Εισαγωγή

1.1 Επισκόπηση

Με τον όρο Personal Communication Services (PCS) συνηθίζουμε να εννοούμε ότι η πρόσβαση σε ένα κοινόχρηστο δίκτυο Επικοινωνίας είναι εφικτή από «οποιοδήποτε σημείο» επιθυμεί ο χρήστης του δικτύου. Λέγοντας «οποιοδήποτε σημείο», εννοούμε συγκεκριμένες γεωγραφικές ζώνες κυψελοειδούς μορφής [1], για τις οποίες έχει γίνει η πρόβλεψη και η εγκατάσταση του απαραίτητου εξοπλισμού που θα επιτρέπει την παραπάνω πρόσβαση. Βασικά κριτήρια για την επιλογή αυτών των ζωνών, είναι η πληθυσμιακή τους πυκνότητα και η εμπορική - οικονομική τους σημασία. Μέσα σε αυτές τις συγκεκριμένες ζώνες είναι εφικτή η ασύρματη πρόσβαση στο Επικοινωνιακό Δίκτυο, αρκεί ο πελάτης - χρήστης του Δικτύου να διαθέτει τον ανάλογο εξοπλισμό. Ο εξοπλισμός αυτός είναι αφ' ενός υπεύθυνος για την επίτευξη της σύνδεσης ανάμεσα στον χρήστη και στο δίκτυο και την κωδικοποίηση και μετάδοση της πληροφορίας από και προς τον χρήστη και αφ' ετέρου πρέπει να είναι εύχρηστος, ελαφρύς και μικρός σε όγκο ώστε να επιτρέπει την εύκολη μεταφορά και χρησιμοποίησή του ακόμα και εν κινήσει [2] [3].

Ποια ακριβώς όμως είναι τα είδη της πληροφορίας τα οποία ο χρήστης παράγει και το δίκτυο καλείται να εξυπηρετήσει; Η φωνή είναι σίγουρα η σπουδαιότερη κατηγορία υπηρεσίας που ένα τέτοιο σύστημα οφείλει να υποστηρίξει. Επιπλέον, δεδομένα που προέρχονται από μετακίνηση αρχείων Η/Υ (ftp), εφαρμογές πολυμέσων ή διάλογος κειμένου (text chatting) είναι επίσης πολύ πιθανό να παρουσιαστούν. Εκείνο όμως, που προβάλλει ως πραγματική πρόκληση, είναι η μετάδοση πάνω από ένα τέτοιο ψηφιακό επικοινωνιακό δίκτυο δεδομένων που προέρχονται από ψηφιακό βίντεο. Η μέχρι τώρα φυσική υλοποίηση των ασύρματων δικτύων και των πρωτοκόλλων, που αυτά χρησιμοποιούν, είναι απαγορευτική για την μετάδοση δεδομένων ψηφιακού βίντεο.

Τόσο τα ασύρματα δίκτύα πρώτης όσο και αυτά δεύτερης και τρίτης γενιάς βασίζονται στην ιδέα της κυψελοειδούς δομής [1]. Με τον όρο «κυψελοειδής δομή» εννοούμε την κάλυψη μιας εκτεταμένης γεωγραφικής περιοχής από περισσότερους από έναν σταθμούς βάσης, διεσπαρμένους σε κατάλληλες θέσεις έτσι ώστε να βελτιστοποιείται η κάλυψη της δεδομένης περιοχής. Μέσα σε μια κυψέλη της παραπάνω δομής τα κινούμενα τερματικά μοιράζονται ένα ράδιο-κανάλι μέσω του οποίου συνδέονται με τον σταθμό βάσης. Μέσα σε κάθε κυψέλη ο αντίστοιχος σταθμός είναι απολύτως υπεύθυνος για την δέσμευση των πόρων του δικτύου, την μετάδοση και συλλογή πληροφοριών από και προς τους κινούμενους χρήστες και για την διασύνδεση της κυψέλης με τις γειτονικές κυψέλες και κατ' επέκταση με το υπόλοιπο δίκτυο σταθερής ή κινητής επικοινωνίας. Μια σύντομη ανασκόπηση και περιγραφή των παρελθόντων, υπαρχόντων αλλά και μελλοντικών ασύρματων δικτύων ακολουθεί παρακάτω.

α. Ασύρματα Δίκτυα Πρώτης Γενιάς

Τα ασύρματα δίκτυα πρώτης γενιάς είχαν σχεδιαστεί για χρήστες υψηλής κινητικότητας αλλά και ισχύος εκπομπής. Η χρήση τους απευθυνόταν κυρίως σε χρήστες ειδικών απαιτήσεων, π.χ. κινούμενους μέσα σε οχήματα, ενώ ο σχετικά μεγάλος όγκος των τερματικών συσκευών δυσχέραινε την χρήση τους. Για την λειτουργία τους χρησιμοποιούσαν Διαμόρφωση Συχνότητας (FM) και Πολυπλεξία Διαίρεσης Συχνότητας (FDM). Εγκαταστάσεις τέτοιας τεχνολογίας υπάρχουν ακόμη και σήμερα σε λειτουργία στην Β. Αμερική (από το 1983 με το Advanced Mobile Phone Service, AMPS) ενώ εξελιγμένες παραλλαγές τους στηριζόμενες σε TDMA (Digital AMPS, IS-136) προβάλλουν σήμερα ως οικονομικές λύσεις αντί της τοποθέτησης ενσύρματων δικτύων επικοινωνιών σε διάφορες περιπτώσεις κυρίως απομακρυσμένων και αραιοκατοικημένων περιοχών [4].

β. Ασύρματα Δίκτυα Δεύτερης Γενιάς

Εδώ εντάσσονται τα περισσότερα σημερινά εν λειτουργία ασύρματα δίκτυα. Η ισχύς εκπομπής των τερματικών συσκευών είναι κάπως μειωμένη. Τεχνικές Time Division Multiple Access (TDMA) και Code Division Multiple Access (CDMA) χρησιμοποιούνται προκειμένου να γίνει εφικτή η επαναχρησιμοποίηση συχνοτήτων με σκοπό την αύξηση της χωρητικότητας του καναλιού. Παραδείγματα τέτοιων συστημάτων είναι το GSM με συχνότητα λειτουργίας στα 900 MHz, το DCS 1800 που ουσιαστικά είναι GSM με συχνότητα λειτουργίας στα 1800 MHz καθώς και το αμερικανικό IS-95 (CDMA) [5]-[8].

γ. Ασύρματα Δίκτυα Τρίτης Γενιάς

Στα Δίκτυα Τρίτης Γενιάς ο αντικειμενικός στόχος είναι να εξυπηρετηθούν όλων των ειδών οι απαιτήσεις (μετάδοση φωνής, video και δεδομένων), χωρίς ο χρήστης να αντιλαμβάνεται τη διαφορά. Το δίκτυο δηλαδή, καλείται να *ολοκληρώσει* τις υπηρεσίες που προσφέρει. Διάφορα πρότυπα έχουν αναπτυχθεί και αναπτύσσονται τόσο σε ευρωπαϊκό όσο και σε παγκόσμιο επίπεδο. Ενδεικτικά αναφέρονται τα Universal Mobile Telecommunications Standards (UMTS) και το Future Public Land Mobile Telecommunications (FPLMTS) από το ευρωπαϊκό ETSI και την ITU, αντίστοιχα [9].

Οι ιδιαίτερες ανάγκες που θα υπάρξουν στα Ασύρματα Δίκτυα Τρίτης Γενιάς για παροχή πολύ μεγάλου εύρους ζώνης στις εφαρμογές και δεδομένων των δυσκολιών που παρουσιάζονται σε όλα τα ασύρματα δίκτυα εξ' αιτίας του μέσου μετάδοσης της πληροφορίας, δηλαδή του αέρα, οδηγούν αυτόματα σε συγκεκριμένες επιλογές για την υλοποίηση των δικτύων αυτών [10]

Η πρώτη επιλογή είναι ότι πρέπει να υπάρξει διαφορετική αντιμετώπιση των χρηστών ανάλογα με τον τρόπο κίνησης τους (ή μη) καθώς και με το είδος της πληροφορίας που εκείνη την συγκεκριμένη στιγμή επιθυμούν να μεταδώσουν στο δίκτυο. Έτσι, το δίκτυο για τους μεν χρήστες χαμηλής ή και μηδενικής κινητικότητας θα παρέχει υψηλές ταχύτητες πρόσβασης και

υπηρεσίες, όπως εξυπηρέτησης multimedia κίνησης ή video conferencing, ενώ για χρήστες υψηλής κινητικότητας θα παρέχεται η κλασική υπηρεσία φωνής καθώς και η μετάδοση δεδομένων σε σχετικά χαμηλές ταχύτητες (αρκετών δεκάδων kbps), αρκετά υψηλότερες ωστόσο, από αυτές που παρέχουν τα σημερινά ασύρματα δίκτυα (μερικά kbps στην καλύτερη των περιπτώσεων).

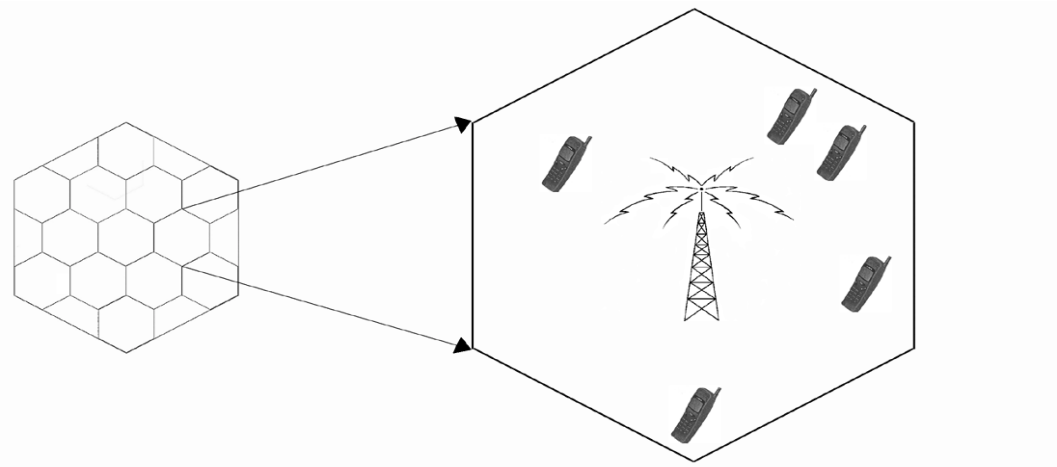
Η δεύτερη επιλογή, που απορρέει από την παραπάνω ξεχωριστή - ιεραρχική αντιμετώπιση των χρηστών (σταθεροί, χαμηλής κινητικότητας, υψηλής κινητικότητας), αντιστοιχεί και σε δύο ή τρεις διαφορετικές φυσικές υλοποιήσεις του δικτύου. Έτσι, οι σταθεροί και χαμηλής κινητικότητας χρήστες εξυπηρετούνται από κυψέλες μικρής ακτίνας (μικροκυψέλες, το πολύ 100 μέτρων) οι οποίες δημιουργούνται από την πολύ πυκνή τοποθέτηση των σταθμών βάσης (Σ.Β.) στην περιοχή κάλυψης. Με αυτόν τον τρόπο επιτυγχάνεται πιο αξιόπιστη μεταφορά της πληροφορίας από το δίκτυο, μείωση έως εξάλειψη των λαθών και μειωμένοι χρόνοι καθυστέρησης μετάδοσης (propagation) του σήματος. Αντίστοιχα πυκνό πρέπει να είναι το δίκτυο διασύνδεσης των Σ.Β.. Το δίκτυο αυτό, λόγω του ότι καλείται να διοχετεύσει ολόκληρο τον όγκο πληροφοριών που δέχεται από τους Σ.Β. στο ενσύρματο τηλεφωνικό δίκτυο ή σε άλλο δίκτυο δεδομένων (π.χ. Internet), πρέπει να είναι και ανάλογα ισχυρό. Διάφορες προτάσεις, που έχουν γίνει, βασίζονται σε τεχνολογίες DQDB, FDDI και ATM. Το αυξημένο κόστος αυτών των εγκαταστάσεων είναι μια πολύ σημαντική παράμετρος που κάνει απαραίτητη την επιλεκτική τοποθέτηση των Σ.Β. σε πολύ μικρής έκτασης περιοχές, όπου τα πληθυσμιακά και οικονομικά κριτήρια το επιτρέπουν.

Οι χρήστες που έχουν έντονη κινητικότητα θα εξυπηρετούνται από κυψέλες μεγαλύτερης ακτίνας (μακροκυψέλες ακτίνας περίπου 1 χιλιομέτρου). Οι Σ.Β. σε αυτές τις κυψέλες θα διασυνδέονται σε σχήμα αστέρα ανά ομάδες με κεντρικούς σταθμούς διασύνδεσης τους τόσο μεταξύ τους όσο και με το υπόλοιπο τηλεφωνικό δίκτυο.

Και στις δύο παραπάνω περιπτώσεις σε περίπτωση αλλαγής κυψέλης λόγω κινητικότητας του χρήστη, οι αντίστοιχοι Σ.Β. αντιλαμβάνονται την

αλλαγή και αναλαμβάνουν να μετάγουν την ζεύξη με τον κινούμενο χρήστη από τον ένα Σ.Β. στον άλλον.

Σχ. 1.1



Πολύ σημαντικό θέμα επίσης, είναι ο μηχανισμός βάσει του οποίου διαμοιράζονται οι διαθέσιμες συχνότητες καναλιού στους χρήστες. Εδώ υπάρχουν δυνατότητες για FDMA, TDMA ή CDMA τεχνικές [11].

1.2 Πρόβλημα

Στην εργασία αυτή σχεδιάζονται και υλοποιούνται τεχνικές για την αποτελεσματικότερη πολυπλεξία πηγών φωνής και πηγών video στο ράδιο κανάλι από τα κινητά τερματικά προς τον Σ.Β. μιας κυψέλης. Ο σκοπός αυτών των τεχνικών είναι να ελαχιστοποιηθεί το κόστος σε πόρους ενός τέτοιου συστήματος, χρησιμοποιώντας στο μέγιστο βαθμό την διαθέσιμη χωρητικότητα του καναλιού ενώ ταυτόχρονα διατηρούνται σε κατάλληλα επίπεδα ένας αριθμός από μετρικές που σχετίζονται με τις υπηρεσίες όπως η πιθανότητα απώλειας πακέτου από τις πηγές. Το σύνολο των παραπάνω μετρικών αποτελούν αυτό που κοινώς έχει επικρατήσει να ονομάζεται «Ποιότητα Υπηρεσίας» (Quality of Service), και συνήθως επιβάλλεται στο σύστημά μας από τις εφαρμογές. Η διαφορετική φύση των ροών πληροφορίας που θα κληθεί το σύστημά μας να ενοποιήσει, οι συχνά αντικρουόμενες απαιτήσεις τους, καθώς και η προσυμφωνημένη «Ποιότητα Υπηρεσίας» που πρέπει να τους παρασχεθεί, καθιστά ιδιαίτερα δύσκολη την σχεδίαση αποτελεσματικών τεχνικών πολυπλεξίας.

Η εργασία ακολουθεί στα επόμενα κεφάλαιά την εξής διάρθρωση: στο κεφάλαιο 2 παρουσιάζεται η δομή και τα φυσικά χαρακτηριστικά του δικτύου, αναλύονται οι απαραίτητες συμβάσεις που έχουν γίνει, γίνεται μία σύντομη περιγραφή των τερματικών συσκευών και των μοντέλων, που προσομοιώνουν τις εξόδους τους και αναλύεται η λογική που έχει ακολουθηθεί προκειμένου να δομηθούν οι αλγόριθμοι χρονοδρομολόγησης που τελικά εφαρμόζονται. Στο κεφάλαιο 3 παρουσιάζονται τα αποτελέσματα των προσομοιώσεων και γίνονται συγκρίσεις και σχολιασμός. Στο κεφάλαιο 4 καταλήγουμε στα συμπεράσματα μας και παραθέτουμε τον κώδικα της προσομοίωσης.

Κεφάλαιο 2

Τερματικές Συσκευές, Δίκτυο

Στο κεφάλαιο αυτό παρουσιάζονται οι τερματικές συσκευές καθώς και τα μαθηματικά μοντέλα που χρησιμοποιούνται προκειμένου να προσομοιωθεί η διττή έξοδός τους. Επίσης, αναλύεται το δίκτυο και οι φυσικές του παράμετροι.

2.1 Παράμετροι και χαρακτηριστικά των τερματικών φωνής

Υποθέτουμε ότι κάθε τερματικό φωνής είναι εφοδιασμένο με έναν ανιχνευτή ενεργών περιόδων ομιλίας (Voice Activity Detector, VAD). Ο ανιχνευτής αυτός έχει την δυνατότητα να αντιλαμβάνεται τις μεταβάσεις της ανθρώπινης φωνής από ομιλία σε σιωπή και αντίστροφα, ενεργοποιώντας ή απενεργοποιώντας τον κωδικοποιητή φωνής της συσκευής. Με αυτόν τον τρόπο η τερματική συσκευή απαιτεί τη μετάδοση δεδομένων στον δίαυλο μόνο όταν ο χρήστης μιλάει ενώ όταν σιωπά το δίκτυο δεν θα επιβαρύνεται με «σιωπηλά» δεδομένα. Αυτές οι περίοδοι σιωπής μπορούν να χρησιμοποιηθούν για την μετάδοση πακέτων άλλων πηγών φωνής, που εκείνη την στιγμή δεν είναι σε σιωπή, επιτυγχάνοντας κατ' αυτόν τον τρόπο κέρδος λόγω της πολυπλεξίας. Εδώ αξίζει να σημειωθεί ότι τέτοιοι ανιχνευτές ομιλίας χρησιμοποιούνται και από τις τερματικές συσκευές των δικτύων δεύτερης γενιάς για να ενεργοποιούν τον πομπό μόνο κατά τις ενεργές περιόδους επιτυγχάνοντας έτσι χαμηλή ενεργειακή κατανάλωση. Υποθέτουμε ότι ο VAD είναι «αργός» [12],[13] και αντιλαμβάνεται διάρκειες ομιλίας μεγαλύτερες από 20 msec και διάρκειες παύσεων μεγαλύτερες από 200 msec. Σε μια δεδομένη χρονική στιγμή μια πηγή φωνής μπορεί να βρίσκεται σε μία από δύο καταστάσεις: την «ενεργή» (ομιλίας) ή την «ανενεργή» (σιωπής). Η πηγή φωνής μπορεί να μοντελοποιηθεί με αλυσίδα Markov διακριτού χρόνου, δύο

καταστάσεων, όπως φαίνεται στο σχήμα 2.1 (ο λόγος για αυτήν την επιλογή καθώς και η διάρκεια της μονάδας χρόνου θα εξηγηθούν παρακάτω).

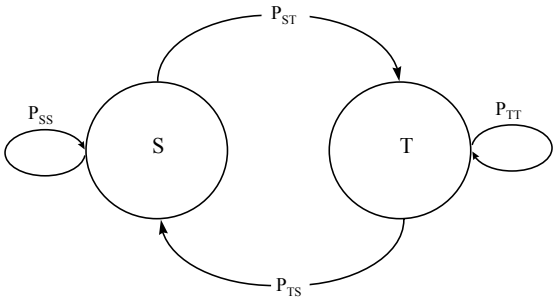
Η πιθανότητα μετάβασης από την ανενεργή στην ενεργή κατάσταση συμβολίζεται ως P_{ST} και η πιθανότητα μετάβασης από την ενεργή κατάσταση στην ανενεργή ως P_{TS}. Όμοια οι πιθανότητες παραμονής στην ενεργή ή στην ανενεργή κατάσταση συμβολίζονται ως P_{SS} και P_{TT}, αντίστοιχα.

Οι χρόνοι παραμονής μιας πηγής φωνής στην ανενεργή και ενεργή κατάσταση είναι γεωμετρικά κατανεμημένοι με μέση τιμή $1/P_{ST}$ και $1/P_{TS}$ μονάδες χρόνου αντίστοιχα. Στην κατάσταση ευστάθειας της αλυσίδας Markov η πιθανότητα η πηγή να βρίσκεται σε ενεργή κατάσταση P_T, ή σιωπηλή κατάσταση P_S, δίνεται από τις παρακάτω εξισώσεις:

$$P_T = P_{ST} / (P_{ST} + P_{TS})$$

$$P_S = 1 - P_T$$

Σχ. 2.1



2.2 Παράμετροι και χαρακτηριστικά των πηγών video

Η μετάδοση video πάνω από ένα επικοινωνιακό δίκτυο είναι κάτι που εξ' αιτίας αυτής ακριβώς της φύσης των κινούμενων εικόνων παρουσιάζει ιδιαίτερες δυσκολίες. Για να γίνει το πρόβλημα πιο συγκεκριμένο ας υποθέσουμε το εξής παράδειγμα: έστω μία σειρά από διαδοχικές έγχρωμες εικόνες σχετικά μικρών διαστάσεων, για παράδειγμα 100 × 100 στοιχείων (pixels). Αν η εικόνα αυτή είναι κωδικοποιημένη διατηρώντας την πληροφορία του χρώματος από μία παλέτα, ας πούμε 256 χρωμάτων, τότε θα απαιτούνται 8 bits για την αναπαράσταση κάθε ενός από αυτά τα pixels. Έτσι, συνολικά για κάθε εικόνα θα θέλαμε 100 × 100 = 10.000 pixels και στη συνέχεια 10.000 × 8 = 80.000 bits πληροφορίας. Δεδομένου του γεγονότος ότι το video αποτελείται από ταχύτατα διαδεχόμενες η μία την άλλη τέτοιες εικόνες (π.χ. 30 μέσα σε ένα δευτερόλεπτο), τότε συνολικά η αναπαράσταση ενός δευτερολέπτου video θα αντιστοιχούσε σε 2.400.000 bits πληροφορίας! Με άλλα λόγια θα ήταν αδύνατη η μεταφορά έστω και μίας ροής video ακόμη και πάνω από ένα ταχύτατο ασύρματο δίκτυο (π.χ. ταχύτητα μετάδοσης 1.8 Mbps), αν δεν υπήρχε κάποιος τρόπος για περιορισμό της έντασης με την οποία γεννάται πληροφορία από τις πηγές video.

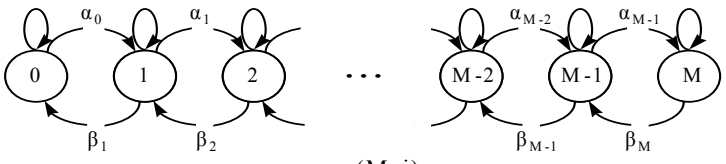
Υπάρχουν δύο βασικοί τρόποι για να επιτευχθεί αυτό. Ο ένας, αφορά τον συσχετισμό εικονοστοιχείων με ακριβώς ίδιες συντεταγμένες σε διαδοχικές χρονικά εικόνες. Όσο πιο αργές είναι οι κινήσεις των αντικειμένων στο video και όσο πιο πυκνά χρονικά είναι τα καρέ μεταξύ τους, τόσο πιο έντονη είναι η συσχέτιση των εικονοστοιχείων μεταξύ τους. Ο δεύτερος τρόπος, αφορά την συσχέτιση γειτονικών εικονοστοιχείων πάνω στην ίδια εικόνα. Και οι δύο αυτοί τρόποι εκμεταλλεύονται τις δυνατότητες και τις ευαισθησίες του ανθρώπινου ματιού όσον αφορά τα χρώματα και την ταχύτητα εναλλαγής εικόνων. Τα τελευταία χρόνια έχουν γίνει πολύ έντονες προσπάθειες σε αυτό το πεδίο που ως αποτέλεσμα είχαν την καθιέρωση ενός μεγάλου αριθμού προτύπων κωδικοποίησης video κατάλληλα για όλες σχεδόν

τις εφαρμογές. Ολοένα και συχνότερα μάλιστα εμφανίζονται και καινούργια πρότυπα τα οποία επιτυγχάνουν αφ' ενός μεν υψηλότερη συμπίεση, αφ' ετέρου ταχύτερη κωδικοποίηση και αποκωδικοποίηση. Ενδεικτικά θα αναφέρουμε τα πρότυπα MPEG1, MPEG2 για απαιτήσεις υψηλής πιστότητας και κατά συνέπεια υψηλών απαιτήσεων σε bandwidth, το πρότυπο MPEG4 κατάλληλο για εφαρμογές, όπως η δική μας, καθώς και τα πρότυπα H.263 και H.264 κατάλληλα κυρίως για ψηφιοποίηση ροών video «χαμηλής κινητικότητας». Ο όρος «χαμηλή κινητικότητα» αφορά ροές οι οποίες μεταφέρουν περιεχόμενο στο οποίο τα αντικείμενα δεν παρουσιάζουν απότομες αλλαγές στην κίνηση τους. Έτσι, ένας ποδοσφαιρικός αγώνας, ο οποίος εξ' ορισμού περιέχει έντονες αλλαγές πλάνων καθώς και αντικείμενα τα οποία μετακινούνται με ταχύτητα στην οθόνη, αντιστοιχεί σε μία ροή υψηλής κινητικότητας, ενώ ένα άτομο το οποίο δίνει συνέντευξη ή εκφωνεί ένα λόγο συνθέτει μια ροή χαμηλής κινητικότητας. Η ουσιαστική διαφορά στην αντιμετώπιση των δύο αυτών διαφορετικών περιεχομένων βρίσκεται στο ότι ενώ για την πρώτη κατηγορία δεν μπορεί να επιτευχθεί ουσιαστική συμπίεση από την συσχέτιση εικονοστοιχείων με ίδιες συντεταγμένες σε γειτονικά καρέ, στην δεύτερη περίπτωση τα αποτελέσματα από μία τέτοια αντιμετώπιση είναι πολύ εντυπωσιακά.

Σε αυτήν την εργασία το μοντέλο πηγής video που χρησιμοποιείται στις προσομοιώσεις είναι ανεξάρτητο από το πρότυπο κωδικοποίησης που χρησιμοποιείται. Η μόνη απαίτηση αφορά την χαμηλή κινητικότητα του περιεχομένου του, όπως την ορίσαμε, πιο πάνω και την μέγιστη χρονική απόσταση δύο διαδοχικών καρέ. Το μοντέλο αυτό προτείνεται στο [14] και είναι ένα Μαρκοβιανό birth and death σύστημα (βλέπε Σχ. 2.2). Σύμφωνα με την λογική που ακολουθείται προκειμένου να μοντελοποιηθεί ικανοποιητικά η συμπεριφορά μίας πραγματικής πηγής video υποτίθεται ένα άνω και ένα κάτω όριο στο ρυθμό γέννησης πληροφορίας. Η πρώτη κατάσταση (ή κατάσταση 0) στην αλυσίδα Markov αναπαριστά την πηγή στην περίπτωση που γεννά πακέτα στον ελάχιστο ρυθμό, L. Η τελευταία κατάσταση (ή κατάσταση M) της

αλυσίδας αντιστοιχεί στον μέγιστο ρυθμό γέννησης πληροφορίας, P. Ενδιάμεσα τους υπάρχει ένας πεπερασμένος αριθμός από διακριτά επίπεδα ρυθμού γέννησης πληροφορίας. Το πλήθος των πεπερασμένων αυτών ρυθμών ορίζεται σαφώς στο [14] και είναι ανεξάρτητο από τις ακραίες τιμές P και L. Οι ρυθμοί μετάβασης από την μία κατάσταση στην άλλη εξαρτώνται άμεσα από την τρέχουσα κατάσταση στην οποία βρίσκεται η πηγή και προσδιορίζονται από το ταίριασμα (matching) της μέσης τιμής και της διασποράς του ρυθμού γέννησης πληροφορίας με αυτά της πραγματικής πηγής video.

Σχ. 2.2



$\alpha_i=(M-i)a$
$\beta i=ib$
$i=0..M$

Από τις εξισώσεις ισορροπίας ($\Pi_0\alpha_0=\Pi_1\beta_1$, $\Pi_1\alpha_1=\Pi_2\beta_2$ κ.ο.κ.) του παραπάνω συστήματος προκύπτει ότι η πιθανότητα το σύστημα να βρίσκεται στην κατάσταση i μια δεδομένη χρονική στιγμή είναι:

$$\Pi_i=\Pi_0(c_1/c_2)^{\iota}\binom{M}{\iota}$$

Έτσι προκύπτει ότι ο μέσος ρυθμός δίνεται από:

$$E(\lambda)= L+MA\ c_1/(c_1+c_2) \text{ με } A=(P-L)/M.$$

Ενώ η διασπορά του ρυθμού γέννησης πληροφορίας από:

$$Var(\lambda)= A^2M\ (c_1c_2)/(c_1+c_2)^2$$

Αν υποθέσουμε ότι οι ροές video αντιστοιχούν σε εικόνες μικρών διαστάσεων (π.χ. 80 × 60 pixels) τότε ο μέσος ρυθμός γέννησης δεδομένων από την πηγή είναι 62.400 bits ανά sec, η τυπική απόκλιση του ρυθμού γέννησης είναι 30580 bits ανά sec (0,23 bits/pixel [14]), ο ελάχιστος ρυθμός είναι 9.600 bits ανά sec και ο μέγιστος ρυθμός είναι 169.200 bits ανά sec [14].

Κάνοντας matching τις παραπάνω εκφράσεις με τις επιθυμητές τιμές που χαρακτηρίζουν τις πηγές μας και υποθέτοντας ότι $c_2=1$ προκύπτει ότι το $M=6$ (δηλαδή η αλυσίδα Markov έχει 7 καταστάσεις) και το $c_1=6$.

Πρέπει εδώ να τονιστεί ότι σε ορισμένα πρότυπα κωδικοποιήσεων ο όρος «ανάλυση» δεν ανταποκρίνεται απόλυτα στην πραγματικότητα. Στην κωδικοποίηση MPEG για παράδειγμα αναγνωρίζονται blocks από ψηφία τα οποία θεωρούνται και οι ελάχιστες μονάδες υλοποίησης της εικόνας.

2.3 Παράμετροι και χαρακτηριστικά του δικτύου.

Στο σύστημά μας υπάρχουν δύο φυσικοί επικοινωνιακοί δίαυλοι: ένας από τα τερματικά προς τον Σ.Β. και ένας από τον Σ.Β. προς τα τερματικά.

Όπως θα δούμε αναλυτικότερα και στο κεφάλαιο 3 οι δύο αυτοί δίαυλοι έχουν τα ίδια φυσικά χαρακτηριστικά, όπως και οι πληροφορίες που τους διοχετεύεται. Δεδομένης της φυσικής ομοιότητας των δύο διαύλων σε αυτήν την εργασία θα εστιάσουμε στον έναν από αυτούς και συγκεκριμένα στον πρώτο.

Ο δίαυλος αυτός είναι χωρισμένος σε ίσα περιοδικά χρονικά διαστήματα (time frames). Τα χρονικά αυτά διαστήματα έχουν διάρκεια όση ακριβώς απαιτείται για να γεννηθεί ένα πακέτο πληροφορίας από μια πηγή φωνής. Το δίκτυο λοιπόν, είναι «ευθυγραμμισμένο» με τις χρονικές κλίμακες των πηγών φωνής και όχι των πηγών video και τούτο διότι αφ΄ ενός υποθέτουμε ότι η εξυπηρέτηση της φωνής είναι σημαντικότερη από εκείνη του video και αφ' ετέρου, όπως θα δούμε και παρακάτω, η συνήθης κατάσταση είναι σε ένα channel frame να υπερτερούν αριθμητικά οι πηγές φωνής και όχι οι πηγές video. Αυτό προκύπτει από την εκτίμηση ότι οι χρήστες απαιτούν πολύ συχνότερα υπηρεσίες φωνής απ΄ ότι video. Μια αντίθετη αντιμετώπιση θα είχε σαν αποτέλεσμα την σπατάλη bandwidth στο κανάλι. Σημαντικό είναι επίσης εδώ να αναφερθεί ότι έχει υποτεθεί η απουσία λαθών λόγω θορύβου στο κανάλι. Τούτο ανταποκρίνεται σε ικανοποιητικό βαθμό στην πραγματικότητα, καθώς, όπως ήδη αναφέρθηκε, οι μικροκυψέλες του δικτύου μας έχουν πολύ μικρή ακτίνα με αποτέλεσμα την καλή ποιότητα επικοινωνίας τερματικών συσκευών και Σ.Β..

Τα φυσικά χαρακτηριστικά του συστήματός μας αναλύονται στον πίνακα 2.1. Η ταχύτητα μετάδοσης του καναλιού είναι από τα [15], [16] καθώς και από το [4]. Το μέγεθος του πακέτου πληροφορίας είναι σε συμφωνία με τα πακέτα του ενσύρματου δικτύου ATM (48 bytes πληροφορίας και 5 bytes επικεφαλίδα). Η ταχύτητα γέννησης δεδομένων από τις πηγές φωνής προκύπτει από κωδικοποίηση Adaptive Pulse Code Modulation στα 32 kbps.

Η διάρκεια του channel frame προκύπτει όπως προαναφέρθηκε, βάσει του χρονικού διαστήματος δημιουργίας ενός πακέτου φωνής. Έτσι, αν η ταχύτητα γέννησης δεδομένων από μια πηγή φωνής σε ενεργή κατάσταση

είναι 32 kbps και το μέγεθος του πακέτου πληροφορίας 48x8 = 384 bits, τότε για να γεννηθούν αυτά τα 384 bits απαιτείται χρονική διάρκεια 384/32000 = 0.012 δευτερολέπτων ή 12 ms. Προκειμένου να μεταδοθούν από τον πομπό αυτά τα 53 bytes, με ταχύτητα 1,8 Mbps απαιτούνται 424 bits / 1800000 bps = 0,23555 msecs. Δηλαδή μέσα στο χρονικό διάστημα ενός channel frame, 12 msecs, προλαβαίνουν να μεταδοθούν περίπου 51 πακέτα από τον πομπό. Συνεπώς κάθε channel frame υποδιαιρείται σε 50 ίσα χρονικά κλάσματα - θυρίδες (slots) τα οποία και «γεμίζουν» από μεταδόσεις πακέτων. Κάθε μία από αυτές τις χρονικές θυρίδες χωρά ακριβώς ένα πακέτο. Στην προσομοίωση έχουμε θεωρήσει ότι ο αριθμός αυτών των θυρίδων είναι 50 και όχι 51, υποθέτοντας ότι το πρώτο από τα channel slots χρησιμοποιείται για άλλες ανάγκες (π.χ. συγχρονισμού) [17]. Το μέγιστο όριο καθυστέρησης ενός πακέτου φωνής ορίζεται σε 24 ms δηλαδή ίσο με 2 channel frames. Τέλος, η μέση διάρκεια μιας περιόδου ομιλίας υποτίθεται ίση με 1.41 sec, ενώ μιας περιόδου σιωπής ίση με 1.78 sec [13]. Αυτή η υπόθεση έχει άμεση συσχέτιση με το πόσο αργός ή γρήγορος είναι ο VAD. Πάντως η σταθερή αναλογία ανάμεσα στις διάρκειες της ενεργής και σιωπηλής κατάστασης είναι περίπου 44% έναντι 56% αντίστοιχα [12].

Πίνακας 2.1

Παράμετρος	ΤΙΜΗ
CHANNEL RATE	1.8 Mbps
SPEECH RATE	32 Kbps
INFORMATION PACKET	424 bits
FRAME DURATION	12 msecs
MEAN TALKSPURT DURATION	1.41 secs
MEAN SILENCE DURATION	1.78 secs

Σύμφωνα με τα παραπάνω και δοθέντος ότι το video προϋποθέτει ένα ρυθμό εναλλαγής των καρέ περίπου 30 ανά sec, μπορούμε να κάνουμε την απλή σύμβαση ότι οι εναλλαγές των καρέ συμβαίνουν ανά τρία channel frames δηλαδή ανά 36 msecs. Σε αυτή την περίπτωση, μέσα σε ένα δευτερόλεπτο αντί

για 30 θα έχουμε 27,7 αλλαγές καρέ. Αυτό αφ' ενός δεν αναμένεται να έχει καμία ουσιαστική επίπτωση στην αναπαράσταση του video αφ' ετέρου μας παρέχει σημαντική ευκολία στη σχεδίαση των αλγορίθμων πολυπλεξίας. Επιπλέον η υπόθεση αυτή καθορίζει και την μέγιστη ανοχή σε καθυστερήσεις των πακέτων video σε τρία channel frames ή 36 msecs.

Κεφάλαιο 3

Ο αλγόριθμος χρονοδρομολόγησης (BS Scheduling Algorithm)

Εδώ θα παρουσιαστεί ο μηχανισμός βάσει του οποίου επιτυγχάνεται η μετάδοση των πακέτων πληροφορίας στον Σ.Β.. Θα αναλυθεί βήμα προς βήμα η λογική που ακολουθείται προκειμένου να χτιστεί ο αλγόριθμος ούτως ώστε να έχουμε τις ελάχιστες και όσο το δυνατόν πιο «ανώδυνες» απώλειες πακέτων.

Όπως έχει ήδη αναφερθεί, το δίκτυό μας ουσιαστικά αποτελείται από διαφορετικά φυσικά κανάλια μέσα στα οποία ρέει η πληροφορία υπό μορφή πακέτων. Έτσι είναι διαφορετικό το κανάλι, που εξυπηρετεί τα πακέτα που οδεύουν από τις κινητές συσκευές στον Σ.Β., από αυτό που εξυπηρετεί τα πακέτα πληροφορίας που οδεύουν προς τις κινητές συσκευές. Φυσικά αυτά τα δύο κανάλια έχουν παρόμοια αν όχι πανομοιότυπα χαρακτηριστικά. Άλλωστε, το είδος της πληροφορίας που μεταδίδεται και στα δύο θεωρούμε ότι είναι ακριβώς το ίδιο από πλευράς περιεχομένου (video και ομιλία) όσο και από στατιστικής πλευράς.

Ωστόσο, υπάρχουν και δύο άλλα φυσικά κανάλια τα οποία αν και πολύ μικρότερης (συγκριτικά τουλάχιστον) χωρητικότητας παίζουν ιδιαίτερα σημαντικό ρόλο. Κάθε ένα από αυτά τα δύο μικρότερα κανάλια αντιστοιχεί σε ένα από τα μεγαλύτερα. Τα τερματικά (ή ο Σ.Β. στην αντίστοιχη περίπτωση) μεταδίδουν εκεί πληροφορία σχετικά με τις απαιτήσεις τους σε πόρους του βασικού καναλιού στο οποίο μεταδίδουν πληροφορία. Το bandwidth του συμπληρωματικού αυτού καναλιού αν και μικρό, υποτίθεται αρκετό έτσι ώστε να μην έχουμε συμφόρηση. Κι αυτό γιατί οι αιτήσεις των τερματικών με τις απαιτήσεις εύρους ζώνης πρέπει να φτάνουν στον Σ.Β. σε ελάχιστο χρόνο

προκειμένου να ξεκινήσει από τον τελευταίο η εκτέλεση του αλγορίθμου για την χρονοδρομολόγηση των πακέτων πληροφορίας.

Σε αυτό το σημείο αξίζει να αναφερθεί ότι ένα επιπλέον πολύ σημαντικό πλεονέκτημα της πολύ μικρής διάστασης των κυψελών του δικτύου μας (πέρα της καθαρότητας του λαμβανόμενου σήματος) είναι η πολύ μικρή χρονική καθυστέρηση διάδοσης (propagation) που υφίσταται το σήμα μέχρι να φτάσει από τον Σ.Β. στις τερματικές συσκευές και αντίστροφα. Έτσι στην χειρότερη περίπτωση (δηλαδή αν ο χρήστης βρίσκεται στην περίμετρο της κυψέλης) η μέγιστη χρονική καθυστέρηση διάδοσης του σήματος, Pmax, θα είναι:

$$Pmax = 100\ m\ /\ (300.000.000\ msecs\ *\ 2/3) \approx 0.00005\ sec = 50\ \mu secs$$

Ο χρόνος αυτός είναι 24000 φορές μικρότερος από την χρονική διάρκεια ενός channel frame και κατά συνέπεια μπορούμε να θεωρήσουμε ότι ο Σ.Β. ενημερώνεται άμεσα για τις απαιτήσεις των κινητών τερματικών σε πόρους του καναλιού και αρχίζει αμέσως την εκτέλεση του αλγορίθμου χρονοδρομολόγησης για την κατανομή του χρόνου του καναλιού στα κινητά τερματικά.

Ο αλγόριθμός δίνει προτεραιότητα στην εξυπηρέτηση πακέτων πληροφορίας που προέρχονται από την κωδικοποίηση πηγών φωνής. Έτσι σε περίπτωση που ο αριθμός των πακέτων πληροφορίας που θα γεννηθούν από τα τερματικά - πηγές μέσα σε ένα channel frame υπερβαίνει τα 50, τότε επιλέγονται από τον scheduler στο Σ.Β. και τους δίνεται άδεια να μεταδοθούν αρχικά αυτά τα οποία προέρχονται από πηγές φωνής και στη συνέχεια εκείνα τα οποία μεταφέρουν πληροφορία video. Στην σπάνια περίπτωση που τα πακέτα φωνής πλεονάζουν, υπερβαίνοντας τα 50 μέσα σε ένα channel frame (δηλαδή υπάρχουν ταυτόχρονα πάνω από 50 ενεργές πηγές φωνής), τότε δίνεται προτεραιότητα στις πηγές οι οποίες βρίσκονται ήδη από τα προηγούμενα channel frames σε κατάσταση ενεργούς ομιλίας. Κι αυτό γιατί θεωρούμε ότι η

διατήρηση του επιπέδου ποιότητας μιας ήδη εγκατεστημένης κλήσης είναι σημαντικότερη από ότι είναι η εγκατάσταση μιας νέας. Τα πακέτα φωνής τα οποία δεν πρόλαβαν να εξυπηρετηθούν μέσα στο τρέχον channel frame δεν προσπαθούν να επαναμεταδοθούν στο επόμενο. Θεωρείται ότι εξάντλησαν το όριο καθυστέρησης και απορρίπτονται.

Στο τέλος αυτού του κεφαλαίου θα διευκρινιστεί ο λόγος που γίνεται αυτή η υπόθεση. Προς το παρόν, αυτό το οποίο πρέπει να τονιστεί είναι ότι τέτοιες περιπτώσεις είναι ιδιαίτερα σπάνιες για δύο βασικούς λόγους. Ο πρώτος είναι ότι λόγω του περιορισμένου εμβαδού των μικροκυψελών το πλήθος των ενεργών τερματικών μέσα σε κάθε μία από αυτές δεν αναμένεται να είναι ιδιαίτερα υψηλό. Ο δεύτερος είναι ότι οι κλήσεις φωνής πριν αρχίσουν να εξυπηρετούνται από το δίκτυο (δηλαδή πριν ακόμα ξεκινήσει μία συνομιλία), ζητούν άδεια από τον Σ.Β. για να εγκαταστήσουν την σύνδεση μαζί του. Αν ο Σ.Β. κρίνει ότι το δίκτυο έχει τα περιθώρια για να εξυπηρετήσει την νέα συνδιάλεξη ενημερώνει την τερματική συσκευή και φροντίζει για την προώθηση των πακέτων φωνής της κλήσης. Σε αντίθετη περίπτωση ενημερώνει το τερματικό ότι η υπηρεσία είναι προς το παρόν μη διαθέσιμη (κατειλημμένο σήμα) και το καλεί έμμεσα να ξαναδοκιμάσει αργότερα. Η αρχή που εφαρμόζεται εδώ είναι απλή: είναι προτιμότερο μία συνδιάλεξη να μην ξεκινήσει καθόλου παρά να διακοπεί ξαφνικά ή να μην έχει την απαιτούμενη ποιότητα. Η τεχνική που υλοποιεί αυτή την αρχή είναι γνωστή ως έλεγχος εισόδου (admission control) και χρησιμοποιείται από όλα τα δίκτυα που εγγυώνται ποιότητα υπηρεσίας (QoS).

Μόλις όλα τα πακέτα φωνής εξυπηρετηθούν ο Σ.Β. αρχίζει να διαθέτει ελεύθερες χρονοθυρίδες (αν φυσικά υπάρχουν διαθέσιμες) από το συγκεκριμένο frame σε πακέτα που προέρχονται από κωδικοποίηση video. Εδώ πρέπει να αναφερθεί ότι στην περίπτωση που υπάρχει αδυναμία να παραχωρηθούν από τον scheduler στο ακέραιο οι χρονοθυρίδες* που έχουν

*Θυμίζουμε εδώ ότι ένα τερματικό video μπορεί να έχει για μετάδοση περισσότερα από ένα πακέτα σε ένα channel frame.

ζητηθεί από κάποια πηγή video στο συγκεκριμένο channel frame, τότε στην συγκεκριμένη πηγή δεν παραχωρείται καμία χρονοθυρίδα. Και αυτό γιατί γνωρίζουμε ότι στους σύγχρονους τρόπους κωδικοποίησης ροών video (MPEG-4, H.xxx κτλ) τα υψηλά ποσοστά συμπίεσης που επιτυγχάνονται οφείλονται κυρίως στον υψηλό βαθμό αυτοσυσχέτισης. Σε περίπτωση συνεπώς απώλειας κάποιου πακέτου μιας ακολουθίας video, είναι ιδιαίτερα δύσκολη η αποκωδικοποίηση της πληροφορίας που περιλαμβάνεται σε ένα σημαντικό αριθμό από γειτονικά πακέτα. Επιλέγουμε λοιπόν στην περίπτωση που δεν είναι δυνατόν να ικανοποιηθούν στο ακέραιο οι απαιτήσεις μιας πηγής video σε χρονοθυρίδες στο τρέχον channel frame να γίνει προσπάθεια να ικανοποιηθούν στο ακέραιο οι απαιτήσεις κάποιας άλλης πηγής video που ζητά λιγότερες χρονοθυρίδες από την πρώτη.

Η σειρά εξυπηρέτησης των πηγών video εξαρτάται από το πλήθος των μη ικανοποιημένων αιτήσεων που είχαν στο παρελθόν. Η προτεραιότητα φυσικά δίνεται στις πηγές με τις μεγαλύτερες απώλειες. Η στρατηγική αυτή επιτυγχάνει δίκαιη αντιμετώπιση όλων των πηγών video και μεγιστοποιεί το χρονικό διάστημα ανάμεσα σε δύο διαδοχικές απώλειες πακέτων από την ίδια πηγή video.

Σε αυτό το σημείο δύο διαφορετικές προσεγγίσεις μπορούν να γίνουν για τον υπολογισμό των απωλειών κάθε πηγής. Στην πρώτη η μετρική προτεραιοποίησης είναι ο αριθμός των channel frames στα οποία δεν ικανοποιήθηκαν οι αιτήσεις της πηγής για μετάδοση πακέτων. Η μετρική αυτή είναι γνωστή ως συχνότητα κηλίδας (glitch rate) καθώς σε κάθε τέτοιο channel frame ο αποδέκτης του αντίστοιχου σήματος video αντιλαμβάνεται ένα στιγμιαίο τρεμούλιασμα ή μία θαμπή περιοχή στην οθόνη του. Στην δεύτερη προσέγγιση η μετρική προτεραιοποίησης είναι το σύνολο των πακέτων που μία πηγή video έχει προσπαθήσει ανεπιτυχώς να στείλει στον Σ.Β.. Ουσιαστικά οι δύο προσεγγίσεις είναι ισοδύναμες σε μόνιμη κατάσταση, καθώς όλες οι πηγές video γεννούν πακέτα με τον ίδιο ρυθμό και κατά συνέπεια και ο μέσος

αριθμός πακέτων που έχουν για μετάδοση σε ένα channel frame είναι σταθερός.

Στην προσομοίωση του συστήματος έχει χρησιμοποιηθεί η δεύτερη προσέγγιση. Οι μόνες απαιτήσεις για τον scheduler στον Σ.Β. κατά την εκτέλεση του αλγορίθμου είναι η ανάγκη ύπαρξης ενός μετρητή «αποτυχημένων» προσπαθειών μετάδοσης πακέτων, για κάθε μία από τις πηγές video και η ιεράρχηση των μετρητών αυτών στο τέλος κάθε channel frame. Λόγω όμως του σχετικά μικρού αναμενόμενου αριθμού πηγών video μέσα σε κάθε μικροκυψέλη, δεν είναι απαγορευτικό.

Όπως ήδη αναφέρθηκε στο τέλος του κεφαλαίου 2 το άνω όριο για την καθυστέρηση των πακέτων video είναι 36 msecs ή τρία channel frames. Αυτό σημαίνει ότι αν κάποιο πακέτο video δεν κατορθώσει να πάρει έγκριση για μετάδοση στο πρώτο channel frame που θα ζητήσει τότε υπάρχει το χρονικό περιθώριο ο scheduler να δώσει άδεια μετάδοσης σε κάποιο από τα επόμενα δύο channel frames. Κάτι τέτοιο φυσικά δεν πρέπει να γίνει σε βάρος των «φρέσκων» πακέτων που για πρώτη φορά ζητούν χρονοθυρίδα για να φτάσουν στον Σ.Β. και γι' αυτό η ευκαιρία για επαναμετάδοση δίδεται μόνο μετά το τέλος της μετάδοσης όλων των «φρέσκων» πακέτων. Εδώ βέβαια υπάρχει και η πιθανότητα της ανάγκης για αναδιάταξη των πακέτων video στην σωστή σειρά τους όταν αυτά φτάσουν στον αποκωδικοποιητή. Το παραπάνω, σε συνδυασμό με την ανάγκη για «απομνημόνευση» του αριθμού των χαμένων πακέτων στον πομπό είναι εξαιρετικά δαπανηρό σε υπολογιστική ισχύ. Κατά την προσομοίωση έχουν ακολουθηθεί και τα δύο σενάρια και με αυτόν τον τρόπο παρουσιάζονται και οι διαφορές στα αποτελέσματα των δύο αλγορίθμων.

Τέλος, πρέπει να αναφέρουμε τρεις βασικές απλοποιήσεις και συμβάσεις που έχουν ληφθεί υπ' όψιν στην προσομοίωση:

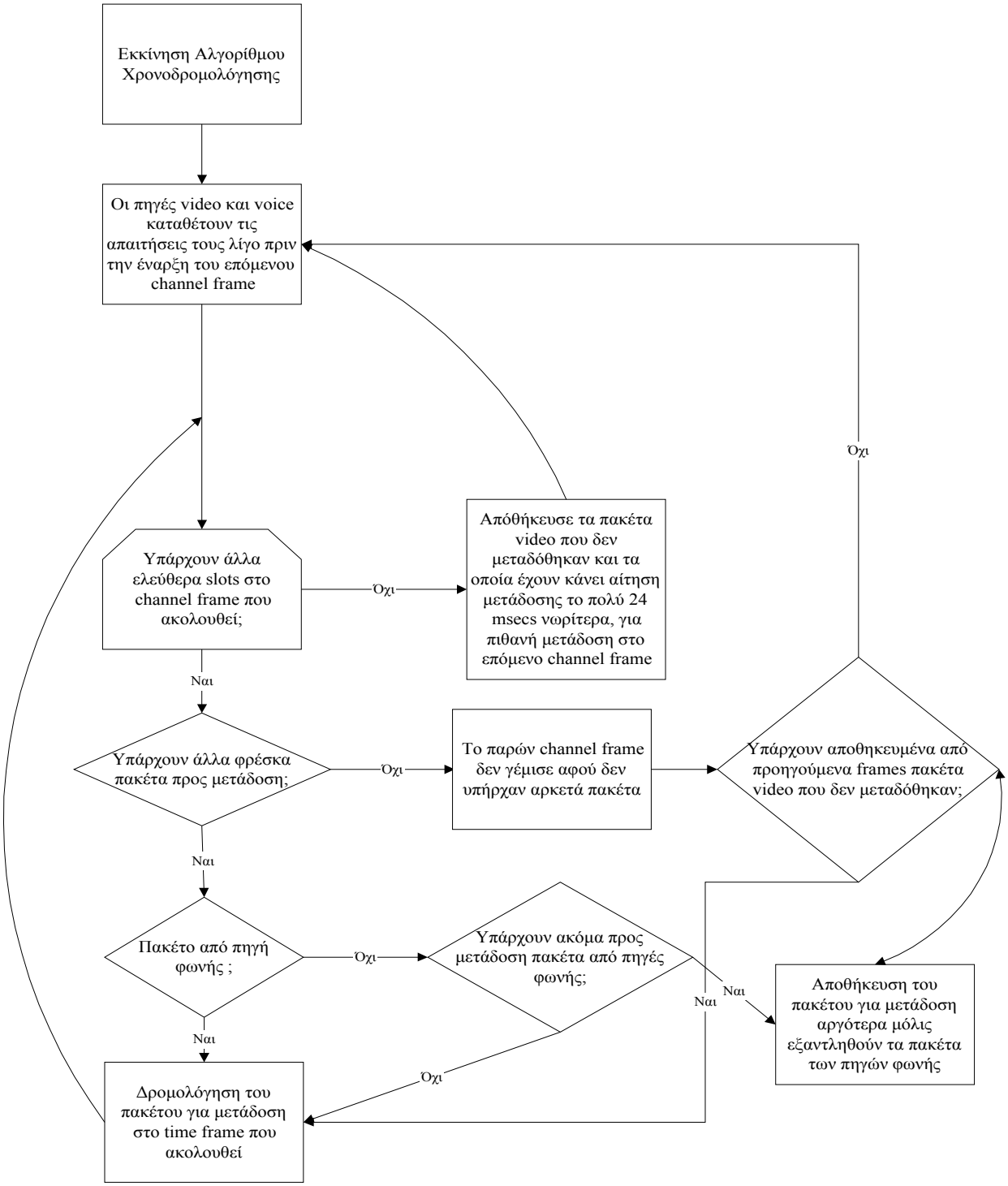
α. Ο αριθμός των κινητών τερματικών, που υποθέτουμε, είναι σταθερός. Τούτο διότι οι αλλαγές στον αριθμό των τερματικών σε μια συγκεκριμένη κυψέλη, συμβαίνουν σε χρονικά διαστήματα της τάξης μερικών δεκάδων δευτερολέπτων, χρονική διάρκεια ιδιαίτερα μεγάλη για ένα τέτοιο γρήγορο δίκτυο.

β. Οι μεταβάσεις των πηγών φωνής από την μία κατάσταση στην άλλη (Silence to Talkspurt και Talkspurt to Silence) λαμβάνουν χώρα μόνο στην αρχή των channel frames. Αυτή η σύμβαση ουσιαστικά δημιουργεί μία ελαστικότητα στον τρόπο με τον οποίο μετράται η χρονική καθυστέρηση ενός πακέτου πληροφορίας. Στην πραγματικότητα ένα πακέτο μπορεί να δημιουργηθεί από μία πηγή φωνής ακόμη και στην αρχή κάποιου channel frame επιβαρυνόμενη με αυτόν τον τρόπο (στην χειρότερη των περιπτώσεων) με σχεδόν 12 msecs καθυστέρηση κατά κάποιο τρόπο ad hoc. Το όριο καθυστέρησης όμως ενός πακέτου φωνής είναι 24 msecs ή δύο channel frames. Το σχήμα χρονοδρομολόγησης όμως που ακολουθεί ο Σ.Β. προκειμένου να δεσμεύσει τους πόρους του καναλιού όπως είπαμε δεν περιλαμβάνει την δυνατότητα επαναμετάδοσης ενός πακέτου φωνής. Έτσι η μέγιστη χρονική καθυστέρηση ενός πακέτου φωνής στο σύστημά μας εκ' των πραγμάτων δεν θα υπερβεί ποτέ τα 24 msecs. Σε αντίθετη περίπτωση το πακέτο θεωρείται χαμένο.

γ. Όμοια με το β., οι αλλαγές στον ρυθμό γέννησης πληροφορίας των πηγών video θεωρείται ότι λαμβάνουν χώρα στην αρχή των channel frames. Και εδώ η θέσπιση ενός συντηρητικού ορίου άνω καθυστέρησης (δηλ. των 36

msecs) ισοσταθμίζει αυτή την «ευκολία» τοποθετώντας το σύστημά μας σε ρεαλιστική βάση.

Στο σχήμα της επόμενης σελίδας παρουσιάζεται σχηματικά ένα βασικό κομμάτι του scheduling αλγόριθμου που ακολουθείται. Δυστυχώς λόγω της πολυπλοκότητάς του ο αλγόριθμος είναι δύσκολο να παρασταθεί σχηματικά ακέραιος σε μία σελίδα και γι' αυτό ακριβώς το λόγο δίνεται μόνο η βασική του δομή. Για παράδειγμα δεν φαίνεται η προτεραιοποίηση των πηγών φωνής από τον scheduler με βάση τις απώλειες που έχουν υποστεί μέχρι στιγμής.


Εκκίνηση Αλγορίθμου Χρονοδρομολόγησης
Οι πηγές video και voice καταθέτουν τις απαιτήσεις τους λίγο πριν την έναρξη του επόμενου channel frame
Υπάρχουν άλλα ελεύθερα slots στο channel frame που ακολουθεί;
Όχι
Απόθήκευσε τα πακέτα video που δεν μεταδόθηκαν και τα οποία έχουν κάνει αίτηση μετάδοσης το πολύ 24 msecs νωρίτερα, για πιθανή μετάδοση στο επόμενο channel frame
Όχι
Ναι
Υπάρχουν άλλα φρέσκα πακέτα προς μετάδοση;
Όχι
Το παρών channel frame δεν γέμισε αφού δεν υπήρχαν αρκετά πακέτα
Υπάρχουν αποθηκευμένα από προηγούμενα frames πακέτα video που δεν μεταδόθηκαν;
Ναι
Πακέτο από πηγή φωνής ;
Όχι
Υπάρχουν ακόμα προς μετάδοση πακέτα από πηγές φωνής;
Ναι
Ναι
Αποθήκευση του πακέτου για μετάδοση αργότερα μόλις εξαντληθούν τα πακέτα των πηγών φωνής
Ναι
Όχι
Δρομολόγηση του πακέτου για μετάδοση στο time frame που ακολουθεί

Κεφάλαιο 4

Πειραματικά αποτελέσματα

Στο κεφάλαιο αυτό θα παρουσιαστούν αναλυτικά τα αποτελέσματα των προσομοιώσεων των αλγορίθμων χρονοπρογραμματισμού του καναλιού και θα γίνουν συγκρίσεις και σχόλια για την αποτελεσματικότητά τους κάτω από διαφορετικές συνθήκες φορτίου.

4.1 Συνθήκες και Σκοπός της Προσομοίωσης

Η προσομοίωση θεωρεί μία μικροκυψέλη σε λειτουργία η οποία περιέχει ένα σταθερό αριθμό από ενεργές πηγές video. Το πλήθος τους κυμαίνεται από 2 έως 9. Ο στόχος μας ήταν αφ' ενός να προσδιορίσουμε τον μέγιστο αριθμό από πηγές φωνής που μπορούν να εξυπηρετηθούν από το κανάλι έτσι ώστε το video packet loss rate να μην υπερβεί κάποια συγκεκριμένα άνω όρια (π.χ., 10^{-4}) και αφ' ετέρου να προσδιορίσουμε το αντίστοιχο ποσοστό χρησιμοποίησης του καναλιού.

Κάθε κύκλος προσομοίωσης αντιστοιχεί σε 100000 channel frames, δηλαδή 20 λεπτά πραγματικού χρόνου λειτουργίας του δικτύου. Μία επιπρόσθετη αρχική περίοδος 5000 channel frames (1 λεπτό πραγματικής λειτουργίας) χρησιμοποιήθηκε σαν περίοδος προθέρμανσης του προσομοιωτή. Κάθε αποτέλεσμα προήλθε από το τρέξιμο 6 ανεξάρτητων προσομοιώσεων (Monte Carlo runs).

Παρουσιάζονται τα αποτελέσματα και από τις δύο προσεγγίσεις σχετικά με την επαναμετάδοση των video πακέτων.

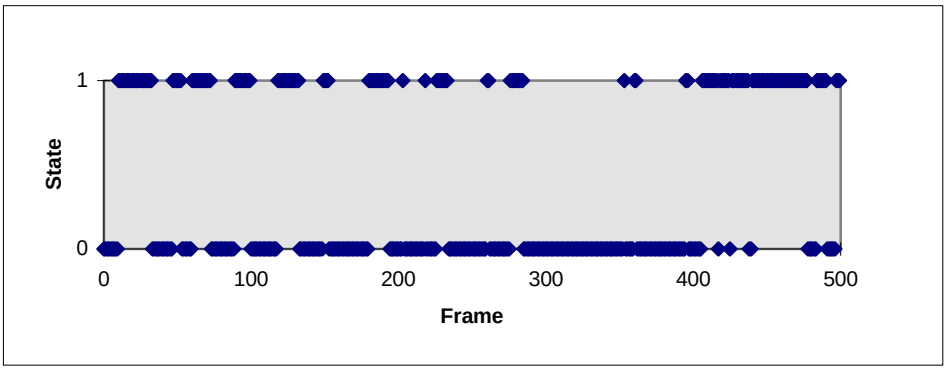
Ο κώδικας της προσομοίωσης περιέχεται στο τέλος της εργασίας. Για το compilation χρησιμοποιήθηκε η Borland C++ v4.5 σε λειτουργικό σύστημα

Windows 95. Για τις εκτελέσεις των προγραμμάτων χρησιμοποιήθηκε ένας υπολογιστής PC με επεξεργαστή Intel Pentium 166Mhz και η χρονική διάρκεια της κάθε εκτέλεσης ήταν από 2 έως 8 λεπτά περίπου ανάλογα με το πλήθος των τερματικών φωνής και video που θεωρούσαμε.

4.2 Πηγές Φωνής

Όπως προείπαμε οι πηγές φωνής κωδικοποιούν το αναλογικό σήμα φωνής χρησιμοποιώντας ADPCM [2], παράγοντας έτσι στις ενεργές περιόδους τους δεδομένα με ρυθμό ίσο με 32 Kbps. Επίσης αναφέραμε ότι για την προσομοίωση των πηγών αυτών χρησιμοποιούμε αντίστοιχο αριθμό από αλυσίδες Markov δύο καταστάσεων. Στην αρχή κάθε channel frame (δηλαδή κάθε 12 msecs) για κάθε πηγή φωνής (δηλαδή για κάθε αλυσίδα Markov) ο προσομοιωτής «ρίχνει» ένα νόμισμα και α) αν η πηγή βρίσκεται ήδη σε κατάσταση ομιλίας τότε με πιθανότητα 0.008510638 μεταβαίνει στην κατάσταση σιωπής, ή β) αν ήδη βρίσκεται σε σιωπηρή κατάσταση με πιθανότητα 0.006741573 μεταβαίνει στην κατάσταση ομιλίας. Στο σχήμα 4.1 παρουσιάζεται γραφικά η συμπεριφορά ενός από τους προσομοιωτές πηγών φωνής για το πολύ σύντομο χρονικό διάστημα των 500 channel frames , δηλαδή 6 secs. Με 1 συμβολίζεται η ενεργή κατάσταση (ομιλία, talkspurt) και με 0 η ανενεργή κατάσταση (σιωπή, silence).

Σχ. 4.1



Όπως φαίνεται στο σχήμα αυτό ο ανιχνευτής ομιλίας (Voice Activity Detector) αν και σχετικά αργός προλαβαίνει να αντιληφθεί αρκετά μικρές παύσεις (της

τάξης των 10 channel frames ή και λιγότερο, δηλαδή περίπου 10 εκατοστά του δευτερολέπτου) οι οποίες δεν γίνονται αντιληπτές από τα ανθρώπινα αισθητήρια (όπως για παράδειγμα κενά ανάμεσα σε δύο διαδοχικούς φθόγγους). Επίσης είναι φανερό ότι οι περίοδοι σιωπής είναι κατά μέσον όρο μεγαλύτερες και πυκνότερες (αντιστοιχούν σε περίπου 56% του χρόνου).

Το ανώτατο ανεκτό όριο για απώλειες πακέτων φωνής είναι 1 πακέτο στα 100. Απώλεια πακέτου φωνής είναι πρακτικά απίθανο να συμβεί στο κανάλι μας αφού όπως ήδη έχουμε αναφέρει δίνεται απόλυτη προτεραιότητα στα πακέτα των πηγών φωνής έναντι του video. Εάν είχαμε απώλεια πακέτων φωνής, αυτή θα οφειλόταν σε συγκρούσεις πακέτων δύο ή περισσοτέρων διαφορετικών πηγών φωνής πράγμα που καθιστούσε αδύνατη την εισαγωγή στο κανάλι και πηγών video (ακόμη πιο ευαίσθητων στις απώλειες και ούτως ή άλλως πιο «κακομεταχειρισμένων» από το πρωτόκολλό μας). Ωστόσο είναι χρήσιμο να καθοριστεί ένα τέτοιο άνω γιατί μας βοηθάει στον προσδιορισμό και του equivalent bandwidth του καναλιού όπως θα δούμε στην παράγραφο 4.4.

4.3 Πηγές Video

Στο κεφάλαιο 2 παρουσιάσαμε αναλυτικά το μαθηματικό μοντέλο που χρησιμοποιήσαμε για την προσομοίωση των πηγών video και δώσαμε στοιχεία για τα φυσικά χαρακτηριστικά των πηγών που επιθυμούμε να αναπαραστήσουμε. Τονίστηκε επίσης η γενικότητα με την οποία το μοντέλο μας ανταποκρίνεται στους διαφορετικούς τρόπους κωδικοποίησης μιας ροής video. Υποθέσαμε εικόνα μικρής ανάλυσης με στόχο την επίτευξη ελάχιστου ρυθμού γέννησης δεδομένων 9600 bps, μέγιστου ρυθμού 169200 bps και μέσου ρυθμού 62400 bps. Όσον αφορά την τυπική απόκλιση (standard deviation) σ του ρυθμού γέννησης δεδομένων, αυτή προσδιορίστηκε αναλυτικά από το Μαρκοβιανό μοντέλο ίση με 30580 bits ανά sec. Η προσομοίωση του Μαρκοβιανού μοντέλου των πηγών video προσέγγισε με ακρίβεια τον προδιαγεγραμμένο μέσο ρυθμό γέννησης δεδομένων, δηλαδή

υπήρξε καλό «model fitting». Σαν άνω όριο απώλειας πακέτων πηγών video θεωρείται η απώλεια ενός πακέτου στα 10000 που θα γεννηθούν.

4.4 Παρουσίαση και Σχολιασμός των Αποτελεσμάτων

Η χρησιμοποίηση δύο διαφορετικών μετρικών με βάση τις οποίες προτεραιοποιούνται οι μεταδόσεις των πηγών video (Packet Loss Rate και Glitch Rate), πρακτικά δεν έδειξε καμία απολύτως διαφοροποίηση στα αποτελέσματα.

Ωστόσο η χρησιμοποίηση δύο διαφορετικών μεθόδων αντιμετώπισης των περιπτώσεων συγκρούσεων έδωσε διαφορετικά αποτελέσματα. Στο σχήμα 4.2α φαίνονται τα αποτελέσματα των απωλειών των video πακέτων σε συνάρτηση με τον αριθμό των τερματικών φωνής που βρίσκονται εκείνη την στιγμή στην μικροκυψέλη για την περίπτωση που δεν επιχειρείται επαναμετάδοση των συγκρουόμενων πακέτων video. Η παράμετρος είναι το πλήθος των video τερματικών. Τα αποτελέσματα παρουσιάζονται σε λογαριθμική κλίμακα. Από τα αποτελέσματα του σχήματος συμπεραίνουμε ότι οι απώλειες πακέτων των video πηγών αυξάνονται αργά σε σχέση με τον αριθμό των πηγών φωνής για έναν σταθερό αριθμό από video πηγές. Αντίθετα οι απώλειες αυτές αυξάνονται με πολύ πιο γρήγορο ρυθμό όταν προσθέσουμε μία ή περισσότερες πηγές video για ένα σταθερό αριθμό από πηγές φωνής. Αυτό βέβαια είναι αναμενόμενο, εφ' όσον οι πηγές video όχι μόνο γεννούν κατά μέσο όρο περισσότερα πακέτα ανά μονάδα χρόνου αλλά είναι και πολύ περισσότερο «εκρηκτικές».

Στο σχήμα 4.2β παρουσιάζονται τα αποτελέσματα των απωλειών video πακέτων χρησιμοποιώντας αυτή τη φορά επαναμετάδοση των συγκρουόμενων πακέτων video. Οι διαφορές ανάμεσα στα δύο σενάρια είναι σαφείς. Για παράδειγμα όταν στο κανάλι μας πολυπλέκονται 5 πηγές video και 58 πηγές φωνής, δηλαδή όταν το φορτίο του καναλιού είναι σχετικά χαμηλό, τότε η λογική της επαναμετάδοσης των video πακέτων φαίνεται ότι αποδίδει σχετικά

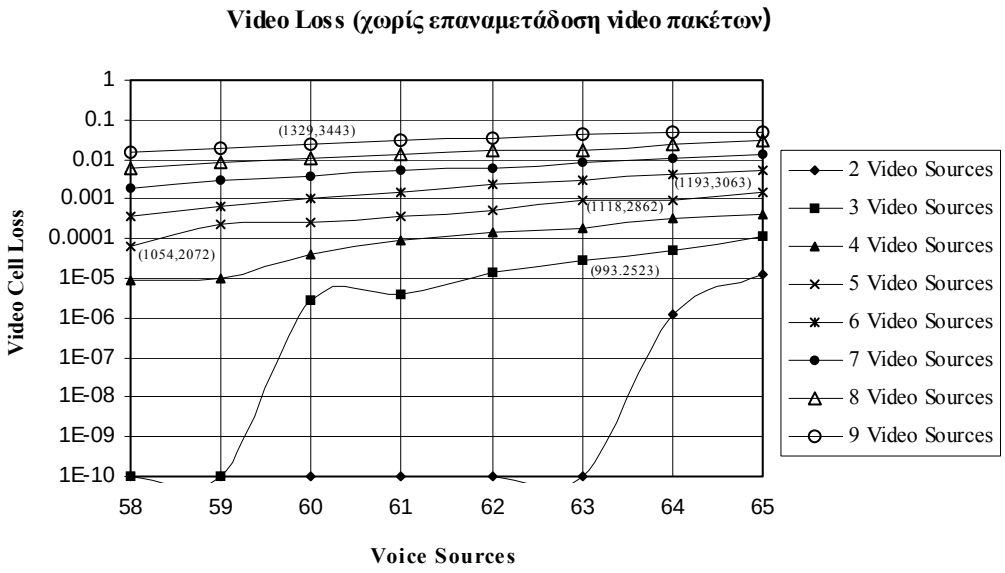
καλά. Έτσι ενώ χωρίς επαναμετάδοση το ποσοστό του Video Loss είναι της τάξης του 10^{-4}, με την επαναμετάδοση το ποσοστό πέφτει σε 10^{-5}.

Παρατηρούμε όμως ότι η απόδοση της δεύτερης προσέγγισης πλησιάζει την πρώτη όσο πιο πολύ αυξάνει ο φόρτος του καναλιού. Έτσι όταν τα τερματικά φωνής αυξηθούν σε 63 τότε η διαφορά απόδοσης των δύο προσεγγίσεων παύει να είναι αισθητή και το ποσοστό απώλειας video πακέτων για την μεν πρώτη είναι ακριβώς 10^{-4} ενώ για την δεύτερη ελαφρά μικρότερο. Στην περίπτωση μάλιστα που προσθέσουμε μία ακόμα πηγή video στην μικροκυψέλη τότε τα αποτελέσματα των δύο προσεγγίσεων πρακτικά συμπίπτουν. Αυτή η συμπεριφορά είναι αναμενόμενη, αφού η επανάληψη μιας μη επιτυχούς μετάδοσης πακέτου προϋποθέτει ότι στα 2 αμέσως επόμενα channel frames θα βρεθεί διαθέσιμο κενό slot.

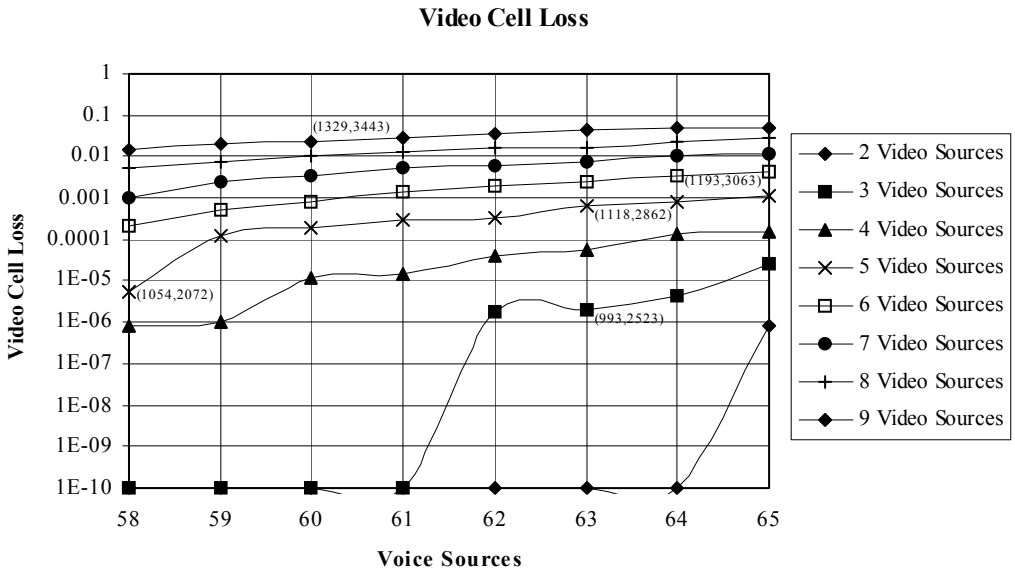
Σε ορισμένα σημεία των γραφικών παραστάσεων (μεταξύ των οποίων και τα παραπάνω) έχει τοποθετηθεί μέσα σε παρένθεση σε μορφή (x,y) το μέσο (average) και το μέγιστο (peak) φορτίο του καναλιού σε kbps, όπως αυτά προκύπτουν αν θεωρήσουμε ότι οι πηγές φωνής και video παράγουν πακέτα είτε σύμφωνα με τον μέσο ρυθμό γέννησης πληροφορίας είτε με τον μέγιστο ρυθμό τους. Δηλαδή x=Vo*32*0,4+Vi*62,4 και y=Vo*32+Vi*169,2 (όπου Vi, Vo ο αριθμός των πηγών φωνής και video, αντίστοιχα).

.

Σχ. 4.1α


Video Loss (χωρίς επαναμετάδοση video πακέτων)
Video Cell Loss
1
0.1
0.01
0.001
0.0001
1E-05
1E-06
1E-07
1E-08
1E-09
1E-10
58
59
60
61
62
63
64
65
Voice Sources
(1329,3443)
(1193,3063)
(1118,2862)
(1054,2072)
(993,2523)
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

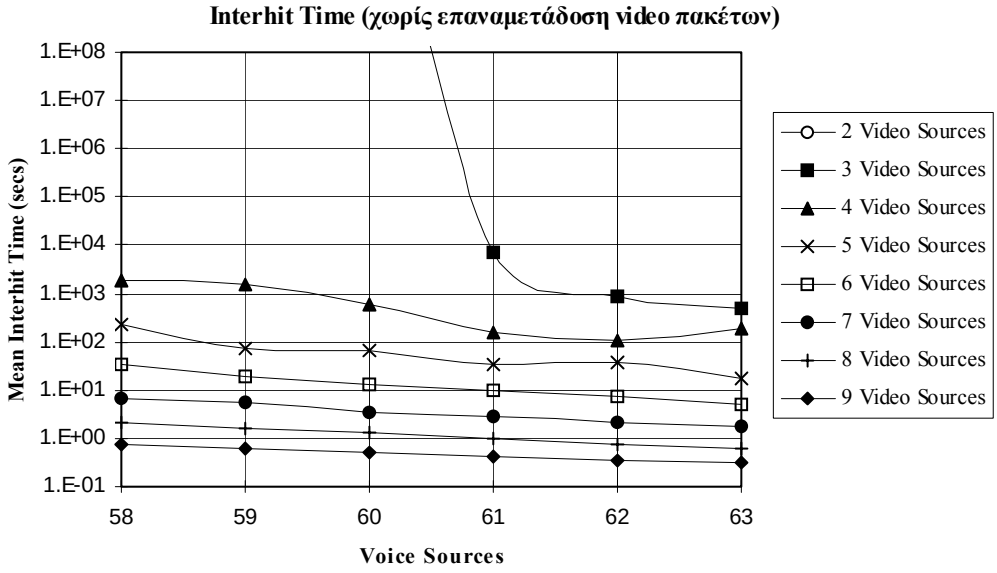
Σχ. 4.1β


Video Cell Loss
Video Cell Loss
1
0.1
0.01
0.001
0.0001
1E-05
1E-06
1E-07
1E-08
1E-09
1E-10
58
59
60
61
62
63
64
65
Voice Sources
(1329,3443)
(1193,3063)
(1118,2862)
(1054,2072)
(993,2523)
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

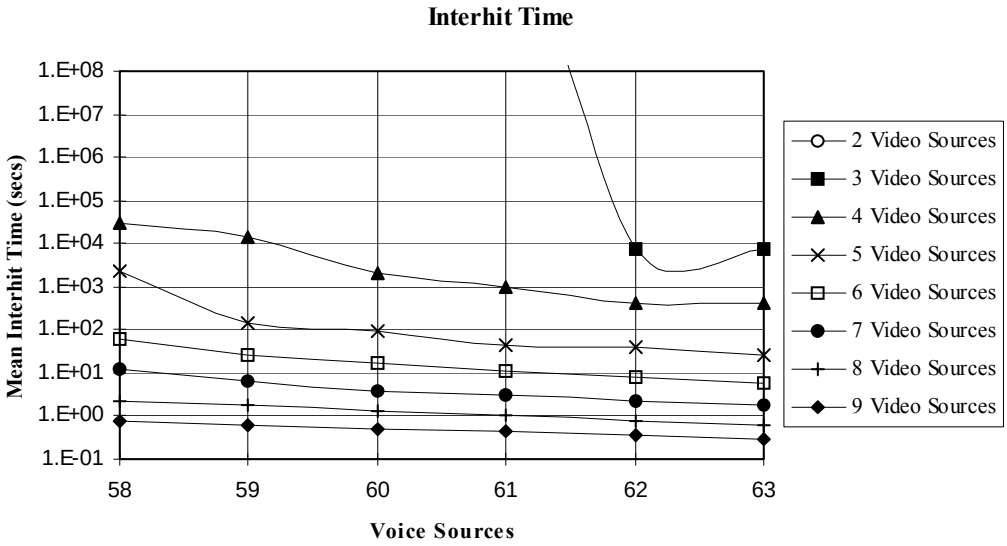
Μία επίσης πολύ σημαντική μετρική που εξετάζουμε αφορά τις πηγές video. Η μετρική αυτή είναι η μέση χρονική διάρκεια ανάμεσα σε δύο διαδοχικές απώλειες πακέτων από μία πηγή video (mean video inter-hit time). Η μετρική αυτή είναι αντιπροσωπευτική μετρική του QoS που απολαμβάνουν οι εφαρμογές video, καθώς μετρά πόσο συχνά ο τελικός χρήστης της υπηρεσίας video είναι πιθανό να αντιληφθεί ασυνέχειες (glitches) στην εικόνα που λαμβάνει στην κινητή τερματική του συσκευή. Εδώ πρέπει να τονίσουμε ότι στην περίπτωση που η μεμονωμένη απώλεια (hit) μιας πηγής video περιορίζεται σε λίγα (ή συχνά και ένα) video πακέτα και όχι σε ένα μεγάλο αριθμό, τότε είναι εξαιρετικά πιθανό ο τελικός χρήστης να μην αντιληφθεί μείωση της ποιότητας του video στην οθόνη του. Το γεγονός αυτό είναι αναμενόμενο εφ' όσον το video που υποθέσαμε αντιπροσωπεύει αλληλουχία εικόνων με ρυθμό πάνω από 27 καρέ ανά δευτερόλεπτο, ενώ το κάθε πακέτο πληροφορίας αναπαριστά μόνον ένα μικρό κλάσμα ενός τέτοιου καρέ.

Και σε αυτήν την περίπτωση παραθέτουμε τα αποτελέσματα και των δύο διαφορετικών προσεγγίσεων του πρωτοκόλλου. Στο σχήμα 4.2α φαίνεται ο μέσος χρόνος ανάμεσα σε δύο διαδοχικές απώλειες video πακέτων στην περίπτωση που δεν επιχειρούμε επαναμετάδοση των συγκρουόμενων video πακέτων, ενώ στο 4.2β τα αντίστοιχα αποτελέσματα με επαναμετάδοση.

Σχ 4.2α


Interhit Time (χωρίς επαναμετάδοση video πακέτων)
Mean Interhit Time (secs)
1.E+08
1.E+07
1.E+06
1.E+05
1.E+04
1.E+03
1.E+02
1.E+01
1.E+00
1.E-01
58
59
60
61
62
63
Voice Sources
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

Σχ 4.2β


Interhit Time
Mean Interhit Time (secs)
1.E+08
1.E+07
1.E+06
1.E+05
1.E+04
1.E+03
1.E+02
1.E+01
1.E+00
1.E-01
58
59
60
61
62
63
Voice Sources
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

Παρατηρούμε ότι τα αποτελέσματα και εδώ έχουν ανάλογη συμπεριφορά με τα αποτελέσματα του Video Loss. Έτσι όταν για παράδειγμα στο κανάλι πολυπλέκονται 3 πηγές video και 62 πηγές φωνής για την μεν περίπτωση της μη επαναμετάδοσης ο μέσος χρόνος ανάμεσα σε δύο διαδοχικές απώλειες πακέτων από την ίδια πηγή video είναι της τάξης των 1000 δευτερολέπτων, ενώ αν επιχειρούμε επαναμετάδοση ο χρόνος είναι της τάξης των 10000 δευτερολέπτων. Έτσι φαίνεται ότι ο δεύτερος αλγόριθμός επιτυγχάνει την εισαγωγή στο κανάλι άλλης μιας πηγής video χωρίς να επηρεάσει καθόλου το συνολικό QoS. Όταν όμως προστεθούν μερικές ακόμα πηγές video έτσι ώστε ο συνολικός αριθμός τους να γίνει για παράδειγμα 8, χωρίς να μεταβληθεί ο αριθμός των πηγών φωνής τότε και στις δύο περιπτώσεις ο μέσος χρόνος δύο διαδοχικών απωλειών πακέτων από τις πηγές αυτές μειώνεται αισθητά σε λιγότερο από 1 δευτερόλεπτο. Η τακτική της επαναμετάδοσης φαίνεται να αποδίδει μόνο στην περίπτωση που το κανάλι δεν είναι υπερβολικά φορτωμένο για τον ίδιο ακριβώς λόγο που αναπτύξαμε προηγουμένως.

Είναι συνεπώς ενδιαφέρον να βρούμε από πιο σημείο και μετά παύει να είναι αποτελεσματική η τεχνική της επαναμετάδοσης. Στον πίνακα 4.1 παρουσιάζονται οι λόγοι των μέσων χρόνων ανάμεσα σε δύο διαδοχικές απώλειες πακέτων για όλους τους συνδυασμούς πηγών φωνής και video. Οι λόγοι αυτοί προκύπτουν αν διαιρέσουμε τον μέσο χρόνο δύο διαδοχικών hits για τον αλγόριθμο επαναμετάδοσης με τον μέσο χρόνο του αλγορίθμου μη επαναμετάδοσης. Ουσιαστικά διαιρούμε δηλαδή τα αποτελέσματα των σχημάτων 4.2α και 4.2β.

Πίνακας 4.1

	58 Voice Sources	59 Voice Sources	60 Voice Sources	61 Voice Sources	62 Voice Sources	63 Voice Sources
2 Video Sources	≈0	≈0	≈0	≈0	≈0	≈0
3 Video Sources	≈0	≈0	≈0	5.56E-09	0.115385	0.068182
4 Video Sources	0.066667	0.105263	0.285714	0.16129	0.259542	0.461538
5 Video Sources	0.090909	0.544699	0.686025	0.791271	0.910359	0.711445
6 Video Sources	0.54386	0.743955	0.7461	0.913888	0.881636	0.878356
7 Video Sources	0.550203	0.845772	0.946484	0.982061	0.960964	0.927693
8 Video Sources	0.87771	0.937729	0.956703	0.944253	0.965183	0.992724
9 Video Sources	0.975712	0.973065	0.986489	0.971903	0.989906	1.006185

Παρατηρούμε ότι υπάρχουν εξαιρετικά απότομες μεταβολές ανάμεσα σε διαφορετικούς διαδοχικούς συνδυασμούς πλήθους πηγών φωνής και video όσον αφορά τους παραπάνω λόγους. Έτσι, αν για παράδειγμα στο κανάλι μας υπάρχουν 58 πηγές φωνής και 5 πηγές video η τεχνική της επαναμετάδοσης φαίνεται να δίνει 10 φορές καλύτερα αποτελέσματα από την μη επαναμετάδοση ενώ αν προστεθεί μία ακόμη πηγή video τα αποτελέσματα γίνονται μόλις 2 φορές καλύτερα. Αν και ακόμα δεν έχουμε παρουσιάσει τα αποτελέσματα που αφορούν το throughput καναλιού μας (θα γίνει στην επόμενη παράγραφο) αξίζει εδώ να αναφερθούμε σε αυτήν την παράμετρο σε σχέση με τον πιο πάνω πίνακα. Υπάρχουν κάποιοι οριακοί συνδυασμοί πλήθους πηγών φωνής και video από τους οποίους παύει ο αλγόριθμος της επαναμετάδοσης να αποδίδει. Ο λόγος δηλαδή που προαναφέραμε αρχίζει να πλησιάζει την μονάδα. Τέτοιοι συνδυασμοί είναι οι εξής: (58 Voice Sources, 8 Video Sources), (59 Voice Sources, 7 Video Sources), (60 Voice Sources, 7 Video Sources), (61 Voice Sources, 6 Video Sources), (62 Voice Sources, 6 Video Sources) κ.ο.κ.. Προτρέχοντας στα σχήματα 4.4 όπου παρουσιάζονται τα αποτελέσματα της χρησιμοποιήσης του καναλιού θα παρατηρήσουμε ότι όλοι αυτοί οι συνδυασμοί αντιστοιχούν σε Channel Throughput από 0.8 έως

0.85. Συνεπώς μόλις η χρησιμοποίηση του καναλιού αγγίξει το 80% του συνολικού η επαναμετάδοση παύει να ωφελεί σημαντικά τις video πηγές.

4.5 Χρησιμοποίηση καναλιού

Στο σημείο αυτό αξίζει να δούμε ορισμένους θεωρητικούς υπολογισμούς πριν προχωρήσουμε στην παράθεση των πειραματικών αποτελεσμάτων.

Σαν *ισοδύναμη χωρητικότητα* (equivalent bandwidth) μιας επαλληλίας πηγών πληροφορίας, ορίζουμε τη χωρητικότητα (bandwidth) που προκύπτει έτσι ώστε η πιθανότητα απώλειας πληροφορίας (σημ. δηλαδή η πιθανότητα όλες οι πηγές μαζί να παράγουν πληροφορία με ρυθμό μεγαλύτερο από την χωρητικότητα (bandwidth) του διαύλου), να παραμένει κάτω από μία μέγιστη επιθυμητή τιμή.

Μία από τις μεθόδους προσδιορισμού του ισοδύναμου bandwith είναι η Gaussian approximation που αναφέρεται σε πληροφορία που γεννιέται από On Off πηγές με εκθετικά κατανεμημένους χρόνους παραμονής στην κάθε κατάσταση και ανεξάρτητους μεταξύ τους. Κάθε πηγή i χαρακτηρίζεται από το μέσο ρυθμό γέννησης δεδομένων m_i και τυπική απόκλιση σ_i. Αν A είναι η τυχαία μεταβλητή που συμβολίζει το συνολικό ρυθμό γέννησης πληροφορίας της υπέρθεσης των πηγών, τότε ζητείται να προσδιοριστεί η τιμή του c_0 έτσι ώστε $Pr\{A> c_0) < \in$, όπου $\in$ η επιθυμητή τιμή της μέγιστης πιθανότητας απώλειας πληροφορίας. Όπως μπορεί να δειχτεί [18]

$$c_0 \approx m+\alpha\sigma$$

όπου με m και σ συμβολίζουμε την μέση τιμή και την τυπική απόκλιση (standard deviation) του συνολικού ρυθμού γέννησης πληροφορίας, αντίστοιχα.

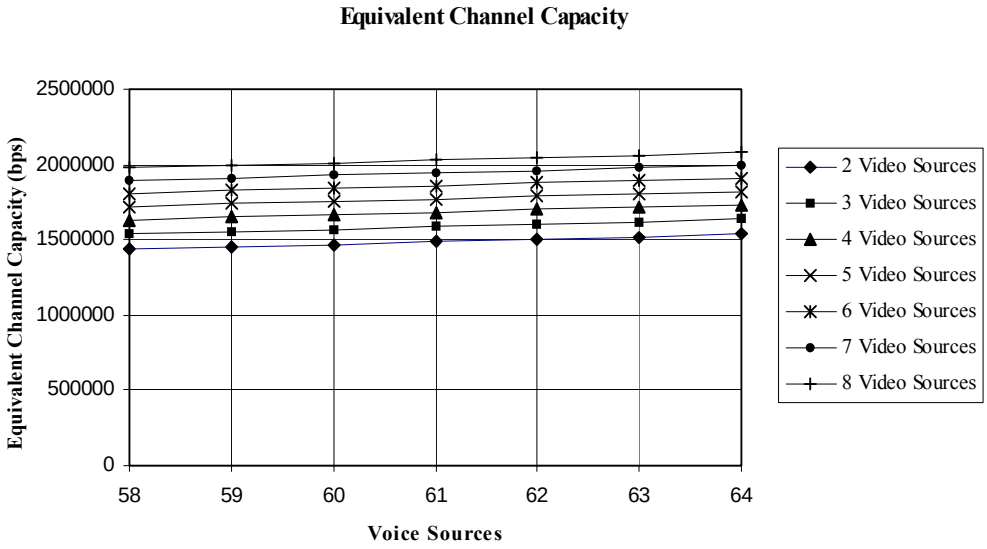
Το α προκύπτει σαν η αντίστροφη της Gaussian κατανομής με τιμή:

$$\alpha=\sqrt{(2\ln(1/\epsilon)-\ln 2\pi)}$$

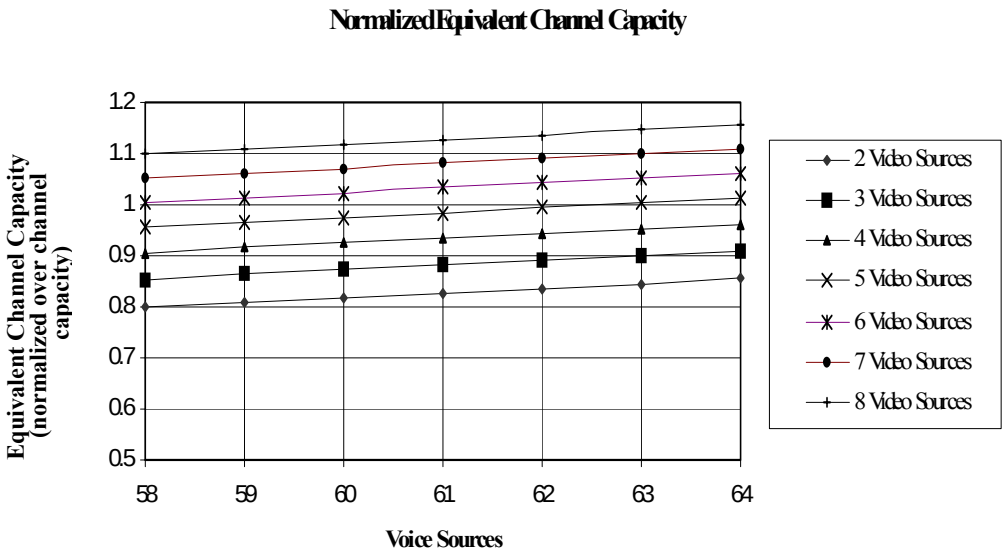
Ο παραπάνω προσεγγιστικός υπολογισμός του equivalent bandwidth συνήθως υπερεκτιμά την πραγματική τιμή, πράγμα που φαίνεται και από την σύγκριση με τα αντίστοιχα πειραματικά αποτελέσματα που παραθέτουμε παρακάτω. Μας παρέχει όμως ένα γρήγορο και εύκολο τρόπο για να υπολογίζουμε το ισοδύναμο bandwidth που απαιτεί η πολυπλεξία των πηγών μας. Ο ίδιος υπολογισμός μπορεί να χρησιμοποιηθεί και από τον χρονοπρογραμματιστή (scheduler) του ΣΒ, έτσι ώστε να γίνεται εκτίμηση του κατά πόσο το κανάλι «αντέχει» παραπάνω φορτίο και ανάλογα να επιτρέπεται σε μία νέα συνδιάλεξη να εγκατασταθεί ή όχι (admission control). Το σχήμα 4.3α δείχνει τα αποτελέσματα που προκύπτουν από την εφαρμογή της Gaussian approximation.

Παρατηρούμε ότι για αρκετούς συνδυασμούς οι θεωρητικά υπολογισμένες προσεγγιστικές τιμές υπερβαίνουν το συνολικό bandwidth του καναλιού. Στο σχήμα 4.3β παρουσιάζεται η ισοδύναμη χωρητικότητα του καναλιού σαν κλάσμα της πραγματικής χωρητικότητας του καναλιού μας (1.8 Mbps).

Σχ. 4.3α

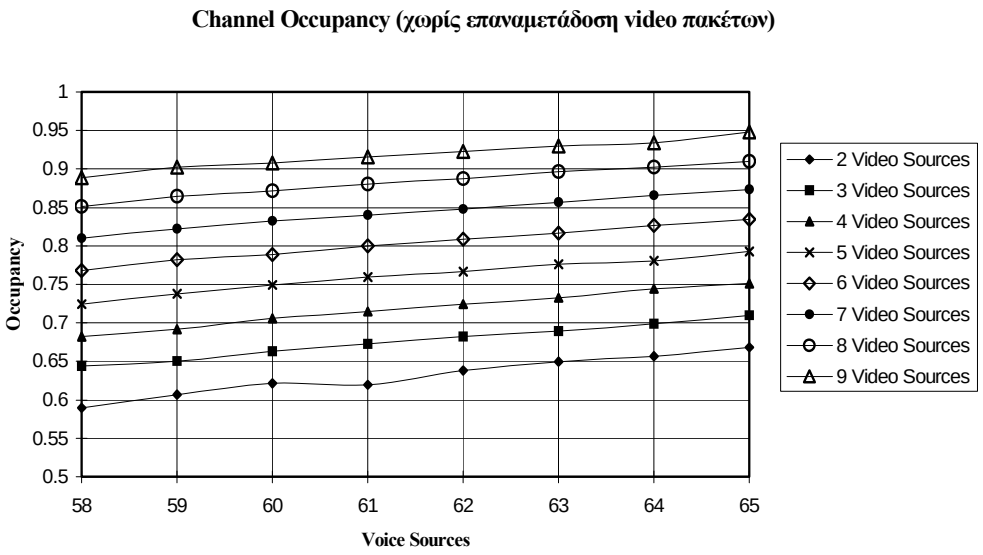


Σχ. 4.3β

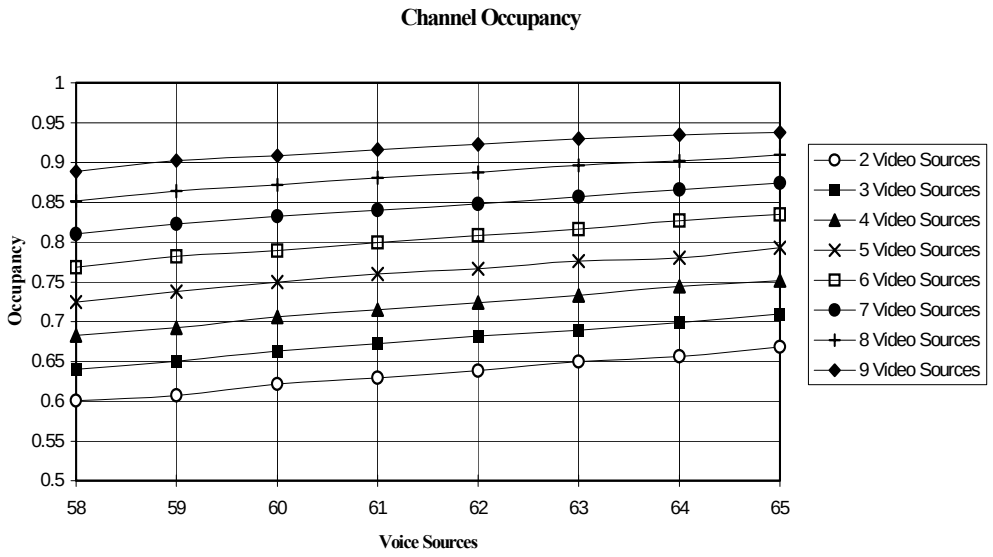


Στα σχήματα 4.4α και 4.4β λοιπόν φαίνεται η πραγματική χρησιμοποίηση του καναλιού για την περίπτωση της μη επαναμετάδοσης και επαναμετάδοσης, αντίστοιχα.

4.4α


Channel Occupancy (χωρίς επαναμετάδοση video πακέτων)
Occupancy
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
58
59
60
61
62
63
64
65
Voice Sources
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

4.4β


Channel Occupancy
Occupancy
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
58
59
60
61
62
63
64
65
Voice Sources
2 Video Sources
3 Video Sources
4 Video Sources
5 Video Sources
6 Video Sources
7 Video Sources
8 Video Sources
9 Video Sources

Εδώ οι διαφορές στα αποτελέσματα των δύο προσεγγίσεων μοιάζουν αμελητέες. Αυτό είναι αναμενόμενο καθώς για τις μικρές σχετικά χρησιμοποιήσεις καναλιού οι αποτυχημένες μεταδόσεις πακέτου video είναι σπάνιες οπότε ακόμα και αν αυτές όλες τους θεωρηθούν επιτυχημένες ελάχιστα θα αλλάξει ο όγκος της πληροφορίας που διακινείται πάνω στο κανάλι.

Για τις μεγάλες χρησιμοποιήσεις καναλιού (εκεί δηλαδή που ένας μεγάλος αριθμός από πηγές φωνής και video πολυπλέκονται) τα ποσοστά αποτυχημένων μεταδόσεων δεν μειώνονται δραστικά αν επιχειρείται επαναμετάδοση (βλέπε τα σχήματα 4.1α και 4.1β) και κατά συνέπεια οι διαφορές και όσον αφορά την χρησιμοποίηση του καναλιού ανάμεσα στις δύο προσεγγίσεις είναι αμελητέες.

Παρατηρώντας λοιπόν ότι και στις δύο περιπτώσεις δεν υπάρχει ουσιαστική μεταβολή μπορούμε επιλέγοντας τα αποτελέσματα μιας εξ' αυτών να κάνουμε την σύγκριση ανάμεσα στις θεωρητικά υπολογισμένες προσεγγιστικές τιμές και σε αυτές που προέκυψαν από την προσομοίωση.

Η παραπάνω σύγκριση επιβεβαιώνει την παρατήρηση που είχαμε κάνει προηγουμένως σχετικά με την υπερεκτίμηση του απαιτούμενου (equivalent) bandwidth που κάνει ο προσεγγιστικός υπολογισμός του.

4.6 Γενικά Συμπεράσματα

Προτείναμε και υλοποιήσαμε έναν αλγόριθμο χρονοδρομολόγησης (scheduling) για την αποτελεσματική πολυπλεξία πηγών φωνής και video χαμηλής κινητικότητας πάνω σε ένα ασύρματο φυσικό δίαυλο χωρητικότητας 1,8 Mbps. Το δίκτυο μας αναφέρεται σε ασύρματο μικροκυψελικό δίκτυο 3ης γενιάς παρέχοντας μας ένα καθαρό από λάθη περιβάλλον.

Ο αλγόριθμος εκμεταλλεύεται τα φυσικά χαρακτηριστικά των ροών video και φωνής επιτυγχάνοντας υψηλά ποσοστά χρησιμοποίησης του

καναλιού, διασφαλίζοντας ταυτόχρονα τις παραμέτρους QoS που απαιτούν οι πηγές.

Προσπαθώντας να επιτύχουμε μία όσο το δυνατόν πιο ρεαλιστική αναπαράσταση των φυσικών χαρακτηριστικών του video χρησιμοποιήσαμε ένα Μαρκοβιανό μοντέλο που αποδείχθηκε εξαιρετικά ευέλικτο και αποτελεσματικό κατά την προσομοίωση, αναπαριστώντας σε ικανοποιητικό βαθμό την εκρηκτικότητα της πραγματικής ροής.

Θεωρώντας ότι η φωνή είναι σαν πληροφορία πιο κρίσιμη από το video σε ένα ασύρματο δίκτυο σαν αυτό που θεωρήσαμε, ο αλγόριθμος χρονοπρογραμματισμού της δίνει απόλυτη προτεραιότητα. Ο αλγόριθμος δεν επιτρέπει partial allocation των πόρων του καναλιού για την μετάδοση video πακέτων, κι αυτό γιατί η υψηλή συμπίεση των κωδικοποιήσεων του video καθιστά δύσκολη την αποσυμπίεση διακεκομμένων ακολουθιών πακέτων.

Ανάμεσα στις video πηγές προτεραιότητα παρέχεται σε εκείνες που κατά το παρελθόν έχουν υποστεί τις μεγαλύτερες απώλειες σε πακέτα. Στην πιο απλή του περίπτωση ο αλγόριθμος δεν επιχειρεί την επαναμετάδοση των πακέτων video για τα οποία δεν μπορεί να χρονοπρογραμμαιστεί άμεση μετάδοση. Στην βελτιωμένη του μορφή ο αλγόριθμος επιχειρεί επαναμεταδόσεις αυτών των πακέτων επιτυγχάνοντας θετικά αποτελέσματα ειδικά στις περιπτώσεις χαμηλού και μέσου φορτιού του καναλιού (για channel throughput μικρότερο του 80%). Στα τελικά αποτελέσματα παρουσιάζονται αναλυτικά όλα εκείνα τα στοιχεία που θα επιτρέψουν στον αναγνώστη να κάνει μια σφαιρική εκτίμηση των δυνατοτήτων του αλγορίθμου.

4.7 Μελλοντικές Επεκτάσεις

Υπάρχουν ενδιαφέροντα σημεία προς τα οποία αξίζει να στραφεί μία μελλοντική επέκταση των δυνατοτήτων του αλγορίθμου.

Αναφέρουμε τα τρία σημαντικότερα:

- Να μην παρέχεται απόλυτη προτεραιότητα στα πακέτων φωνής. Έτσι να γίνει εκμετάλλευση του γεγονότος ότι το άνω όριο καθυστέρησης για τα πακέτα φωνής είναι 24 msecs (2 CFs) και έτσι να επιτευχθούν ακόμα καλύτερα αποτελέσματα όσον αφορά το Video Loss.
- Οι αιτήσεις των πηγών φωνής και video σε slots να φτάνουν στον σταθμό βάσης όχι μόνο στην αρχή κάθε channel frame αλλά και κατά την διάρκειά του, αναγκάζοντας τον scheduler να προσπαθεί να χρονοπρογραμματίσει μεταδόσεις για την εξυπηρέτηση των συγκεκριμένων αιτήσεων και στο τρέχον channel frame.
- Μελέτη της επίδρασης, στην απόδοση του συστήματος, του γεγονότος ότι οι αιτήσεις των πηγών φωνής και video προκειμένου να φτάσουν στον σταθμό βάσης πρέπει να μεταδοθούν επιτυχώς πάνω σε ένα βοηθητικό κανάλι ελέγχου (control channel), όπου υπόκεινται σε συγκρούσεις με αποτέλεσμα κάποιες από αυτές να καθυστερήσουν.

Α. Βιβλιογραφία

[1] V.H. Mac Donald, "The Cellular Concept", *Bell Sys. Tech. Journal*, Vol. 58, No. 1, Jan. 1979, pp. 15-41

[2] D.C. Cox , "Wireless Personal Communications: What is it?", *IEEE Pers. Commun*. Vol. 2, Apr. 1995, pp. 20-35

[3] V.O.K. Li & X. Qui, "Personal Communication Systems", *Proc. IEEE*, Vol. 83, No. 9, Sept. 1995, pp. 1210-1243.

[4] M. Buhrmann, "D-AMPS: The Quiet Global Success Story", *International Edition Telecommunications Mag.,* Vol. 31, October 1997, pp. 48-60.

[5] J.E. Padgett, C. G. Gunther, T. Hattori, "Overview of Wireless Personal Communications", *IEEE Commun. Magaz.,* Jan. 1995, pp. 28-41.

[6] M. Rahnema, "Overview of the GSM System and Protocol Architecture", *IEEE Commun. Mag*., Apr. 1993, pp. 92-100

[7] P. A. Ramsdale, "Personal Communications in the UK-Implementation of PCN using DCS 1800", *Int. J. Wireless Info. Networks,* Vol. 1, No. 1, Jan. 1994, pp. 29-36

[8] A. J. Viterbi, "Spread Spectrum Communications – Myths an Realities", *IEEE Commun. Mag*., May 1979, pp. 11-18.

[9] C. C. Evci, "Race-UMTS for Third Generation Wireless Networks" *Ann Telecommun*., Vol. 47, No 7-8, Jul.-Aug. 1992, pp. 267-281.

[10] L. J. Greenstein & R.D. Gitlin, "A Microcell/Macrocell Architecture for Low and High Mobility Usres*", IEEE JSAC*, Vol. 11, No. 6, Aug. 1993, pp. 885-891.

[11] A. Gil, Av. Freedman, M. Levin, "Quantum Leaps for Frequency Hopping", *International Edition Telecommunications Mag.,* Vol. 31, October 1997, pp. 105-106.

[12] S. Nanda, D. J. Goodman and U. Timor, "Performance of RPMA: A Packet Voice Protocol for Cellular Sytems*", IEEE Trans. Weh. Technology*, Vol. 40, Aug. 1991, pp. 584-598.

[13] P. T. Brady, "A technique for Investigating On-Off Patterns of Speech", *Bell Sys. Tech. Journal*, Jan. 1965.

[14] Basil Maglaris, Dimitris Anastassiou, Prodip Sen, Gunnar Karlsson & John Robbins, "Performance Models of Statistical Multiplexing in Packet Video Communications", *IEEE Transactions on Communications*, Vol. 36, No. 7, July 1998.

[15] J. M. De Vile, "A Reservation Based Multiple Access Scheme for a Future Universal Mobile Telecommunications System*", Proc. Of the 7th IEEE Conf. On Mobile and Personal Commun.*, Brighton, UK, Dec. 1993, pp. 210-215.

[16] J. Dunlop, J. Irvine, D. Robertson & P. Cosimini, "Performance of a Statistically Multiplexed Access Mechanism for a TDMA Radio Interface", *IEEE Pers. Commun*., Vol. 2, No.3, Jun. 1995, pp. 56-64.

[17] A.C. Cleary, "Voice Data Access Integration in Third Generation Microcellular Wireless Networks: Design an Performance Evaluation", *Ph.D. Dissertation*, CIS Dept Univ. of Delaware, 1997.

[18] R. Onvural, "Asynchronous Transfer Mode Networks: Performance Issues", Appendix B pp. 479-481.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <conio.h>
#define Total_Frames 100000
#define WARMUP 5000
#define Voice_Source_N  63
#define Video_Source_N  9


struct Source_Model
{
 int CURRENT_State;    /* 1=ACTIVE STATE, 0=SILENT STATE FOR VOICE AND
0,1,2,3,4,5,6 FOR VIDEO*/
 int OLD_State;
 int INIT_DIGIT;         /* A digit between 0 and 2 which helps us randomize the
video transitions over time */
 long LOST_CELLS_C;           /* Lost cells counter */
 long TOTAL_CELLS_C;         /* Total Cells Counter */
 /*long R_HICKUP_SPACE;       /* Recent Space between two Hickups */
 long HICKUP_C;             /* Number of Hickups */
 } ;


struct Source_Model Voice_Source_Array[Voice_Source_N];  /* ALL the Voice
sources */
struct Source_Model Video_Source_Array[Video_Source_N];  /* ALL the Video
sources */

float A_TRANSITION_RATES[7],B_TRANSITION_RATES[7];
FILE *fptr;
long Frame_Counter,longtmp,HickTotal;
float randomvar;
int FRAME[50],CELL_NUM,Voice_Counter,Video_Counter,tmp,i,TMPVC,repeat;    /* IF
FRAME[x]=0 then empty slot, =1 voice slot */
int VoiceGen(int);                      /* Voice Generator Function */
       /* IF FRAME[x]=2 video slot, =3 collision */
int VideoGen(int);           /* Video Generator Function */
long TOTALVCC,TOTALViCC,TOTALVHC,TOTALViHC;
struct Source_Model  *rand_sort(struct Source_Model  *source_arr);   /* Shufling
Function */


int partition(int lb,int ub )
{
  int up;
  int down;
  int j;
  struct Source_Model a,temp;

  a.LOST_CELLS_C=Video_Source_Array[lb].LOST_CELLS_C;
  a.CURRENT_State=Video_Source_Array[lb].CURRENT_State;
  a.OLD_State=Video_Source_Array[lb].OLD_State;
  a.INIT_DIGIT=Video_Source_Array[lb].INIT_DIGIT;
  a.TOTAL_CELLS_C=Video_Source_Array[lb].TOTAL_CELLS_C;
  a.HICKUP_C=Video_Source_Array[lb].HICKUP_C;
  up=ub;
  down=lb;
        while ( down < up )
            {
```

```
            while ( ((Video_Source_Array[down].LOST_CELLS_C) <= a.LOST_CELLS_C)
&& (down< ub) )
             {
                  down=down + 1;
                               }

            while ( (Video_Source_Array[up].LOST_CELLS_C) > a.LOST_CELLS_C)
              {
             up=up-1;
                    }

             if ( down < up )
              {

            temp.LOST_CELLS_C=Video_Source_Array[up].LOST_CELLS_C;
            temp.CURRENT_State=Video_Source_Array[up].CURRENT_State;
            temp.OLD_State=Video_Source_Array[up].OLD_State;
            temp.INIT_DIGIT=Video_Source_Array[up].INIT_DIGIT;
            temp.TOTAL_CELLS_C=Video_Source_Array[up].TOTAL_CELLS_C;
            temp.HICKUP_C=Video_Source_Array[up].HICKUP_C;


      Video_Source_Array[up].LOST_CELLS_C=Video_Source_Array[down].LOST_CELLS_C;

      Video_Source_Array[up].CURRENT_State=Video_Source_Array[down].CURRENT_Stat
e;
            Video_Source_Array[up].OLD_State=Video_Source_Array[down].OLD_State;

      Video_Source_Array[up].INIT_DIGIT=Video_Source_Array[down].INIT_DIGIT;

      Video_Source_Array[up].TOTAL_CELLS_C=Video_Source_Array[down].TOTAL_CELLS_
C;
            Video_Source_Array[up].HICKUP_C=Video_Source_Array[down].HICKUP_C;

            Video_Source_Array[down].LOST_CELLS_C=temp.LOST_CELLS_C;
            Video_Source_Array[down].CURRENT_State=temp.CURRENT_State;
            Video_Source_Array[down].OLD_State=temp.OLD_State;
            Video_Source_Array[down].INIT_DIGIT=temp.INIT_DIGIT;
            Video_Source_Array[down].TOTAL_CELLS_C=temp.TOTAL_CELLS_C;
            Video_Source_Array[down].HICKUP_C=temp.HICKUP_C;

                                    }

                                          }


Video_Source_Array[lb].LOST_CELLS_C=Video_Source_Array[up].LOST_CELLS_C;

Video_Source_Array[lb].CURRENT_State=Video_Source_Array[up].CURRENT_State;

Video_Source_Array[lb].OLD_State=Video_Source_Array[up].OLD_State;

Video_Source_Array[lb].INIT_DIGIT=Video_Source_Array[up].INIT_DIGIT;

Video_Source_Array[lb].TOTAL_CELLS_C=Video_Source_Array[up].TOTAL_CELLS_C;

Video_Source_Array[lb].HICKUP_C=Video_Source_Array[up].HICKUP_C;


                  Video_Source_Array[up].LOST_CELLS_C=a.LOST_CELLS_C;
                  Video_Source_Array[up].CURRENT_State=a.CURRENT_State;
```

```
                    Video_Source_Array[up].OLD_State=a.OLD_State;
                    Video_Source_Array[up].INIT_DIGIT=a.INIT_DIGIT;
                    Video_Source_Array[up].TOTAL_CELLS_C=a.TOTAL_CELLS_C;
                    Video_Source_Array[up].HICKUP_C=a.HICKUP_C;



                    j=up;

                    return j;
}

void quick( int lb,int ub  )
  {
      int j;

      if ( lb < ub )
            {
            j=partition(lb,ub);
            quick(lb,j-1);
            quick(j+1,ub);
            }
  }



void main(void)
{
A_TRANSITION_RATES[0]=6.381;
A_TRANSITION_RATES[1]=5.318;
A_TRANSITION_RATES[2]=4.255;
A_TRANSITION_RATES[3]=3.192;
A_TRANSITION_RATES[4]=2.129;
A_TRANSITION_RATES[5]=1.066;
A_TRANSITION_RATES[6]=0;
B_TRANSITION_RATES[0]=0;
B_TRANSITION_RATES[1]=2.8363;
B_TRANSITION_RATES[2]=5.6726;
B_TRANSITION_RATES[3]=8.5089;
B_TRANSITION_RATES[4]=11.3452;
B_TRANSITION_RATES[5]=14.1815;
B_TRANSITION_RATES[6]=17.0178;

clrscr();fptr = fopen("simulate.txt", "w");
printf("\n\n\n");
printf("\n       ************************************************************");

printf("\n       *       Voice - Video Integration SIMULATION PROGRAM      *");
printf("\n       *   Currently simulating 400x300 head & shoulders video   *");
printf("\n       *                    by Spyros Psychis                    *");
printf("\n       *         Technical University of Crete, June 1997        *");
printf("\n       ************************************************************");
printf("\n"); printf("\n"); printf("\n");

printf("The number of Voice Sources is %d \n",Voice_Source_N);
printf("The number of Video Sources is %d \n",Video_Source_N);

for (repeat=0; repeat<6; repeat++)
{
TOTALVCC=TOTALViCC=TOTALVHC=TOTALViHC=HickTotal=0;randomize();
```

```
for (Voice_Counter=0 ; Voice_Counter<Voice_Source_N ; Voice_Counter++)
 {
  randomvar=random(32000)/31999.0;
  Voice_Source_Array[Voice_Counter].OLD_State=0;
  if (randomvar<0.446202) Voice_Source_Array[Voice_Counter].OLD_State=1;
/* Starting state initilization for all Voice sources */
  Voice_Source_Array[Voice_Counter].INIT_DIGIT=0;   /* This field is useless for
the Voice Sources */
  Voice_Source_Array[Voice_Counter].TOTAL_CELLS_C=0;
  Voice_Source_Array[Voice_Counter].LOST_CELLS_C=0;
  Voice_Source_Array[Voice_Counter].HICKUP_C=0;
 }

for (Video_Counter=0 ; Video_Counter<Video_Source_N ; Video_Counter++)
{

 randomvar=random(32000)/31999.0;                                    /* Video
Generator Initilization */
 if (randomvar<0.14798)
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=0;
 if ((0.14798<=randomvar) && (randomvar<0.48089))
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=1; else
 if ((0.48089<=randomvar) && (randomvar<0.792998))
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=2; else
 if ((0.792998<=randomvar) && (randomvar<0.949068))
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=3; else
 if ((0.949068<=randomvar) && (randomvar<0.992979))
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=4; else
 if ((0.992979<=randomvar) && (randomvar<0.9995712875))
Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=5; else

Video_Source_Array[Video_Counter].CURRENT_State=Video_Source_Array[Video_Counter
].OLD_State=6 ;
 Video_Source_Array[Video_Counter].LOST_CELLS_C=0;
/* Lost Cells counter   */
 Video_Source_Array[Video_Counter].INIT_DIGIT=random(3);
/* FRAME NUMBER initilization for all Video sources */
 Video_Source_Array[Video_Counter].TOTAL_CELLS_C=0;
 Video_Source_Array[Video_Counter].HICKUP_C=0;
}



for (Frame_Counter=0 ; Frame_Counter<WARMUP ; Frame_Counter++) /*  WARMUP PERIOD
*/
{                                                                    /*
SCHEDULER LOOP, Begin of Frame */

      for (Voice_Counter=0 ; Voice_Counter<Voice_Source_N ; Voice_Counter++)
       {

Voice_Source_Array[Voice_Counter].CURRENT_State=VoiceGen(Voice_Source_Array[Voic
e_Counter].OLD_State); /* Call the Voice generator for the specific Voice source
*/
```

```
Voice_Source_Array[Voice_Counter].OLD_State=Voice_Source_Array[Voice_Counter].CU
RRENT_State;
       }


      for (Video_Counter=Video_Source_N-1 ; Video_Counter>=0 ; Video_Counter--)
/* Start the transmition of the Video Sources begining with the one which has
the most losses */
       {
        if (Frame_Counter%3==(Video_Source_Array[Video_Counter].INIT_DIGIT))
/* If its time for the video source to display aq new frame i.e. ~~36 msec=3
channel frames then run the chain simulator */

Video_Source_Array[Video_Counter].CURRENT_State=VideoGen(Video_Source_Array[Vide
o_Counter].OLD_State); /* Call the Video generator for the specific Video source
*/

Video_Source_Array[Video_Counter].OLD_State=Video_Source_Array[Video_Counter].CU
RRENT_State;
       }

}




for (Frame_Counter=0 ; Frame_Counter<Total_Frames ; Frame_Counter++) /*  Frame
counter  */
{                                                                      /*
SCHEDULER LOOP, Begin of Frame */
for (CELL_NUM=0;CELL_NUM<50;CELL_NUM++) FRAME[CELL_NUM]=0;       /* The rest of
the cells are empty so fill them with 0's */
/*fprintf(fptr,"\n The Frame num is %ld ",Frame_Counter);        */
/*rand_sort(&(Voice_Source_Array[0]));*/
/*  Voice Source Array Sufling */
CELL_NUM=0;            /* Cell Counter initilization */
quick(0,Video_Source_N-1);                  /* Sorting of the Video Sources
Array based on the LostCellsCounter */


      for (Voice_Counter=0 ; Voice_Counter<Voice_Source_N ; Voice_Counter++)
       {


Voice_Source_Array[Voice_Counter].CURRENT_State=VoiceGen(Voice_Source_Array[Voic
e_Counter].OLD_State); /* Call the Voice generator for the specific Voice source
*/
       if (Voice_Source_Array[Voice_Counter].CURRENT_State==1)
       {
            if (CELL_NUM<50)
            {
             FRAME[CELL_NUM]=1;
             CELL_NUM++;
            } else Voice_Source_Array[Voice_Counter].LOST_CELLS_C++;
       Voice_Source_Array[Voice_Counter].TOTAL_CELLS_C++;
       }

Voice_Source_Array[Voice_Counter].OLD_State=Voice_Source_Array[Voice_Counter].CU
RRENT_State;
```

```
       }



      for (Video_Counter=Video_Source_N-1 ; Video_Counter>=0 ; Video_Counter--)
/* Start the transmition of the Video Sources begining with the one which has
the most losses */
       {
        if (Frame_Counter%3==(Video_Source_Array[Video_Counter].INIT_DIGIT)){
/* If its time for the video source to display a new frame i.e. ~~36 msec=3
channel frames then run the chain simulator */

Video_Source_Array[Video_Counter].CURRENT_State=VideoGen(Video_Source_Array[Vide
o_Counter].OLD_State); /* Call the Video generator for the specific Video source
*/

        switch (Video_Source_Array[Video_Counter].CURRENT_State)
        {

        case 0:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+1;
            if (CELL_NUM<50)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+1;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 1:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+2;
            if (CELL_NUM<49)
            {
             for (tmp=0; tmp<2; tmp++)
              {
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
              }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+2;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 2:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+2;
            if (CELL_NUM<49)
            {
```

```
             for (tmp=0; tmp<2; tmp++)
              {
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
              }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+2;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 3:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+3;
            if (CELL_NUM<48)
            {
             for (tmp=0; tmp<3; tmp++)
             {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
             }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+3;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 4:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+4;
            if (CELL_NUM<47)
            {
            for (tmp=0; tmp<4; tmp++)
             {
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
             }
        } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+4;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                        }
        break;

        case 5:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+5;
            if (CELL_NUM<46)
            {
            for (tmp=0; tmp<5; tmp++)
             {
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
```

```
             }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+5;
                            Video_Source_Array[Video_Counter].HICKUP_C++;
                           }
            break;

        case 6:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+6;
            if (CELL_NUM<45)
            {
            for (tmp=0; tmp<6; tmp++)
             {
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
             }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+6;
                            Video_Source_Array[Video_Counter].HICKUP_C++;
                           }
            break;

        }

Video_Source_Array[Video_Counter].OLD_State=Video_Source_Array[Video_Counter].CU
RRENT_State;
       } else

       {
       switch (Video_Source_Array[Video_Counter].OLD_State)
        {

        case 0:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+1;
            if (CELL_NUM<50)
            {
             FRAME[CELL_NUM]=2;CELL_NUM++;
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+1;
                            Video_Source_Array[Video_Counter].HICKUP_C++;
                           }
            break;

        case 1:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+2;
            if (CELL_NUM<49)
            {
             for (tmp=0; tmp<2; tmp++)
             {
```

```
              FRAME[CELL_NUM]=2;
              CELL_NUM++;
             }
            } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+2;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 2:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+2;
            if (CELL_NUM<49) {
            for (tmp=0; tmp<2; tmp++)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            }
        } else {
                         Video_Source_Array[Video_Counter].LOST_CELLS_C++;
                         Video_Source_Array[Video_Counter].HICKUP_C++;
                        }
            break;

        case 3:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+3;
            if (CELL_NUM<48) {
            for (tmp=0; tmp<3; tmp++)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            }
        } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+3;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                        }
            break;

        case 4:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+4;
            if (CELL_NUM<47) {
            for (tmp=0; tmp<4; tmp++)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            }
        } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+4;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                        }
```

```
            break;

        case 5:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+5;
            if (CELL_NUM<46) {
            for (tmp=0; tmp<5; tmp++)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            }
        } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+5;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        case 6:

      Video_Source_Array[Video_Counter].TOTAL_CELLS_C=Video_Source_Array[Video_C
ounter].TOTAL_CELLS_C+6;
            if (CELL_NUM<45) {
            for (tmp=0; tmp<6; tmp++)
            {
             FRAME[CELL_NUM]=2;
             CELL_NUM++;
            }
        } else {

Video_Source_Array[Video_Counter].LOST_CELLS_C=Video_Source_Array[Video_Counter]
.LOST_CELLS_C+6;
                          Video_Source_Array[Video_Counter].HICKUP_C++;
                         }
            break;

        }
       }


       }

/*fprintf(fptr, "and has the following cells: ");
for (tmp=0; tmp<50; tmp++) fprintf(fptr," %d",FRAME[tmp]);      */
}


for (Voice_Counter=0; Voice_Counter<Voice_Source_N; Voice_Counter++)
{
 TOTALVCC=TOTALVCC+Voice_Source_Array[Voice_Counter].TOTAL_CELLS_C;
 TOTALVHC=TOTALVHC+Voice_Source_Array[Voice_Counter].LOST_CELLS_C;
/* fprintf(fptr,"\nVOICE_TOTAL_CELLS[%d]=%ld
VOICE_LOST_CELLS[%d]=%ld",Voice_Counter,Voice_Source_Array[Voice_Counter].TOTAL_
CELLS_C,Voice_Counter,Voice_Source_Array[Voice_Counter].LOST_CELLS_C);  */
}
for (Video_Counter=0; Video_Counter<Video_Source_N; Video_Counter++)
{
 TOTALViCC=TOTALViCC+Video_Source_Array[Video_Counter].TOTAL_CELLS_C;
 TOTALViHC=TOTALViHC+Video_Source_Array[Video_Counter].LOST_CELLS_C;
```

```
 fprintf(fptr,"\nVIDEO_TOTAL_CELLS[%d]=%ld VIDEO_LOST_CELLS[%d]=%ld
VIDEO_HICKUP[%d]=%ld",Video_Counter,Video_Source_Array[Video_Counter].TOTAL_CELL
S_C,Video_Counter,Video_Source_Array[Video_Counter].LOST_CELLS_C,Video_Counter,V
ideo_Source_Array[Video_Counter].HICKUP_C);
HickTotal=HickTotal+Video_Source_Array[Video_Counter].HICKUP_C;
}
fprintf(fptr,"\nTOTALVCC= %ld TOTALVHC= %ld TOTALViCC= %ld TOTALViHC= %ld
HICKUPS= %ld",TOTALVCC,TOTALVHC,TOTALViCC,TOTALViHC,HickTotal);
}           /* END OF LOOP REPEAT */
fclose(fptr);
}



int VoiceGen(int X_State)  /* if X_State=0 the terminal is innactive, else
active */
{
int ret;
float tem;
tem=random(32000)/31999.0;
if (X_State==0 && tem<0.006741573) ret=1; else
if (X_State==1 && tem<0.008510638) ret=0; else ret=X_State;
return(ret);
}


int VideoGen(int X_State) /* X_State declares the current bit rate of the video
source */
{
int ret;
float tem;
tem=random(32000)/31999.0;
if (tem<0.036*B_TRANSITION_RATES[X_State]) ret=(X_State-1);else
if (tem<0.036*(A_TRANSITION_RATES[X_State]+B_TRANSITION_RATES[X_State]))
ret=(X_State+1); else ret=X_State;
return(ret);
}


struct Source_Model  *rand_sort(struct Source_Model  *source_arr)
{
int j,t,tmp1;
struct Source_Model tmp2;
for(i=Voice_Source_N-1;i>=0;i--)
{
 j=(Voice_Source_N-i);
 tmp1=random(j+1);
 tmp2=source_arr[i];
 source_arr[i]=source_arr[tmp1];
 source_arr[tmp1]=tmp2;
}
return(&(source_arr[0]));
}
```