Σχεδιασμός Πρωτοκόλλων

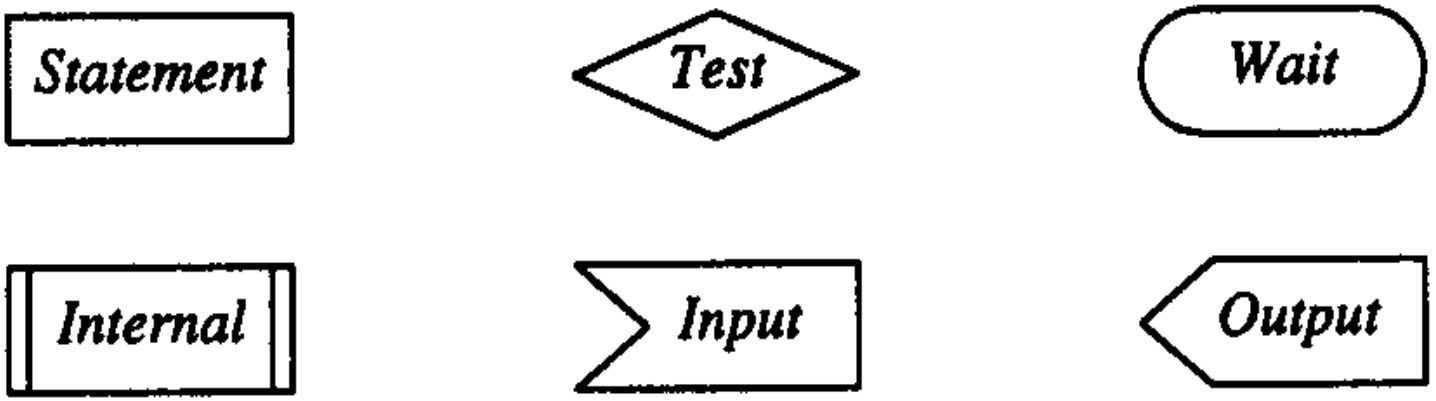
Χειμερινό εξάμηνο

Περίληψη

- Διαγράμματα ροής
- Στοιχεία πρωτοκόλλων
- Παράδειγμα
- Υπηρεσίες και περιβάλλον
- Λεξιλόγιο και μορφοποίηση
- Διαδικαστικοί κανόνες
- Κανόνες σχεδιασμού

Διαγράμματα Ροής

Χρησιμοποιούνται έξι διαφορετικοί τύποι συμβόλων στα διαγράμματα ροής:



Αυτά τα σύμβολα παριστάνουν:

1. **Δηλώσεις (Statements), π.χ. εκχωρήσεις (assignments)**
2. **Δοκιμές τιμών Μπουλ (Boolean tests), π.χ. εκφράσεις (expressions)**
3. **Συνθήκες αναμονής (wait conditions), π.χ. λήψεις (receives)**
4. **Εσωτερικά συμβάντα, π.χ. timeouts**
5. **Είσοδοι και έξοδοι μηνυμάτων (message inputs and outputs)**

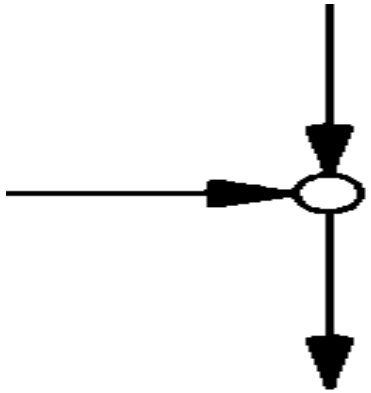
Διαγράμματα Ροής

Άλλα χρησιμοποιούμενα σύμβολα:

- **Κατευθυνόμενα βέλη (directed arcs)**
- **Σύνδεσμοι (connectors)**

Παρατηρήσεις:

1. **Τα κατευθυνόμενα βέλη υποδηλώνουν ότι ο έλεγχος ροής μπορεί να συγκλίνει μόνο σε συνδέσμους.**
2. **Μπορούν ενδεχομένως να αποκλίνουν, χωρίς συνδέσμους, σε συνθήκες αναμονής και σε δοκιμές τιμών Μπουλ.**

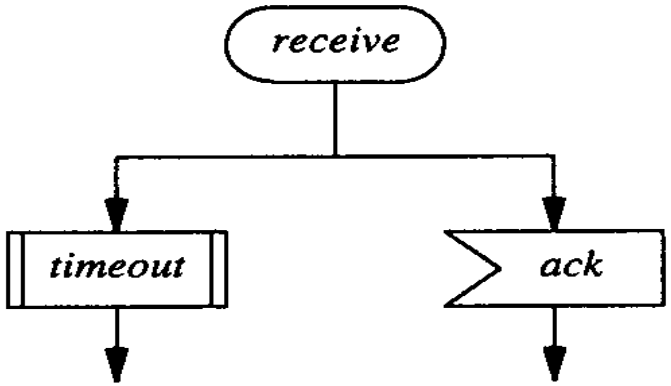


Διαγράμματα Ροής

- ***Γενικές Αρχές:***

- **Οι έξοδοι, οι δηλώσεις, οι συνθήκες αναμονής, τα εσωτερικά συμβάντα και οι δοκιμές τιμής Μπουλ μπορούν να εμφανίζονται οπουδήποτε σε ένα διάγραμμα ροής.**
- **Οι είσοδοι μπορούν μόνο να ακολουθούν ένα σύμβολο αναμονής, που φέρει την ετικέτα *receive (λήψη).***
- **Μπορούν να εμφανίζονται περισσότερες από μια είσοδοι.**

Διαγράμματα Ροής



Μια συνθήκη αναμονής (wait condition) που φέρει την ετικέτα *receive (λήψη)* θα καθυστερεί τη διεργασία εκτέλεσης μέχρις ότου η ουρά μηνυμάτων της διεργασίας, να περιέχει στην πρώτη θυρίδα της, ένα μήνυμα προδιαγραφόμενο σε μια από τις εισόδους που έπονται του συμβόλου αναμονής.

Συνιστά σφάλμα του πρωτοκόλλου, το εάν το μήνυμα στην πρώτη θυρίδα της ουράς είναι μήνυμα άλλου τύπου!!!

Διαγράμματα Ροής

- **Ένα timeout παριστάνεται ως ένα εσωτερικό συμβάν. Η αντίστοιχη συνθήκη τελικά πάντοτε θα καθίσταται αληθής (true).**

- **Εάν προδιαγράφεται ένα timeout που ακολουθεί ένα σύμβολο αναμονής (το οποίο φέρει την ετικέτα *receive (λήψη))*, η διεργασία εκτέλεσης μπορεί να ματαιώνει την αναμονή για ένα εισερχόμενο μήνυμα και να συνεχίζει με την εκτέλεση των δηλώσεων (statements) που ακολουθούν το συμβάν.**

- **Το σύμβολο αναμονής μπορεί επίσης να εμφανίζεται με μια έκφραση. Τότε, μια διεργασία εκτέλεσης θα υπόκειται σε καθυστέρηση, όταν αξιολογείται, αποδίδοντας στην τιμή Μπουλ την τιμή *true (αληθής)* (ή οποιαδήποτε μη μηδενική ακέραια τιμή).**

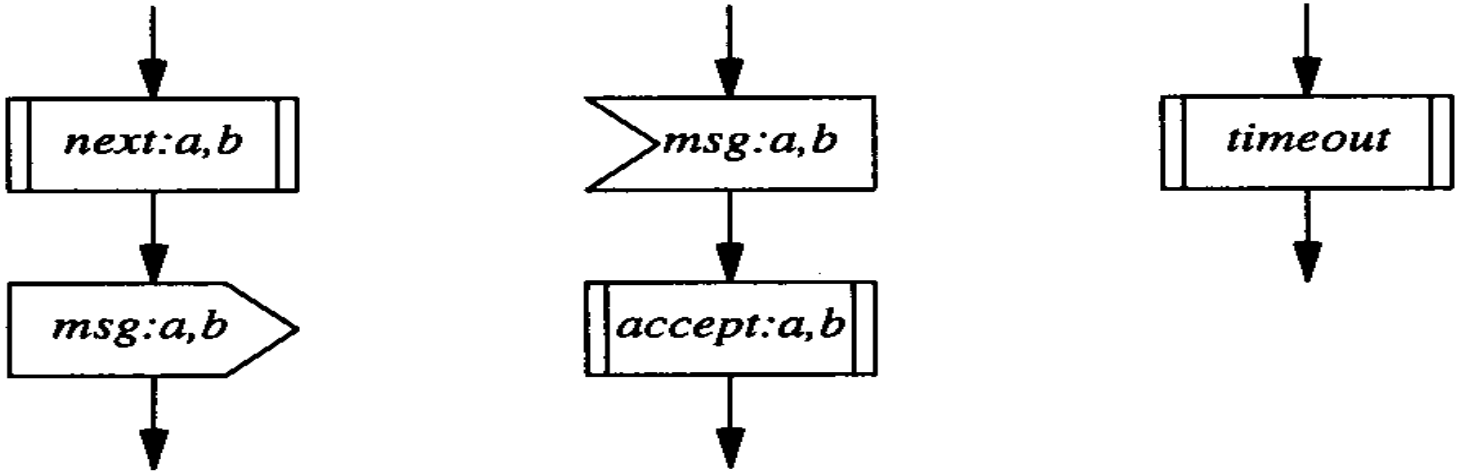
Διαγράμματα Ροής

- **Ορίζονται δύο ειδικές εσωτερικές δράσεις (internal actions) που μοντελοποιούν πρόσβαση σε δεδομένα:**
- ***next***
- ***accep*t.**

- **Ο συμβολισμός *next:a,b* υποδηλώνει την εσωτερική ανάκτηση των στοιχείων δεδομένων *a* και *b* από μια εσωτερική δομή.**

- **Ο συμβολισμός *accept:a,b* υποδηλώνει την αποθήκευση των στοιχείων δεδομένων σε μια εσωτερική δομή.**

Διαγράμματα Ροής



- **Η χρήση μεταβλητών και αφαιρετικών τύπων δεδομένων (abstract data types) δεν περιορίζεται από τη γλώσσα διαγραμμάτων ροής.**

- **Παρομοίως, τα περιεχόμενα ενός statement box μπορεί να είναι οτιδήποτε το οποίο δεν εμπλέκει συνθήκες αναμονής, μηνύματα λήψης ή αποστολής, timeouts και δοκιμές τιμών Μπουλ.**

Δομή Πρωτοκόλλων

Σύνολα κανόνων που ορίζουν την αλληλεπίδραση μεταξύ παράλληλων διαδικασιών

- Αρχή ανταλλαγής δεδομένων
- Συγχρονισμός διεργασιών
- Ανακάλυψη & διόρθωση λαθών
- Μορφοποίηση και κωδικοποίηση
- Λήξη ανταλλαγής δεδομένων

Στοιχεία πρωτοκόλλων

- **Μια προδιαγραφή πρωτοκόλλου αποτελείται από πέντε (-5-) διακριτά τμήματα:**
- **Την Υπηρεσία (service) που πρόκειται να περιγραφεί και να υλοποιηθεί**
- **Τις Υποθέσεις (assumptions) για το περιβάλλον λειτουργίας όπου εφαρμόζεται το πρωτόκολλο**
- **Το «Λεξιλόγιο» (vocabulary) των μηνυμάτων που χρησιμοποιούνται για την υλοποίηση του πρωτοκόλλου**
- **Τη Μορφοποίηση (formatting) των μηνυμάτων στο λεξιλόγιο**
- **Τους Διαδικαστικούς Κανόνες (procedure rules) επικοινωνίας, αναφορικά με τη συνέπεια των ανταλλαγών μηνυμάτων.**

- **Οι διαδικαστικοί κανόνες συνιστούν το δυσκολότερο τμήμα τόσο ως προς τη σχεδίαση όσο και ως προς την επαλήθευση!**

Παράδειγμα

Προδιαγραφή Υπηρεσίας

1. Μεταφορά ASCII αρχείων σαν σειρά χαρακτήρων
2. Προστασία για λάθη μετάδοσης
3. Αμφίδρομη επικοινωνία
4. Ύπαρξη θετικών και αρνητικών επιβεβαιώσεων (ACK, NACK)

Παράδειγμα

Υποθέσεις για το περιβάλλον λειτουργίας

1. Δύο χρήστες
2. Κανάλι επικοινωνίας

Χρήστες: Ζητάνε μεταφορά και περιμένουν να ολοκληρωθεί

Κανάλι: Μπορεί να καταστρέψει μηνύματα αλλά δεν χάνει, δεν δημιουργεί αντίγραφα, ή αλλάζει τη σειρά τους.

Παράδειγμα

Λεξιλόγιο

Τρεις τύποι μηνυμάτων:

1. ***ack*** μήνυμα με θετική αναγνώριση
2. ***nack*** μήνυμα με αρνητική αναγνώριση
3. ***err*** μήνυμα με λάθος μετάδοσης

Μορφοποίηση Μυνημάτων

```
enum control {ack, nack, err};

struct message {
        enum control tag;
        unsigned char data;
};
```

Παράδειγμα

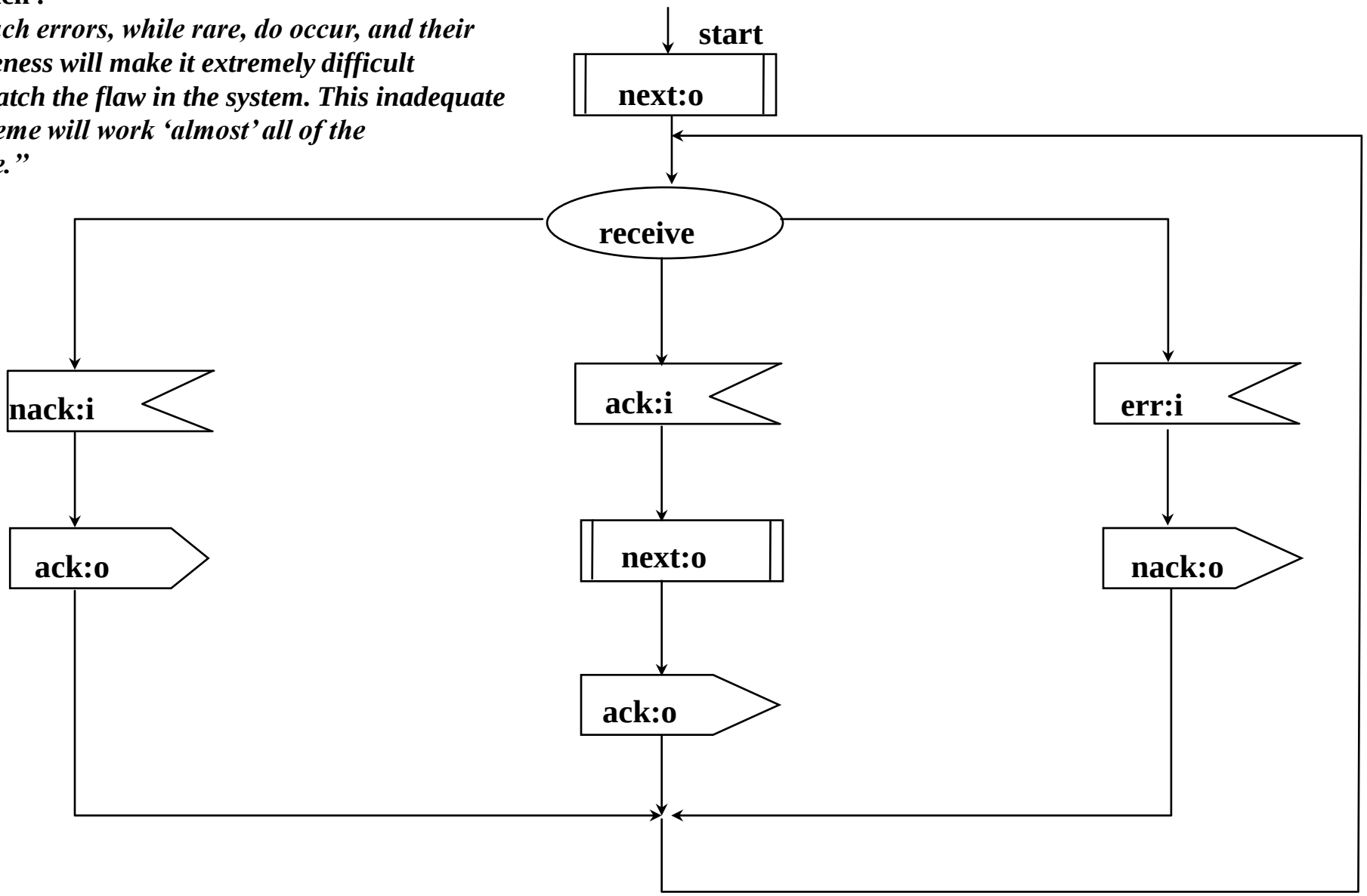
Διαδικαστικοί κανόνες

1. Αν η προηγούμενη λήψη χωρίς λάθη => το επόμενο μήνυμα με ack
2. Αν η προηγούμενη λήψη με λάθος => το επόμενο μήνυμα με nack
3. Αν η προηγούμενη λήψη με nack ή err => ξαναστείλε το παλιό μήνυμα
4. Αν η προηγούμενη λήψη με ack => στείλε ένα καινούργιο μήνυμα

Παράδειγμα

Lynch :

"Such errors, while rare, do occur, and their rareness will make it extremely difficult to catch the flaw in the system. This inadequate scheme will work 'almost' all of the time."

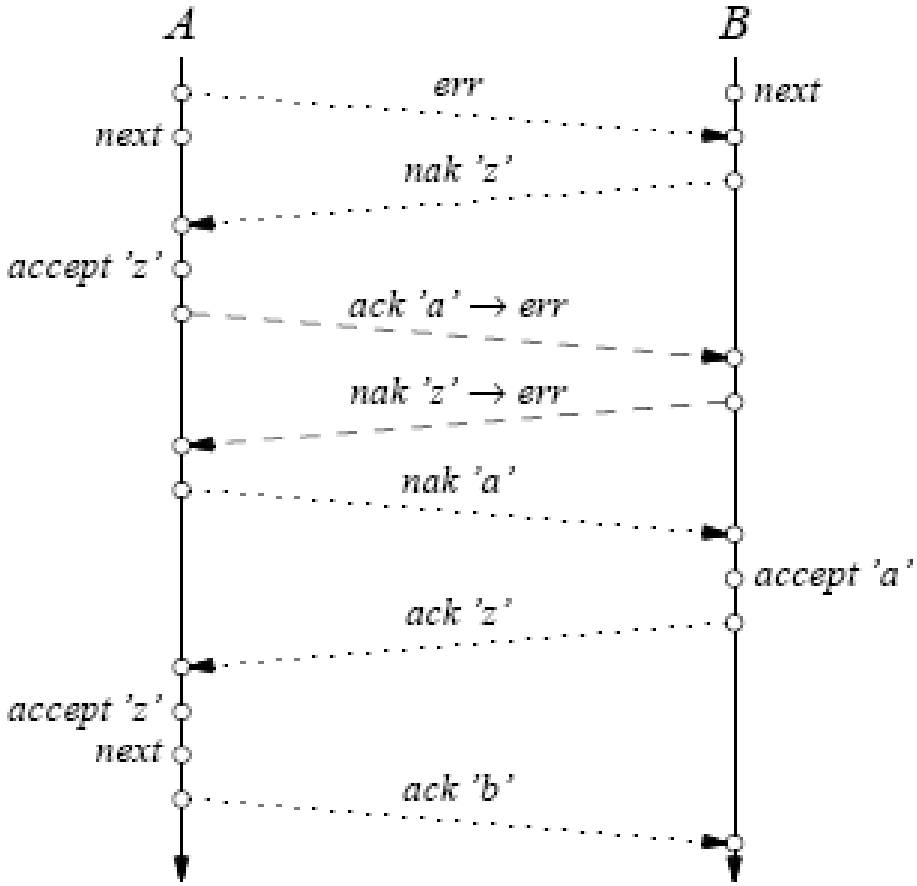


Παράδειγμα

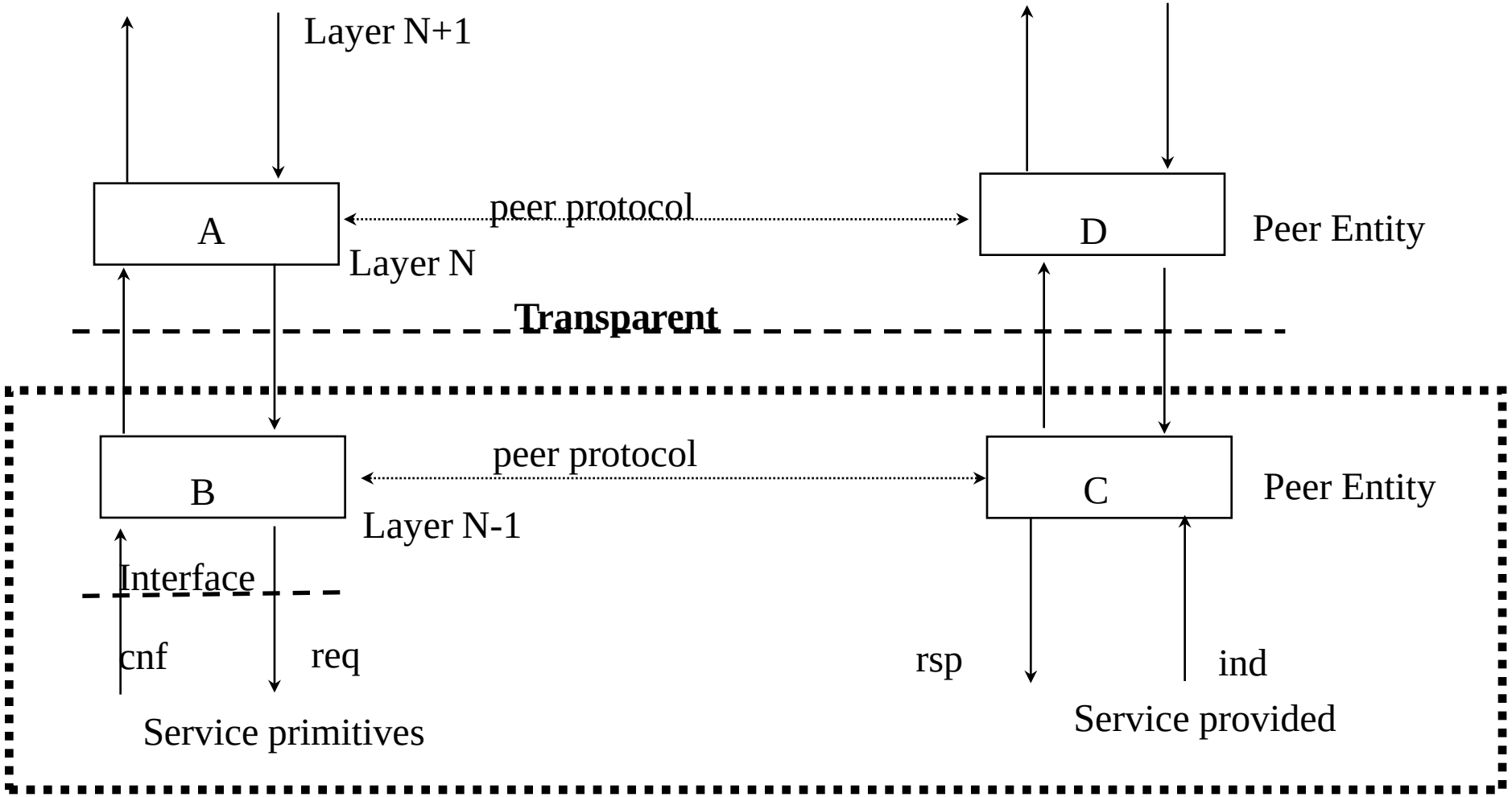
Σχεδιαστικά λάθη

1. Μεταφορά προς τη μία κατεύθυνση μόνο αν γίνεται και από την άλλη
2. Πώς αρχίζει-τελειώνει η μεταφορά
3. Αντίγραφα μηνυμάτων μπορούν να γίνουν αποδεκτά

Π.χ., ας υποθέσουμε οτι ο sender επιχειρεί να στειλει την ακολουθία a-z και ο receiver την z-a



Υπηρεσίες & Περιβάλλον



Virtual Channel

Υπηρεσίες & Περιβάλλον

- **Επίπεδο (Layer):** ομαδοποίηση παρόμοιων διαδικασιών
- **Διασύνσεση (Interface):** σύνολο από service access points (σημεία προσπέλασης υπηρεσίας)
- **SAP:** έχουν μοναδικές διευθύνσεις (π.χ., sockets, message queues identifiers)

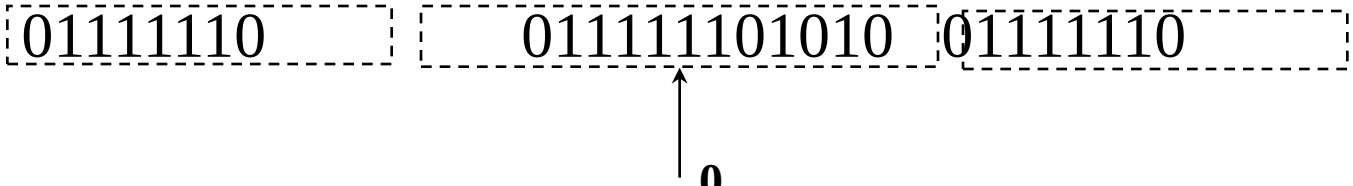
- **Υπηρεσία:** δίνεται στο πιο πάνω επίπεδο

- **Υποθέσεις:** σχετικά με τις υπηρεσίες των πιο κάτω επιπέδων

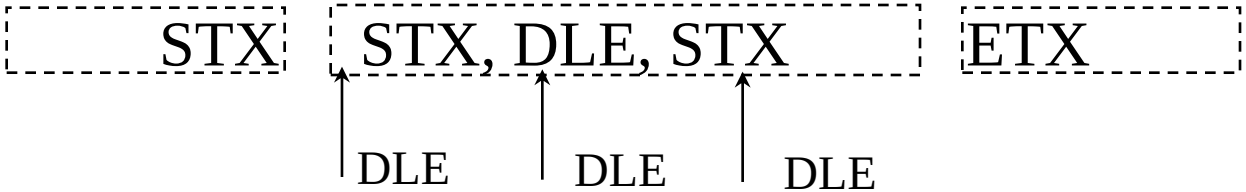
Λεξιλόγιο & Μορφοποίηση

Μέθοδοι μορφοποίησης πρωτοκόλλων

Bit oriented: σειρές από bits



Character oriented: πολλαπλά n bits



character stuffing.

data link escape: IBM's *Bisync* protocol

Byte count oriented: μετά το STX (**start of text**) έχει τον ακριβή αριθμό bytes

Λεξιλόγιο & Μορφοποίηση

Παράδειγμα

format = {header, data, trailer}

header = { type, destination, seq no, count}

trailer = {checksum, return address}

type; dest; seqnr; count

Διαδικαστικοί κανόνες

1. **No Unreachable/Unexecuted Code**
2. **Not Incomplete**
3. **No Deadlocks:** καταστάσεις από τις οποίες όλες οι διεργασίες περιμένουν γεγονότα που δεν μπορούν να συμβούν
4. **No Livelocks:** εκτέλεσεις που μπορούν να συμβούν απείρως συχνά χωρίς ουσιαστική πρόοδο
5. **No Improper Terminations**

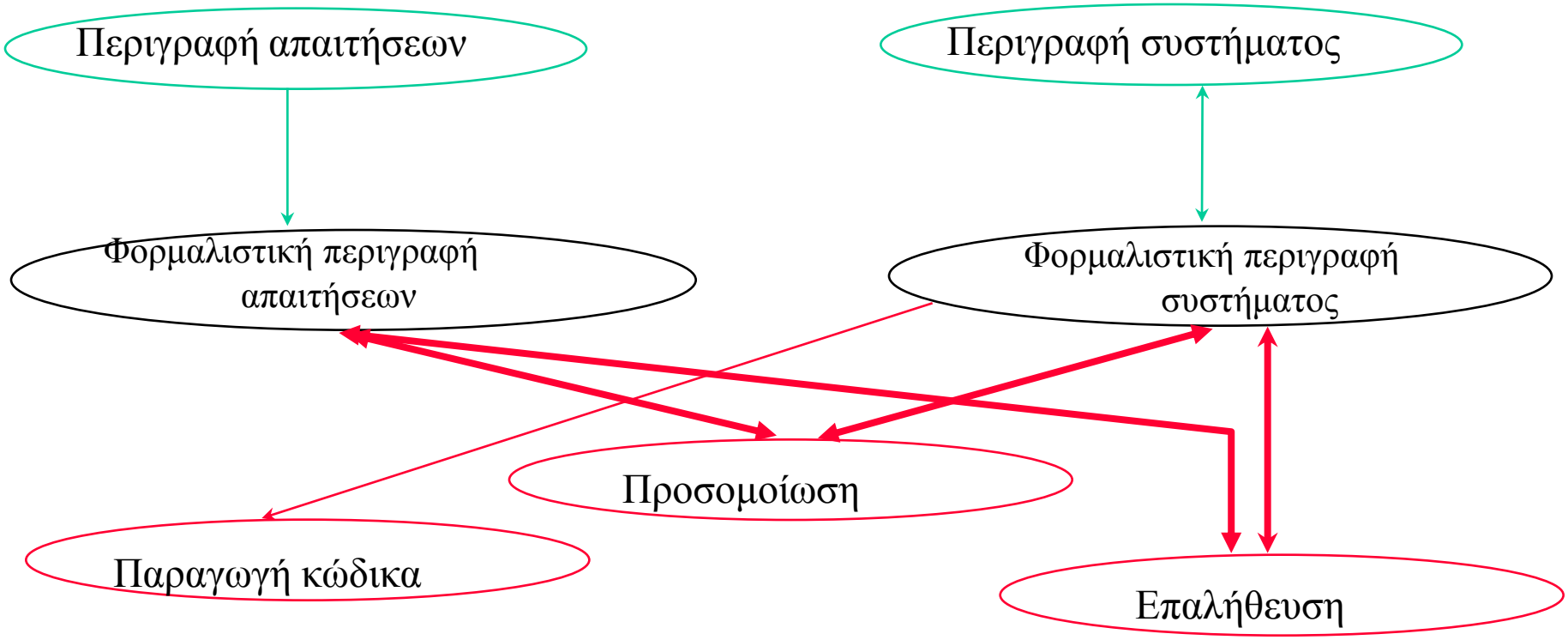
Κανόνες σχεδιασμού πρωτοκόλλων

1. Σαφής προσδιορισμός του προβλήματος. Απαρίθμηση όλων των κριτηρίων, απαιτήσεων και περιορισμών πριν από την έναρξη του σχεδιασμού.
2. Προσδιορισμός της υπηρεσίας που θα παρασχεθεί, σε αφαιρετική στάθμη, προτού αποφασιστεί ποιες δομές θα χρησιμοποιηθούν για υλοποίηση της υπηρεσίας *(το τί προηγείται του πώς)*.
3. Σχεδιασμός της *εξωτερικής* λειτουργικότητας πριν από *την εσωτερική* λειτουργικότητα. Πρώτη θεώρηση της λύσης ως «μαύρο κουτί» (black-box) και απόφαση σχετικά με το πώς θα υφίσταται διάδραση με το περιβάλλον. Έπειτα λήψη απόφασης για την εσωτερική οργάνωση του μαύρου κουτιού και τον καταμερισμό σε μικρότερα ανάλογα δομικά στοιχεία.
4. Διατήρηση της απλότητας. Σύνθετα πρωτόκολλα δεν είναι πάντοτε εύκολα στην υλοποίηση, στην επαλήθευση, ενώ δεν εγγυώνται την αποτελεσματικότητα. Ο σωστός σχεδιασμός μπορεί να βοηθήσει στην αναγνώριση προβλημάτων, στον επιμερισμό και στην επίλυσή τους.
5. Αποφυγή της σύνδεσης ανεξάρτητων τμημάτων. Διαχωρισμός μεταξύ των εννοιών των παραστάσεων σε ορθογώνια κουτιά.

Κανόνες σχεδιασμού πρωτοκόλλων

1. Αποφυγή της χρήσης επουσιωδών τμημάτων. Ένας «καλός σχεδιασμός» είναι εύκολα επεκτάσιμος και επιλύει μάλλον μια κατηγορία προβλημάτων παρά μια μεμονωμένη υπόσταση.
2. Πριν από την υλοποίηση ενός πρωτοκόλλου, συνιστάται σχεδιασμός του σε υψηλό επίπεδο και η επαλήθευση πλήρωσης των κριτηρίων σχεδιασμού.
3. Υλοποίηση του σχεδιασμού, μέτρηση των επιδόσεων, και στο βαθμό που είναι εφικτό, βελτιστοποίηση των επιδόσεων.
4. Έλεγχος αναφορικά με το ότι η τελική βελτιστοποιημένη υλοποίηση είναι ισοδύναμη με το σχεδιασμό υψηλού επιπέδου που έτυχε της επαλήθευσης.
5. Με κανέναν τρόπο δεν θα πρέπει να παραβαίνουμε τους κανόνες 1 έως 7.

Μεθοδολογία



Έλεγχος ροής

- **Βασικές προοπτικές:**
- **Η απλούστερη μορφή μιας διάταξης ελέγχου ροής ελέγχει/ρυθμίζει τον ρυθμό βάσει του οποίου ένας αποστολέας παράγει δεδομένα, σε σχέση με τον ρυθμό που ένας παραλήπτης-δέκτης μπορεί να τα δέχεται.**
- **Ειδικότερες διατάξεις μπορούν επίσης να παρέχουν προστασία έναντι της εξάλειψης, παρεμβολής, αντιγραφής και επαναδιάταξης των δεδομένων.**
- ***Εξέταση της απλούστερης εκδοχής του προβλήματος, που χρησιμοποιείται:***
- **Για να παρέχεται διαβεβαίωση ότι τα δεδομένα δεν αποστέλλονται γρηγορότερα από ό,τι μπορούν να τύχουν επεξεργασίας.**
- **Για να παρέχεται βελτιστοποίηση της χρήσης του καναλιού-μέσου επικοινωνίας.**
- **Για την αποφυγή της υπερφόρτωσης των ζεύξεων μετάδοσης με δεδομένα.**

Έλεγχος ροής

Ο δεύτερος και ο τρίτος στόχος είναι συμπληρωματικοί:

- **Η αποστολή δεδομένων με πάρα πολύ αργό ρυθμό είναι περιττή και άσκοπη…**
- **Η ταχεία αποστολή δεδομένων μπορεί να προκαλέσει συμφόρηση.**

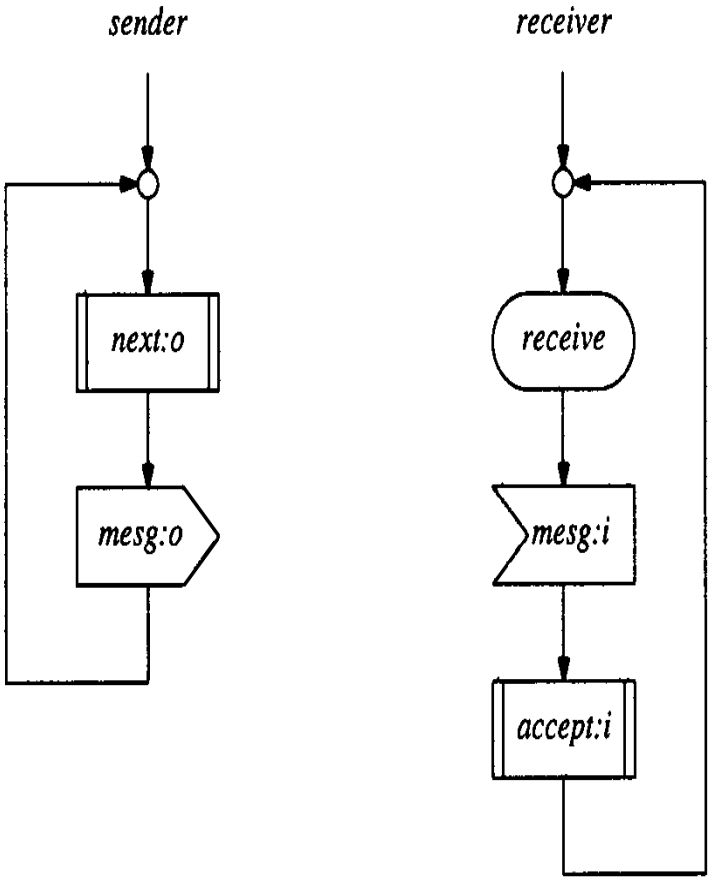
Επισημάνσεις:

- **Η διαδρομή δεδομένων μεταξύ του αποστολέα και του παραλήπτη μπορεί να περιέχει σημεία μεταφοράς με μια περιορισμένη χωρητικότητα, συγκριτικά με την αποθήκευση μηνυμάτων μεταξύ ορισμένων ζευγών αποστολέα-παραλήπτη.**
- **Μια «συνετή» διάταξη ελέγχου ροής παρεμποδίζει ένα τέτοιο «ζεύγος» από το να καταλαμβάνει όλο το διαθέσιμο διάστημα αποθήκευσης.**

Παράδειγμα

Ένα «απλό» πρωτόκολλο χωρίς καμία μορφή ελέγχου ροής, που χρησιμοποιείται για μεταφορά δεδομένων μόνο κατά τη μία κατεύθυνση (*simplex*).

Παράδειγμα



No Flow Control

Παρατηρήσεις:

Το πρωτόκολλο λειτουργεί αξιόπιστα, μόνο εάν παρέχονται εγγυήσεις ότι η διεργασία του παραλήπτη είναι ταχύτερη από εκείνη του αποστολέα.

Μπορούμε να έχουμε τέτοιες εγγυήσεις?

X-on / X-off

- **Η παλαιότερη (αλλά και λιγότερο αξιόπιστη) τεχνική ελέγχου ροής που μπορεί να χρησιμοποιείται για την αντιμετώπιση αυτού του προβλήματος συγχρονισμού, δεν απαιτεί καμία προηγούμενη διαπραγμάτευση μεταξύ αποστολέα και παραλήπτη, σχετικά με τον ρυθμό που μπορούν να εκπέμπονται τα δεδομένα.**

- **Η μέθοδος χρησιμοποιεί δύο μηνύματα ελέγχου:**
- **ένα μήνυμα για *αναστολή* (*suspend)* κυκλοφορίας και**
- **ένα μήνυμα για *συνέχιση* (*resume)* κυκλοφορίας.**

- **Τα μηνύματα αυτά μερικές φορές αποκαλούνται ως *x-off* και *x-on*.**

X-on / X-off

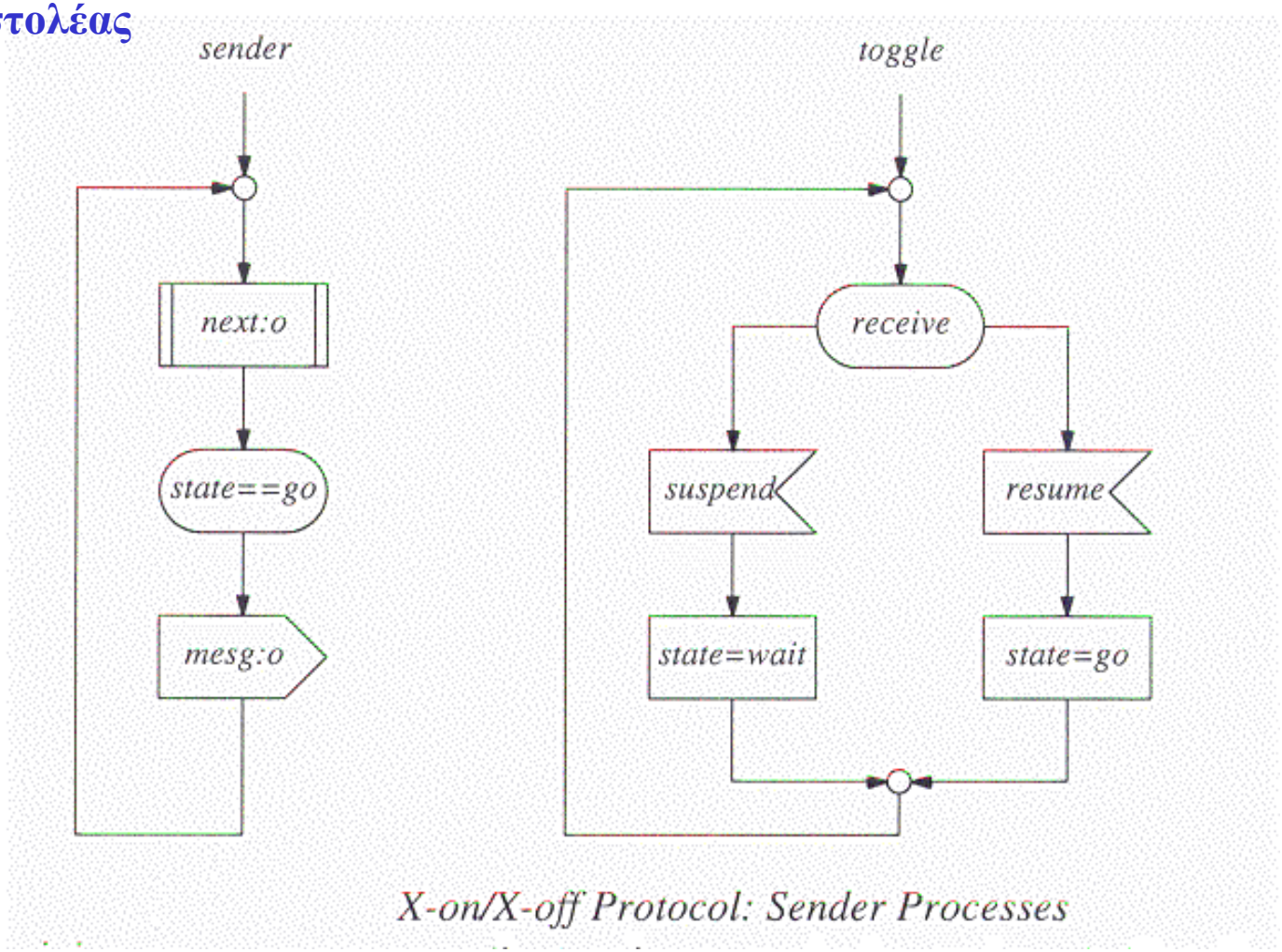
- **Ας υποτεθεί ότι έχουμε ένα κανάλι που δεν δημιουργεί σφάλματα (error-free channel) και ένα λεξιλόγιο πρωτοκόλλου με τους ακόλουθους τρεις τύπους μηνυμάτων:**
- *V = { mesg, suspend, resume }*

- **Τα δύο βασικά μηνύματα ελέγχου *αναστολή* (*suspend)* και *συνέχιση (resume)* χρησιμοποιούνται για την υλοποίηση της μεθόδου ελέγχου ροής.**

- **Οι διαδικαστικοί κανόνες του πρωτοκόλλου μπορούν τώρα να προστεθούν.**
- **Η υλοποίηση περιγράφεται με δύο πρόσθετες διαδικασίες: μια στον αποστολέα και μια στον παραλήπτη.**

X-on / X-off

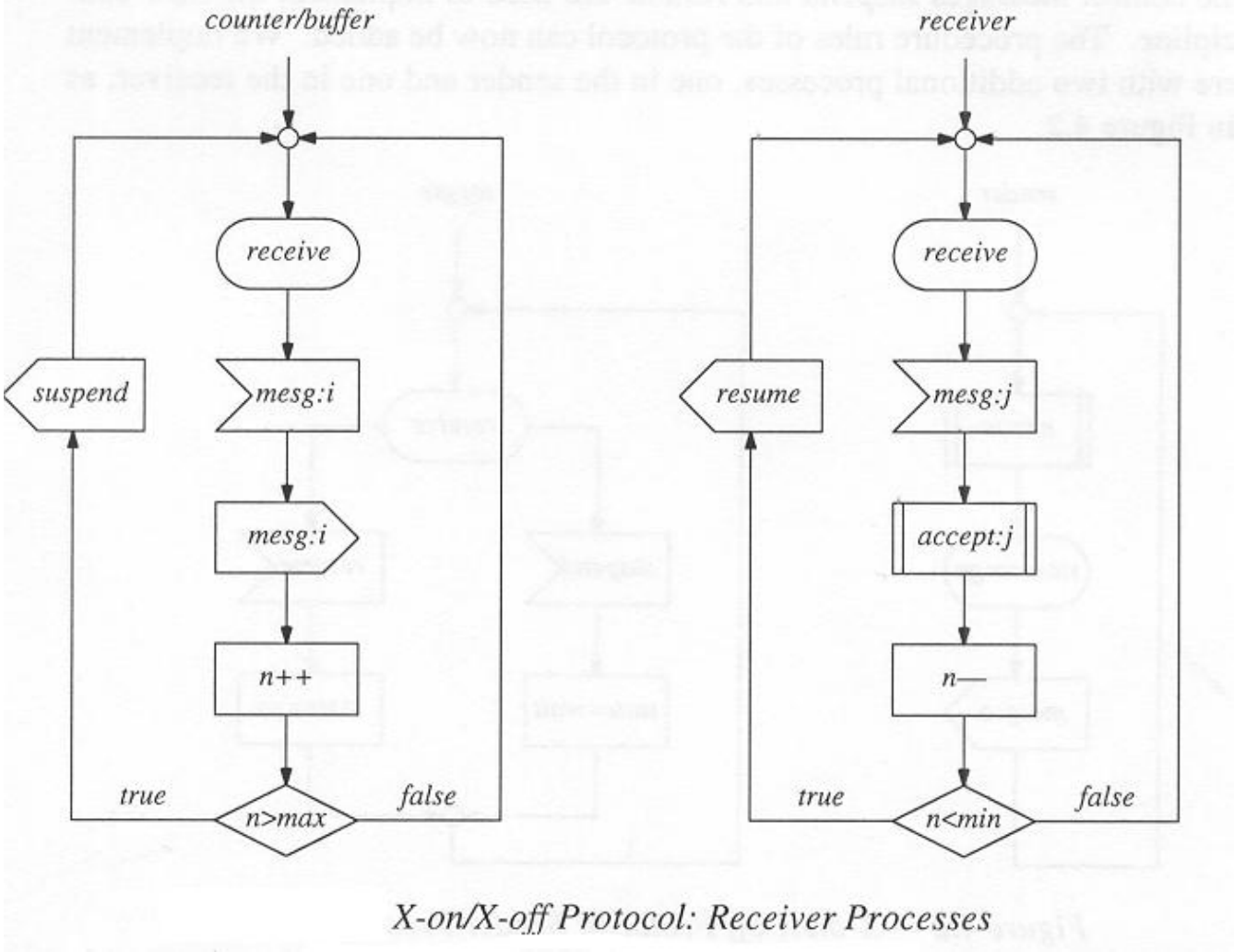
Αποστολέας



X-on/X-off Protocol: Sender Processes

X-on / X-off

Παραλήπτης



X-on/X-off Protocol: Receiver Processes

X-on / X-off

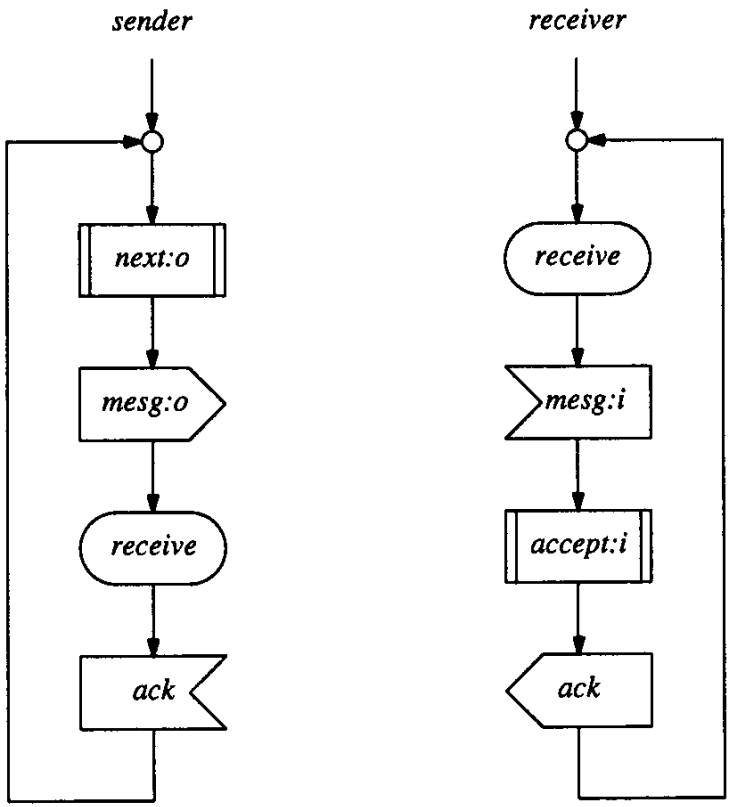
- ***Βασικά προβλήματα προς επίλυση:***

- **Η ορθή λειτουργία του πρωτοκόλλου εξαρτάται από τις ιδιότητες του καναλιού μετάδοσης. (Εάν χαθεί ή ακόμα καθυστερήσει ένα μήνυμα *αναστολή (suspend)*).**
- **Η λειτουργία ενός πρωτοκόλλου δεν θα πρέπει να εξαρτάται από τον χρόνο που χρειάζεται για να φθάσει στον παραλήπτη ένα μήνυμα ελέγχου.**
- **Σε ακόμα χειρότερη περίπτωση, εάν χάνεται ένα μήνυμα *συνέχιση (resume)*, το σύστημα επεξεργασίας το οποίο αποτελείται από τέσσερις επιμέρους διεργασίες, οριστικά σταματάει.**

- **Εντοπισμός προβλημάτων:**

1. **Προστασία έναντι προβλημάτων καθυστέρησης κατά τη μετάδοση (overrun errors), κατά έναν περισσότερο αξιόπιστο τρόπο.**
2. **Προστασία έναντι απώλειας μηνυμάτων.**

Stop and Wait



Ping-Pong Protocol

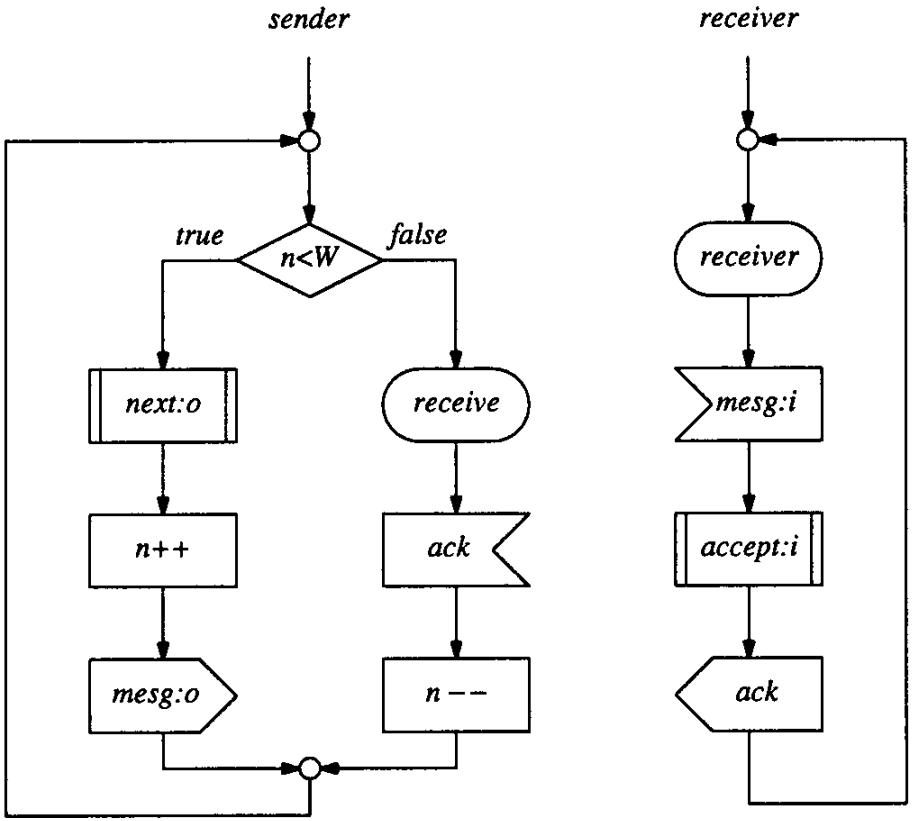
Τι λύσαμε?
Τι παραμένει ως πρόβλημα?

Το θέμα συγχρονισμού
Το πρόβλημα που προκύπτει από την απώλεια

Window Protocols

- call-setup phase: ο παραλήπτης μπορεί να αναφέρει στον αποστολέα το buffer space για τα εισερχόμενα μηνύματα.
- Στον αποστολέα τότε χορηγείται μια μορφή πίστωσης (credit) για έναν σταθερό αριθμό εκκρεμών μηνυμάτων (outstanding messages).
- Η πίστωση μπορεί να αναθεωρηθεί, δυναμικά, όταν μεταβάλλεται το μέγεθος της διαθέσιμης μνήμης προσωρινής αποθήκευσης.
- Κάθε μήνυμα που λαμβάνεται, γνωστοποιείται με ένα απλό μήνυμα ελέγχου *ack*, πάνω σε ένα κανάλι επιστροφής.
- Πρέπει να διατηρηθεί ο αριθμός των μηνυμάτων που έχουν αποσταλεί μέσα στο καθορισμένο πλαίσιο W

Window Protocols



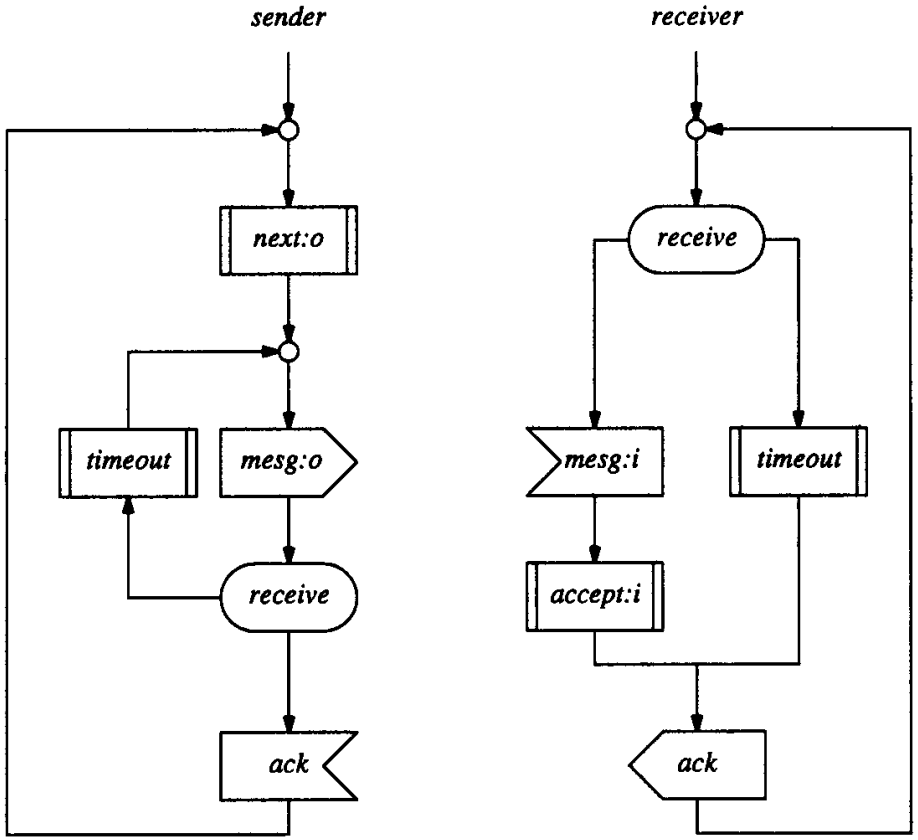
Window Protocol for an Ideal Channel

Window Protocols

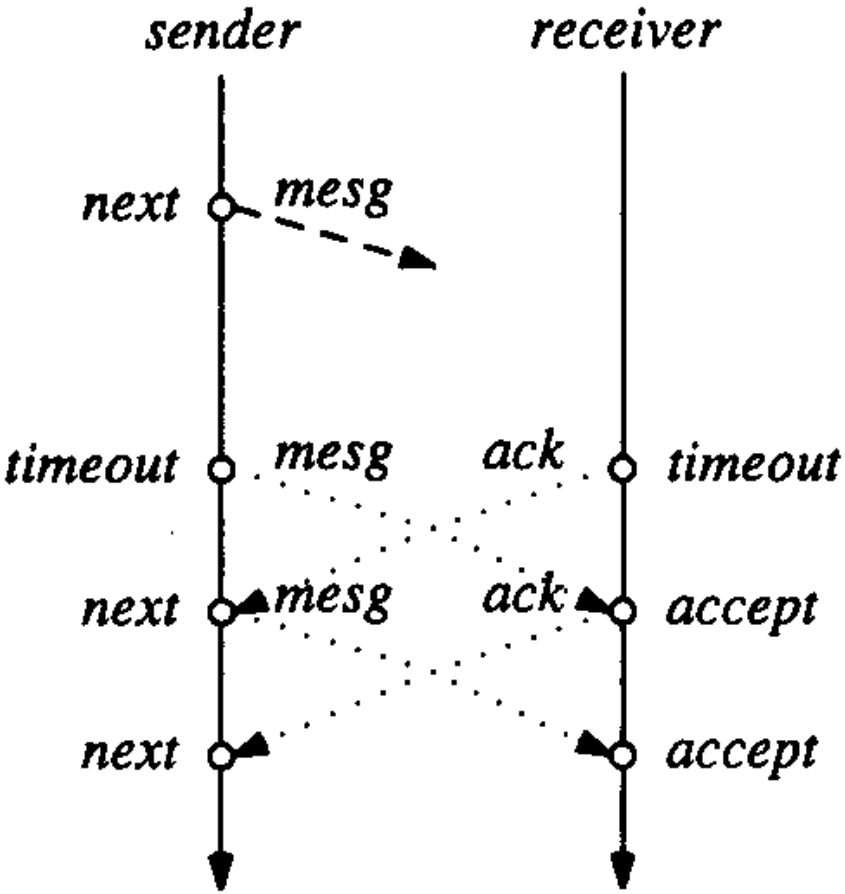
- Η μέγιστη πίστωση W αποκαλείται ως μέγεθος παραθύρου (window size) του πρωτοκόλλου.
- Κατά τη διάρκεια μιας μεταφοράς, η τρέχουσα πίστωση ποικίλει μεταξύ των τιμών μηδέν και W, ανάλογα με τις σχετικές ταχύτητες του αποστολέα και του παραλήπτη.
- Ο αποστολέας υπόκειται σε καθυστέρηση μόνο όταν η πίστωση μειώνεται σε μηδέν.
- Αυτή η μέθοδος ελέγχου ροής μπορεί να βελτιστοποιήσει την επικοινωνία πάνω σε κανάλι με μεγάλες διαβιβαστικές καθυστερήσεις (transit delays)
- Ποιο πρόβλημα εξακολουθεί να παραμένει?

Stop and Wait με timeouts

Για την αντιμετώπιση της πιθανής απώλειας μηνυμάτων: εισαγωγή χρονιστή



Stop and Wait με timeouts



Alternating Bit Protocol

Χρησιμοποιούνται δύο τύποι μηνυμάτων, *mesg* και *ack*, π.χ. με τη μορφή

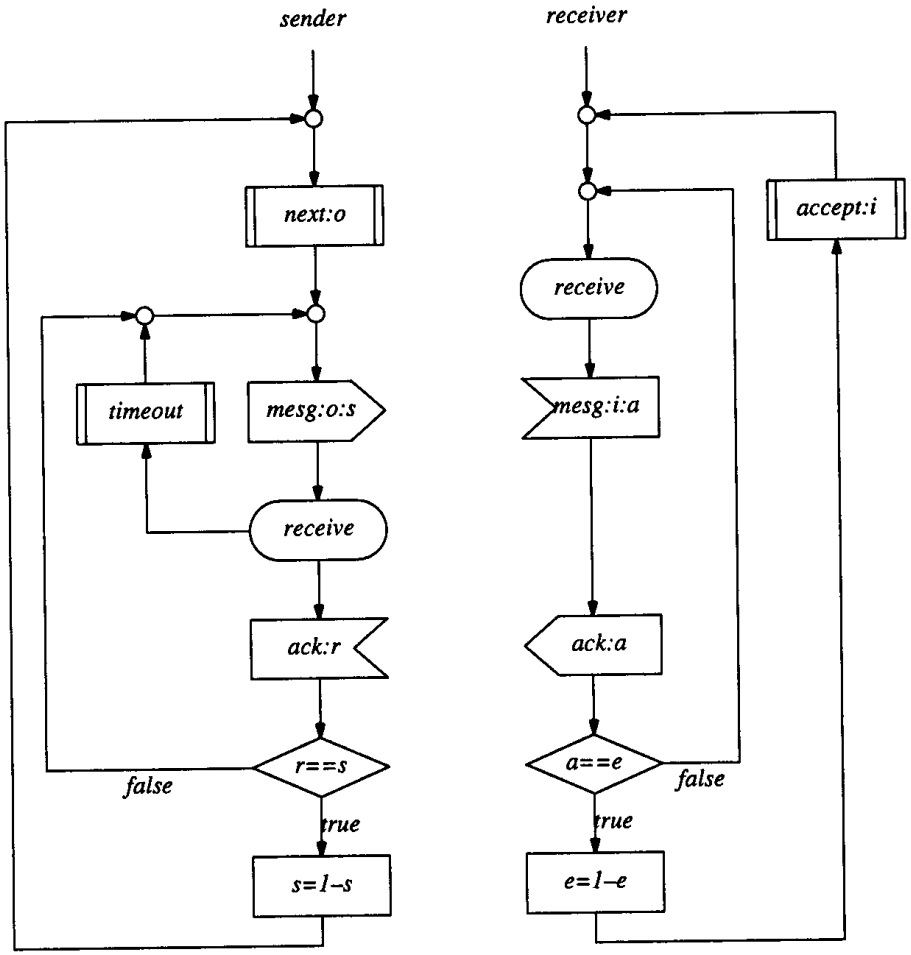
{ mesg, data, sequence number }

και

{ ack, sequence number }

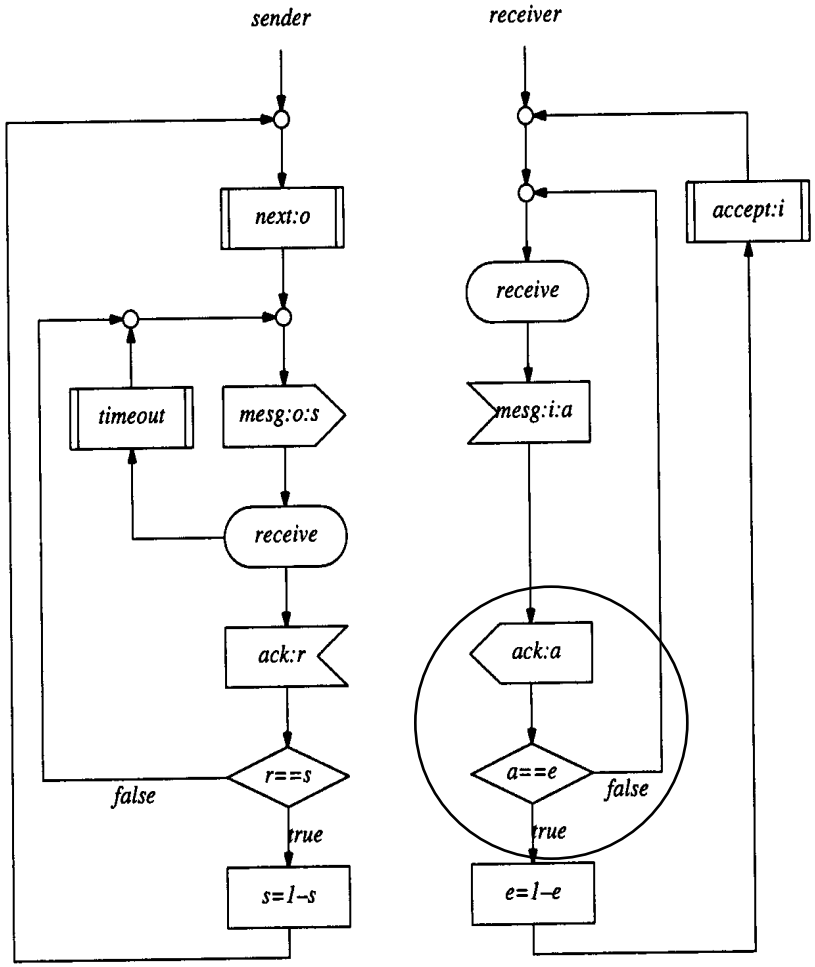
Στο διάγραμμα ροής, το *mesg:o:s*
υποδηλώνει ένα μήνυμα *mesg* με πεδίο δεδομένων *o*
και πεδίο αριθμού ακολουθίας *s*.

Alternating Bit Protocol



Alternating Bit Protocol with Timeouts

Alternating Bit Protocol



Alternating Bit Protocol with Timeouts

Χρησιμοποιούνται τέσσερις single-bit μεταβλητές: *a*, *e*, *r*, και *s*.

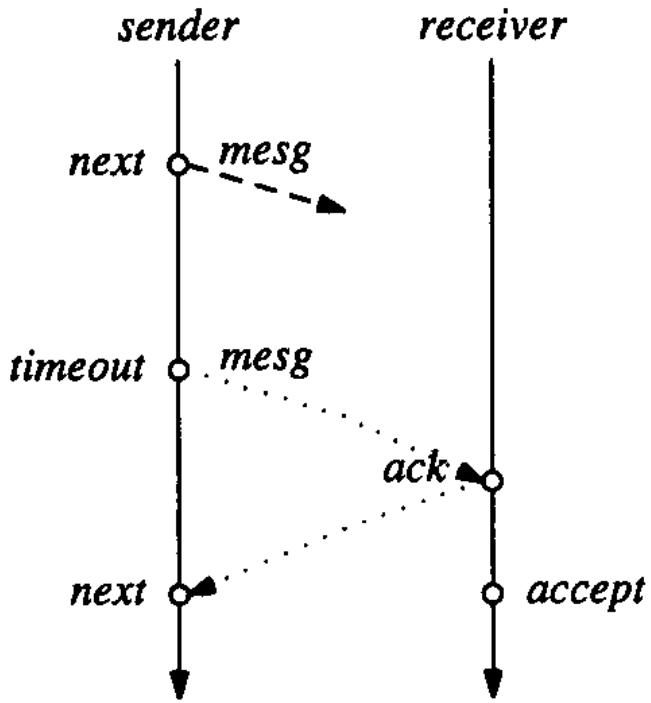
Η μεταβλητή *s* χρησιμοποιείται από τον αποστολέα για να αποθηκεύσει τον τελευταίο αριθμό ακολουθίας που εστάλη,

το *r* διατηρεί τον τελευταίο αριθμό ακολουθίας που ελήφθη.

Ο παραλήπτης χρησιμοποιεί το *e* για να συγκρατήσει τον επόμενο αριθμό που αναμένεται να αφιχθεί, και την μεταβλητή *a* για την αποθήκευση του τελευταίου πραγματικού αριθμού ακολουθίας που έχει ληφθεί.

Όλες οι μεταβλητές έχουν μια αρχική τιμή μηδέν.

Alternating Bit



Time Sequence Diagram of Error

Window Protocols

- Περίπτωση αντιγραφής (duplication) και επαναδιάταξης (reordering) μηνυμάτων π.χ. σε datagram networks.

Προφανής λύση:

- Η χρήση ενός «μεγάλου» αριθμού ακολουθίας, ο οποίος προσαρτάται σε κάθε μήνυμα. Π.χ., 16 bits→65.536 διαφορετικά πακέτα
- Σε πόση ώρα τελειώνουν τα διακριτά πακέτα αν έχω πακέτα των 1500 Bytes σε μία 2Mbps γραμμή; Η' αλλιώς ποιο είναι το μεγαλύτερο αρχείο που μπορώ να στείλω; (6,54 λεπτά περίπου και 99 MB)

Sliding Window Protocol

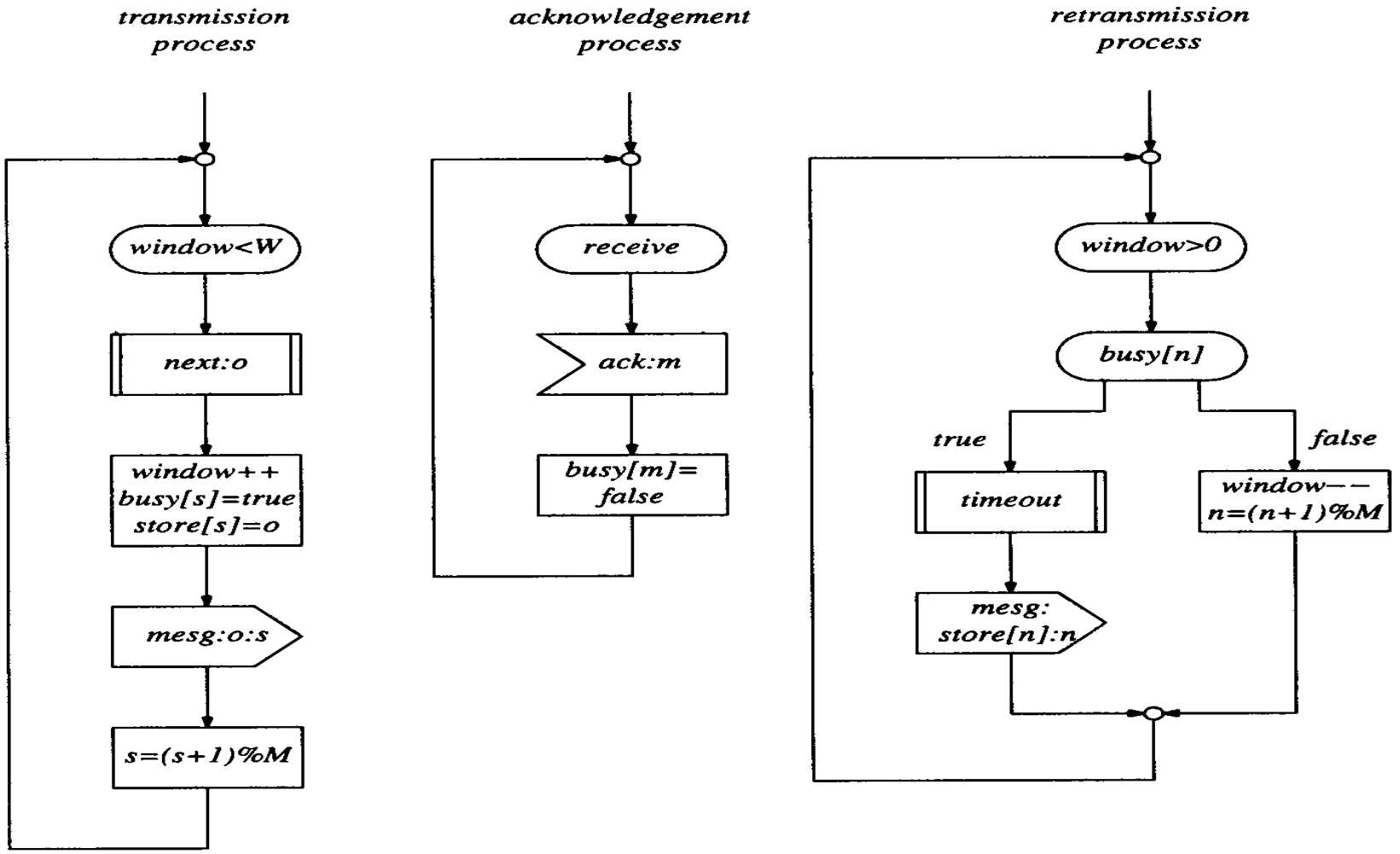
- Θεωρήσεις:
- Μια περιοχή-εύρος M των διαθέσιμων αριθμών ακολουθίας και μια αρχική πίστωση, W, μηνυμάτων.
- Το M είναι επαρκώς μεγαλύτερο από το W, για την αποφυγή σύγχυσης με τους ανακυκλωμένους αριθμούς ακολουθίας (recycled sequence numbers).
- Ο αποστολέας πρέπει να θυμάται για κάθε μήνυμα σε εκκρεμότητα.

- Χρησιμοποιούμε δύο πίνακες για αυτόν το σκοπό.
- busy[s] → true εάν ένα μήνυμα με αριθμό ακολουθίας s εστάλη και δεν έχει ακόμα γνωστοποιηθεί.
- store[s] αποθηκεύουμε το μήνυμα με αριθμό ακολουθίας s, που εκπέμφθηκε.

Sliding Window Protocol

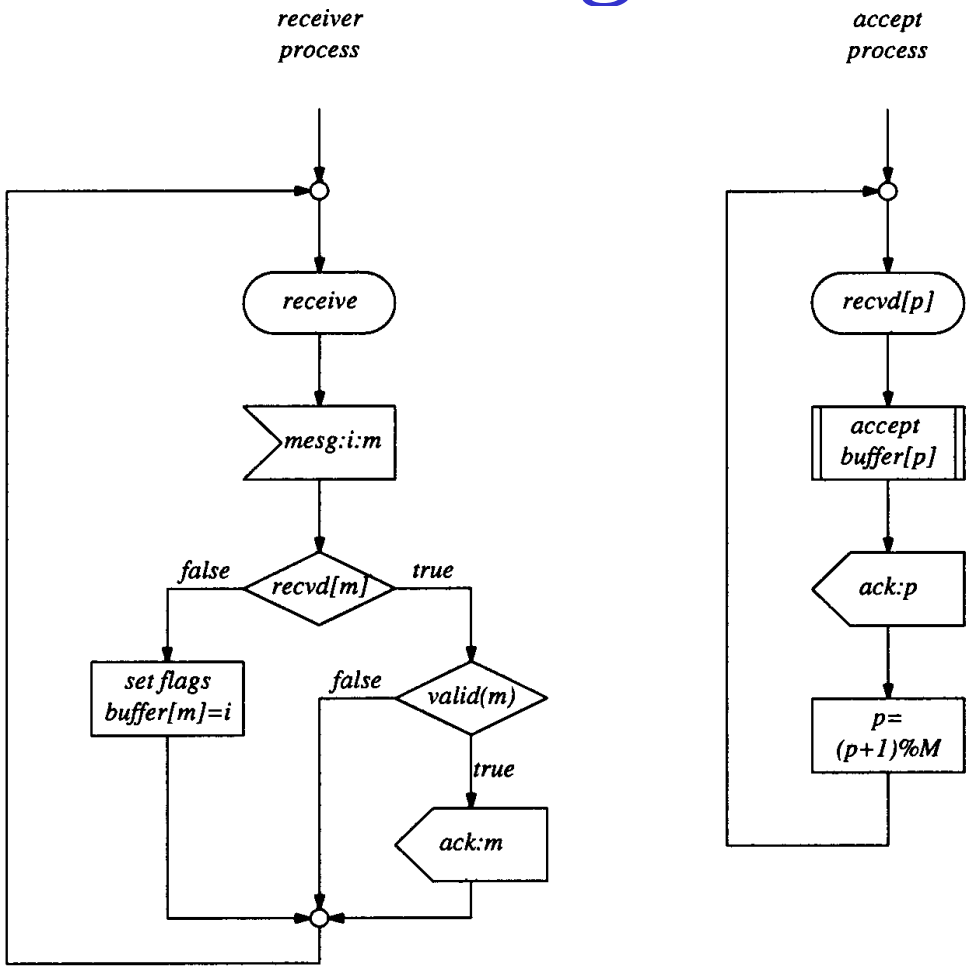
- Επιπλέον με τις σταθερές W και M, χρησιμοποιούνται οι ακόλουθες τέσσερις μεταβλητές (όλες με μια αρχική τιμή μηδέν):
- s: ο αριθμός ακολουθίας του επόμενου μηνύματος που πρόκειται να αποσταλεί
- window: ο αριθμός των σε εκκρεμότητα μηνυμάτων που δεν έχουν επιβεβαιωθεί
- n: ο αριθμός ακολουθίας (sequence number) του παλαιότερου μηνύματος που δεν έχει επιβεβαιωθεί
- m: ο αριθμός ακολουθίας του τελευταίου μηνύματος που έχει επιβεβαιωθεί

Sliding Window Protocol



— *Sender Processes, Sliding Window Protocol*

Sliding Window Protocol

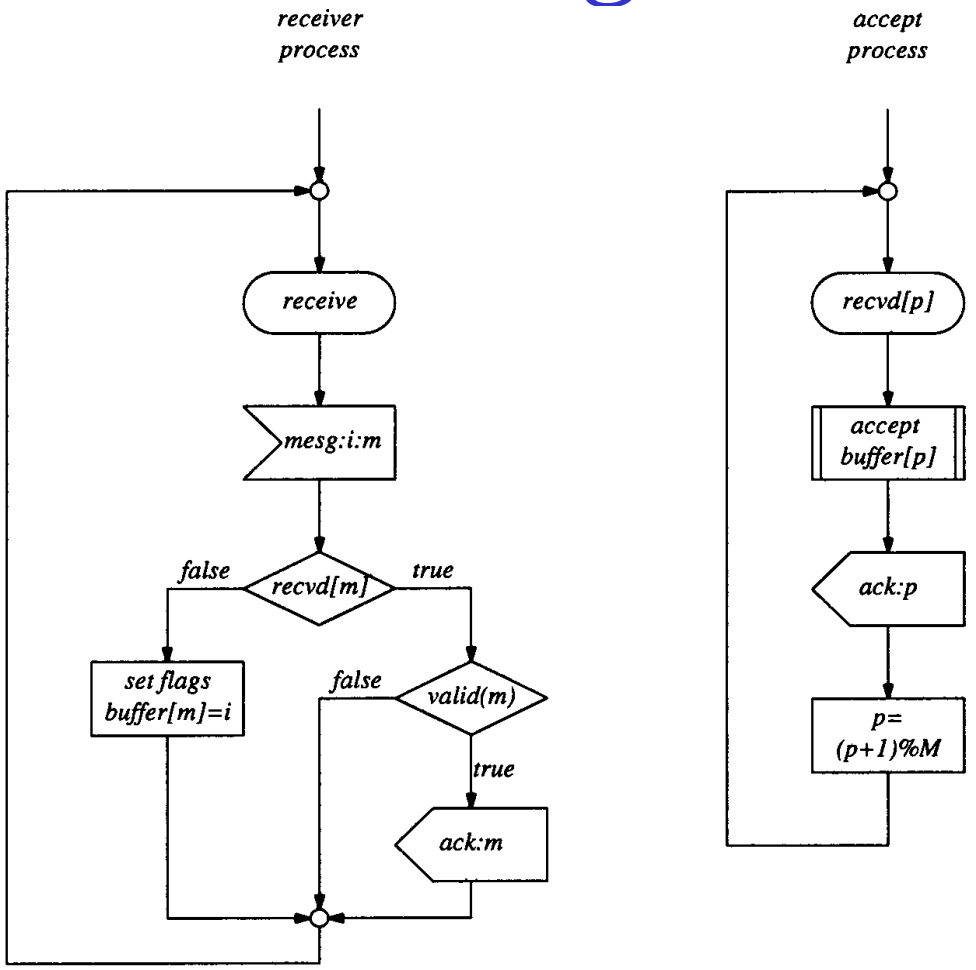


— Receiver Processes, Sliding Window Protocol

- Recvd[m]: αριθμός ακολουθίας μηνυμάτων που έχουν ληφθεί αλλά δεν έχουν γίνει ακόμα αποδεκτά
- Buffer[m]: φυλάσσονται τα περιεχόμενα αυτών των μηνυμάτων
- P: αριθμός ακολουθίας του επόμενου μηνύματος που πρέπει να γίνει αποδεκτό

- Flags: i) recvd[m]=true
  ii) recvd[(m-W)%M]=false

Sliding Window Protocol



— Receiver Processes, Sliding Window Protocol

- Θεωρούμε modulo M
- Αν δεν είχαμε modulo τότε→ valid(m) = m<p
- Valid(m) = (0<p-m≤W) || (0<p+M-m≤W)

Αποστολή εκ νέου της επιβεβαίωσης μόνο αν έχει γίνει αποδεκτό και απλά χάθηκε η αρχική επιβεβαίωση

Αν δεν έχει γίνει αποδεκτό δεν στέλνουμε επιβεβαίωση

Negative ack

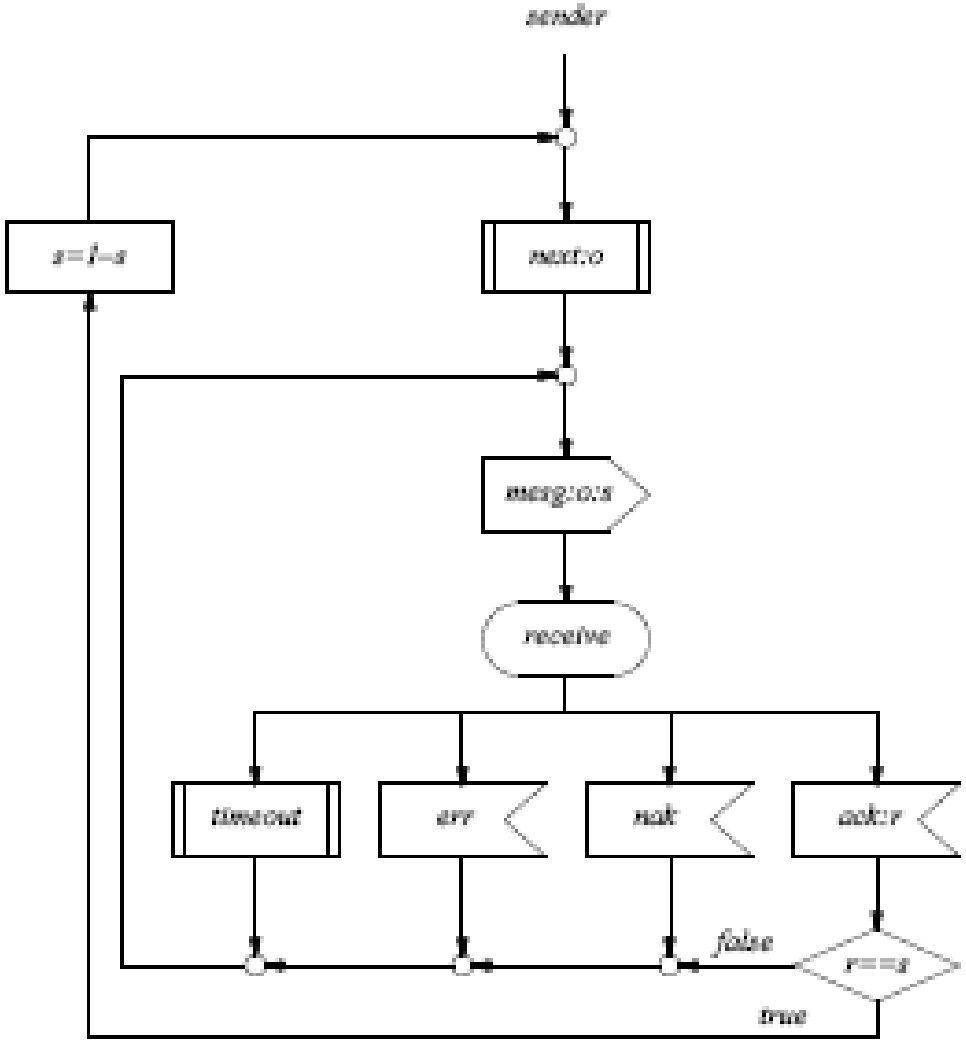


Figure 4.13 — Sender, Extended Alternating Bit Protocol

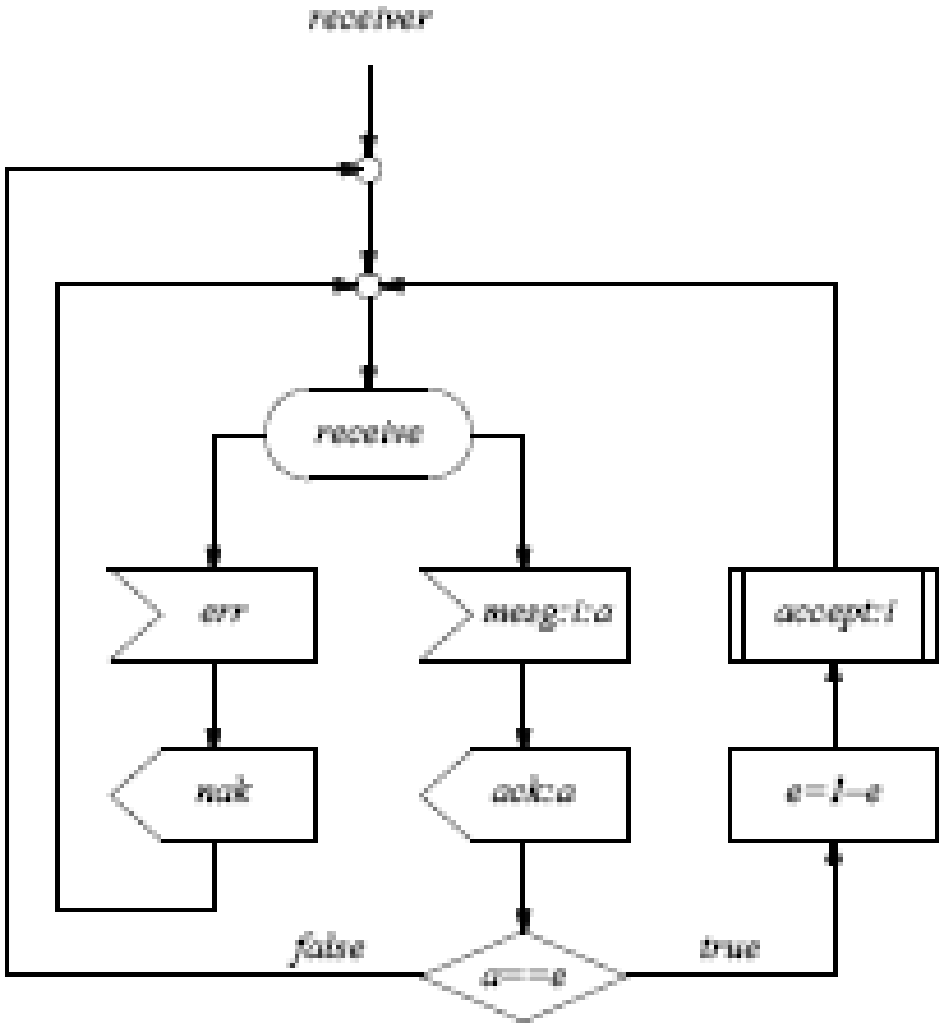
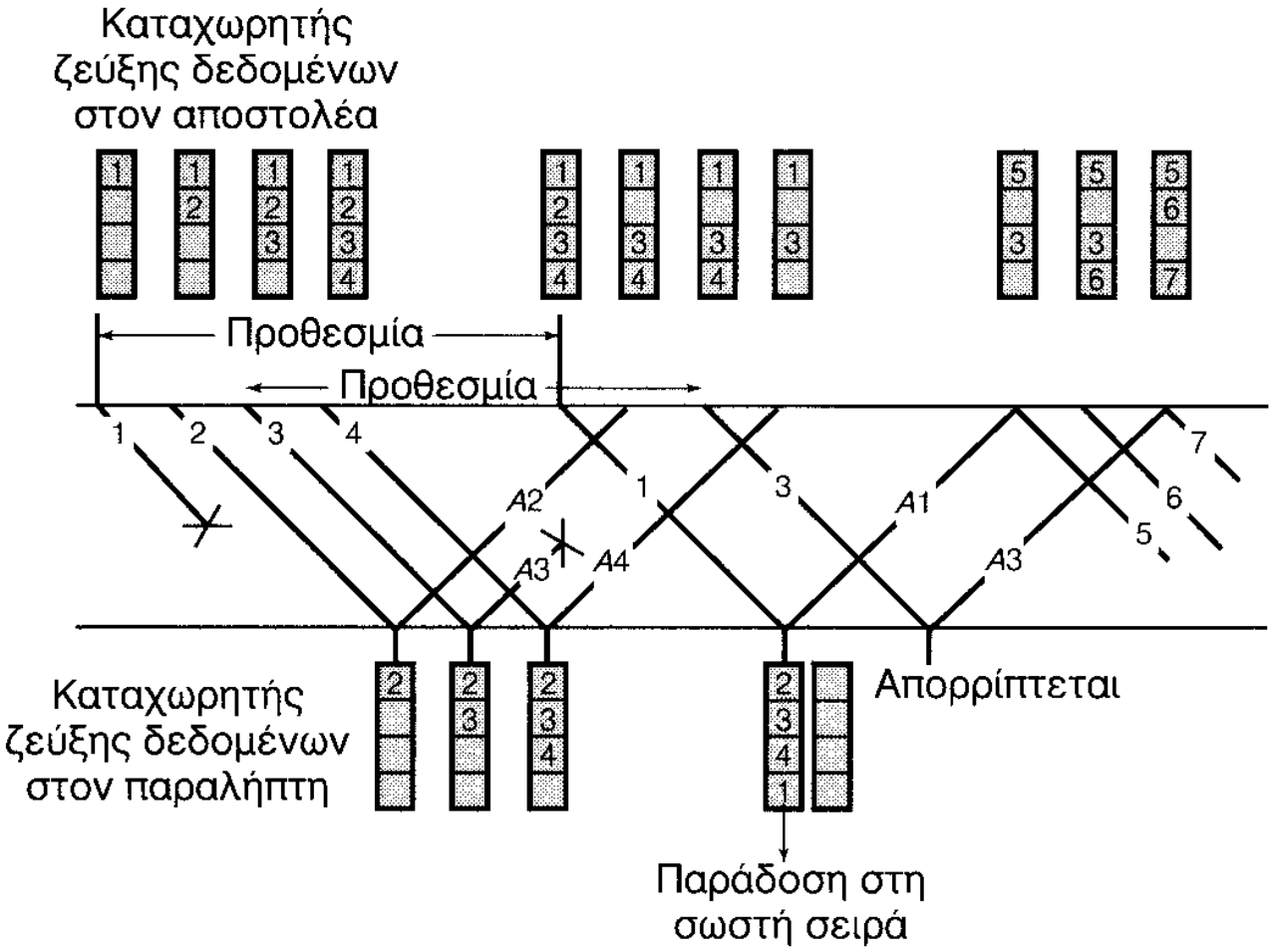


Figure 4.14 — Receiver, Extended Alternating Bit Protocol

- The method of using acknowledgments to control the retransmission of messages is usually referred to as an *ARQ method, where ARQ stands for Automatic Repeat* Request. There are three main variants:
- Stop-and-wait ARQ
- Selective repeat ARQ
- Go-back-N continuous ARQ

Πρωτόκολλο επιλεκτικής επανάληψης

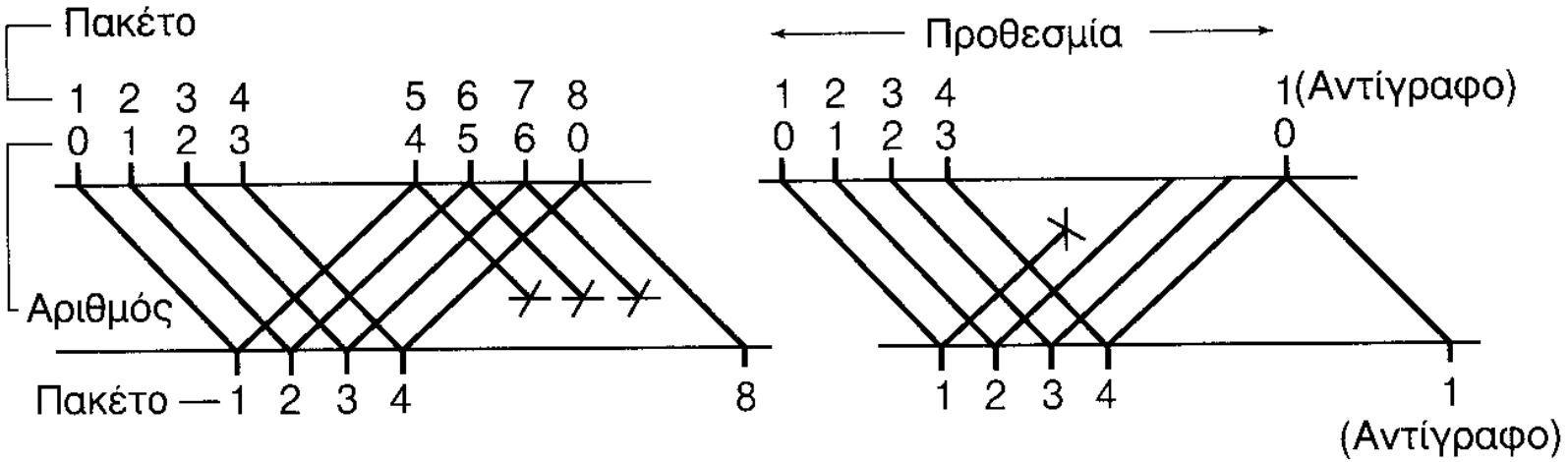


Πρωτόκολλο επιλεκτικής επανάληψης

1. Τα πακέτα αριθμούνται διαδοχικά 1,2,3, κ.ο.κ
2. Οι αριθμοί των ανεπιβεβαίωτων πακέτων διαφέρουν λιγότερο από W
3. Ο αποστολέας αποθηκεύει αντίγραφα των ανεπιβεβαίωτων πακέτων και τα επαναμεταδίδει μετά τη λήξη του χρόνου αναμονής
4. Ο παραλήπτης αποθηκεύει αντίγραφα των πακέτων που λαμβάνει εκτός σειράς και επιβεβαιώνει όλα τα σωστά πακέτα. Παραδίδει W πακέτα που είναι στη σωστή σειρά.

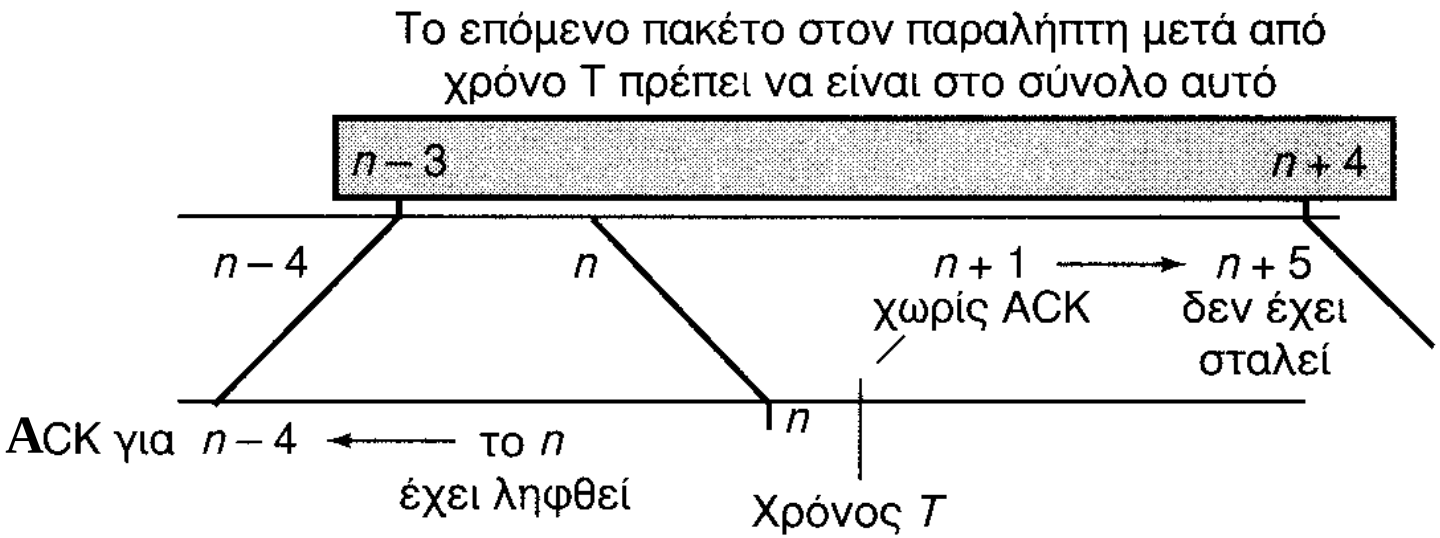
Πρωτόκολλο επιλεκτικής επανάληψης

- Το SRP με αρίθμηση mod 7 και W =4



Πρωτόκολλο επιλεκτικής επανάληψης

Αρίθμηση modulo 8 & n το τελευταίο πακέτο που παραδόθηκε στο NL του παραλήπτη

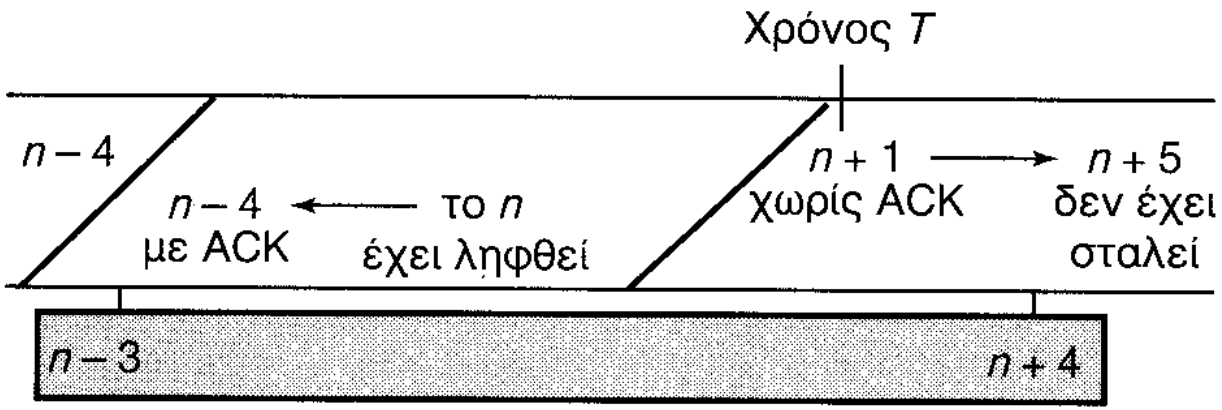


Π.χ., n=3 τότε OLD(3) = 0,1,2,3 και NEW(3) = 4,5,6,7

Αν Αρίθμηση modulo 7 για n=3 τότε OLD(3) = 0,1,2,3 και NEW(3) = 4,5,6,0

Πρωτόκολλο επιλεκτικής επανάληψης

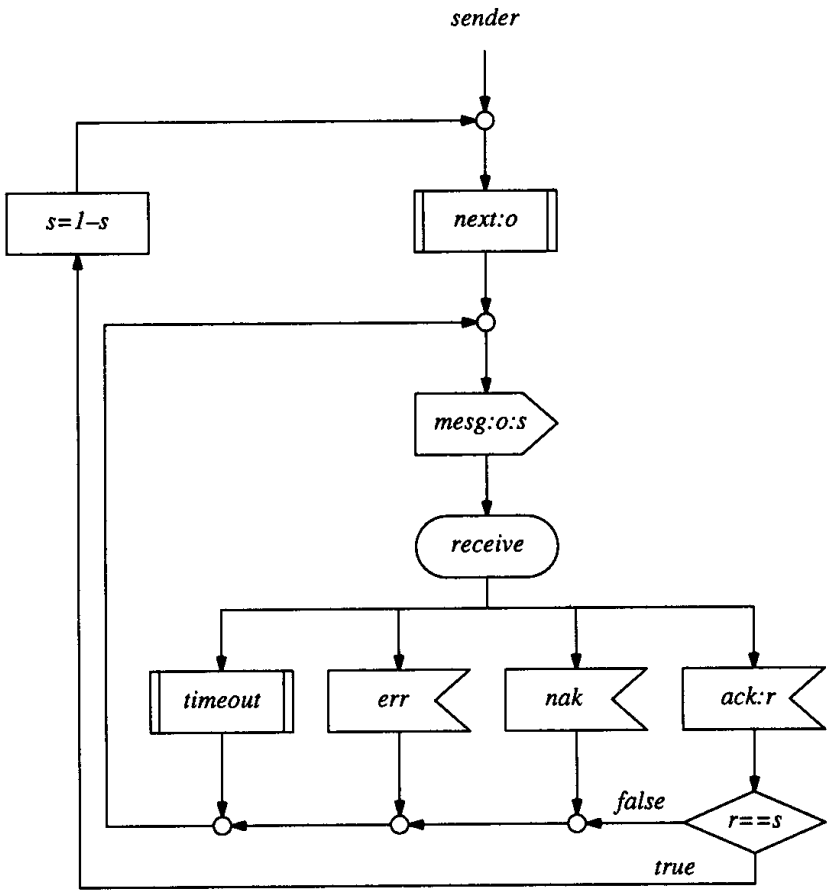
Αρίθμηση modulo 8 και ο αποστολέας έχει λάβει το ACK n



Η επόμενη ACK στον αποστολέα μετά από χρόνο T πρέπει να είναι στο σύνολο αυτό

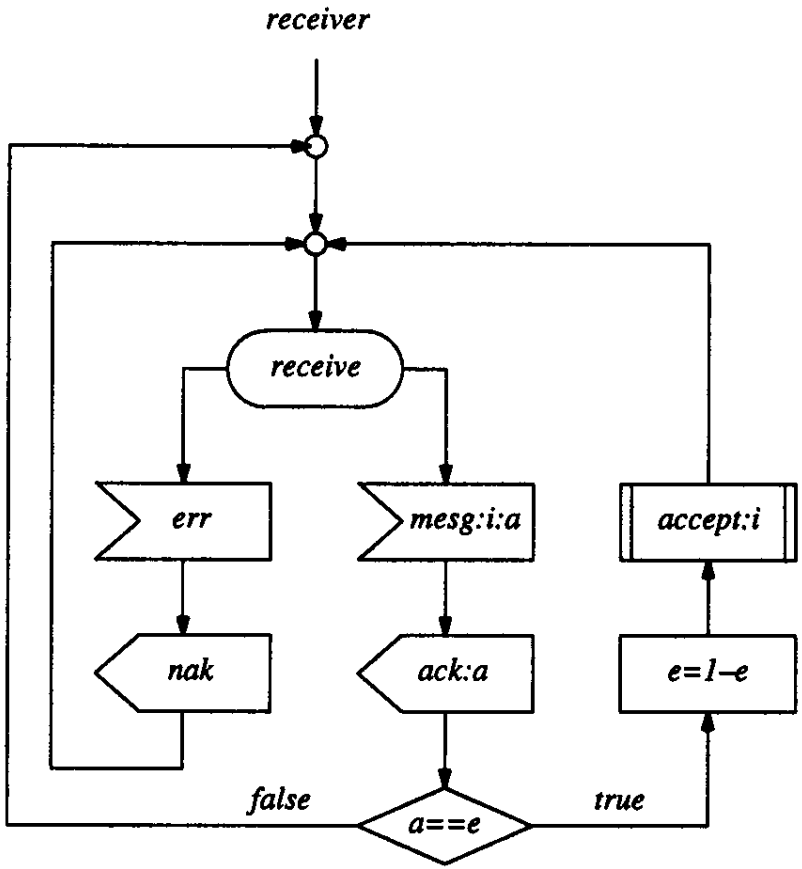
Αρνητικές επιβεβαιώσεις

Τι μπορούν να προσφέρουν οι αρνητικές επιβεβαιώσεις;



Sender, Extended Alternating Bit Protocol

Αρνητικές επιβεβαιώσεις



Receiver, Extended Alternating Bit Protocol