Δίκτυα Υπολογιστών 20/11 8ο

rdt2.1: συζήτηση

παραλήπτης:  
▪ ελέγχει εάν ληφθέν πακέτο είναι διπλότυπο

• κατάσταση υποδεικνύει εάν αναμενόμενος # ακολουθίας είναι 0 ή 1

▪ σημείωση: παραλήπτης *δεν* μπορεί να γνωρίζει εάν το τελευταίο του ACK/NAK  
παρελήφθη σωστά από τονΝαποστολέα

αποστολέας:  
▪ # ακολουθίας προστίθεται στο πακέτο  
▪ δύο # ακολουθίας (0,1) αρκούν.  
Γιατί;  
▪ πρέπει να ελέγχει εάν ληφθέν ACK/NAK αλλοιωμένο  
▪ δύο φορές περισσότερες καταστάσεις

• Κατάσταση πρέπει να “θυμάται” εάν  
“αναμενόμενο” πακέτο θα έπρεπε να

έχει # ακολουθίας 0(παλίο) ή 1(καινούργιο)

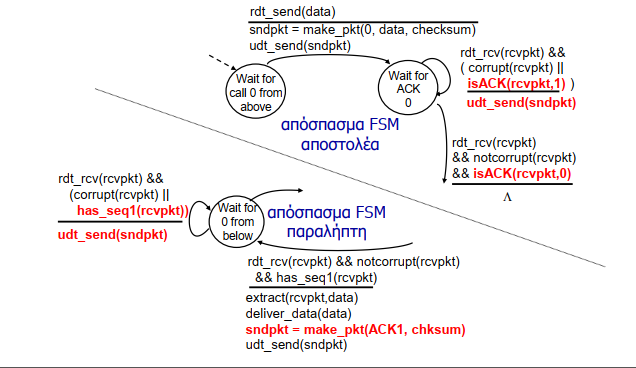
rdt2.2: ένα πρωτόκολλο χωρίς NAK, άλλη μία τροποποίηση στο προτοκολό μας

▪ ίδια λειτουργικότητα με rdt2.1, χρησιμοποιώντας μόνον πακέτα ACK ( ποιο πακέτο επιβεβαιώνει αν το πήρα σωστά -> το 0 ή 1, αλλίος στέλνει ξανά το ACK)  
▪ αντί για NAK, ο παραλήπτης στέλνει ACK για το τελευταίο πακέτο που έλαβε σωστά

* Παραλήπτης πρέπει να συμπεριλάβει ρητά τον # ακολουθίας του πακέτου που επιβεβαιώνει

▪ διπλότυπο ACK στον αποστολέα έχει ως αποτέλεσμα την ίδια ενέργεια όπως για το NAK: *επαναμετάδοση τρέχοντος πακέτου*Όπως θα δούμε, TCP ακολουθεί αυτήν την προσέγγιση ώστε να μην χρησιμοποιεί NAK

rdt2.2: αποσπάσματα αποστολέα, παραλήπτη

Η γραμμή ονομάζεται=> μηχανή πεπερασμένης

*rdt3.0: κανάλια με σφάλματα και απώλεια*

*Νέα υπόθεση για το κανάλι: υποκείμενο κανάλι μπορεί επίσης να χάνει πακέτα (δεδομένων, ACKs)*

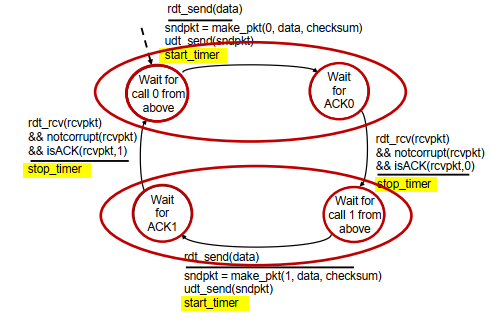
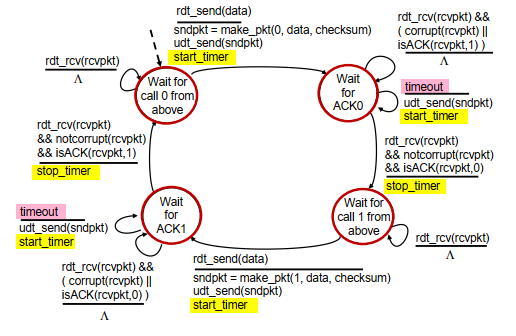
* *checksum, # ακολουθίας, ACKs, επαναμεταδόσεις θα βοηθήσουν… αλλά δεν είναι αρκετά*

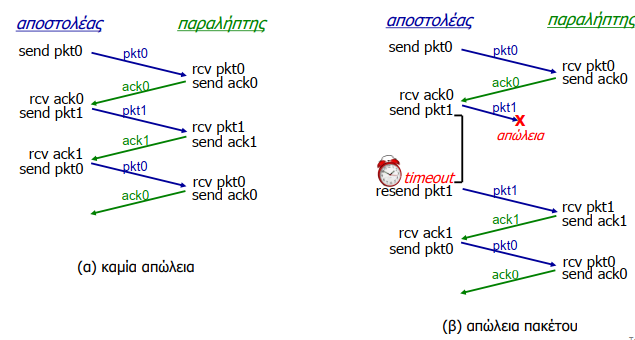
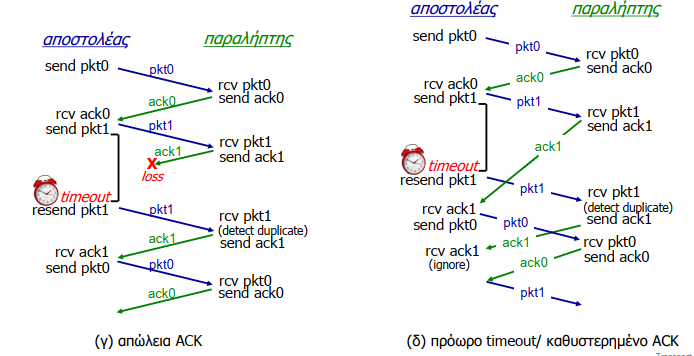
***Ε****: Πώς χειρίζονται οι άνθρωποι απώλειες λέξεων μεταξύ αποστολέα και παραλήπτη σε μία συζήτηση;*

*Προσέγγιση: αποστολέας περιμένει “λογικό” χρόνο για ACK  
▪ επαναμεταδίδει εάν δεν λάβει ACK στον χρόνο αυτόν  
▪ εάν πακέτο (ή ACK) απλά καθυστέρησε (δεν χάθηκε):*

* *επαναμετάδοση θα είναι διπλότυπο, αλλά αριθμοί ακολουθίας ήδη το αντιμετωπίζουν*
* *παραλήπτης πρέπει να ορίσει # ακολουθ. πακέτου που επιβεβαιώνει*

*▪ χρήση χρονομέτρου αντίστροφης→timer μέτρησης για διακοπή μετά από  
“λογικό” ποσό χρόνου*

rdt3.0 αποστολέας

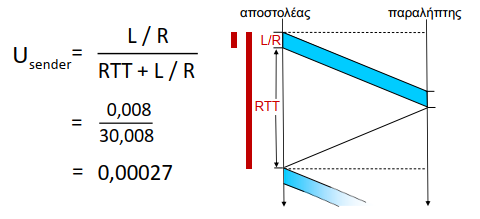
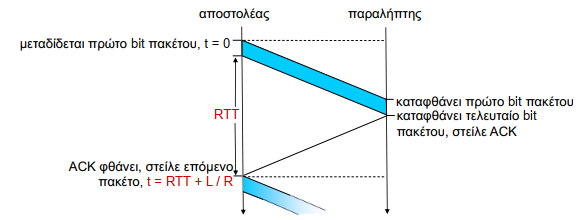
rdt3.0 σε δράση

Απόδοση του rdt3.0 (stop-and-wait)

▪*Usender*: *βαθμός χρήσης* – κλάσμα χρόνου που χρησιμοποιεί το κανάλι ο αποστολέας

▪ παράδειγμα: ζεύξη 1 Gbps, καθυστέρηση διάδοσης 15 ms, πακέτο 8000 bit

L=χρονος για να πραξω   
▪ απαιτούμενος χρόνος για μετάδοση πακέτου στο κανάλι:

rdt3.0: λειτουργία stop-and-wait

**15ms + 8 μs**

**8μs**

**Συνολικος χρόνος για να πράξει**

**Συνολικος χρόνος χρήσης**

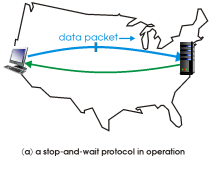
**15 ms**

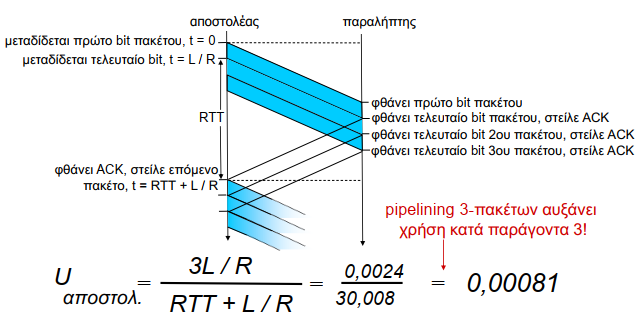
**800 bit**

▪ απόδοση πρωτοκόλλου rdt 3.0 χάλια!  
▪ πρωτόκολλο περιορίζει αποδοτικότητα υποκείμενης υποδομής (καναλιού)

rdt3.0: λειτουργία pipelined πρωτοκόλλων

pipelining (διοχέτευση): αποστολέας επιτρέπει πολλαπλά πακέτα “σε πτήση”, που δεν έχουν (ακόμα) επιβεβαιωθεί

* εύρος αριθμών ακολουθίας πρέπει να αυξηθεί
*  buffering (ενταμίευση) σε αποστολέα και/ή παραλήπτη ( έτσι ώστε αν δεν μπορώ να στείλω τα πακέτα τότε τα κρατάω σε ένα buffer, έτσι να μπορώ να ξανα στέλω )

Pipelining: αυξημένος βαθμός χρήσης

Go-Back-N: αποστολέας DSD

▪ αποστολέας: “παράθυρο” έως *N*, διαδοχικά μεταδοθέντα αλλά ανεπιβεβαίωτα (*μη ACKed*) πακέτα  
• # ακολουθίας *k*-bit στην κεφαλίδα του κάθε πακέτου

* Τα έστειλε , τέλειωσε και η επιβεβαίωση
* Μόνο τα έστειλε
* Μπορεί να τα στείλει
* Δεν υπάρχει χώρος να τα στείλει (άσπρο)

▪ σωρευτικό ACK: ACK(n): ACKs όλα τα πακέτα έως και # ακολουθίας n

* κατά την λήψη ACK(n): ολισθαίνει παράθυρο εμπρός για να ξεκινά από n+1

▪ χρονόμετρο για παλαιότερο πακέτο “σε πτήση”

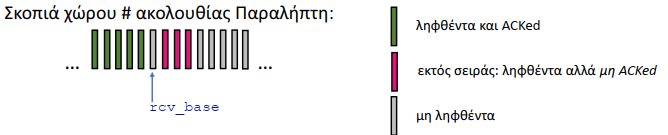
▪ timeout(n): επαναμετάδοση πακέτου n και όλων των πακέτων υψηλοτέρου # ακολουθίας που είναι στο παράθυρο

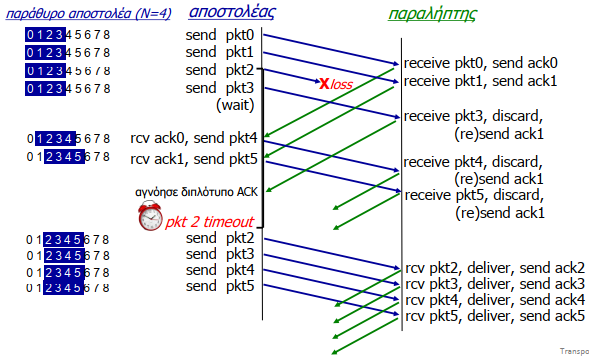
Go-Back-N: παραλήπτης

▪ ACK-μόνον: πάντα στέλνει ACK για ορθά ληφθέν έως τώρα πακέτο, με τον μεγαλύτερο # ακολουθίας **in-order (σε σωστή σειρά)**

* μπορεί να δημιουργήσει διπλότυπα ACK
* χρειάζεται να θυμάται μόνον το rcv\_base

▪ κατά την λήψη πακέτου εκτός σειράς:

* μπορεί να απορρίψει (δεν ενταμιεύει) ή να ενταμιεύσει: απόφαση της υλοποίησης
*  επανεπιβεβαιώνει πακέτο με μέγιστο # ακολουθίας in-order (σε σωστή σειρά)

Go-Back-N σε δράση

Selective repeat (επιλεκτική επανάληψη)

▪παραλήπτης επιβεβαιώνει *ξεχωριστά* όλα τα ορθώς ληφθέντα πακέτα

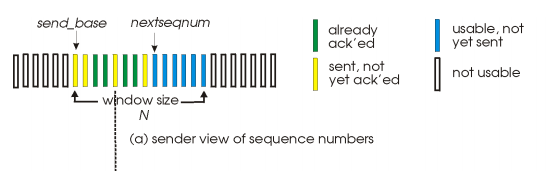
* Ενταμιεύει (buffers) πακέτα, όπως χρειάζεται, για τελική παράδοση στο άνω επίπεδο, με την σωστή σειρά

▪αποστολέας times-out/επαναμεταδίδει πακέτα ξέχωρα μόνον για τα οποία δεν έλαβε AC

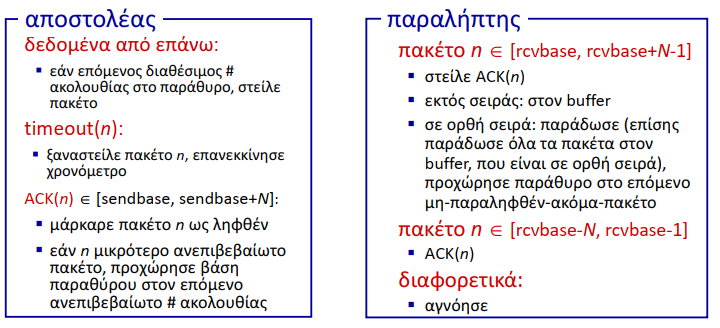
* αποστολέας τηρεί χρονόμετρο για κάθε ανεπιβεβαίωτο πακέτο

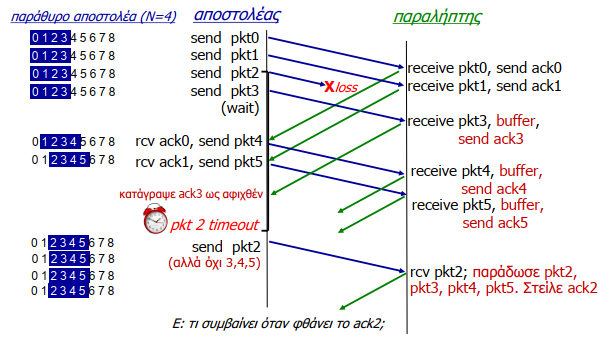
▪παράθυρο αποστολέα

* *N* διαδοχικοί # ακολουθίας
* περιορίζει πλήθος # ακολουθίας απεσταλμένων, ανεπιβεβαίωτων πακέτων

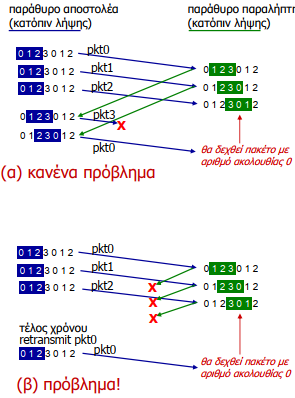
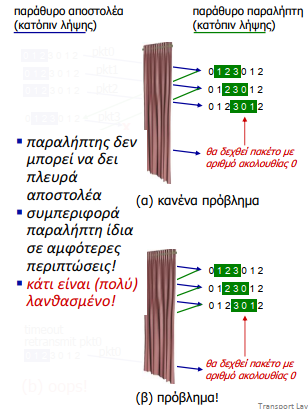
Selective repeat: παράθυρα αποστολέα, παραλήπτη

Selective repeat: αποστολέας και παραλήπτης



Selective Repeat σε δράση

Στέλνει μόνο για αυτά που δεν πήραν επιβεβαίωση



Selective repeat: ένα δίλημμα!

παράδειγμα:  
▪ # ακολουθίας: 0, 1, 2, 3 (μέτρηση με βάση 4)  
▪ μέγεθος παραθύρου = 3

**Ε:** ποια σχέση χρειάζεται μεταξύ  
# ακολουθίας και μεγέθους  
παραθύρου για αποφυγή του  
προβλήματος στο σενάριο (β);→2N+1

Συνδεσμική μεταφορά: TCP  
• Δομή τμήματος (segment)  
• Αξιόπιστη μεταφορά δεδομένων  
• Έλεγχος ροής  
• Διαχείριση συνδέσεων

TCP: επισκόπηση RFCs: 793,1122, 2018, 5681, 7323

▪ σημείο-προς-σημείο:

▪ συνδεσμικό:

* χειραψία (ανταλλαγή μηνυμάτων ελέγχου) αρχικοποιεί κατάσταση  
  αποστολέα, παραλήπτη πριν την ανταλλαγή δεδομένων

▪ ελεγχόμενης ροής: flow control

* αποστολέας δεν θα πλημμυρίσει παραλήπτη
* ένας αποστολέας, ένας παραλήπτης

▪ αξιόπιστο, με σωστή σειρά, ρεύμα από *byte:*

* δεν υπάρχουν “όρια μηνύματος"

▪ πλήρως αμφίδρομα δεδομένα:

* αμφίδρομη ροή δεδομένων μέσω της ίδιας ζεύξεως
* MSS: maximum segment size (μέγιστο μέγεθος τμήματος)

▪ σωρευτικά ACKs  
▪ pipelining:

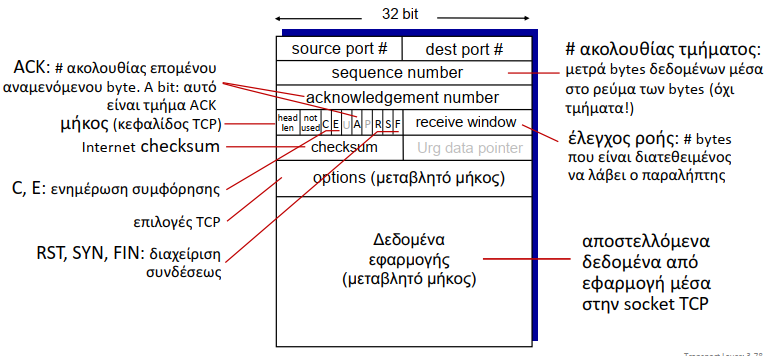
* συμφόρηση TCP έλεγχος ροής  
  καθορίζουν μέγεθος παραθύρου

Δομή τμήματος (segment)

**Πόσο μέγεθος σε byte είναι η γραμμή**

**32/8=8**

**Σώμα**



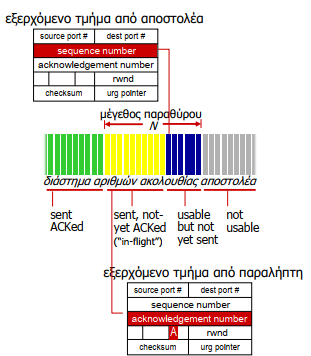
**42 +1 = 15 byte**

**5 byte**

**FIN=FINISH -> τερματικός συνδεσης ζητα SYN= δεν ζητά**

**STOP <->(Αντιστροφα)**

**Π.χ: the connect was RST**

TCP: αριθμοί ακολουθίας, ACK

*Αριθμοί ακολουθίας:*

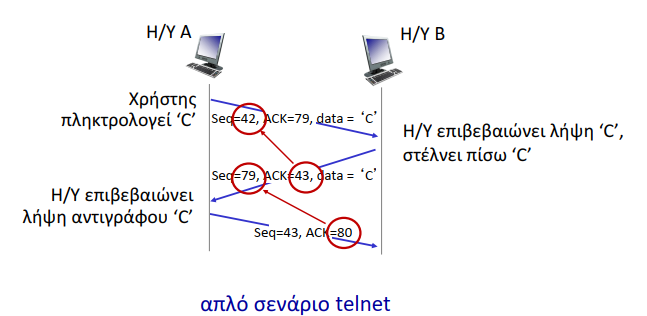
* “αύξων αριθμός” ρεύματος byte του πρώτου byte στα δεδομένα του τμήματος

*επιβεβαιώσεις*:

* # ακολουθίας του επόμενου byte που αναμένεται από την άλλη πλευρά
* σωρευτικό ACK

*Ε*: πώς χειρίζεται ο παραλήπτης τμήματα εκτός σειράς;

* *Α:* δεν αναφέρουν κάτι οι προδιαγραφές του TCP – αφήνεται στην υλοποίηση

42+1 γιατί ήταν ένα byte

79+1

**→Μπορώ να έχω ένα απλό συρμα μέσω κάποιο απομακρυσμένο telnet= app**