ΕΠΙΠΕΔΟ ΜΕΤΑΦΟΡΑΣ
TCP/UDP

Τμήμα Πληροφορικής, Σχολή Θετικών Επιστημών,
Παν/μιο Θεσσαλίας

Στόχοι της Ενότητας

- Κατανόηση εννοιών των υπηρεσιών του επιπέδου μεταφοράς:
  - πολύπλεξη, αποπολύπλεξη
  - αξιόπιστη μεταφορά δεδομένων
  - έλεγχος ροής (flow control)
  - έλεγχος συμφόρησης (congestion control)
- Γνωριμία με τα πρωτόκολλα επιπέδου μεταφοράς στο Internet:
  - UDP: μεταφορά χωρίς σύνδεση
  - TCP: μεταφορά με σύνδεση
  - Έλεγχος συμφόρησης στο TCP

Ορολογία Δικτυακών Εφαρμογών

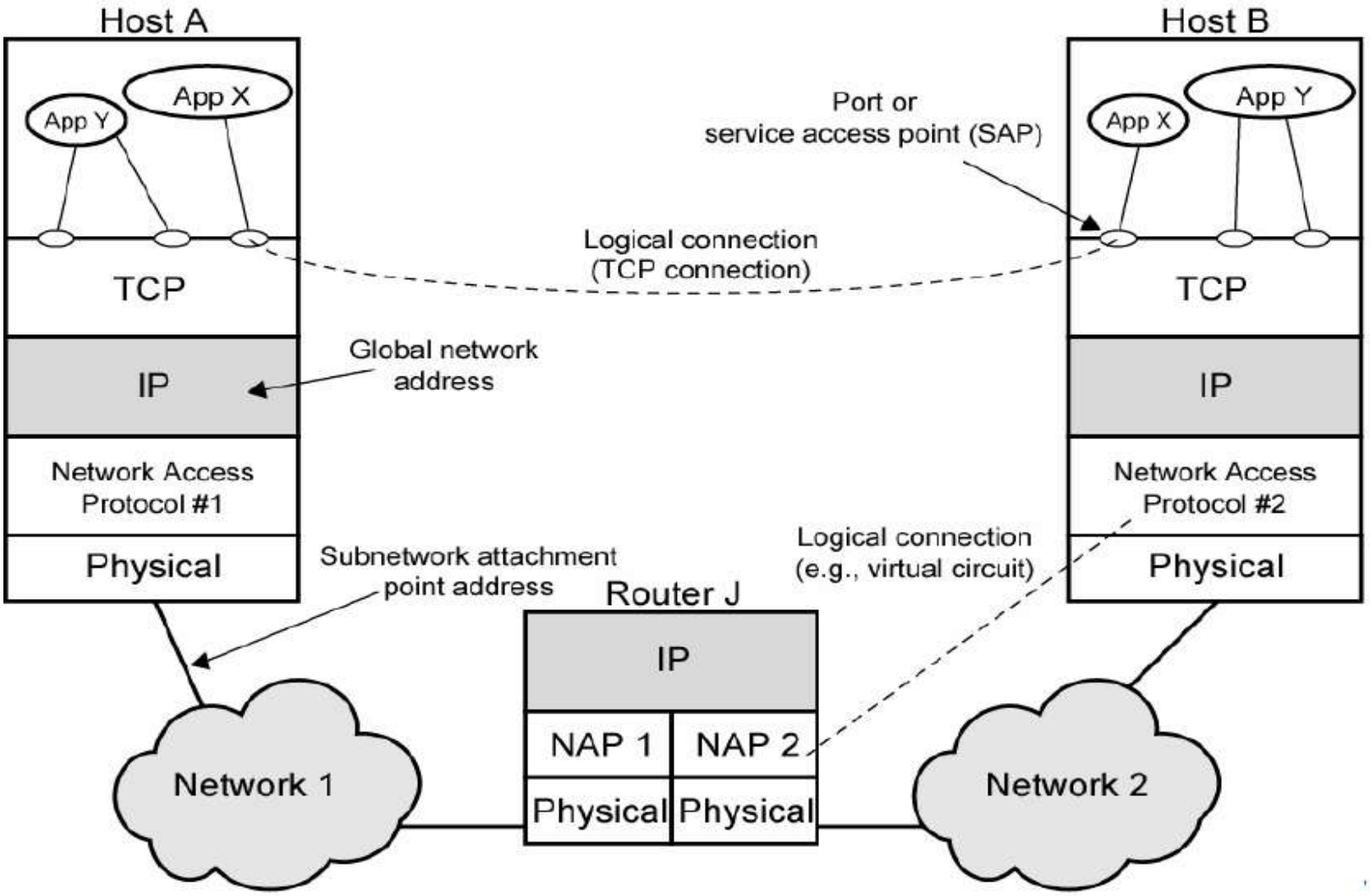
Διεργασία: πρόγραμμα που τρέχει σε ένα σύστημα

- Μέσα στον ίδιο κόμβο, δύο διεργασίες επικοινωνούν χρησιμοποιώντας Inter-Process Communication – IPC (ορίζεται από το ΛΣ)
- Διεργασίες που τρέχουν σε διαφορετικά συστήματα επικοινωνούν με ένα πρωτόκολλο επιπέδου εφαρμογής

Πράκτορας χρήστη (user agent): αλληλεπιδρά με το χρήστη (προς τα «πάνω») και το δίκτυο (προς τα «κάτω»)

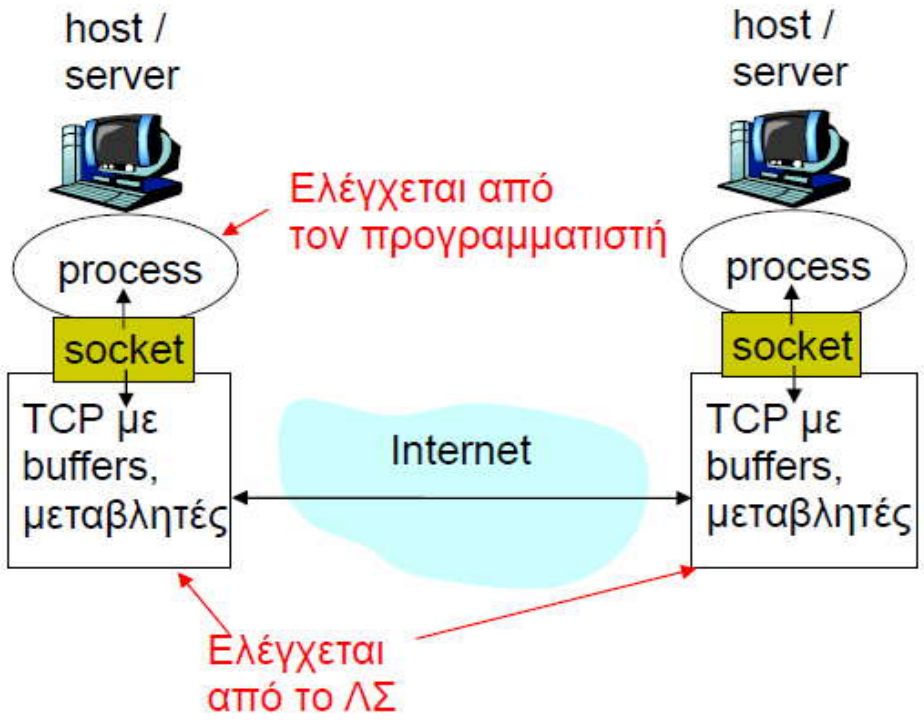
- Υλοποιεί τη διεπαφή με το χρήστη και το πρωτόκολλο εφαρμογής
  - Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

Έννοιες του TCP/IP



Επικοινωνία Διεργασιών σε Δίκτυο

- Η διεργασία στέλνει/ λαμβάνει μηνύματα προς/από την υποδοχή (socket)
- Socket: Προγραμματιστική διεπαφή (API)
  - (1) επιλογή πρωτοκόλλου μεταφοράς
  - (2) ορισμός παραμέτρων επικοινωνίας



Τι Υπηρεσία Επιπέδου Μεταφοράς Χρειάζεται μια Εφαρμογή;

Απώλεια δεδομένων

- Ορισμένες εφαρμογές (π.χ., ήχος) αντέχουν απώλειες
- Άλλες εφαρμογές (π.χ., μεταφορά αρχείων, telnet) απαιτούν 100% αξιόπιστη μεταφορά δεδομένων

Χρονισμός

- Ορισμένες εφαρμογές (π.χ., διαδικτυακή τηλεφωνία, διαδραστικά παιχνίδια) απαιτούν χαμηλή καθυστέρηση

Εύρος Ζώνης

- Ορισμένες εφαρμογές (π.χ., πολυμέσα) απαιτούν μια ελάχιστη ποσότητα εύρους ζώνης να είναι διαθέσιμη
- Άλλες εφαρμογές κάνουν χρήση όσου εύρους ζώνης διαθέτουν

Απαιτήσεις των Εφαρμογών από την Υπηρεσία Επιπέδου Μεταφοράς

Εφαρμογή	Απώλεια	Εύρος Ζώνης	Ευαισθησία Χρόνου
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Υπηρεσίες Επιπέδου Μεταφοράς που παρέχει το Internet στις εφαρμογές

- Το Internet παρέχει δύο ειδών υπηρεσίες στις εφαρμογές που το χρησιμοποιούν:
  - την υπηρεσία με σύνδεση (connection-oriented transport)
  - την υπηρεσία χωρίς σύνδεση (connectionless transport)

Υπηρεσία με Σύνδεση (Connection-oriented Transport)

- Στην υπηρεσία με σύνδεση, δημιουργείται «σύνδεση» μεταξύ των τελικών συστημάτων (end-to-end) πριν τη μετάδοση των πακέτων
- Κάθε πακέτο περιέχει τη διεύθυνση προορισμού του και δρομολογείται αξιόπιστα και ανεξάρτητα από τα υπόλοιπα, βήμα-βήμα μέσα από το πλέγμα των διασυνδεδεμένων δρομολογητών διατηρώντας τη σειρά του
- Στο Internet η αξιόπιστη υπηρεσία με σύνδεση προσφέρεται από το πρωτόκολλο TCP

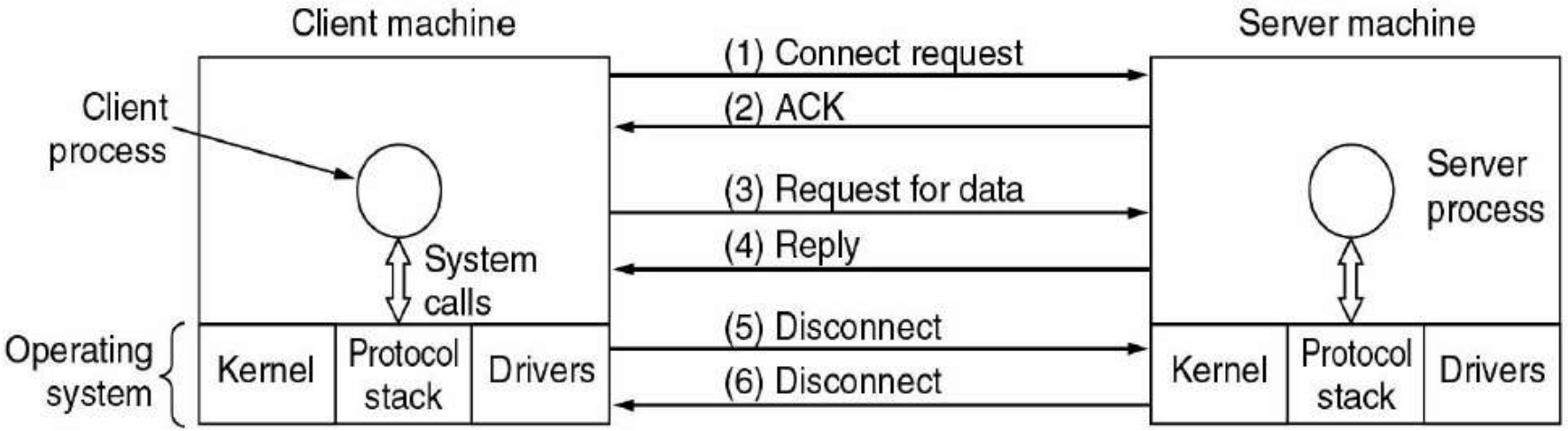
Υπηρεσία χωρίς Σύνδεση (Connectionless Transport)

- Στην υπηρεσία χωρίς σύνδεση, κάθε πακέτο περιέχει τη διεύθυνση προορισμού του και δρομολογείται ανεξάρτητα από τα υπόλοιπα, βήμα-βήμα μέσα από το πλέγμα των διασυνδεδεμένων δρομολογητών, χωρίς να δημιουργείται σύνδεση, χωρίς να παρέχεται αξιοπιστία κατά τη μεταφορά και χωρίς να διατηρείται η σειρά των πακέτων
- Στο Internet η υπηρεσία χωρίς σύνδεση προσφέρεται από το πρωτόκολλο UDP

Συνδέσεις και Δίκτυα Μεταγωγής Πακέτων

- «Σύνδεση» ονομάζουμε την προετοιμασία των τελικών συστημάτων ώστε να παρέχεται αξιοπιστία στις εφαρμογές που τη χρησιμοποιούν
- Η αξιόπιστη υπηρεσία με σύνδεση προσφέρεται μεταξύ τελικών συστημάτων (end-to-end) χωρίς να εμπλέκονται οι ενδιάμεσοι δρομολογητές
- Κάθε «σύνδεση» δεσμεύει χώρους προσωρινής αποθήκευσης (buffers) στα συστήματα που θα επικοινωνήσουν

Αλληλεπίδραση Πελάτη – Εξυπηρετητή με Σύνδεση



- Επισκόπηση της αλληλεπίδρασης τμημάτων λογισμικού στην υπηρεσία με σύνδεση (TCP)

Υποδοχές Berkeley (Berkeley Sockets)

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Υπηρεσίες Πρωτοκόλλων Μεταφοράς του Internet

Υπηρεσία TCP:

- connection-oriented: απαιτείται εγκατάσταση σύνδεσης μεταξύ πελάτη και εξυπηρετητή
- μεταφορά δεδομένων χωρίς σφάλματα και με τη σειρά
- έλεγχος ροής: ο αποστολέας δεν θα κατακλύσει τον παραλήπτη
- έλεγχος συμφόρησης: αυξομείωση ρυθμού μετάδοσης αποστολέα όταν το δίκτυο είναι υπερφορτωμένο
- δεν παρέχει: χρονισμό, εγγυήσεις ελάχιστου εύρους ζώνης

Υπηρεσία UDP:

- μη-αξιόπιστη μεταφορά δεδομένων μεταξύ διεργασιών αποστολέα και παραλήπτη
- δεν παρέχει: δημιουργία σύνδεσης, αξιοπιστία, έλεγχο ροής, έλεγχο συμφόρησης, χρονισμό ή εγγυήσεις εύρους ζώνης

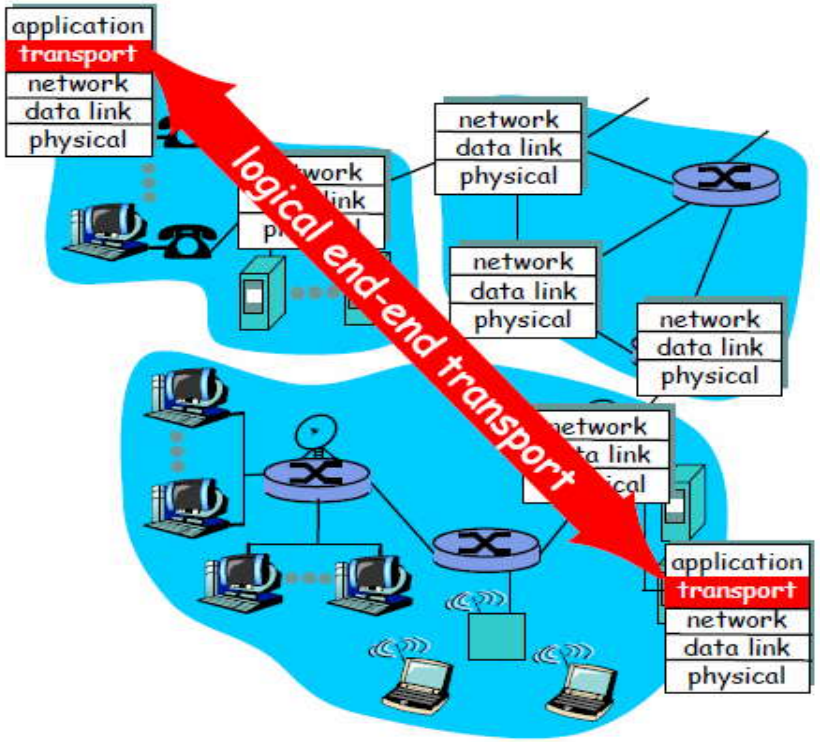
Ερ: Γιατί υπάρχει το UDP;

Πρωτόκολλα Μεταφοράς που Χρησιμοποιούν οι Εφαρμογές

Εφαρμογή	Πρωτόκολλο Εφαρμογής	Πρωτόκολλο Μεταφοράς
e-mail	SMTP [RFC 2821]	TCP
απομακρυσμένη πρόσβαση	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
μεταφορά αρχείων	FTP [RFC 959]	TCP
streaming multimedia	RTSP, RTP, HTTP, ή ιδιωτικό	TCP ή UDP
Τηλεφωνία στο Internet	VoIP ή ιδιωτικό	τυπικά UDP

Υπηρεσίες και Πρωτόκολλα Επιπέδου Μεταφοράς

- Αξιόπιστη παράδοση, με τη σειρά: TCP
  - έλεγχος συμφόρησης
  - έλεγχος ροής
  - εγκατάσταση σύνδεσης (connection setup)
- Μη-αξιόπιστη παράδοση, χωρίς σειρά: UDP
  - απλά η επέκταση της βέλτιστης προσπάθειας ("best-effort") του IP



- Υπηρεσίες που **δεν** παρέχονται:
  - εγγυήσεις καθυστέρησης
  - εγγυήσεις εύρους ζώνης

Πολύπλεξη/Αποπολύπλεξη

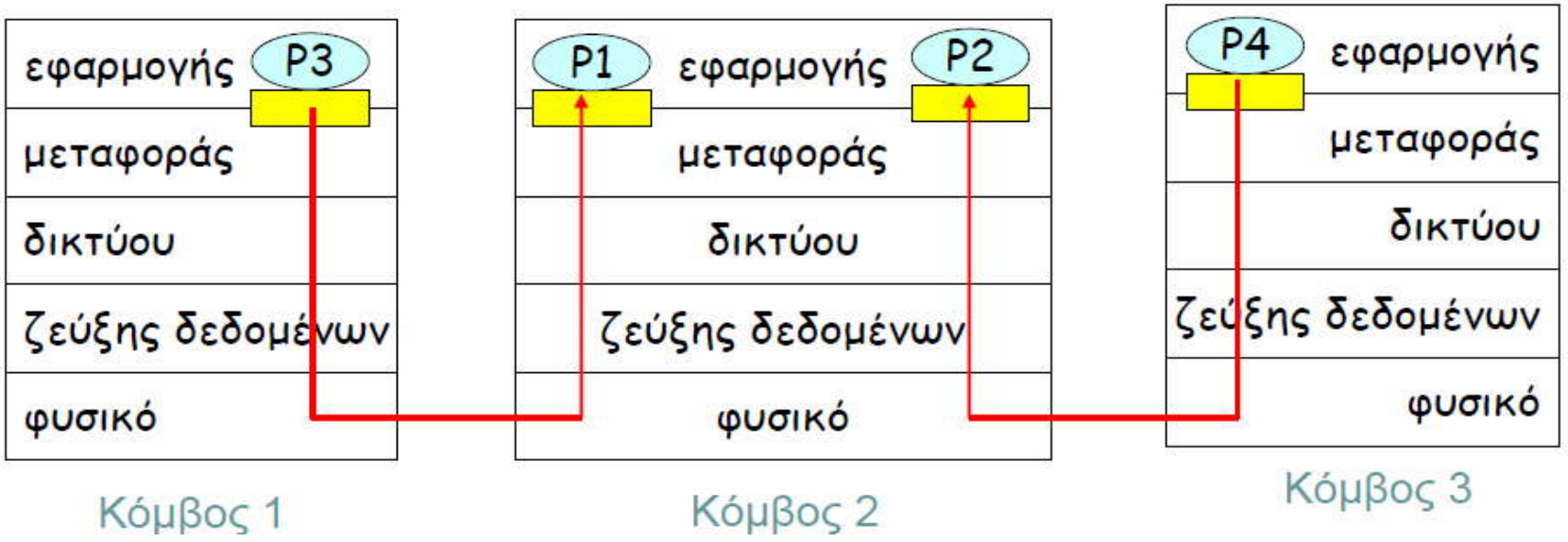
Αποπολύπλεξη στον παραλήπτη:

Παράδοση των τμημάτων που λαμβάνονται στη σωστή υποδοχή

Πολύπλεξη στον αποστολέα:

Συλλογή δεδομένων από πολλές υποδοχές, τοποθέτηση επικεφαλίδων (αργότερα θα χρησιμοποιηθούν για αποπολύπλεξη)

= socket = process



Πολύπλεξη/Αποπολύπλεξη

- Κάθε αριθμός θύρας (port number) είναι ένας ακέραιος 16 bit (από 0-65535)
- Οι αριθμοί από 0-1023 είναι δεσμευμένοι: προκαθορισμένοι (well-known) αριθμοί θυρών για χρήση από πρωτόκολλα εφαρμογών όπως το HTTP (αριθμός θύρας 80) ή το FTP (21)
- Ο αριθμός θύρας συσχετίζεται με μια υποδοχή, που με τη σειρά της συνδέεται με μια διεργασία

Πως δουλεύει η αποπολύπλεξη

- Ο κόμβος λαμβάνει αυτοδύναμα πακέτα (ΑΠ) IP
  - κάθε ΑΠ έχει διεύθυνση IP πηγής, και διεύθυνση IP προορισμού
  - κάθε ΑΠ μεταφέρει ένα τμήμα (segment) επιπέδου μεταφοράς
  - κάθε τμήμα έχει αριθμό θύρας πηγής και προορισμού (προκαθορισμένοι αριθμοί θυρών για συγκεκριμένες εφαρμογές)
- Ο κόμβος χρησιμοποιεί διευθύνσεις IP, πρωτόκολλα επιπέδου μεταφοράς & αριθμούς θυρών για να στείλει το τμήμα στην κατάλληλη υποδοχή

μορφή τμήματος TCP/UDP

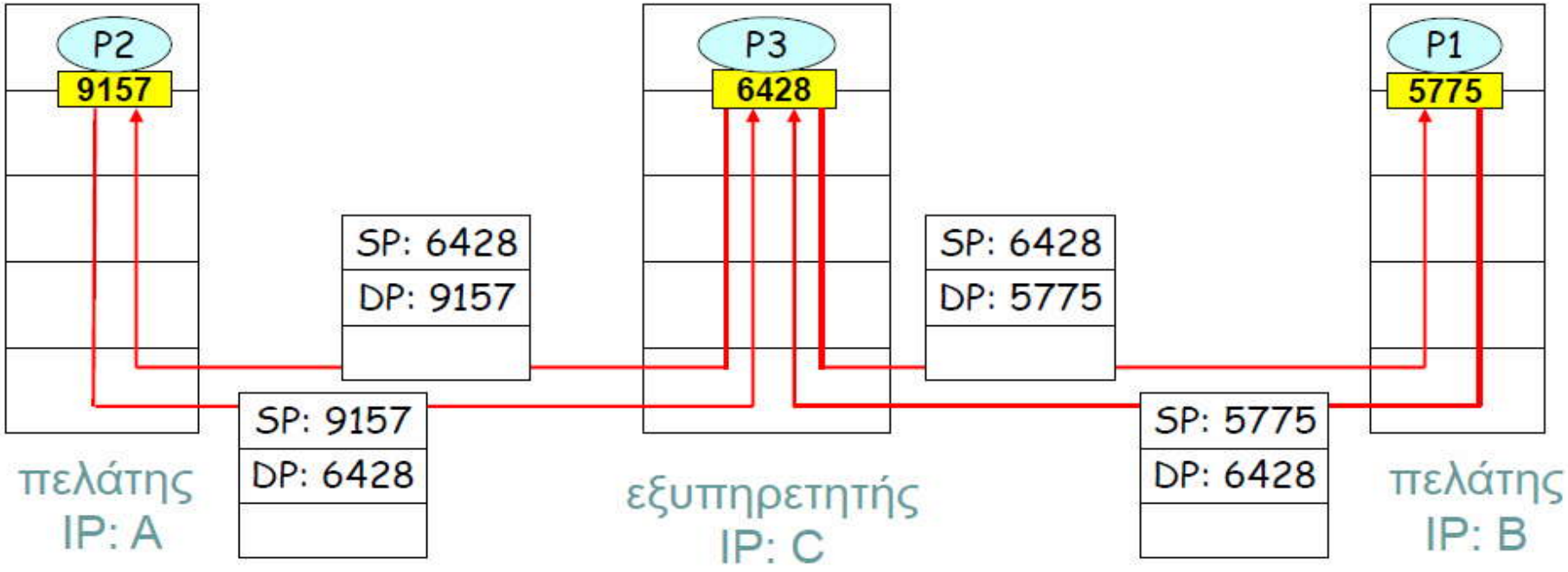
Προκαθορισμένες Θύρες (Well-known ports)

Θύρα	Πρωτόκολλο	Χρήση
21	FTP	Μεταφορά αρχείων
23	Telnet	Απομακρυσμένη πρόσβαση
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Αναζήτηση πληροφοριών για χρήστη
80	HTTP	World Wide Web
110	POP-3	Απομακρυσμένη πρόσβαση e-mail
119	NNTP	USENET news

Αποπολύπλεξη χωρίς σύνδεση – Ι

- Μια υποδοχή UDP προσδιορίζεται από τη δυάδα (2-tuple): (διεύθυνση IP προορισμού, αριθμός θύρας προορισμού)
- Όταν ένας κόμβος λαμβάνει ένα τμήμα UDP:
  - ελέγχει τον αριθμό θύρας προορισμού στο τμήμα
  - κατευθύνει το τμήμα UDP στην υποδοχή με αυτόν τον αριθμό θύρας
- ΑΠ IP συνήθως κατευθύνονται στην ίδια υποδοχή σε εξυπηρετητές χωρίς σύνδεση

Αποπολύπλεξη χωρίς σύνδεση – II

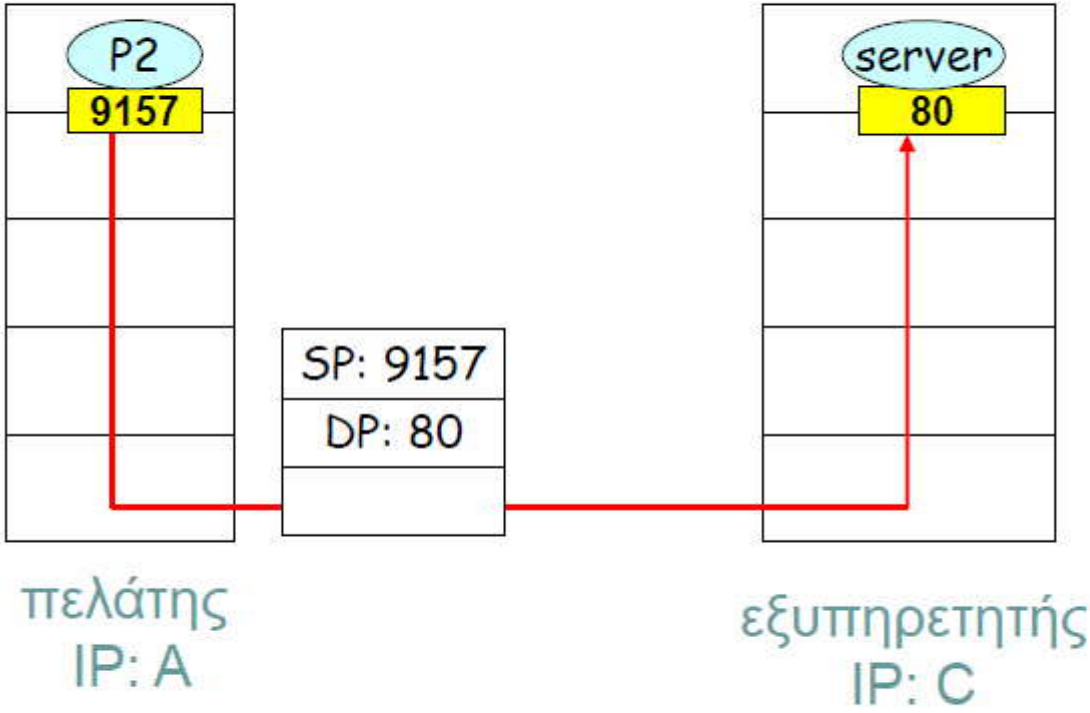


Οι διεργασίες-πελάτες P1 (αριθμός θύρας 5775), P2 (αριθμός θύρας 9157) επικοινωνούν χωρίς σύνδεση με την ίδια διεργασία εξυπηρετητή (P3, με αριθμό θύρας 6428)

Αποπολύπλεξη με σύνδεση – Ι

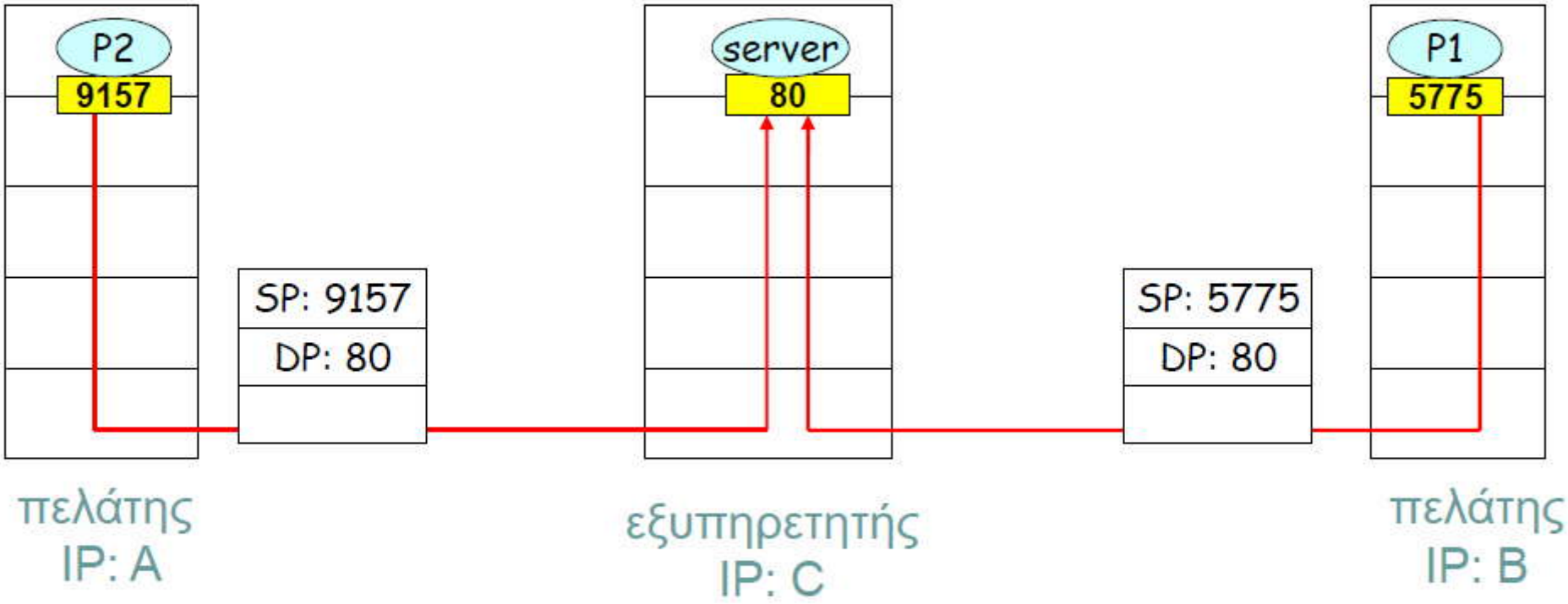
- Μια υποδοχή TCP προσδιορίζεται από μια τετράδα (4-tuple):
  - διεύθυνση IP πηγής
  - αριθμός θύρας πηγής
  - διεύθυνση IP προορισμού
  - αριθμός θύρας προορισμού
- Ο παραλήπτης χρησιμοποιεί την τετράδα για να στείλει το τμήμα στην κατάλληλη υποδοχή
- Ένας εξυπηρετητής μπορεί να υποστηρίζει πολλές ταυτόχρονες υποδοχές TCP:
  - κάθε υποδοχή προσδιορίζεται από τη δική της τετράδα
- Οι εξυπηρετητές Web έχουν διαφορετικές υποδοχές για κάθε πελάτη που συνδέεται
  - το μη επίμονο HTTP θα έχει διαφορετική υποδοχή για κάθε αίτηση

Αποπολύπλεξη με σύνδεση – II



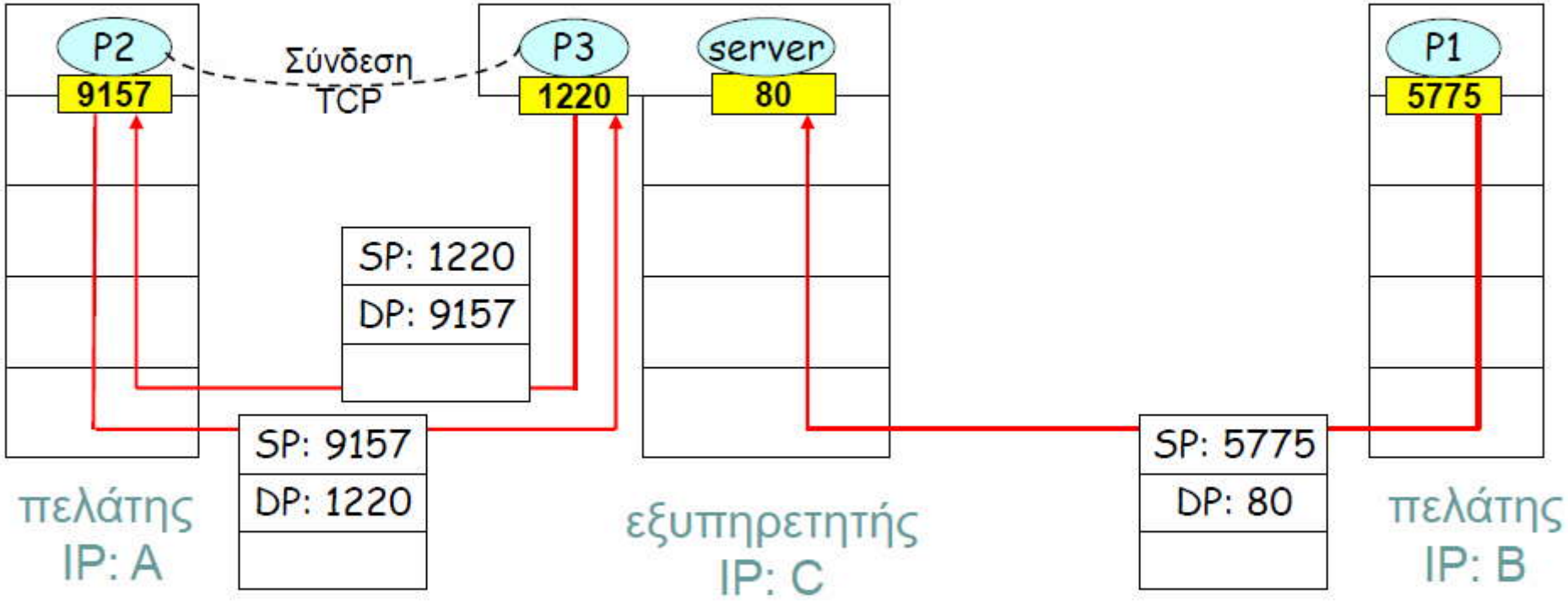
Η διεργασία-πελάτης P2 (αριθμός θύρας 9157) κάνει αίτηση στον εξυπηρετητή που «ακούει» στη θύρα 80 (web server)

Αποπολύπλεξη με σύνδεση – III



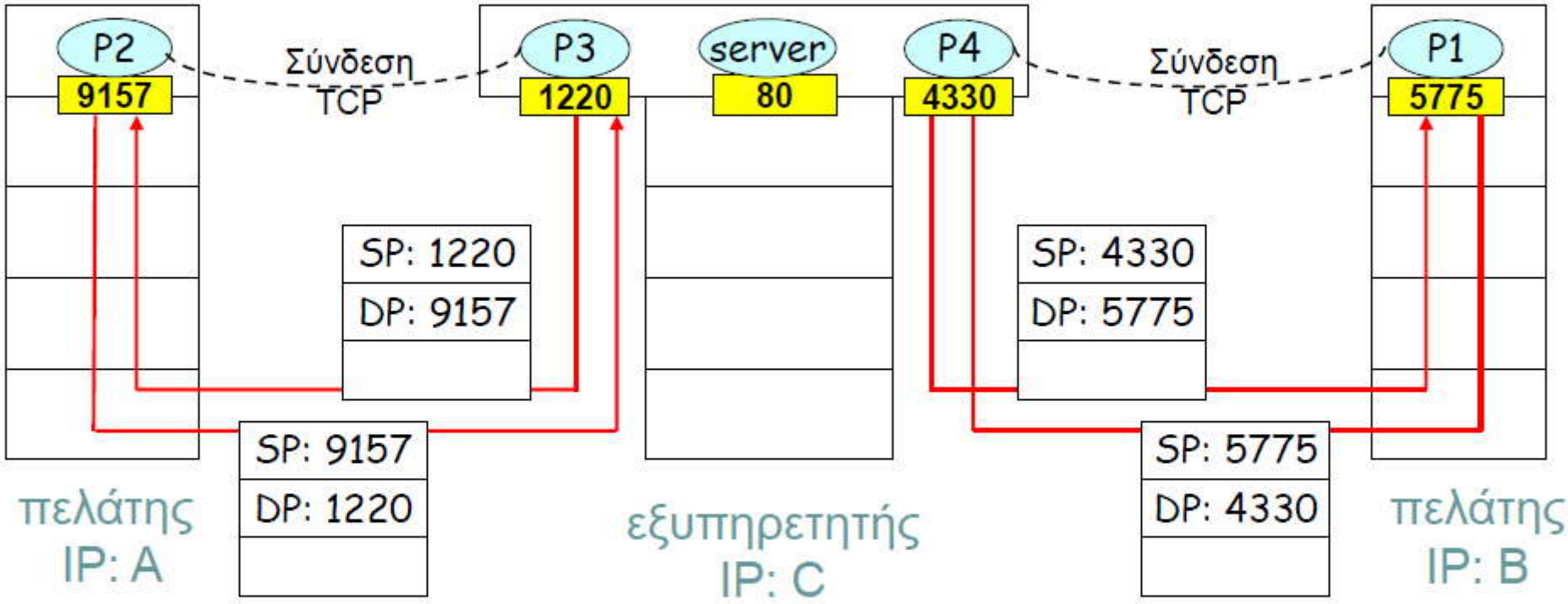
Στη συνέχεια και η διεργασία-πελάτης P1 (αριθμός θύρας 5775) κάνει αίτηση στον εξυπηρετητή που «ακούει» στη θύρα 80

Αποπολύπλεξη με σύνδεση – IV



Ο εξυπηρετητής δημιουργεί αρχικά τη διεργασία παιδί P3 (αριθμός θύρας 1220) η οποία θα αναλάβει την εξυπηρέτηση της διεργασίας P2

Αποπολύπλεξη με σύνδεση – V



Ο εξυπηρετητής δημιουργεί μια δεύτερη διεργασία P4 (αριθμός θύρας 4330). Οι P3 και P4 αναλαμβάνουν την «ταυτόχρονη» εξυπηρέτηση των P2 και P1. Ο εξυπηρετητής αναμένει την άφιξη νέων αιτήσεων

Χώρος Θυρών TCP και UDP

- Η επικεφαλίδα του IP περιέχει το πρωτόκολλο επιπέδου μεταφοράς για το οποίο προορίζεται ένα πακέτο
- Οι χώροι διευθύνσεων τελικού σημείου (θύρες) UDP και TCP είναι διαφορετικοί (μια υποδοχή TCP με αριθμό θύρας X μπορεί να συνυπάρχει στο ίδιο σύστημα με μια υποδοχή UDP με αριθμό θύρας X)
- Τα πακέτα TCP με αριθμό θύρας X θα παραδοθούν σε διαφορετική διεργασία από τα πακέτα UDP με αριθμό θύρας X

Μεταφορά χωρίς σύνδεση: UDP

- Στοιχειώδες πρωτόκολλο μεταφοράς του Internet
- Υπηρεσία "best effort", τα τμήματα UDP μπορεί:
  - να χαθούν
  - να παραδοθούν εκτός σειράς στην εφαρμογή
- χωρίς σύνδεση:
  - χωρίς συμφωνία μεταξύ αποστολέα-παραλήπτη UDP
  - χειρισμός κάθε τμήματος UDP ανεξάρτητα από τα άλλα

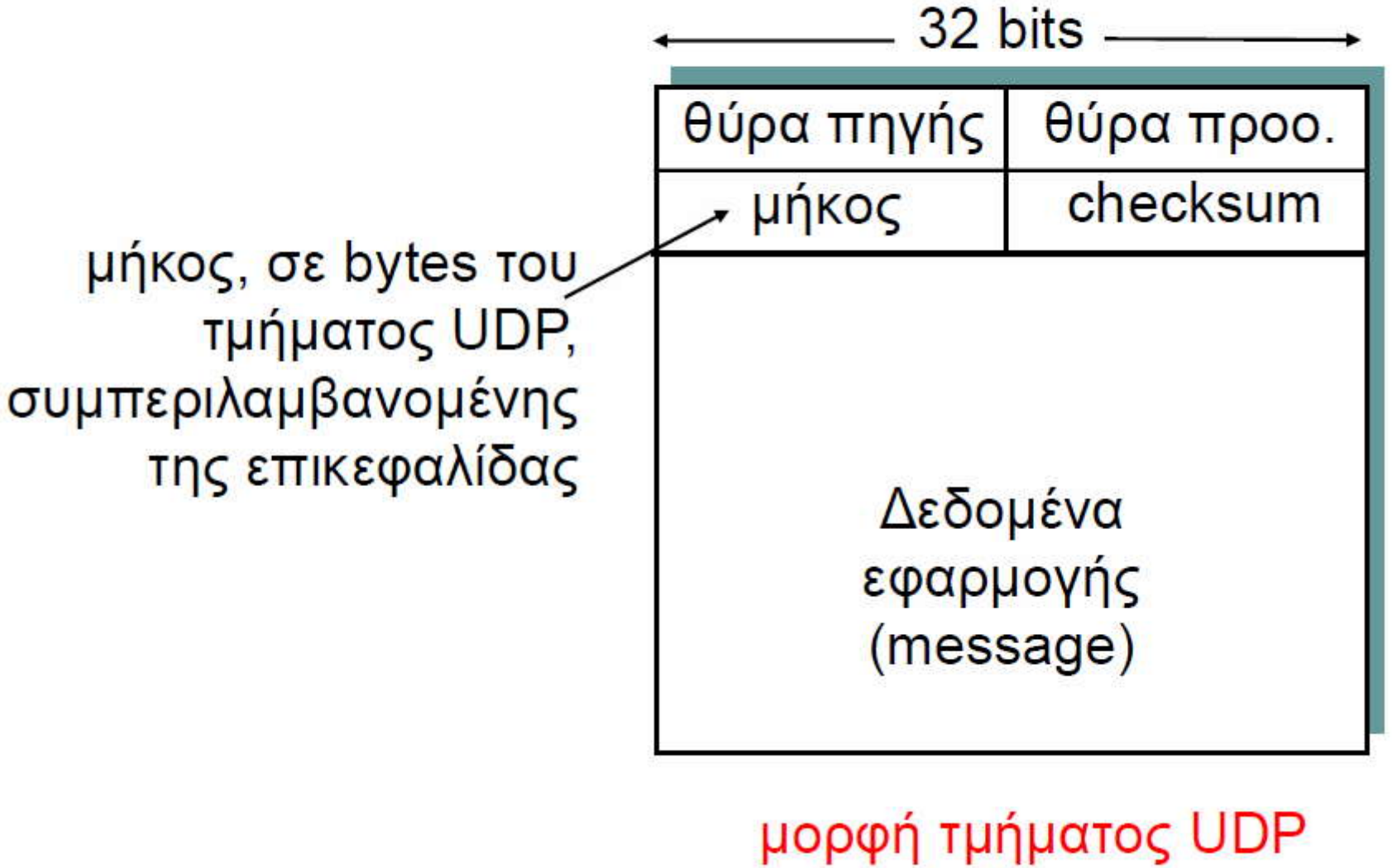
Γιατί υπάρχει το UDP;

- χωρίς δημιουργία σύνδεσης (που δημιουργεί καθυστέρηση)
- απλό: δεν τηρείται κατάσταση σύνδεσης σε αποστολέα και παραλήπτη
- μικρή επικεφαλίδα τμήματος
- χωρίς έλεγχο συμφόρησης: το UDP μπορεί να στέλνει όσο γρήγορα απαιτείται/είναι εφικτό

UDP: User Datagram Protocol [RFC 768]

- Συνήθως χρησιμοποιείται για εφαρμογές ροής πολυμέσων (streaming multimedia apps)
  - ανθεκτικές σε απώλειες (loss tolerant)
  - ευαίσθητες στο ρυθμό μεταφοράς (rate sensitive)
- Άλλες χρήσεις του UDP:
  - DNS
  - SNMP (διαχείριση δικτύου)
- Αξιόπιστη μεταφορά πάνω από UDP: προσθήκη αξιοπιστίας στο επίπεδο εφαρμογής
  - ανάνηψη από λάθη αναλαμβάνει η εφαρμογή!

UDP: User Datagram Protocol



μορφή τμήματος UDP

UDP Checksum – I

Στόχος: ανίχνευση λαθών (π.χ., αλλαγή τιμών bits) στο τμήμα που μεταδίδεται

Αποστολέας:

- αντιμετώπιση των περιεχομένων του τμήματος ως ακολουθία από ακέραιους 16-bit
- checksum: πρόσθεση και συμπλήρωμα ως προς 1 των περιεχομένων
- ο αποστολέας τοποθετεί την τιμή στο πεδίο checksum του UDP

Παραλήπτης:

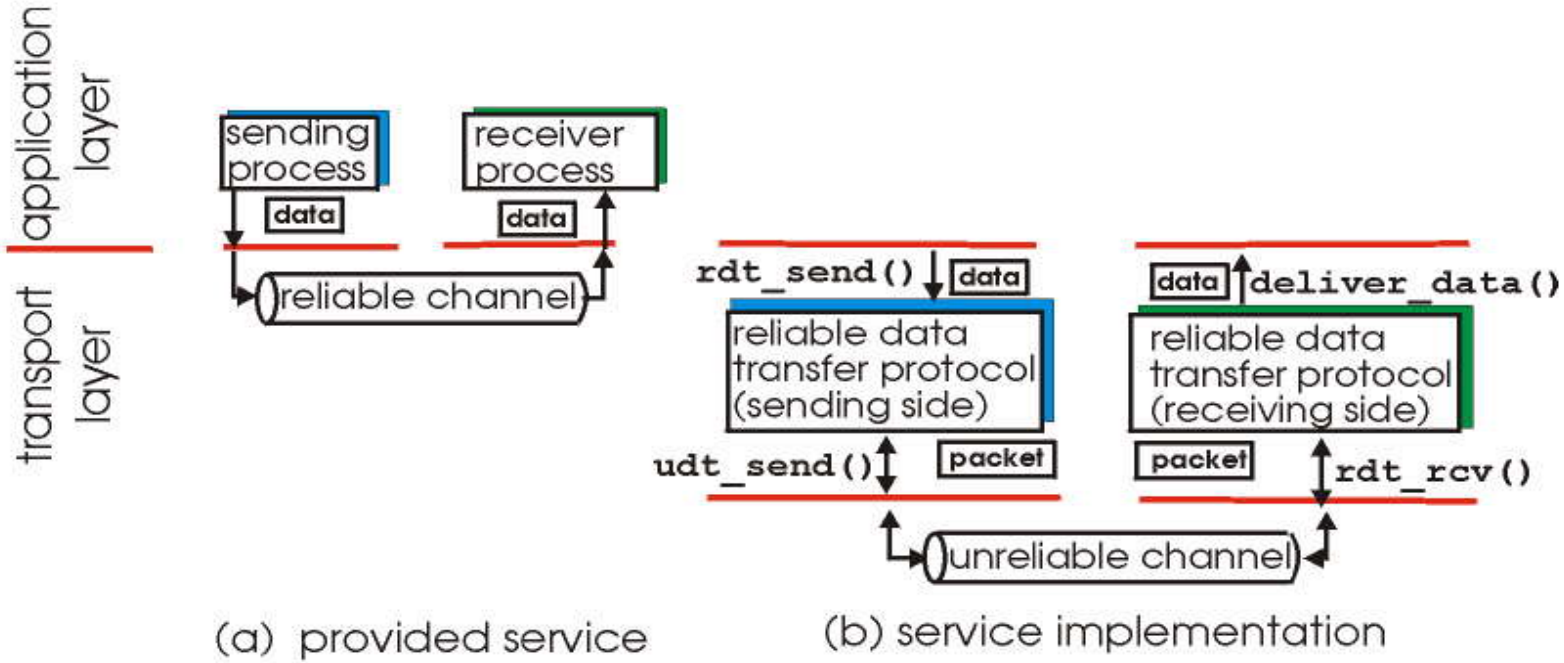
- υπολογισμός του checksum του τμήματος που λήφθηκε
- έλεγχος αν το checksum που υπολογίστηκε ισούται με την τιμή του πεδίου checksum:
  - ΟΧΙ – ανιχνεύθηκε λάθος, απορρίπτεται το τμήμα
  - ΝΑΙ – δεν ανιχνεύθηκε λάθος (ίσως όμως να υπάρχει…)

UDP Checksum – Απόδοση

- Η τεχνική του checksum ανιχνεύει όλα τα λάθη που συμβαίνουν σε περιττό πλήθος bits
- Ανιχνεύει τα περισσότερα λάθη που συμβαίνουν σε άρτιο πλήθος από bits
- Αν ένα ή περισσότερα bits ενός τμήματος είναι κατεστραμμένα και το αντίστοιχο(α) bit(s) της αντίθετης τιμής σε ένα δεύτερο τμήμα είναι επίσης κατεστραμμένα, το άθροισμα αυτών των στηλών δεν θα αλλάξει και ο παραλήπτης δεν θα ανιχνεύσει πρόβλημα

Αρχές της Αξιόπιστης Μεταφοράς Δεδομένων

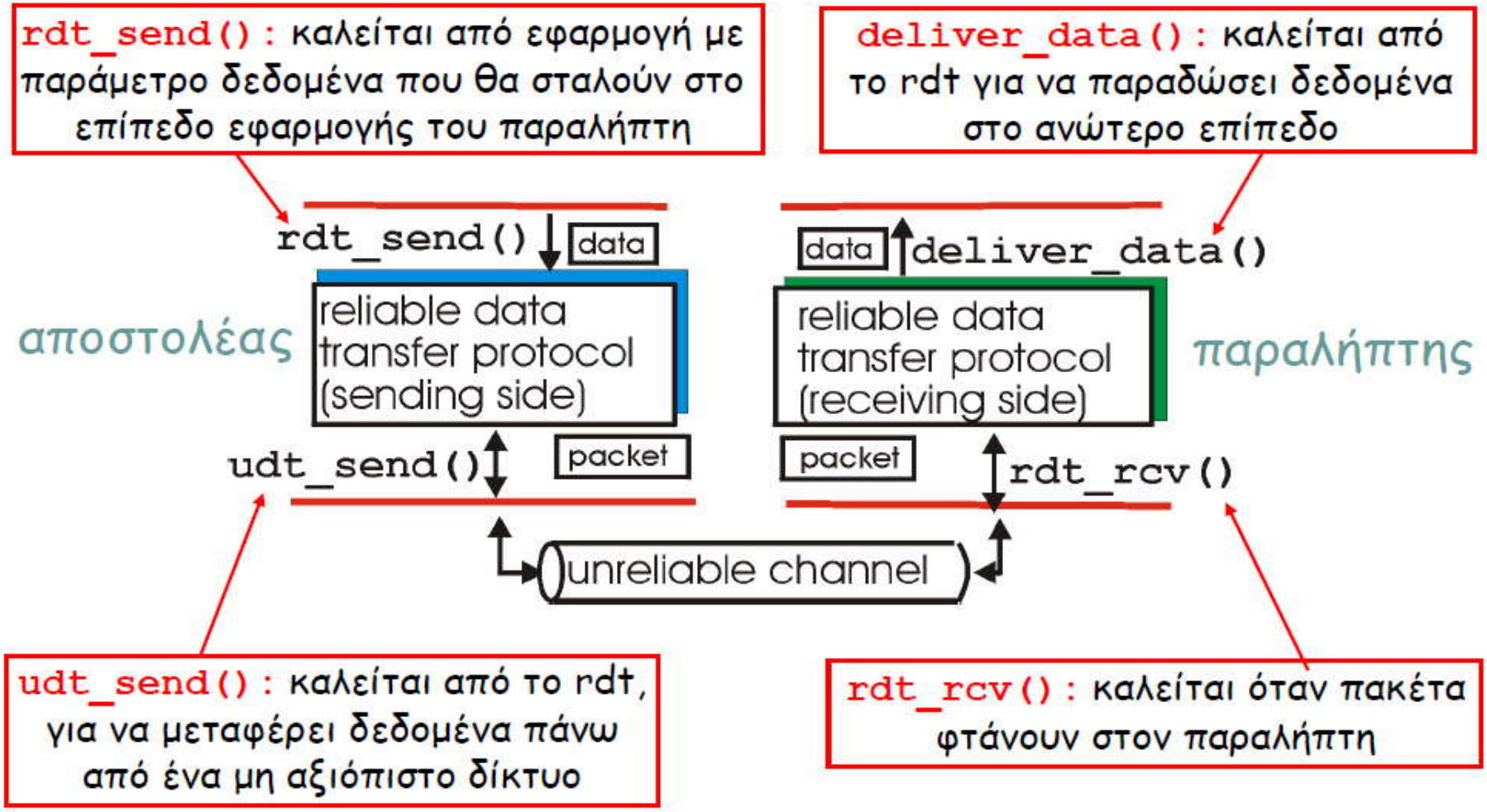
- Σημαντική στα επίπεδα εφαρμογής, μεταφοράς και ζεύξης
- Ένα από τα 10 πιο σημαντικά ζητήματα των δικτύων!



(a) provided service (b) service implementation

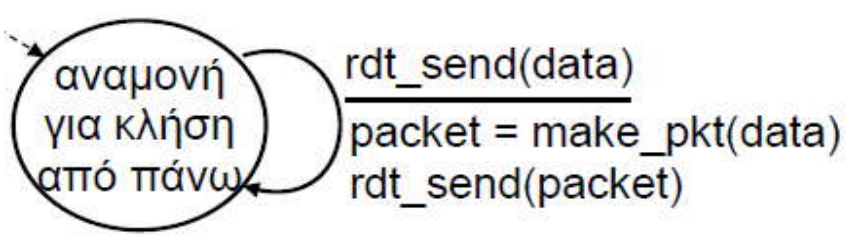
- Τα χαρακτηριστικά του μη αξιόπιστου καναλιού θα καθορίσουν την πολυπλοκότητα του πρωτοκόλλου αξιόπιστης μεταφοράς δεδομένων (reliable data transfer protocol (rdt))

Σενάριο Περιγραφής Αξιόπιστης Μεταφοράς Δεδομένων

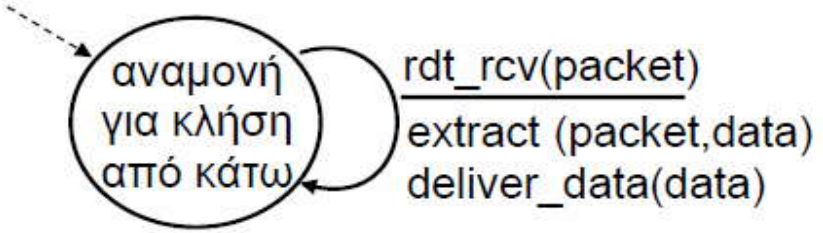


rdt1.0: Αξιόπιστη Μεταφορά πάνω από Αξιόπιστο Κανάλι

- Το κανάλι που χρησιμοποιείται είναι απολύτως αξιόπιστο
  - χωρίς σφάλματα σε bits
  - χωρίς απώλειες πακέτων
- Ξεχωριστές ΜΠΚ για αποστολέα, παραλήπτη:
  - ο αποστολέας στέλνει δεδομένα στο υφιστάμενο κανάλι
  - ο παραλήπτης διαβάζει δεδομένα από το υφιστάμενο κανάλι



αποστολέας



παραλήπτης

rdt2.0: Στα bits μπορεί να συμβούν λάθη στο κανάλι

- Το κανάλι μπορεί να αλλάξει τιμές των bits στο πακέτο
  - θυμηθείτε: το checksum του UDP ανιχνεύει λάθη σε bit
- *Ερώτηση*: πως γίνεται η ανάνηψη από τα λάθη;
  - *acknowledgements (ACKs):* ο παραλήπτης λέει σαφώς στον αποστολέα ότι το πακέτο παραλήφθηκε σωστά
  - *negative acknowledgements (NAKs):* ο παραλήπτης λέει σαφώς στον αποστολέα ότι το πακέτο έχει λάθη
  - ο αποστολέας ξαναστέλνει το πακέτο μόλις λάβει NAK
- Νέοι μηχανισμοί στο rdt2.0 (επιπλέον του rdt1.0):
  - ανίχνευση λαθών
  - ανάδραση από τον παραλήπτη: μηνύματα ελέγχου (ACK,NAK) από παραλήπτη προς αποστολέα

Ανίχνευση και Διόρθωση Σφαλμάτων

- Δύο βασικές στρατηγικές αντιμετώπισης των σφαλμάτων:
  - Ο πρώτος τρόπος είναι να περιλαμβάνεται αρκετή πλεονάζουσα πληροφορία ώστε να μπορεί ο δέκτης να συμπεραίνει ποιος χαρακτήρας μεταδόθηκε
  - Ο άλλος τρόπος είναι να παρέχεται απλά πλεονάζουσα πληροφορία, ώστε να μπορεί ο δέκτης να συμπεράνει ότι συνέβη ένα σφάλμα (αλλά όχι ποιο σφάλμα) και να ζητάει επαναμετάδοση (retransmission)

Διόρθωση Σφαλμάτων

- Διόρθωση: Χρησιμοποιούνται οι κώδικες Hamming (η διόρθωση κατάλληλη για δεδομένα πραγματικού χρόνου)
- Οι κώδικες διόρθωσης σφαλμάτων έχει περισσότερο νόημα να χρησιμοποιούνται σε μονόδρομο (simplex) δίαυλο όπου δεν μπορούν να ζητηθούν επαναμεταδόσεις

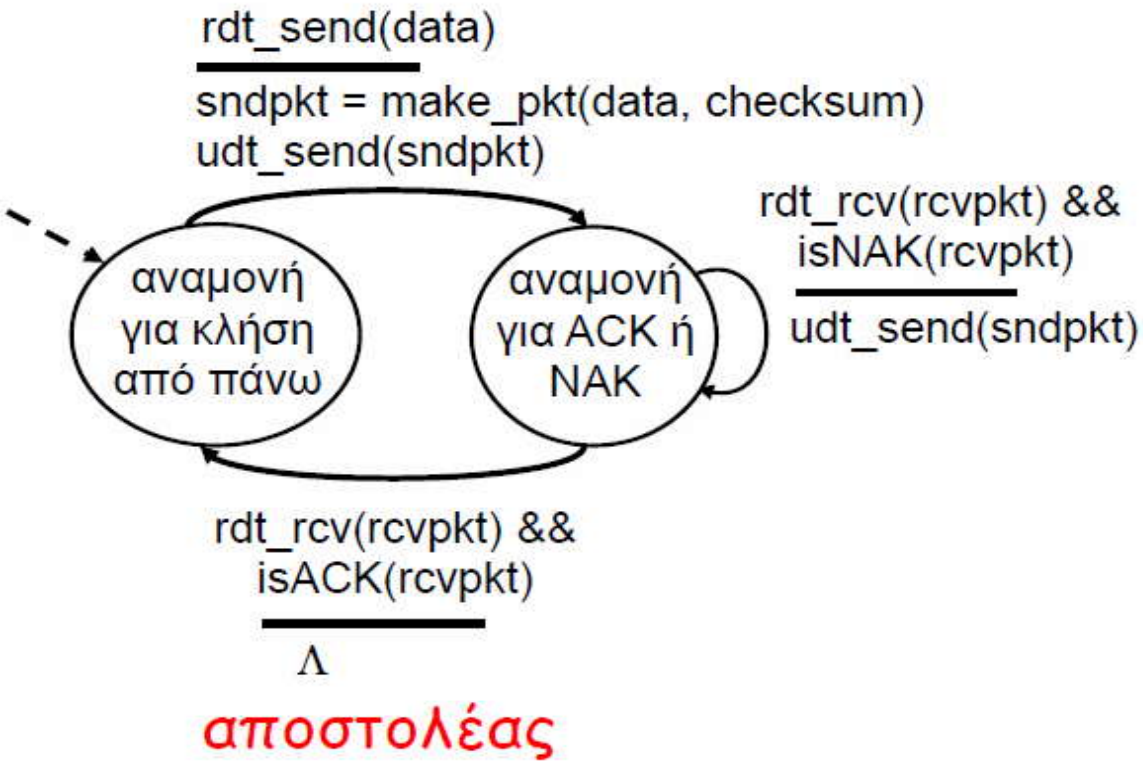
Κώδικες Ανίχνευσης Σφαλμάτων

- Ένα απλό παράδειγμα κώδικα ανίχνευσης σφαλμάτων, είναι αυτό που προστίθεται από το δέκτη ένα bit ισοτιμίας (parity bit) στα δεδομένα που πρόκειται να μεταδοθούν (προκειμένου να είναι άρτιος ή περιττός ο συνολικός αριθμός των 1 σε ένα μήνυμα)

Κώδικες Ανίχνευσης και Διόρθωσης Σφαλμάτων

- Η επαναμετάδοση είναι συνήθως αποδοτικότερη αφού για τη διόρθωση σφαλμάτων απαιτείται μετάδοση μεγαλύτερης ποσότητας πληροφοριών ελέγχου
- Στην πράξη χρησιμοποιείται ευρέως ο πολυωνυμικός κώδικας, ή κυκλικός κώδικας πλεονασμού (Cyclic Redundancy Check – CRC)
- Οι πολυωνυμικοί κώδικες βασίζονται στην αναπαράσταση μιας ακολουθίας bits ως πολυώνυμο με συντελεστές 0 ή 1

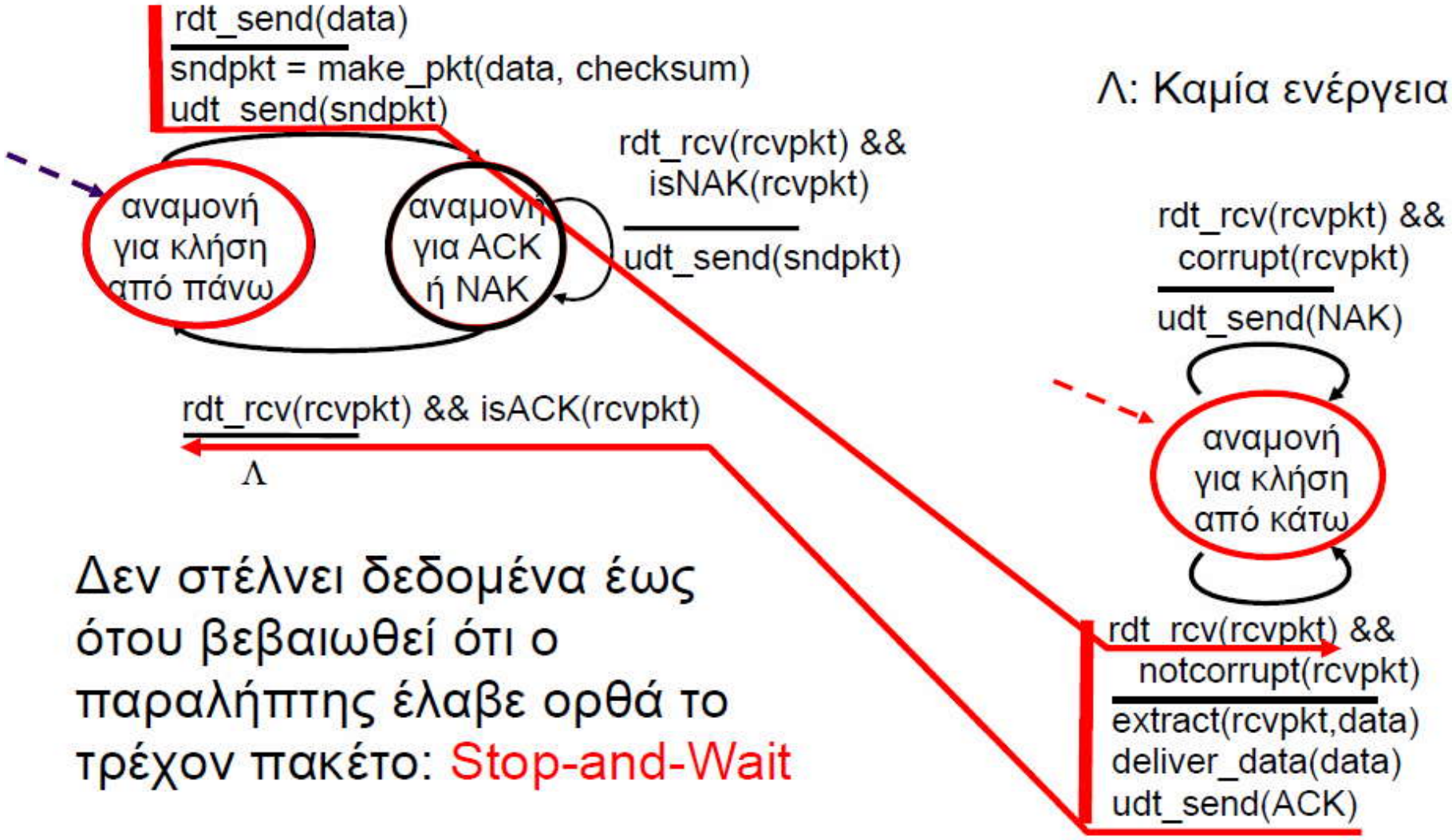
rdt2.0: Προδιαγραφές ΜΠΚ



ACK: επαλήθευση ή αναγνώριση

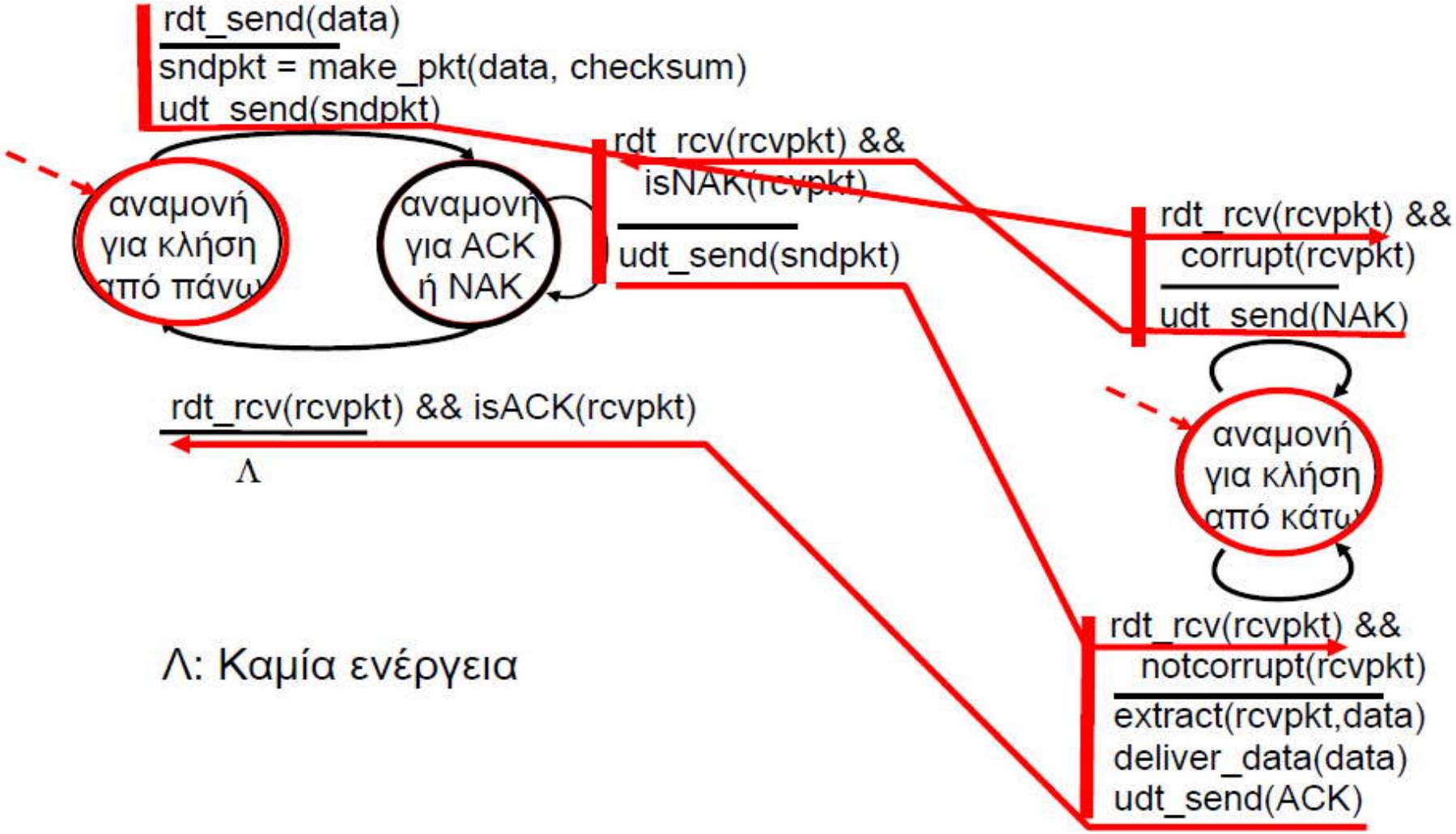
NAK: αρνητική επαλήθευση

rdt2.0: Λειτουργία χωρίς Λάθη



Δεν στέλνει δεδομένα έως ότου βεβαιωθεί ότι ο παραλήπτης έλαβε ορθά το τρέχον πακέτο: Stop-and-Wait

rdt2.0: Λειτουργία με Λάθη



Υπάρχει όμως ένα σοβαρό πρόβλημα…

Τι γίνεται αν αλλοιωθεί το ACK;

- Ο αποστολέας δεν ξέρει τι συνέβη στον παραλήπτη!
- Δεν μπορεί απλά να το ξαναστείλει: πιθανώς θα είναι διπλότυπο (duplicate)

Τι κάνουμε;

- Ο αποστολέας στέλνει ACK/NAK για τα ACK/NAK του παραλήπτη; Τι γίνεται αν αλλοιωθεί το ACK/NAK του αποστολέα;
- επαναμετάδοση, αυτό όμως μπορεί να προκαλέσει μετάδοση ενός πακέτου που έχει ήδη ληφθεί ορθά

Χειρισμός διπλότυπων:

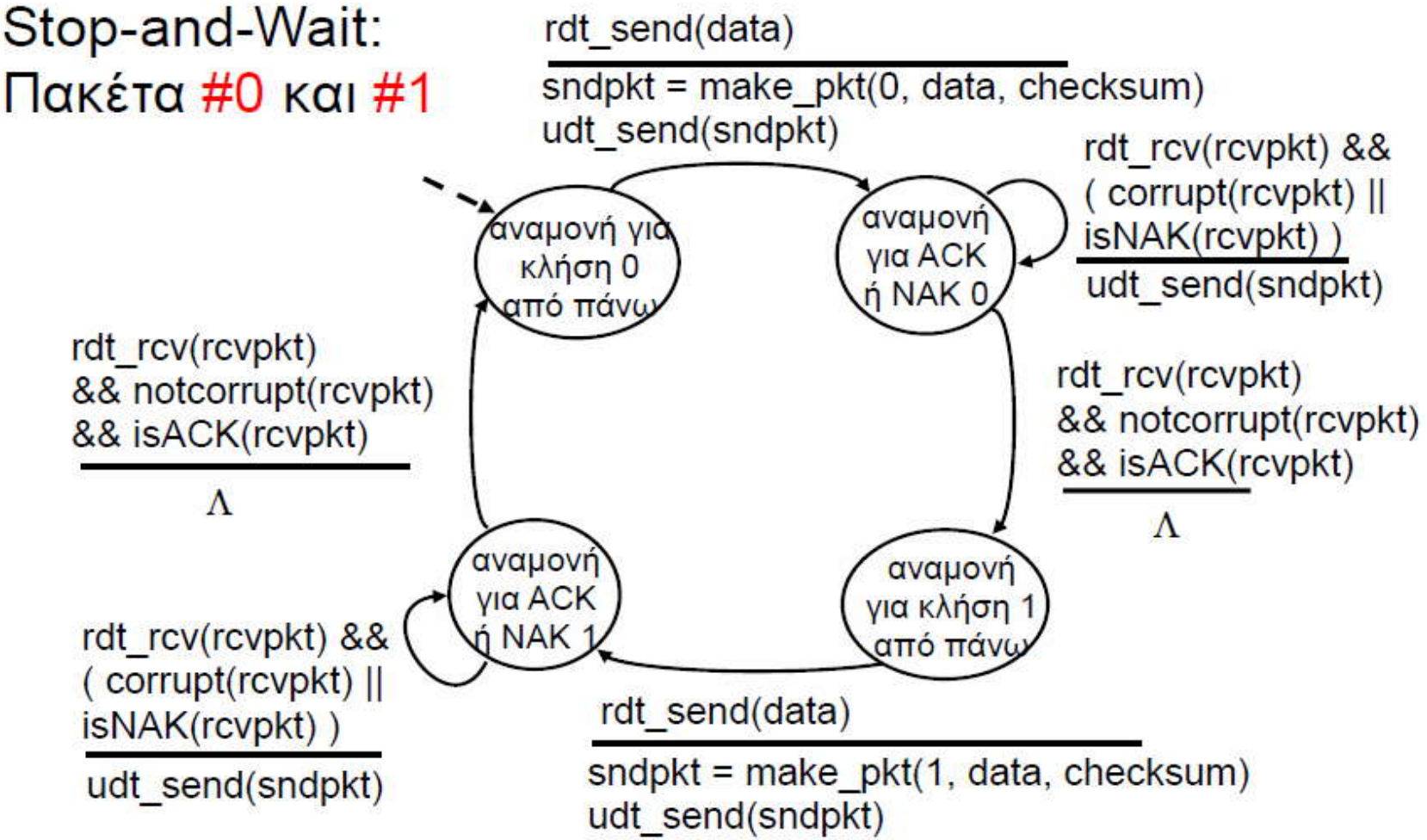
- Ο αποστολέας βάζει αριθμούς ακολουθίας σε κάθε πακέτο
- Ο αποστολέας ξαναστέλνει το τρέχον πακέτο αν το ACK/ NAK είναι εσφαλμένο
- Ο παραλήπτης απορρίπτει (δεν παραδίδει προς τα πάνω) διπλότυπα πακέτα (ίδιο seq #)

παύση και αναμονή

Ο αποστολέας στέλνει ένα πακέτο, στη συνέχεια περιμένει την απόκριση του παραλήπτη

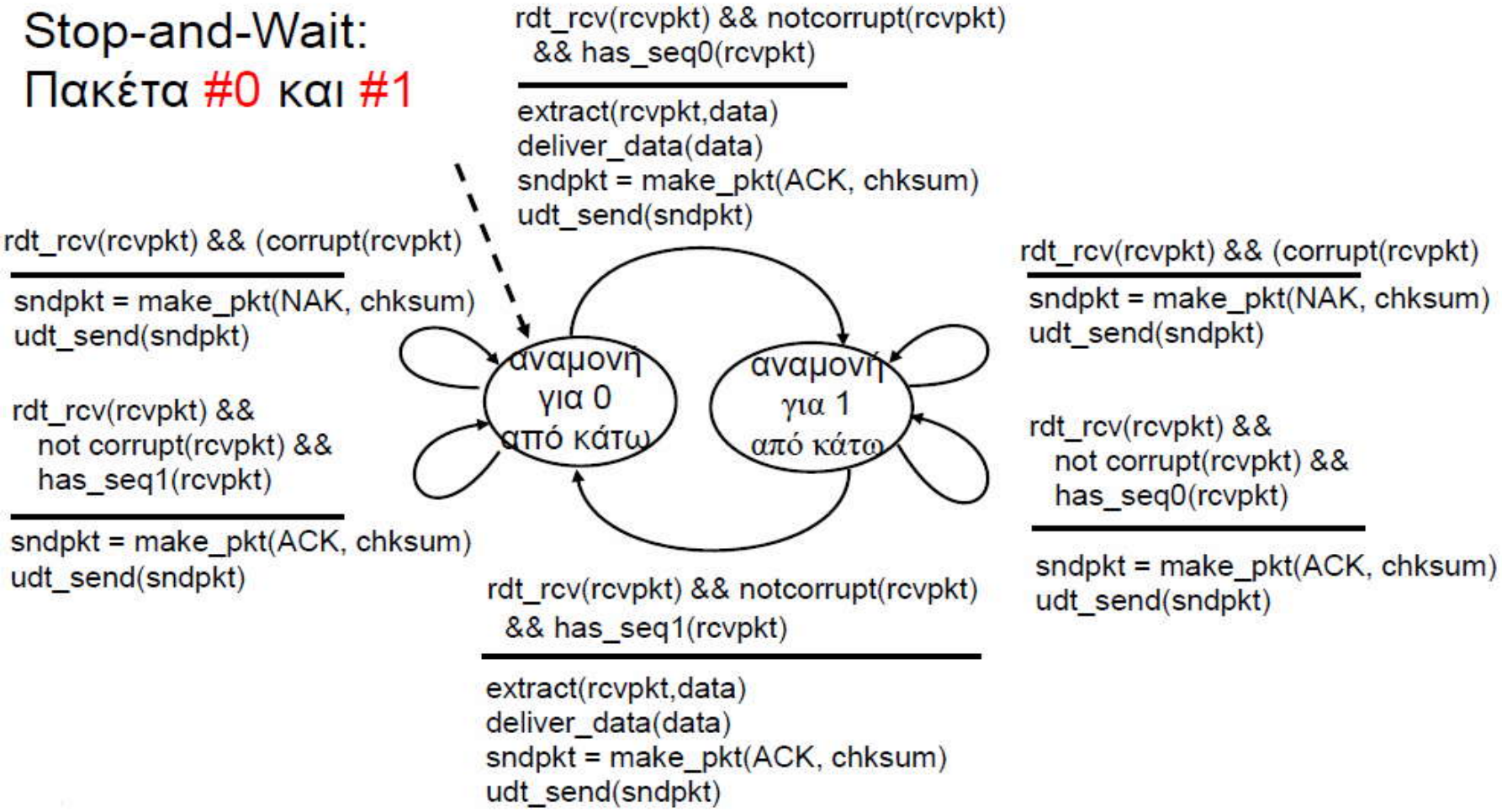
rdt2.1: Αποστολέας με Χειρισμό Προβληματικών ACKs/NAKs

Stop-and-Wait:
Πακέτα #0 και #1



rdt2.1: Παραλήπτης με Χειρισμό Προβληματικών ACKs/NAKs

Stop-and-Wait:
Πακέτα #0 και #1



rdt2.1: Συζήτηση

Αποστολέας:

- Βάζει αριθμό ακολουθίας (sequence number, seq #) στο πακέτο
- Δύο αριθμοί (0,1) αρκούν
- Πρέπει να ελέγχει αν το ACK/NAK που λαμβάνει έχει αλλοιωθεί
- Διπλάσιες καταστάσεις
  - Η κατάσταση πρέπει να «θυμάται» αν το τρέχον πακέτο έχει seq # 0 ή 1

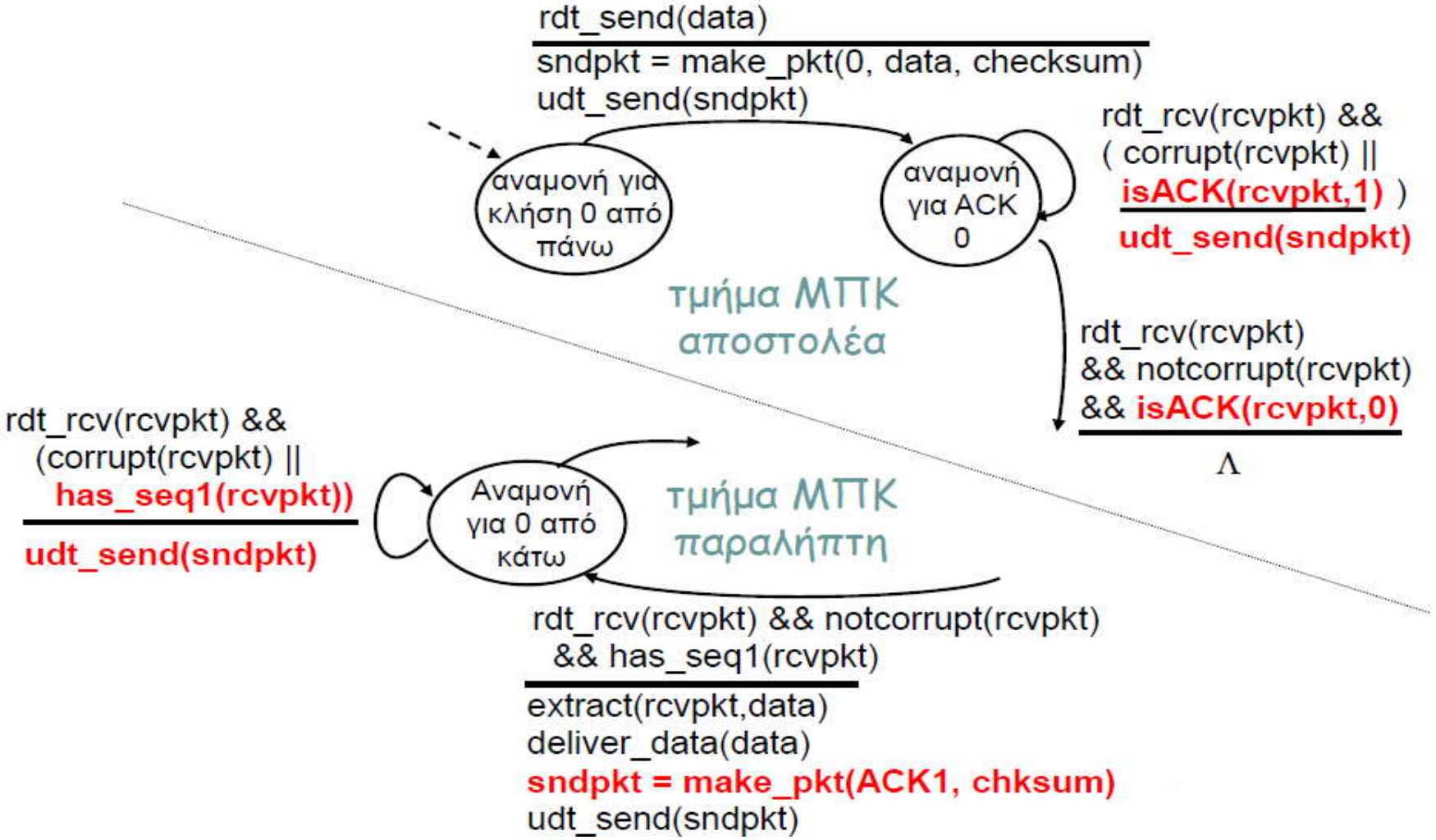
Παραλήπτης:

- Πρέπει να ελέγξει αν το πακέτο που έλαβε είναι διπλότυπο
  - η κατάσταση δείχνει αν το 0 ή το 1 είναι ο αναμενόμενος seq # του πακέτου
- Σημείωση: ο παραλήπτης δεν μπορεί να γνωρίζει αν το τελευταίο του ACK/NAK παραλήφθηκε ορθά από τον αποστολέα

rdt2.2: Ένα Πρωτόκολλο χωρίς NAK

- Ίδια λειτουργικότητα με το rdt2.1, με τη χρήση μόνο ACKs
- Αντί για NAK, ο αποστολέας στέλνει ACK για το τελευταίο πακέτο που έλαβε ορθά
  - ο παραλήπτης πρέπει με σαφήνεια να συμπεριλάβει τον seq # του πακέτου που επαληθεύει
- Όταν διπλότυπο ACK φτάνει στον αποστολέα αυτός κάνει ότι και στην περίπτωση λήψης NAK: ξαναστέλνει το τρέχον πακέτο

rdt2.2: Τμήματα Αποστολέα και Παραλήπτη



rdt3.0: Κανάλια με λάθη και απώλειες

Νέα υπόθεση: το κανάλι μπορεί και να χάσει πακέτα (δεδομένα ή ACKs)

- checksum, seq #, ACKs, επαναμεταδόσεις βοηθούν, δεν είναι όμως αρκετά

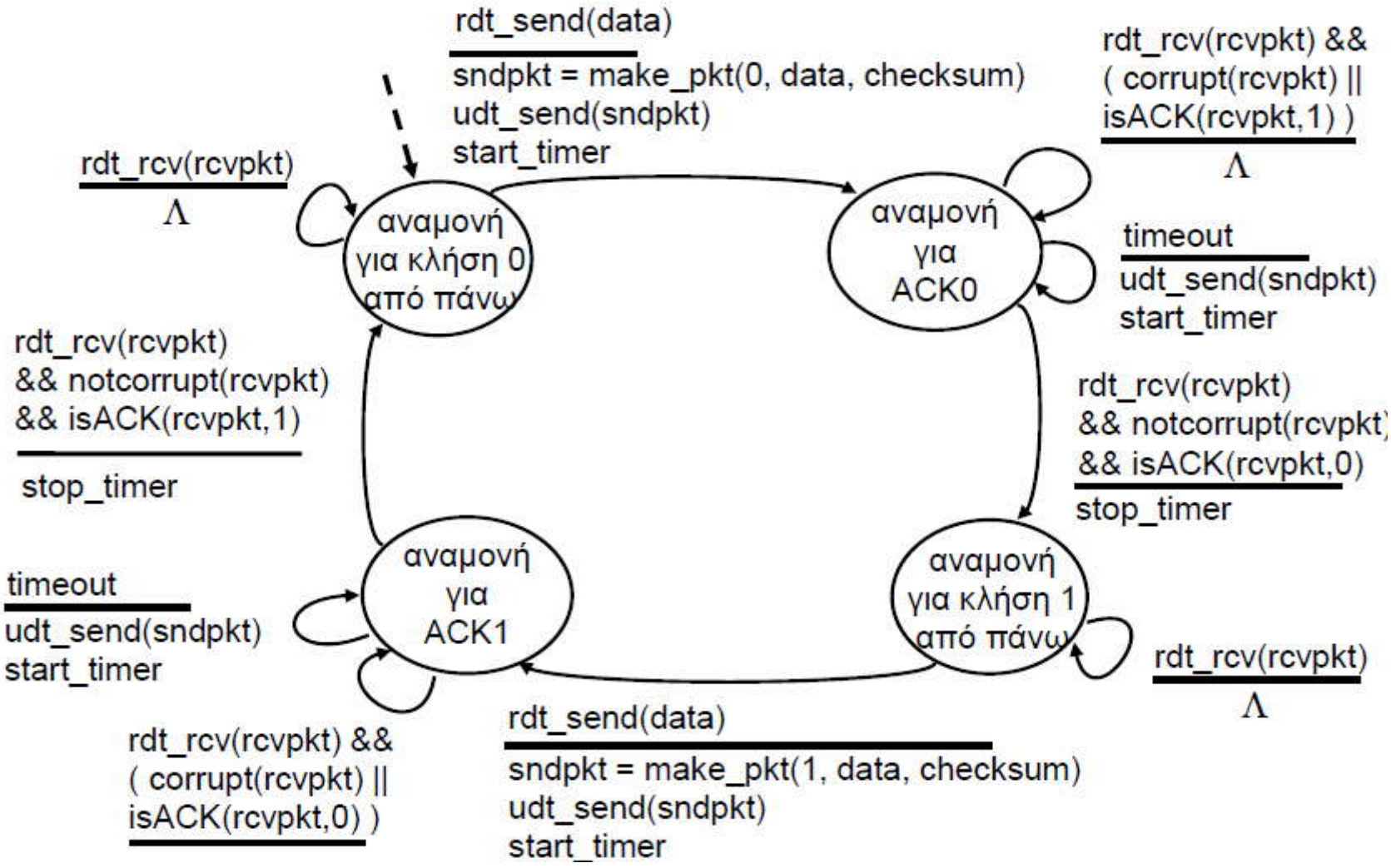
Ερ: Τι συμβαίνει όταν έχουμε απώλεια;

- Ο αποστολέας βεβαιώνεται ότι χάθηκαν δεδομένα ή ACK, και τα στέλνει ξανά

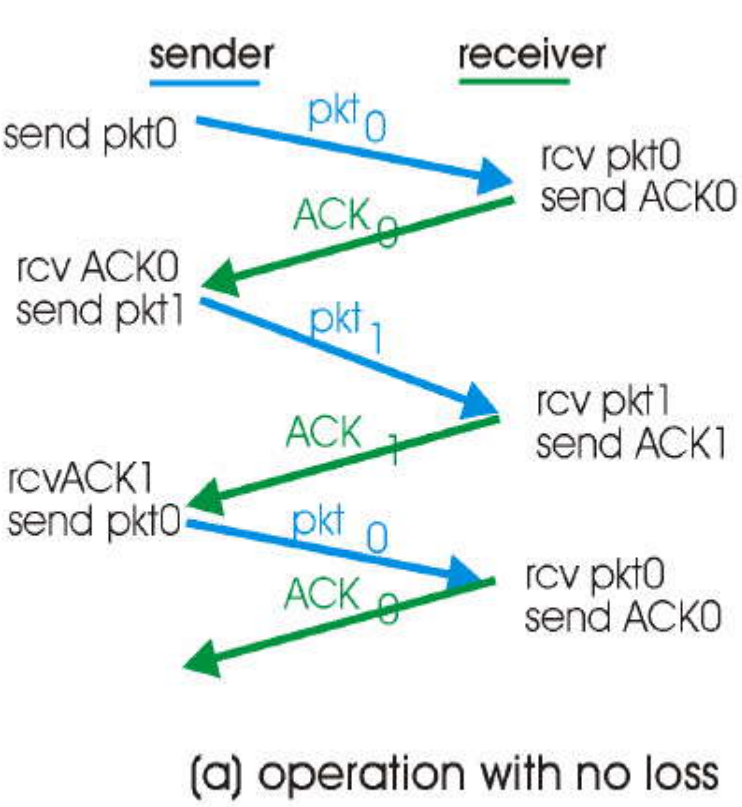
Προσέγγιση: ο αποστολέας περιμένει «λογικό» χρονικό διάστημα για ACK

- Μεταδίδει ξανά αν δεν το λάβει
- Αν το πακέτο (ή το ACK) απλά καθυστέρησαν:
  - το πακέτο θα είναι διπλότυπο, έχουμε όμως seq #!
  - ο παραλήπτης πρέπει να προσδιορίσει τον seq # του πακέτου που επαληθεύει
- Απαιτεί ύπαρξη μετρητή

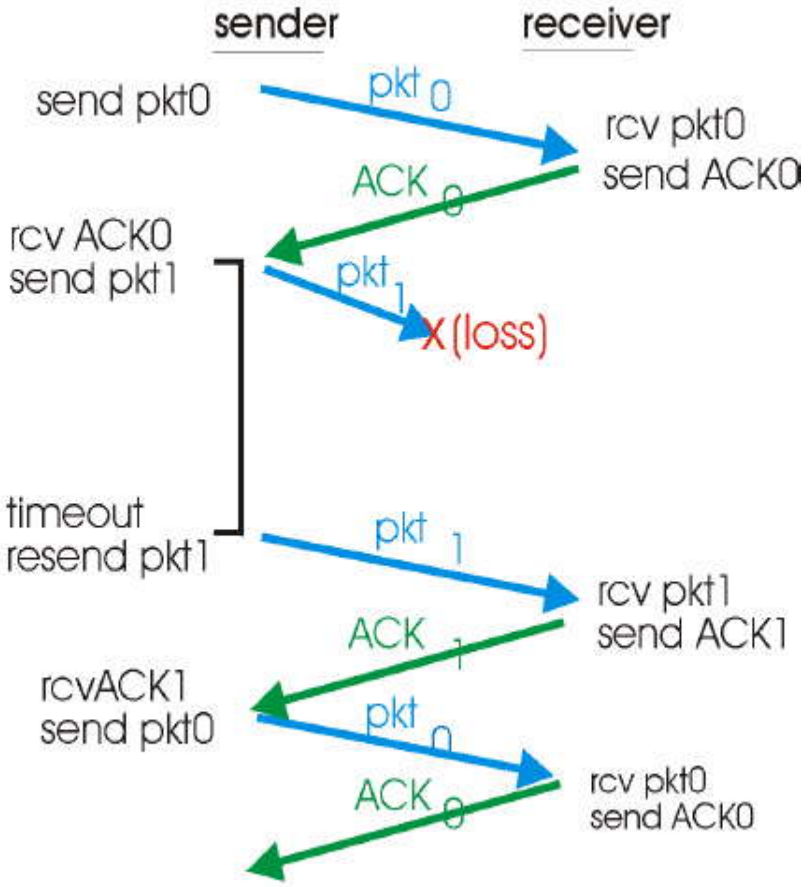
rdt3.0: Αποστολέας



rdt3.0: Σε Δράση – I

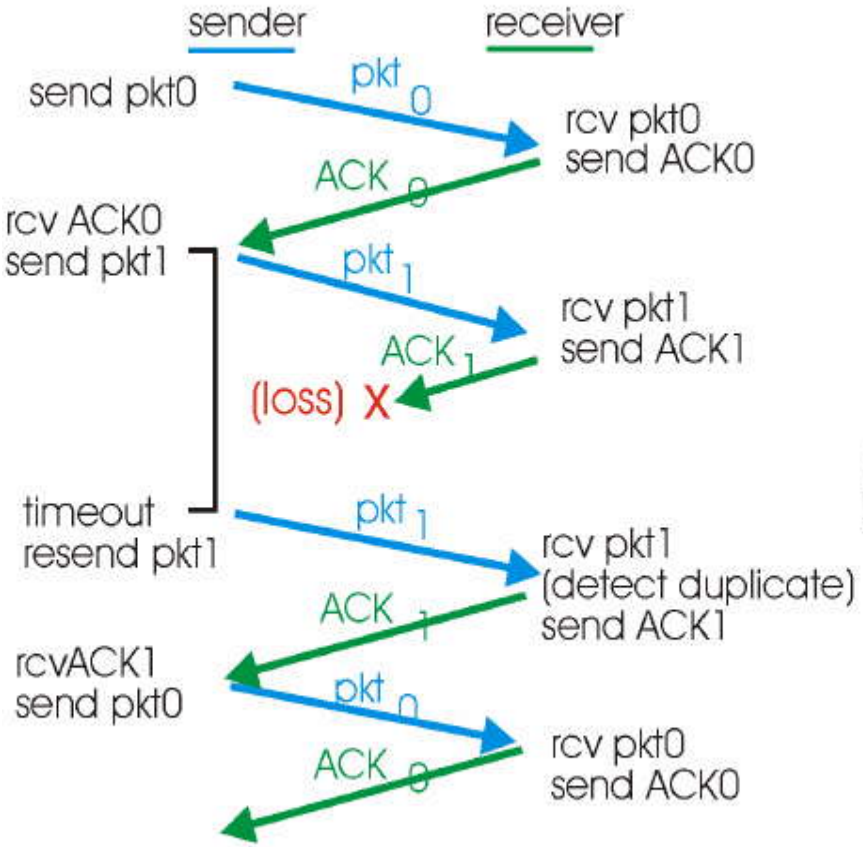


(a) operation with no loss

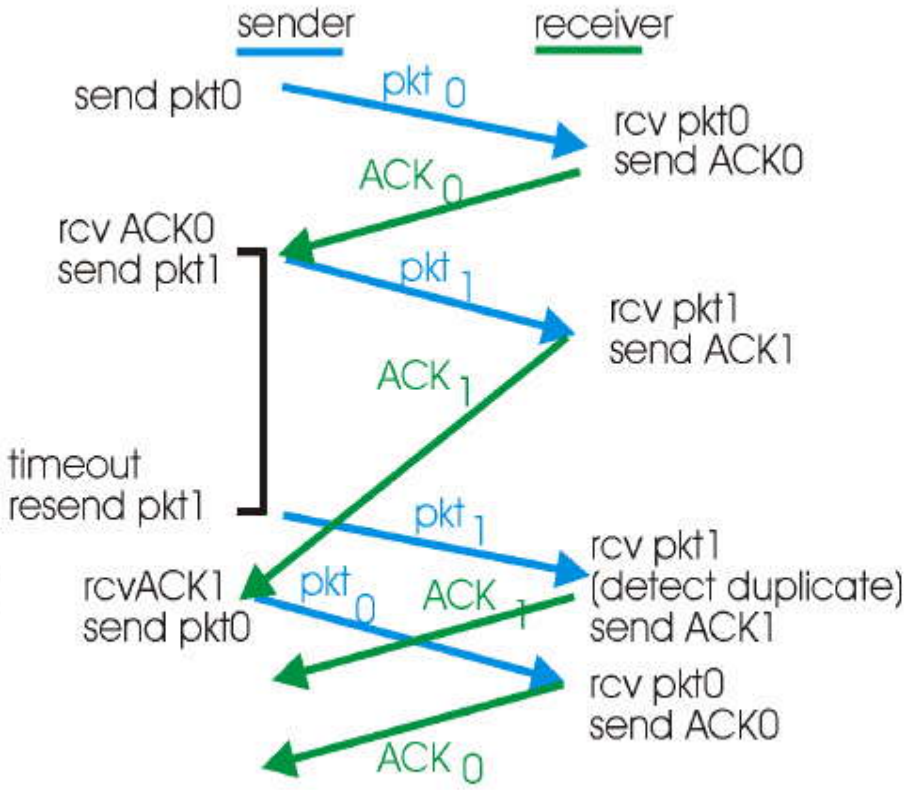


(b) lost packet

rdt3.0: Σε Δράση – II



(c) lost ACK



(d) premature timeout

Απόδοση του rdt3.0

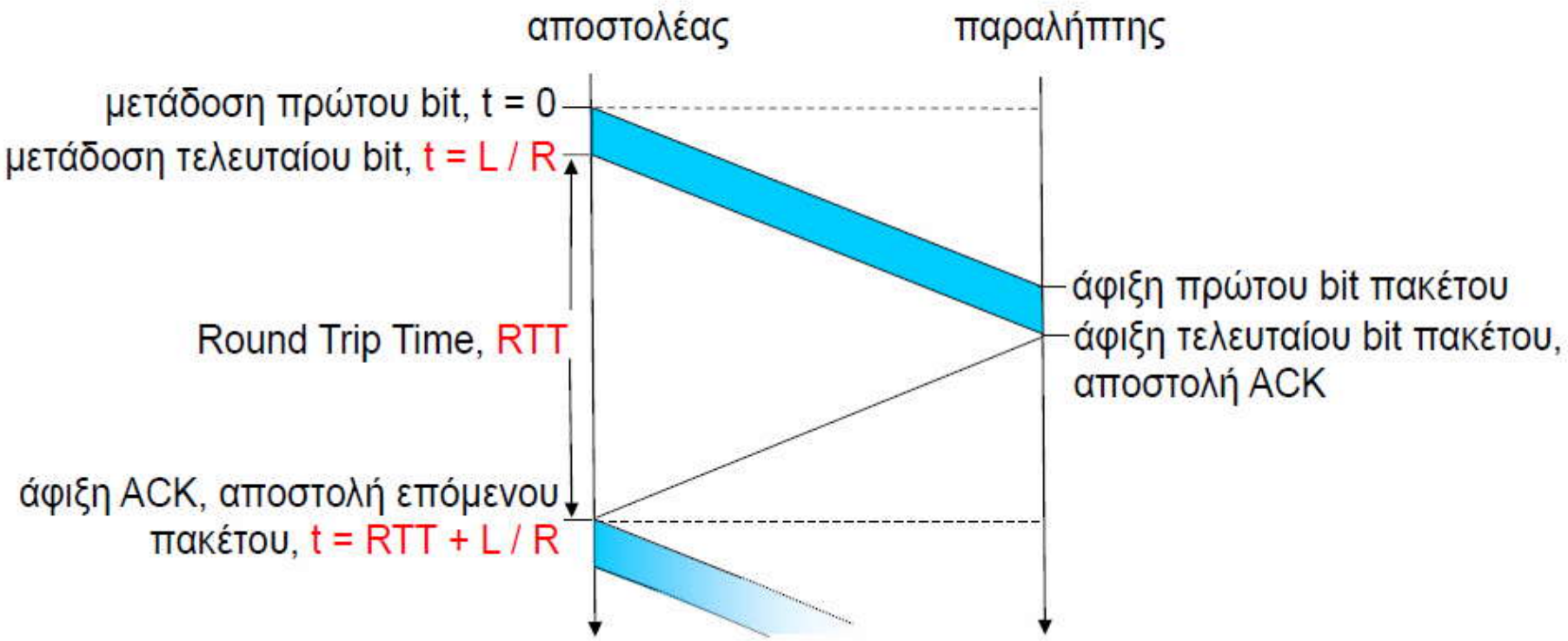
- Το rdt3.0 δουλεύει, χωρίς όμως καλή απόδοση
- Παράδειγμα: κύκλωμα 1 Gbps, καθυστέρηση διάδοσης 15 ms (από άκρο σε άκρο), πακέτο 1KB:

$$T_{transmit} = \frac{L\ (\text{πακέτο σε bits})}{R\ (\text{ρυθμός μετάδοσης, bps})} = \frac{8kb/pkt}{10^{**}9\ b/sec} = 8\ microsec$$

$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

- U_{sender}: χρησιμοποίηση – κλάσμα του χρόνου που ο αποστολέας στέλνει ένα πακέτο
- 1KB pkt κάθε 30 msec -> 33KB/sec(!) ωφέλιμο σε κύκλωμα 1 Gbps
- Το πρωτόκολλο περιορίζει τη χρήση των πόρων του δικτύου!

rdt3.0: Λειτουργία Stop-and-Wait



Πρωτόκολλα Συνεχούς Διοχέτευσης (Pipelined Protocols)

- Τι γίνεται αν επιτρέψουμε στον πομπό να μεταδώσει *n* πακέτα πριν να έρθει η επαλήθευση του πρώτου πακέτου;
- Αυτό καλείται συνεχής διοχέτευση (pipelining) και αυξάνει τη χρησιμοποίηση του δικτύου
- Τι γίνεται αν καταστραφεί ένα πακέτο και όλα τα άλλα (προηγούμενα και επόμενα) φτάσουν ορθά; Δύο λύσεις:
  - Οπισθοδρόμηση κατά N (Go-Back-N)
  - Επιλεκτική Επανάληψη (Selective Repeat)

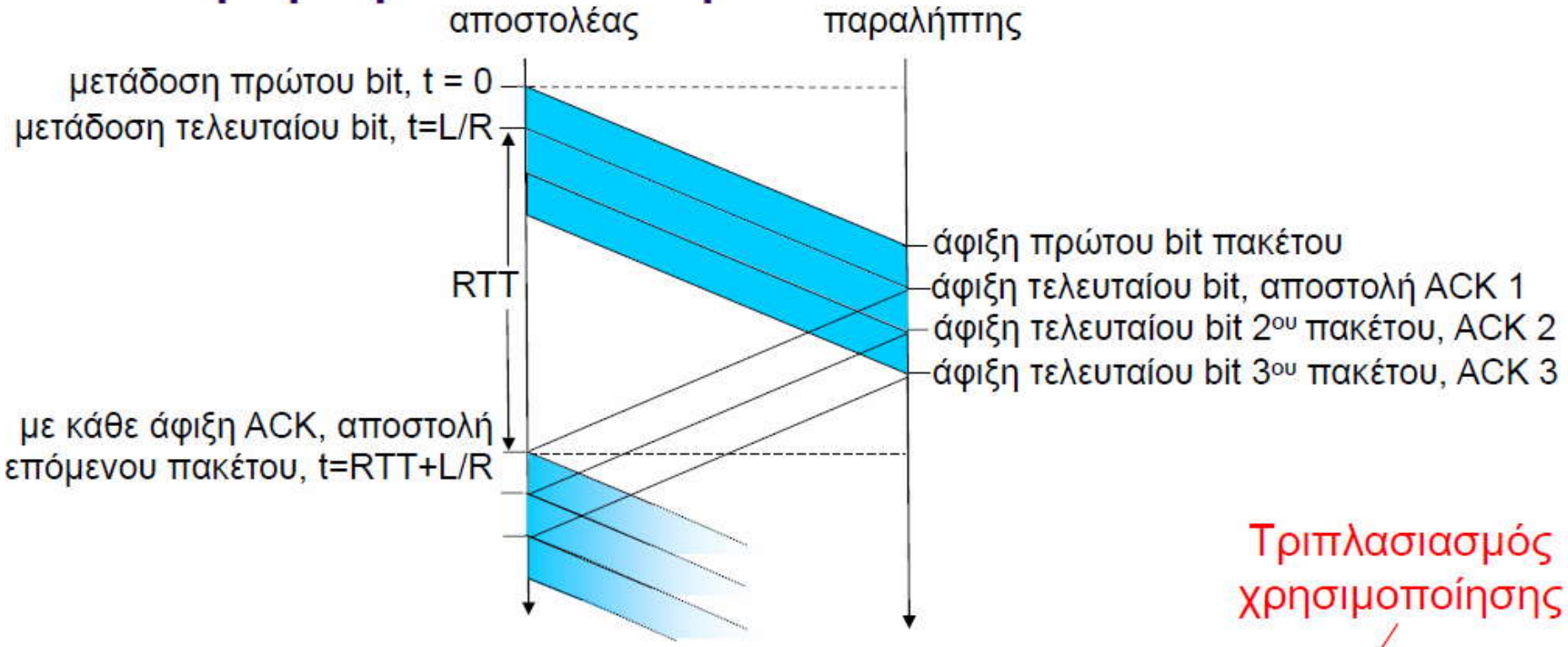
Πρωτόκολλα Συνεχούς Διοχέτευσης – Ι

- Ζητήματα: εφόσον επιτρέπεται η μετάδοση περισσότερων του ενός πακέτων πριν τη λήψη επαληθεύσεων:
  - θα πρέπει να μεγαλώσει το εύρος των αριθμών ακολουθίας (σε αντίθεση με τη λειτουργία stop-and-wait) που τοποθετούνται στις επικεφαλίδες των πακέτων (έστω *k* bits για τον αριθμό ακολουθίας)
  - αποστολέας και παραλήπτης θα πρέπει να έχουν δυνατότητα προσωρινής αποθήκευσης (buffering)

Πρωτόκολλα Συνεχούς Διοχέτευσης – II

- Ο αποστολέας υπολογίζει ένα σύνολο αυξόντων αριθμών που τοποθετεί στα πακέτα που είναι σε θέση να στείλει (παράθυρο αποστολής)
- Παρομοίως, ο παραλήπτης έχει ένα παράθυρο λήψης με τα πακέτα που μπορεί να δεχθεί
- Μην ξεχνάτε: τα πακέτα θα πρέπει να παραδίδονται στην εφαρμογή με την ίδια σειρά με την οποία εστάλησαν

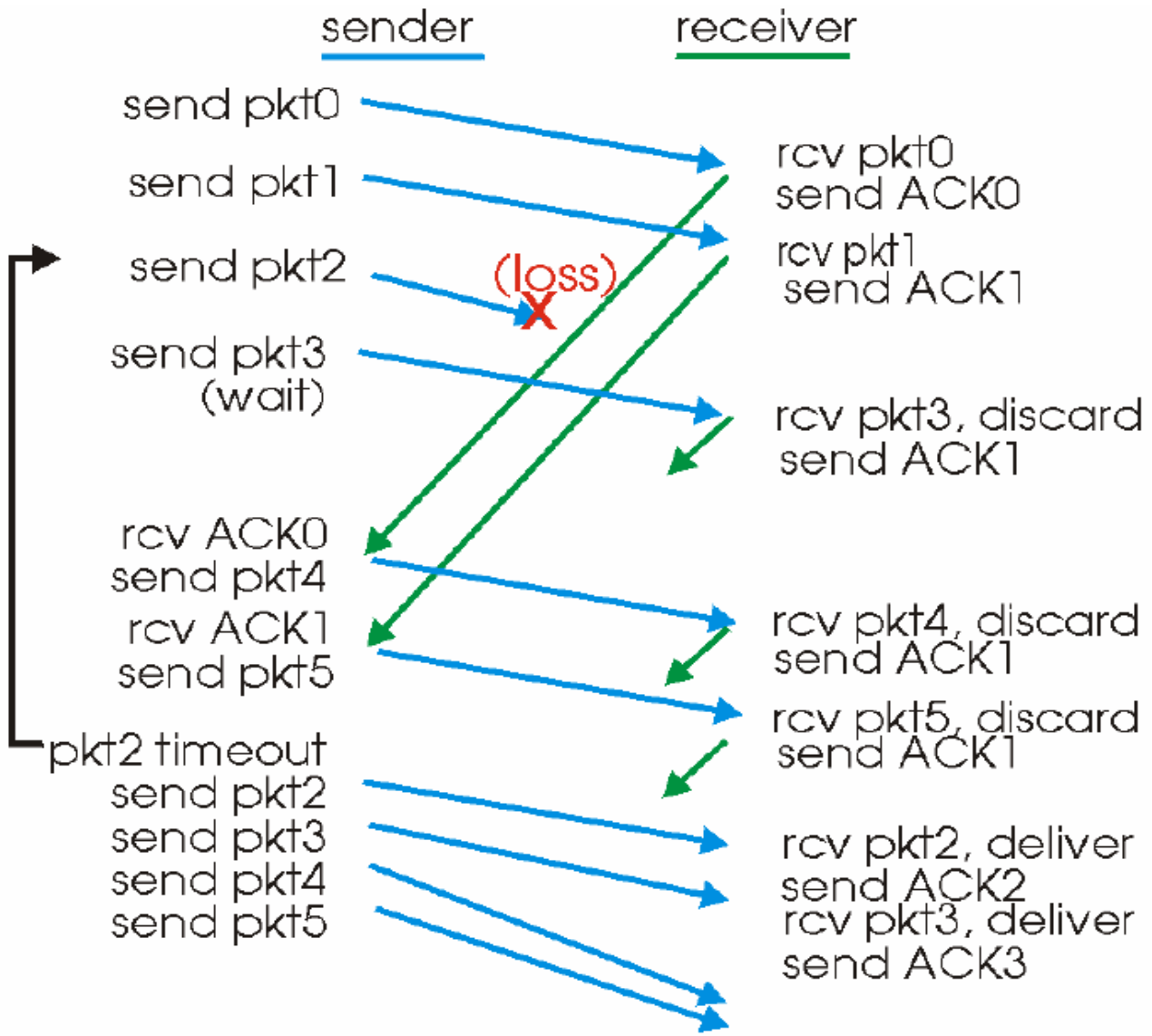
Η Συνεχής Διοχέτευση Αυξάνει τη Χρησιμοποίηση του Δικτύου



Τριπλασιασμός χρησιμοποίησης

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Go-Back-N


sender
receiver
send pkt0
send pkt1
send pkt2
(loss)
send pkt3
(wait)
rcv ACK0
send pkt4
rcv ACK1
send pkt5
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Επιλεκτική Επανάληψη (Selective Repeat) – I

- Ο παραλήπτης επαληθεύει ανεξάρτητα το κάθε πακέτο που έχει λάβει ορθά
  - αποθηκεύει προσωρινά τα πακέτα, προκειμένου να τα παραδώσει με τη σειρά στο ανώτερο επίπεδο
- Ο αποστολέας ξαναστέλνει μόνο τα πακέτα για τα οποία δεν έχει λάβει ACK
  - μετρητής για κάθε πακέτο που δεν έχει επαληθευτεί
- Τι καθορίζει το παράθυρο του αποστολέα;
  - Ν συνεχόμενους seq #
  - το πλήθος των πακέτων που στέλνονται χωρίς ACK

Επιλεκτική Επανάληψη – ΙΙ

αποστολέας

Δεδομένα από ανώτερο επίπεδο:

- Αν ο επόμενος διαθέσιμος seq # είναι εντός παραθύρου, στείλε το πακέτο

timeout(n):

- Ξαναστείλε το πακέτο n, επανεκκίνηση μετρητή

Το ACK(n) βρίσκεται στο `[sendbase, sendbase+N]`:

- Το πακέτο n παρελήφθη ορθά
- Αν το n είναι ο μικρότερος seq # που δεν έχει επαληθευτεί, προχώρησε τη βάση του παραθύρου στον επόμενο seq # που δεν έχει επαληθευτεί

παραλήπτης

Το πακέτο n στο `[rcvbase, rcvbase+N-1]`

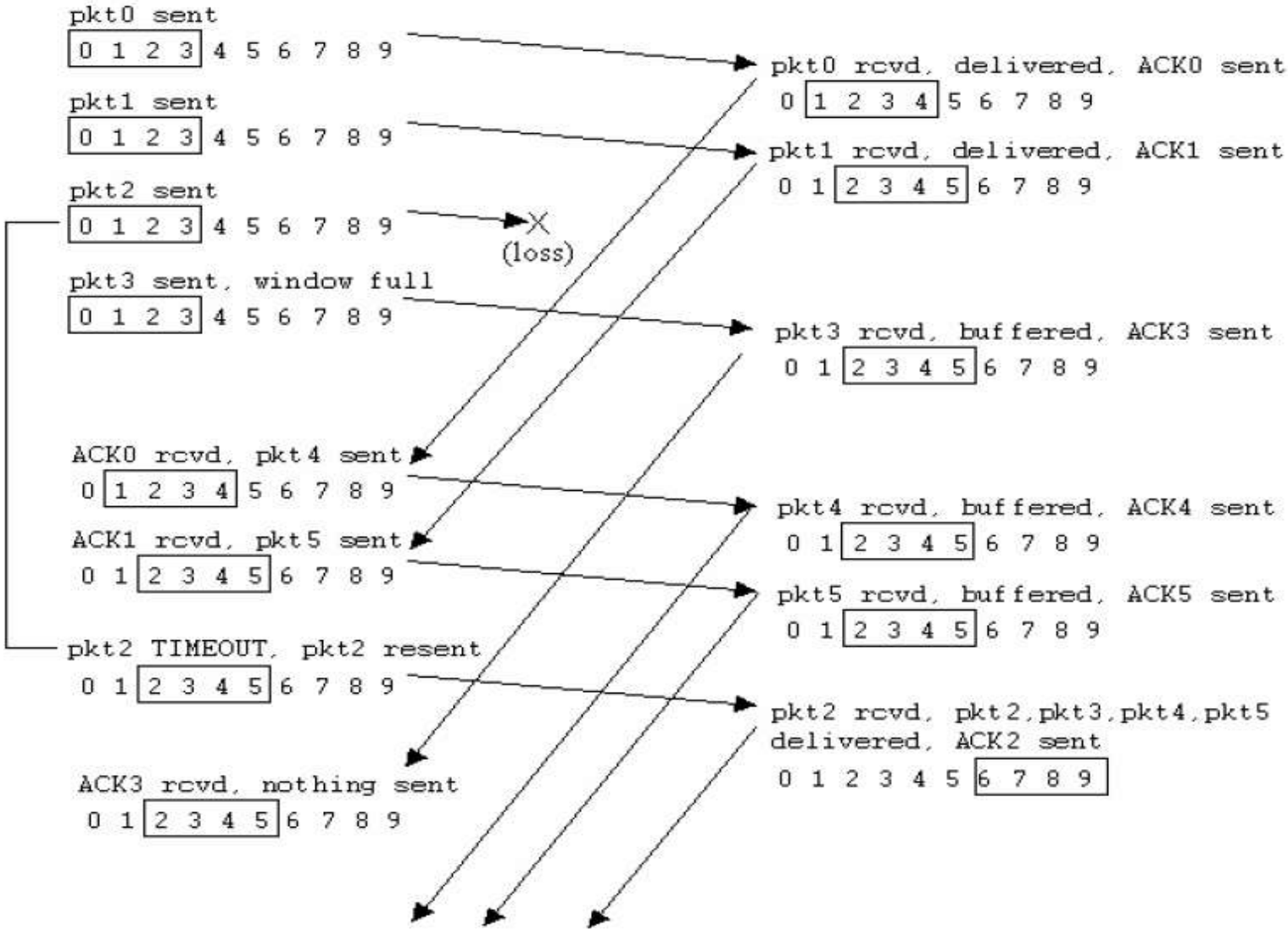
- στείλε ACK(n)
- Εκτός σειράς: αποθήκευση
- Σε σειρά: παράδοση (επίσης και όσα είχαν αποθηκευτεί και είναι σε σειρά), προχώρησε το παράθυρο στο επόμενο πακέτο που δεν έχει παραληφθεί

Το πακέτο n στο `[rcvbase-N, rcvbase-1]`

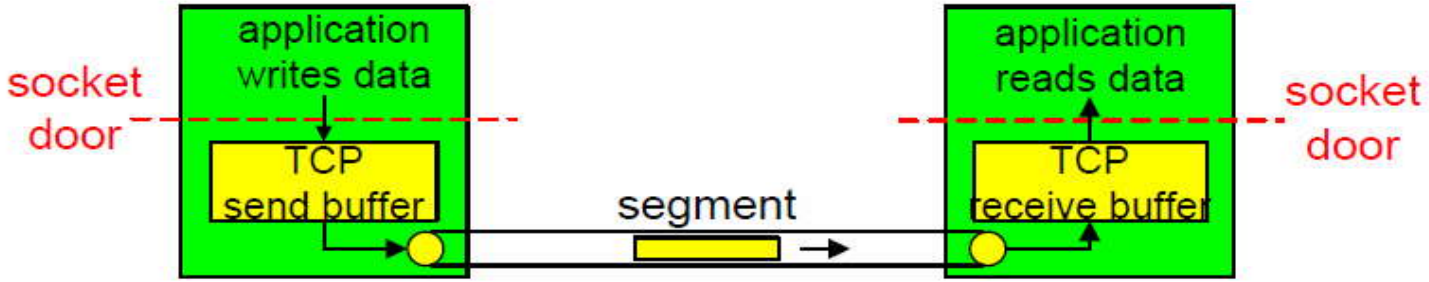
- ACK(n)

Αλλιώς, αγνόησε το πακέτο

Επιλεκτική Επανάληψη – III



Γενική Επισκόπηση του TCP – I
(RFCs: 793, 1122, 1323, 2018, 2581)



- Σημείο-προς-σημείο (point-to-point):
  - ένας αποστολέας, ένας παραλήπτης
- Αξιόπιστη, με τη σειρά ροή δεδομένων (reliable, in-order byte steam):
  - χωρίς "όρια στα μηνύματα"
- Συνεχής διοχέτευση (pipelined):
  - ο έλεγχος συμφόρησης και ροής του TCP καθορίζουν το μέγεθος του παραθύρου
- Buffers αποστολής και λήψης δεδομένων

Γενική Επισκόπηση του TCP – II

- Πλήρως αμφίδρομα δεδομένα (full duplex data):
  - πλήρως αμφίδρομη ροή δεδομένων στην ίδια σύνδεση
  - MSS: Μέγιστο μέγεθος τμήματος (maximum segment size)
- Με σύνδεση (connection-oriented):
  - χειραψία (handshaking): ανταλλαγή μηνυμάτων ελέγχου: αρχικοποιεί την κατάσταση αποστολέα και παραλήπτη πριν την ανταλλαγή δεδομένων
- Με έλεγχο ροής (flow controlled):
  - ο αποστολέας δεν θα κατακλύσει τον παραλήπτη

Διαχείριση Σύνδεσης στο TCP – I

- Θυμηθείτε: αποστολέας και παραλήπτης TCP, δημιουργούν "σύνδεση" πριν την ανταλλαγή τμημάτων δεδομένων
- Αρχικοποίηση μεταβλητών TCP:
  - αριθμοί ακολουθίας (seq#s)
  - buffers, πληροφορία ελέγχου ροής (π.χ. **RcvWindow**)
- *client:* ξεκινάει τη σύνδεση
- *server:* έρχεται σε επαφή μαζί του ο πελάτης

Διαχείριση Σύνδεσης στο TCP – II

Τριπλή Χειραψία (Three way handshake):

Βήμα 1: ο πελάτης στέλνει τμήμα SYN στον εξυπηρετητή

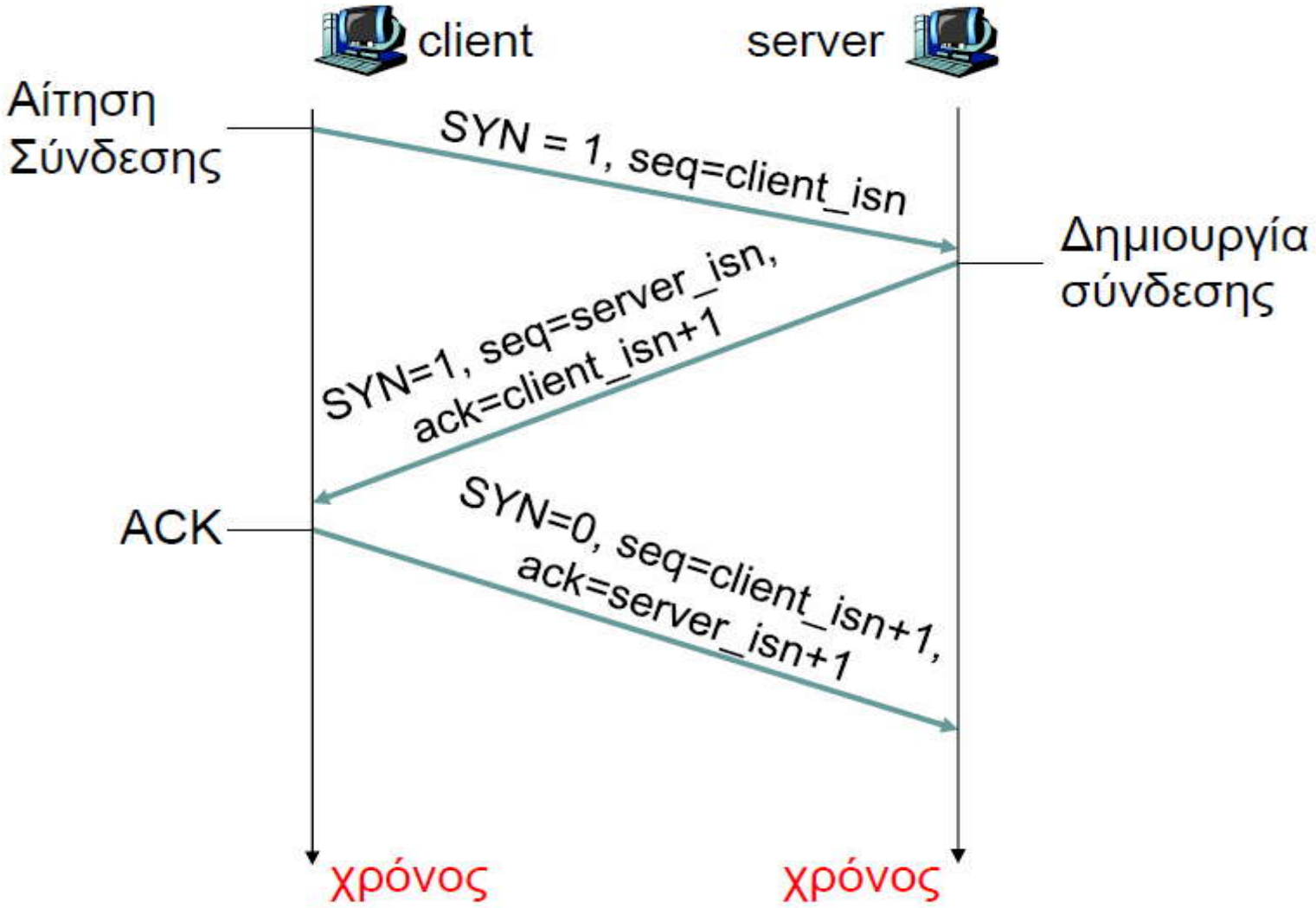
- καθορίζει τον αρχικό seq#
- δεν περιέχει δεδομένα

Βήμα 2: ο εξυπηρετητής λαμβάνει το SYN, αποκρίνεται με ένα τμήμα SYNACK

- ο εξυπηρετητής εκχωρεί προσωρινή αποθήκευση στη σύνδεση
- καθορίζει τον αρχικό αριθμό ακολουθίας (seq#) του εξυπηρετητή

Βήμα 3: ο πελάτης λαμβάνει το SYNACK, απαντάει με ένα τμήμα ACK, το οποίο μπορεί να περιέχει δεδομένα

Διαχείριση Σύνδεσης στο TCP – III



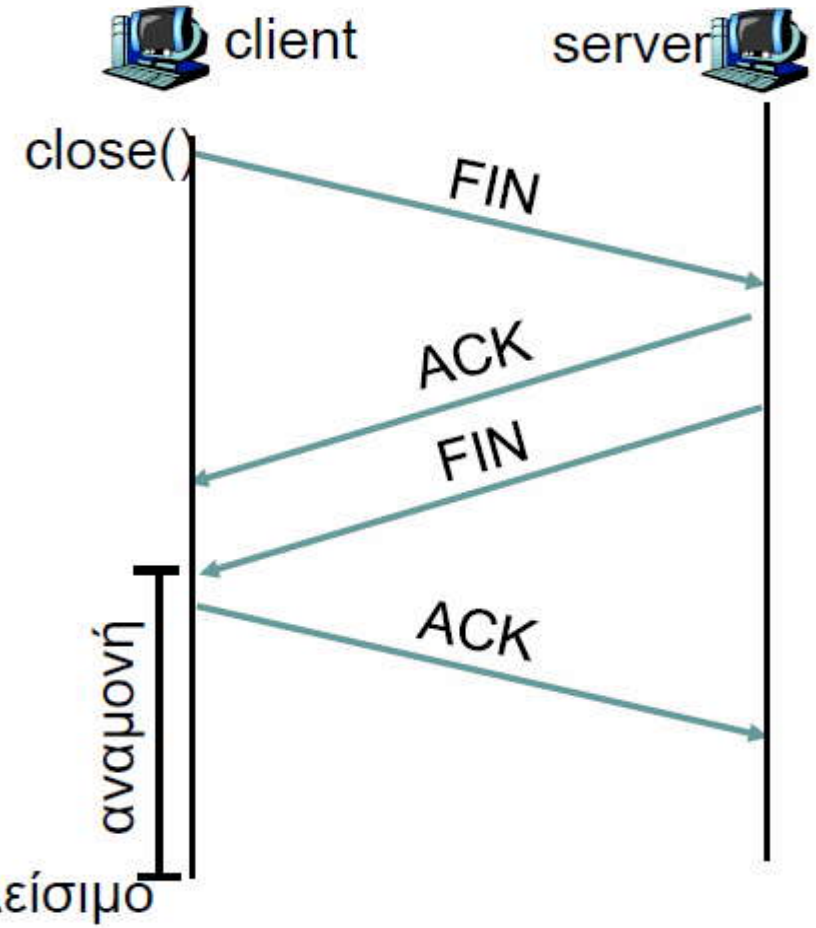
Διαχείριση Σύνδεσης στο TCP – IV

Τερματισμός σύνδεσης:

Ο πελάτης κλείνει τη σύνδεση (κλήση συστήματος `close()`)

Βήμα 1: ο πελάτης στέλνει ένα τμήμα ελέγχου TCP FIN στον εξυπηρετητή

Βήμα 2: ο εξυπηρετητής λαμβάνει το FIN, και αποκρίνεται με ACK. Κλείνει τη σύνδεση, στέλνοντας FIN



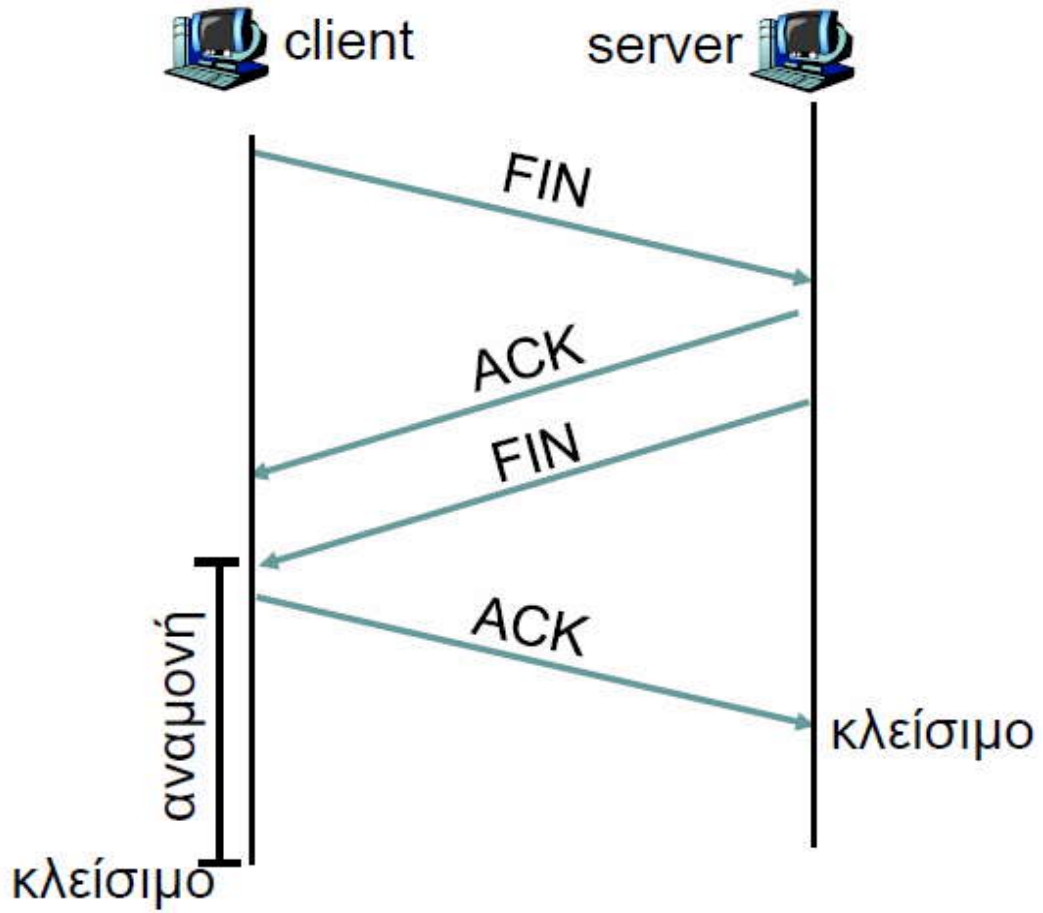
Διαχείριση Σύνδεσης στο TCP – V

Βήμα 3: Ο πελάτης λαμβάνει το FIN, αποκρίνεται με ACK

- αναμένει – θα αποκριθεί με ACK στα FINs που έλαβε

Βήμα 4: Ο εξυπηρετητής λαμβάνει το ACK. Η σύνδεση κλείνει (οι πόροι απελευθερώνονται)

Σημείωση: με μικρή τροποποίηση μπορεί να χειριστεί ταυτόχρονα FINs



Αξιόπιστη Μεταφορά Δεδομένων (Reliable Data Transfer)

- Το TCP δημιουργεί την υπηρεσία αξιόπιστης μεταφοράς δεδομένων πάνω από τη μη αξιόπιστη υπηρεσία του IP
- Συνεχής διοχέτευση τμημάτων (pipelining)
- Αθροιστικά (cumulative) ACKs
- Το TCP χρησιμοποιεί χρονομετρητή επαναμετάδοσης (retransmission timer)
- Οι επαναμεταδόσεις προκαλούνται από:
  - γεγονότα λήξης χρόνου (timeouts)
  - διπλότυπα (duplicate) ACKs

Λήξη Χρόνου και Χρόνος Round Trip (RTT) στο TCP – I

- Ερ: Ποια πρέπει να είναι η τιμή για το timeout;
- Μεγαλύτερη από το RTT (Round Trip Time)
  - αλλά το RTT μεταβάλλεται
- Πολύ μικρή: πρώιμες λήξεις χρόνου (timeouts)
  - μη απαραίτητες επαναμεταδόσεις
- Πολύ μεγάλη: αργή αντίδραση σε απώλειες τμημάτων

Ερ: πως υπολογίζεται το RTT;

- `SampleRTT`: ο χρόνος από τη μετάδοση του τμήματος μέχρι την παραλαβή του ACK
  - αγνοούμε τις επαναμεταδόσεις
- Το **`SampleRTT`** θα μεταβάλλεται, χρειαζόμαστε το εκτιμώμενο RTT «ομαλότερο»
  - χρησιμοποιούμε τις πρόσφατες μετρήσεις και όχι μόνο το τρέχον **`SampleRTT`**

Λήξη Χρόνου και Χρόνος Round Trip (RTT) στο TCP – II

Υπολογισμός:

```
EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT
```

- Εκθετικός με βάρη μεταβαλλόμενος μέσος όρος (Exponential Weighted Moving Average – EWMA)
- Η επίδραση των παλαιότερων δειγμάτων μειώνεται με εκθετικό ρυθμό
- Τυπική τιμή: $\alpha = 0,125$

Λήξη Χρόνου και Χρόνος Round Trip (RTT) στο TCP – III

Υπολογισμός της λήξης χρόνου:

- **EstimatedRTT** συν ένα «περιθώριο ασφαλείας»
  - μεγάλη διακύμανση στο **EstimatedRTT** ⇒ μεγαλύτερο περιθώριο ασφαλείας
- Αρχικά υπολογισμός του πόσο το **SampleRTT** αποκλίνει από το **EstimatedRTT**:

DevRTT = (1-β)*DevRTT +
β*|SampleRTT-EstimatedRTT|

(Τυπική τιμή: $\beta = 0.25$)

Στη συνέχεια το χρονικό διάστημα λήξης χρόνου γίνεται:

TimeoutInterval = EstimatedRTT + 4*DevRTT

Γεγονότα Αποστολέα TCP

Λήψη δεδομένων:

- Δημιουργία τμήματος με αριθμό ακολουθίας (seq #)
- Ο seq # είναι ο αριθμός σειράς του πρώτου byte δεδομένων του τμήματος
- Εκκίνηση μετρητή αν δεν τρέχει ήδη
- Χρονικό διάστημα εκπνοής: TimeOutInterval

Λήξη χρόνου (timeout):

- Επαναμετάδοση τμήματος που προκάλεσε τη λήξη χρόνου
- Επανεκκίνηση μετρητή

Λήψη ACK:

- Αν αναγνωρίζει τμήματα που δεν έχουν αναγνωριστεί:
  - ενημέρωσε ότι είναι γνωστό ότι έχει αναγνωριστεί
  - εκκίνηση μετρητή αν υπάρχουν τμήματα που περιμένουν

Αριθμοί Ακολουθίας και Επαλήθευσης στο TCP

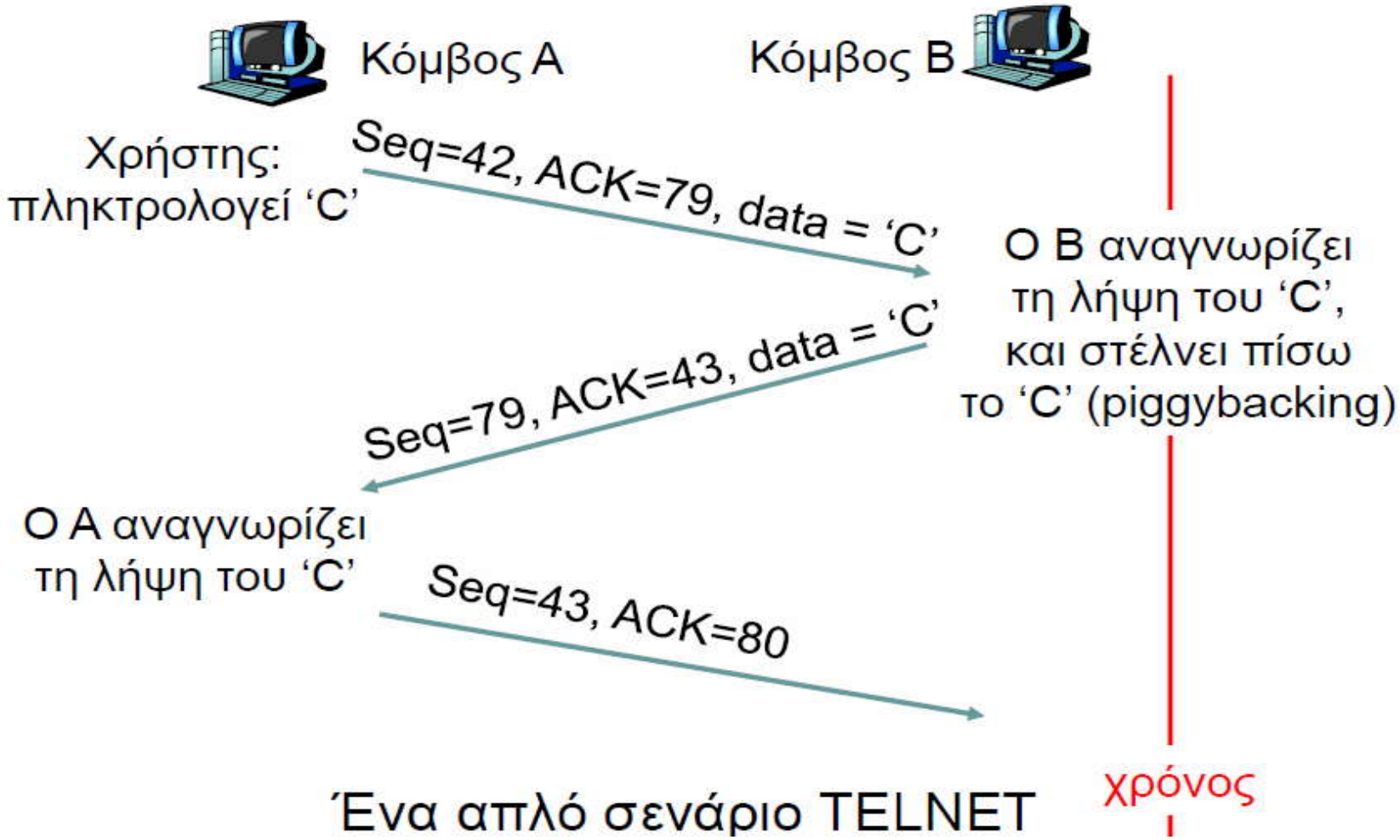
Αριθμός ακολουθίας:

- Ο αριθμός ροής πρώτου byte στα δεδομένα του τμήματος (όλα τα προς αποστολή bytes αριθμούνται)

ACKs:

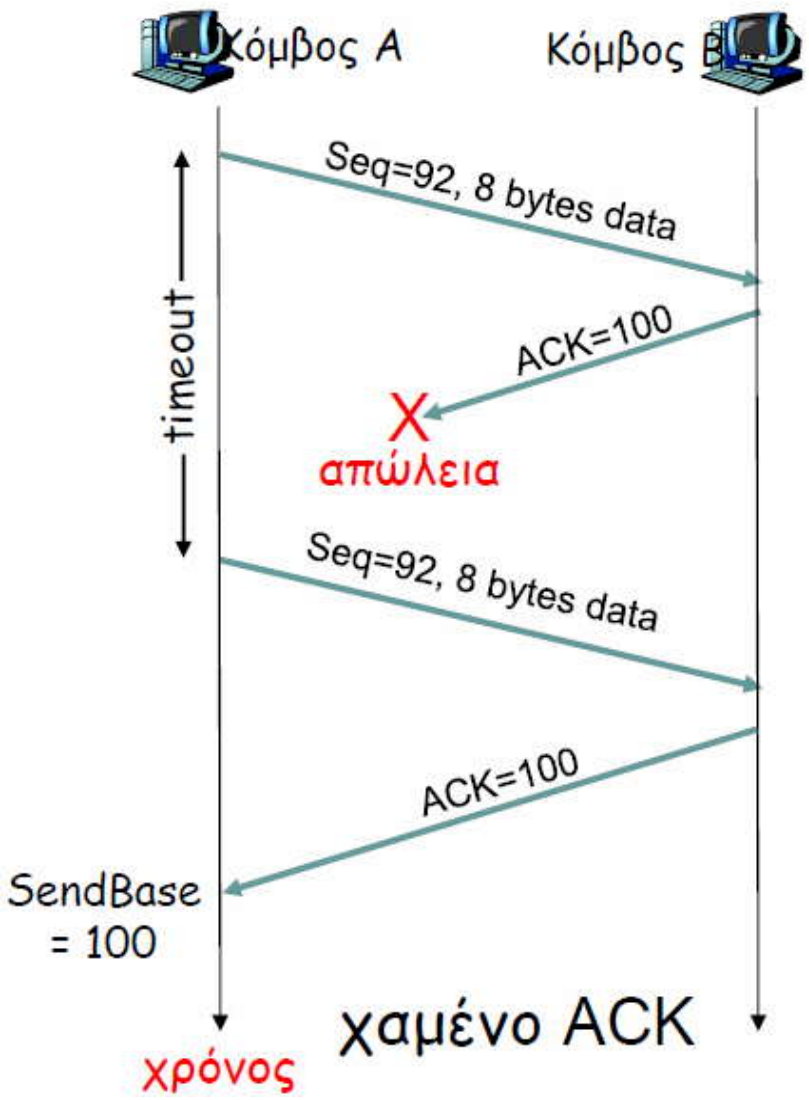
- Αριθμός ακολουθίας του επόμενου byte που αναμένεται από την άλλη πλευρά
- Αθροιστικές αναγνωρίσεις (cumulative ACK)

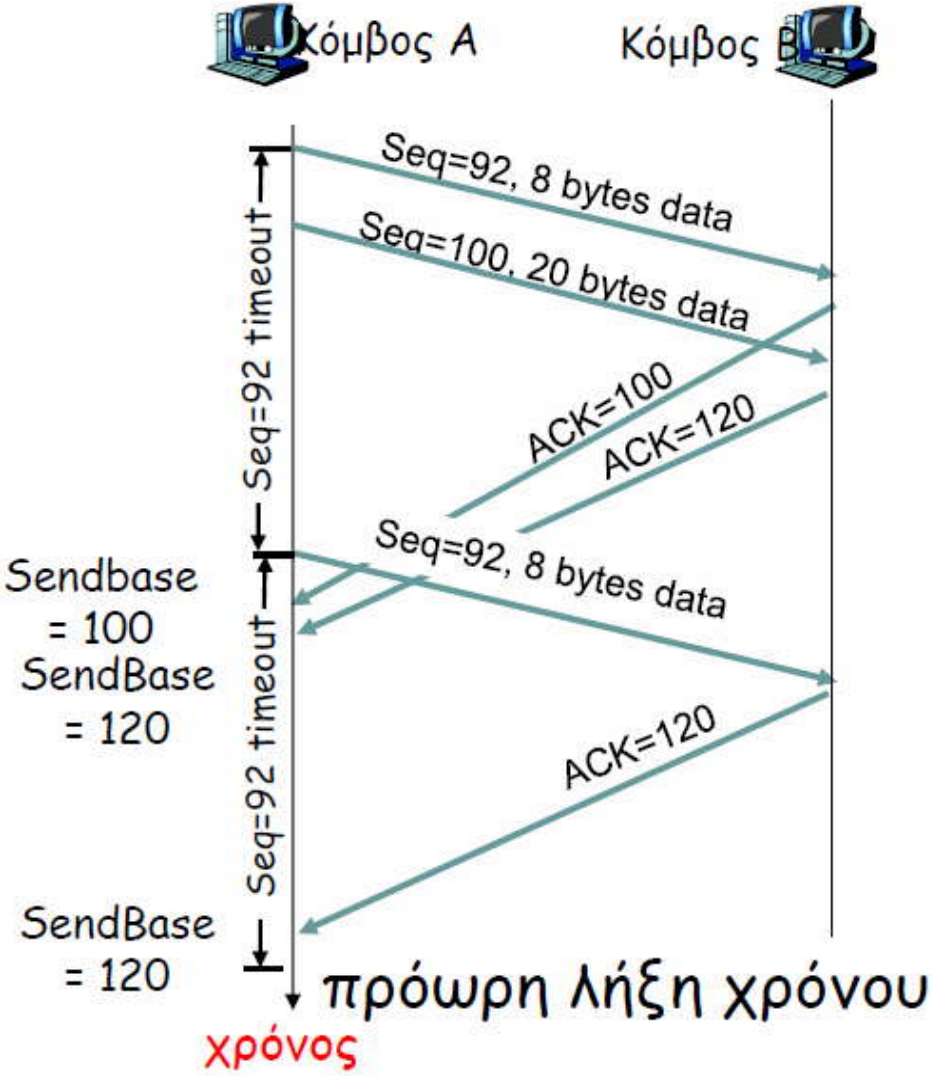
Αριθμοί Ακολουθίας και Επαλήθευσης – Παράδειγμα



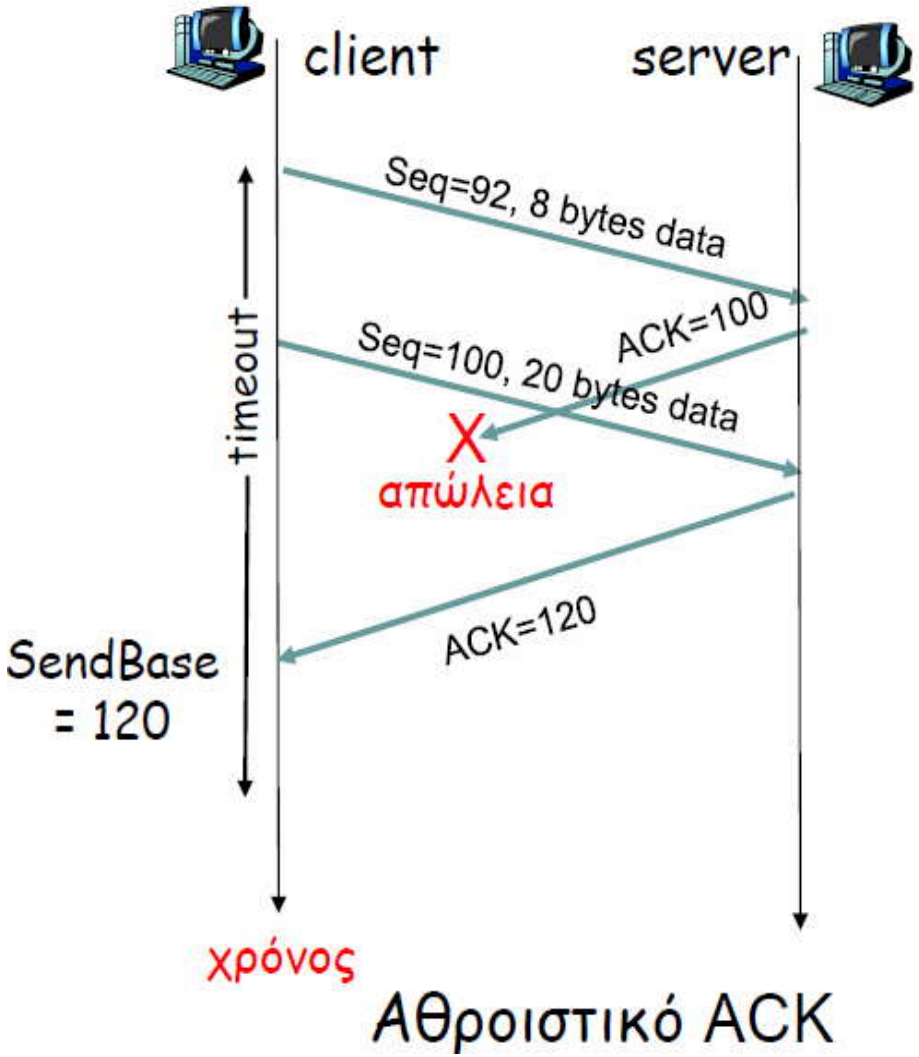
Ένα απλό σενάριο TELNET

Σενάρια Επαναμετάδοσης – Ι





Σενάρια Επαναμετάδοσης – II



Αθροιστικό ACK

- Κάθε φορά που γίνεται επανεκπομπή διπλασιάζεται η προηγούμενη τιμή της λήξης χρόνου
- Τα διαστήματα αυξάνονται εκθετικά μετά από κάθε επαναμετάδοση
- Πρόκειται για μια μορφή ελέγχου συμφόρησης

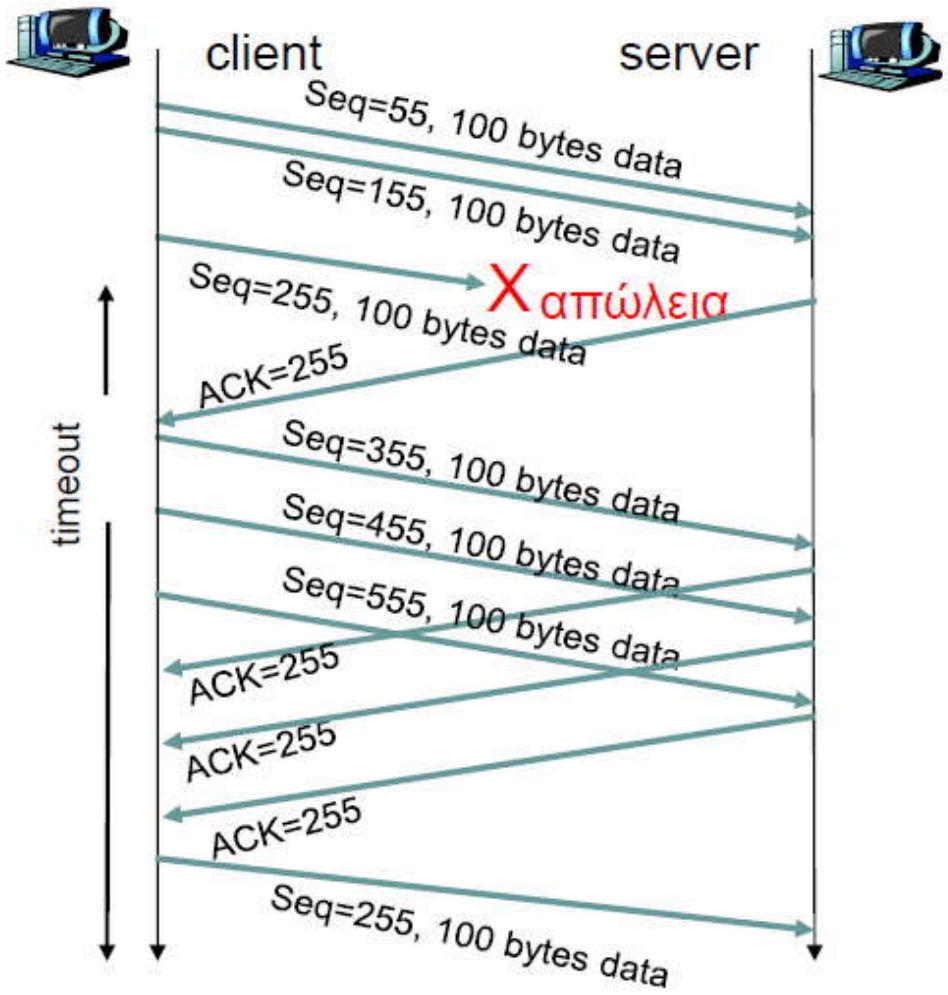
Δημιουργία ACKs στο TCP (RFC 1122, RFC 2581)

Γεγονός στον Παραλήπτη	Ενέργεια του Παραλήπτη TCP
Άφιξη τμήματος εντός σειράς με τον αναμενόμενο αριθμό seq. Όλα τα δεδομένα έως εκεί έχουν ACK	ACK με καθυστέρηση. Αναμονή έως 500ms για το επόμενο τμήμα. Αν δεν υπάρξει επόμενο τμήμα, αποστολή ACK
Άφιξη τμήματος εντός σειράς με τον αναμενόμενο αριθμό seq. Ένα άλλο τμήμα αναμένει ACK	Άμεση αποστολή ενός αθροιστικού ACK, αναγνωρίζοντας και τα δύο τμήματα που ήρθαν με τη σωστή σειρά
Άφιξη τμήματος εκτός σειράς με μεγαλύτερο από τον αναμενόμενο αριθμό seq. Ανίχνευση κενού	Άμεση αποστολή διπλότυπου ACK, που περιέχει τον αριθμό seq του επόμενου αναμενόμενου byte
Άφιξη τμήματος που μερικώς ή εξ ολοκλήρου γεμίζει το κενό	Άμεση αποστολή ACK, εφόσον το τμήμα ξεκινάει από το κάτω άκρο του κενού

Γρήγορη Επαναμετάδοση (Fast Retransmit) – I

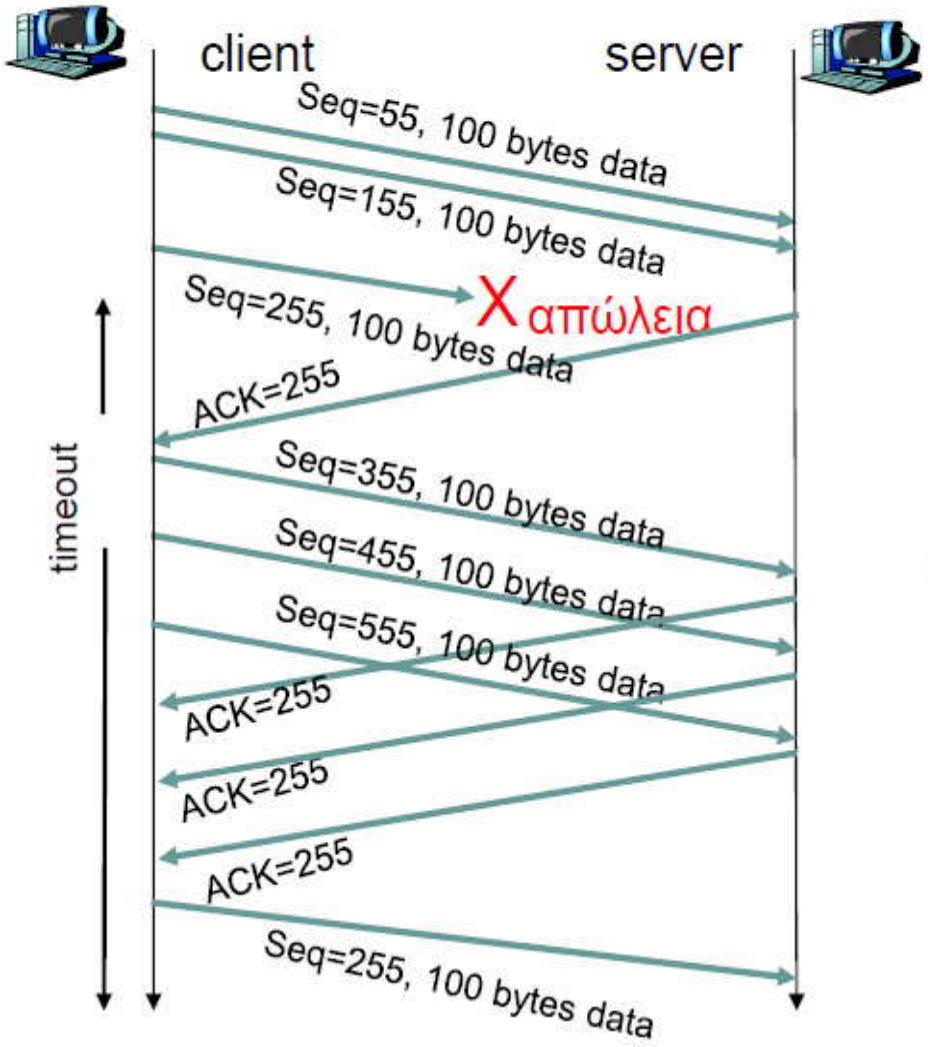
- Η περίοδος λήξης χρόνου είναι συνήθως σχετικά μεγάλη:
  - μεγάλη καθυστέρηση πριν την εκ νέου αποστολή του χαμένου πακέτου
- Ανίχνευση χαμένων τμημάτων μέσω διπλότυπων ACKs
  - ο αποστολέας συνήθως στέλνει πολλά τμήματα συνεχόμενα
  - αν ένα τμήμα χαθεί, θα υπάρξουν πιθανά πολλά διπλότυπα ACKs
- Αν ο αποστολέας λάβει 3 διπλότυπα ACKs για τα ίδια δεδομένα, υποθέτει ότι χάθηκε το τμήμα μετά από τα δεδομένα για τα οποία υπάρχει ACK:
  - γρήγορη επαναμετάδοση: αποστολή εκ νέου του τμήματος πριν λήξει ο μετρητής

Γρήγορη Επαναμετάδοση – II



- Το πρώτο ACK αναγνωρίζει αθροιστικά τα τμήματα με Seq=55 και 155
- Το τμήμα 255 χάνεται
- Τα τμήματα 355, 455 και 555 φτάνουν ορθά στον παραλήπτη, αλλά εκτός σειράς

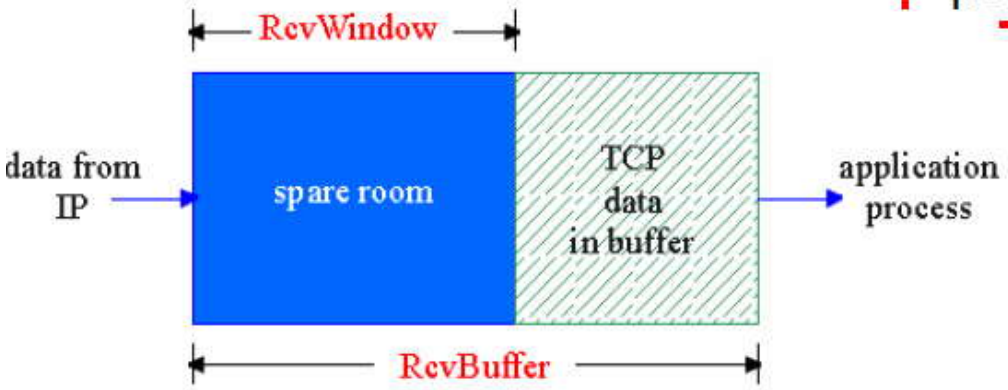
Γρήγορη Επαναμετάδοση – III



- Κάθε τμήμα που φτάνει εκτός σειράς:
  - αποθηκεύεται στο buffer λήψης
  - αναγνωρίζεται (δηλ. στέλνεται ACK ξανά) για το τελευταίο τμήμα που ήρθε με τη σειρά που αναμενόταν
- Στις τρεις συνεχόμενες αφίξεις του ίδιου ACK (μετά την αρχική), ξαναστέλνεται το τμήμα που λείπει (το τμήμα που υποδηλώνεται από τα ACK, στο παράδειγμα το 255) πριν συμβεί λήξη χρόνου

Ο Έλεγχος Ροής στο TCP (TCP Flow Control)

- Η πλευρά λήψης της σύνδεσης TCP έχει ένα buffer λήψης:



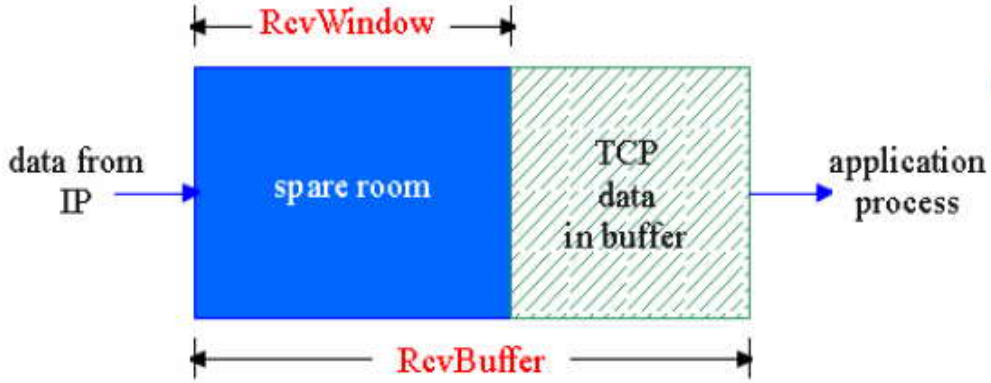
- Η διεργασία εφαρμογής μπορεί να είναι αργή στην ανάγνωση από το buffer

flow control

Ο αποστολέας δεν θα υπερχειλίσει την προσωρινή μνήμη (buffer) του παραλήπτη μεταδίδοντας γρήγορα μεγάλο όγκο δεδομένων

- Υπηρεσία συνταιριάσματος ταχυτήτων (speed matching): ταίριασμα του ρυθμού αποστολής με το ρυθμό λήψης του παραλήπτη

Ο Έλεγχος Ροής στο TCP: Πως Δουλεύει



(Υποθέτουμε ότι ο παραλήπτης TCP απορρίπτει τα εκτός σειράς τμήματα)

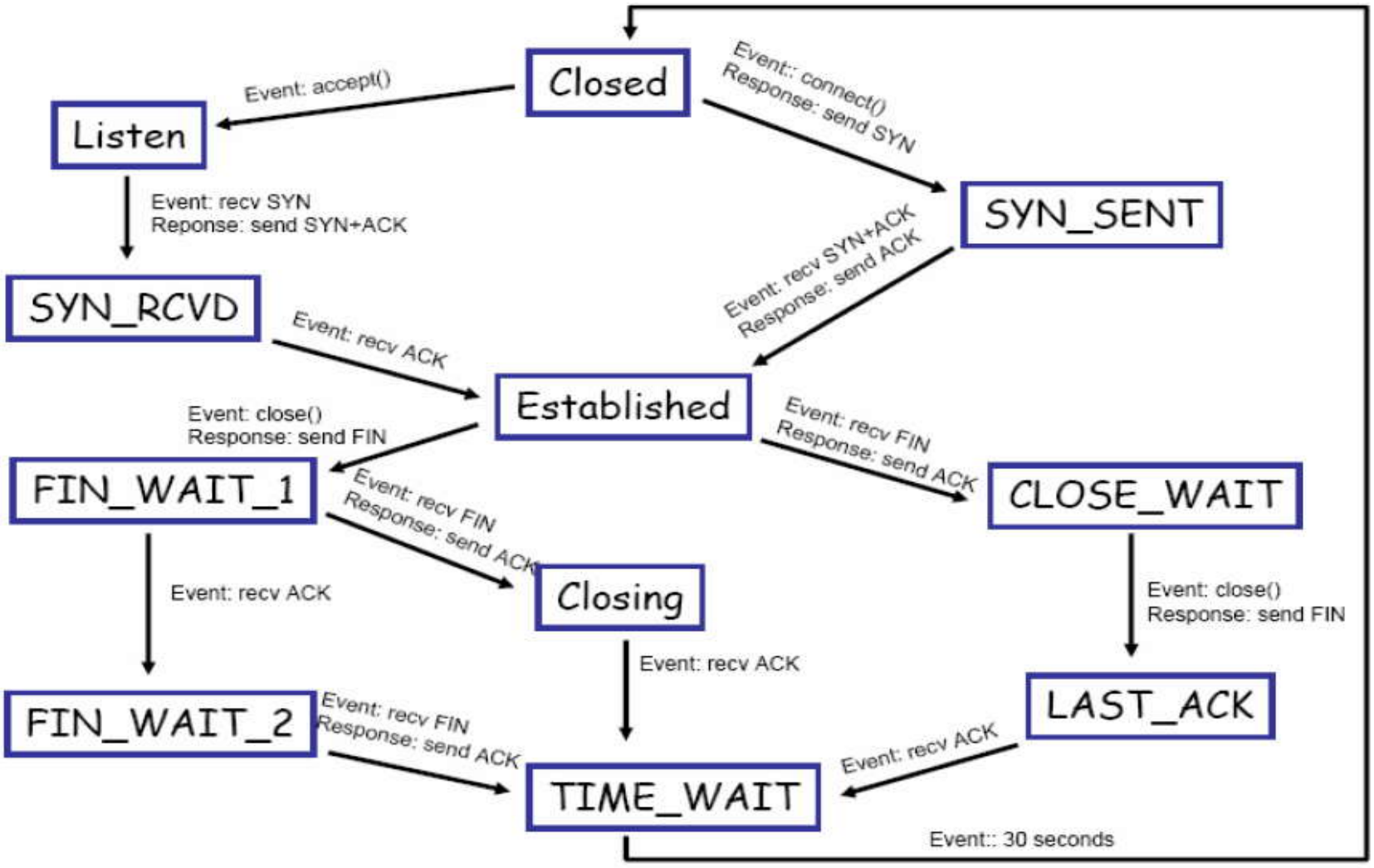
- ελεύθερος χώρος στο buffer

= **RcvWindow**

= **RcvBuffer-[LastByteRcvd - LastByteRead]**

- Ο παραλήπτης διαφημίζει τον ελεύθερο χώρο βάζοντας την τιμή της **RcvWindow** στα τμήματα
- Ο αποστολέας περιορίζει τα χωρίς ACK δεδομένα σε **RcvWindow**
  - εγγυάται ότι ο χώρος προσωρινής αποθήκευσης λήψης (receive buffer) δεν θα υπερχειλίσει

Mini-TCP FSM


Closed
Listen
Event: accept()
Event:: connect()
Response: send SYN
SYN_SENT
Event: recv SYN
Reponse: send SYN+ACK
SYN_RCVD
Event: recv SYN+ACK
Response: send ACK
Event: recv ACK
Established
Event: close()
Response: send FIN
Event: recv FIN
Response: send ACK
FIN_WAIT_1
CLOSE_WAIT
Event: recv FIN
Response: send ACK
Event: recv ACK
Closing
Event: close()
Response: send FIN
Event: recv ACK
LAST_ACK
FIN_WAIT_2
Event: recv FIN
Response: send ACK
Event: recv ACK
TIME_WAIT
Event:: 30 seconds

Αρχές του Ελέγχου Συμφόρησης

Συμφόρηση:

- άτυπα: «πολλές πηγές, στέλνουν πολλά δεδομένα, πολύ γρήγορα για τις δυνατότητες χειρισμού που έχει το δίκτυο»
- Διαφορετική από τον έλεγχο ροής!
- Συμπτώματα:
  - απώλειες πακέτων (υπερχείλιση των buffers στους δρομολογητές)
  - μεγάλες καθυστερήσεις (αναμονή σε ουρές στους buffers των δρομολογητών)
- Ένα από τα 10 σημαντικότερα προβλήματα των δικτύων

Προσεγγίσεις για τον έλεγχο συμφόρησης

Δύο βασικές προσεγγίσεις για τον έλεγχο συμφόρησης:

Έλεγχος συμφόρησης από άκρο σε άκρο:

- χωρίς ανάδραση από το δίκτυο
- η συμφόρηση γίνεται αντιληπτή από τις απώλειες στα τελικά συστήματα και την καθυστέρηση
- η προσέγγιση που χρησιμοποιεί το TCP

Έλεγχος συμφόρησης με βοήθεια από το δίκτυο:

- οι δρομολογητές παρέχουν ανάδραση στα τελικά συστήματα
  - ένα bit που υποδηλώνει συμφόρηση (SNA, DECbit, TCP/IP ECN, ATM)
  - σαφής ρυθμός με τον οποίο πρέπει να στέλνει ο αποστολέας

Ο Έλεγχος Συμφόρησης στο TCP

- Έλεγχος από άκρο σε άκρο (το IP δεν παρέχει καμία βοήθεια)
- Ο αποστολέας περιορίζει τη μετάδοση:

  LastByteSent-LastByteAcked ≤ **min{CongWin, RcvWindow}**
- Κατά προσέγγιση,

$$\text{ρυθμός} = \frac{\text{CongWin}}{\text{RTT}} \text{ bytes/sec}$$

- Το **CongWin** είναι δυναμικό, συνάρτηση της αντιληπτής συμφόρησης του δικτύου

Πως αντιλαμβάνεται τη συμφόρηση ο αποστολέας;

- Απώλεια = λήξη χρόνου *ή* 3 διπλότυπα ACKs
- Ο αποστολέας TCP μειώνει το ρυθμό (**CongWin**) μετά από κάθε απώλεια

Τρεις μηχανισμοί:

- Additive Increase Multiplicative Decrease (AIMD)
- Αργή Εκκίνηση (Slow start)
- Συντηρητικό μετά από γεγονότα λήξης χρόνου

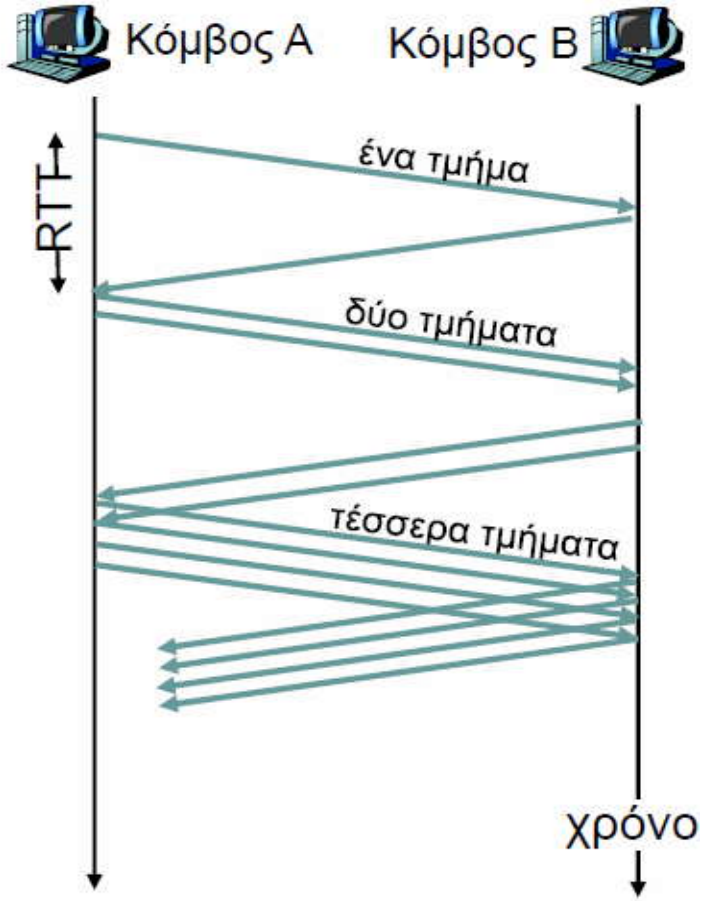
Additive Increase Multiplicative Decrease (AIMD) στο TCP

πολλαπλασιαστική μείωση: μείωσε το **CongWin** στο μισό μετά από κάθε απώλεια

προσθετική αύξηση: αύξανε το **CongWin** κατά 1 MSS κάθε RTT όσο δεν υπάρχουν απώλειες: *έλεγχος (probing)*

TCP Slow Start – II

- Όταν ξεκινάει η σύνδεση, αύξανε το ρυθμό εκθετικά έως την πρώτη απώλεια:
  - διπλασίασε το **`CongWin`** κάθε RTT
  - γίνεται με την αύξηση κατά 1 MSS του **`CongWin`** για κάθε ACK που λαμβάνεται
- Συνεπώς: ο αρχικός ρυθμός είναι αργός, αλλά αυξάνεται με εκθετικό ρυθμό



Απώλειες και TCP – I

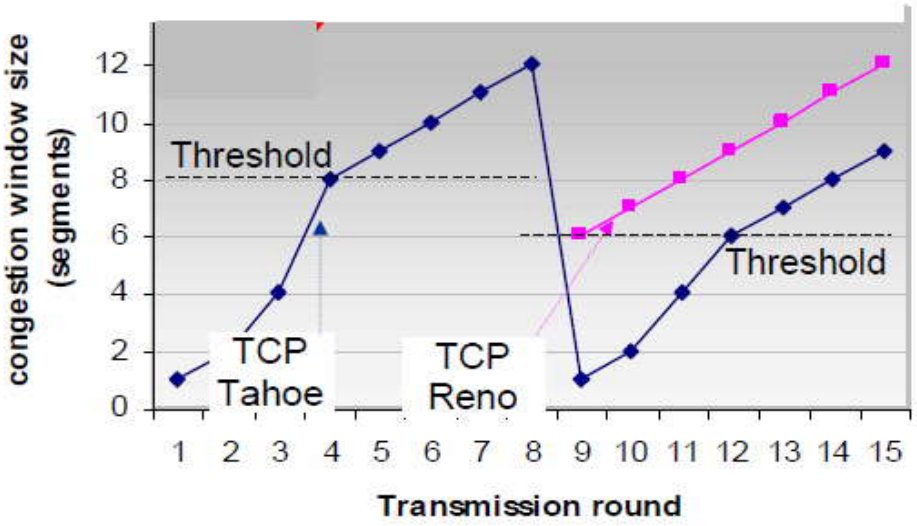
- Μετά από τρία διπλότυπα ACKs:
  - το **CongWin** διαιρείται δια 2
  - το παράθυρο αυξάνεται γραμμικά (TCP Reno)
- Αλλά μετά από λήξη χρόνου:
  - το **CongWin** παίρνει την τιμή 1 MSS
  - το παράθυρο αυξάνεται εκθετικά
  - έως ένα κατώφλι, στη συνέχεια μεγαλώνει γραμμικά

Αιτιολόγηση:

- 3 διπλότυπα ACKs υποδηλώνουν ότι το δίκτυο είναι σε θέση να παραδώσει κάποια τμήματα
- η λήξη χρόνου πριν τη λήψη 3 διπλότυπων ACKs είναι περισσότερο «ανησυχητική»

Απώλειες και TCP – II

- Ε: πότε η εκθετική αύξηση αλλάζει σε γραμμική;
- Α: όταν το **CongWin** φτάσει στο 1/2 της τιμής που είχε πριν τη λήξη χρόνου



Υλοποίηση:

- Χρήση μεταβλητής **Threshold** (αρχικά μεγάλη τιμή)
- Αν συμβεί απώλεια, η **Threshold** παίρνει τιμή το 1/2 που είχε το **CongWin** ακριβώς πριν την απώλεια

Περίληψη: Έλεγχος Συμφόρησης στο TCP

- **CongWin** < **Threshold**: ο αποστολέας βρίσκεται στη φάση αργής εκκίνησης, το παράθυρο μεγαλώνει εκθετικά
- **CongWin** ≥ **Threshold**: ο αποστολέας βρίσκεται στη φάση αποφυγής συμφόρησης, το παράθυρο μεγαλώνει γραμμικά
- Όταν ληφθούν 3 διπλότυπα ACKs, η **Threshold** παίρνει τιμή **CongWin/2** και το **CongWin** παίρνει τιμή **Threshold**
- Όταν συμβεί λήξη χρόνου, η **Threshold** παίρνει τιμή **CongWin/2** και το **CongWin** παίρνει τιμή 1 MSS