ΠΑΝΕΠΙΣΤΉΜΙΟ ΠΑΤΡΏΝ

ΠΟΛΥΤΕΧΝΙΚΉ ΣΧΟΛΉ

ΤΜΉΜΑ ΜΗΧΑΝΙΚΏΝ ΗΛΕΚΤΡΟΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ & ΠΛΗΡΟΦΟΡΙΚΉΣ

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

ΓΙΑ ΤΟ ΜΑΘΗΜΑ

ΔΙΚΤΥΑ ΔΗΜΟΣΙΑΣ ΧΡΗΣΗΣ ΚΑΙ ΔΙΑΣΥΝΔΕΣΗ ΔΙΚΤΥΩΝ

Σχεδιασμός Πρωτοκόλλων

ΑΡΒΑΝΙΤΗΣ ΝΙΚΟΣ

4686

ΔΙΔΆΣΚΩΝ: ΧΡΉΣΤΟΣ ΜΠΟΎΡΑΣ

ΠΆΤΡΑ 2014

ΠΕΡΙΕΧΌΜΕΝΑ

Κεφάλαιο 1: Εισαγωγη

1.1 Τι είναι ένα πρωτόκολο

Μέσα στην καθημερινότητα και τη ζωή των ανθρώπων ανέκαθεν υπάρχει η ανάγκη για επικοινωνία. Από πολύ παλιά αναπτύχθηκαν διάφοροι τρόποι επικοινωνίας με πρωτόλεια μέσα αρχικά, αλλά με την πάροδο του χρόνου και την εξέλιξη της τεχνολογίας και της γλώσσας οι τρόποι επικοινωνίας των ανθρώπων αυξάνονταν σε ποσότητα και ποιότητα. Σήματα καπνού, ζωγραφιές σε σπήλαια, ταχυδρομικά περιστέρια στο τότε, τηλέφωνο, ταχυδρομείο, τηλεόραση, ραδιόφωνο, διαδίκτυο στο σήμερα αποτυπώνουν την εξέλιξη των μέσων επικοινωνίας των ανθρώπων. Όμως δεν αρκούν τα μέσα αλλά χρειάζονται και κάποιες συμβάσεις άτυπες ή και ρητά διατυπωμένες, ώστε η επικοινωνία να επιτευχθεί ομαλά και με την δυνατόν μεγαλύτερη ακρίβεια. Το διαδίκτυο λοιπόν και γενικότερα τα δίκτυα (ενσύρματα ή μη) δε θα μπορούσαν να αποτελούν εξαίρεση.

Για να γίνουμε πιο συγκεκριμένοι, τις συμβάσεις που αναφέραμε τις ονομάζουμε πρωτόκολα επικοινωνίας και μπορούμε να πούμε πως «ένα πρωτόκολλο ορίζει τη μορφή και τη σειρά των μηνυμάτων που ανταλλάσσονται ανάμεσα σε δυο ή περισσότερες οντότητες, όπως και τις ενέργειες που γίνονται κατά τη διάρκεια της μετάδοσης και/ή της λήψης ενός μηνύματος ή άλλου γεγονότος.» Ειδικότερα για τα δίκτυα υπολογιστών(ή άλλων μηχανών, π.χ. κινητά τηλέφωνα) πρωτόκολλο είναι ένα σύνολο από συμβάσεις που καθορίζουν το πώς ανταλλάσσουν μεταξύ τους δεδομένα οι υπολογιστές(συσκευές) του δικτύου.

Ένα πρωτόκολλο επικοινωνίας χαρακτηρίζεται από γραμματική, συντακτικό, και λεξιλόγιο. Όπως λοιπόν τα πρωτόκολλα της ανθρώπινης επικοινωνίας, έτσι κι αυτά των δικτύων υπολογιστών αποτελούν μια μορφή ερωταπαντήσεων. Τα πρωτόκολλα αποτελούνται από διάφορα επίπεδα τα οποία επικοινωνούν με αντίστοιχης “βαθμίδας” επίπεδα και όλα μαζί αποτελούν την αρχιτεκτονική του δικτύου.

Σε χαμηλού επιπέδου αφαιρετικότητα ένα πρωτόκολλο είναι μια μηχανή καταστάσεων, η οποία καθορίζει ποιες ενέργειες λαμβάνουν χώρα και πως να ανταποκριθεί σε διάφορα γεγονότα. Μια μηχανή καταστάσεων για επικοινωνία αποτελείται από: σύνολο καταστάσεων, μετάβαση καταστάσεων και ουρές μηνυμάτων.

1.2 Σχεδιασμός πρωτοκόλλου

Ο σχεδιασμός ενός πρωτοκόλλου δεν αφορά μόνο την εκτέλεσή του. Η διαδικασία ανάπτυξης του περιλαμβάνει: απαιτήσεις, προδιαγραφές και επικύρωση, εκτέλεση, έλεγχος και εξέλιξη του. Όλα τα παραπάνω είναι ένα δύσκολο έργο ακόμα και για ένα απλό πρωτόκολλο.

Είναι πολύ πιθανό κατά τη σχεδίαση ή ακόμα και μετά την ολοκλήρωσή του ένα πρωτόκολλο να παρουσιάσει σφάλματα(μιλώντας ακόμα και σε επίπεδο εφαρμογής χρήστη). Τα σφάλματα αυτά μπορεί να εντοπιστούν στην αρχή της σχεδίασης του πρωτοκόλλου και να επηρεάσουν όλη την ροή της επικοινωνίας. Για να αποφύγουμε αυτή την περίπτωση θα πρεπει να λάβουμε υπ' όψιν τα σφάλματα που μπορεί να προκύψουν(προβλεπόμενα και μή), να επιτρέψουμε στους handlers να “πιάσουν” μη προβλεπόμενα σφάλματα, να ελέγξουμε αν υπάρχουν ή όχι αχρείαστες ή άκυρες παραδοχές.

Η δυσκολία στη σχεδίαση πρωτοκόλλων έγκειται στο πως θα βρούμε τους κατάλληλους κανόνες για επικοινωνία που θα είναι: απλοική, λογικά συνεπής, πλήρης και αποδοτική. Αρχικά ξεκινάμε με μια περιγραφή των κανόνων, των μηνυμάτων και των παραδοχών για το περιβάλλον, δηλαδή όλα τα στοιχεία ενός πρωτοκόλλου.

1.3 Βασικά στοιχεία ενός πρωτοκόλλου

Αφού κάναμε κάποια γενικά σχόλια πάνω στο σχεδιασμό πρωτοκόλλου είναι σημαντικό να δούμε από ποιά βασικά κομμάτια αποτελείται ένα πρωτόκολλο. Ένα πρωτόκολλο αποτελείται από πέντε βασικά στοιχεία τα οποία χρειάζεται να λάβουμε υπ όψιν τόσο πριν όσο και κατά το σχεδιασμό του πρωτοκόλλου. Έτσι λοιπόν τα πέντε αυτά στοιχεία όπως διατυπώθηκαν από τον Holzmann το 1991 είναι τα παρακάτω:

- η παρεχόμενη υπηρεσία του πρωτοκόλλου
- οι παραδοχές γύρω από το περιβάλλον στο οποίο θα εκτελεστεί το πρωτόκολλου
- το λεξιλόγιο των μηνυμάτων που θα χρησιμοποιηθούν κατά την εκτέλεση του πρωτοκόλλου
- η κωδικοποίηση του μηνύματος στο λεξιλόγιο
- κανόνες που φυλάνε τη συνοχή κατά την ανταλλαγή μηνυμάτων.

1.4 Παράδειγμα – το πρωτόκολλο του Lynch

Πριν γίνουμε πιο συγκεκριμένοι και προχωρήσουμε σε επόμενα κεφάλαια της εργασίας, θα κλείσουμε το εισαγωγικό κομμάτι αναφερόμενοι σε ένα παράδειγμα πρωτοκόλλου και αυτό είναι το πρωτόκολλο του Lynch. Είναι ένα πρωτόκολλο πλήρους αμφίδρομης εναλλαγής μετάδοσης δεδομένων για μια ημί – αμφίδρομη τηλεφωνική γραμμή και αποτελεί απτό παράδειγμα της δυσκολίας στη σχεδίαση ενός πρωτοκόλλου.

Πιο συγκεκριμένα για να μπορέσουμε να προσδιορίσουμε το πρωτόκολλο του Lynch θα αναφέρουμε τα παρακάτω:

- υπηρεσία : το πρωτόκολλο θα πρέπει να επιτρέπει αξιόπιστη και πλήρως αμφίδρομη μετάδοση μηνυμάτων. Η μετάδοση ενός μηνύματος θα απαντιέται από μια θετική ή αρνητική απόκριση.

- περιβάλλον : δύο χρήστες μοιράζονται ένα κανάλι επικοινωνίας. Τα μηνύματα μπορεί να διαγραφούν αλλά ποτέ δεν χάνονται. Τα κατεστραμμένα μηνύματα εντοπίζονται και μετατρέπονται σε σωστά μηνύματα.

- λεξιλόγιο μηνυμάτων : τα μηνύματα, η απόκριση και μια σημαία σφάλματος

{ACK, NACK, ERR}

- μορφή μηνύματος : data(char), ack flag(binary) + error flag(binary)

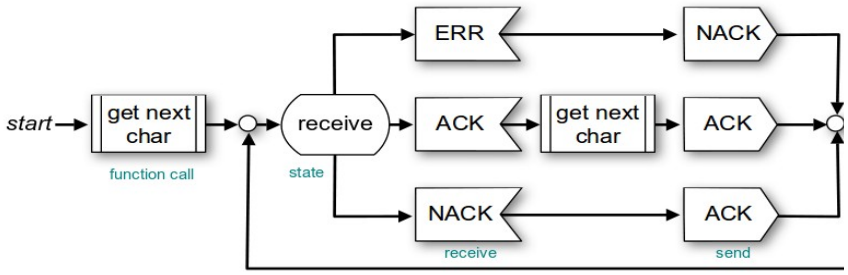
- διαδικαστικοί κανόνες : (ανεπίσημοι)

Σε αυτό το σημείο θα δούμε δύο βήματα όσον αφορά την ροή κατά την λήψη και την μετάδοση των bit δεδομένων:

1. αν η προηγούμενη λήψη ήταν χωρίς λάθη, το bit απόκρισης στην επόμενη μετάβαση θα ναι 1, αλλιώς το bit απόκρισης θα ναι 0.

2. αν το bit απόκρισης της προηγούμενης λήψης ήταν 0, ή αν η προηγούμενη λήψη ήταν λάθος,τότε ξαναμετέδωσε το παλιό μήνυμα ή φέρε ένα νέο μήνυμα για μετάδοση.

1.



[Holzmann 19

διάγραμμα ροής:

η μετάδοση προς μια κατεύθυνση δεν συνεχίζεται αν το get_next_char δεν επιστρέψει δεδομένα καθώς επίσης η αρχικοποίηση και ο τερματισμός δεν έχουν προσδιοριστεί.

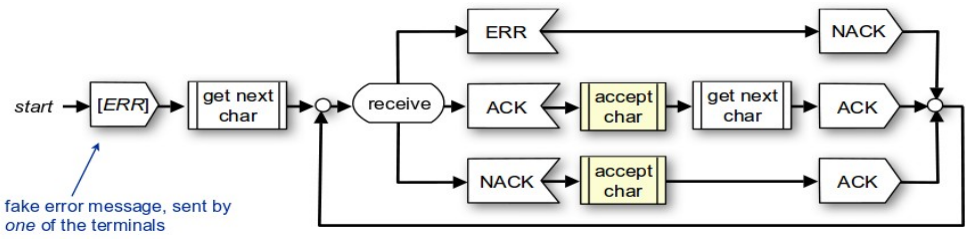
Στην περίπτωση που φτάσουμε σε ένα τέλμα και δεν προχωρήσει η ροή των δεδομένων όπως θα θέλαμε προβένουμε στις παρακάτω κινήσεις ώστε να διορθώσουμε την κατάσταση και πιο συγκεκριμένα:

- χρησιμοποιούμε μετάδοση ενός δρόμου: χρησιμοποιούμε γεμίζοντα μηνύματα ώστε το get_next_char να μην μπλοκάρει,

- αρχικοποιούμε στέλνοντας ένα ψεύτικο λάθος μήνυμα ώστε να μην μπλοκάρει το receive,

- πιο πολλά προβλήματα: η ερώτηση για το πότε ένα μήνυμα θα γίνει αποδεχτό είναι ακόμα αναπάντητη. Αν ένα τερματικό δεχτεί όλα τα χωρίς-λάθος μηνύματα δε θα μπορούν να εντοπιστούν τα διπλότυπα.

2.



Συμπεραίνουμε από τα παραπάνω πως τα διαγράμματα ροής δεν είναι πάντα προφανή και δεν είναι εύκολο να αποτυπωθούν, επειδή κάποια λάθη προκύπτουν σπάνια. Τα διαγράμματα είναι συνήθως δύσκολο να διορθωθούν αλλά ακόμα και τα απλά πρωτόκολλα χρειάζονται καλή σχεδίαση. Αναφέραμε λίγα πράγματα με το παράδειγμα στο πρωτόκολλο του Lynch σαν μία καλή εισαγωγή. Στη συνέχεια θα μιλήσουμε πιο συγκεκριμένα αναλύνοτας ανά κεφάλαιο την όλη διαδικασία του σχεδιασμού.

ΚΕΦΆΛΑΙΟ 2: ΣΧΕΔΙΑΣΤΙΚΕΣ ΑΡΧΕΣ

2.1 Σχεδιαστικά θέματα

Στο κεφάλαιο αυτό κάνουμε λόγο για σχεδιαστικά θέματα. Θα δούμε ουσιαστικά κάποιους κανόνες που θα πρέπει να ακολουθήσουμε τόσο για τη δική μας διευκόλυνση κατά την διαδικασία του σχεδιασμού, όσο και για την σωστή λειτουργία του ίδιου του πρωτοκόλλου και την αποφυγή λαθών και μη επιθυμητών καταστάσεων.

Οι γενικές αρχές σχεδίασης πρωτοκόλλων είναι οι εξής :

ΑΠΛΟΤΗΤΑ – Η ΠΕΡΙΠΤΩΣΗ ΤΩΝ ΕΛΑΦΡΥΩΝ ΠΡΩΤΟΚΟΛΛΩΝ

Κάθε πρωτόκολλο θα πρέπει να χτίζεται από έναν αριθμό μικρότερων που το κάθε ένα επικεντρώνεται σε μία μόνο λειτουργία. Μια ελαφρυά σχεδίαση είναι πιο εύκολο να εφαρμοστεί, να επαληθευτεί και να διατηρηθεί.

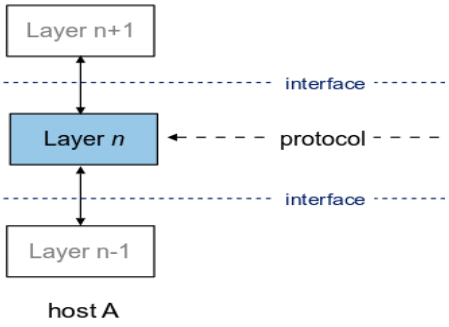
MODULARITY – ΙΕΡΑΡΧΙΑ ΤΩΝ ΛΕΙΤΟΥΡΓΙΩΝ

Ένα σύμπλεγμα λειτουργιών που χτίζεται από ανεξάρτητες και ατομικές οντότητες. Μια οντότητα δεν κάνει καθόλου παραδοχές για άλλες οντότητες. Εδώ οι κύριες δομικές τεχνικές είναι :

-η διαστρωμάτωση πρωτοκόλλων:

όπου διαχωρίζονται οι εργασίες υψηλότερων επιπέδων από τις λεπτομέρειες χαμηλότερων επιπέδων. π.χ. το OSI μοντέλο. ορίζονται τα επίπεδα αφαιρετικότητας, ενσωματώνονται σχετικές συναρτήσεις, και θα πρέπει να υπάρχουν μικρές και καλά ορισμένες διεπαφές.

3.



παράδειγμα διαστρωμάτωσης

Η διαστρωμάτωση μπορεί και σπάει τα προβλήματα σε μικρότερα κομμάτια και με αυτόν τον τρόπο εκτελούνται πιο ελαφρυές οντότητες, οι οποίες είναι ανταλλάξιμες και επανχρησιμοποιήσιμες. Το πρόβλημα όμως είναι ότι η απόκρυψη πληροφορίας μπορεί να οδηγήσει σε μικρότερη απόδοσης.

- κατασκευή δεδομένων: η μορφή των δεδομένων για χαμηλού επιπέδου είναι προσανατολσμένες-στο-bit, στο-χαρακτήρα ή στο-byte. Υπάρχουν διαχωριστικά πλαίσια για ακολουθία bit, χαρακτήρα ή υποδεικνύονται από κάποιον μετρητή. Ενώ για υψηλού επιπέδου δομούνται κεφαλίδες από ακολουθίες αριθμών και checksums(ο αριθμός σωστών ψηφίων σε μια σειρά δεδομένων) με τήρηση της προτεραιότητας ανάμεσα σε κάθε αποστολέα και λήπτη.

ΟΡΘΑ ΜΟΡΦΟΠΟΙΗΜΕΝΑ ΠΡΩΤΟΚΟΛΛΑ

Ένα καλά μορφοποιημένο πρωτόκολλο:

- δεν είναι ούτε υπέρ ούτε υπό προσδιορισμένο
- οριοθετείται, δηλαδή φροντίζει για τα όρια του συστήματος
- επιστρέφει σε μια ορισμένη κατάσταση μετά από ένα παροδικό σφάλμα
- προσαρμόζεται στις αλλαγές του περιβάλλοντος

ΔΥΝΑΜΗ(ΕΥΡΩΣΤΙΑ)

- σωστή εκτέλεση κάτω από ορισμένες καταστάσεις
- τα πρωτόκολλα θα πρέπει να τηρούν ένα μίνιμαλ σχεδιασμό

ΣΤΑΘΕΡΟΤΗΤΑ

δηλαδή αποφυγή των ασταθών καταστάσεων και των επαναλήψεων κατα την εκτέλεση ενός πρωτοκόλλου η οποία δεν οδηγεί κάπου ή η αποφυγή του λάθους τερματισμού πρωτοκόλλου.

Ας δούμε τώρα κάποιους κανόνες-συμβουλές που διατύπωσε ο Holzmann to 1991 οι οποίοι σχετίζονται βέβαια και με τις παραπάνω σχεδιαστικές αρχές

Οι δέκα κανόνες σχεδιασμού:

1. Σιγουρέψου ότι το πρόβλημα είναι καλά σχεδιασμένο
2. Διατύπωσε την υπηρεσία πρώτα. (το “τι” προηγείται του “πως”)
3. Σχεδίασε την εξωτερική λειτουργικότητα πριν την εσωτερική
4. Κράτα το απλό

5. Μην συνδέεις αυτά που είναι ανεξάρτητα

6. Μην επιβάλλεις περιορισμούς σε άσχετα θέματα

7. Χτίσε ένα υψηλού επιπέδου πρότυπο πρώτα και πιστοποίησε το

8. Εφάρμοσε το σχεδιασμό, αξιολόγησέ το και βελτιστοποίησέ το

9. Τσέκαρε την ισοδυναμία των προτοτύπων και την εφαρμογή τους

10. Μην παραβλέπεις τους κανόνες 1-7

2.2 Σχεδιαστικά αρχές για Internet

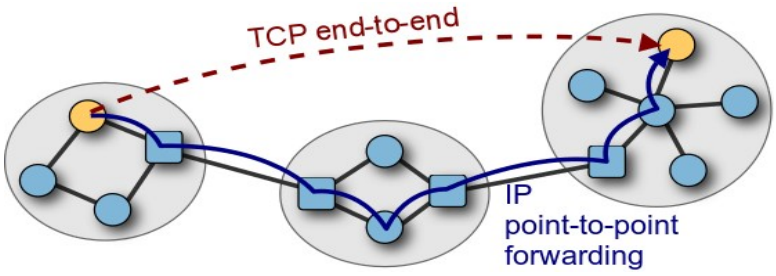
Μιλώντας για δικτυακά πρωτόκολλα δεν θα μπορούσαμε να μην αναφερθούμε στις σχεδιαστικές αρχές αυτών για το διαδικτύο και πιο συγκεκριμένα για τα πρωτόκολλα TCP/IP. Τα πρωτόκολλα του Internet TCP/IP σχεδιάστηκαν το 1970 και ακόμα και σήμερα χρησιμοποιούνται επιτυχώς. Τα βασικά χαρακτηριστικά της επικοινωνίας στο Intenet είναι: μεταγωγή πακέτων, ασύρματες υπηρεσίες, διαστρωματωμένα πρωτόκολλα.

Το TCP είναι μια αξιόπιστη υπηρεσία μεταφοράς από-τερματικό-σε-τερματικό και χρησιμοποιεί το IP.

Το IP είναι ασύρματη επικοινωνία διαγράμματος δεδομένων.

Το TCP/IP επιτρέπει την επικοινωνία τερματικού-σε-τερματικό σε διασυνδεδεμένα δίκτυα.

4.



επικοινωνία τερματικού με τερματικό σε TCP

Στο διαδίκτυο κατά των σχεδιασμό των πρωτοκόλλων βάζουμε κάποιους στόχους που θέλουμε να ικανοποιήσουμε οι οποίοι μπορεί να είναι πρωτεύοντες ή δευτερεύοντες, αλλά όπως και να έχει είναι σημαντικό να προσπαθήσουμε προς αυτήν την κατεύθυνση.

Σαν πρωτεύον στόχο θέτουμε την αποτελεσματική αξιοποίηση των υπαρχόντων διασυνδεδεμένων δικτύων. Σαν στοιχεία του έχουμε τα δίκτυα μεταγωγής πακέτων και υπάρχουν δύο επιλογές κατά τη σχεδίαση είτε η πολυπλεξία μέσω μεταγωγής πακέτου είτε η αποθήκευση και προώθηση πακέτων μέσω πυλών(gateways). Επίσης επιλέγεται η διασύνδεση δικτύου δηλαδή η χρήση ενός δικτύου μικρότερων δικτύων χαμηλότερης πολυπλοκότητας το κάθε ένα, όπου μεταξύ των υποδικτύων αυτών γίνεται μετάφραση στις πύλες (gateways).

Σαν δευτερεύον επιπέδου σχεδιαστικούς στόχους αλλά ίσοννος σημασίας αναφέρουμε τους παρακάτω:

1. Η επικοινωνία στο διαδίκτυο πρέπει να συνεχίζεται παρά την απώλεια υποδικτύων ή πυλών(gateways)

2. Το διαδίτκυο πρέπει να υποστηρίζει πολλαπλών τύπων υπηρεσίες επικοινωνίας.

3. Η αρχιτεκτονική του διαδικτύου πρέπει να “φιλοξενεί” μία ποικιλία δικτύων.

4. Η αρχιτεκτονική του διαδικτύου πρέπει να επιτρέπει κατανεμημένη διαχείριση των πηγών του.

5. Η αρχιτεκτονική του διαδικτύου πρέπει να είναι αποδοτική.

6. Η αρχιτεκτονική του διαδικτύου πρέπει να επιτρέπει σύνδεση των host με την λιγότερη δυνατή προσπάθεια.

7. Οι πηγές που χρησιμοποιούνται στο διαδίκτυο πρέπει να είναι αριθμήσιμες.

Κάνοντας μία ανασκόπηση στα TCP/IP διαδικτυακά πρωτόκολλα μπορούμε να πούμε πως οι σχεδιαστικές αρχές που έχουν εφαρμοστεί σε αυτά είναι οι απλότητα και το modularity που διαχωρίζουν το διάγραμμα δεδομένων(IP) και την υπηρεσία μεταφοράς(TCP)

Κλείνοντας αυτό το κεφάλαιο ας αναφέρουμε κάποια προβλήματα που αποτελούν και πρόκληση προς νέους σχεδιαστές πρωτοκόλλων. Η έλλειψη ελέγχου ροής τερματικού-σε-τερματικό οδήγησε σε

πρόβλημα συμφόρησης το '86. Επίσης μπορεί να συμβεί η δρομολόγηση να είναι μη-βέλτιστη παρουσιάζοντας προβλήματα. Τέλος σημαντικό πρόβλημα είναι η έλλειψη έλλειψη IP διευθύνσεων και το πρόβλημα NAT.

Κεφάλαιο 3: Ο Σχεδιασμος

3.1 Διαδικασία ανάπτυξης

Από το κεφάλαιο αυτό θα αρχίσουμε να γινόμαστε πιο συγκεκριμένοι και να μιλάμε πιο “πρακτικά” για τον σχεδιασμό και εδώ αρχικά θα μιλήσουμε για την διαδικασία ανάπτυξης. Για να χτίσουμε τη διαδικασία ανάπτυξης θα πρέπει να χτίσουμε τη διαδικασία ανάπτυξης του software σε βήματα και να έχουμε ορίσει καλά τις μεταβάσεις. Παραδείγματα αποτελούν τα μοντέλα Build and Fix, Waterfall, Boehm's spiral.

Η διαδικασία ανάπτυξης υλοποιείται σταδιακά και σε αυτήν λαμβάνουν χώρα οι παρακάτω βασικές εργασίες (που παρουσιάζονται στα περισσότερα μοντέλα) που είναι οι εξής:

- ανάλυση απαιτήσεων
- προσδιορισμός σχεδιασμού
- επικύρωση
- εκτέλεση
- έλεγχος και εκτίμηση
- ανάπτυξη
- διατήρηση

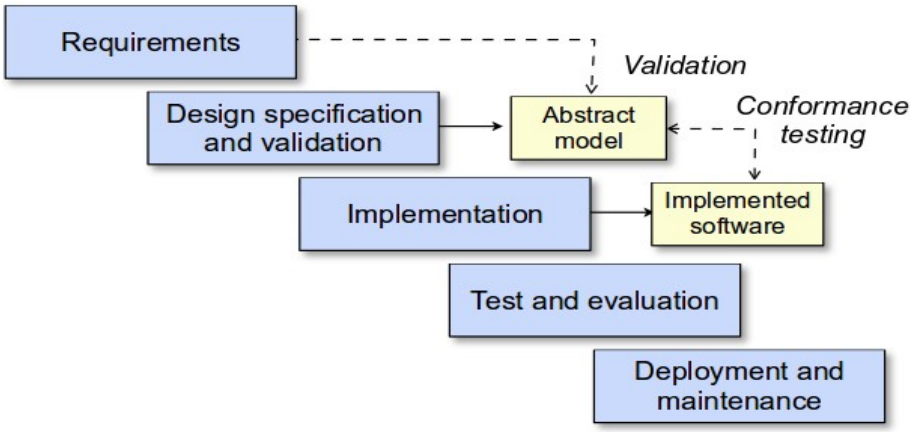
Η επιλογή ενός σχεδιαστικού μοντέλου εξαρτάται από τον τύπο και την πολυπλοκότητα του προτζεκτ. Τα δικτυακά πρωτόκολλα συχνά χρειάζονται αναθεώρηση και επέκταση εξαιτίας της αλλαγής των απαιτήσεων ή άλλων προβλημάτων. Η σχεδίαση πρωτοκόλλου είναι μια επαναληπτική διαδικασία.

Οι απαιτήσεις περιγράφουν την επιθυμητή συμπεριφορά ενός πρωτοκόλλου και είναι ανεξάρτητες του μεταγενέστερου σχεδιασμού και της εκτέλεσης. Υπάρχουν τριών ειδών απαιτήσεις, λειτουργικές απαιτήσεις δηλαδή η συμπεριφορά του συστήματος και η μορφή των δεδομένων, ποιοτικές απαιτήσεις όπως η απόδοση και τρίτον οι σχεδιαστικοί περιορισμοί όπως το περιβάλλον και η διεπαφή.

Κατά την επικύρωση απαιτήσεων ελέγχεται αν ο προσδιορισμός ταιριάζει με τις απαιτήσεις του χρήστη. Για να υπάρχει συνέπεια στις απαιτήσεις θα πρέπει να λύσουμε τις συγκρούσεις, όπως για παράδειγμα αυτό που θέλουμε ίσως συγκρούεται με αυτό που είναι υλοποιήσιμο.

Για την μοντελοποίηση και τον προσδιορισμό χρησιμοποιούνται επίσημη σημειογραφία και γλώσσες όπως μηχανές καταστάσεων, UML(unified modeling languages), SDL(specification and description language), διαγράμματα ακολουθίας μηνυμάτων.

5.



μοντέλο ελέγχου

ΚΕΦΆΛΑΙΟ 4 : ΠΡΟΣΔΙΟΡΙΣΜΌΣ ΠΡΩΤΟΚΌΛΛΟΥ

4.1 FSM, UML, μηχανές καταστάσεων, UML sequence διαγράμματα

Στο κεφάλαιο αυτό όπως λέει και ο τίτλος του θα μιλήσουμε για θέματα που αφορούν τον προσδιορισμό πρωτοκόλλου. Ο προσδιορισμός μπορεί να γίνει με μηχανές πεπερασμένων αυτομάτων αλλά και με πιο εξειδικευμένες γλώσσες όπως η UML και αυτές θα μας απασχολήσουν παρακάτω.

Ο προσδιορισμός με μηχανές καταστάσεων σημαίνει ότι ένα πρωτόκολλο αλληλεπιδρά με το περιβάλλον δηλαδή ενεργοποιείται από γεγονότα, ανταποκρίνεται σε δράσεις απόδοσης και η συμπεριφορά του εξαρτάται από παρελθοντικά γεγονότα.

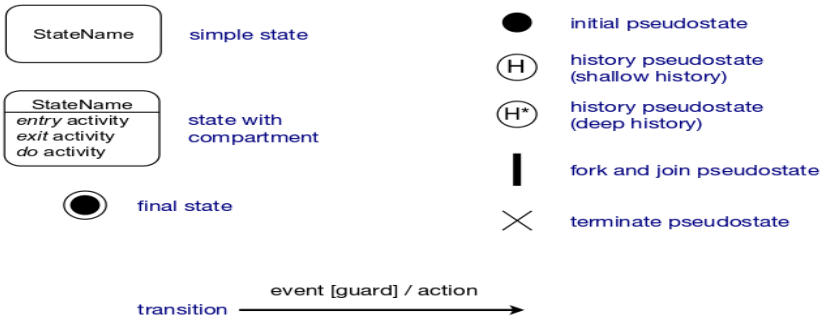
Χρησιμοποιούνται μηχανές καταστάσεων αντί για γλώσσες προγραμματισμού λόγω έλλειψης επίσημης σημειολογίας, ρίσκου υπερπροσδιορισμού και του ότι οι απαιτήσεις προσδιορισμού θα πρέπει να ναι ανεξάρτητες της εφαρμογής.

Το πρόβλημα με την οριοθέτηση των FSM έγκειται στην πεπερασμήνη μνήμη και στην μοντελοποίηση των καναλιών επικοινωνίας. Επίσης υπάρχει σημαντικό πρόβλημα με τις παράλληλες FSM αφού δεν υπάρχει επικοινωνία καναλιών, συγχρονισμός και η σύνθεση αλληλεπιδρόντων FSM οδηγεί σε νέες καταστάσεις και σε μια “έκρηξη” του χώρου καταστάσεων. Τα πρωτόκολλα επικοινωνίας μπορούν να ειδωθούν ως παράλληλες μηχανές επικοινωνίας.

Τα γνήσια FSM δεν είναι κατάλληλα για μοντελοποίηση και προσδιορισμό διεργασιών σε κατανεμημένα συστήματα. Οπότε οδηγούμαστε σε επεκτάσιμα μοντέλα όπως CFSM, διάγραμμα καταστάσεων Harell(υπερκαταστάσεις, παράλληλες καταστάσεις), EFSM(μεταβλητές, τελεστές, συνθήκες), η βάση για γλώσσες μοντελοποίησης όπως SDL, UML.

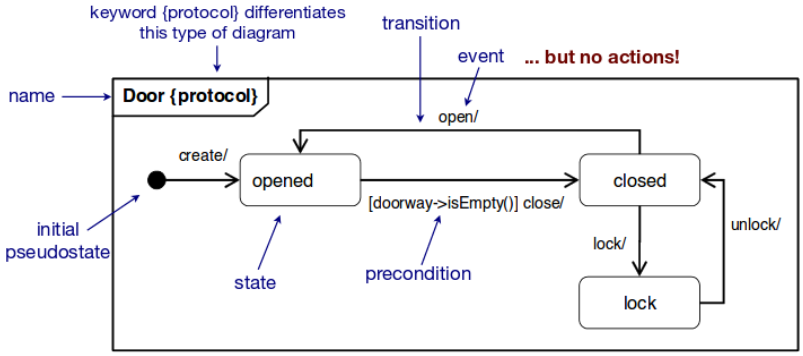
Μιλώντας τώρα για την UML μπορούμε να πούμε πως αποτελεί μία γενικού προσδιορισμού γλώσσα για μοντελοποίηση και προσδιορισμό, που αποτελείται από μηχανές καταστάσεων και διαγράμματα ακολουθιών, όπως φαίνεται και παρακάτω.

6.



καταστάσεις και μεταβάσεις

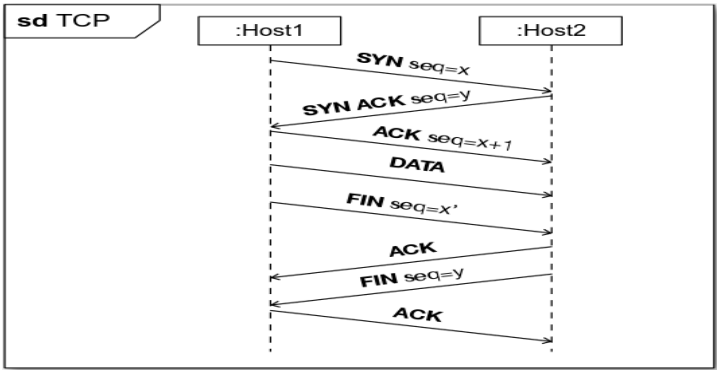
7.



παράδειγμα μηχανής καταστάσεων UML πρωτοκόλλου

Οι μηχανές καταστάσεων πρωτοκόλλων δεν μπορούν να περιγράψουν αποκρίσεις όπως η αποστολή μηνυμάτων αναγνώρισης, αλλά επιτρέπουν μια ανακλαστική περιγραφή της συμπεριφοράς.

8.



TCP παράδειγμα βασισμένο σε UML

4.2 Επεκτάσιμα FSM, διάγραμμα SDL καταστάσεων, MSC/LSC

Τα FSM έχουν περιορισμένη εκφραστικότητα σε αντίθεση με τα UML διαγράμματα καταστάσεων που είναι πιο δυνατά μεν, αλλά παρουσιάζουν σημασιολογική μεταβολή. Ακόμη απαιτείται επίσημη σημασιολογία για την επικύρωση. Τα επεκτάσιμα μοντέλα μηχανών καταστάσεων και οι επίσημες γλώσσες χωρίζονται σε τρεις κατηγορίες τα communicating FSM που έχουν σαν επιπρόσθετο τις ουρές μηνυμάτων, extended FSM που έχουν επιπλέον με τα άλλα μεταβλητές και η γλώσσα SDL που είναι βασισμένη στα extended FSM, αλλά επιπροσθέτως έχει δομικά σενάρια.

Παρακάτω παραθέτουμε τους ορισμούς ενός cfsm και ενός extended-fsm.

ΟΡΙΣΜΟΣ ΕΝΟΣ CFSM:

→ Μια ουρά μηνύματος είναι μια τριάδα (S,N,C) όπου ,

S είναι ένα πεπερασμένο σύνολο που ονομάζεται λεξιλόγιο της ουράς/μηνύματος

το N ορίζει το μέγεθος της ουράς και το C τα περιεχόμενα

→ Ένα CFSM είναι μια τετράδα (Q, q_0 , M, T) όπου,

το Q είναι ένα πεπερασμένο μη-κενό σύνολο καταστάσεων,

το q_0 είναι η αρχική κατάσταση,

το M είναι ένα σύνολο από ουρές μηνυμάτων,

και το T είναι η σχέση μετάβασης καταστάσεων,T: QxA –>Q, όπου το A είναι το σύνολο των ενεργειών

Ως τώρα μπορούμε να ανταλλάσουμε μηνύματα μεταξύ CFSM αλλά μας λείπουν οι μεταβλητές και η ικανότητα να ανταλλάσουμε αυθαίρετες τιμές. 'Αρα πάμε στα Extended FSM.

ΟΡΙΣΜΟΣ ΕΝΟΣ EXTENDED FSM:

→ είναι μια τετράδα (Q, q_0 ,M, V, T) όπου:

το Q είναι ένα πεπερασμένο μη-κενό σύνολο καταστάσεων,

το q_0 είναι η αρχική κατάσταση,

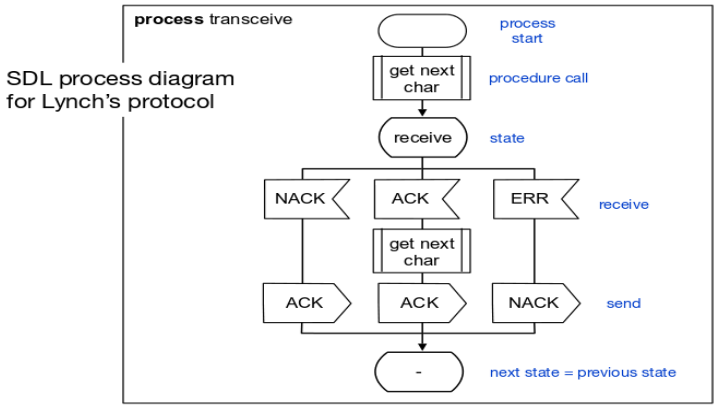
το M είναι ένα σύνολο από ουρές μηνυμάτων,

το T είναι η σχέση μετάβασης καταστάσεων,T: QxA –>Q, όπου το A είναι το σύνολο των ενεργειών και το V είναι ένα σύνολο μεταβλητών.

Πρόκειται για ένα επίσημο μοντέλο για τον προσδιορισμό παράλληλων διεργασιών, όπου μπορούν να εφαρμοστούν η ελαχιστοποίηση και ο συνδυασμός FSM. Η ελαχιστοποίηση ενός FSM σημαίνει να βρούμε μια ισοδύναμη μηχανή καταστάσεων με τον ελάχιστο αριθμό καταστάσεων.

Ορίζοντας την γλώσσα SDL μπορούμε να πούμε πως είναι μία προσδιοριστική και περιγραφική γλώσσα που αρχικά αναπτύχθηκε για προσδιορισμό τηλεπικοινωνιακών συστημάτων όπως για παράδειγμα τα ISDN πρωτόκολλα. Είναι μια επίσημη γλώσσα που βασίζεται σε extended FSM.

9.



παράδειγμα SDL

Η SDL περιγράφει παράλληλες διεργασίες και την αλληλεπίδραση αυτών. Ένας SDL προσδιορισμός συστήματος περιγράφει τη δομή, την επικοινωνία, τη συμπεριφορά και τα δεδομένα.

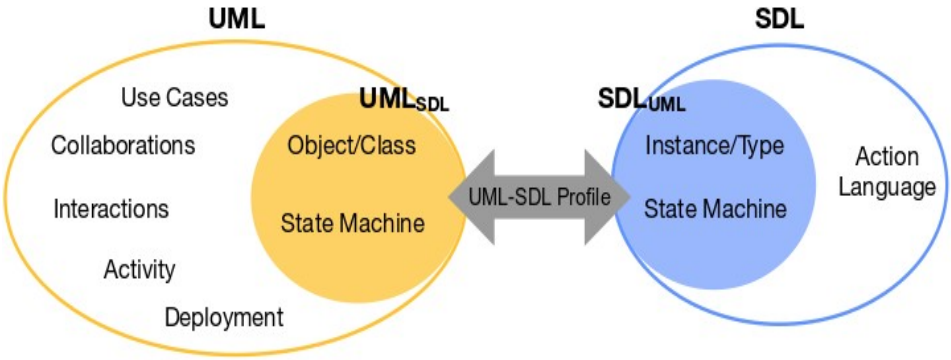
Οι διεργασίες περιγράφουν συμπεριφορά, τρέχουν παράλληλα, μπορούν να επικοινωνούν και είναι ομαδοποιημένες σε μπλοκ. Περιέχουν ένα extended-FSM και επικοινωνούν με σήματα.

Τα μπλοκ περιγράφουν τη δομή, μπορούν να συνδεθούν ή να συμπεριληφθούν σε άλλα μπλοκ και το πιο εξωτερικό μπλοκ που περιλαμβάνει τα υπόλοιπα ονομάζεται σύστημα. Τα μπλοκ-διεργασίες λέγονται πράκτορες. Οι πράκτορες επικοινωνούν είτε ασύγχρονα με σήματα μέσω ενός καναλιού(τα κανάλια περιγράφουν μονοπάτια επικοινωνίας), είτε σύγχρονα με διαδικαστικές κλήσεις.

Κάθε στιγμιότυπο διεργασίας έχει την ουρά εισόδου του(FIFO). Τα σήματα λαμβάνονται οποιαδήποτε στιγμή και αυτά που είναι πλήρως έγκυρα εισέρχονται στην ουρά. Αν μια διεργασία είναι σε μία συγκεκριμένη κατάσταση και η ουρά δεν είναι άδεια και υπάρχουν σήματα που σχετίζονται με μεταβάσεις από αυτήν την κατάσταση, τότε το σήμα φεύγει από την ουρά και η μετάβαση ενεργοποείται. Για απροσδιόριστους συνδυασμούς σημάτων ή καταστάσεων, το σήμα ¨καταναλώνεται¨ χωρίς να γίνει κάποια ενέργεια.

Τώρα όσον αφορά τις μεταβλητές, αυτές δηλώνονται με κάποιο σύμβολο κειμένου. Αντί για το πέρασμα ενός σήματος, μια μεταβλητή μπορεί να εξαχθεί από μια διεργασία και να εισαχθεί σε μία άλλη.

10.

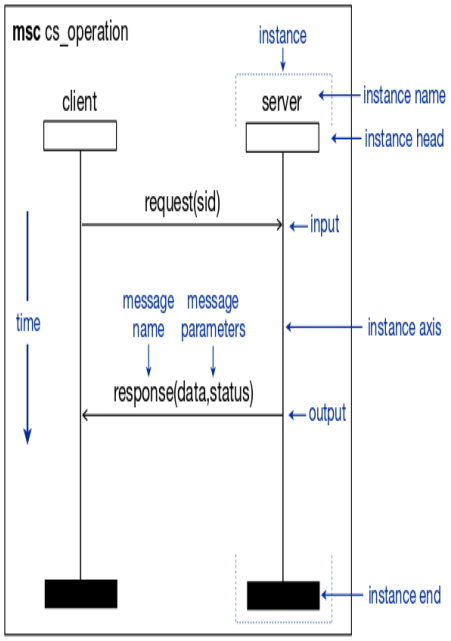


SDL και UML: όπως βλέπουμε και στο σχήμα οι δύο γλώσσες αλληλοδιαπλέκονται.

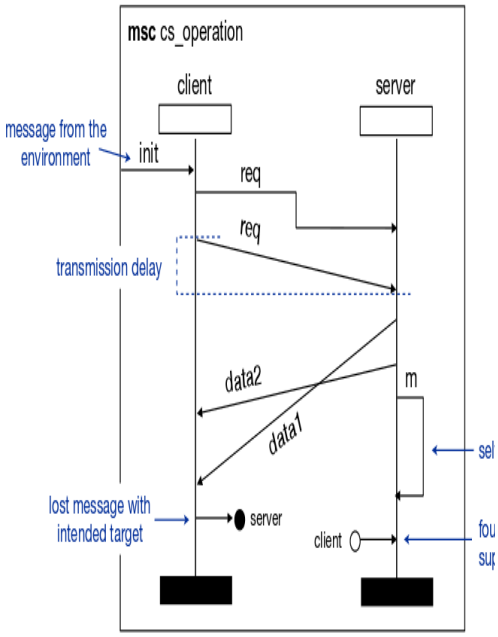
Τα αρχικά μοντέλα μηχανών καταστάσεων δεν ήταν σαφή ως προς τη μοντελοποίηση παράλληλων και επικοινωνούντων διεργασιών(παράδειγμα για τα δικτυακά πρωτόκολλα). Αλλά έκαναν την εμφάνισή τους τα extended-FSM με κανάλια και μεταβλητές πάνω στα οποία βασίζονται και οι διεργασίες στην SDL. Υπάρχουν πολλές ομοιότητες μεταξύ της UML και της SDL όμως η κύρια διαφορά που κάνει την SDL να ξεχωρίζει είναι ότι έχει πολύ ισχυρότερη σημειολογία.

Συνεχίζοντας θα αναφέρουμε τα MSC τα οποία αποτελούν διαγράμματα ακολουθίας μηνυμάτων, όμοια με αυτά της UML. Είναι μία επίσημη γλώσσα γραφικής απεικόνισης που περιγράφει τη συμπεριφορά επικοινωνούντων περιστατικών για συγκεκριμένες εκτελέσεις.

11.



12.



Σαν επέκταση του MSC έχουμε τα LSC(life sequence charts) τα οποία αποτελούνται από υποχρεωτικά(hot) και προσωρινά(cold) στοιχεία. Τα μηνύματα που ανταλάσσουν είναι ασύγχρονα και στιγμιαία και αυτά δομούνται από συνθήκες και σταθερές. Η μεταφορά σε μια μηχανή καταστάσεων προϋποθέτει ένα επιπρόσθετο μοντέλο σημασιολογίας. Η σημασιολογία στα LSC είναι ακόμα πιο δυνατή από τα MSC.

4.3 Προσδιορισμός μορφής δεδομένων/μηνύματος

Η δόμηση των δεδομένων είναι μέρος της σχεδιαστικής διαδικασίας αλλά πολλές φορές δεν της δίνουμε την δέουσα σημασία. Οι σχεδιαστικές αποφάσεις δεν θα έπρεπε να οδηγούνται από τη μορφή των δεδομένων ή τα διάφορα θέματα κωδικοποίησης όπως γίνεται αρκετές φορές. Βέβαια μία ορισμένη μορφή δεδομένων/μηνύματος και η αντίστοιχη κωδικοποίηση αυτών είναι σημαντικά για την διαλειτουργικότητα. Οπότε κατά τη διαδικασία καθορισμού των προδιαγραφών είναι αναγκαία μία σημειογραφία. Προκύπτουν δύο τύποι σημειογραφίας:

- Πινακοειδής σημειογραφία: λίστα με πεδία μηνυμάτων που περιέχουν τύπο, μέγεθος και περιγραφή

- Σημειογραφία “κουτί”: πίνακας με πεδία μηνυμάτων όπου το μέγεθος του κουτιού υποδεικνύει το μέγεθος του πεδίου. Χρησιμοποιείται από το IETF.

Στη σημειογραφία κουτί γίνεται προσδιορισμός της μορφής των δεδομένων των διαδικτυακών πρωτοκόλλων. Τα TCP και IP πακέτα έχουν ένα μέρος κεφαλίδας και ένα μέρος δεδομένων. Η πληροφορία κεφαλίδας περιλαμβάνει: την IP και TCP, διεύθυνση πηγής και το port, TTL και ακολουθία αριθμών, σημαίες κ.α. Είναι ένας διαισθητικός τρόπος περιγραφής της μορφής δεδομένων. Οι τύποι των πεδίων και η κωδικοποίηση πρέπει να προσδιορίζονται ξεχωριστά. Βέβαια έχουμε τον περιορισμό του ότι δεν υπάρχουν πεδία μήκους μεταβλητής.

Για την αναπαράσταση του μεγέθους της μεταβλητής ή του πεδίου μηνύματος χρησιμοποιείται η TLV, όπου το μέγεθος της μεταβλητής και το πεδίο μηνύματος μπορούν να αναπαρασταθούν χωρίς να έχουμε κατανοήσει τη σημασία του πεδίου.

Φεύγοντας από το αφαιρετικό επίπεδο και καθώς γινόμαστε όλο και πιο συγκεκριμένοι έχουμε φτάσει στην ABNF σημειογραφία(Augmented Backus Naur Form). Είναι μία γλώσσα ορισμού για κάποια IETF πρωτόκολλα. Η BNF περιγράφει γραμματικές χωρίς συμφραζόμενα με κανόνες παραγωγής της μορφής:

<symbol>::=expression , στην αριστερή μεριά είναι τα μη-τερματικά σύμβολα και στην δεξιά είναι τόσο τα μη-τερματικά όσο και τα τερματικά σύμβολα. Η έκφραση στην δεξιά μεριά είναι μία ακολουθία συμβόλων ή επιλογή ακολουθιών.

13.

```
date-time       =   [ day-of-week "," ] date time [CF
day-of-week     =   ([FWS] day-name)
day-name        =   "Mon" / "Tue" / "Wed" / "Thu" /
                    "Fri" / "Sat" / "Sun"
date            =   day month year
day             =   ([FWS] 1*2DIGIT FWS)
month           =   "Jan" / "Feb" / "Mar" / "Apr" /
                    "May" / "Jun" / "Jul" / "Aug" /
                    "Sep" / "Oct" / "Nov" / "Dec"
year            =   (FWS 4*DIGIT FWS)
time            =   time-of-day zone
time-of-day     =   hour ":" minute [ ":" second ]
hour            =   2DIGIT
minute          =   2DIGIT
second          =   2DIGIT
zone            =   (FWS ( "+" / "-" ) 4DIGIT)
```

παράδειγμα της ABNF

Η ονοματοδοσία των κανόνων δεν πάει ανάλογα την περίπτωση. Υπάρχουν συγκεκριμένοι κανόνες για την περίπτωση των κεφαλαίων. Οι κανόνες μπορούν να μπουν σε σύμβολα όπως < > αλλά αυτό δεν είναι δεσμευτικό. Με το ; αρχίζουν σχόλια και οι δυαδικές ή δεκαεξαδικές τιμές ορίζονται ως %b1101,%h98C3.

Πέρα απ'ο την ABNF, ένα άλλο πρότυπο είναι το CSN.1(concrete syntax notation one).Αποτελεί περιγραφή κωδικοποίησης δεδομένων, αναπτύχθηκε από την κοινότητα GSM και βασίζεται στην BNF. Ορίζει έγκυρα stream κωδικοποιημένων δεδομένων στο επίπεδο του bit.

14.

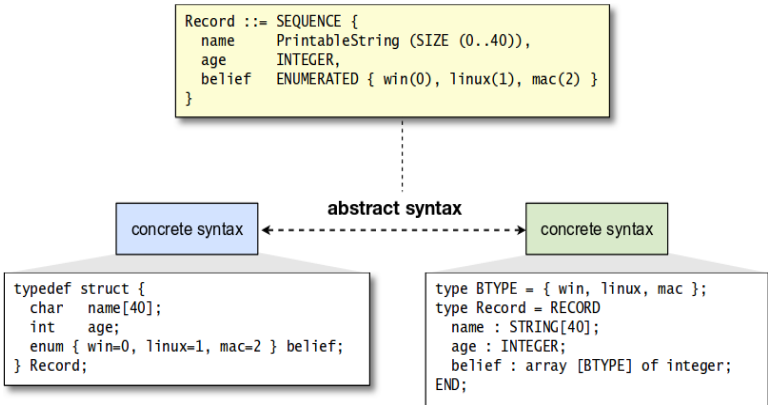
```
<bit> ::= {0 | 1}

<octet> ::= {0 | 1} {0 | 1} {0 | 1} {0 | 1}
            {0 | 1} {0 | 1} {0 | 1} {0 | 1} ;

<octet> ::= <bit>*8 ;

<octet string> ::= <octet>**;

<octet string(40)> ::= <octet>*(8*(4+1)) ;

<all bit strings> ::= null |  {<all bit strings> {0
```

παράδειγμα γραμματικής στο csn.1

Όπως καταλαβαίνουμε είναι μία μέθοδος επίσημης περιγραφής με απλούς κανόνες, είναι μεταγλωτίσσιμη, έχει εξελίξιμα standards και συχνά χρησιμοποιούνται extensions και μη standard σημειογραφία.

Μετά την csn.1 μία άλλη περιγραφική γλώσσα είναι η ASN.1 (abstract syntax notation 1).Είναι μία γλώσσα αφαιρετικής περιγραφής δομής δεδομένων. Παρέχει αφαιρετικό συντακτικό ορισμό για επικοινωνία και είναι ανεξάρτητο πλατφόρμας αφού έχει ειδική κωδικοποίηση. Επίσης είναι πολύ σημαντικό στις επικοινωνίες.

15.



Τώρα ας συγκρίνουμε τις δύο γλώσσες, την ASN.1 και την CSN.1 :
η ASN.1 ορίζει δομές δεδομένων με παρόμοιο τρόπο με τη C, η κωδικοποίηση και η αποκωδικοποίηση μετατρέπουν κανόνες σε τοπικές δομές δεδομένων.

το CSN.1 ορίζει έγκυρα κωδικοποιημένα bit streams, το bit stream είναι αποκωδικοποιημένο από έναν αναλυτή που βασίζεται σε μία ορισμένη λειτουργία όπου αναγνωρίζει ένα έγκυρο στοιχείο.

16.

	CSN.1	ASN.1 + unaligned PER	ASN.1 + BER	Tabular
Compactness (ranking)	1st	2nd	4th	3rd
Extensibility (ranking)	4th	3rd	1st	2nd
Optional values	yes	yes	yes	yes
Default values	no	yes	yes	yes
Partial decoding	yes	yes	yes	yes

πίνακας σύγκρισης ASN.1, CSN.1 και πινακοειδούς(tabular) σημειογραφίας

ΚΕΦΆΛΑΙΟ 5: ΕΠΙΚΥΡΩΣΗ

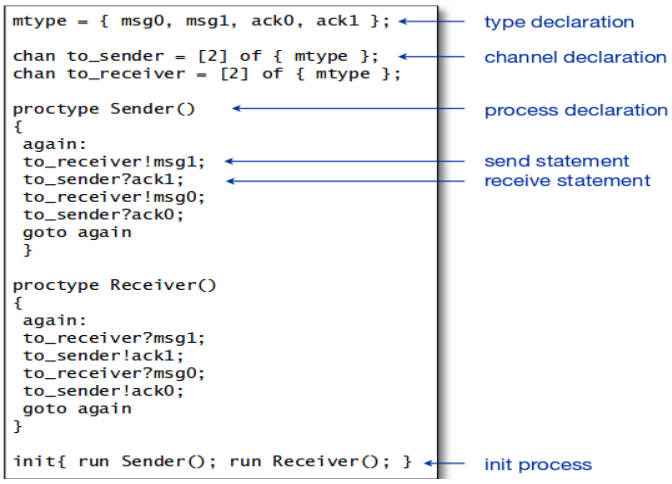
5.1 Μοντέλα επικύρωσης

Μετά τον σχεδιασμό και την διαδικασία του προσδιορισμού του πρωτοκόλλου όπου ουσιαστικά υλοποιήσαμε ένα σημαντικό κομμάτι του, μπένουμε στην διαδικασία της επικύρωσης και στα μοντέλα που δουλεύουμε πάνω σε αυτήν. Υπάρχουν τρεις τύποι μοντέλων επικύρωσης για πρωτόκολλα: η περιγραφή διαδικαστικών κανόνων, το μοντέλο πεπερασμένης κατάστασης, το πρωτότυπο εκτέλεσης. Ο έλεγχος του μοντέλου ουσιαστικά είναι μία τεχνική αυτματοποιημένης επαλήθευσης. Δηλαδή θέλουμε να διαπιστώσουμε αν το πρωτόκολλο ικανοποιεί κάποιες προκαθορισμένες ιδιότητες.

Η περιγραφή των μοντέλων επικύρωσης βασίζεται σε μία περιγραφική γλώσσα την PROMELA(protocol meta language), η οποία είναι και μία εφαρμογή για σχεδιασμό αντιδραστικών συστημάτων. Θα μιλήσουμε εδώ για την Promela όχι εκτενώς αλλά θα αναφέρουμε κάποια βασικά θέματα που αφορούν την γλώσσα αυτή.

Είναι λοιπόν ένα αφαιρετικό μοντέλο που εστιάζει σε διαδικαστικούς κανόνες όπως η συμπεριφορά του πρωτοκόλου και βασίζεται στην επικοινωνία μοντέλων μηχανών πεπερασμένων καταστάσεων. Η επικοινωνία γίνεται με απλοποιημένα μηνύματα δεδομένων και κανάλια. Η Promela δεν είναι γλώσσα εκτέλεσης αλλά μία γλώσσα περιγραφής συστήματος.

17.



παράδειγμα της promela:(δεξιά φαίνονται τα

Στην promela οι διεργασίες περιέχουν μία λίστα δηλώσεων μέσω καναλιών ή μέσω global μεταβλητών. Οι διεργασίες τρέχουν παράλληλα και περιέχουν μία ακολουθία δηλώσεων που είναι είτε εκχωρήσεις τιμών, είτε εκφράσεις(π.χ. συνθήκες ισότητας). Οι δηλώσεις είναι είτε εκτελέσιμες είτε μπλοκαρισμένες. Οι εκχωρήσεις τιμών είναι πάντα εκτελέσιμες .

Οι μεταβλητές δηλώνονται όπως στην C ενώ υπάρχουν 9 τύποι δεδομένων. Οι δομές ελέγχου ροής και οι επανάληψεις είναι παρόμοιες με την C αλλά με λίγο διαφορετικό τρόπο σύνταξης(if...fi, do...od). Οι δηλώσεις και οι δομές ελέγχου ροής μπορεί να προηγούνται από μία ετικέτα, η οποία ετικέτα για παράδειγμα μπορεί να είναι ο προορισμός μίας goto συνθήκης.

Η επικοινωνία πραγματοποιείται με την αποστολή και την λήψη μηνυμάτων από και προς τα κανάλια. Τα κανάλια είναι FIFO ουρές μηνυμάτων. Τα κανάλια θα πρέπει να αρχικοποιηθούν προτού χρησιμοποιηθούν. Δηλώνονται ως εξής:

chan onoma=[xwritikotita] of {listas twn typwn}

Οι τύποι μηνυμάτων δηλώνονται χρησιμοποιώντας την λέξη mtype ως εξής:

mtype = {msg, ack, error}

Εάν το κανάλι το επιτρέπει τα μηνύματα μπορούν να μεταφέρουν μεταβλητές.

Όσον αφορά τα κανάλια υπάρχουν διάφοροι τελεστές, συνθήκες ή λέξεις κλειδιά που ορίζουν το πως γίνεται η ανταλλαγή μηνυμάτων μεταξύ των καναλιών, π.χ. *!full(ch) -> ch!msg* : ελέγχεται αν το κανάλι είναι γεμάτο πριν σταλεί ένα μήνυμα.

Τέλος η είσοδος και η έξοδος γίνονται όπως στην C διαβάζοντας δηλαδή την είσοδο και με printf βγάζοντας την έξοδο. Παρακάτω θα δούμε πως ελέγχονται και επικυρώνονται τα μοντέλα της promela.

5.2 Έλεγχος προτύπου, αξιώσεις ορθότητας

Για να μπορέσουμε να ελέγξουμε και να επικυρώσουμε τα μοντέλα της Promela χρησιμοποιούμε έναν διερμηνέα, το SPIN(simple promela interpreter) το οποίο αναπτύχθηκε στα εργαστήρια της bell από τον holzmann και είναι ανοιχτού κώδικα. Το SPIN συντάσσεται ως εξής:

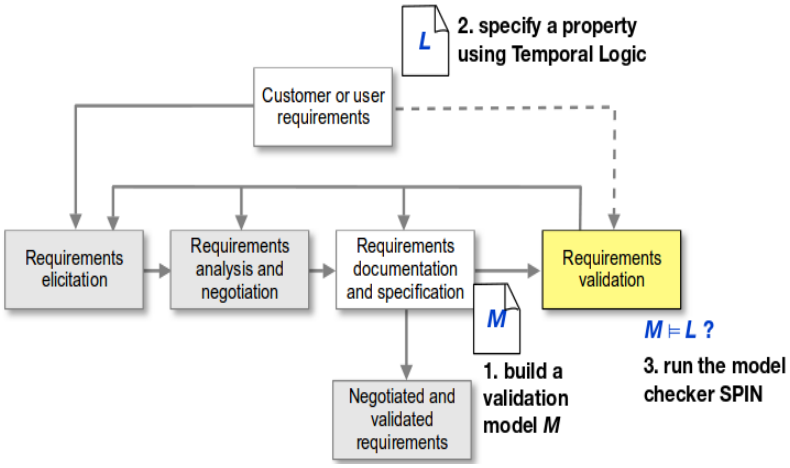
spin [epiloges] file ,όπου οι επιλογές μπορεί να είναι:

με το – r : εκτύπωση ληφθέντων γεγονότων,

με το – c : παράγουν μία προσέγγιση του MSC στο ASCII,

και με το -a : παράγουν έναν αναλυτή

18.



παράδειγμα ελέγχου μοντέλου με το SPIN

Για την μοντελοποίηση διεργασιών στο μοντέλο χαμηλότερου επιπέδου τα μηνύματα δεδομένων όπως και οι αποκρίσεις περνάνε από τον αποστολέα στον παραλήπτη, και τα δύο περιέχουν ένα εναλασσόμενο μπιτ που καθορίζει εάν τα δεδομένα ή η απόκριση βρίσκονται στον αποστολέα ή στον παραλήπτη.

Κατά την ανάκτηση των δεδομένων υπάρχει σιγουριά πως αυτά αποτελούν μία ακολουθία ακεραίων. Παρ' όλα αυτά ο παραλήπτης θα πρέπει να δεχτεί μόνο τα μηνύματα δεδομένων με την σωστή τιμή ακεραίου.

Το πρωτόκολλο τρέχει σε τυχαίες προσομοιώσεις όμως δεν γνωρίζουμε εάν δουλεύει σωστά για κάθε κατάσταση, γι αυτό λοιπόν χρειάζεται ο ελεγκτής. Εδώ θα παραθέσουμε κάποιες αξιώσεις ορθότητας όπως διατυπώθηκαν από τον *Holzmann (1991):*

-*ασφάλεια*: ένα σύνολο ιδιοτήτων που το σύστημα δεν μπορεί να παραβιάσει,

-*ζωντάνια:* ένα σύνολο ιδιοτήτων που το σύστημα πρέπει να ικανοποιεί,

-προσβάσιμες και μη-προσβάσιμες καταστάσεις (ιδιότητες κατάστασης),

-εφικτές και ανέφικτες καταστάσεις (ιδιότητες μονοπατιού),

-*αμετάβλητο σύστημα:* παραμένει σε κάθε προσβάσιμη κατάσταση,

-*ισχυρισμός διεργασίας:* παραμένει μόνο σε συγκεκριμένες προσβάσιμες καταστάσεις

Κατά τον έλεγχο το SPIN ελέγχει για αδιέξοδα και για κύκλους που δεν παρουσιάζουν πρόοδο. Δεν υπάρχει τρόπος να ορίσουμε την σχετική ταχύτητα, αλλά τότε υπάρχει η πιθανότητα μία διεργασία να είναι πολύ αργή και μία άλλη πολύ γρήγορη και άρα η αργή διεργασία δε θα μπορέσει να εκτελέσει την επόμενη κατάσταση. Παρ' όλα αυτά το SPIN μας επιτρέπει να ελέγχουμε το μοντέλο κάτω από τον ισχυρισμό της *ευθύτητας*.

Δεν υπάρχει κάποιος ισχυρισμός που να σχετίζεται με την ταχύτητα εκτέλεσης, παρ' ότι είναι πιθανό να εμφανιστούν αχανής καθυστερήσεις. Μία δίκαιη αντιμετώπιση των διεργασιών κατά την εκτέλεσή τους εκφράζεται από τον ισχυρισμό της *πεπερασμένης προόδου*. Δηλαδή οποιαδήποτε διεργασία που μπορεί να εκτελέσει μία κατάσταση θα βρεθεί εν τέλει να την εκτελεί.

Ακόμη υπάρχουν κάποιες ετικέτες τελικών καταστάσεων δηλαδή, κάποιες ετικέτες με το πρόθεμα *end* που δηλώνουν κάποια έγκυρη τελική κατάσταση. Οι προεπιλεγμένες τελικές καταστάσεις είναι αυτές που βρίσκονται στο τέλος του κώδικα της διεργασίας. Οι ετικέτες τελικών καταστάσεων αφήνουν τον ελεγκτή να διαλέξει ανάμεσα σε έγκυρες και μη-έγκυρες τελικές καταστάσεις, αλλά από μόνο του το SPIN ελέγχει για μη-έγκυρες καταστάσεις. Μία κατάσταση συστήματος είναι έγκυρη εάν όλες οι διεργασίες έχουν φτάσει σε έγκυρη τελική κατάσταση και όλες οι ουρές μηνυμάτων είναι άδειες.

Στην promela χρησιμοποιείται αλγόριθμος σημαφόρου, ο οποίος προϋποθέτει ασφάλεια και ζωντάνια για να λειτουργήσει. Ασφάλεια εννοούμε πως μόνο μία διεργασία μπορεί να είναι στο critical section κάθε στιγμή, ενώ με τον όρο ζωντάνια εννοούμε πως αν μία διεργασία θέλει, τελικά θα μπει στο critical section. Κατά τον έλεγχο ζωντάνιας ψάχνουμε για ατέρμονους κύκλους.

Θέλουμε να δείξουμε πως ένα μήνυμα λαμβάνεται τουλάχιστον μία φορά. Δεν υπάρχει κάποια άπειρη ακολουθία διπλότυπων μηνυμάτων εκτός κι αν έχουν παραμορφωθεί. Παρ' όλα αυτά δεν θα πρεπε *ποτέ* ο παραλήπτης να φτάσει στην κατάσταση του να εντοπίζει ένα διπλότυπο και να επισκέπτεται αυτή την κατάσταση ξανά χωρίς να έχει λάβει κάποιο έγκυρο μήνυμα. Γιατί αν συμβεί αυτή η περίπτωση ο παραλήπτης δεν πρόκειται αργότερα να λάβει έγκυρο μήνυμα. Ονομάστηκε λοιπόν ο παραπάνω ισχυρισμός *ποτέ.*

Περνώντας τώρα σε κάποιους άλλους ισχυρισμούς, τους ισχυρισμούς “ίχνους”, μπορούμε να πούμε πως αυτοί περιγράφουν τις ιδιότητες ορθότητας των καναλιών μηνυμάτων. Αιτούνται για να λαβουν και να στείλουν λειτουργίες στα κανάλια μηνυμάτων. Οι ισχυρισμοί αυτοί χρησιμοποιούνται για να συγκεκριμενοποιήσουν έγκυρες ακολουθίες γεγονότων και επιτρέπονται σ αυτούς μόνο απλές λειτουργίες λήψης και αποστολής.

Συμπερασματικά η επικύρωση περιλαμβάνει ελέγχους διαφορετικών ιδιοτήτων. Βασικές ιδιότητες ορθότητας μπορεί να εκφραστούν με ισχυρισμούς και ειδικές ετικέτες στην promela. Οι προσωρινές απαιτήσεις αναφέρονται στον έλεγχο ροής και μπορούν να συγκεκριμενοποιηθούν από τον ισχυρισμό “ποτέ”.

5.3 Προσδιορίζοντας την ορθότητα

Για να συνεχίσουμε από πριν δεν είναι εύκολο να μετατρέψουμε τις απαιτήσεις σε ισχυρισμούς “ποτέ”. Οπότε χρησιμοποιούμε έναν πιο βολικό τρόπο επισημοποίησης χρησιμοποιώντας την LTL(linear temporal logic). Η φόρμουλα της LTL είναι πιο εύκολο να κατανοηθεί απ' ότι οι ισχυρισμοί “ποτέ”. Οι φόρμουλες μπορεί να αλλάζουν τις τιμές αληθείας τους δυναμικά καθώς αλλάζει η κατάσταση του συστήματος. Οι LTL φόρμουλες ορίζονται από άπειρες ακολουθίες μεταβάσεων. Η LTL επεκτείνει την προτασιακή λογική με τυπικούς φορείς. Καλά σχηματισμένες φόρμουλες είναι οι προτασιακές καταστάσεις που περιλαμβάνουν και το σωστό/λάθος, και επίσης εάν έχουμε ένα p και ένα

q που είναι καλά σχηματισμένα τότε και τα α*p, p*β*q, (p) είναι επίσης καλά σχηματισμένες φόρμουλες όπου τα α,β είναι μοναδιαίοι/δυαδικοί τελεστές.

19.
```
ltl ::= operand | ( ltl ) | ltl binary_operator ltl |
        unary_operator ltl
```

παράδειγμα ltl γραμματικής

Operator	Description	Definition
X	Next	$\sigma[i] \models X\ p \Leftrightarrow \sigma_{i+1} \models p$

Operator	Description	Definition
W	Weak Until	$\sigma[i] \models (p\ W\ q) \Leftrightarrow \sigma_i \models q \vee (\sigma_i \models q \wedge \sigma[i+1] \models (p\ W\ q))$
U	Strong Until	$\sigma[i] \models (p\ U\ q) \Leftrightarrow \sigma_i \models (p\ W\ q) \wedge \exists\ j, j \geq i\ \sigma_j \models q$

Operator	Description	Definition
□	Always (also called Globally, G)	$\sigma \models \Box p \Leftrightarrow \sigma \models (p\ W\ false)$
◊	Eventually (also called Finally, F)	$\sigma \models \Diamond p \Leftrightarrow \sigma \models (true\ U\ p)$

στις παραπάνω τρεις εικόνες φαίνονται παραδείγματα ltl τελεστών

Κάθε κατάσταση του συστήματος στην οποία το p είναι αληθές τελικά ακολουθείται από μία κατάσταση του συστήματος στην οποία το q είναι αληθές. Αλλά αυτό δεν μπορεί να εκφραστεί από τον συμβολισμό p-->q αλλά από το (!p∨q) και ισχύει ως προτασιακός ισχυρισμός στην πρώτη κατάσταση του συστήματος.

Τώρα ας δούμε πως η LTL εφαρμόζεται στο SPIN. Το SPIN δέχεται:

- τα προτασιακά σύμβολα συμπεριλαμβανομένων του true / false,

- τους χρονικους τελεστές *always*[], *eventually*<>, *strong until* (U)

- τους λογικούς τελεστές *and*(&&), *or*(||), *not*(!)

- συμπεράσματικός (→) και ισοδυναμίας (↔)

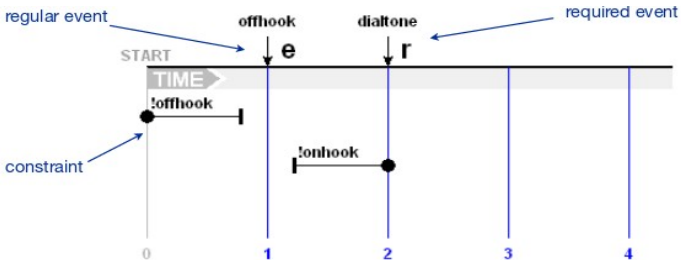
Οι αριθμητικές και σχεσιακές εκφράσεις δεν υποστηρίζονται αλλά μπορούν να αντικατασταθούν με προτασιακές προτάσεις. Επίσης θα πρέπει να οριστούν τα ακόλουθα σύμβολα:

ds–αποστολή δεδομένων, dr–λήψη δεδομένων, od–αποστολή άλλων δεδομένων, err–λάθος μήνυμα,

αυτά τα σύμβολα αναφέρονται σε λειτουργίες μηνυμάτων καναλιών

Μία ακόμα μέθοδος για να ορίσουμε χρονικούς ισχυρισμούς είναι το *χρονοδιάγραμμα(timeline)*, το οποίο ορίζει αιτιώδεις σχέσεις μεταξύ των γεγονότων

20.



παράδειγμα χρονοδιαγράμματος

Τα μοντέλα της promela περιγράφουν διεργασίες που επικοινωνούν με μηχανές πεπερασμένων καταστάσεων. Οι διεργασίες μπορούν να περιγραφούν από πεπερασμένα αυτόματα. Το παράγωγο των διεργασιών δίνει το χώρο καταστάσεων.

Με την τυπική έννοια της αποδοχής δεν μπορούμε να εκφράσουμε εν εξελίξη, δυνητικά απεριόριστες εκτελέσεις. Ένα αποδεκτό “τρέξιμο” ενός πεπερασμένου αυτόματου είναι μία πεπερασμένη μετάβαση που οδηγεί σε μία αποδεκτή κατάσταση τερματισμού. Εδώ έχουμε να κάνουμε με μη-πεπερασμένες μεταβάσεις ακολουθιών που λέγονται *ω-runs*.

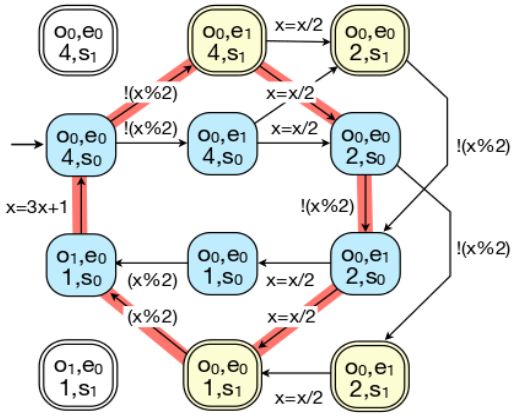
Ένα αποδεκτό *ω-run* ενός πεπερασμένου αυτόματου κατάστασης είναι οποιδήποτε πεπερασμένο “τρέξιμο” που περιέχει μία αποδεκτή κατάσταση. Τα αυτόματα αυτά δέχονται ακολουθίες εισόδου που ορίζονται ως προς μη-πεπερασμένα “τρεξίματα”. Ένα τέτοιο αυτόματο αποδέχεται εάν και μόνο εάν μία αποδεκτή κατάσταση είναι επισκέψιμη πολύ συχνά. Τα παραπάνω αυτόματα λέγονται Buchi αυτόματα.

Η LTL έχει μία απευθείας σύνδεση με τα παραπάνω αυτόματα: μπορεί να φανεί πως σε κάθε LTL φόρμουλα υπάρχει ένα Buchi αυτόματο που δέχεται ακριβώς τα “τρεξίματα” που προσδιορίζονται από τη φόρμουλα. Το SPIN μεταφράζει την LTL φόρμουλα σε ισχυρισμούς *never*, οι οποίοι παριστάνουν τα Buchi αυτόματα. Τότε ο ελεγκτής τσεκάρει εάν το Buchi αυτόματο ταιριάζει με το τρέξιμο του συστήματος.

Ένα παράγωγο αυτόματο αποτελείται από το καρτεσιανό παράγωγο του συνόλου καταστάσεων των εμπλεκόμενων αυτόματων και μεταβάσεων. Έχουμε το ασύγχρονο παράγωγο, όπου όλες οι πιθανές διεμπλοκές των διεργασιών του συστήματος περιγράφονται από αυτό. Ακόμη έχουμε τα σύγχρονα παράγωγα , όπου οι σύγχρονες εκτελέσεις αναπαριστώνται από σύγχρονα παράγωγα.

Για τον έλεγχο ορθότητας αρχικά ορίζουμε έναν ισχυρισμό *never* όπου θεωρούμε πως ένα χ θα γίνει 1. Η ορθότητα ενός ισχυρισμού *never* ελέγχεται υπολογίζοντας το σύγχρονο παράγωγο του χώρου κατάστασης αυτόματου και τον ισχυρισμό αυτόματου. Όπως φαίνεται και στο σχήμα υπάρχει ένας κύκλος αποδοχής όπου μια μη-πεπερασμένη ακολουθία φτάνει σε μία κατάσταση αποδοχής.

21.



Θα μπορούσε κάποιος να αναρωτηθεί γιατί να χρησιμοποιεί το SPIN αρνητικούς ισχυρισμούς (όπως οι ισχυρισμοί *never*). Οι θετικοί ισχυρισμοί αποδεικνύουν πως η γλώσσα του αυτόματου συστήματος περιλαμβάνεται στη γλώσσα του ισχυρισμού του αυτόματου. Αυτό βέβαια έχει σαν μειονέκτημα πως ο χώρος κατάστασης για την συμπερίληψη της γλώσσας έχει το πολύ το μέγεθος του καρτεσιανού παράγωγου. Εν αντιθέσει ο αρνητικός ισχυρισμός αποδεικνύει ότι η γλώσσα της διατομής του αυτόματου είναι άδεια και εδώ το πλεονέκτημα είναι πως στην καλύτερη περίπτωση ο χώρος κατάστασης είναι μικρότερος ακόμα και μηδενικός.

Κάνοντας έναν απολογισμό της ενότητας μπορούμε να πούμε πως το SPIN δεν δείχνει απ' ευθείας την ορθότητα αλλά προσπαθεί να βρεί μετρήσιμα παραδείγματα για να προσδιορίσει τους ισχυρισμούς ορθότητας. Οι ιδιότητες της ζωντάνιας εκφράζονται απο ισχυρισμούς *never* ή την LTL φόρμουλα και απαιτούν το μεγαλύτερο υπολογισμό για την επαλήθευση.

ΚΕΦΆΛΑΙΟ 6: ΤΕΧΝΙΚΈΣ ΣΧΕΔΙΑΣΜΟΎ ΚΑΙ ΕΚΤΈΛΕΣΗΣ

6.1 Σχεδιαστικές τεχνικές, χτίσιμο μπλοκ πρωτοκόλλων, εκτέλεση μηχανών καταστάσεων

Κατά τις σχεδιαστικές αποφάσεις η επικοινωνία των πρωτοκόλλων υπόκειται σε περιορισμούς πόρων. Ένα πρωτόκολλο επικοινωνίας μπορεί να είναι κομμάτι ενός μεγαλύτερου συστήματος το οποίο με τη σειρά του βάζει κι άλλους περιορισμούς. Οι περιορισμοί πόρων μπορεί να είναι εξαρτημένοι ή συγκρουόμενοι όμως είναι απίθανο το να συναντήσεις όλους τους περιορισμούς μαζί. Οι σχεδιαστικές τεχνικές βοηθάνε στο να βρούμε ανταλλαγές περιορισμών.

Ο σχεδιασμός του συστήματος περιορίζεται από τους περιορισμούς των πόρων. Κάποιοι από τους βασικούς περιορισμούς πόρων είναι οι παρακάτω:

χρόνου(χρόνος απόκρισης, ποσότητα πληροφορίας), χώρου(μνήμη, χωρητικότητα του buffer, εύρος ζώνης), υπολογισμού, εργασίας, χρημάτων. Κι επίσης υπάρχουν και κάποιοι κοινωνικοί περιορισμοί όπως τα στάνταρ και οι απαιτήσεις της αγοράς.

Εντοπίζοντας τους πόρους θα πρέπει να ταυτοποιήσουμε τον πιο περιορισμένο πόρο, δηλαδή το δεσμευτικό περιορισμό ή αλλιώς την συμφόρηση. Είναι σημαντικό να αφαιρέσουμε την συμφόρηση γιατί έτσι θα βρεθούμε αντιμέτωποι με άλλες συμφορήσεις στις οποίες θα κληθούμε να κάνουμε το ίδιο. Ο γενικότερος στόχος μας είναι να ισσοροπήσουμε το όλο σύστημα κάτι που πολλές φορές απλά δεν γίνεται. Παρ' όλα αυτά υπάρχουν κάποιες σχεδιαστικές τεχνικές με τις οποίες μπορούμε να βρούμε κάποιες λύσεις. Το σκεπτικό είναι να αρχίσουμε με την ταυτοποίηση των περιορισμών, και μετά να ανταλλάσουμε έναν πόρο για έναν άλλο ώστε να μεγιστοποιήσουμε την χρησιμότητα.

Μία λύση είναι να χρησιμοποιήσουμε *πολυπλεξία*, δηλαδή να κάνουμε διαμοιρασμό των πόρων.Βέβαια εκεί θα βρεθούμε αντιμέτωποι με το δίλημμα του να προτιμήσουμε το χρόνο και το χώρο ή τα χρήματα.

Επίσης μπορεί να χρησιμοποιηθεί η μέθοδος του *παραλληλισμού* όπου διαιρούμε τις εργασίες σε μικρότερες ανεξάρτητες υποκατηγορίες. Σε αντίθεση με πριν εδώ κερδίζουμε και αρκετά μάλιστα σε χρόνο αλλά θα πρέπει να ξέρουμε πως ο υπολογισμός μας θα έχει μικρότερη ακρίβεια. Ο βαθμός του παραλληλισμού μπορεί να υπολογιστεί εύκολα από το γινόμενο του χρόνου απόκρισης επί τον αριθμό των εργασιών ανά μονάδα χρόνου(*throughput*).

Ένας τρίτος τρόπος είναι *ανά παρτίδες*, δηλαδή ομαδοποιούμε τις εργασίες μαζί ώστε να φτάσουμε τον έμμεσο χρόνο υπολογισμού(*overhead*). Εδώ θα πάρουμε μεν καλύτερο χρόνο απόκρισης αλλά θα χάσουμε στον αριθμό εργασιών ανά μονάδα χρόνου(*throughput*). Βέβαια θα πρέπει να λάβουμε υπ' όψιν μας πως η *ανά παρτίδες* ομαδοποίηση είναι αποτελεσματική μόνο αν ο έμμεσος χρόνος υπολογισμού(*overhead*) για N εργασίες είναι μικρότερος από N φορές τον έμμεσο χρόνο υπολογισμού για μία εργασία.

Μία άλλη περίπτωση είναι να αξιοποιήσουμε την *τοπικότητα*, δηλαδή τα δεδομένα που προσπελάσαμε πρόσφατα να μείνουν στην cache. Εδώ βέβαια μπορεί να κερδίζουμε σε χρόνο έχοντας γρήγορη πρόσβαση στα πρόσφατα προσπελασθέντα δεδομένα αλλά χάνουμε σε χώρο αφού η cache θα 'ναι απασχολημένη με τα δεδομένα αυτά και δε θα υπάρχει αρκετός χώρος για άλλη πληροφορία. Για παράδειγμα, όταν μεταφέρεται ένα αρχείο αυτό τεμαχίζεται σε πολλά μικρότερα πακέτα. Αν κρατήσει τα μη-αναγνωρίσιμα πακέτα στην cache θα μπορεί ν επαναφέρει τα πακέτα εύκολα χωρίς να πρέπει να τα ξαναγεννήσει.

Ένας άλλος τρόπος είναι οι τεχνικές του *binding* και της *έμμεσης αναφοράς(indirection).* Αυτό που κάνει το binding είναι να αναφέρεται από κάτι αφηρημένο σε ένα συγκεκριμένο στιγμιότυπο ενώ η έμμεση αναφορά ουσιαστικά παίρνει κάτι αφηρημένο και βρίσκει τις τιμές του.

Το *virtualization* αποτελεί συνδυασμό *πολυπλεξίας*(multiplexing) και *έμμεσης αναφοράς*(indirection). Η λειτουργία του είναι να επιτρέπει το διαμοιρασμό ενός πόρου σαν να μπορούσε να χρησιμοποιηθεί μονοσήμαντα.

Η περίπτωση της *μαλακής κατάστασης(*soft state), φανερώνει μία κατάσταση που περιέχει πληροφορία που καθορίζει τη μελλοντική συμπεριφορά. Η κατάσταση μπορεί να αποθηκευτεί στο δίκτυο και μπορούμε να εισάγουμε την κατάσταση σε ένα κύκλωμα μεταγωγής τηλεφωνικού δικτύου. Όμως η ατελής μετακίνηση οδηγεί σε προβλήματα, όπως το ότι οι πόροι παραμένουν δεσμευμένοι. Το να προσπαθήσουμε να διορθώσουμε τα λάθη ή τους κακούς τερματισμούς μπορεί να οδηγήσει σε περίπλοκο σχεδιασμό. Έρχεται λοιπόν να αποτελέσει λύση η *μαλακή κατάσταση* δηλαδή μία κατάσταση

που δεν είναι μόνιμη αλλά ανανεώνεται(απαιτεί ευρυζωνικότητα) και μετά από κάποια ώρα θα αφαιρεθεί.

Οι επικοινωνιακές οντότητες συχνά χρειάζεται να ανταλλάξουν κατάσταση. Αυτή η ανταλλαγή καλό είναι να γίνεται ρητά. Ας πούμε πως ένα πρωτόκολλο μεταφοράς αρχείων στέλνει ένα πακέτο σε κομμάτια και τα συναρμολογεί πάλι στον παραλήπτη. Αλλά ο παραλήπτης πως μπορεί να εντοπίσει κάποια απώλεια πακέτου ; Αυτό μπορεί να γίνει είτε εμμέσως εξετάζοντας το ωφέλιμο φορτίο, κάτι που απαιτεί γνώση στο επίπεδο εφαρμογής. Είτε ρητά αντιστοιχίζοντας ακολουθίες αριθμών στα πακέτα από τον αποστολέα.

Τώρα εάν βρεθούμε στην περίπτωση όπου η κατάσταση του συστήματος εξαρτάται από την τιμή της μεταβλητής, τότε μικρές διακυμάνσεις της τιμής αυτής γύρω από το όριο θα 'χουν ως αποτέλεσμα συχνές αλλαγές στην κατάσταση. Αυτό όμως δεν το θέλουμε γιατί μπορεί να οδηγήσει σε ανεπιθύμητες συμπεριφορές. Η τεχνική λοιπόν της *υστέρησης* που δουλεύουμε σ' αυτήν την περίπτωση, αυτό που κάνει είναι να εφαρμόζει το ελάχιστο όριο εξάρτησης ώστε να αποφύγει τις ταλαντώσεις που αναφέραμε πιο πριν.

Ο *διαχωρισμός των δεδομένων* ανά-μονοπάτι είναι μία μεθοδος που μπορεί να αυξήσει τον αριθμό εργασιών ανά μονάδα χρόνου(throughput), αλλά προϋποθέτει την ύπαρξη της πληροφορίας κατάστασης στο δίκτυο.

Βελτιστοποιώντας την πιο κοινή περίπτωση : πολλά συστήματα υπακούν στον νόμο του Pareto που λέει πως το 20% του κώδικα χρησιμοποιείται μόνο κατά το 80% του χρόνου χρήσης. Άρα αν βελτιστοποιήσουμε αυτό το 20% βελτιώνουμε τη συνολική απόδοση.

Άλλη περίπτωση είναι αυτή της *επεκτασιμότητας*. Ο σχεδιασμός μας είναι τέτοιος που να επιτρέπει μελλοντικές επεκτάσεις στο σύστημα. Ας πούμε το header του IP πακέτου περιέχει έναν αριθμό έκδοσης τέτοιον που να υποδηλώνει τη μορφή του υπόλοιπου κομματιού του header.

Παραπάνω παρουσιάσαμε μία σειρά από σχεδιαστικές τεχνικές με ξεχωριστά πλεονεκτήματα η κάθε μία. Μπορούμε να χρησιμοποιήσουμε μία ή συνδυασμό αυτών των τεχνικών ανάλογα την περίπτωση του συστήματος που θέλουμε να σχεδιάσουμε και τις απαιτήσεις που θέλουμε από αυτό λαμβάνοντας βέβαια υπ' όψιν και τα αντίστοιχα κόστη.

Τώρα θα περάσουμε σε κάποια θέματα που αφορούν την αρχιτεκτονική. Η αρχιτεκτονική βασίζεται στην αποσύνθεση του όλου σε λειτουργικές μονάδες. Η πιο κοινή προσέγγιση είναι *η διαστρωμάτωση του πρωτοκόλλου*. Κάθε επίπεδο του πρωτοκόλλου ορίζει και ένα επίπεδο

αφαιρετικότητας. Ο σχεδιαστικός μας στόχος είναι η ολοκλήρωση των σχετικών λειτουργιών με σαφώς καθορισμένους στόχους. Μία εναλλακτική προσέγγιση είναι αυτή της *ολοκληρωμένης επεξεργασίας του επιπέδου(intergrated layer processing),* δηλαδή αυτό που κάνουμε είναι να επεξεργαζόμαστε τα δεδομένα σε μία ολοκληρωμένη εφαρμογή του επιπέδου, με στόχο να ελαχιστοποιήσουμε την πρόσβαση στα δεδομένα και την αντιγραφή λειτουργιών ειδικά για τα πάνω επίπεδα.

Οι βασικές μέθοδοι που τα περισσότερα πρωτόκολλα εκτελούν είναι πρώτον ο έλεγχος του λάθους, όπου εντοπίζονται και διορθώνονται τα λάθη κατά τη μεταφορά και δεύτερον ο έλεγχος ροής όπου προσαρμόζεται ο ρυθμός μεταφοράς στο ρυθμό εξυπηρέτησης του δέκτη. Κι οι δύο μέθοδοι λειτουργούν ως τερματικό-με-τερματικό μηχανισμοί και συνήθως εκτελούνται στο επίπεδο σύνδεσης και στο επίπεδο μεταφοράς.

Για να μιλήσουμε τώρα για τεχνικές εκτέλεσης πρέπει αρχικά να πούμε ότι η συμπεριφορά των πρωτοκόλλων μοντελοποιείται από extended-FSM. Υπάρχουν στάνταρ τεχνικές να μετατρέψουμε διαγράμματα μηχανών καταστάσεων σε κώδικα: εμφωλευμένες switch/case, οδηγούμενο-από-πίνακα, σχεδιαστικό πρότυπο κατάστασης. Υπάρχουν γεννήτορες κώδικα διαθέσιμοι για UML/SDL μηχανές καταστάσεων κι επίσης διάφοροι πίνακες που υποστηρίζουν πίνακες μετάβασης καταστάσεων.

Κλείνοντας αυτήν την ενότητα θα αναφέρουμε λίγα πράματα για το πως πάμε από FSM σε κώδικα. Κατ' ρχάς όλες οι καταστάσεις και τα γεγονότα πρέπει να είναι καλά ορισμένα. Η μηχανή κατάστασης ή ο πίνακας μετάβασης καταστάσεων θα πρέπει να καλύπτει όλους τους συνδυασμούς καταστάσεων και γεγονότων και να καθορίζει την αντίστοιχη ενέργεια, δηλαδή θα πρέπει να περιλαμβάνει όλες τις καταστάσεις και τις μεταβάσεις αυτών για οποιαδήποτε είσοδο ή οποιοδήποτε ερέθισμα. Είναι τέλος πολύ σημαντικό να βελτιστοποιούμε πάντα την πιο κοινή πείπτωση όπως αναφέρθηκε και προηγούμενα δηλαδή με τον νόμο του *Pareto*.

ΚΕΦΑΛΑΙΟ 7: ΠΡΟΣΟΜΟΙΩΣΗ

7.1 Πρότυπα προσομοίωσης, διεργασίες άφιξης

Μετά το κεφάλαιο των τεχνικών σχεδιασμού και εκτέλεσης και πριν μιλήσουμε για την εκτέλεση, θα μας απασχολήσει το κομμάτι της εκτίμησης και της προσομοίωσης. Αρχικά λοιπόν θα δούμε κάποιες τεχνικές εκτίμησης της απόδοσης οι οποίες είναι:

- πειραματικά: δηλαδή μετρώντας την απόδοση για μία συγκεκριμένη περίπτωση σε πραγματικό περιβάλλον,

- με προσομοίωση: εδώ γίνεται αριθμητική αξιολόγηση του μοντέλου του συστήματος αλλά σε τεχνητό περιβάλλον,

- ανάλυση: περιγράφονται οι ιδιότητες μίας μαθηματικής αφαίρεσης του συστήματος

Η τρίτη περίπτωση αναφέρεται σε μοντέλα μαθηματικού συστήματος τα οποία χωρίζονται σε στατικά και δυναμικά. Τα στατικά καλύπτουν μία συγκεκριμένη κατάσταση του συστήματος και οι αλλαγές της κατάστασης δεν λαμβάνονται υπ' όψιν. Από την άλλη τα δυναμικά μοντέλα αποτυπώνουν τις αλλαγές στις καταστάσεις ανά τον χρόνο. Το δυναμικό με τη σειρά του χωρίζεται σε διακριτού και συνεχούς χρόνου. Στο συνεχές οι μεταβλητές συστήματος είναι συνεχής συναρτήσεις ανά τον χρόνο, ενώ στο διακριτό οι αλλαγές καταστάσεων συμβαίνουν σε διακριτά χρονικά σημεία. Η επιλογή του μοντέλου εξαρτάται από τους στόχους και τους σκοπούς μας.

Ο στόχος της γνώσης προσομοίωσης και πειραματισμού είναι η εκτίμηση κάποιων ιδιοτήτων του συστήματος, δηλαδή η απόκτηση γνώσης πάνω στη συμπεριφορά του συστήματος, να μπορέσουμε να κρίνουμε την απόδοση, να χρησιμοποιήσουμε τα αποτελέσματα για να βελτιώσουμε το σχεδιασμό και να μειώσουμε το κόστος. Κατά τη διαδικασία εκτίμησης χρησιμοποιούμε κάποιες παραμέτρους που καθορίζουν τη συμπεριφορά και κάποιες μετρήσεις που χαρακτηρίζουν το σύστημα.

Στην προσομοίωση των πρωτοκόλλων επικοινωνίας τα μοντέλα είναι συνήθως δυναμικά και διακριτού χρόνου. Ακολουθούμε μία γενική διαδικασία που περιλαμβάνει τα εξής στάδια: εκτέλεση του μοντέλου για να δούμε τη συμπεριφορά του, ορίζουμε τις παραμέτρους, τρέχουμε την προσομοίωση, παρατηρούμε τις μετρήσεις, κάνουμε εκτίμηση των αποτελεσμάτων.

Κατά την προσομοίωση της ουράς η κατάσταση του μοντέλου περιλαμβάνει τα παρακάτω: χρόνος προσομοίωσης, ουρά(FIFO) προσομοίωσης των εργασιών, κατάσταση του σέρβερ(αδρανής ή απασχολημένη), γεγονότα όπως άφιξη και αναχώρηση. Κατά την άφιξη προστίθεται στην ουρά μία εργασία, ενώ κατά την αναχώρηση αφαιρείται μία εργασία από την ουρά. Στην αρχικοποίηση γεννιέται ο πρώτος χρόνος αναχώρησης.

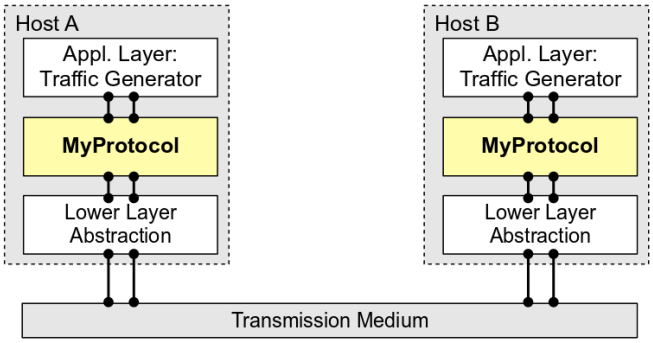
Μιλώντας για τον αντίτκυπο του χρόνου προσομοίωσης είναι γεγονός πως για σύντομες προσομοιώσεις δηλαδή για μικρό αριθμό εργασιών, τα αποτελέσματα αποκλίνουν ιδιαίτερα από τις αναλυτικές τιμές. Για μεγαλύτερες σε διάρκεια προσομοιώσεις τα “τρεξίματα” φαίνεται να δίνουν καλύτερα αποτελέσματα. Αυτό συμβαίνει επειδή αρχικά η ουρά είναι άδεια και χρειάζεται χρόνο για να γεμίσει.

Περνώντας τώρα στην αντικειμενοστραφή σχεδίαση θα δούμε πως υπάρχουν ξεχωριστές ενότητες όπου τα αντικείμενα προσομοιώνονται, υπάρχει ένας “αποστολέας” γεγονότων που λειτουργεί ως handler και ένας γεννήτορας φόρτωσης. Οι ενότητες αυτές είναι extended-FSM και επικοινωνούν με ανταλλαγή μηνυμάτων. Η δομή των δεδομένων για μελλοντικά σύνολα γεγονότων περιλαμβάνει ουρά προτεραιότητας και γεγονότα που ταξινομούνται σύμφωνα με το χρόνο ενεργοποίησης.

Μία τυπική εγκατάσταση προσομοίωσης δικτύου τρέχει με πρωτόκολλα που είναι επικοινωνούντα FSM. Τα πρωτόκολλα αυτά τροφοδοτούνται από μία γεννήτρια φορτίου. Η μηχανή προσομοίωσης περνάει τα μηνύματα μεταξύ των ενοτήτων και ενεργοποιεί τον χρονιστή γεγονότων.

Στη συνέχεια κατά την προσομοίωση του συστήματος πρώτα για το φορτίο του μοντέλου γίνεται περιγραφή της εξέλιξης των αιτημάτων που φτάσαν στο σύστημα και στο σφάλμα του μοντέλου περιγράφεται η απόκλιση από την κανονική συμπεριφορά.

22.



παράδειγμα τυπικού μοντέλου συστήματος

Μία διεργασία άφιξης είναι μία στοχαστική περιγραφή των αφίξεων. Κάποιοι από τους τύπους άφιξης είναι: σταθερός ρυθμός μπιτ(CBR), διεργασία ανανέωσης, διεργασία Markov.

Συμπερασματικά τα αποτελέσματα των προσομοιώσεων είναι απλά μία εκτίμηση της πραγματικής συμπεριφοράς του συστήματος. Επίσης υπάρχουν παγίδες που μπορεί να οδηγήσουν σε κακή εξαγωγή αποτελεσμάτων όπως κακή γεννήτρια τυχαίων αριθμών ή μη επαρκής χρόνος προσομοίωσης.

7.2 Προσομοίωση δικτύου, OMNeT

Πιο πάνω μιλήσαμε για τις δίαφορες τεχνικές προσομοίωσης, τις ιδιότητές τους και το πως μπορούμε να δουλέψουμε με αυτές αλλά τώρα θα δούμε ένα περιβάλλον πάνω στο οποίο μπορούμε να υλοποιήσουμε την προσομοίωση και αυτό είναι το OMNeT. Αφορά έναν προσομοιωτή διακριτών γεγονότων που βασίζεται στα επιμέρους συστατικά και παρέχει βασικά εργαλεία προσομοίωσης του οποίου ο κώδικας είναι ανοιχτός.

Ένα μοντέλο προσομοίωσης αποτελείται από ενότητες(επικοινωνούντα-FSM). Οι ενότητες επικοινωνούν ανταλλάσοντας μηνύματα μέσα από συνδέσεις. Οι ενότητες εκτελούν μία συγκεκριμένη συμπεριφορά της εφαρμογής και είναι αντικείμενα της C++.

Οι πύλες είναι καλά ορισμένες διεπαφές. Η λειτουργικότητα μέσα σε μία ενότητα είναι ανεξάρτητη των συνδέσεων, δηλαδή μία ενότητα μπορεί να μεταχειριστεί ως “μαύρο κουτί” κι επίσης

είναι ανταλλάξιμη. Μία ενότητα στέλνει μήνυμα σε εξωτερικές πύλες αλλά κι απ' ευθείας σε άλλη ενότητα.

Ας δούμε τώρα τα 5 βήματα κατά τη διαδικασία “γραψίματος” μιίας προσομοίωσης:

1. όρισε τις ενότητες και την τοπολογία δικτύου(.ned).

2. όρισε τα μηνύματα(.msg)

3. εφάρμοσε τη συμπεριφορά των απλών ενοτήτων(.cc)

4. κάνε compile το project(makefile)

5. όρισε της παραμέτρους της προσομοίωσης(omnetpp.ini)

Πιο κάτω θα δούμε κάποια πράματα όσον αφορά τα ασύρματα δίκτυα και την συμπεριφρά και επικοινωνία αυτών.

Κάποια από τα αρνητικά των ασύρματων δικτύων είναι όσον αφορά τις ασύρματες συνδέσεις τα σφάλματα πακέτων, η απώλεια πακέτων, καθυστέρηση, ενώ όσον αφορά τα κινητά οι συνδέσεις δεν είναι μόνιμες. Άρα αυτό που απαιτείται για τα δίκτυα αυτά ευκρινές μοντέλο καναλιού, μοντέλο κινητής και διαχείριση δυναμικών συνδέσεων.

Οι ασύρματες μεταδόσεις έχουν δύο λειτουργίες τη διάδοση ραδιοκυμάτων όπου υπολογίζεται η λειφθήσα δύναμη του σήματος και την παρεμβολή όπου υπολογίζεται η απώλεια πακέτου για παράδειγμα λόγω θορύβου. Βέβαια υπάρχουν επιπτώσεις στη διάδοση με ραδιοκύματα όπως η απόσβεση λόγω απόστασης, αντανάκλαση λόγω εμποδίων, περίθλαση λόγω εμποδίων με αιχμηρές άκρες, διασκόρπιση από αντικείμενα που είναι μικρά σε σχέση με το μήκος κύματος.

Τα μοντέλα διάδοσης χωρίζονται σε δύο κατηγορίες:

-τα στοιχειώδη που αποτελούνται από: τα μοντέλα ελεύθερου χώρου διάδοσης, two-ray μοντέλα διάδοσης εδάφους και τα σκιώδη.

-τα εμπειρικά μοντέλα δηλαδή τα εξωτερικού χώρου(αντανάκλασης εδάφους) και εσωτερικού χώρου(απώλεια μονοπατιού λόγω εμποδίων).

Τα μοντέλα κινητής καθορίζουν την κίνηση των κόμβων του δικτύου. Έχουν τρία επίπεδα λεπτομέρειας: το μικροσκοπικό, το μακροσκοπικό και αυτό της συνολικής συμπεριφοράς.

Οι ενότητες για προσομοίωση δικτύων παρέχονται από πλαίσια: πλαίσιο κινητής που υποστηρίζουν κόμβους κινητής και ασύρματο μέσο καθώς και το INET πλαίσιο που υποστηρίζει ασύρματα πρωτόκολλα και κινητή.

ΚΕΦΆΛΑΙΟ 8: ΑΝΑΛΥΤΙΚΉ ΕΚΤΊΜΗΣΗ

8.1 Μελέτη πριπτώσεων ALOHA, TCP

Το κομμάτι της αναλυτικής εκτίμησης αποτελεί το τελευταίο στάδιο στο σχεδιασμό πρωτοκόλλου και είναι αυτό που θα μας απασχολήσει στο τελευταίο αυτό κεφάλαιο. Αρχικά μας απασχολεί η διαδικασία της αναλυτικής επικύρωσης. Εδώ θέλουμε να αποδείξουμε την ορθότητα και τους ισχυρισμούς γύρω απ' αυτήν αλλά και να δείξουμε πως το σύστημα είναι απαλλαγμένο από αδιέξοδα. Ύστερα κατά την αναλυτική εκτίμηση της επίδοσης απαιτείται ένα μοντέλο αφαίρεσης και κάποιες μέθοδοι ανάλυσης κατανεμημένων συστημάτων. Μία τέτοια μέθοδος ανάλυσης είναι η θεωρία ουράς η οποία αποτελεί μία περιγραφή συστήματος μέσω διεργασιών.

Η πρώτη περίπτωση που θα μελετήσουμε είναι αυτή του *ALOHA*.

Το ALOHA αποτελεί ένα ασύρματο πακέτο ράδιο δικτύου με τοπολογία αστέρα. Έχει δύο κανάλια μέσω των οποίων τα μηνύματα στέλνονται από τον host στον κομβικό σταθμό(host bound) χρησιμοποιώντας το εισερχόμενο κανάλι. Ο κόμβος κάνει broadcast το μήνυμα σε όλους τους σταθμούς που χρησιμοποιούν το εξερχόμενο κανάλι. Το πρωτόκολλο ALOHA έχει το εξής σκεπτικό: όποτε έχει δεδομένα, τα στέλνει ενώ εάν υπάρχει σύγκρουση προσπαθεί να επαναπροωθήσει το πακέτο.

Για να αναλύσουμε τον αριθμό εργασιών ανά μονάδα χρόνου(throughput) κάνουμε τις παρακάτω παραδοχές: ο αριθμός των σταθμών είναι N, ο χρόνος μεταφοράς πακέτου είναι T, κάθε σταθμός μεταφέρει με πιθανότητα p ανά διάστημα T. Εισροή πακέτων ακολουθεί μία poisson διαδικασία με ρυθμό άφιξης $\lambda = N*P$.

Στην περίπτωση του “χαλασμένου” πρωτοκόλλου ALOHA, οι μεταδόσεις συγχρονίζονται και ξεκινούν σε χρονοθυρίδες μήκους Τ. Έτσι η ευάλωτη περίοδος μειώνεται σε Τ.

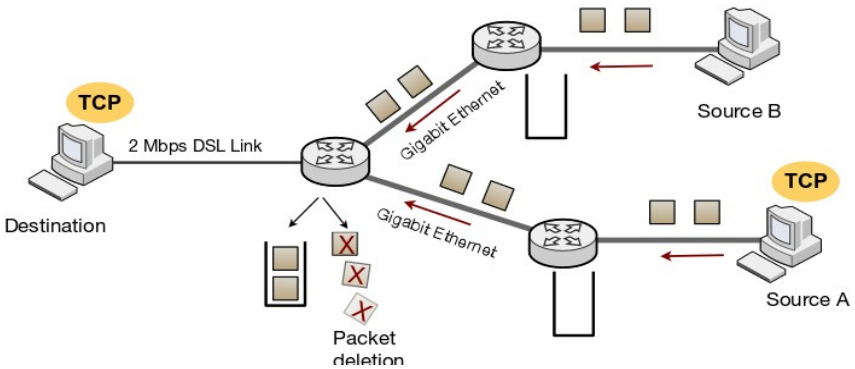
Η κατάσταση του συστήματος είναι ο αριθμός των σταθμών με καθυστερημένα πακέτα. Ενώ η μετατόπιση είναι η αλλαγή των καθυστερημένων σταθμών ανά μονάδα χρόνου και φανερώνει την κατεύθυνση προς την οποία η κατάσταση του συστήματος αλλάζει.

Αν αυξήσουμε την πιθανότητα αναμετάδοσης r, τα καθυστερημένα πακέτα μειώνονται και η ασταθής θέση μπορεί γρήγορα να ξεπεραστεί. Το να μειώσουμε το r αυξάνει την καθυστέρηση. Επίσης υπάρχουν αλγόριθμοι που εξασφαλίζουν τη σταθερότητα. Πρακτικά καλό είναι να κρατήσουμε το ρυθμό άφιξης κάτω από το μέγιστο.

Τώρα θα μιλήσουμε για την περίπτωση της ανάλυσης του ελέγχου συμφόρησης του TCP.

Το TCP παρέχει μία υπηρεσία παράδοσης διαγράμματος απόκρισης τερματικό-σε-τερματικό. Χρησιμοποιεί IP και μοιράζεται την ευρυζωνικότητα με άλλη κίνηση. Σε καταστάσεις κυκλοφοριακής συμφόρησης, οι δρομολογητές ρίζνουν πακέτα. Το TCP αντιδρά προσαρμόζοντας τον ρυθμό εισροής.Κατά την ανάκληση είναι η μόνη διαθέσιμη πληροφορία για να εντωπίσουμε καταστάσεις συμφόρησης είναι οι αποκρίσεις.

23.



Ο μηχανισμός ελέγχου ανάκλησης του TCP χρησιμοποιείται από πολλούς που μοιράζονται την ευρυζωνικότητα. Αν κάποιος χρήστης μειώσει το ρυθμό δεδομένων, η ευρυζωνικότητα θα 'ναι διαθέσιμη για όλους.

Αρχικά χρειαζόμαστε ένα αφαιρετικό μοντέλο του αλγορίθμου για το περιβάλλον. Η αλγοριθμική αρχή πίσω από τον έλεγχο συμφόρησης του TCP είναι: αν γίνεται να αυξήσουμε το ρυθμό

δεδομένων και να μειώσουμε το ρυθμό δεδομένων στο μισό στην περίπτωση απώλειας πακέτου. Κάνουμε τις παρακάτω αφαιρέσεις :

- δεν υπολογίζουμε την αργή αρχική φάση,

- θεωρούμε ένα κυκλικό μοντέλο και μία δυαδική ανατροφοδότηση,

- το κανάλι επικοινωνίας μοιράζεται και μπορεί να χρησιμοποιηθεί για συγκεκριμένη ευρυζωνικότητα.

Το TCP βασικά χρησιμοποιεί τον ακόλουθο μηχανισμό για να θέσει τον ρυθμό δεδομένων χ.

Αρχικοποιεί με χ = 1. Αν φτάσει η απόκριση για ένα κομμάτι, κάνει αλυξηση κατά ένα, χ = χ + 1.

Αν υπάρξει απώλεια πακέτου διαιρεί διά 2, χ = χ / 2.

Η αναλυτική εκτίμηση της απόδοσης βασίζεται σε αφαιρετικά μοντέλα. Προϋποθέτει βαθιά γνώση του συστήματος και μπορεί να πραγματοποιηθεί σε συνδυασμό με τις προσομοιώσεις για να ελέγξουμε αν οι αφαιρέσεις στο μοντέλο είναι έγκυρες. Λόγω της αφαιρετικότητας δεν θα πρέπει να αγνοήσουμε τις παρενέργειες στο σύστημα.

Κεφάλαιο 9: Επίλογος

Στην παρούσα εργασία μας απασχόλησε το ομολογουμένως δύσκολο ζήτημα του σχεδιασμού πρωτοκόλλων και προσπαθήσαμε να δώσουμε μία γενική εικόνα για όλα τα στάδια που απαιτούνται κατά τον σχεδιασμό. Πιο συγκεκριμένα καλύψαμε θέματα που αφορούν το τι πρέπει να λάβουμε υπ όψιν πριν ξεκινήσουμε δηλαδή κάποιες σχεδιαστικές αρχές που καλό είναι να τηρήσουμε και στην συνέχεια μιλήσαμε για την διαδικασία του σχεδιασμού αυτού κάθε αυτού και του προσδιορισμού του πρωτοκόλλου. Το επόμενο βήμα ήταν αυτό της επικύρωσης όπου ελέγχουμε κατά πόσο είναι ορθά όσα διατυπώσαμε μέχρι τώρα και ύστερα μιλήσαμε για τις τεχνικές σχεδιασμού και την εκτέλεση του πρωτοκόλλου. Τέλος τα δύο τελευταία κομμάτια αφορούσαν την προσομοίωση και την αναλυτική εκτίμηση όπου ουσιαστικά επαληθεύαμε όλες τις λειτουργίες τις οποίες θέλαμε να τρέχει το πρωτόκολλο και εξετάζαμε εάν και κατά πόσο υπάρχουν λάθη ή περιπτώσεις που θα έπρεπε να επανεξετάσουμε.

Α Κ Ρ Ω Ν Ύ Μ Ι Α

ABNF(augmented backus naur form) = μετάγλώσσα βασισμένη στην BNF, περιγραφική για την επικοινωνία πρωτοκόλλων εκτός των άλλων

ALOHA = είναι ένα απλό σχήμα επικοινωνίας όπου κάθε πόρος σ ένα δίκτυο στέλνει δεδομένα όποτε υπάρχει κενό πλαίσιο

ASCII(american standard code for information interchange) = κωδικοποιημένο σύνολο χαρακτήρων της λατινικής

ASN.1(abstract syntax notation one) = μία σημειογραφία που περιγράφει κανόνες και δομές κατά την επικοινωνία πρωτοκόλλων

BNF(backus naur form) = μία από τις κύριες σημειολογικές τεχνικές για γραμματικές χωρίς συμφραζόμενα

CBR(constant bit rate) = ένας όρος στις τηλεπικοινωνίες που αναφέρεται στην ποιότητα της υπηρεσίας

CSN.1(concrete syntax notation one) = μία σημειογραφία που περιγράφει κανόνες και δομές κατά την επικοινωνία πρωτοκόλλων

CFSM(communicating finite state machine) = αποτελεί ένα πεπερασμένο αυτόματο μηχανής που έχει επίσης τις έιτοθργίες στείλε και λάβε

EFSM(extended finite state machine) = fsm όπου η μετάβαση μπορεί να εκφραστεί από μία συνθήκη if

FIFO(first in first out) = μέθοδος οργάνωσης δεδομένων σε ουρά

FSM(finite state machine) = μαθηματικό μοντέλο υπολογισμού που χρησιμοποείται για προγράμματα υπολογιστών και κυκλώματα λογικών ακολουθιών

GSM(global system for mobile communications) = πρότυπο που περιγράφει πρωτόκολλα 2G για κινητά

IETF(internet engineering task force) = αναπτύσσει και προωθεί πρότυπα διαδικτύου και ειδικά αυτά που ταιριάζουν με το TCP/IP

ISDN(integrated service digital network) = σύνολο προτύπων επικοικωνίας για την μετάδοση δεδομένων μέσω του δημόσιου τηλεφωνικού δικτύου

INET = μία προσομοίωση επικοινωνίας δικτύων για το OMNet

LSC(life sequence charts) = διαγράμματα ακολουθίας μηνυμάτων

LTL(linear temporal logic) = μοντέλο αναπαράστασης χρόνου

MSC(message sequence charts) = γραφική σημειογραφία για περιγραφή επικοινωνιακών σεναρίων σε ασύγχρονες διεργασίες

NAT(network address translation) = μέθοδος τροποποίησης της πληροφορίας των διευθύνσεων σε header πακέτα IP διαγραμμάτων

OMNeT = περιβάλλον προσομοίωσης διακριτών γεγονότων

OSI (open systems interconnections)= το πρώτο στρώμα ή φυσικό στρώμα ενός δικτύου

PROMELA(process or protocol meta language) = γλώσσα προτύπων επικύρωσης

SPIN = ένας τύπος διαγράμματος για αναπαράσταση καταστάσεων

SDL(specification and description language) = γλώσσα για τον προσδιορισμό και την περιγραφή της συμπεριφοράς αλληλεπιδραστικών και κατανεμημένων συστημάτων

TCP/IP(Transmission Control Protocol/Internet Protocol)=Πρωτόκολλο Ελέγχου Μετάδοσης και πρωτόκολλο του Internet)

TLV(type length value) = στα επικοινωνιακά πρωτόκολλα δεδομένων κάποιες πληροφορίες κωδικοποιούνται ως type-length-value

UML(unified modeling language) = πρότυπη γλώσσα μοντελοποίησης για τη γραφική απεικόνιση, προσδιορισμό, κατασκευή και τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού

Β Ι Β Λ Ι Ο Γ Ρ Α Φ Ί Α

Βιβλία: *Δίκτυα υπολογιστών, Andrew S. Tanenbaum*

Δημοσιεύσεις:

http://ru6.cti.gr/ru6/system/files/bouras_site/ergasies_foithtwn/206_KASTAN__HS_DHHTRIOS_3871_KABOURGIAS_GEORGIOS_3659.pdf?language=el

URLs:

http://archive.cone.informatik.uni-freiburg.de/teaching/lecture/protocol__design-s09/slides/

http://user.informatik.uni-goettingen.de/~fu/design_principles.htm

http://www.drdobbs.com/architecture-and-design/designing-a-network- protocol/240157979? pgno=1