3.3 Εφαρμογή Ουράς: Προσομοίωση Κέντρου Πληροφόρησης

Όπως αναφέρθηκε και στην αρχή, οι ουρές μπορούν να χρησιμοποιηθούν ως πρότυπα για γραμμές αναμονής. Συνήθως οι γραμμές αναμονής είναι δυναμικές, δηλαδή το μήκος τους μεταβάλλεται με την πάροδο του χρόνου: μεγαλώνουν καθώς νέα στοιχεία φτάνουν και προστίθενται στις ουρές και μικραίνουν καθώς στοιχεία αφαιρούνται από αυτές και εξυπηρετούνται. Ο όρος **προσομοίωση** (**simulation**) αναφέρεται στην προτυποποίηση μιας τέτοιας δυναμικής διαδικασίας και στη χρησιμοποίηση αυτού του προτύπου για μελέτη της συμπεριφοράς ή της διαδικασίας. Η συμπεριφορά κάποιων **ντετερμινιστικών** διαδικασιών μπορεί να προτυποποιηθεί με μια εξίσωση ή ένα σύνολο εξισώσεων. Για τις ουρές, ωστόσο, δεν είναι γνωστό εκ των προτέρων πότε ακριβώς θα έχουμε νέα άφιξη ή πόσος ακριβώς χρόνος θα χρειαστεί για την εξυπηρέτηση ενός στοιχείου. Κάτι τέτοιο σημαίνει ότι η συμπεριφορά μιας ουράς συνήθως συνεπάγεται τυχαία μεταβλητότητα και ένα πρόγραμμα που προσομοιώνει μια τέτοια συμπεριφορά θα πρέπει να εμπεριέχει τυχαία μεταβλητότητα.

Ως παράδειγμα εφαρμογής προσομοίωσης ουράς παίρνουμε τη λειτουργία ενός κέντρου πληροφόρησης/κράτησης που εξυπηρετεί τηλεφωνήματα πελατών σε ένα νούμερο χωρίς χρέωση, όπως για παράδειγμα το τηλεφωνικό νούμερο μιας αεροπορικής εταιρίας ή μιας εταιρίας ενοικίασης αυτοκινήτων. Όταν φτάνει ένα τηλεφώνημα σ' ένα τέτοιο κέντρο και ο τηλεφωνητής είναι διαθέσιμος, τότε εξυπηρετείται άμεσα. Αν, όμως, ο τηλεφωνητής είναι κατειλημμένος, τότε το τηλεφώνημα τίθεται σε ουρά αναμονής για να εξυπηρετηθεί αργότερα. Θα γράψουμε, λοιπόν, ένα πρόγραμμα που προσομοιώνει τη λειτουργία ενός τέτοιου κέντρου πληροφόρησης και υπολογίζει διάφορα στατιστικά μεγέθη για να μετρήσει την απόδοσή του.

Ανάλυση Προβλήματος και Εξειδίκευση

Η προσομοίωση θα πρέπει να γίνει ως εξής: Ένα προσομοιωμένο ρολόι αρχικοποιείται στο 0 και αυξάνεται κατά 1 (λεπτό) μέχρι να φτάσει κάποιο δοσμένο χρονικό όριο (σε λεπτά). Σε κάθε *χτύπο* του ρολογιού, γίνεται ένας έλεγχος για να καθοριστεί αν η εξυπηρέτηση του τρέχοντος τηλεφωνήματος έχει ολοκληρωθεί και, αν ναι, τότε ένα άλλο τηλεφώνημα λαμβάνεται από την ουρά τηλεφωνημάτων που περιμένουν για εξυπηρέτηση (αν υπάρχουν) και η προσομοιωμένη εξυπηρέτησή του ξεκινά. Επίσης, γίνεται έλεγχος για το αν έχει φτάσει νέο τηλεφώνημα. Στην περίπτωση που έχει φτάσει, καταγράφεται ο χρόνος άφιξής του, καθορίζεται και καταγράφεται ο χρόνος εξυπηρέτησής του και τοποθετείται στην ουρά αναμονής για να υποστεί επεξεργασία κατά τρόπο "*πρώτος-έρχεται - πρώτος-εξυπηρετείται*" (*first-come-first-served*), όταν ο τηλεφωνητής γίνει διαθέσιμος. Όταν φτάσουμε το

προκαθορισμένο χρονικό όριο, δεν γίνονται δεκτά άλλα τηλεφωνήματα, όμως, η εξυπηρέτηση συνεχίζεται μέχρι να ολοκληρωθεί η επεξεργασία όλων των τηλεφωνημάτων που βρίσκονται στην ουρά αναμονής.

Όταν ολοκληρωθεί η προσομοίωση, πρέπει να γίνει αναφορά κάποιων στατιστικών, όπως για παράδειγμα ο αριθμός των τηλεφωνημάτων που εξυπηρετήθηκαν και ο μέσος χρόνος αναμονής για κάθε τηλεφώνημα, που μετρούν την απόδοση του κέντρου πληροφόρησης. Επίσης, σε ώρες αιχμής κάποια εισερχόμενα τηλεφωνήματα μπορεί να απορριφθούν, επειδή δεν θα χωρούν στην ουρά, και γι' αυτό θα ήταν χρήσιμο να γνωρίζουμε το πλήθος αυτών των τηλεφωνημάτων. Στην παρούσα προσομοίωση θα υπολογιστούν τα παρακάτω στατιστικά μεγέθη:

- Αριθμός τηλεφωνημάτων που επεξεργάζονται
- Μέσος χρόνος αναμονής ανά τηλεφώνημα
- Αριθμός τηλεφωνημάτων που απορρίπτονται

Επειδή η προσομοίωση θα διαρκέσει για συγκεκριμένο χρονικό διάστημα, είναι προφανές ότι ως στοιχείο εισόδου θα πρέπει να υπάρχει κάποιο *χρονικό όριο*. Επιπλέον, επειδή είναι απαραίτητο να διεγείρουμε την άφιξη και την εξυπηρέτηση των τηλεφωνημάτων, χρειαζόμαστε πληροφορίες για τους *ρυθμούς άφιξης* και τους *χρόνους εξυπηρέτησης*. Επομένως, τα δεδομένα εισόδου θα είναι:

- Ρυθμός άφιξης των τηλεφωνημάτων
- Κατανομή των χρόνων εξυπηρέτησης
- Χρονικό όριο

Δομές Δεδομένων

Τα τρία στοιχεία εισόδου, *ρυθμός άφιξης*, *χρόνοι εξυπηρέτησης* και *χρονικό όριο*, είναι οι βασικές παράμετροι της προσομοίωσης και γι' αυτό είναι βολικό να κατασκευαστεί μια εγγραφή για αυτά:

```
typedef    struct{
             double ArrivalRate;
             double ServicePerc[NumLimits];
             int TimeLimit;
           } SimParam;
```

ArrivalRate είναι η πιθανότητα ένα τηλεφώνημα να φτάσει σε ένα δοσμένο λεπτό. Για παράδειγμα, αν ο μέσος χρόνος μεταξύ τηλεφωνημάτων είναι πέντε λεπτά, τότε ο

ρυθμός άφιξης είναι 1/5 ή 0.2 τηλεφωνήματα ανά λεπτό.

☛Στον πίνακα ServicePerc καταγράφονται πληροφορίες που αφορούν τον χρόνο εξυπηρέτησης, δηλαδή:

ServicePerc[1]	=ποσοστό (%) τηλεφωνημάτων που εξυπηρετούνται σε 1 λεπτό ή λιγότερο
ServicePerc[2]	=ποσοστό (%) τηλεφωνημάτων που εξυπηρετούνται σε 2 λεπτά ή λιγότερο
ServicePerc[3]	=ποσοστό (%) τηλεφωνημάτων που εξυπηρετούνται σε 3 λεπτά ή λιγότερο

κ.ο.κ.

☛TimeLimit είναι το προκαθορισμένο χρονικό όριο για την προσομοίωση.

Το βασικό αντικείμενο αυτής της προσομοίωσης είναι ένα τηλεφώνημα στο κέντρο πληροφόρησης. Ένα τηλεφώνημα χαρακτηρίζεται από την *χρονική στιγμή άφιξής* του, από τον *χρόνο* που απαιτείται *για να εξυπηρετηθεί* και από την *χρονική στιγμή ολοκλήρωσης της εξυπηρέτησής* του. Γι' αυτό χρησιμοποιούμε πάλι μια εγγραφή για τα προσομοιωμένα τηλεφωνήματα, όπως φαίνεται παρακάτω:

```
typedef struct {
        int TimeOfArrival,
        ServiceTime,
        TimeOfCompletion;
      CallRecord;
}
```

Η τελευταία δομή δεδομένων που χρειαζόμαστε είναι μια ουρά αναμονής, OnHold, με εγγραφές τηλεφωνημάτων για τα τηλεφωνήματα που πρέπει να τεθούν σε αναμονή, όταν ο τηλεφωνητής είναι κατειλημμένος. Έχουμε, δηλαδή, τον τύπο:

QueueElementType CallRecord;

Πλάνο Σχεδίασης

Τρεις είναι οι κύριες εργασίες που πρέπει να εκτελεί το πρόγραμμα. Επομένως, σε πρώτο επίπεδο, θα αποτελείται από 3 modules:

Modules Πρώτου Επιπέδου

- **GetSimParameters**:

  Λειτουργία: Παίρνει τις παραμέτρους της προσομοίωσης.

  Επιστρέφει: Μια εγγραφή τύπου SimParam.

- **Simulate**:

  Δέχεται: Μια εγγραφή τύπου SimParam.

  Λειτουργία: Εκτελεί την προσομοίωση.

  Επιστρέφει: Συνολικό αριθμό τηλεφωνημάτων, συνολικό χρόνο αναμονής και πλήθος απορριφθέντων τηλεφωνημάτων.

- **PrintReport**:

  Δέχεται: Συνολικό αριθμό τηλεφωνημάτων, συνολικό χρόνο αναμονής και πλήθος απορριφθέντων τηλεφωνημάτων.

  Λειτουργία: Υπολογίζει και αναφέρει τα στατιστικά της προσομοίωσης.

  Επιστρέφει: Πλήθος τηλεφωνημάτων, μέσο χρόνο αναμονής και πλήθος απορριφθέντων τηλεφωνημάτων.

Το module GetSimParameters θα ζητήσει από το χρήστη να εισάγει το *ρυθμό άφιξης*, μια κατανομή για τους *χρόνους εξυπηρέτησης* και ένα *χρονικό όριο* για την προσομοίωση και όλα αυτά θα αποθηκευτούν σε μια εγγραφή τύπου SimParam. Για τους χρόνους εξυπηρέτησης, ο χρήστης θα εισάγει το ποσοστό τηλεφωνημάτων που μπορούν να εξυπηρετηθούν σε 1 λεπτό ή λιγότερο, το ποσοστό τηλεφωνημάτων που απαιτούν πάνω από 1 λεπτό αλλά λιγότερο από 2, και ομοίως για τα υπόλοιπα ποσοστά. Αυτά τα ποσοστά αποθηκεύονται στον πίνακα ServicePerc. Ο πίνακας αυτός θα χρησιμοποιηθεί για τον καθορισμό του χρόνου εξυπηρέτησης κάθε τηλεφωνήματος.

Το module PrintReport δεν χρειάζεται περαιτέρω ανάλυση, επομένως έμεινε το Simulate, που περιγράφεται στον παρακάτω αλγόριθμο:

Αλγόριθμος για το module Simulate

1. Αρχικοποίησε το ρολόι, την ουρά αναμονής και τις άλλες μεταβλητές της προσομοίωσης.
2. **Όσο** το ρολόι δεν έχει φτάσει στο καθορισμένο χρονικό όριο **επανάλαβε**
   α. Αύξησε το ρολόι κατά μία χρονική μονάδα.
   β. Συνέχισε την εξυπηρέτηση του τρέχοντος τηλεφωνήματος ή ξεκίνησε την εξυπηρέτηση ενός από τα τηλεφωνήματα της ουράς αναμονής(αν υπάρχουν).

γ. Έλεγξε για άφιξη νέου τηλεφωνήματος.

Τέλος_ επανάληψης

3. **Όσο** τα τηλεφωνήματα παραμένουν στην ουρά αναμονής **επανάλαβε**

α. Αύξησε το ρολόι.

β. Αφαίρεσε το τηλεφώνημα που βρίσκεται στην εμπρός άκρη της ουράς και εξυπηρέτησέ το.

Τέλος_ επανάληψης

Όπως βλέπουμε, υπάρχουν τρεις κύριες υπο-εργασίες που πρέπει να εκτελέσει το module Simulate και γι' αυτό θα πρέπει να κατασκευαστούν τρία modules δευτέρου επιπέδου:

Modules Δευτέρου Επιπέδου

- **Initialize**:

  Λειτουργία: Αρχικοποιεί τις μεταβλητές της προσομοίωσης και μια γεννήτρια τυχαίων αριθμών.

  Επιστρέφει: Μια κενή ουρά *OnHold*, την τιμή 0 για τις *NumCalls*, *WaitingTime*, *Rejected* και *Clock* και την τιμή FALSE για την boolean μεταβλητή *InService*, που δείχνει ότι κανένα τηλεφώνημα δεν εξυπηρετείται την τρέχουσα στιγμή.

- **Service**:

  Δέχεται: Ένα τηλεφώνημα *Call*, την *Clock*, την *WaitingTime* και την *InService*.

  Λειτουργία: Συνεχίζει την εξυπηρέτηση του τρέχοντος τηλεφωνήματος *Call* ή ξεκινά την εξυπηρέτηση ενός τηλεφωνήματος από την ουρά *OnHold*.

  Επιστρέφει: Ενημερωμένες τις τιμές των *Call*, *Clock*, *WaitingTime* και *InService*.

- **CheckForNewCall**:

  Δέχεται: Τις *Clock*, *Rejected*, την εγγραφή *SimParams* και την ουρά *OnHold*.

  Λειτουργία: Ελέγχει αν έχει φτάσει νέο τηλεφώνημα και αν ναι, το εισάγει στην ουρά *OnHold*, αν είναι δυνατό. Διαφορετικά, ενημερώνει την τιμή της *Rejected*.

  Επιστρέφει: Ενημερωμένες τις τιμές των *Rejected*, *NumCalls* και *OnHold*.

Το module Initialize δεν χρειάζεται ανάλυση, όμως θα αναλύσουμε τα άλλα δύο. Ένας αλγόριθμος για το Service περιγράφεται παρακάτω :

/*Αλγόριθμος για το module Service */

1. **Αν** *InService* = **TRUE** /*ένα τηλεφώνημα εξυπηρετείται αυτή τη στιγμή*/ **τότε**

   Αν *Clock* = *Call*.*TimeOfCompletion* **τότε**

   InService ← FALSE /*Η εξυπηρέτηση του τρέχοντος τηλεφωνήματος ολοκληρώθηκε*/

   Τέλος_ αν

   Τέλος_ αν

2. **Αν** *InService* = **FALSE** /*κανένα τηλεφώνημα δεν εξυπηρετείται αυτή τη στιγμή*/ **τότε**

   Αν η *OnHold* δεν είναι κενή **τότε**

   α. Διάγραψε ένα τηλεφώνημα από την *OnHold*

   β. *Call*.*TimeOfCompletion* ← *Clock* + *Call*.*ServiceTime*

   γ. Ενημέρωσε την *WaitingTime* προσθέτοντας *Clock -Call*.*TimeOfArrival* /*ο χρόνος που το τηλεφώνημα Call ήταν στην OnHold*/

   δ. *InService* ← **TRUE** /*τοποθέτησε το Call σε εξυπηρέτηση*/

   Τέλος_ αν

   Τέλος_ αν

Το module CheckForNewCall ελέγχει αν έχει αφιχθεί νέο τηλεφώνημα και, αν ναι, προσπαθεί να το εισάγει στην ουρά OnHold. Για να καθοριστεί η άφιξη ενός τηλεφωνήματος στη διάρκεια ενός συγκεκριμένου λεπτού, δημιουργούμε έναν τυχαίο αριθμό μεταξύ 0 και 1 και, αν αυτός ο αριθμός είναι μικρότερος από SimParams.ArrivalRate, θεωρούμε ότι έχει αφιχθεί τηλεφώνημα κατά τη διάρκεια αυτού του λεπτού. Οι αριθμοί που παράγονται από μια γεννήτρια τυχαίων αριθμών υποθέτουμε ότι κατανέμονται ομοιόμορφα στο διάστημα μεταξύ 0 και 1. Συνεπώς, αν παραχθούν πολλοί τυχαίοι αριθμοί, περιμένουμε ότι περίπου το 20% από αυτούς θα βρίσκεται στο διάστημα (0,0.2). Αυτό σημαίνει ότι η πιθανότητα για ένα τηλεφώνημα να φτάσει σε ένα δεδομένο λεπτό θα πρέπει να είναι περίπου0.2.

Όταν φτάνει ένα νέο τηλεφώνημα, πρέπει να καταγραφεί ο χρόνος άφιξής του καθώς και ο χρόνος που απαιτείται για να εξυπηρετηθεί. Ο χρόνος εξυπηρέτησης καθορίζεται από τον πίνακα ServicePerc. Έστω, για παράδειγμα, ότι ο πίνακας ServicePerc περιέχει τις τιμές 0.30, 0.55, 0.60, 0.70 και 0.95. Αν δημιουργήσουμε έναν τυχαίο αριθμό στο διάστημα (0,1), τότε

το υπο-διάστημα(0,0.30], (0.30,0.55], (0.55,0.60],(0.60,0.70], (0.70,0.95] στο οποίο βρίσκεται ο αριθμός αυτός καθορίζει το χρόνο εξυπηρέτησης του τηλεφωνήματος.

Για το module CheckForNewCall μπορεί να γραφεί ο παρακάτω αλγόριθμος:

Αλγόριθμος για το module CheckForNewCall

1. Δημιούργησε έναν τυχαίο αριθμό *N*
2. **Αν** *N* < *SimParams*.*ArrivalRate* /*έφτασε νέο τηλεφώνημα*/ **τότε**
   - α. *NewCall*.*TimeOfArrival* ← *Clock*
   - β. *NumCalls* ← 1
     /*Καθόρισε το χρόνο εξυπηρέτησής του*/
   - γ. Δημιούργησε έναν νέο τυχαίο αριθμό *N*
   - δ. *i* ← 1
   - ε. **Όσο** *N*>*SimParams*.*ServicePerc*[*i*] **επανάλαβε**
     i ← *i* + 1
     Τέλος_ επανάληψης
   - ζ. *NewCall*.*ServiceTime* ← *i*
   - η. Πρόσθεσε το τηλεφώνημα *Call* στην ουρά *OnHold*
   - θ. **Αν** *(FullQ(Queue))* /*η ουρά είναι γεμάτη*/ **τότε**
     Rejected ← *Rejected* + 1
     Τέλος_αν

   Τέλος_αν

Το πρόγραμμα InfoCent.c υλοποιεί όλους τους παραπάνω αλγορίθμους για το πρόβλημα της προσομοίωσης ενός κέντρου πληροφόρησης. Για τον ΑΤΔ Ουρά χρησιμοποιείται μια παραλλαγή του QueueADT.c, το QCallADT.c. Η μόνη διαφορά είναι ότι τα στοιχεία της ουράς είναι τύπου CallRecord.