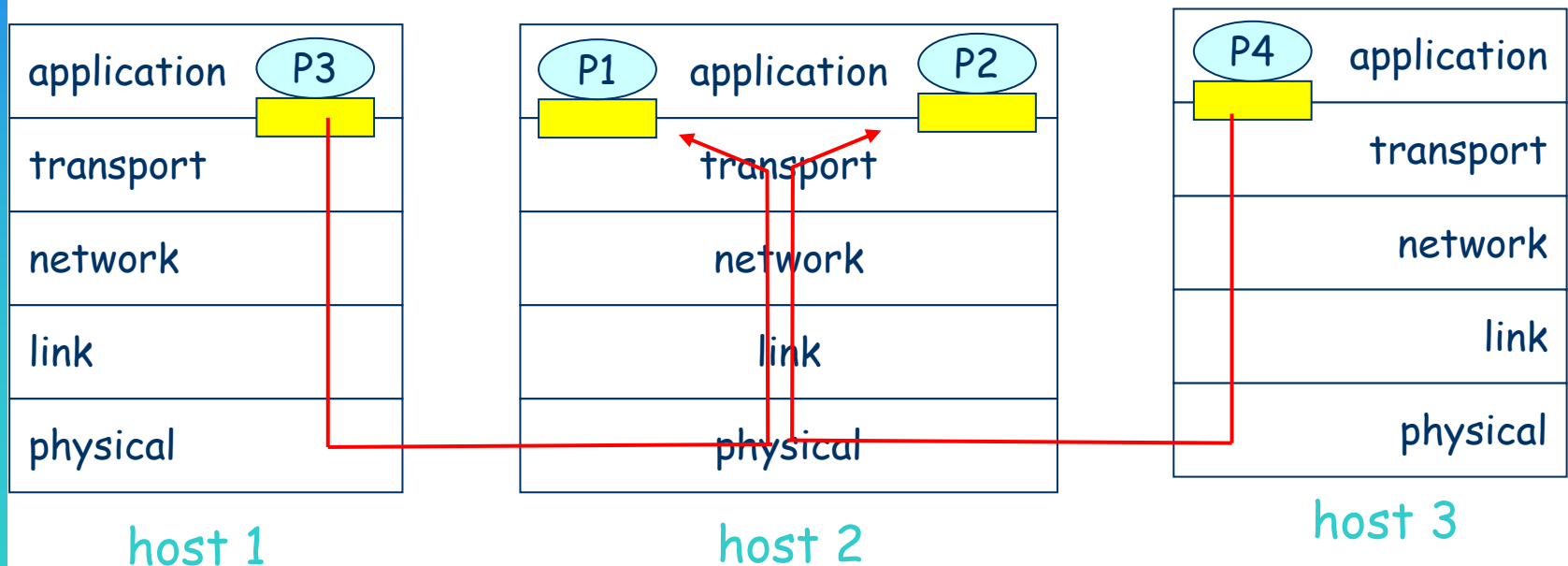
Πολυπλεξία στον host αποστολής:

συλλογή των δεδομένων από πολλές υποδοχές, χαρακτηρισμός τους με επικεφαλίδα (χρησιμοποιείται μετά στην αποπολυπλεξία)

Αποπολυπλεξία στον host λήψης:

παράδοση των λαμβανομένων τεμαχίων στις σωστές υποδοχές

■ = υποδοχή ○ = διαδικασία



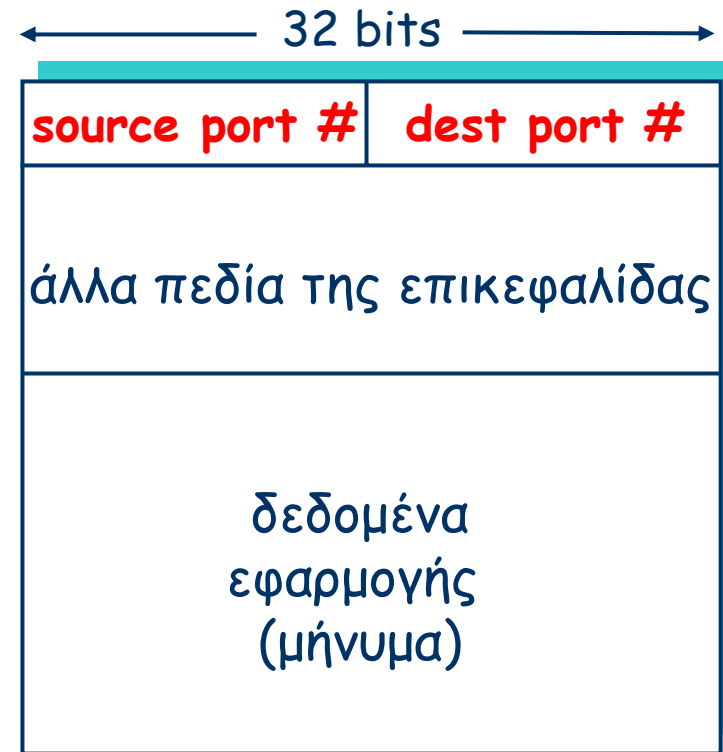


Πώς λειτουργεί η αποπολυπλεξία

- ο host λαμβάνει δεδομενογράμματα IP

- κάθε δεδομένογραμμα έχει διευθύνσεις source IP και destination IP
- κάθε δεδομένογραμμα μεταφέρει 1 τεμάχιο στρώματος μεταφοράς
- κάθε τεμάχιο έχει αριθμούς source και destination port

- ο host χρησιμοποιεί τις διευθύνσεις IP και τους αριθμούς θυρών για να κατευθύνει το τεμάχιο στην κατάλληλη υποδοχή



μορφή τεμαχίου στρώματος μεταφοράς

Αποπολυπλεξία χωρίς σύνδεση



- Δημιουργία υποδοχών με αριθμούς θυρών:

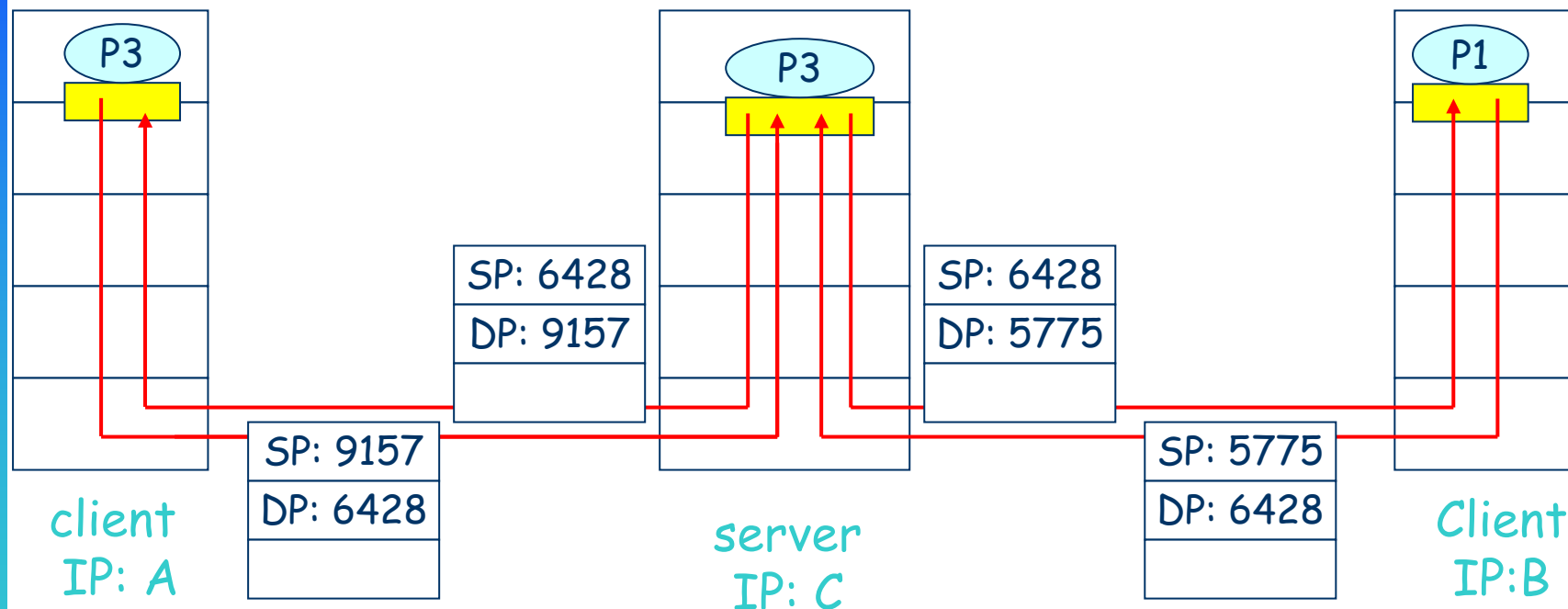
```
DatagramSocket mySocket1 = new DatagramSocket(9157);  
DatagramSocket mySocket2 = new DatagramSocket(9222);
```

- Μια υποδοχή UDP καθορίζεται από το ζεύγος:

(dest IP address, dest port number)

- Όταν ο host λαμβάνει τεμάχιο UDP:
 - ελέγχει τον αριθμό destination port στο τεμάχιο
 - κατευθύνει το τεμάχιο UDP στην υποδοχή με αυτόν τον αριθμό
- Δεδομενογράμματα IP με διαφορετικές διευθύνσεις source IP και/ή source port αριθμούς, αλλά με ίδια διεύθυνση dest IP και ίδιο αριθμό dest port, οδηγούνται στην ίδια υποδοχή

Αποπολυπλεξία χωρίς σύνδεση



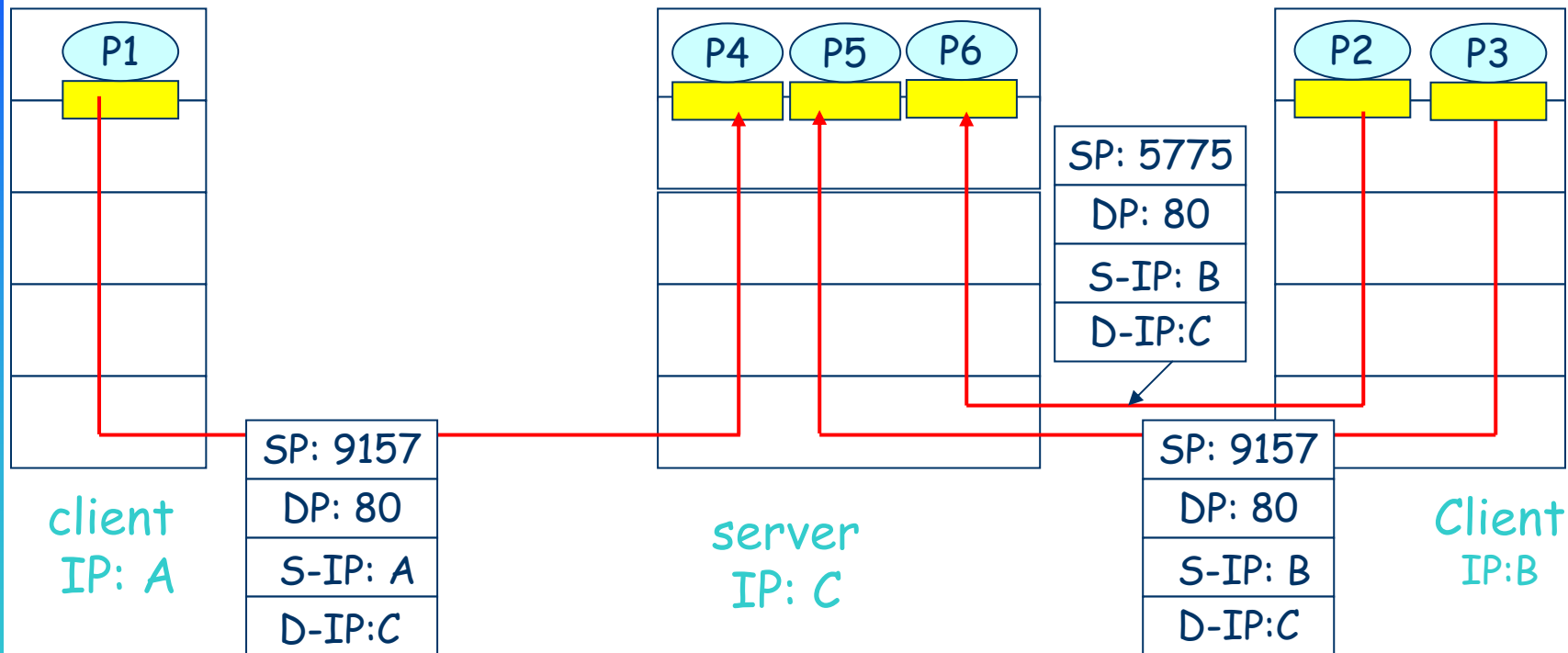
SP είναι η "διεύθυνση επιστροφής"

Αποπολυπλεξία με σύνδεση



- Η υποδοχή TCP καθορίζεται από την τετράδα:
 - source IP address, source port number, dest IP address, dest port number
- Ο host λήψης χρησιμοποιεί και τις 4 τιμές για να κατευθύνει το τεμάχιο στην κατάλληλη υποδοχή
- Ένας Server host μπορεί να υποστηρίζει πολλές υποδοχές TCP ταυτόχρονα:
 - κάθε υποδοχή καθορίζεται από τη δική της τετράδα
- Οι Web servers έχουν διαφορετικές υποδοχές για κάθε συνδεδεμένο client
 - στο μη-επίμονο HTTP έχουμε διαφορετική υποδοχή για κάθε αίτηση/απάντηση

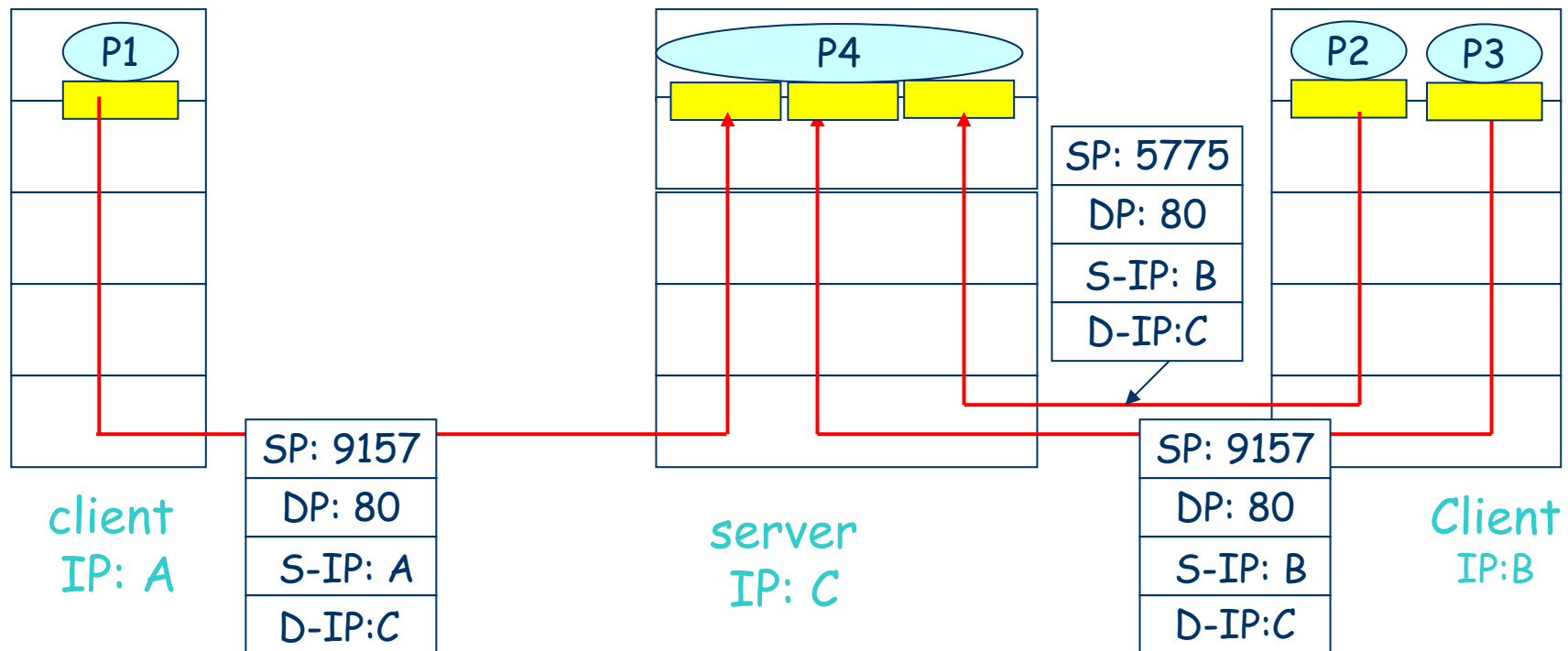
Αποπολυπλεξία με σύνδεση





Αποπολυπλεξία με σύνδεση

Threaded Web Server



UDP: User Datagram Protocol [RFC 768]

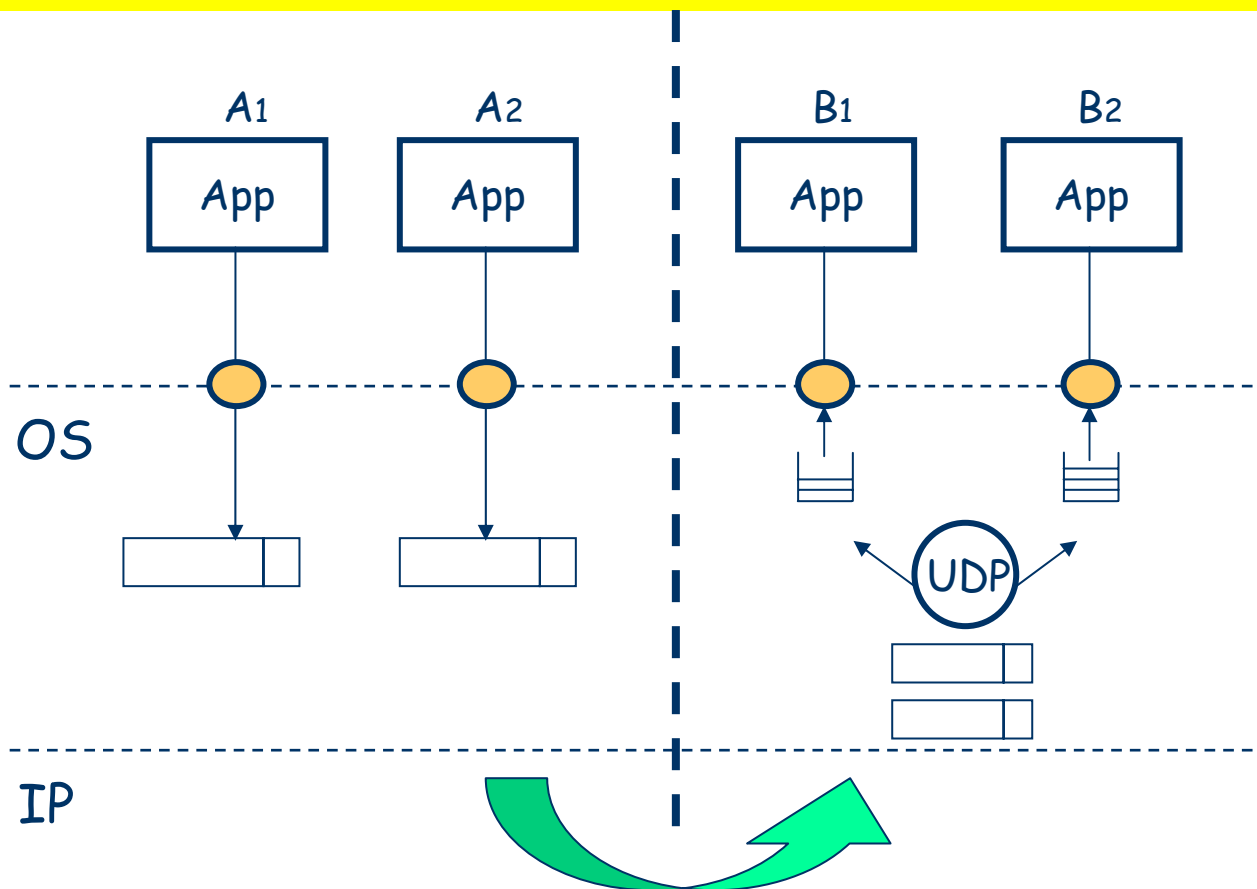


- Λιτό πρωτόκολλο μεταφοράς στο Internet
- υπηρεσία "καλύτερης προσπάθειας", τα τεμάχια UDP μπορεί:
 - να χαθούν
 - να παραδοθούν εκτός σειράς στο ανώτερο στρώμα
- *χωρίς σύνδεση:*
 - όχι χειραψία μεταξύ του πομπού και του δέκτη UDP
 - κάθε τεμάχιο UDP αντιμετωπίζεται ανεξάρτητα από τα άλλα

Γιατί υπάρχει το UDP;

- δεν εγκαθίσταται σύνδεση (που μπορεί να εισάγει καθυστέρηση)
- απλό: δεν διατηρείται κατάσταση σύνδεσης στον πομπό, δέκτη
- μικρή επικεφαλίδα στο τεμάχιο
- όχι έλεγχος συμφόρησης: το UDP μπορεί να στέλνει όσο γρήγορα μπορεί

UDP



Το UDP, όπως και το TCP, χρησιμοποιεί αριθμούς θυρών για να αποπολυπλέξει τα πακέτα

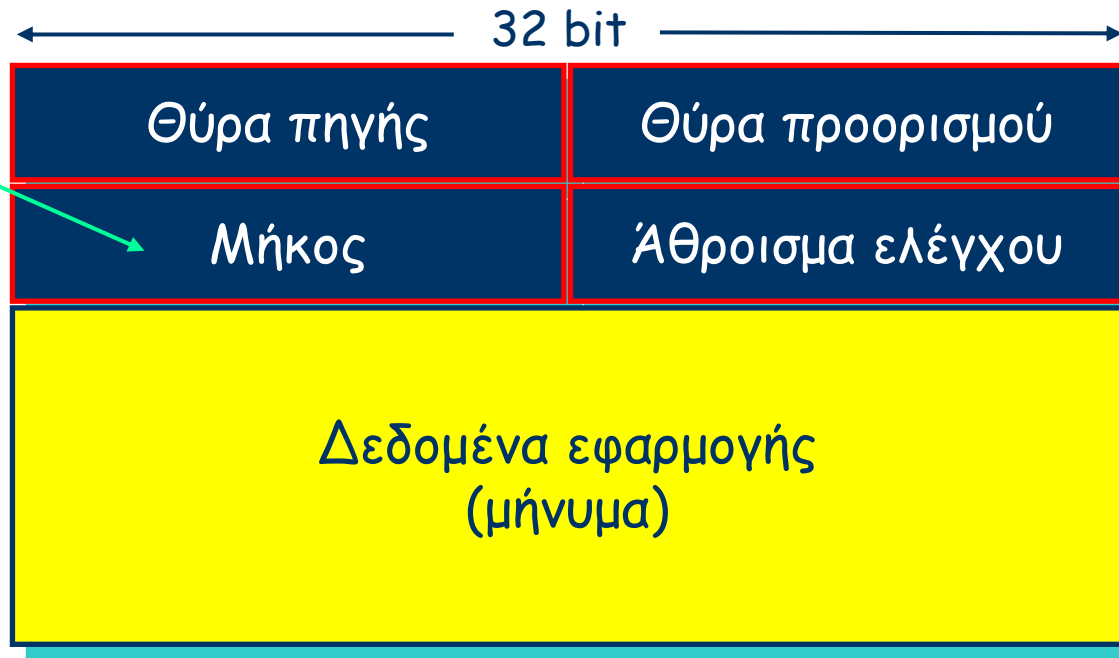


- μηνύματα μέχρι και 64KB
- παρέχει πολυπλεξία/αποπολυπλεξία στο IP
- πλεονεκτεί έναντι του TCP στο ότι δεν αυξάνει την καθυστέρηση απ' άκρη σ' άκρη πάνω από το IP
- χρησιμοποιείται σε εφαρμογές streaming multimedia
 - ανοχή σε απώλειες
 - ευαισθησία στον ρυθμό μετάδοσης
- άλλες χρήσεις του UDP
 - DNS
 - SNMP
 - RIP
- για αξιόπιστη μεταφορά με UDP, χρειάζεται προσθήκη αξιοπιστίας στο στρώμα εφαρμογής
 - αποκατάσταση λαθών ειδική για την εφαρμογή

Μορφή τεμαχίου UDP



Μήκος τεμαχίου
UDP σε byte,
μαζί με την
επικεφαλίδα



Άθροισμα ελέγχου UDP



Στόχος: ανίχνευση σφαλμάτων στο μεταδιδόμενο τεμάχιο

- Συμπεριλαμβάνει την ψευδοεπικεφαλίδα
- **Πομπός:**
 - μεταχειρίζεται τα περιεχόμενα του τεμαχίου ως ακολουθία ακεραίων των 16-bit
 - checksum: πρόσθεση των περιεχομένων του τεμαχίου και λήψη του συμπληρώματος ως προς 1 του αθροίσματος
 - ο πομπός τοποθετεί την τιμή του checksum στο πεδίο checksum του UDP
 - Γιατί το UDP προβλέπει checksum;
- **Δέκτης:**
 - υπολογίζει το checksum του λαμβανόμενου τεμαχίου, συμπεριλαμβανομένου και του πεδίου checksum
 - ελέγχει αν το υπολογισθέν checksum ισούται με την τιμή 1111111111111111 :
 - ΟΧΙ - ανιχνεύθηκε σφάλμα
 - ΝΑΙ - δεν ανιχνεύθηκε σφάλμα. Αλλά μπορεί να υπάρχουν σφάλματα....

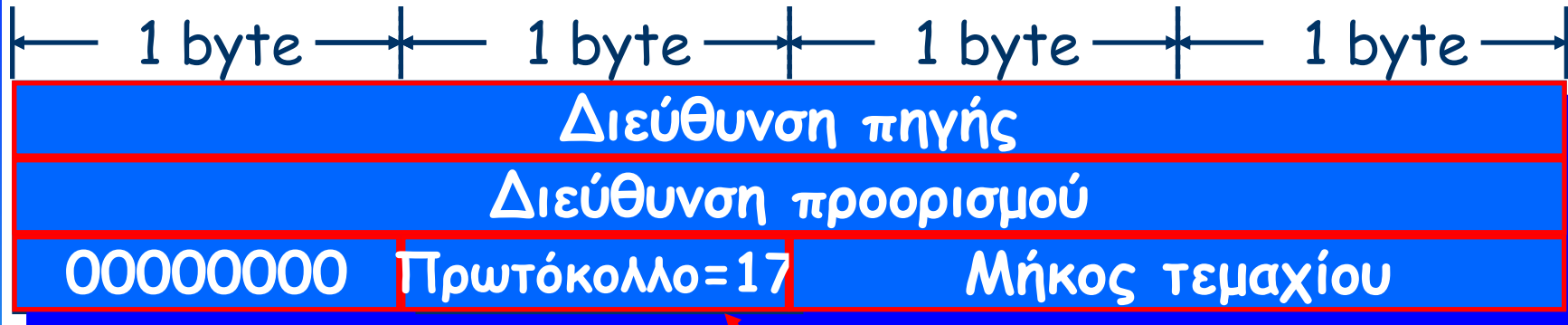
Παράδειγμα αθροίσματος ελέγχου



- Σημείωση
 - Κατά την πρόσθεση αριθμών, το κρατούμενο από το πιο σημαντικό bit πρέπει να προστίθεται στο αποτέλεσμα
- Παράδειγμα: πρόσθεση δύο ακεραίων 16-bit

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0	
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	

Ψευδοεπικεφαλίδα



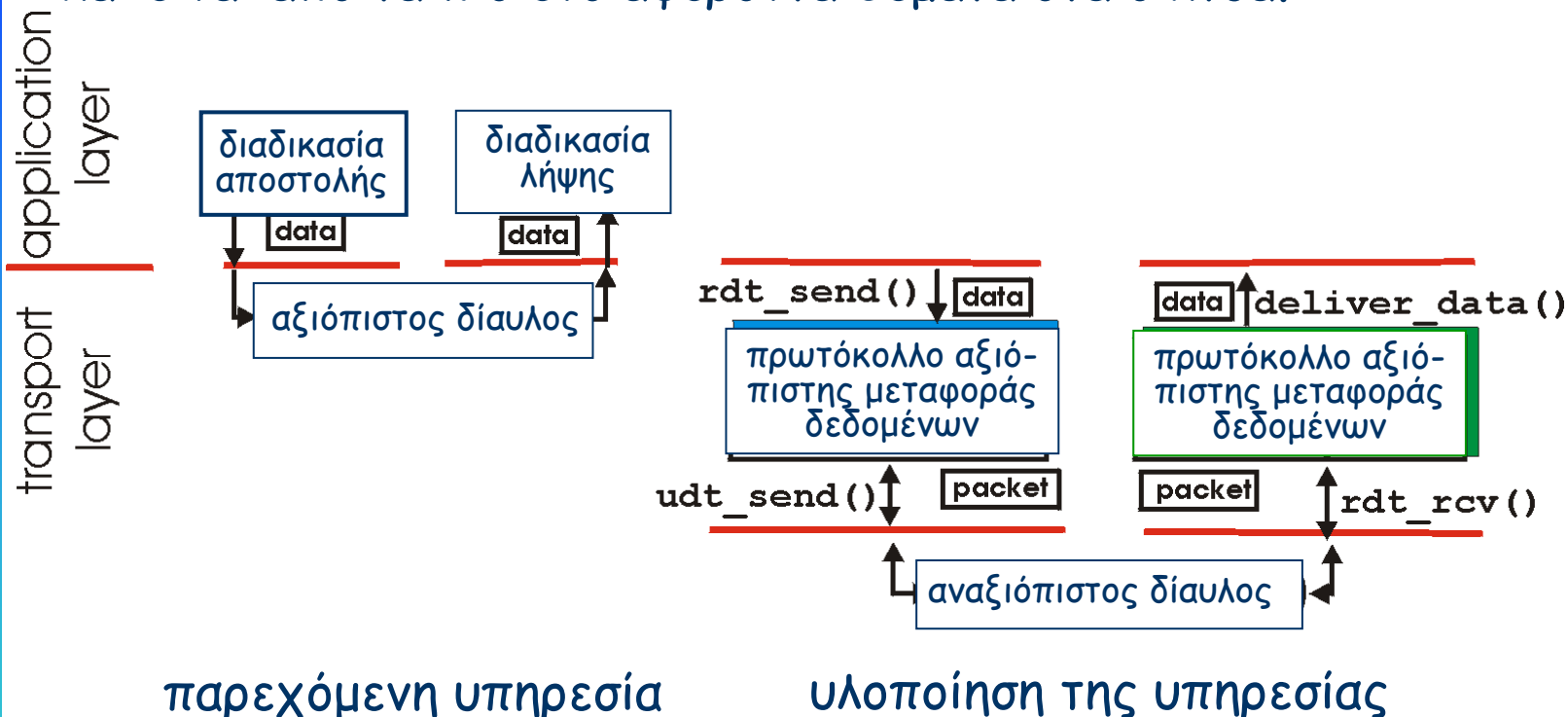
6, για το TCP

- Η ψευδοεπικεφαλίδα εξασφαλίζει ότι το datagram έχει πραγματικά οδηγηθεί στο σωστό προορισμό, host και θύρα
- Χρησιμοποιείται μόνο για τον υπολογισμό του Checksum και δεν μεταδίδεται

Αξιόπιστη μετάδοση δεδομένων



- ενδιαφέρει στα στρώματα εφαρμογής, μεταφοράς, ζεύξης δεδομένων και είναι από τα πιο ενδιαφέροντα θέματα στα δίκτυα.



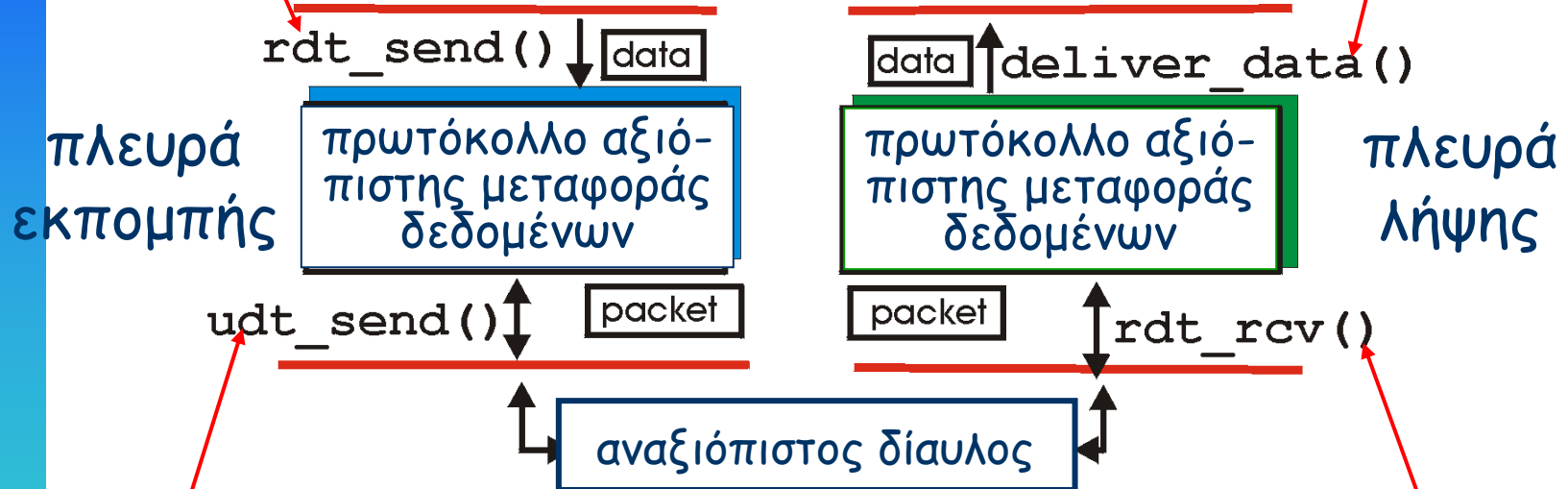
- τα χαρακτηριστικά το αναξιόπιστου διαύλου καθορίζουν την πολυπλοκότητα του πρωτοκόλλου αξιόπιστης μετάδοσης δεδομένων (rdt)

Αξιόπιστη μετάδοση δεδομένων



rdt_send(): καλείται από άνω (π.χ., από app.) για να παραδοθούν τα δεδομένα που προορίζονται για το ανώτερο στρώμα του δέκτη

deliver_data(): καλείται από την rdt για την παράδοση δεδομένων προς τα άνω



udt_send(): καλείται από την rdt, για τη μεταφορά πακέτου πάνω από τον αναξιόπιστο δίαυλο προς τον δέκτη

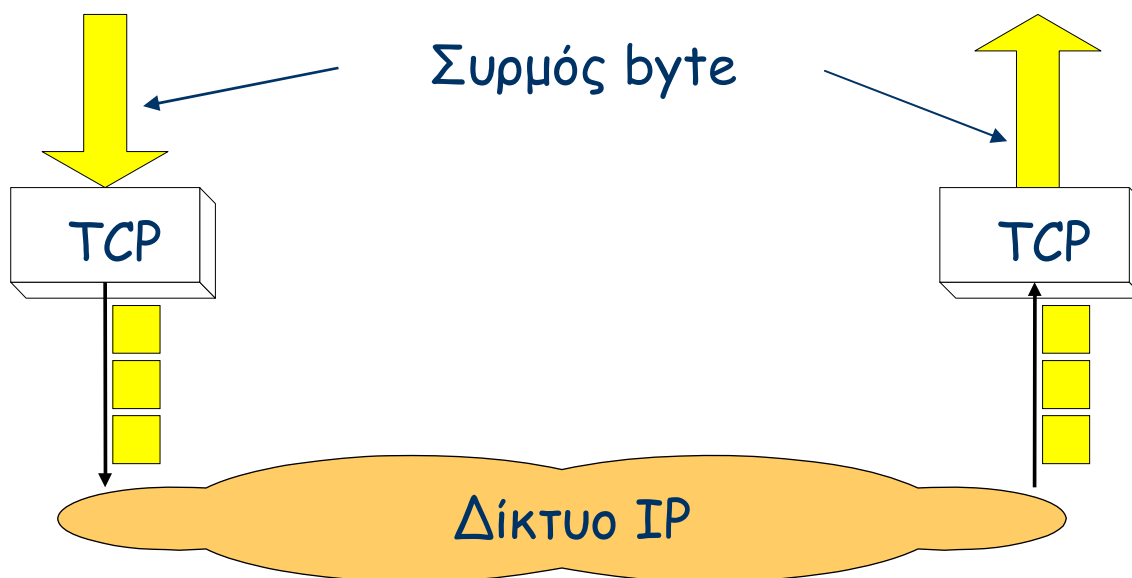
rdt_rcv(): καλείται όταν το πακέτο φθάνει στην πλευρά λήψης του διαύλου

TCP: Εισαγωγή



TCP = Transmission Control Protocol

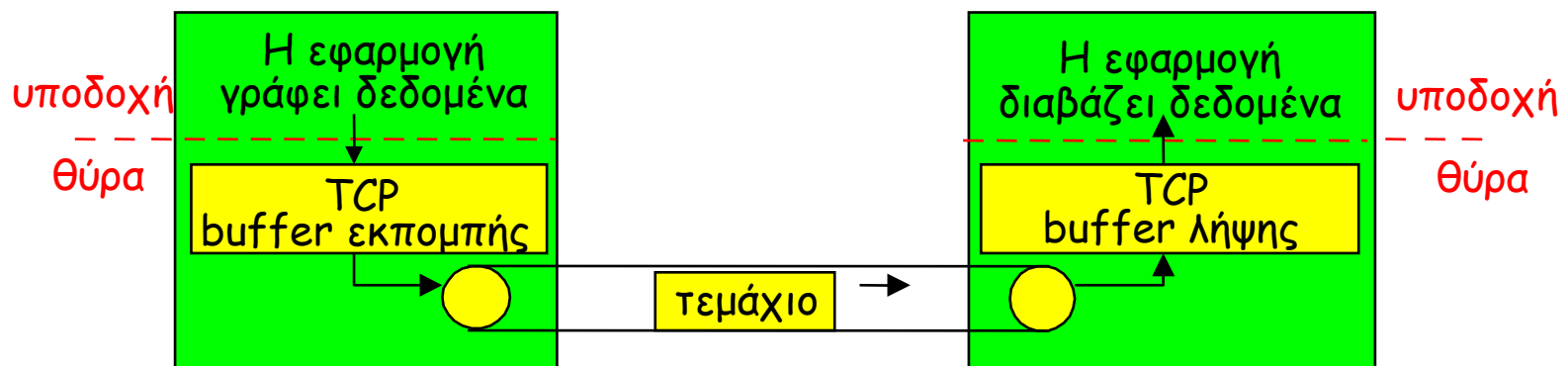
- Είναι πρωτόκολλο με **σύνδεση**
- Παρέχει **αξιόπιστη** μετάδοση **συρμού byte** απ' άκρη σ' άκρη πάνω από μη αξιόπιστο δίκτυο.



TCP: Εισαγωγή



- μετάδοση σημείου προς σημείο:
 - ένας πομπός ένας δέκτης
- αξιόπιστη μετάδοση συρμού byte που παραδίνονται με τη σειρά
 - το μήνυμα μπορεί να έχει αυθαίρετο μήκος
- συνεχής παροχή
 - ο έλεγχος συμφόρησης και ροής στο TCP καθορίζουν το μέγεθος του παραθύρου
- buffers εκπομπής και λήψης



TCP: Εισαγωγή

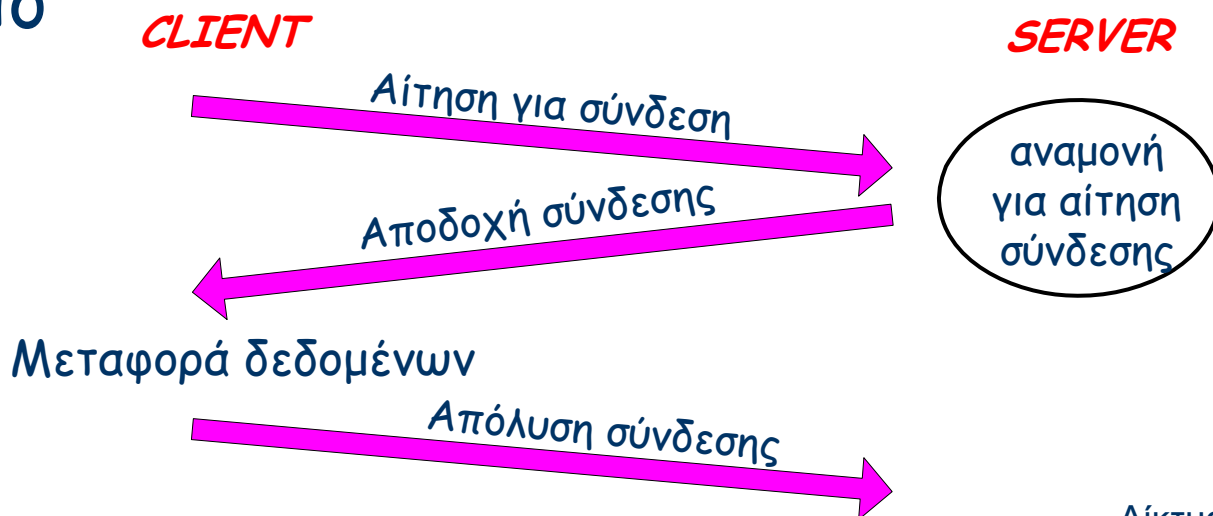


- αμφίδρομη μετάδοση
 - αμφίδρομη ροή δεδομένων στην ίδια σύνδεση
 - MSS: maximum segment size
- υπηρεσία με σύνδεση
 - τριμερής χειραψία αρχικοποιεί την κατάσταση πομπού και δέκτη πριν την ανταλλαγή δεδομένων
- έλεγχος ροής
 - ο πομπός δεν υπερχειλίζει τον δέκτη
- παρέχει πολυπλεξία/αποπολυπλεξία πάνω από το IP
- έλεγχος και αποφυγή συμφόρησης
- παραδείγματα χρησιμοποίησης: file transfer, chat, web, SMTP (e-mail)

Πρωτόκολλο με σύνδεση



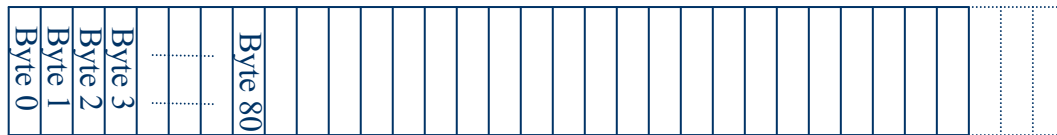
- Πριν από οποιανδήποτε μεταφορά δεδομένων, το TCP εγκαθιστά μια **σύνδεση**:
 - Η μια οντότητα TCP αναμένει για σύνδεση ("server")
 - Η άλλη οντότητα TCP ("client") συνδέεται με τον server
- Η διαδικασία εγκατάστασης σύνδεσης είναι στην πραγματικότητα πιο πολύπλοκη.
- Κάθε σύνδεση είναι αμφίδρομη, σημείου προς σημείο



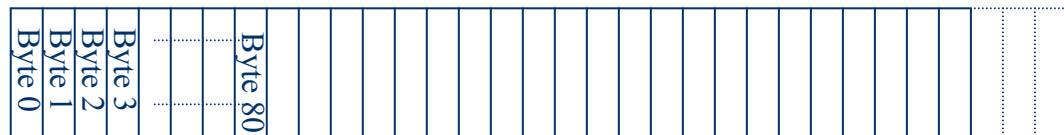
Υπηρεσία συρμού byte



Host A



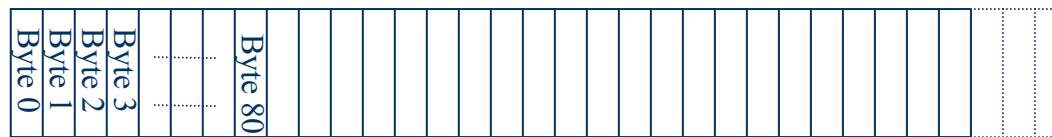
Host B



Υπηρεσία συρμού byte



Host A



TCP Data

Το τεμάχιο στέλνεται όταν:

1. Είναι πλήρες (MSS byte),
2. Όχι πλήρες, αλλά λήγει ο χρόνος
3. "Ωθείται" από την εφαρμογή.

TCP Data

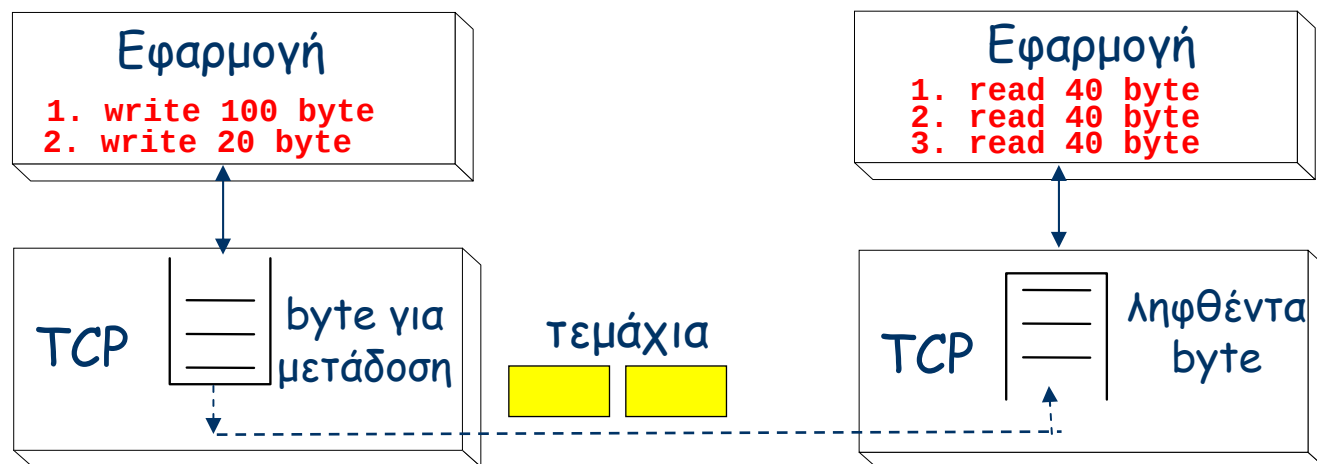
Host B



Υπηρεσία συρμού byte



- Στα κατώτερα στρώματα, το TCP παραδίδει δεδομένα σε τμήματα, τα **τεμάχια**.
- Στα ανώτερα στρώματα, το TCP παραδίδει δεδομένα ως ακολουθία από byte και δεν καθορίζει όρια μεταξύ των byte
- **Συνεπώς**, τα ανώτερα στρώματα δεν γνωρίζουν την αρχή και το τέλος των τεμαχίων!

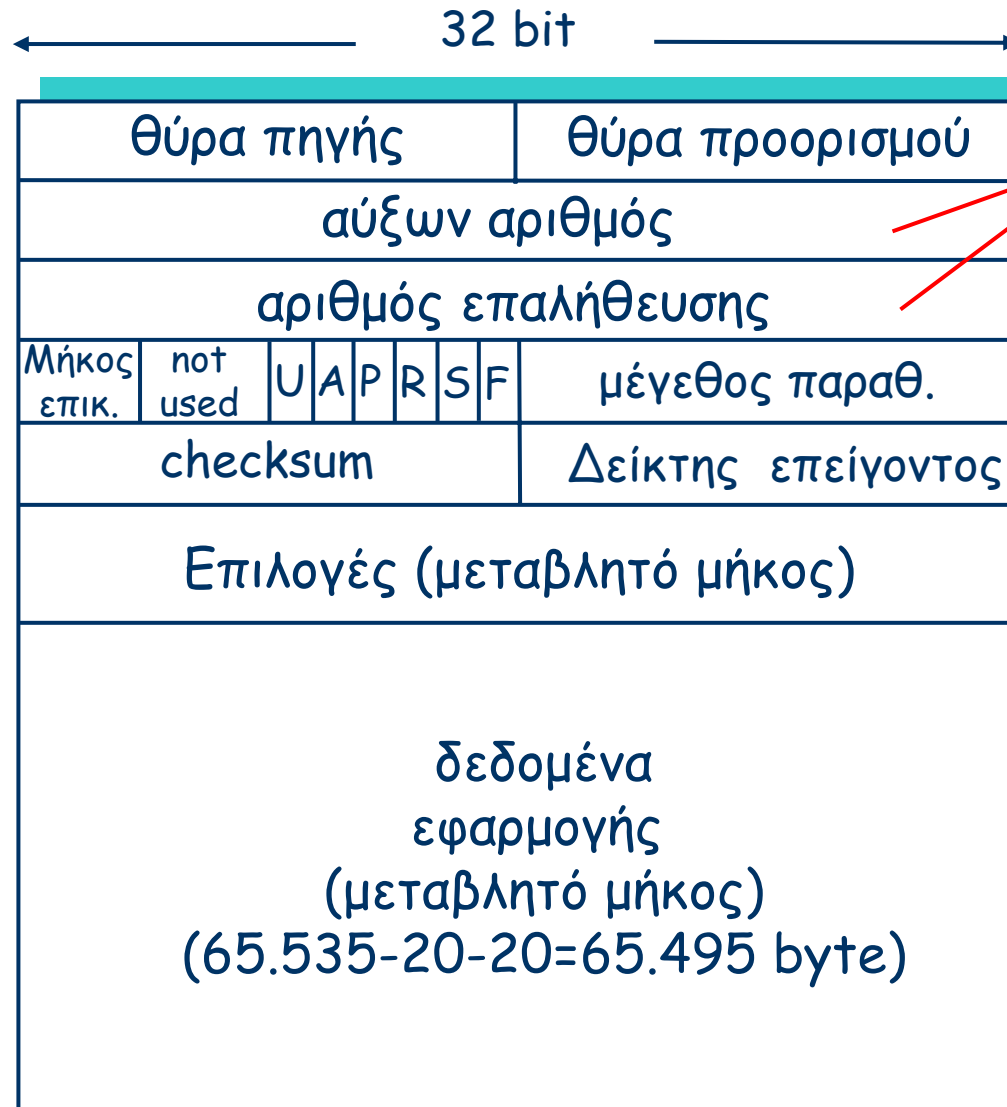


MSS: Maximum segment size



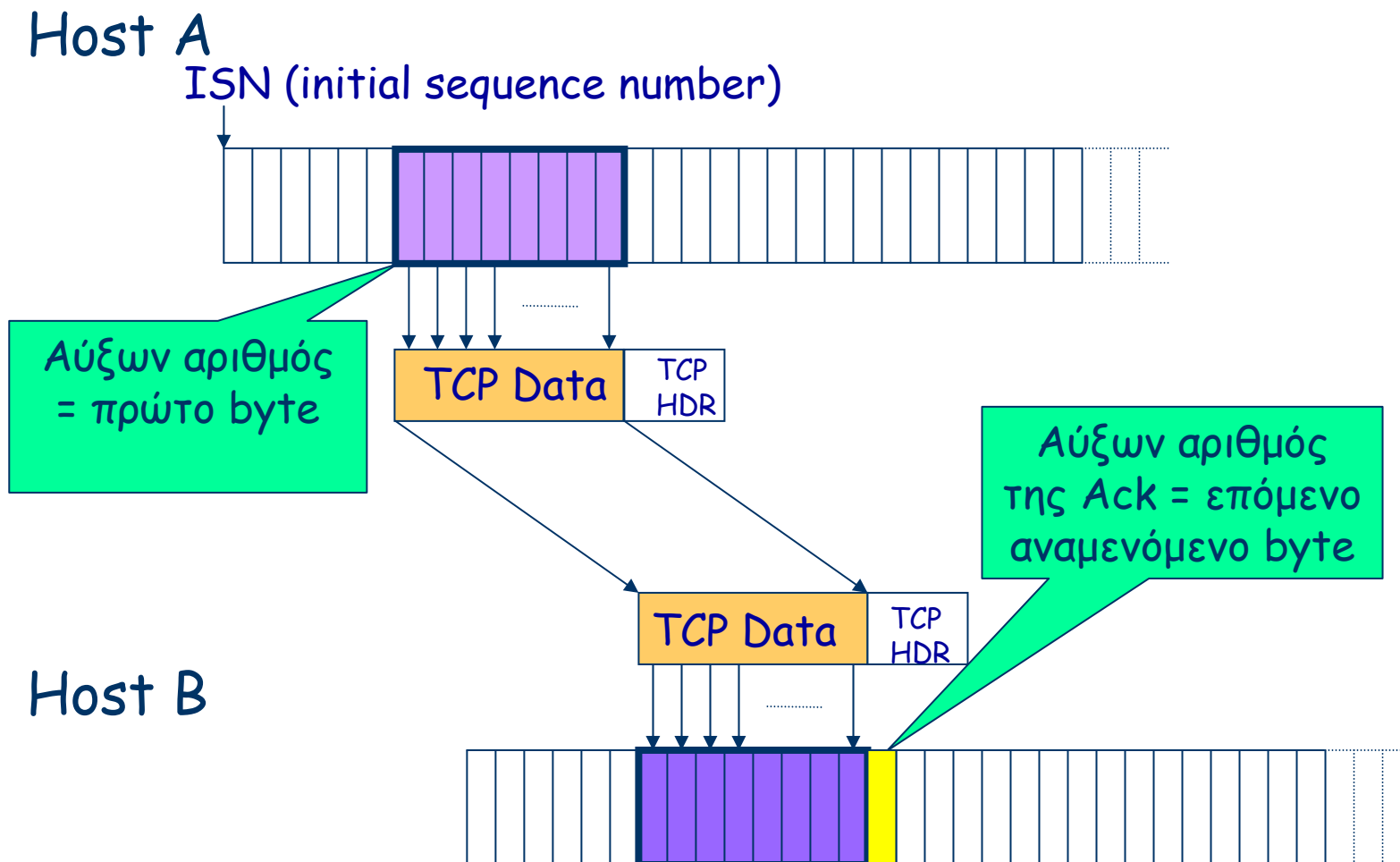
- Εξαρτάται από την υλοποίηση του TCP (καθοριζόμενη από το OS) και μπορεί να επιλεγεί
- Συνήθεις τιμές: 1500, 536, 512 byte
- Επιλέγεται το μέγιστο μέγεθος έτσι ώστε να αποφευχθεί ο θρυμματισμός στο IP
- MSS είναι ο **μέγιστος αριθμός δεδομένων** του στρώματος εφαρμογής που περιέχονται στο τεμάχιο και **όχι το μέγιστο μέγεθος του τεμαχίου**

Δομή τεμαχίου TCP



μέτρηση
σε byte
δεδομένων

Αύξοντες αριθμοί και επαληθεύσεις





Αριθμός επαλήθευσης

- Το TCP χρησιμοποιεί παραλλαγή του **πρωτοκόλλου ολισθαίνοντος παραθύρου** για τον έλεγχο ροής μεταξύ πομπού και δέκτη
 - όχι NACK (Negative ACKnowledgement)
 - μόνο συσσωρευτικές ACK
- **Παράδειγμα:**

Ο πομπός στείλει δύο τεμάχια με "0...1500" και "1501...3000", αλλά ο δέκτης λαμβάνει μόνο το δεύτερο.

Ο δέκτης δεν μπορεί να επαληθεύσει το δεύτερο τεμάχιο. Μπορεί να στείλει μόνο AckNo = 1

Αύξοντες αριθμοί: παράδειγμα Telnet



- **Αύξοντες αριθμοί:**

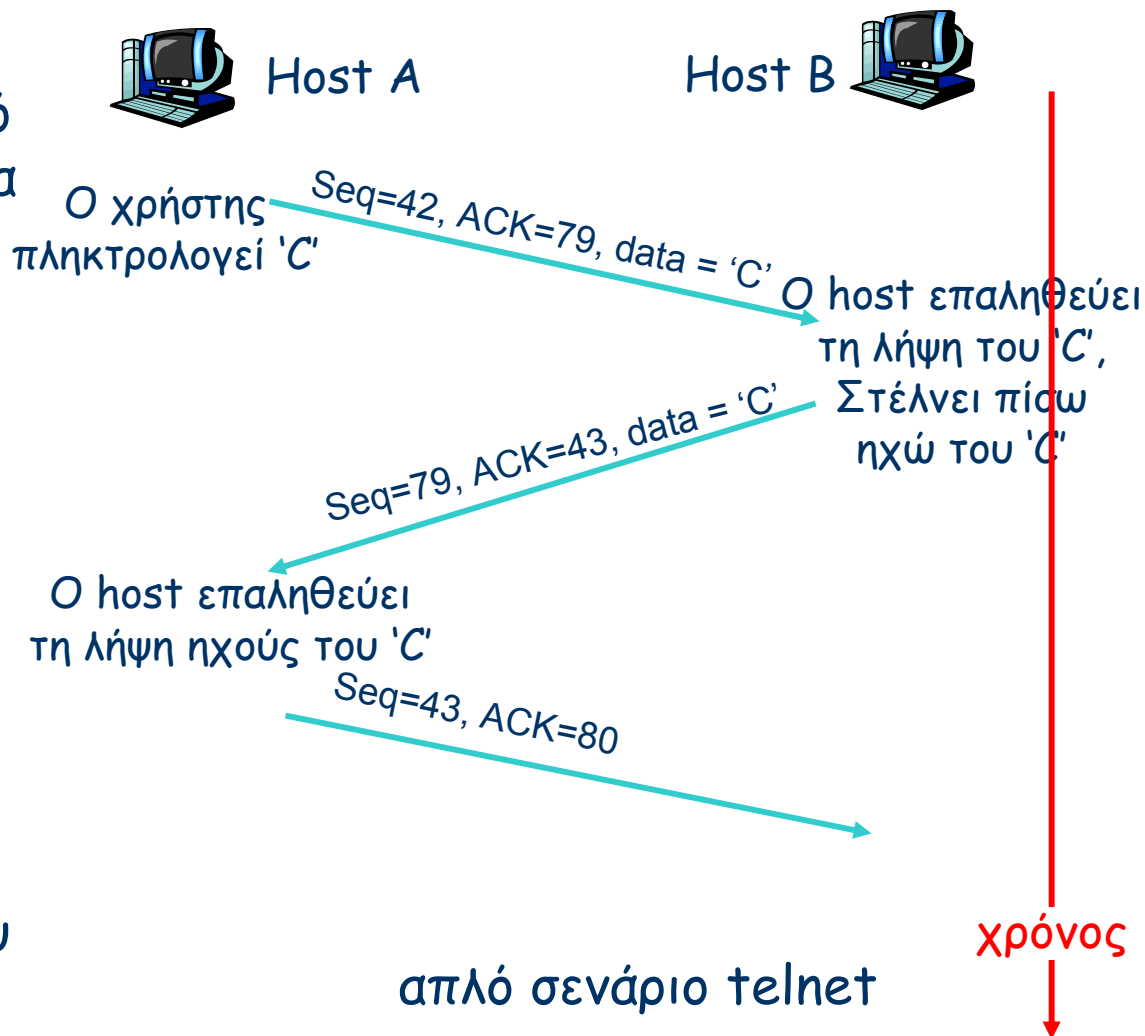
- σε συρμό byte δηλώνουν τον αριθμό του πρώτου byte στα δεδομένα του τεμαχίου

- **Επαληθεύσεις:**

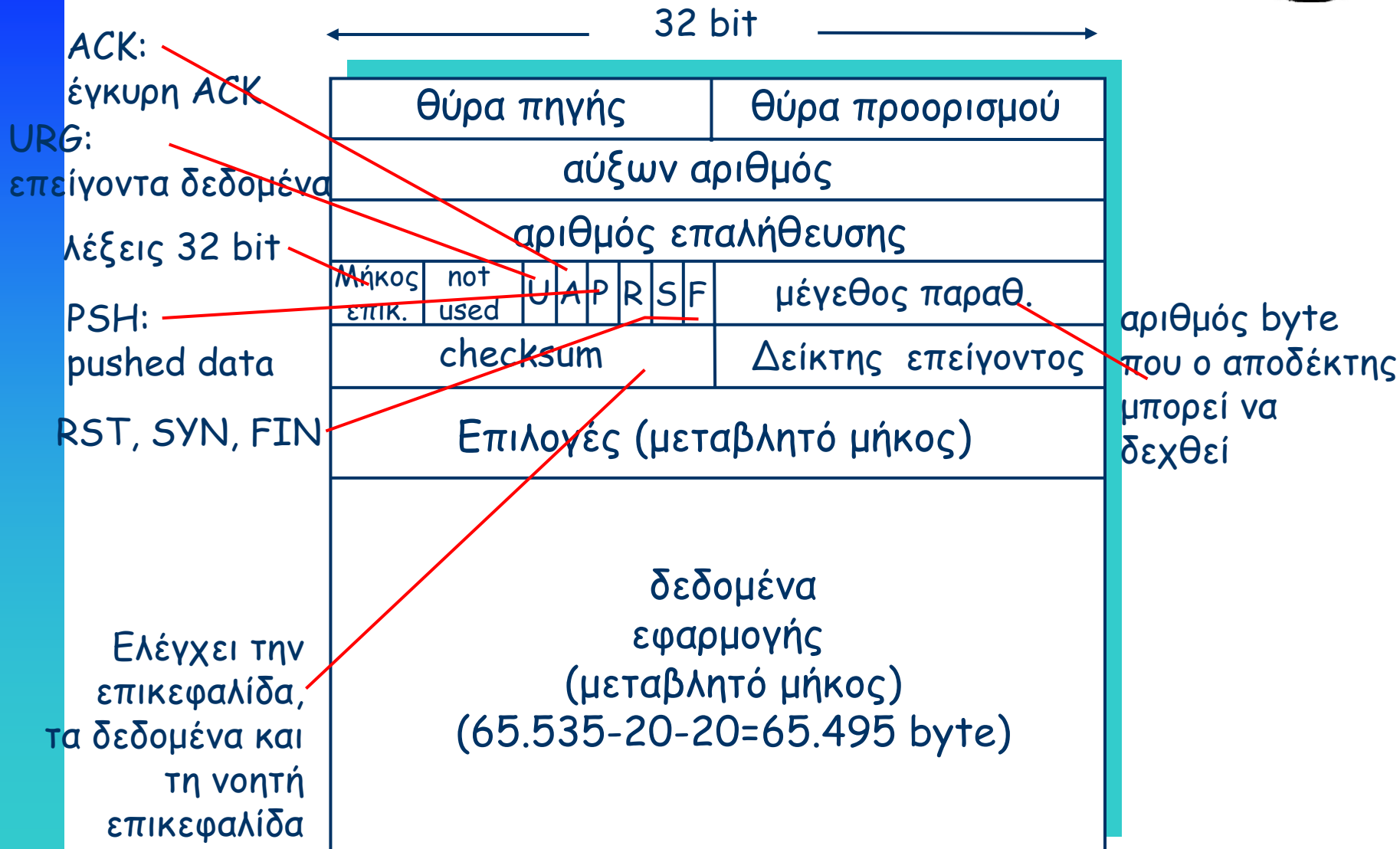
- αύξων αριθμός του επόμενου byte που περιμένει η άλλη πλευρά

- **Πώς αντιμετωπίζει ο δέκτης τα εκτός σειράς τεμάχια:**

- Οι προδιαγραφές του TCP δεν αναφέρουν τίποτε. Εξαρτάται από τον κατασκευαστή



Δομή τεμαχίου TCP



Πεδία της επικεφαλίδας TCP



- Οι προαιρετικές επιλογές (options) είναι ένας τρόπος να προστεθούν επιπλέον δυνατότητες που δεν καλύπτονται από την κανονική επικεφαλίδα

End of Options	kind=0	1 byte			
NOP (no operation)	kind=1	1 byte			
Maximum Segment Size	kind=2	len=4	maximum segment size		
	1 byte	1 byte	2 bytes		
Window Scale Factor	kind=3	len=3	shift count		
	1 byte	1 byte	1 byte		
Timestamp	kind=8	len=10	timestamp value	timestamp echo reply	
	1 byte	1 byte	4 bytes	4 bytes	

Διαχείριση συνδέσεων στο TCP



- Εγκατάσταση σύνδεσης
- Απόλυση σύνδεσης
- Ειδικά σενάρια
- Διαγράμματα καταστάσεων

Εγκατάσταση σύνδεσης στο TCP



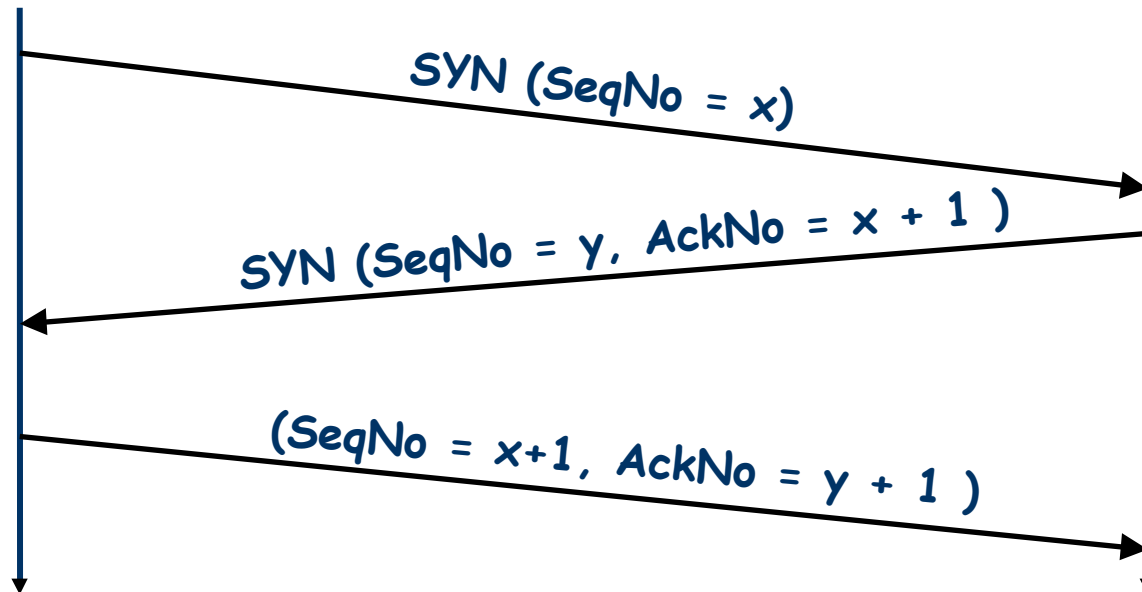
- Το TCP χρησιμοποιεί **τριμερή χειραψία** για να εγκαταστήσει μια σύνδεση:
- (1) **ACTIVE OPEN**: Ο client στέλνει τεμάχιο με
 - το SYN bit ενεργοποιημένο
 - τον αριθμό θύρας του
 - τον αρχικό αύξοντα αριθμό (ISN) του
- (2) **PASSIVE OPEN**: Ο server απαντάει με τεμάχιο με
 - το SYN bit ενεργοποιημένο
 - τον αρχικό αύξοντα αριθμό του
 - ACK για το ISN του client
- (3) Ο client επαληθεύει στέλνοντας τεμάχιο με:
 - ACK ISN του server. Το τεμάχιο μπορεί να περιέχει και δεδομένα

Τριμερής χειραψία



(Active)
Client

(Passive)
Server

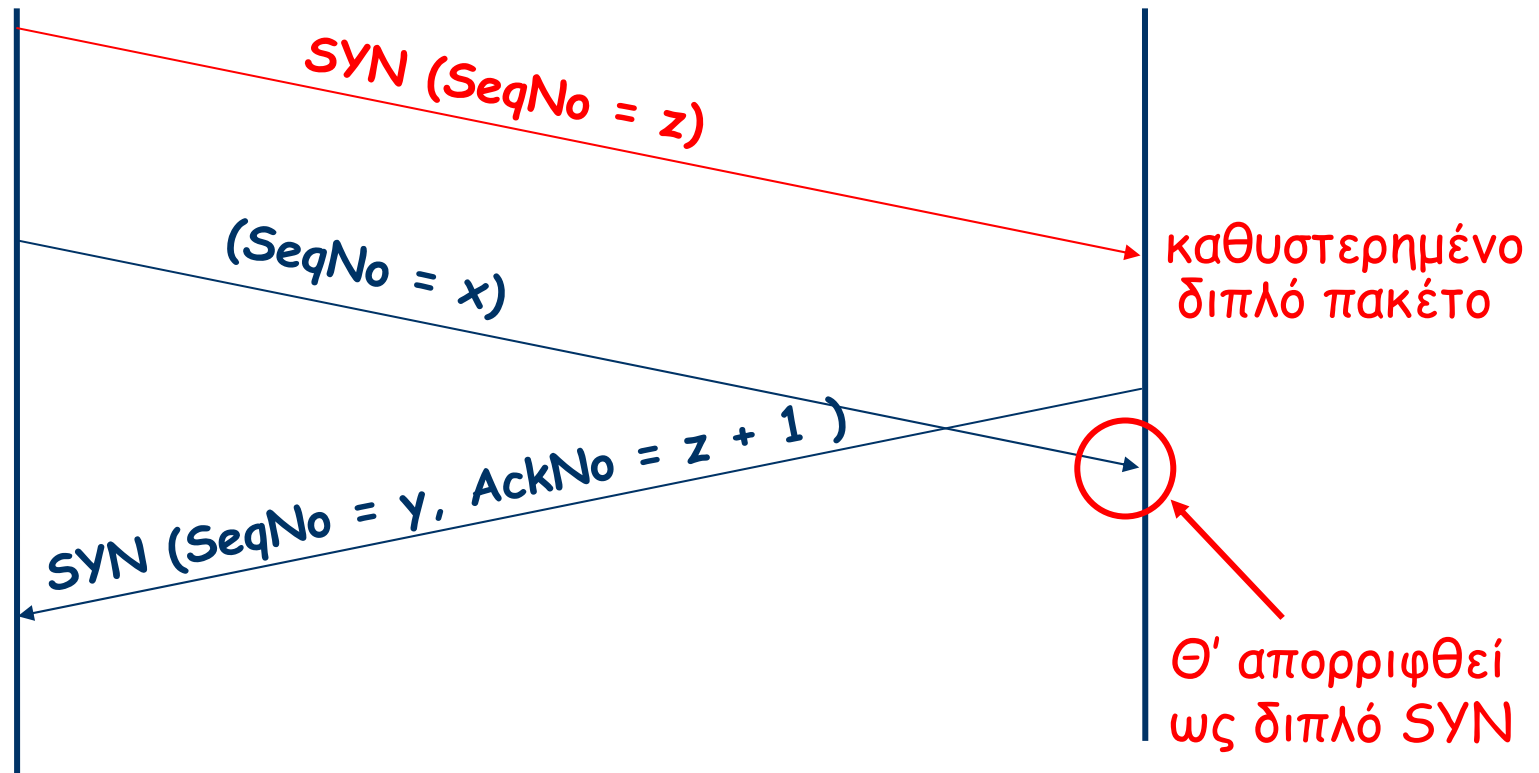


Γιατί δεν αρκεί η διμερής χειραψία



(Active)
Client

(Passive)
Server

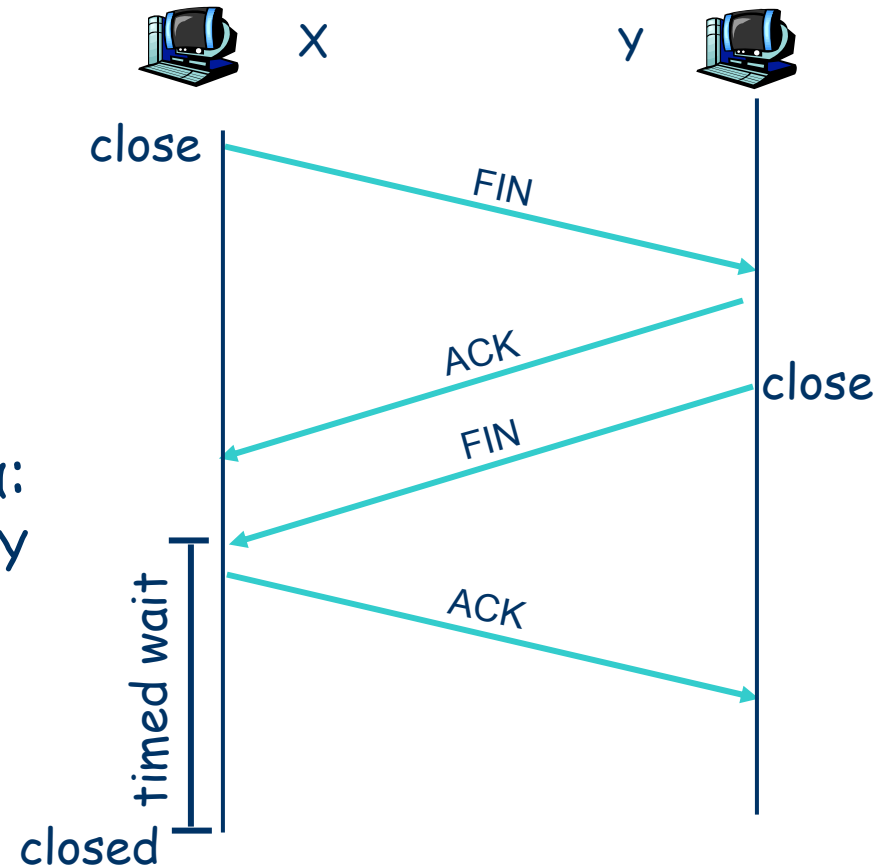


Όταν ο client αρχίζει τη μετάδοση δεδομένων (ξεκινώντας με $\text{SeqNo} = x+1$), ο server θα απορρίψει όλα τα δεδομένα.

Απόλυση σύνδεσης στο TCP



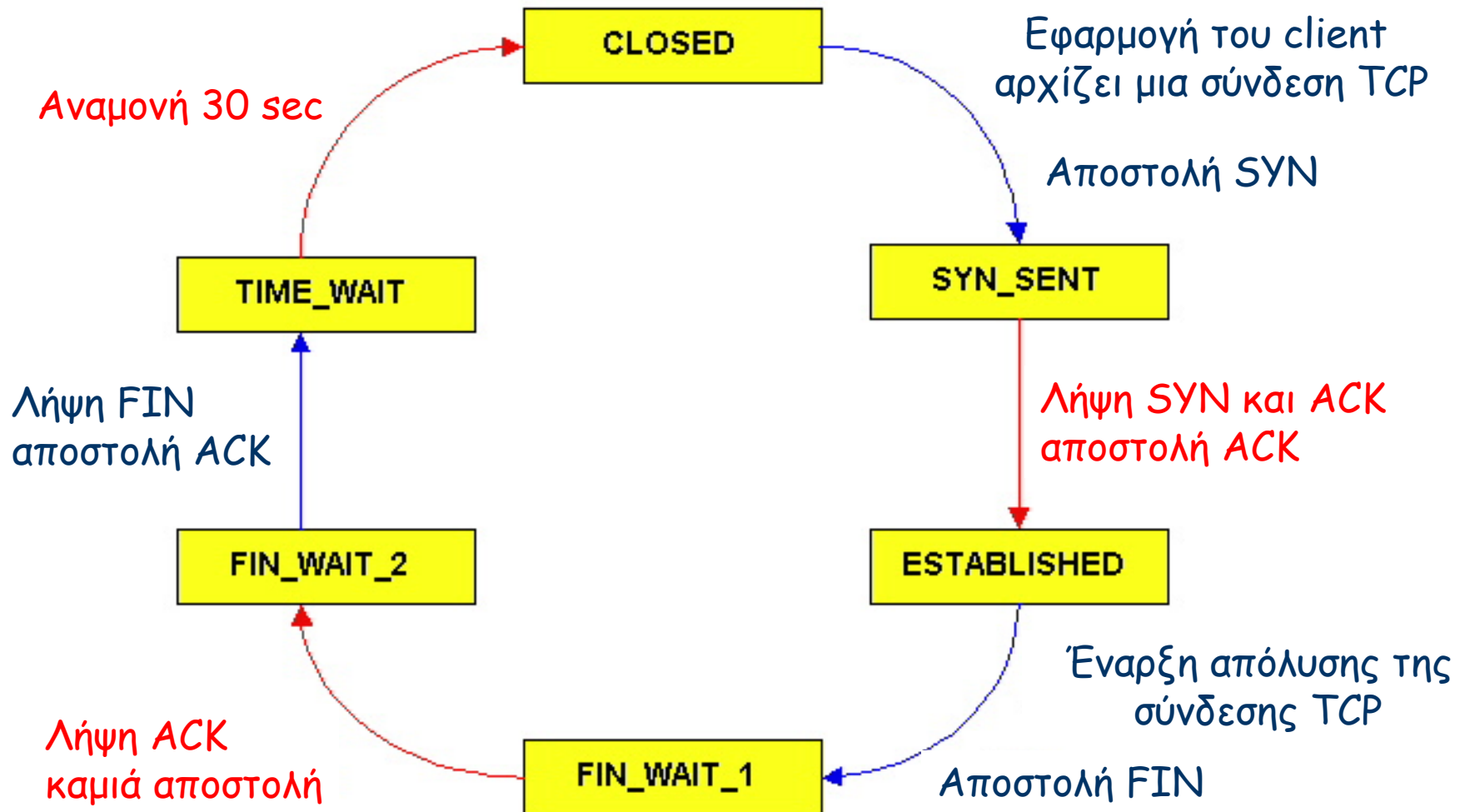
- Κάθε άκρο της ροής δεδομένων πρέπει να τερματίσει ανεξάρτητα ("half-close")
- Αν το ένα άκρο τερματίσει, στέλνει ένα τεμάχιο FIN. Τούτο σημαίνει ότι δεν θα σταλούν άλλα δεδομένα
- Απαιτούνται τέσσερα βήματα:
 - (1) Ο X στέλνει ένα FIN στον Y (**active close**)
 - (2) Ο Y επαληθεύει το FIN, (ταυτόχρονα: Ο Y μπορεί να στείλει ακόμα δεδομένα στον X)
 - (3) και ο Y στέλνει ένα FIN στον X (**passive close**)
 - (4) Ο X επαληθεύει το FIN.



Διάγραμμα μετάβασης καταστάσεων TCP



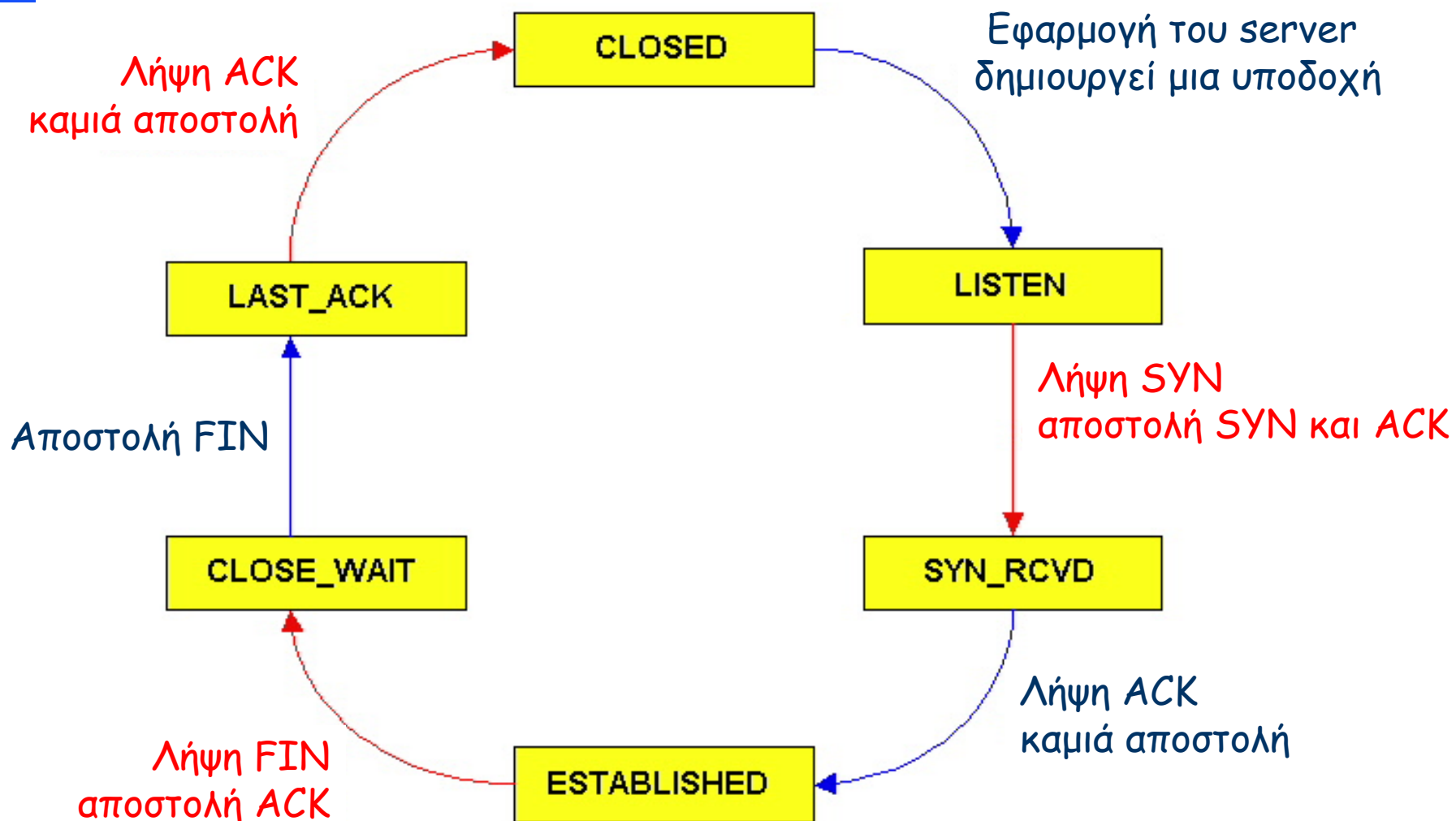
TCP client



Διάγραμμα μετάβασης καταστάσεων TCP



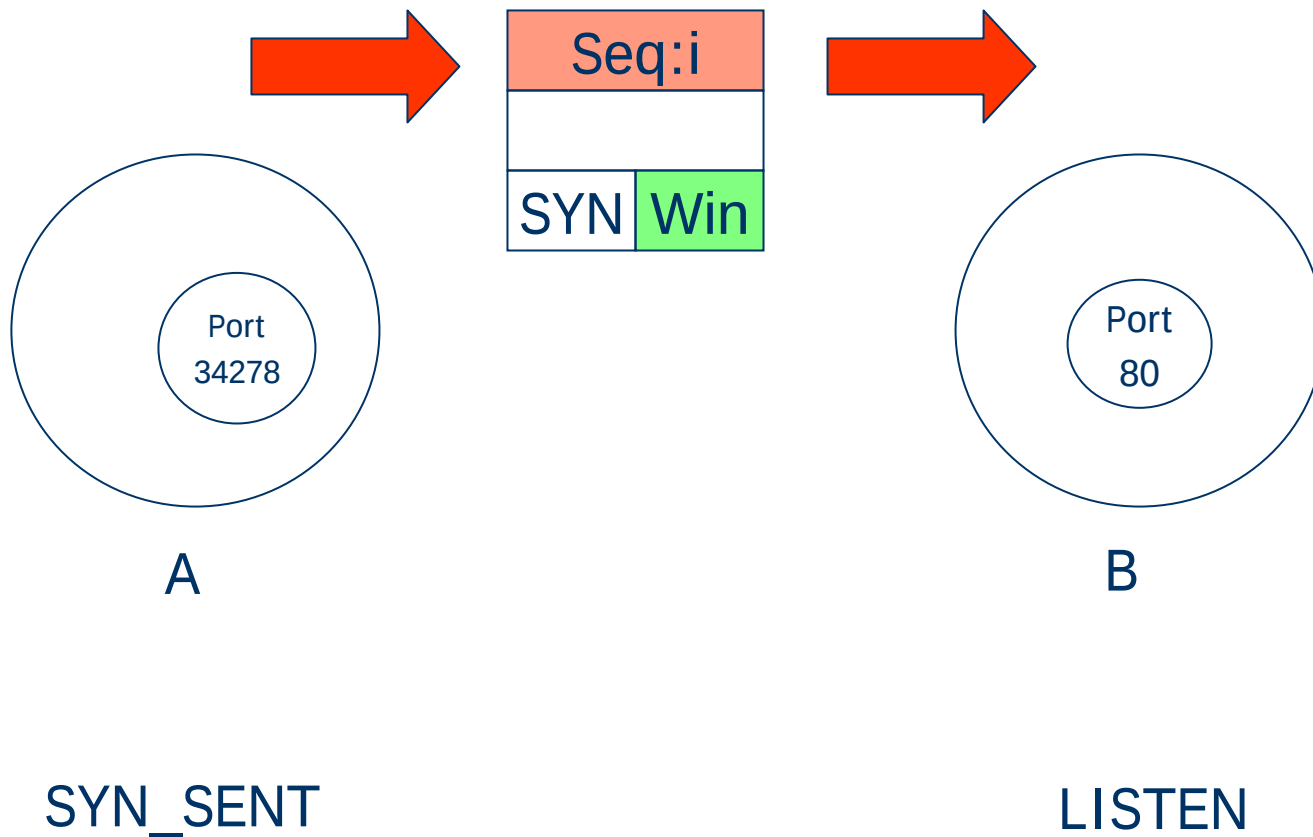
TCP server



Παράδειγμα σύνδεσης



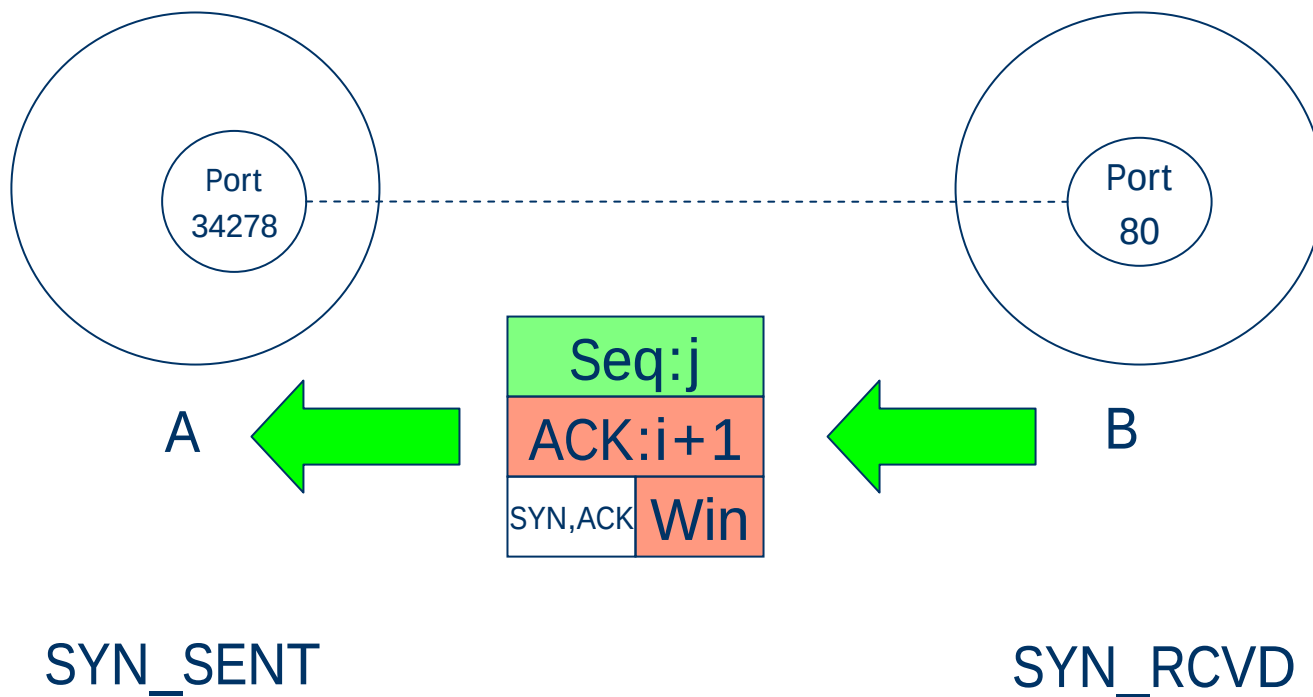
- Εγκατάσταση σύνδεσης...



Παράδειγμα σύνδεσης



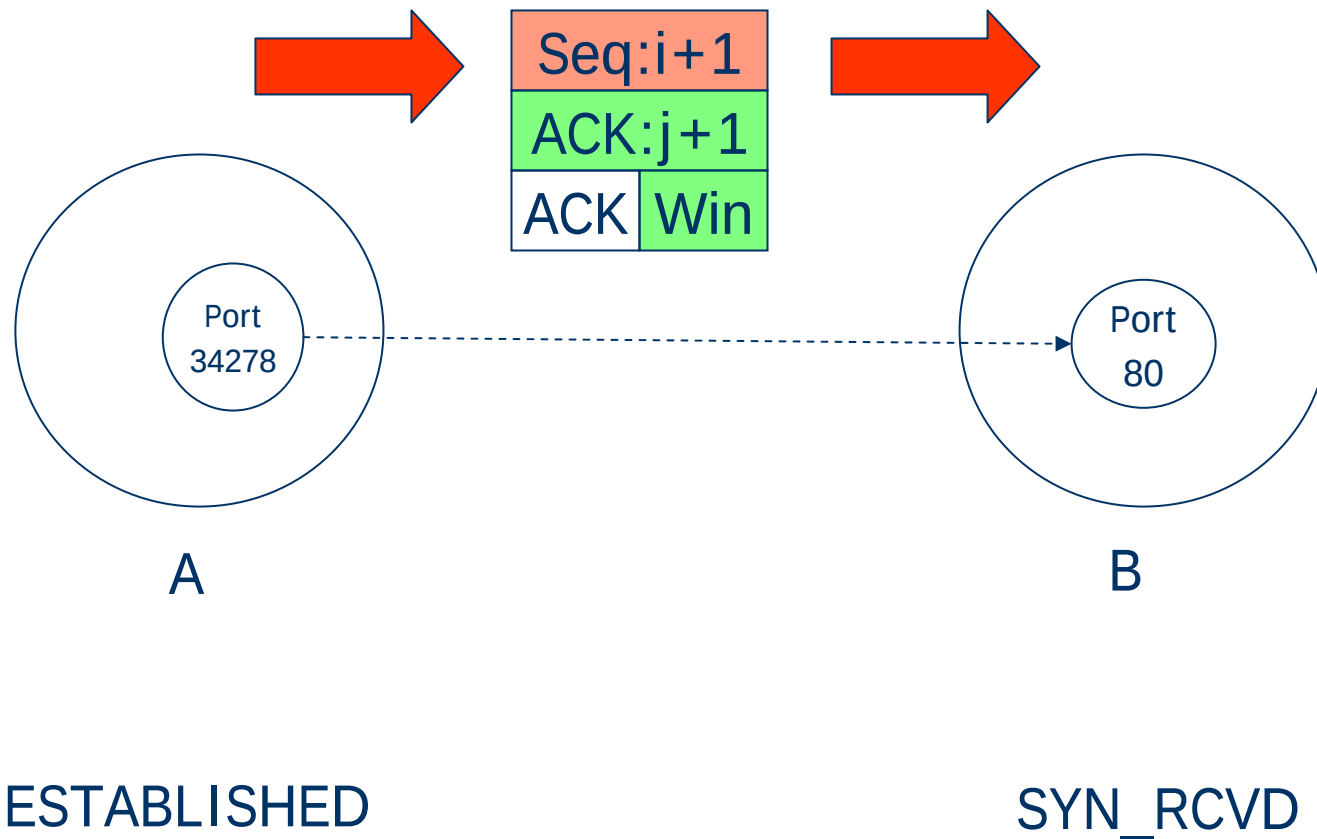
- Εγκατάσταση σύνδεσης...



Παράδειγμα σύνδεσης



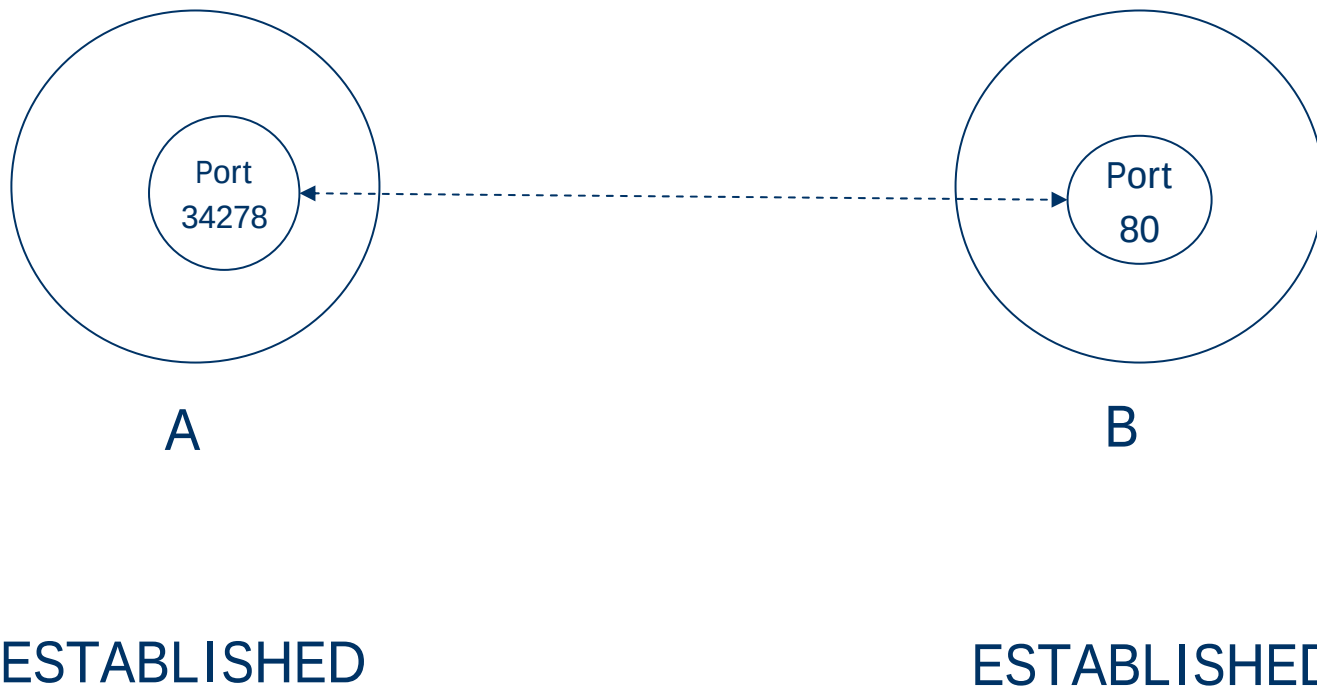
- Εγκατάσταση σύνδεσης...



Παράδειγμα σύνδεσης



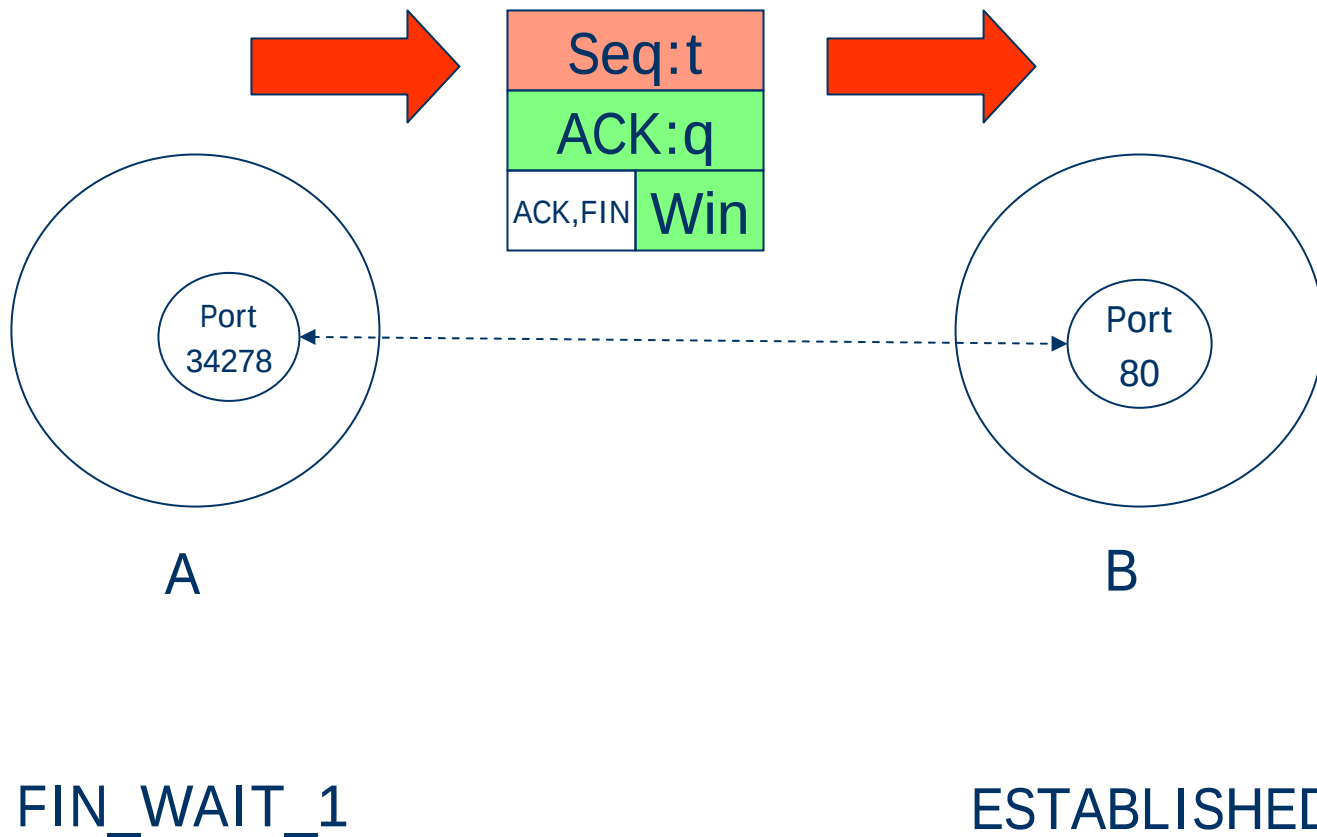
- Και οι δύο στην κατάσταση ESTABLISHED, ανταλλαγή δεδομένων...



Παράδειγμα σύνδεσης



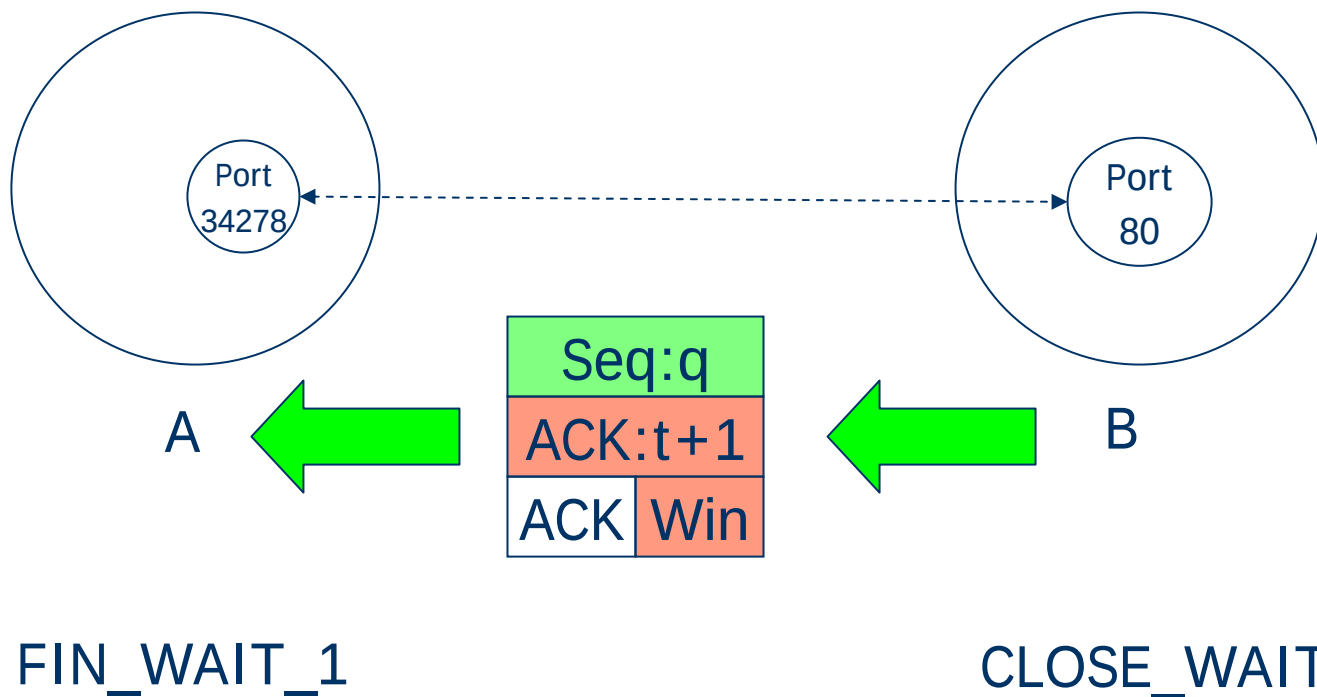
- Απόλυση σύνδεσης...



Παράδειγμα σύνδεσης



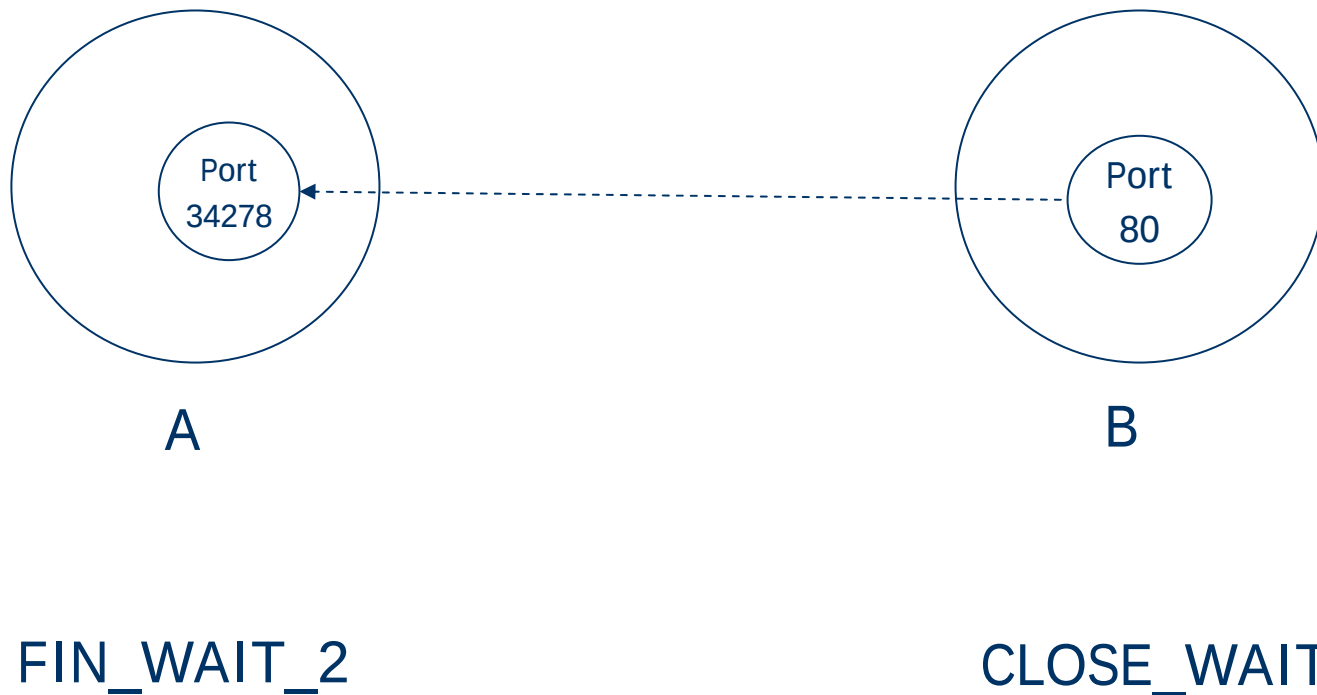
- Απόλυση σύνδεσης...



Παράδειγμα σύνδεσης



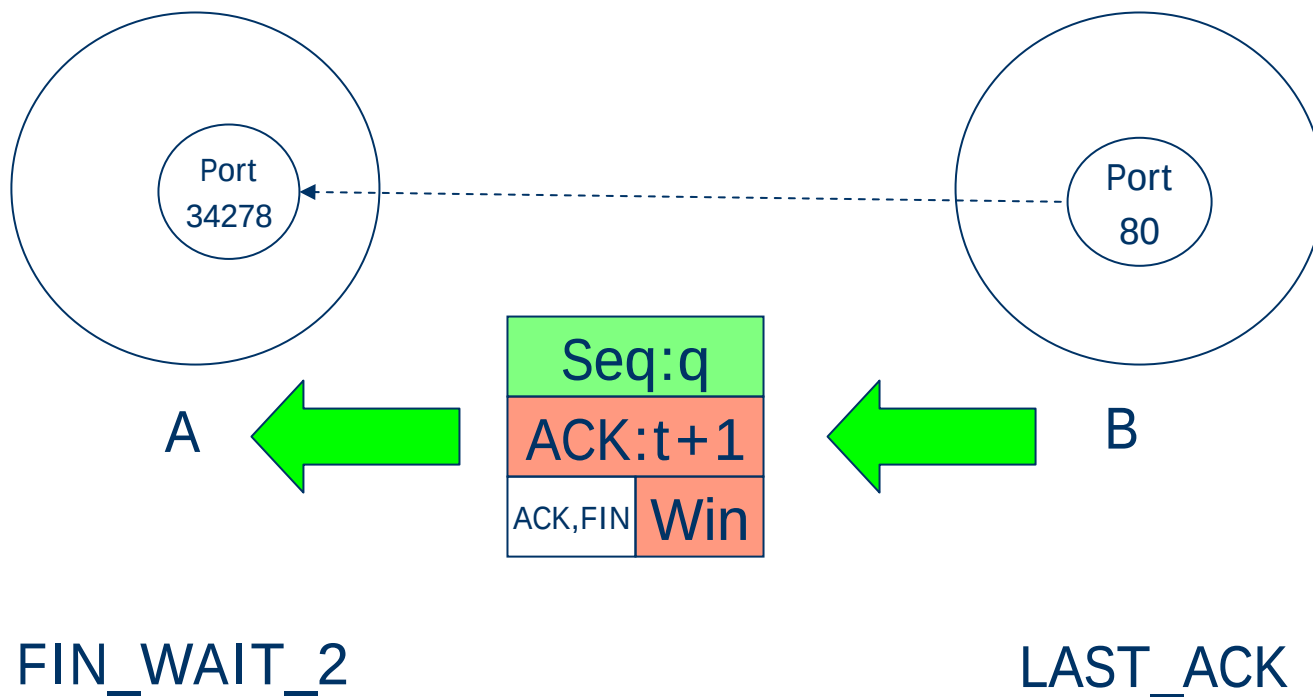
- Η εφαρμογή στον B πρέπει να κλείσει...



Παράδειγμα σύνδεσης



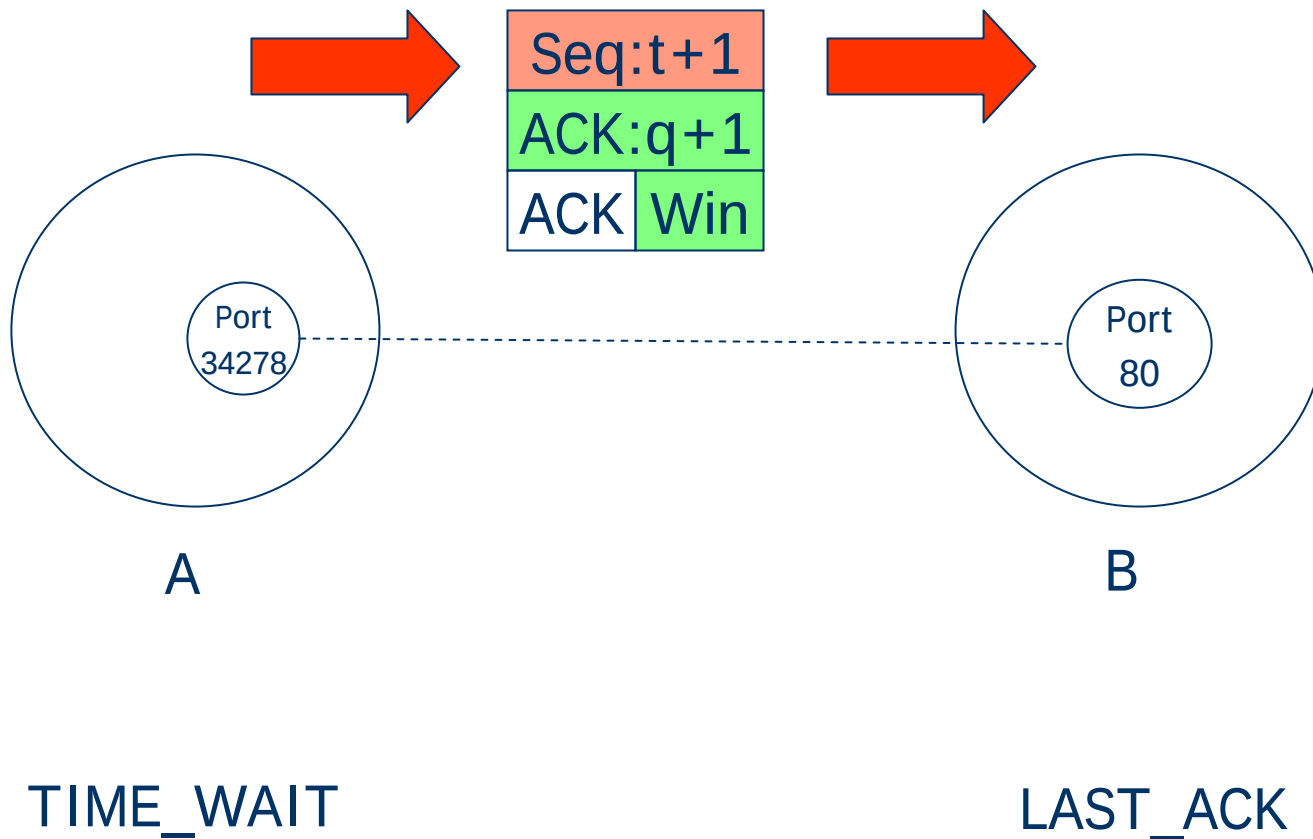
- Απόλυση σύνδεσης...



Παράδειγμα σύνδεσης



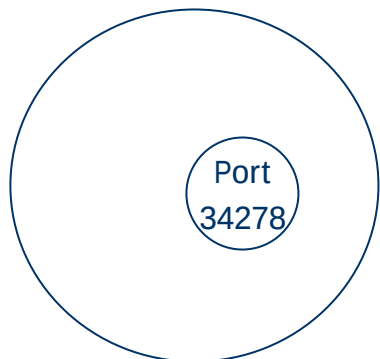
- Connection tear-down...



Παράδειγμα σύνδεσης

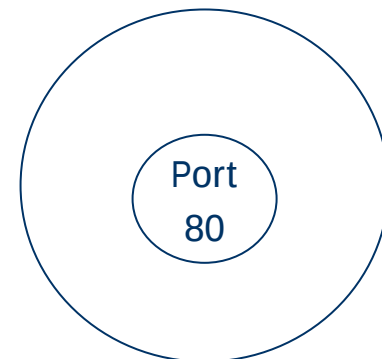


- Ο Client περιμένει $2 * MSL$ πριν μεταβεί στην κατάσταση **CLOSED** (ο B μπορεί να ξαναστείλει ένα FIN)



A

TIME_WAIT



B

CLOSED

Κατάσταση αναμονής 2MSL



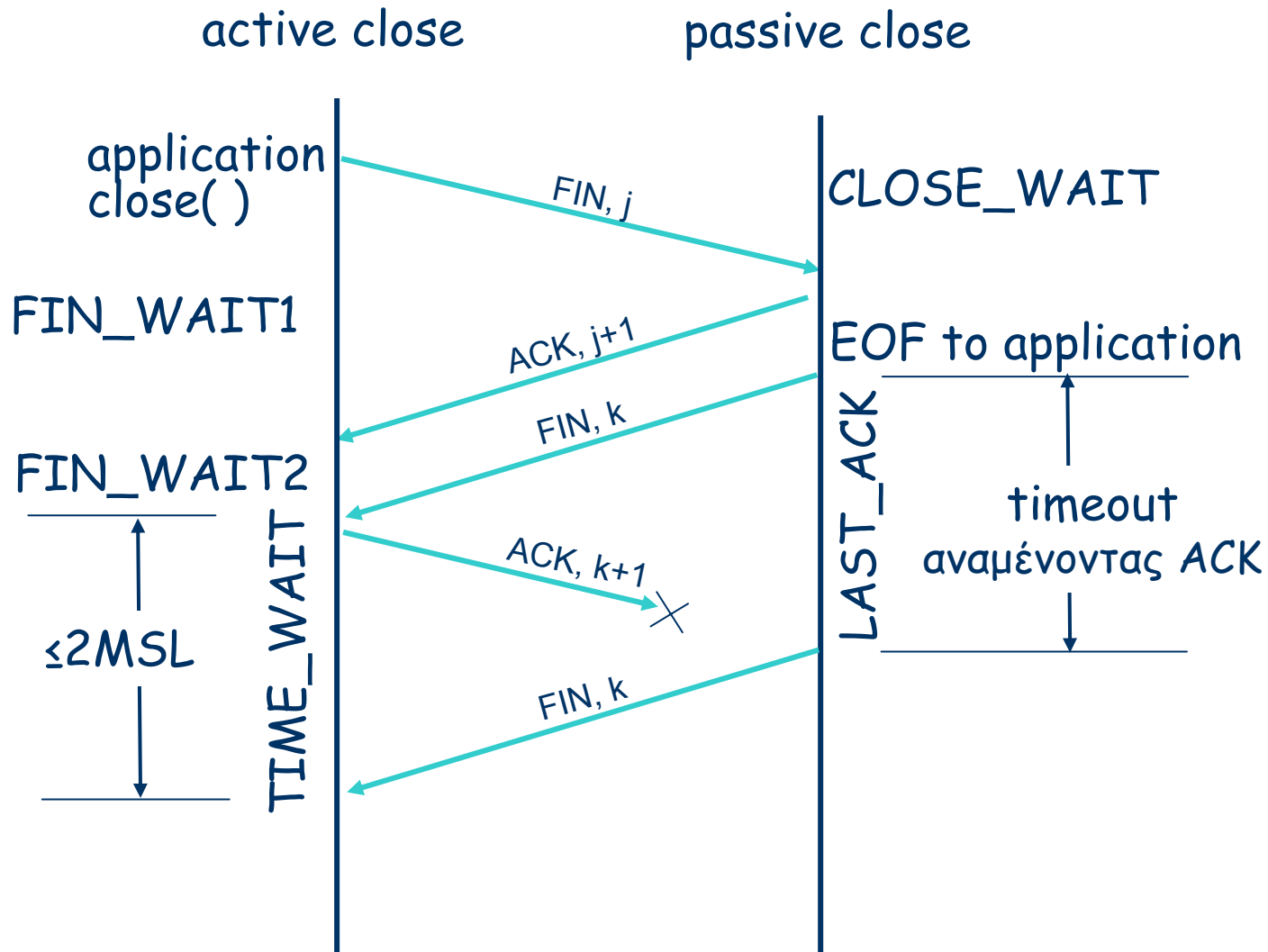
2MSL Wait State = TIME_WAIT

- Όταν το TCP κάνει active close, και στέλνει την τελική ACK, η σύνδεση πρέπει να παραμείνει στην κατάσταση **TIME_WAIT** για διπλάσιο χρόνο από τη μέγιστη ζωή τεμαχίων (maximum segment lifetime, MSL).

$$2MSL = 2 * \text{Maximum Segment Lifetime}$$

- Γιατί;
Δίνεται στον TCP client μια ευκαιρία επαναποστολής της τελικής ACK. (Ο server θα κάνει timeout αφού στείλει το τεμάχιο FIN και ξαναστείλει το FIN)
- Το MSL τίθεται στα 2 min ή 1 min ή 30 sec.

TIME_WAIT



Επαναφορά συνδέσεων



- Η επαναφορά (reset) συνδέσεων γίνεται με την ενεργοποίηση της σημαίας RST
- **Πότε ενεργοποιείται η σημαία RST;**
 - Όταν φθάνει η αίτηση σύνδεσης και δεν αναμένει καμία διαδικασία server στο σημείο προορισμού
 - Όταν αποσταλεί τεμάχιο που δεν αναμένεται καθόλου σε υπάρχουσα σύνδεση, π.χ. ο αύξων αριθμός είναι εκτός περιοχής
- Η επαναφορά μιας σύνδεσης αναγκάζει τον δέκτη του RST να πετάξει τα αποθηκευμένα δεδομένα. Ο δέκτης δεν επαληθεύει το τεμάχιο RST

Μεταφορά δεδομένων στο TCP



Οι εφαρμογές TCP μπορούν να καταταγούν ως:

αλληλοδραστική μεταφορά - telnet, rlogin
μαζική μεταφορά - ftp, mail, http

Το TCP έχει ευριστικές μεθόδους για να χειρίζεται αυτούς τους τύπους εφαρμογών.

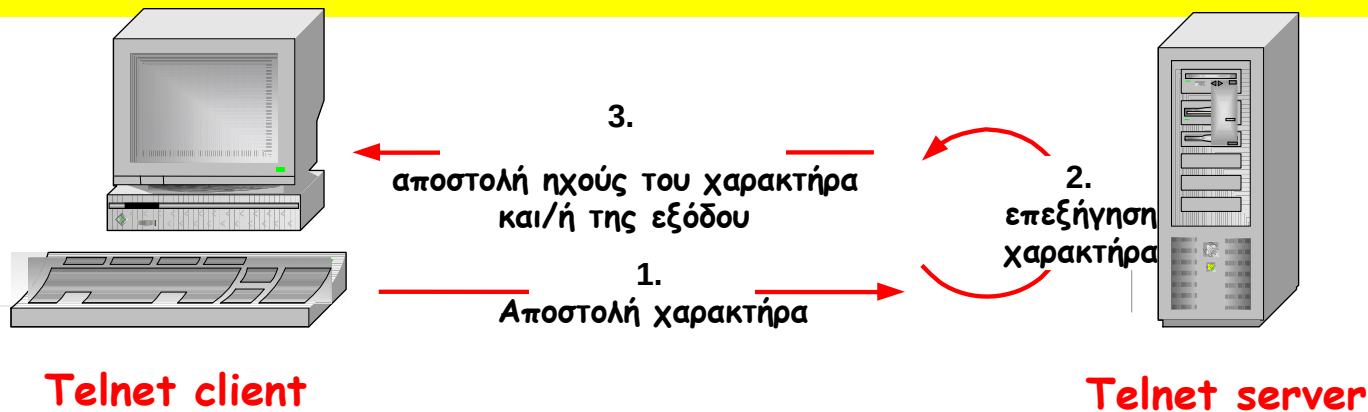
Για αλληλοδραστική μεταφορά δεδομένων:

- Επιχειρεί να περιορίσει τον αριθμό των πακέτων

Για μαζική μεταφορά δεδομένων:

- Επιχειρεί να ελέγξει τη ροή δεδομένων

Αλληλοδραστικές εφαρμογές: Telnet



Οι εφαρμογές απομακρυσμένου τερματικού (π.χ., Telnet) στέλνουν χαρακτήρες σε έναν server. Ο server επεξεργάζεται τον χαρακτήρα και στέλνει την έξοδό του στον client.

Για κάθε πληκτρολογούμενο χαρακτήρα, βλέπουμε 3 πακέτα:

1. **Client → Server**: Αποστολή πληκτρολογούμενου χαρακτήρα
2. **Server → Client**: Ηχώ του χαρακτήρα και επαλήθευση του πρώτου πακέτου
3. **Client → Server**: Επαλήθευση του δεύτερου πακέτου

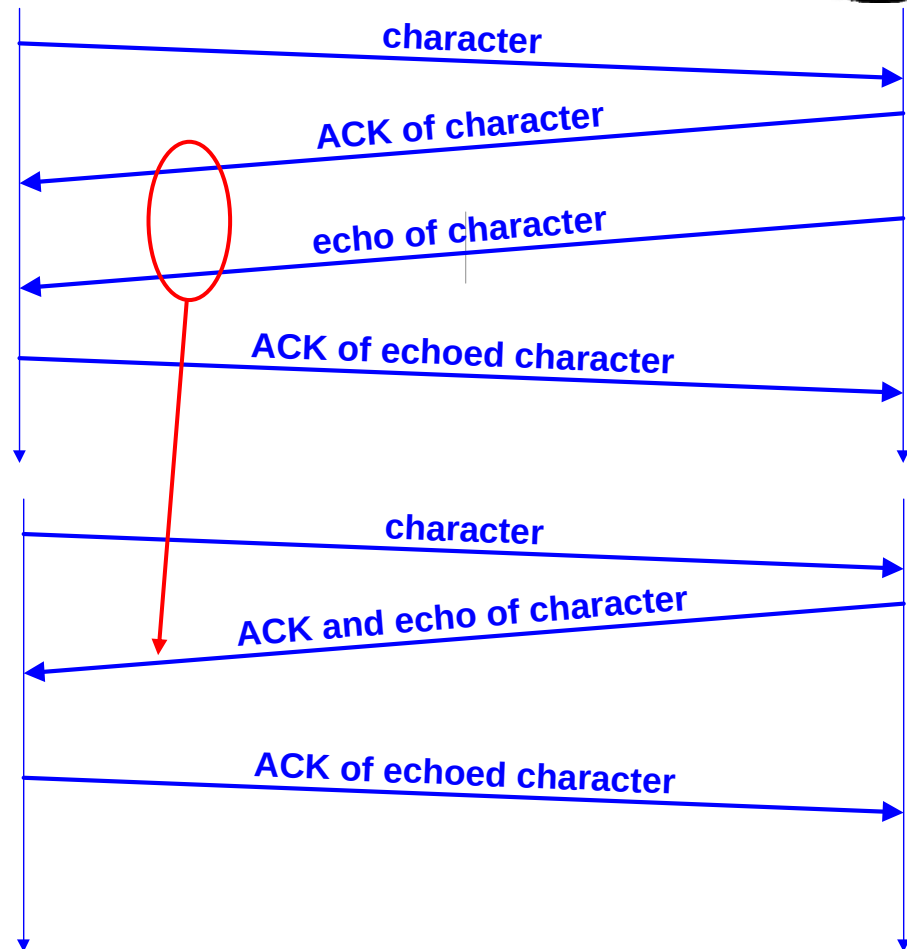
Αλληλοδραστικές εφαρμογές: Telnet



- Θα αναμέναμε 4 πακέτα ανά χαρακτήρα:

- Ωστόσο, εμφανίζεται η παραπλεύρως απεικονιζόμενη μορφή:

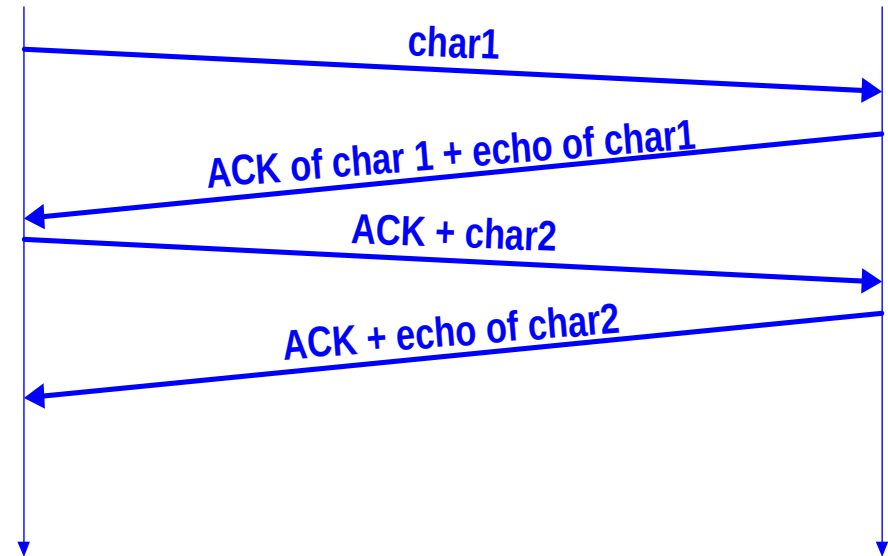
- Το TCP έχει καθυστερήσει την αποστολή μιας ACK



Σύνοδος telnet σε απομακρυσμένο host



- Παρατήρηση: Η μετάδοση των τεμαχίων ακολουθεί διαφορετικό τρόπο, δηλ., υπάρχουν μόνο δύο πακέτα ανά πληκτρολογούμενο χαρακτήρα
- Η καθυστερημένη επαλήθευση δεν εμφανίζεται
- Ο λόγος είναι ότι υπάρχουν πάντα δεδομένα έτοιμα προς αποστολή, όταν φθάνει η ACK





Αλγόριθμος του Nagle

- Έχει ως στόχο την αποφυγή της μη αποτελεσματικής χρήσης του εύρους ζώνης
- Πομπός:
 - αποθηκεύει προσωρινά όλα τα δεδομένα χρήστη, αν εκκρεμούν ανεπαλήθευτα δεδομένα
 - στέλνει, όταν όλα έχουν επαληθευτεί ή έχει δεδομένα που συμπληρώνουν ένα τεμάχιο MSS
- Δέκτης:
 - δίνει εντολή αποστολής, μόνο όταν μπορεί να αυξήσει επαρκώς το παράθυρο λήψης



Ιδιότητες του αλγορίθμου του Nagle

- Εφαρμόζεται μόνο σε μικρά πακέτα. Στις μεταφορές μεγάλων αρχείων υπάρχουν πάντα πλήρη MSS για αποστολή
- Ο αλγόριθμος είναι αυτοχρονιζόμενος:
 - βασικά εφαρμόζει Stop & Wait για μικρά τεμάχια
 - σε LAN, το μικρό RTT δεν εισάγει μεγάλη αναμονή, αλλά ο αλγόριθμος δεν είναι αποτελεσματικός
 - σε WAN, το μεγάλο RTT εισάγει περισσότερη αναμονή, αλλά ο αλγόριθμος είναι πιο αποτελεσματικός σε μακριές ζεύξεις
- Όταν απαιτείται μικρή καθυστέρηση, ο αλγόριθμος προκαλεί ανεπιθύμητες καθυστερήσεις
- Οι εφαρμογές μπορεί να απενεργοποιήσουν τον αλγόριθμο

Έλεγχος ροής στο TCP

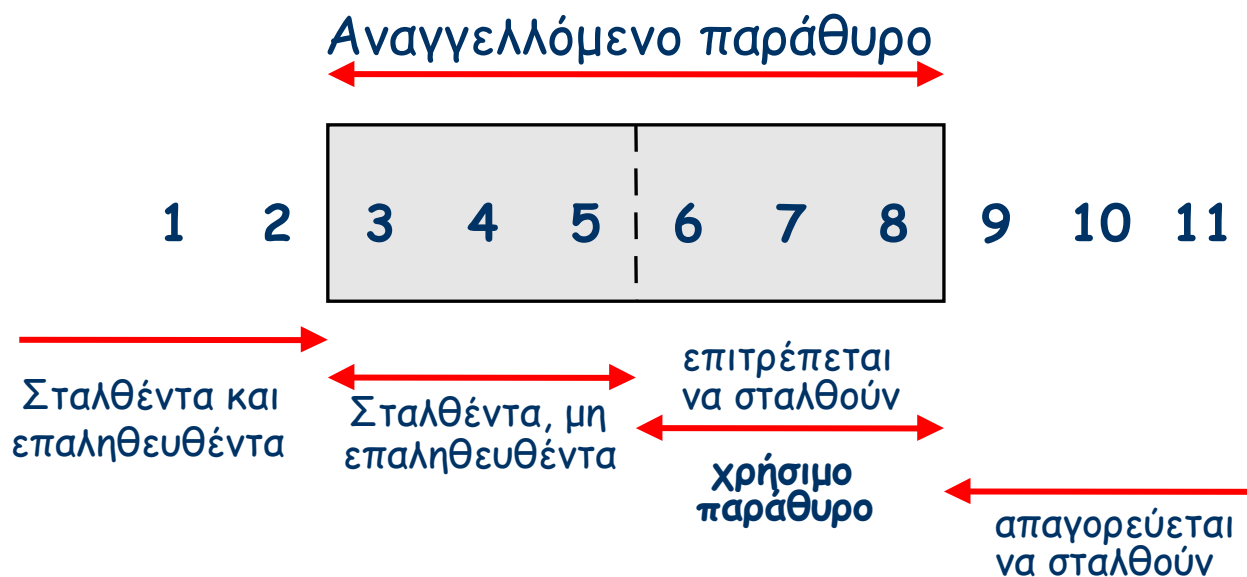


- Το TCP χρησιμοποιεί έλεγχο ροής με ολισθαίνον παράθυρο:
 - δεν χρησιμοποιεί NACK
 - μόνο συσσωρευτικές ACK
- Οι επαληθεύσεις δεν προκαλούν αυτόματα αλλαγές στο μέγεθος παραθύρου του αποστολέα
 - Ο αποδέκτης διαφημίζει στην επικεφαλίδα της επαλήθευσης ένα παράθυρο (win) σχετιζόμενο με τον χώρο προσωρινής μνήμης που διαθέτει
 - Ο αποστολέας μπορεί να στείλει δεδομένα μέχρι το διαφημιζόμενο παράθυρο, δηλαδή, τα byte
AckNo, AckNo+1, ..., AckNo + win -1
- Ο αποδέκτης μπορεί να επαληθεύσει χωρίς να αλλάξει το παράθυρο
- Ο αποδέκτης μπορεί να αλλάξει το παράθυρο χωρίς να επαληθεύσει

Έλεγχος ροής με ολισθαίνον παράθυρο



Το πρωτόκολλο ολισθαίνοντος παραθύρου λειτουργεί σε επίπεδο byte:

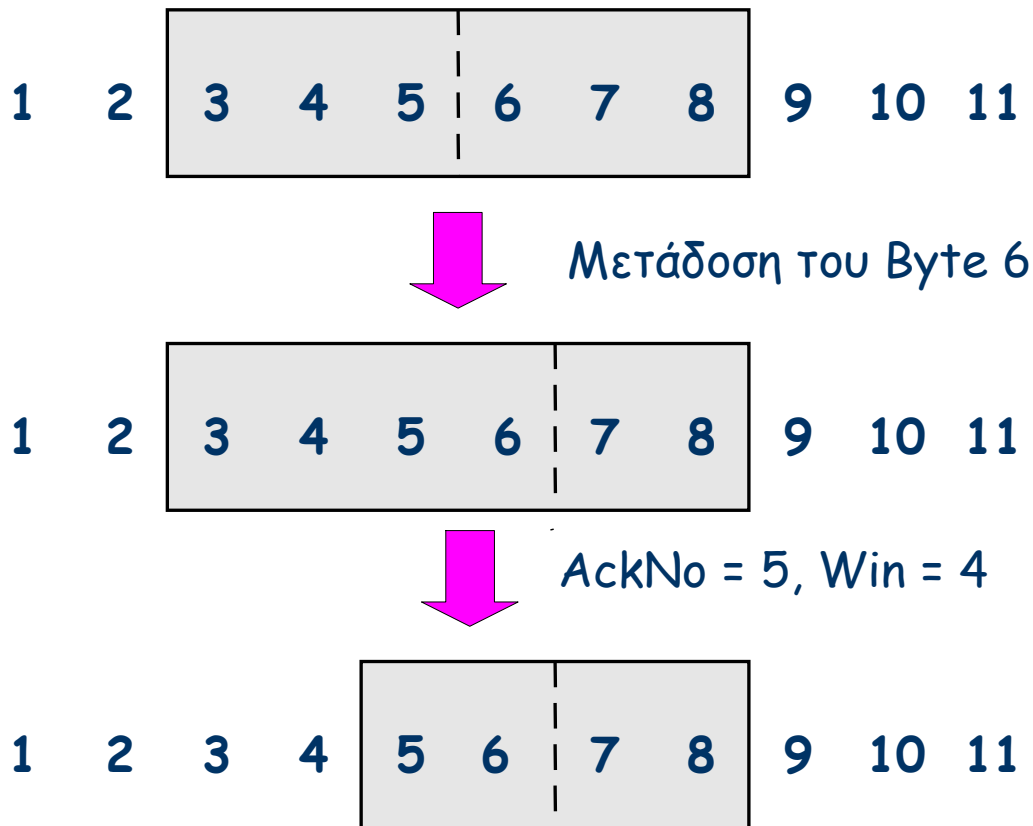


Ο πομπός μπορεί να στείλει μόνο τους αύξοντες αριθμούς 6,7,8.

Ολισθαίνον παράθυρο: κλείσιμο



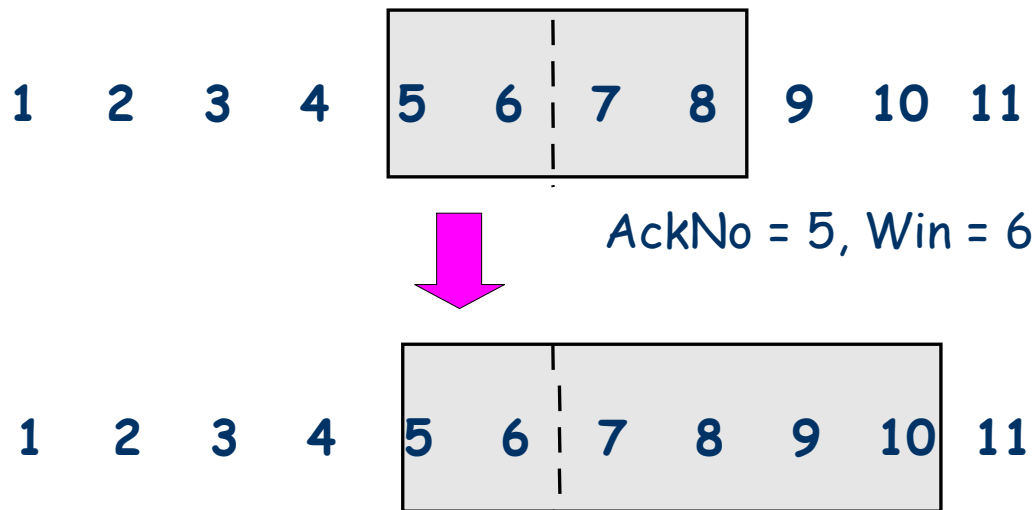
Αποστολή ενός byte (με SeqNo = 6) και λήψη της επαλήθευσης (AckNo = 5, Win=4):



Ολισθαίνον παράθυρο: άνοιγμα



Λήψη επαλήθευσης που μεγαλώνει το παράθυρο προς τα δεξιά
(AckNo = 5, Win=6):

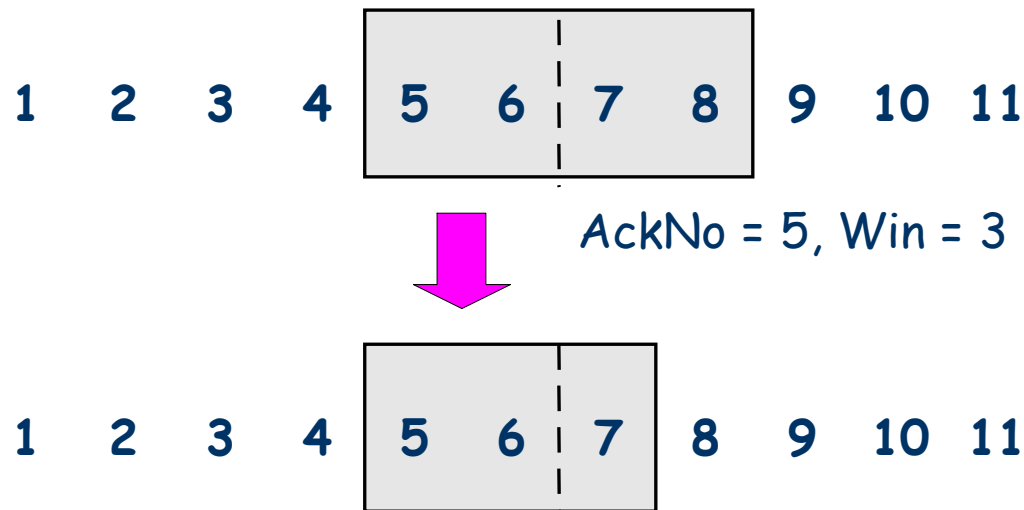


Ο δέκτης ανοίγει το παράθυρο όταν ο καταχωρητής TCP αδειάζει
(εννοώντας ότι τα δεδομένα παραδίδονται στην εφαρμογή).

Ολισθαίνον παράθυρο: συρρίκνωση



Λήψη επαλήθευσης που περιορίζει το παράθυρο από δεξιά (AckNo = 5, Win=3):



Η συρρίκνωση παραθύρου δεν πρέπει να χρησιμοποιείται

Διαχείριση παραθύρου στο TCP



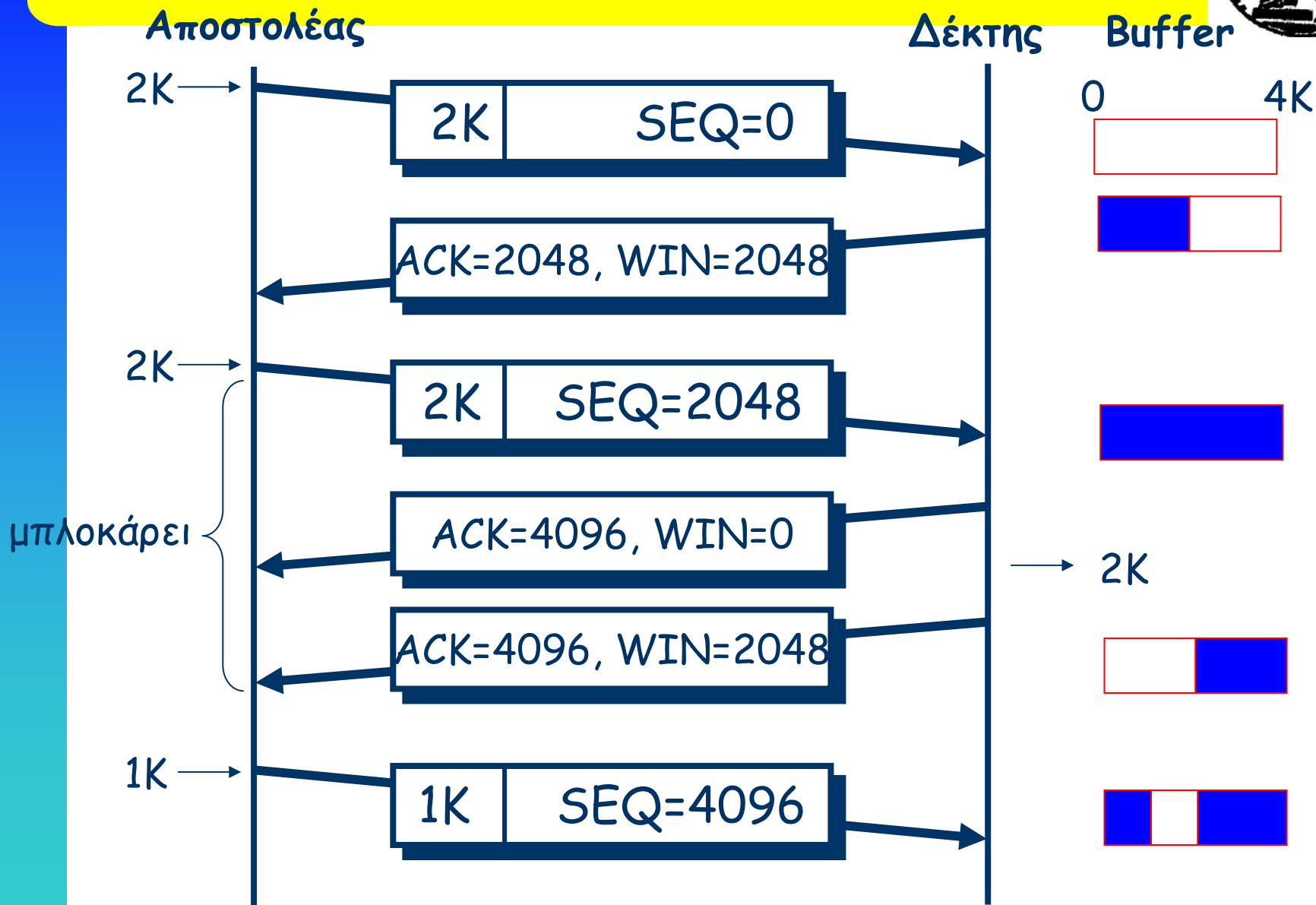
- Ο δέκτης επιστρέφει δύο παραμέτρους στον πομπό

AckNo	window size (win)
32 bits	16 bits

- Η ερμηνεία είναι:
 - Είμαι έτοιμος να δεχθώ νέα δεδομένα με
 $SeqNo = AckNo, AckNo+1, \dots, AckNo+Win-1$
- Ο δέκτης μπορεί να επαληθεύει δεδομένα χωρίς να ανοίγει το παράθυρο
- Ο δέκτης μπορεί να αλλάζει το μέγεθος του παραθύρου χωρίς να επαληθεύει δεδομένα



Ολισθαίνον παράθυρο: Παράδειγμα





Έλεγχος σφαλμάτων

- Δύο τύποι σφαλμάτων:
 - Απωλεσθέντα τεμάχια
 - Κατεστραμμένα τεμάχια
- Ο δέκτης στέλνει επαληθεύσεις (ACKs) για τα σωστά τεμάχια. Οι ACK μπορεί να είναι συσσωρευτικές.
- Το TCP έχει αθροίσματα ελέγχου για την επικεφαλίδα και τα δεδομένα. Τεμάχια με μη έγκυρα αθροίσματα ελέγχου απορρίπτονται
- Ο αποστολέας TCP διατηρεί ένα χρονόμετρο. Αν μια ACK δεν ληφθεί έγκαιρα επαναμεταδίδει, το τεμάχιο.

Έλεγχος σφαλμάτων



γεγονός: λήψη δεδομένων
από την εφαρμογή

δημιουργία και αποστολή τεμαχίου



γεγονός: λήξη χρονομέτρου για
το τεμάχιο με $\text{seq} = y$

επαναμετάδοση τεμαχίου

γεγονός: λήψη ACK,
με $\text{ACK} = y$

επεξεργασία ACK

απλοποιημένος αποστολέας

- μονόφορη μετάδοση δεδομένων
- όχι έλεγχος ροής και συμφόρησης



Επαληθεύσεις στο TCP

- Ο δέκτης στέλνει *ACK* στον πομπό
- Ο αριθμός της *ACK* είναι ο επόμενος αναμενόμενος αύξων αριθμός
- Καθυστερημένη *ACK*: ο δέκτης TCP συνήθως καθυστερεί τη μετάδοση μιας (για περίπου 200ms)
- Οι *ACK* δεν καθυστερούνται όταν τα πακέτα λαμβάνονται εκτός σειράς

TCP: δημιουργία επαληθεύσεων



Γεγονός

Ενέργεια του δέκτη TCP

άφιξη τεμαχίου στην κανονική σειρά, όχι κενά, οτιδήποτε άλλο έχει ήδη επαληθευτεί

καθυστερημένη ACK. Αναμονή 200ms για το επόμενο τεμάχιο. Αν δεν υπάρχει επόμενο τεμάχιο, στέλνει ACK

άφιξη τεμαχίου στην κανονική σειρά, όχι κενά, εκκρεμεί μία καθυστερημένη ACK

άμεση αποστολή μιας συσσωρευτικής ACK

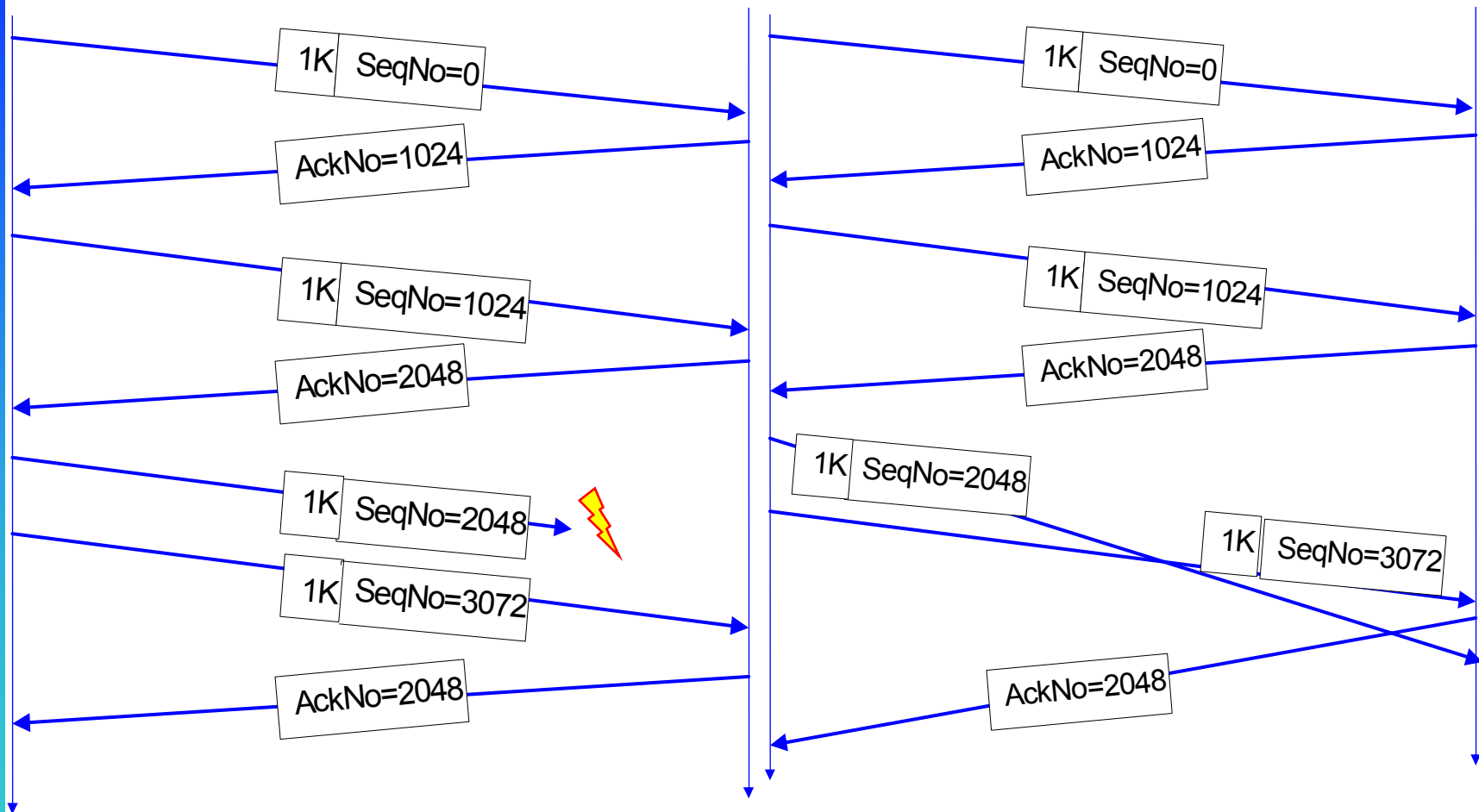
άφιξη τεμαχίου εκτός σειράς με μεγαλύτερο αύξοντα αριθμό από τον αναμενόμενο

αποστολή επαναληπτικής ACK, που να δείχνει τον αύξοντα αριθμό του επόμενου αναμενόμενου byte

άφιξη τεμαχίου που η αρίθμησή του είναι εν μέρει ή πλήρως εντός του αναμενόμενου διαστήματος

άμεση αποστολή ACK, αν το τεμάχιο αρχίζει στο κατώτερο άκρο του αναμενόμενου διαστήματος αρίθμησης

Επαληθεύσεις στο TCP



Απώλεια τεμαχίου

Άφιξη εκτός σειράς

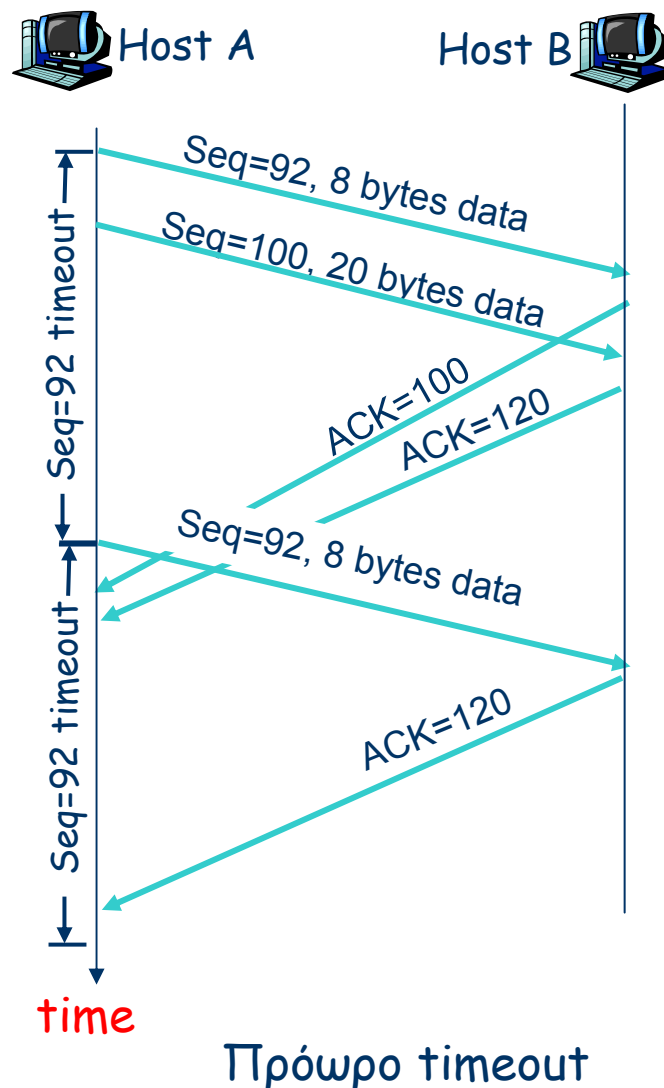
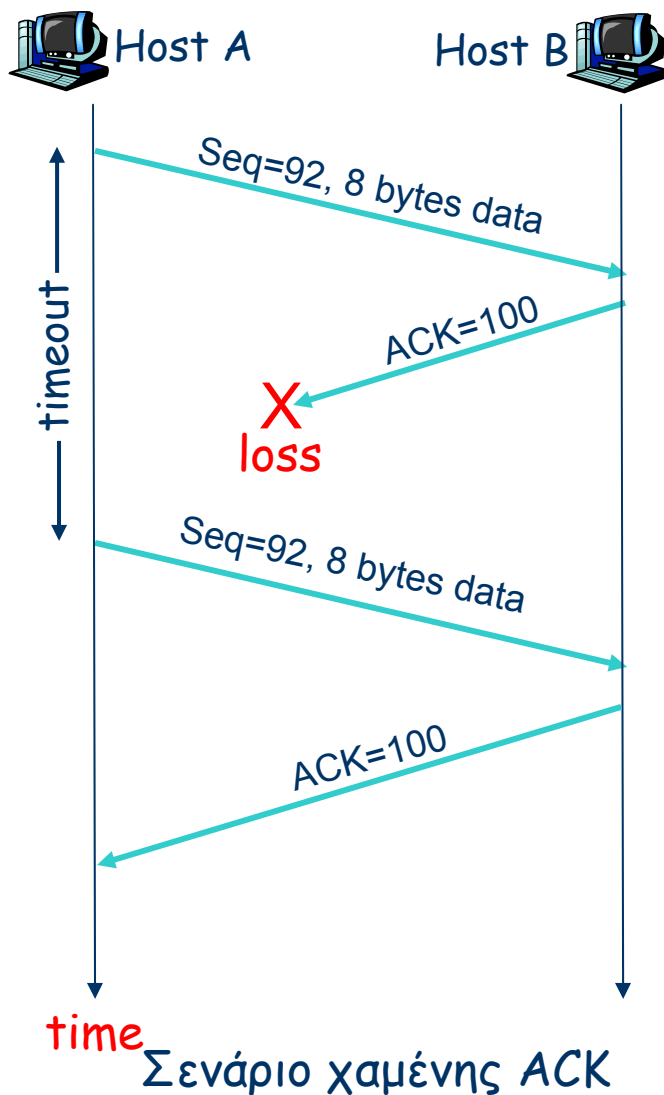


Επαναμεταδόσεις στο TCP

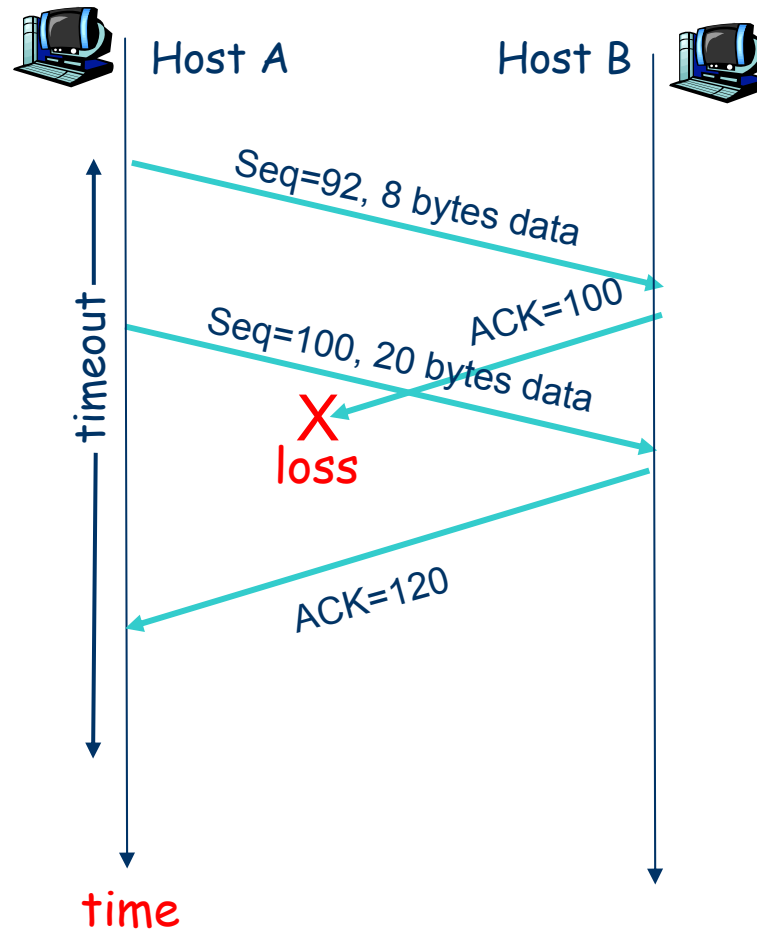
Ένας πομπός TCP επαναμεταδίδει ένα τεμάχιο, όταν θεωρήσει ότι το υπόψη τεμάχιο έχει χαθεί.

- Δεν έχει ληφθεί ACK και έχει λήξει το χρονόμετρο
- Έχουν ληφθεί πολλαπλές ACK για το ίδιο τεμάχιο

TCP: σενάρια επαναμετάδοσης



TCP: σενάρια επαναμετάδοσης



Σενάριο συγκεντρωτικής ACK

Ταχεία επαναμετάδοση

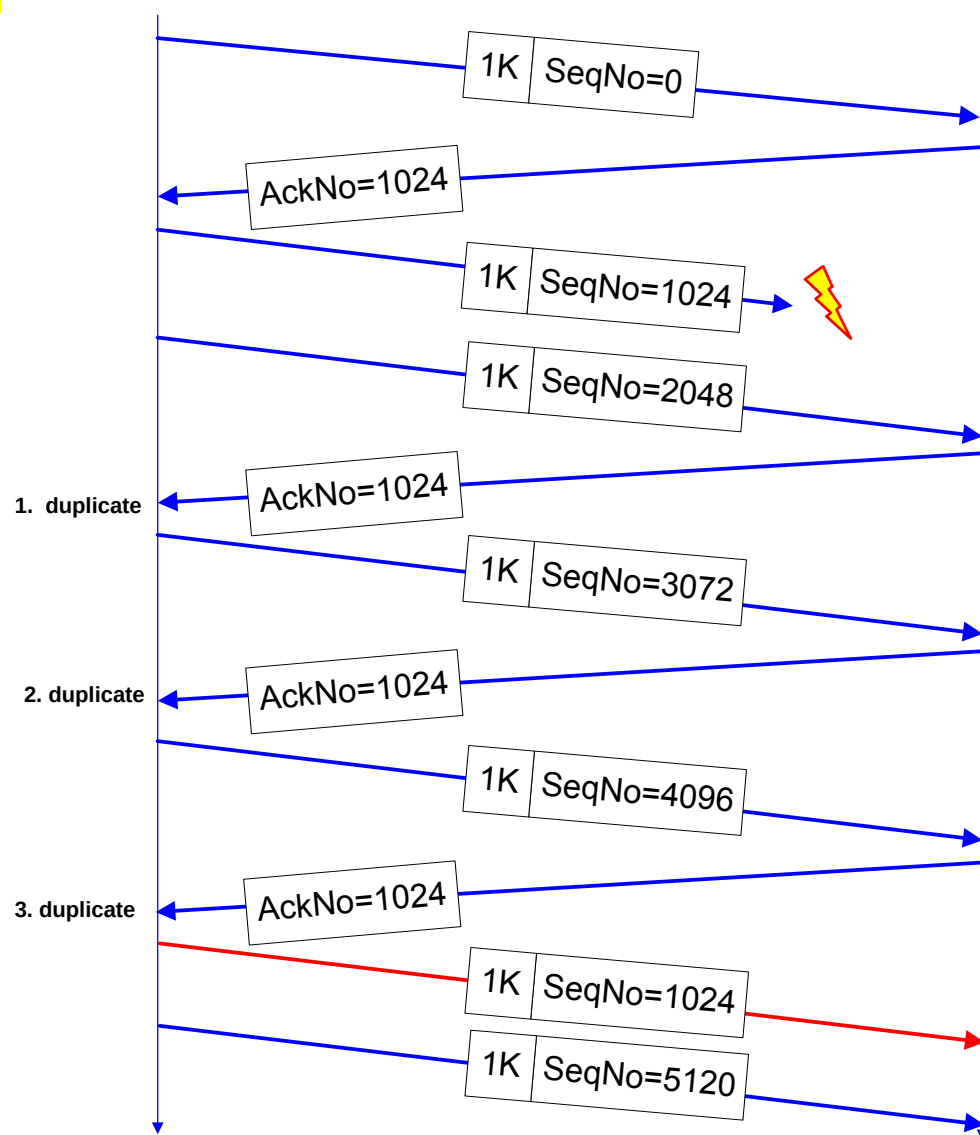


- Η περίοδος timeout είναι συχνά σχετικά μεγάλη:
 - μεγάλη καθυστέρηση πριν την αποστολή του χαμένου πακέτου
- Ανίχνευση των χαμένων τεμαχίων μέσω διπλών επαληθεύσεων.
 - Ο πομπός στέλνει συχνά πολλά τεμάχια το ένα πίσω απ' το άλλο
 - Αν χαθεί τεμάχιο, θα υπάρχουν ενδεχομένως πολλές ίδιες επαληθεύσεις.
- **ταχεία επαναμετάδοση:** επαναποστολή του τεμαχίου πριν τη λήξη της χρονομέτρησης

Λήψη διπλών ACK



- Αν ληφθούν τρεις ή περισσότερες διπλές ACK στη σειρά, ο πομπός TCP θεωρεί ότι χάθηκε τεμάχιο.
- Τότε το TCP επαναμεταδίδει το τεμάχιο που θεωρεί ότι χάθηκε, χωρίς να περιμένει τη λήξη χρόνου.
- Τούτο διορθώνει μεμονωμένες απώλειες τεμαχίων



Διαχείριση χρονομετρητών στο TCP



- Το TCP χρησιμοποιεί πολλούς χρονομετρητές. Ο σπουδαιότερος είναι ο **χρονομετρητής επαναμετάδοσης (retransmission timer)**
- Όταν στέλνεται ένα τεμάχιο, ξεκινά ένας χρονομετρητής αναμετάδοσης
- Αν η λήψη του τεμαχίου επαληθευτεί πριν εκπνεύσει ο χρόνος, τότε ο χρονομετρητής σταματά
- Αν εκπνεύσει ο χρόνος πριν φθάσει η επαλήθευση, το τεμάχιο μεταδίδεται ξανά
- Πόσο μεγάλο πρέπει να είναι το χρονικό διάστημα πριν λήξει η χρονομέτρηση;

TCP: χρόνοι Round Trip και Timeout



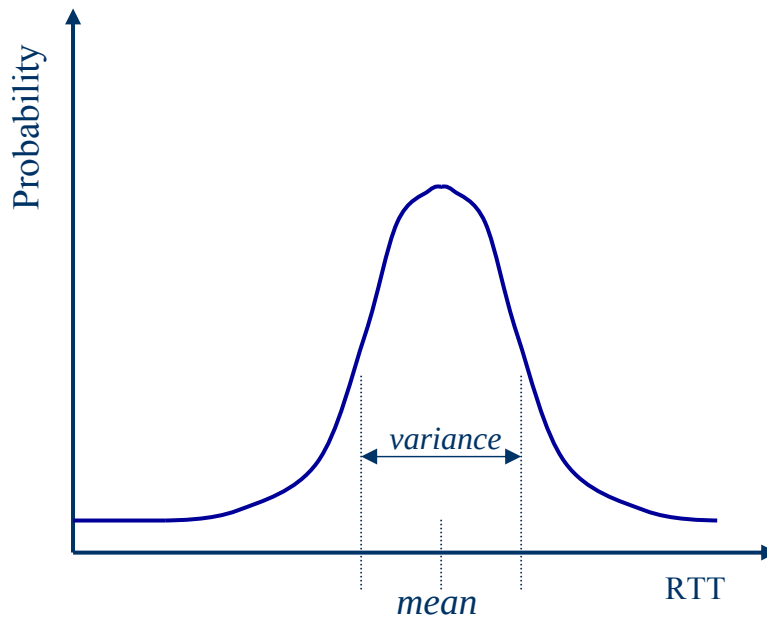
Πώς τίθεται η τιμή του timeout επαναμετάδοσης (Retransmission Timeout, RTO) στο TCP;

- μεγαλύτερη από RTT
 - αλλά το RTT μεταβάλλεται
- πολύ μικρό: πρόωρο timeout
 - Άσκοπες επαναμεταδόσεις
- πολύ μεγάλο: αργή αντίδραση, όταν χάνεται τεμάχιο

TCP: χρόνοι Round Trip και Timeout



- Θα υπάρχει κάποια (άγνωστη) κατανομή των RTT.
- Προσπαθούμε να εκτιμήσουμε ένα RTO για να ελαχιστοποιήσουμε την πιθανότητα μιας εσφαλμένης λήξης χρόνου.



- Οι ουρές στους δρομολογητές μεγαλώνουν όταν υπάρχει περισσότερη κίνηση, μέχρι να γίνουν ασταθείς.
- Καθώς αυξάνει το φορτίο, η variance της καθυστέρησης αυξάνει απότομα.

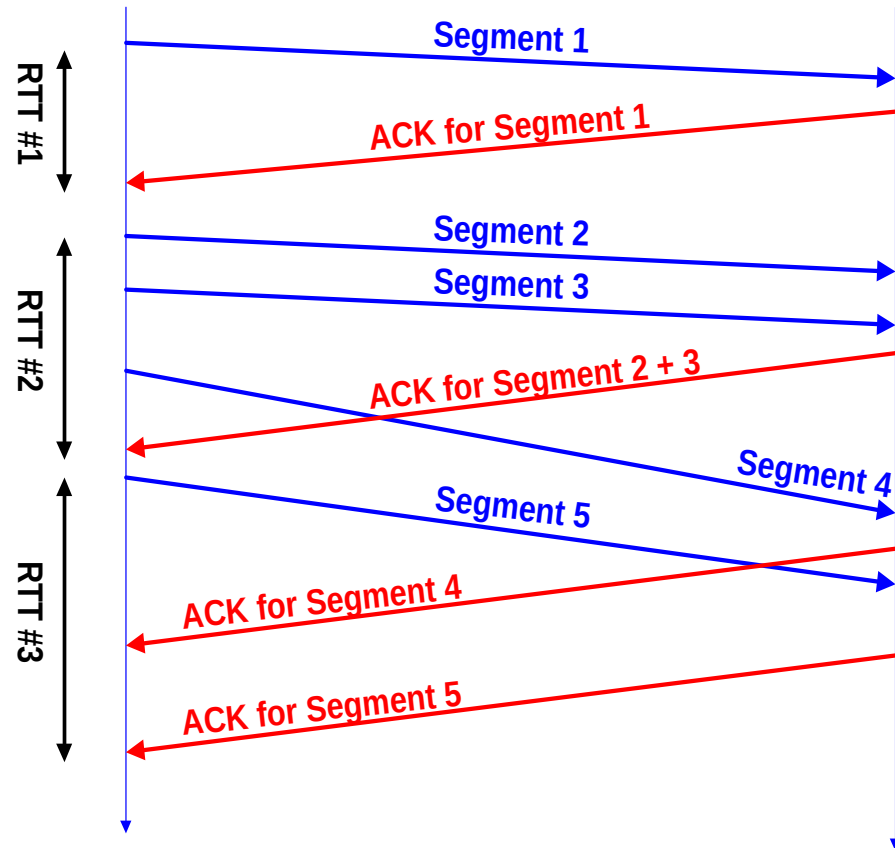


Ρύθμιση της τιμής του Timeout



Η τιμή του RTO τίθεται βάσει των μετρήσεων του round-trip time (RTT) που πραγματοποιεί το TCP

- Κάθε σύνδεση TCP μετράει τη χρονική διαφορά μεταξύ της αποστολής ενός τεμαχίου και της λήψης της αντίστοιχης ACK
- Υπάρχει μόνο μια μέτρηση σε ισχύ κάθε φορά (δηλ., οι μετρήσεις δεν επικαλύπτονται)
- Στο διπλανό σχήμα φαίνονται τρεις μετρήσεις RTT



Ρύθμιση της τιμής του Timeout



- Η RTO υπολογίζεται βάσει των μετρήσεων του RTT
 - Χρησιμοποιείται εκθετικός κινούμενος μέσος όρος για τον υπολογισμό της καθυστέρησης ($srtt$) και της variance της καθυστέρησης ($rttvar$)

- Οι μετρήσεις RTT εξομαλύνονται ως εξής:

$$srtt_{n+1} = \alpha RTT + (1 - \alpha) srtt_n$$

$$rttvar_{n+1} = \beta (| RTT - srtt_n |) + (1 - \beta) rttvar_n$$

$$RTO_{n+1} = srtt_{n+1} + 4 rttvar_{n+1}$$

- Οι τιμές του $\alpha = 1/8$ και $\beta = 1/4$

Ρύθμιση της τιμής του Timeout



- Αρχική τιμή του RTO:
 - Ο πομπός πρέπει να θέσει την αρχική τιμή του RTO σε
$$RTO_0 = 3 \text{ sec}$$
- Υπολογισμός του RTO μετά την άφιξη των πρώτων μετρήσεων RTT
$$srtt_1 = RTT$$
$$rttvar_1 = RTT / 2$$
$$RTO_1 = srtt_1 + 4 rttvar_{n+1}$$
- Όταν εμφανιστεί ένα timeout, η τιμή του RTO διπλασιάζεται
$$RTO_{n+1} = \min (2 RTO_n, 64) \text{ seconds}$$

Τούτο ονομάζεται *εκθετική οπισθοχώρηση*

Αλγόριθμος του Καρν



Αν ληφθεί ACK για επαναμεταδοθέν τεμάχιο, ο πομπός δεν μπορεί να ξέρει αν η ACK ανήκει στο αρχικό ή στο επαναμεταδοθέν.

→ Η μέτρηση RTT είναι αμφίβολη σ' αυτήν την περίπτωση

- **Αλγόριθμος του Καρν:**

- Μην ενημερώνεις την RTT για τεμάχια που επαναμεταδόθηκαν
- Ξαναξεκίνα τις μετρήσεις RTT μόνο μετά τη λήψη ACK που αφορά κανονικό τεμάχιο

