ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΥ ΣΧΕΔΙΑΣΜΟΥ & ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ενδιάμεσο λογισμικό: Αρχές Λειτουργίας και Παραδείγματα

ΣΠΟΥΔΑΣΤΕΣ:

ΔΡΑΚΟ ΡΟΜΠΕΡΤ

ΣΑΝΤΙ ΦΛΟΡΕΝΤΙΝ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ:

ΓΚΟΥΜΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ

ΠΑΤΡΑ 2012

ΠΡΟΛΟΓΟΣ

Το παρόν τεύχος αποτελεί την Πτυχιακή Εργασία που εκπονήθηκε στο Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Πατρών και αναφέρεται στο Ενδιάμεσο λογισμικό: Αρχές Λειτουργίας και Παραδείγματα.

Στη συγκεκριμένη εργασία εξετάστηκαν έννοιες, όπως:

- τα κατανεμημένα συστήματα και τα πλεονεκτήματά τους, έναντι των συγκεντρωτικών συστημάτων,
- το Ενδιάμεσο Λογισμικό (middleware) και ο ρόλος του στην ανάπτυξη κατανεμημένων εφαρμογών,
- μοντέλα προγραμματισμού που υποστηρίζονται από συστήματα Ενδιάμεσου Λογισμικού καθώς και οι λειτουργίες που παρέχονται.

Δώσαμε έμφαση στο μοντέλο των κατανεμημένων αντικειμένων και τα χαρακτηριστικά του. Παρουσιάζουμε τις βασικότερες τεχνολογίες αυτής της κατηγορίας (CORBA, Java RMI, DCOM, GLOBE). Τέλος, γίνεται μια σύγκριση των τεχνολογιών αυτών και αναφέρουμε τα συμπεράσματα μας σε σχέση με το ενδιάμεσο λογισμικό, τα κατανεμημένα συστήματα και τα κατανεμημένα αντικείμενα.

Στο σημείο αυτό θα θέλαμε να ευχαριστήσουμε θερμά τον επιβλέποντα καθηγητή κ. Χρήστο Γκουμόπουλο για την άψογη συνεργασία που είχαμε μαζί του, προσφέροντας μας πολύτιμη βοήθεια και καθοδήγηση σε όλες τις φάσεις τις πτυχιακής μας εργασίας.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία έγινε με σκοπό τη μελέτη του ενδιάμεσου λογισμικού, την εξέταση της συμβολής του στη διαχείριση της πολυπλοκότητας των κατανεμημένων εφαρμογών, την παρουσίαση των διαφόρων τεχνολογιών ενδιάμεσου λογισμικού και παραδειγμάτων αυτών.

Η ανάπτυξη του θέματος γίνεται σε έξι κεφάλαια στα οποία συμπεριλαμβάνονται μερικά υποκεφάλαια. Το πρώτο κεφάλαιο αναφέρεται στα Κατανεμημένα Συστήματα, το δεύτερο κεφάλαιο αναφέρεται στο Ενδιάμεσο Λογισμικό, το τρίτο κεφάλαιο στα Συστήματα Κατανεμημένων Αντικειμένων, το τέταρτο στις τεχνολογίες RMI και CORBA, στο πέμπτο κεφάλαιο γίνεται σύγκριση των τεχνολογιών (DCOM,GLOBE,CORBA) και στο τελευταίο κεφάλαιο βγάζουμε τα συμπεράσματα μας.

Πιο αναλυτικά στο πρώτο κεφάλαιο αναλύονται τα κατανεμημένα συστήματα δίνοντας αρχικά έμφαση στον ορισμό τους, τις λειτουργίες αυτών και τις κατηγορίες στις οποίες χωρίζονται ενώ στην συνέχεια γίνεται μια εισαγωγή στο Ενδιάμεσο Λογισμικό και Ενδιάμεσο Λογισμικό Κατανεμημένου Αντικειμένου (Distributed Object Computing Middleware).

Στο δεύτερο κεφάλαιο αναλύονται οι κατηγορίες του Ενδιάμεσου Λογισμικού (Middleware), όπως είναι το Ενδιάμεσο Λογισμικό βασισμένο σε μήνυμα και επίσης αναφέρονται τα συστήματα στα οποία χρησιμοποιείται. Ακόμη γίνεται αναφορά στις τεχνολογίες Component Middleware.

Στο τρίτο κεφάλαιο αναλύονται τα συστήματα κατανεμημένων αντικειμένων, οι αρχές στις οποίες βασίζονται το DCOM και το Globe καθώς και τα χαρακτηριστικά τους.

Στο τέταρτο κεφάλαιο παρουσιάζονται αναλυτικά το RMI (Remote Method Invocation) και το CORBA (Common Object Request Broker Architecture) δίνοντας έμφαση στις αρχιτεκτονικές τους, τους στόχους και τις λειτουργίες αυτών.

Στο πέμπτο κεφάλαιο γίνεται μια σύγκριση των παραπάνω συστημάτων με αναφορά στα πλεονεκτήματα και τα μειονεκτήματά τους.

Και στο τελευταίο κεφάλαιο γράφουμε τα συμπεράσματά μας σε σχέση με το Middleware και τις τεχνολογίες του.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Σε ένα κατανεμημένο σύστημα υπολογιστών, το ενδιάμεσο λογισμικό είναι το επίπεδο εκείνο που βρίσκεται ανάμεσα στα λειτουργικά συστήματα και τις κατανεμημένες εφαρμογές. Ο ρόλος του είναι να συγκαλύψει την ετερογένεια που υπάρχει ανάμεσα στα διαφορετικά λειτουργικά συστήματα και να παρέχει μια άποψη ενιαίου συνεκτικού συστήματος.

Για διάφορους λόγους, το middleware έχει εξελιχθεί σε καθοριστικό υπόστρωμα για πολλές κατανεμημένες εφαρμογές. Το middleware παρέχει τη δυνατότητα μεταφοράς, προστατεύει τις εφαρμογές από την πολυπλοκότητα της δικτύωσης και της διανομής, ώστε οι προγραμματιστές να μπορούν να επικεντρώνονται στην λογική των αιτήσεών τους. Επίσης αναφέρονται τεχνολογίες κατανεμημένων αντικειμένων, όπως το CORBA, RMI, GLOBE, DCOM με τα πλεονεκτήματα και τα μειονεκτήματά τους, οι διαφορές και οι ομοιότητες μεταξύ τους.

Αυτή η εργασία παρουσιάζει μια επισκόπηση του middleware, των κατανεμημένων συστημάτων, τεχνολογίες κατανεμημένων αντικειμένων. Πως λειτουργούν ώστε να υπάρχει διαφάνεια και διαλειτουργικότητα στα υπολογιστικά συστήματα.

ΚΕΦΑΛΑΙΟ1: ΕΙΣΑΓΩΓΗ ΣΤΑ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

1.1 Ορισμός Κατανεμημένου Συστήματος

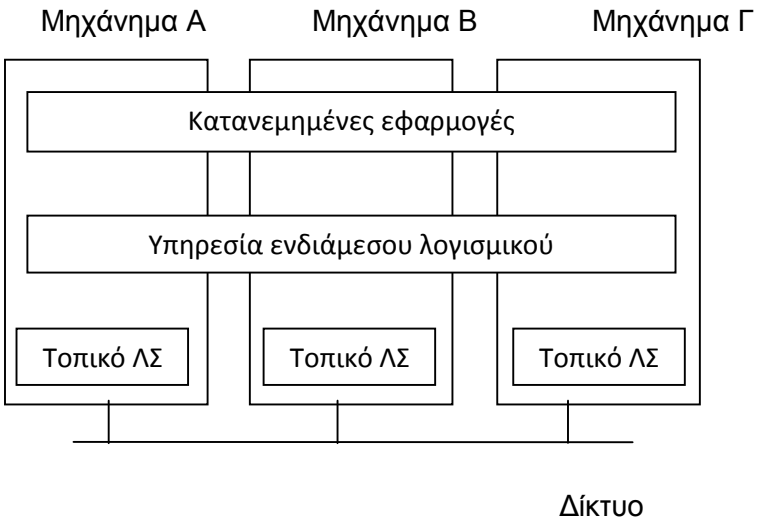
Κατανεμημένο σύστημα είναι μια συλλογή από ανεξάρτητους υπολογιστές, οι οποίοι εμφανίζονται στους χρήστες τους ως ένα ενιαίο συνεκτικό σύστημα.

Ο ορισμός αυτός έχει δύο πλευρές. Η πρώτη αφορά το υλικό: τα μηχανήματα είναι αυτόνομα. Η δεύτερη αφορά το λογισμικό: οι χρήστες θεωρούν ότι έχουν να κάνουν με ένα ενιαίο σύστημα. Και οι δύο πλευρές είναι ουσιώδεις. [16]

Ένα σημαντικό χαρακτηριστικό είναι ότι οι διαφορές μεταξύ των διαφόρων υπολογιστών και οι τρόποι με τους οποίους αυτοί επικοινωνούν παραμένουν κρυφοί στους χρήστες. Το ίδιο ισχύει και για την εσωτερική οργάνωση των κατανεμημένων συστημάτων. Ένα άλλο σημαντικό χαρακτηριστικό είναι ότι χρήστες και εφαρμογές μπορούν να αλληλεπιδρούν με ένα κατανεμημένο σύστημα με συνεπή και ομοιόμορφο τρόπο, ανεξάρτητα από το πού και πότε πραγματοποιείται η αλληλεπίδραση.

Τα κατανεμημένα συστήματα θα πρέπει επίσης να επιτρέπουν με σχετική ευκολία την επέκτασή τους ή την προσαρμογή της κλίμακας του μεγέθους τους. Αυτό το χαρακτηριστικό είναι άμεση συνέπεια της απαίτησης να έχουμε ανεξάρτητους υπολογιστές και ταυτόχρονα να παραμένει κρυφό το πως ακριβώς αυτοί οι υπολογιστές συμμετέχουν στο σύστημα ως σύνολο. Υπό κανονικές συνθήκες, ένα κατανεμημένο σύστημα είναι συνεχώς διαθέσιμο, αν και ορισμένα μέρη του μπορεί να είναι προσωρινά εκτός λειτουργίας. Οι χρήστες και εφαρμογές δε θα πρέπει να είναι σε θέση να δουν ότι κάποια μέρη αντικαθίστανται ή επιδιορθώνονται, ή ότι προστίθενται νέα μέρη για την εξυπηρέτηση περισσότερων χρηστών ή εφαρμογών.

Τα κατανεμημένα συστήματα, προκειμένου να υποστηρίζουν ετερογενείς υπολογιστές και δίκτυα παρέχοντας παράλληλα την άποψη ενός ενιαίου συστήματος, οργανώνονται συχνά με τη χρήση ενός επιπέδου λογισμικού το οποίο από λογική άποψη τοποθετείται ανάμεσα σε ένα ανώτερο επίπεδο που αποτελείται από χρήστες και εφαρμογές και ένα κατώτερο επίπεδο που αποτελείται από λειτουργικά συστήματα, όπως φαίνεται στο Εικόνα (1.1). Με τη λογική αυτή, ένα τέτοιο κατανεμημένο σύστημα ονομάζεται **ενδιάμεσο λογισμικό**.



Εικόνα (1.1) Κατανεμημένο σύστημα οργανωμένο ως ενδιάμεσο λογισμικό. Παρατηρήστε ότι το επίπεδο ενδιάμεσου λογισμικού εκτείνεται σε πολλά μηχανήματα.

1.1.1 Ανοιχτή Λειτουργία

Ένας σημαντικός στόχος των κατανεμημένων συστημάτων είναι η ανοιχτή λειτουργία. **Ανοιχτό κατανεμημένο σύστημα** (open distributed system) ονομάζεται ένα σύστημα που παρέχει υπηρεσίες σύμφωνα με κάποιους καθιερωμένους κανόνες, οι οποίοι περιγράφουν τη σύνταξη και τη σημειολογία αυτών των υπηρεσιών. Για παράδειγμα, στα δίκτυα υπολογιστών υπάρχουν καθιερωμένοι κανόνες που διέπουν τη μορφή, το περιεχόμενο, και τη σημασία των μηνυμάτων που στέλνονται και λαμβάνονται. Οι κανόνες αυτοί τυποποιούνται σε πρωτόκολλα. Στα κατανεμημένα συστήματα, οι υπηρεσίες γενικά καθορίζονται μέσω **διασυνδέσεων** (interfaces), οι οποίες συχνά περιγράφονται σε μια **Γλώσσα Ορισμού Διασυνδέσεων** (Interface Definition Language, **IDL**). Οι ορισμοί διασυνδέσεων που γράφονται σε μια γλώσσα IDL σχεδόν πάντα συλλαμβάνουν μόνο τη σύνταξη των υπηρεσιών. Με άλλα λόγια, καθορίζουν ακριβώς τα ονόματα των διαθέσιμων συναρτήσεων καθώς και τους τύπους των παραμέτρων, τις επιστρεφόμενες τιμές, τις πιθανές εξαιρέσεις που μπορεί να προκύψουν κ.λπ. Το δύσκολο είναι να καθοριστεί ακριβώς το τι κάνουν αυτές οι υπηρεσίες, δηλαδή η σημειολογία των διασυνδέσεων. Στην πράξη, οι προδιαγραφές αυτές δίνονται απλώς με έναν άτυπο τρόπο, σε φυσική γλώσσα.

Οι σωστές προδιαγραφές είναι πλήρεις και ουδέτερες. ¨Πλήρεις¨ σημαίνει ότι όλα όσα χρειάζονται για να δημιουργηθεί μια υλοποίηση, έχουν πραγματικά καθοριστεί. Ωστόσο, πολλοί ορισμοί διασυνδέσεων δεν είναι καθόλου πλήρεις, και έτσι ο κατασκευαστής είναι αναγκασμένος να προσθέτει λεπτομέρειες ειδικά για την συγκεκριμένη υλοποίηση. Το ίδιο σημαντικό είναι το γεγονός ότι οι προδιαγραφές δεν υπαγορεύουν τη μορφή που θα έχει η υλοποίηση, πρέπει να είναι ουδέτερες. Η πληρότητα και ουδετερότητα αποτελούν σημαντικούς παράγοντες για τη διαλειτουργικότητα και τη φορητότητα. Η **διαλειτουργικότητα** (interoperability) χαρακτηρίζει το κατά πόσον δύο υλοποιήσεις συστημάτων ή στοιχείων που προέρχονται από διαφορετικούς κατασκευαστές μπορούν να συνυπάρχουν και να συνεργάζονται βασιζόμενες απλώς η μία στις υπηρεσίες της άλλης, όπως αυτές καθορίζονται από ένα κοινό πρότυπο. Η **φορητότητα** (portability) χαρακτηρίζει το κατά πόσον μια εφαρμογή που έχει αναπτυχθεί για ένα κατανεμημένο σύστημα Α μπορεί να εκτελεστεί χωρίς τροποποιήσεις σε ένα διαφορετικό κατανεμημένο σύστημα Β, το οποίο υλοποιεί τις ίδιες συνδέσεις του Α.

Ένας άλλος σημαντικός στόχος για ένα ανοιχτό κατανεμημένο σύστημα είναι ότι θα πρέπει να είναι ευέλικτο, πράγμα που σημαίνει ότι θα πρέπει να είναι εύκολο να διευθετηθεί με τη χρήση διαφορετικών συστατικών στοιχείων ή η αντικατάσταση υπάρχοντων στοιχείων, χωρίς να επηρεάζονται τα στοιχεία που παραμένουν. Με άλλα λόγια, ένα ανοιχτό κατανεμημένο σύστημα πρέπει να είναι και **επεκτάσιμο**. Για παράδειγμα, σε ένα ευέλικτο σύστημα πρέπει να είναι σχετικά εύκολο να προστεθούν στοιχεία που λειτουργούν σε διαφορετικά λειτουργικά συστήματα, ή ακόμα και να αντικατασταθεί ένα ολόκληρο σύστημα αρχείων. Όπως οι περισσότεροι γνωρίζουμε από την καθημερινή πρακτική, η ευελιξία επιτυγχάνεται ευκολότερα στα λόγια παρά στην πράξη.

1.1.2 Έννοιες σχετικές με το λογισμικό

Το υλικό των κατανεμημένων συστημάτων είναι σημαντικό, αλλά το λογισμικό είναι εκείνο που καθορίζει σε μεγάλο βαθμό το πώς εμφανίζεται, στην πραγματικότητα, ένα κατανεμημένο σύστημα. Τα κατανεμημένα συστήματα έχουν πολλές ομοιότητες με τα παραδοσιακά λειτουργικά συστήματα. Πρώτον ενεργούν ως **διαχειριστές πόρων** (resource managers) για το υποκείμενο υλικό, επιτρέποντας σε πολλούς χρήστες και εφαρμογές να μοιράζονται πόρους, όπως CPU, μνήμες, περιφερειακές συσκευές, δίκτυο, και δεδομένα όλων των ειδών. Δεύτερον, και ίσως σημαντικότερο, τα κατανεμημένα συστήματα προσπαθούν να κρύψουν τις περιπλοκές και την ετερογενή φύση του υποκειμενικού υλικού, παρέχοντας μια εικονική μηχανή στην οποία μπορούν εύκολα να εκτελούνται οι εφαρμογές. [15]

Για να γίνει λοιπόν κατανοητή η φύση των κατανεμημένων συστημάτων, θα ρίξουμε πρώτα μια ματιά στα λειτουργικά συστήματα σε σχέση με τους κατανεμημένους υπολογιστές. Τα λειτουργικά συστήματα για κατανεμημένους υπολογιστές μπορούν χονδρικά να υποδιαιρεθούν σε δύο κατηγορίες: τα στενά συνδεδεμένα συστήματα (tightly-coupled systems) και τα χαλαρά συνδεδεμένα συστήματα (loosely-coupled systems). Στα στενά συνδεδεμένα συστήματα, το λειτουργικό σύστημα ουσιαστικά προσπαθεί να συντηρήσει μια ενιαία, καθολική άποψη των πόρων που διαχειρίζεται. Τα χαλαρά συνδεδεμένα συστήματα μπορούν να θεωρηθούν σύνολα υπολογιστών, καθένας από τους οποίους χρησιμοποιεί το δικό του λειτουργικό σύστημα. Ωστόσο, αυτά τα λειτουργικά συστήματα συνεργάζονται για να θέσουν τις υπηρεσίες τους και τους πόρους τους στη διάθεση των άλλων.

Ένα λειτουργικό σύστημα στενής σύνδεσης χαρακτηρίζεται γενικά ως **κατανεμημένο λειτουργικό σύστημα** (Distributed Operating System, **DOS**), και χρησιμοποιείται για τη διαχείριση πολυεπεξεργαστών και ομοιογενών πολύ-υπολογιστών. Όπως και τα παραδοσιακά λειτουργικά συστήματα, ο βασικός στόχος ενός κατανεμημένου λειτουργικού συστήματος είναι να κρύβει τις περιπλοκές της διαχείρισης του υποκειμένου υλικού, έτσι ώστε να μπορούν να μοιράζονται πολλές διεργασίες.

Το χαλαρά συνδεδεμένο **δικτυακό λειτουργικό σύστημα** (Network Operating System, **NOS**) χρησιμοποιείται στα ετερογενή πολύ-υπολογιστικά συστήματα. Αν και η διαχείριση του υποκειμένου υλικού είναι σημαντικό ζήτημα για ένα NOS, η βασική διαφορά από τα παραδοσιακά λειτουργικά συστήματα έγκειται στο γεγονός ότι τοπικές υπηρεσίες γίνονται διαθέσιμες σε απομακρυσμένους πελάτες.

Για να φτάσουμε πραγματικά σε ένα κατανεμημένο σύστημα, χρειάζεται να γίνουν βελτιώσεις στις υπηρεσίες των δικτυακών λειτουργικών συστημάτων, όπως καλύτερη υποστήριξη για τη διαφάνεια της κατανεμημένης λειτουργίας. Οι βελτιώσεις αυτές οδηγούν στο λεγόμενο **ενδιάμεσο λογισμικό** (middleware) και βρίσκονται στην καρδιά των σημερινών κατανεμημένων συστημάτων.

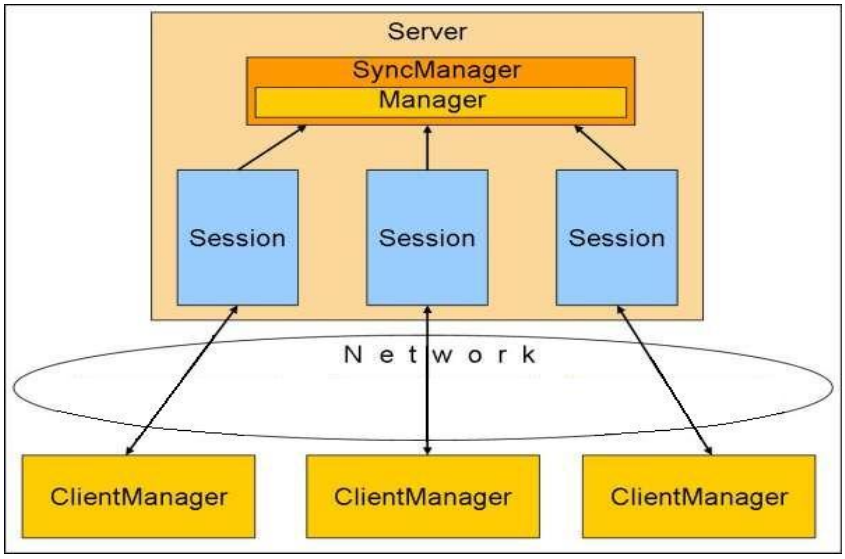
1.1.3 Ενδιάμεσο Λογισμικό

Ούτε τα κατανεμημένα αλλά ούτε και τα δικτυακά λειτουργικά συστήματα δεν ικανοποιούν πραγματικά τα κριτήρια για να θεωρηθούν κατανεμημένα συστήματα, σύμφωνα με τον ορισμό που δώσαμε παραπάνω (κεφ. 1.1). Ο σκοπός ενός κατανεμημένου λειτουργικού συστήματος δεν είναι να χειρίζεται ένα σύνολο ανεξάρτητων υπολογιστών, ενώ ένα δικτυακό λειτουργικό σύστημα δεν παρέχει μια άποψη ενιαίου συνεκτικού συστήματος. Προκύπτει τώρα το ερώτημα αν είναι δυνατόν να δημιουργηθεί ένα κατανεμημένο σύστημα που να έχει τα πλεονεκτήματα και των δύο προσεγγίσεων: την επεκτασιμότητα και την ανοιχτή λειτουργία των δικτυακών λειτουργικών συστημάτων, καθώς και τη διαφάνεια και τη σχετική ευχρηστία των κατανεμημένων λειτουργικών συστημάτων. Η λύση μπορεί να βρεθεί σε ένα πρόσθετο επίπεδο λογισμικού, το οποίο χρησιμοποιείται στα δικτυακά λειτουργικά συστήματα με σκοπό να κρύβει λίγο-πολύ την ετερογένεια του συνόλου των υποκείμενων υπολογιστικών περιβαλλόντων, αλλά και να βελτιώνει τη διαφάνεια της κατανεμημένης λειτουργίας. Πολλά σημερινά κατανεμημένα συστήματα κατασκευάζονται μέσω ενός τέτοιου πρόσθετου επιπέδου, το οποίο λέγεται **ενδιάμεσο λογισμικό** (middleware).

1.2 Τεχνολογίες Ανάπτυξης Λογισμικού

Σήμερα οι κυρίαρχες τεχνολογίες στις οποίες στηρίζεται η ανάπτυξη λογισμικού είναι δύο: η τεχνολογία πελάτη-εξυπηρετητή (client-server) και η τεχνολογία αντικειμένων (object technology).

Η τεχνολογία που στηρίζεται στο μοντέλο πελάτη-εξυπηρετητή χωρίζει το λογισμικό που παράγεται σε δυο κύρια μέρη. Δηλαδή αυτού του μέρους που αιτείται υπηρεσιών και ονομάζεται <<calling module>> και αυτού του μέρους που παρέχει υπηρεσίες και ονομάζεται <<called module>>.[1] Με αυτή την τεχνολογία το λογισμικό προχώρησε αρκετά βήματα εμπρός. Μέχρι και σήμερα οι εξυπηρετητές μπορούν να παρέχουν τις υπηρεσίες τους μαζικά σε πελάτες από μόνο μια θέση τους στο δίκτυο. Επίσης δεν υπάρχει ο περιορισμός ο πελάτης να βρίσκεται στην ίδια φυσική θέση με τον εξυπηρετητή. Η τεχνική που χρησιμοποιείται για την επικοινωνία των δύο αυτών τμημάτων είναι μέσω αποστολής μηνυμάτων. Στην Εικόνα (1.2) φαίνεται αυτή η αρχιτεκτονική:



Εικόνα (1.2) Τυπικό παράδειγμα αρχιτεκτονικής client-server

Η δεύτερη τεχνολογία που αναφερόμαστε είναι του αντικειμενοστραφούς προγραμματισμού. Με βάση αυτή όλα προέρχονται από τύπους αντικειμένων. Οι τύποι αντικειμένων αναφέρονται και ως κλάσεις. Κάθε αντικείμενο μπορεί να περιέχει δεδομένα καθώς και μεθόδους όπου υλοποιούνται οι λειτουργίες του αντικειμένου. Μάλιστα αυτές οι μέθοδοι, κάτω από προϋποθέσεις, εκτίθενται στο περιβάλλον τους και μπορεί να κληθούν από άλλα αντικείμενα. Το προτέρημα αυτής της τεχνολογίας είναι ότι κάθε στιγμιότυπο μιας κλάσης θα έχει την ίδια συμπεριφορά που ορίζεται από την κλάση δημιουργό τους. Αυτό σημαίνει πως κάθε αντικείμενο διαθέτει ένα σαφώς ορισμένο τρόπο που επικοινωνεί με το περιβάλλον του, ο οποίος καθορίζεται από την κλάση του.

Η τεχνολογία λογισμικού είναι μια τεχνολογία η οποία εξελίσσεται με γρήγορους ρυθμούς. Όμως θα ήταν ασύμφορο αν συνεχώς λόγω της εξέλιξης το λογισμικό για να συντηρηθεί ή να ανανεωθεί γραφόταν συνεχώς από την αρχή. Έτσι έχει γίνει σαφής η ανάγκη για λογισμικό που θα πληροί ορισμένα χαρακτηριστικά. Το λογισμικό αυτό θα πρέπει να είναι ανά τμήματα αυτόνομο και θα μπορεί να ενθυλακώνει στο εσωτερικό του λογισμικό προηγουμένων τεχνολογιών. Σαφές παράδειγμα είναι η μετάβαση από παλαιότερες γλώσσες προγραμματισμού σε νεότερες. Στις περισσότερες περιπτώσεις αυτή η μετάβαση προϋποθέτει την επανασυγγραφή σχεδόν του συνόλου του κώδικα και ορισμένες φορές τον επανασχεδιασμό του έργου εξ αρχής. Η ανάγκη της ανάπτυξης αυτόνομων τμημάτων λογισμικού όπου θα μπορούν να επαναχρησιμοποιηθούν ανεξαρτήτως γλώσσας προγραμματισμού, εργαλείων ανάπτυξης και φυσικών συστημάτων πάνω στα οποία θα εγκατασταθούν έχει αναφερθεί από πολύ παλιά. [2] Παρόλα αυτά μόνο στα τέλη της προηγούμενης δεκαετίας κάτι τέτοιο ξεκίνησε να φαίνεται εφικτό.

Ταυτόχρονα υπάρχουν κάποιοι τομείς στους οποίους το λογισμικό που αναπτύσσεται γι αυτούς πρέπει να είναι σθεναρό και ταυτόχρονα επεκτάσιμο. Τέτοιοι τομείς είναι η βιομηχανία και ιδιαίτερα η αμυντική, η ιατρική τεχνολογία, και τα τραπεζικά συστήματα. Σήμερα αυτοί οι τομείς αντιμετωπίζουν τεράστια προβλήματα επέκτασης και εκσυγχρονισμού του λογισμικού τους. Το μεγαλύτερο πρόβλημα είναι ότι μόνον ένα μέρος τους χρειάζεται να ανανεώνεται κάθε φορά αλλά η μετάβαση σε πιο σύγχρονες τεχνολογίες απαιτεί την εξολοκλήρου αντικατάστασή του. Ακόμα στους τομείς αυτούς παρατηρούνται ετερογενή συστήματα, τα οποία λειτουργούν με διαφορετικά λειτουργικά συστήματα και στο πέρασμα των χρόνων έχει αναπτυχθεί γι’ αυτά λογισμικό από διαφορετικές γλώσσες προγραμματισμού που ακολουθούν διαφορετικά η καθεμία μοντέλα ανάπτυξης. Έτσι πολλές φορές αντικείμενα που ήδη έχουν φτιαχτεί δεν μπορούν να συνδυαστούν μεταξύ τους ή υπάρχοντες εξυπηρετητές δεν μπορούν να είναι συμβατοί με νεότερους πελάτες.

Για το λόγο αυτό η βιομηχανία λογισμικού προχώρησε ένα βήμα παραπέρα. Σήμερα προτείνει, στη γενική του μορφή, τη λύση του ενδιάμεσου λογισμικού (middleware). Αυτή είναι μια τεχνολογία που συνδυάζει την τεχνολογία client-server με την αντικειμενοστραφή. Επίσης προσπαθεί να μετριάσει τις δυσκολίες που δημιουργούνται από την συντήρηση και επέκταση του λογισμικού. Μάλιστα παρέχει τη δυνατότητα οι εξυπηρετητές να είναι κατανεμημένοι στο δίκτυο.

1.3 Εισαγωγή στο Ενδιάμεσο Λογισμικό Κατανεμημένου Αντικειμένου (DOC Middleware)

Στα προβλήματα που αναφέραμε παραπάνω απάντηση ήρθε να δώσει η τεχνολογία ενδιαμέσων (Middleware technology). Γενικά ένας διαμεσολαβητής είναι αυτός που εξασφαλίζει την επικοινωνία μεταξύ πελάτη και εξυπηρετητή.[3] Δηλαδή ο ενδιάμεσος αναλαμβάνει εκείνο το κομμάτι που έχει να κάνει με τη διασύνδεση των διεπαφών ενός πελάτη και ενός εξυπηρετητή κάτω από διαφορετικά πρωτόκολλα και από διαφορετικά φυσικά συστήματα.

Ο ρόλος του είναι να δημιουργήσει ένα επικοινωνιακό περιβάλλον για τα διάφορα τμήματα μιας κατανεμημένης εφαρμογής που να είναι το ίδιο για οποιοδήποτε σύστημα και αν εγκατασταθεί η εφαρμογή. Κατά αυτόν τον τρόπο δίνεται η δυνατότητα σε ετερογενή συστήματα να συνεργαστούν μεταξύ τους.

Στη γενική του θεώρηση ένα DOC middleware προσπαθεί να αυξήσει την επαναχρησιμοποίηση τμημάτων του λογισμικού με το να προσφέρει σε όλα τα συστήματα το ίδιο υπόβαθρο πάνω στο οποίο θα στηθούν. Με αυτόν τον τρόπο έχουμε και μείωση των λαθών που γίνονται κατά την ανάπτυξη μιας εφαρμογής. Τα χαρακτηριστικά που μπορεί να προσφέρει ένα ενδιάμεσο λογισμικό στον προγραμματιστή είναι:

Α) Η διεπαφή επικοινωνίας: Ένα κομμάτι μιας εφαρμογής που βρίσκεται σε ένα σύστημα χρησιμοποιώντας αυτή τη διεπαφή δεν χρειάζεται να λαμβάνει υπόψη την υπόλοιπη δομή έξω από το σύστημα για να επικοινωνήσει με τα άλλα κομμάτια της. Το ενδιάμεσο λογισμικό θα αναλάβει όλο το κομμάτι των κλήσεων μεταξύ τους καθώς και την αποστολή παραμέτρων και την αποκωδικοποίηση εισερχομένων αιτήσεων.

Β) Η διεπαφή συστήματος: Κάθε ενδιάμεσο λογισμικό παρέχει μια διεπαφή όπου με αυτήν μια εφαρμογή μπορεί να επικοινωνεί με τους πόρους του συστήματος με κοινό τρόπο για όλα τα συστήματα. Με αυτόν τον τρόπο όταν μεταφερθεί σε ένα άλλο, που δεν είναι όμοιο με το προηγούμενο, το ενδιάμεσο λογισμικό θα αναλάβει να αποκρύψει την ετερογένεια.

Γ) Η διεπαφές υπηρεσιών: Το ενδιάμεσο λογισμικό προσφέρει με μορφή διεπαφών διάφορες υπηρεσίες κοινής χρήσης στον προγραμματιστή. Αυτές είναι υπηρεσίες που συνήθως οι περισσότερες εφαρμογές καλούνται να υλοποιούν και μπορούν να ενσωματωθούν απευθείας. Τέτοιες είναι το security service, το transaction service κ.α.

Δ) Η διεπαφή διασύνδεσης: Αυτή η διεπαφή συνήθως αποκρύπτει από τον προγραμματιστή το μεγαλύτερο μέρος της και είναι αυτή που είναι υπεύθυνη για την επικοινωνία μεταξύ των ενδιάμεσων λογισμικών.

Παρακινούμενη από την ανάπτυξη των δικτυακών εφαρμογών, η τεχνολογία των ενδιάμεσων λογισμικών έχει αποκτήσει μεγάλη σημασία. Καλύπτουν μια μεγάλη γκάμα συστημάτων λογισμικού η οποία περιλαμβάνει κατανεμημένα αντικείμενα και συνιστώσες, επικοινωνία εφαρμογών μέσω μηνυμάτων καθώς και επικοινωνία ενσωματωμένων συστημάτων.
Οι σημερινές πλατφόρμες και μοντέλα που επικρατούν πάνω σε τεχνολογίες ενδιαμέσων είναι τρεις (CORBA, RMI, COM) τις οποίες θα αναλύσουμε σε άλλο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 2: ΕΝΔΙΑΜΕΣΟ ΛΟΓΙΣΜΙΚΟ

2.1 Ενδιάμεσο Λογισμικό (Middleware)

Αποτελεί ζωτικό μέρος κάθε συστήματος πελάτη-διακομιστή. Ευτυχώς, τις περισσότερες φορές, ο προγραμματιστής και ο σχεδιαστής δεν χρειάζεται ν' ανησυχούν για το πώς λειτουργεί. Χρειάζεται μόνο να το χρησιμοποιούν μέσω επικλήσεων μεθόδων. Η έννοια του ενδιάμεσου λογισμικού (middleware) είναι κάπως ομιχλώδης. Ο καλύτερος ορισμός είναι των Orfali, Harkey και Edwards:

Το Middleware είναι ένας ευρύς όρος που καλύπτει όλα τα κατανεμημένα λογισμικά που χρειάζονται για την υποστήριξη της συνεργασίας πελατών και διακομιστών....Πού αρχίζει το middleware και πού τελειώνει; Αρχίζει με την ομάδα API στην πλευρά του πελάτη που χρησιμοποιείται για την επίκληση μιας υπηρεσίας, και καλύπτει την μεταβίβαση του αιτήματος στο δίκτυο και την αντίστοιχη απάντηση. [6]

Το καλύτερο παράδειγμα ενδιάμεσου λογισμικού είναι το λογισμικό διεπαφής ενός browser και του Παγκόσμιου Ιστού. Όταν γίνεται αίτηση για μια ιστοσελίδα, ας πούμε όταν ένας χρήστης κάνει κλικ σ' έναν υπερσύνδεσμο μιας σελίδας, ένα κείμενο, που αποτελεί μέρος ενός πρωτοκόλλου, αποστέλλεται στο ενδιάμεσο λογισμικό ρωτώντας για την σελίδα. Στην συνέχεια, το ενδιάμεσο λογισμικό θα εντοπίσει την σελίδα, θα την ανακτήσει και θα την στείλει πίσω στον browser. Τμήμα του ενδιάμεσου λογισμικού είναι ένα πρωτόκολλο που το αφορά η ανεύρεση σελίδων, η παρακολούθηση λαθών, η αναφορά λαθών, η μεταβίβαση δεδομένων και η επικοινωνία με τα κατώτερα στρώματα λογισμικού, που αποτελεί μέρος του TCP/IP. Υπάρχουν δύο κατηγορίες ενδιάμεσου λογισμικού: το γενικό middleware και η υπηρεσία middleware. Το γενικό middleware είναι λογισμικό που συσχετίζεται με υπηρεσίες που ζητούνται από όλους τους πελάτες και τους διακομιστές. Κανονικά το περισσότερο από αυτό το ενδιάμεσο λογισμικό αφορά τα λειτουργικά συστήματα. Το τυπικό λογισμικό που υπάγεται σ' αυτό το πλαίσιο περιλαμβάνει:

- Λογισμικό για την διεξαγωγή επεξεργασιών όπως η μεταφορά ακατέργαστων δεδομένων χαρακτήρων στο Internet.
- Λογισμικό που συγχρονίζει αντιγραμμένα αρχεία.
- Λογισμικό που διαχειρίζεται μία κατανεμημένη συλλογή αρχείων.

Η υπηρεσία middleware συσχετίζεται με μία συγκεκριμένη υπηρεσία όπως η εκτύπωση αρχείων σ' έναν απομακρυσμένο υπολογιστή. Τα τυπικά παραδείγματα περιλαμβάνουν:

- Λογισμικό που επιτρέπει σ' έναν πελάτη να θέσει ερώτημα σε μία βάση δεδομένων, π.χ. λογισμικό που ερμηνεύει και εκτελεί ένα ερώτημα που τίθεται από τον χρήστη ενός πελάτη υπολογιστή και επεξεργάζεται τα δεδομένα στέλνοντας τα αποτελέσματα στον πελάτη.
- Λογισμικό που αφορά συνδυασμούς κατανεμημένων αντικειμένων όπως το RMI. Το λογισμικό αυτό επιτρέπει στον χρήστη να δημιουργήσει ένα αντικείμενο και στους πελάτες ν' αποστείλουν μηνύματα σ' αυτό το αντικείμενο. Εφαρμόζει λειτουργίες που αφορούν την ασφάλεια, τον εντοπισμό ενός αντικειμένου, την διαβίβαση δεδομένων, που σχετίζονται με την μέθοδο ορισμών σ' ένα κατανεμημένο αντικείμενο και την μεταφορά των αποτελεσμάτων ενός μηνύματος που έχει αποσταλεί σ' ένα κατανεμημένο αντικείμενο.
- Το middleware που αφορά τις ομάδες συζητήσεων που δίνει την δυνατότητα στον χρήστη να διαβάζει και να στέλνει μηνύματα στην ομάδα.

2.1.1 Τοποθέτηση ενδιάμεσου λογισμικού

Κάθε τοπικό σύστημα που αποτελεί μέρος του υποκείμενου δικτυακού λειτουργικού συστήματος θεωρείται ότι παρέχει διαχείριση των τοπικών πόρων καθώς και απλούς τρόπους επικοινωνίας για σύνδεση με άλλους υπολογιστές. Με άλλα λόγια, το ίδιο το ενδιάμεσο λογισμικό δε διαχειρίζεται κανένα μεμονωμένο κόμβο η δουλειά αυτή ανατίθεται στο τοπικό λειτουργικό σύστημα.

Ένας σημαντικός στόχος είναι η απόκρυψη της ετερογένειας των υποκείμενων υπολογιστικών περιβαλλόντων από τις εφαρμογές. Γι’ αυτό, πολλά συστήματα ενδιάμεσου λογισμικού διαθέτουν μια λιγότερο ή περισσότερο πλήρη συλλογή υπηρεσιών και αποθαρρύνουν τη χρήση οποιουδήποτε άλλου μέσου πέρα από τις διασυνδέσεις τους για αυτές τις υπηρεσίες. Δηλαδή, η παράκαμψη του επιπέδου ενδιάμεσου λογισμικού και η απευθείας κλήση υπηρεσιών ενός από τα υποκείμενα λειτουργικά συστήματα συνήθως αποδοκιμάζεται.

2.1.2 Μοντέλα ενδιάμεσου λογισμικού

Για να είναι η ανάπτυξη και η ολοκλήρωση των κατανεμημένων εφαρμογών όσο το δυνατόν απλούστερες, τα περισσότερα ενδιάμεσα λογισμικά βασίζονται σε κάποιο μοντέλο, ή υπόδειγμα (paradigm), για την περιγραφή της κατανεμημένης λειτουργίας και της επικοινωνίας. Ένα σχετικά απλό μοντέλο είναι να αντιμετωπίζονται τα πάντα ως αρχεία. Αυτή είναι η προσέγγιση που εμφανίστηκε αρχικά με το UNIX, και την οποία ακολούθησε με αυστηρότητα το Plan 9 (Pike, κ.ά.).

Μια παρόμοια προσέγγιση, αλλά λιγότερο αυστηρή από του Plan 9, ακολουθείται από το ενδιάμεσο λογισμικό που βασίζεται στα **κατανεμημένα συστήματα αρχείων** (distributed file systems). Σε πολλές περιπτώσεις, αυτό το ενδιάμεσο λογισμικό βρίσκεται στην πραγματικότητα ένα βήμα πέρα από τα δικτυακά λειτουργικά συστήματα, με την έννοια ότι η διαφάνεια της κατανεμημένης λειτουργίας υποστηρίζεται μόνο για τα παραδοσιακά αρχεία (δηλαδή για τα αρχεία που χρησιμοποιούνται απλώς για αποθήκευση δεδομένων). Για παράδειγμα, συχνά απαιτείται η ρητή εκκίνηση διεργασιών σε συγκεκριμένα μηχανήματα. Το ενδιάμεσο λογισμικό που βασίζεται στα κατανεμημένα συστήματα αρχείων έχει αποδειχτεί αρκετά επεκτάσιμο, πράγμα το οποίο συμβάλλει στη δημοτικότητά του.

Ένα άλλο σημαντικό πρώιμο μοντέλο ενδιάμεσου λογισμικού είναι εκείνο που βασίζεται στις **Κλήσεις Απομακρυσμένων Διαδικασιών** (Remote Procedure Calls, **RPC**). Το μοντέλο αυτό δίνει έμφαση στην απόκρυψη της δικτυακής επικοινωνίας, επιτρέποντας σε μια διεργασία να καλέσει μια διαδικασία η οποία μπορεί να υλοποιείται σε κάποιο απομακρυσμένο μηχάνημα. Όταν καλείται μια τέτοια διαδικασία, οι παράμετροι στέλνονται με διαφάνεια στο απομακρυσμένο μηχάνημα, όπου στη συνέχεια εκτελείται η διαδικασία, και μετά τα αποτελέσματα επιστρέφονται στην καλούσα διεργασία. Έτσι, φαίνεται σαν να εκτελέστηκε η κλήση διαδικασίας τοπικά: η καλούσα διεργασία δεν αντιλαμβάνεται το γεγονός ότι πραγματοποιήθηκε επικοινωνία μέσω δικτύου, εκτός ίσως από κάποια απώλεια απόδοσης.

Από τη στιγμή που ο αντικειμενοστραφής προγραμματισμός απέκτησε δημοτικότητα, έγινε φανερό ότι, αφού οι κλήσεις διαδικασιών μπορούσαν να ξεπερνούν τα όρια ενός μηχανήματος, θα έπρεπε επίσης να μπορούν να καλούνται με διαφανή τρόπο και αντικείμενα που βρίσκονται σε απομακρυσμένα μηχανήματα. Αυτό οδήγησε σε διάφορα συστήματα ενδιάμεσου λογισμικού που διαθέτουν την έννοια των **κατανεμημένων αντικειμένων** (distributed objects). Η ουσία των κατανεμημένων αντικειμένων είναι ότι κάθε αντικείμενο υλοποιεί μια διασύνδεση, η οποία κρύβει όλες τις εσωτερικές λεπτομέρειές του από τους χρήστες του. Μια διασύνδεση αποτελείται από τις μεθόδους που υλοποιεί το αντικείμενο, ούτε περισσότερες ούτε λιγότερες. Το μόνο πράγμα που μπορεί να δει μια διεργασία από ένα αντικείμενο είναι η διασύνδεσή του.

Τα κατανεμημένα αντικείμενα συχνά υλοποιούνται ως εξής: το κάθε αντικείμενο τοποθετείται σε ένα μηχάνημα και η διασύνδεσή του γίνεται διαθέσιμη σε πολλά άλλα μηχανήματα. Όταν μια διεργασία καλεί μια μέθοδο, η υλοποίηση της διασύνδεσης στο μηχάνημα όπου εκτελείται η διεργασία απλώς μετασχηματίζει την κλήση της μεθόδου σε ένα μήνυμα που στέλνεται στο αντικείμενο. Το αντικείμενο εκτελεί τη ζητούμενη μέθοδο και στέλνει πίσω το αποτέλεσμα. Κατόπιν, η υλοποίηση της διασύνδεσης μετασχηματίζει το μήνυμα της απάντησης σε μια επιστρεφόμενη τιμή, η οποία στη συνέχεια παραδίδεται στην καλούσα διεργασία. Όπως και στην περίπτωση των κλήσεων απομακρυσμένων διαδικασιών (RPC), η διεργασία, μπορεί να μην έχει καμία επίγνωση για το ότι πραγματοποιήθηκε επικοινωνία μέσω δικτύου.

Οι δυνατότητες που έχουν τα μοντέλα όσον αφορά την απλούστευση της χρήσης των δικτυακών συστημάτων μπορούν ίσως να γίνουν πιο κατανοητές με το παράδειγμα του παγκόσμιου ιστού (World Wide Web). Η επιτυχία του ιστού οφείλεται κυρίως στο εξαιρετικά απλό αλλά πολύ αποτελεσματικό μοντέλο **κατανεμημένων εγγράφων**. Στο μοντέλο του ιστού οι πληροφορίες είναι οργανωμένες σε έγγραφα, καθένα από τα οποία είναι αποθηκευμένο με διαφάνεια σε ένα μηχάνημα που βρίσκεται κάπου στον κόσμο. Τα έγγραφα περιέχουν συνδέσμους που παραπέμπουν σε άλλα έγγραφα. Αν ακολουθήσουμε ένα σύνδεσμο, το έγγραφο στο οποίο αυτός παραπέμπει προσκομίζεται από τη θέση που βρίσκεται και εμφανίζεται στην οθόνη μας. Η έννοια του εγγράφου δεν είναι απαραίτητο να περιορίζεται μόνο σε πληροφορίες που έχουν μορφή κειμένου. Για παράδειγμα, ο ιστός υποστηρίζει επίσης έγγραφα ήχου και βίντεο, καθώς και κάθε είδους αλληλεπιδραστικά έγγραφα με γραφικά.

2.1.3 Υπηρεσίες ενδιάμεσου λογισμικού

Μια σημαντική υπηρεσία, κοινή σε όλο το ενδιάμεσο λογισμικό, είναι η **ονομασία**. Οι υπηρεσίες ονομασίας επιτρέπουν τη κοινή χρήση και την αναζήτηση οντοτήτων (όπως σε καταλόγους), και είναι ανάλογες με τον τηλεφωνικό κατάλογο. Αν και η ονομασία μπορεί να φαίνεται απλή δουλειά με την πρώτη ματιά, παρουσιάζει δυσκολίες όταν ληφθεί υπόψη η επεκτασιμότητα. Προβλήματα προκαλούνται από το γεγονός ότι, για την αποδοτική αναζήτηση ενός ονόματος σε ένα σύστημα μεγάλης κλίμακας, η θέση της κατονομαζόμενης οντότητας πρέπει να θεωρείται σταθερή. Η παραδοχή αυτή γίνεται στον Παγκόσμιο Ιστό, όπου σήμερα κάθε έγγραφο ονομάζεται με τη χρήση μιας διεύθυνσης URL. Μια διεύθυνση URL περιέχει το όνομα του διακομιστή όπου είναι αποθηκευμένο το έγγραφο στο οποίο αναφέρεται το URL. Αν λοιπόν το έγγραφο μετακινηθεί σε άλλο διακομιστή, η διεύθυνση URL του παύει να ισχύει.

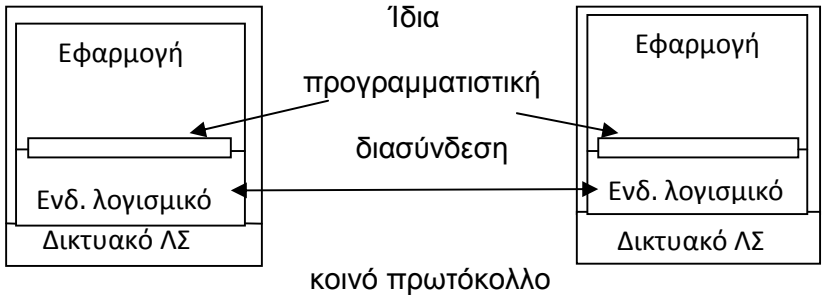
Επίσης, σχεδόν όλα τα συστήματα ενδιάμεσου λογισμικού που χρησιμοποιούνται σε μη πειραματικά περιβάλλοντα παρέχουν μηχανισμούς **ασφαλείας**. Σε σύγκριση με τα δικτυακά λειτουργικά συστήματα, το πρόβλημα με την ασφάλεια στο ενδιάμεσο λογισμικό είναι ότι θα πρέπει να είναι διεισδυτική. Το επίπεδο του ενδιάμεσου λογισμικού δεν μπορεί καταρχήν να βασίζεται στα υποκείμενα λειτουργικά συστήματα για την επαρκή υποστήριξη της ασφάλειας για ολόκληρο δίκτυο. Συνεπώς, η ασφάλεια πρέπει κατά ένα μέρος να υλοποιείται και πάλι στο ίδιο το επίπεδο ενδιάμεσου λογισμικού. Σε συνδυασμό με την ανάγκη για επεκτασιμότητα, η ασφάλεια έχει αποδειχτεί μία από τις δυσκολότερες στην υλοποίηση υπηρεσίες στα κατανεμημένα συστήματα.

2.1.4 Ενδιάμεσο λογισμικό και ανοιχτή λειτουργία

Τα σημερινά κατανεμημένα συστήματα γενικά κατασκευάζονται ως ενδιάμεσο λογισμικό για ένα εύρος από λειτουργικά συστήματα. Με αυτόν τον τρόπο, οι εφαρμογές που κατασκευάζονται για ένα συγκεκριμένο κατανεμημένο σύστημα γίνονται ανεξάρτητες από το λειτουργικό σύστημα. Δυστυχώς, αυτή η ανεξαρτησία συχνά αντισταθμίζεται από μια ισχυρή εξάρτηση από το συγκεκριμένο ενδιάμεσο λογισμικό. Προκαλούνται προβλήματα από το γεγονός ότι το ενδιάμεσο λογισμικό είναι συχνά λιγότερο ανοιχτό από όσο υποστηρίζει ο κατασκευαστής του.

Όπως είδαμε προηγουμένως, ένα αληθινά ανοιχτό κατανεμημένο σύστημα ορίζεται μέσω διασυνδέσεων οι οποίες πρέπει να είναι πλήρεις. ¨Πλήρεις¨ σημαίνει ότι όλα όσα χρειάζονται για την υλοποίηση του συστήματος έχουν πραγματικά οριστεί στις προδιαγραφές. Η μη-πληρότητα των ορισμών των διασυνδέσεων οδηγεί σε μια κατάσταση η οποία μπορεί να υποχρεώσει τους κατασκευαστές συστημάτων να προσθέσουν δικές τους διασυνδέσεις. Συνεπώς, είναι πιθανό να καταλήξουμε σε μια κατάσταση όπου δύο συστήματα ενδιάμεσου λογισμικού που προέρχονται από διαφορετικούς κατασκευαστές ακολουθούν το ίδιο πρότυπο, αλλά οι εφαρμογές που έχουν γραφεί για το ένα σύστημα δεν μπορούν εύκολα να μεταφερθούν στο άλλο.

Εξίσου κακή είναι η περίπτωση όπου η μη-πληρότητα οδηγεί σε μια κατάσταση κατά την οποία δύο διαφορετικές υλοποιήσεις δεν μπορούν ποτέ να συνεργαστούν, παρά το γεγονός ότι υλοποιούν ακριβώς το ίδιο σύνολο διασυνδέσεων αλλά διαφορετικά υποκείμενα πρωτόκολλα. Για παράδειγμα, αν δύο διαφορετικές υλοποιήσεις βασίζονται σε ασύμβατα πρωτόκολλα επικοινωνίας, τα οποία παρέχονται από το υποκείμενο δικτυακό λειτουργικό σύστημα, δεν υπάρχουν πολλές ελπίδες για εύκολη επίτευξη της διαλειτουργικότητας. Εκείνο που χρειάζεται είναι τα πρωτόκολλα του ενδιάμεσου λογισμικού και οι διασυνδέσεις του ενδιάμεσου λογισμικού να είναι ίδια, όπως φαίνεται στην Εικόνα (2.1).



Εικόνα (2.1) Σε ένα ανοιχτό κατανεμημένο σύστημα βασιζόμενο σε ενδιάμεσο λογισμικό, τα πρωτόκολλα που χρησιμοποιούνται από κάθε επίπεδο ενδιάμεσου λογισμικού, καθώς και οι διασυνδέσεις που παρέχουν για τις εφαρμογές, θα πρέπει να είναι ίδια.

Ας δούμε άλλο ένα παράδειγμα. Προκειμένου να εξασφαλιστεί η διαλειτουργικότητα μεταξύ διαφορετικών υλοποιήσεων, είναι απαραίτητο να γίνεται αναφορά στις οντότητες των διαφορετικών συστημάτων με τον ίδιο τρόπο. Αν στο ένα σύστημα γίνεται αναφορά στις οντότητες μέσω διευθύνσεων URL ενώ το άλλο σύστημα υλοποιεί τις αναφορές στις οντότητες χρησιμοποιώντας διευθύνσεις δικτύου, είναι φανερό ότι η συνεννόηση θα αποτελέσει πρόβλημα. Σε τέτοιες περιπτώσεις, οι ορισμοί των διασυνδέσεων θα πρέπει να προδιαγραφούν ποια ακριβώς μορφή θα έχουν οι αναφορές.

2.2 Ενδιάμεσο λογισμικό βασισμένο σε μήνυμα (MOM)

Πριν να προχωρήσουμε αξίζει να ρίξουμε μια πιο λεπτομερή ματιά σε μια γενική μορφή τεχνολογίας γνωστή ως ενδιάμεσο λογισμικό σε μήνυμα. Παρέχει ένα απλό παράδειγμα γενικού ενδιάμεσου λογισμικού.

Το ενδιάμεσο λογισμικό βασισμένο σε μήνυμα (MOM) είναι λογισμικό που διαχειρίζεται τις συναλλαγές μεταξύ πελατών-διακομιστών. Ένα τυπικό κομμάτι MOM περιλαμβάνει έναν αριθμό ουρών που περιέχουν μηνύματα και δεδομένα προερχόμενα είτε από έναν πελάτη είτε ένα διακομιστή. Οι πελάτες – π.χ. ένας πωλητής που χρησιμοποιεί έναν φορητό υπολογιστή - καταθέτουν συνεχώς αιτήματα στο MOM και ο διακομιστής θα τα παραλάβει, θ' ανταποκριθεί σ' αυτά και θα καταθέσει όσα δεδομένα πρέπει ν' αποσταλούν πίσω στο MOM. Το ενδιάμεσο λογισμικό μεσολαβεί μεταξύ των πελατών και διακομιστών. Αυτό το λογισμικό έχει ορισμένα προτερήματα: οι πελάτες δεν χρειάζεται να είναι συνέχεια συνδεδεμένοι - μπορούν απλά να τοποθετήσουν δεδομένα στο ενδιάμεσο λογισμικό και ν' αποσυνδεθούν, συλλέγοντας τις απαντήσεις αργότερα. Επίσης, το μοντέλο διεπαφής είναι απλό: αποτελείται μόνο από τον πελάτη που τοποθετεί δεδομένα σε μια ουρά που συντηρείται από το ενδιάμεσο λογισμικό και από τον διακομιστή που απομακρύνει τα δεδομένα, τα επεξεργάζεται και τοποθετεί μία απάντηση πίσω στην ουρά. Χάρη σ' αυτήν την απλότητα το MOM χρησιμοποιείται συχνά για την σύνδεση λογισμικού με συστήματα κληροδότησης.

2.2.1 Σειρές MQ

Χωρίς αμφιβολία το κυριότερο ενδιάμεσο λογισμικό βασισμένο σε μήνυμα είναι οι **Σειρές MQ** που αναπτύχθηκε από την IBM. Επεξεργάζεται τέσσερεις τύπους μηνυμάτων: Τα ανεξάρτητα δοτοδέματα (Datagrams) είναι μονόδρομα μηνύματα χωρίς την επιστροφή απαντητικού μηνύματος, π.χ. το σήμα ότι ένας πελάτης είναι συνδεδεμένος. Τα μηνύματα αίτησης (Request Messages) χρησιμοποιούνται όταν ο αποστολέας περιμένει απάντηση, π.χ. όταν στέλνει ένα ερώτημα στην βάση δεδομένων. Τα απαντητικά μηνύματα (Reply Messages) είναι τα μηνύματα που αποστέλλονται ως απάντηση σε μηνύματα αίτησης. Και, τέλος, τα μηνύματα αναφοράς (Report Messages) χρησιμοποιούνται για να δώσουν σήμα σ' έναν πελάτη ότι κάποιο αναπάντεχο γεγονός έχει συμβεί. Για παράδειγμα, ότι ένας διακομιστής έχει βλάβη. Το API των *MQSeries* είναι εξαιρετικά απλό και αποτελείται από μόνο 11 κλήσεις προγράμματος.

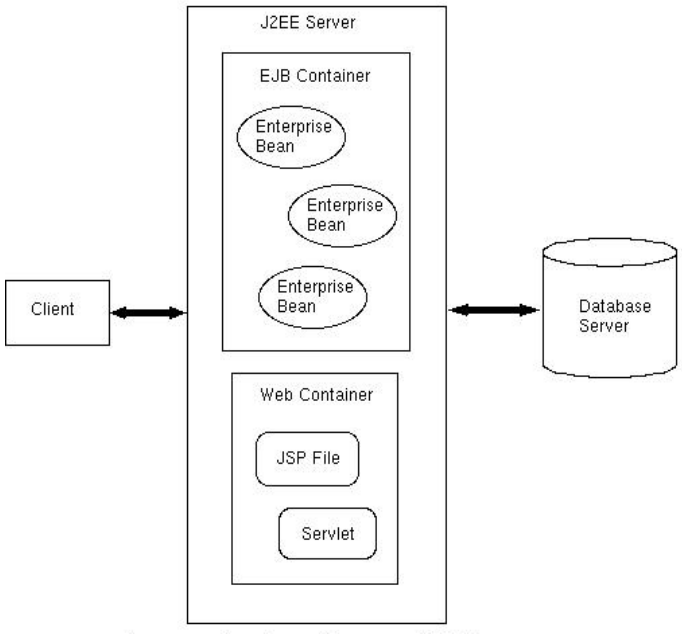
2.3 Τεχνολογίες Component Middleware

Οι σημερινές τεχνολογίες Component middleware που επικρατούν είναι τρεις:

A) EJB (Enterprise Java Beans)

Η εταιρία Sun που εκδίδει την Java και την Java virtual machine, ονομάζει την τεχνολογία αυτή ως Java 2 Standard Edition. Για να συμπεριλάβει λειτουργικότητα συνιστωσών επεκτείνει την Java και ορίζει την έκδοση Java 2 Enterprise Edition. Βασικά δομικά στοιχεία του λογισμικού στην έκδοση αυτή είναι τα Enterprise Java Beans (EJB). Για την υποστήριξη του μοντέλου αυτού παρέχεται ένας διακομιστής που ονομάζεται J2EE server. Αυτός λειτουργεί ως ενδιάμεσος μεταξύ συστημάτων client-server. Η απαίτηση η ο οποία τίθεται από την J2EE είναι ότι ο J2EE server τρέχει πάνω από java virtual machine και οι συνιστώσες να είναι υλοποιημένες σε Java. Ο J2EE server παρέχει υπηρεσίες όπως δοσοληψιών, ασφάλειας και πρόσβασης σε βάσεις δεδομένων. Οι συνιστώσες χρησιμοποιούν τις υπηρεσίες αυτές μέσω των container μέσα στα οποία βρίσκονται και αποτελούν το περιβάλλον εκτέλεσής τους. [4],[5].

Στην Εικόνα (2.2) φαίνεται σχηματικά η αρχιτεκτονική J2EE



Εικόνα (2.2) Η επικοινωνία client-server μέσω J2EE

Β) CCM (CORBA Component Model)

Το CCM είναι μια επέκταση του CORBA2.x και πολλές φορές αναφέρεται και ως CORBA3.x. οι υλοποιήσεις του είναι ως επί το πλείστον ανοιχτού κώδικα και είναι το πλέον χρησιμοποιούμενο σε κρίσιμες εφαρμογές (mission critical).

Γ) COM+ (Component Object Model)

Το COM+ είναι η επέκταση του COM σε συνδυασμό με το DCOM της Microsoft ώστε να μπορεί να υποστηρίζεται και απομακρυσμένη πρόσβαση στα αντικείμενα. Αν και εντάσσεται ως μοντέλο συνιστωσών σε αυτή τη λίστα, δεν είναι διαλειτουργικό με την ευρεία έννοια. Γενικά προϋποθέτει το λειτουργικό σύστημα που βρίσκεται από κάτω να είναι Windows. Αυτό το καθιστά μη χρήσιμο σε συστήματα που διαθέτουν άλλα λειτουργικά. Ακόμα η υλοποίησή του είναι κλειστού τύπου.

ΚΕΦΑΛΑΙΟ 3: ΚΑΤΑΝΕΜΗΜΕΝΑ ΑΝΤΙΚΕΙΜΕΝΑ

3.1 Συστήματα κατανεμημένων αντικειμένων

Τα κατανεμημένα αντικείμενα αποτελούν ένα ισχυρό εργαλείο το οποίο έγινε ευρέως διαθέσιμο στους προγραμματιστές τα τελευταία χρόνια. Το πλεονέκτημα της κατανομής των αντικειμένων δεν έγκειται στο γεγονός ότι αυτά βρίσκονται μοιρασμένα στο δίκτυο, αλλά στο ότι κάθε τμήμα ενός προγράμματος μπορεί απ' ευθείας να αλληλεπιδράσει με αυτά. Με άλλα λόγια αποτελεί ένα εργαλείο για την διεύρυνση των πόρων των κατανεμημένων συστημάτων με τρόπο ακριβή και αποδοτικό.

Στη συνέχεια θα δούμε τα κίνητρα που οδήγησαν στην ανάπτυξη αυτής της τεχνολογίας, τους λόγους της μεγάλης τους χρησιμότητας καθώς και τα χαρακτηριστικά ενός πετυχημένου σχεδίου κατανεμημένων αντικειμένων. Επίσης θα αναλύσουμε μερικά από τα πιο πετυχημένα σχήματα κατανεμημένου προγραμματισμού.[14]

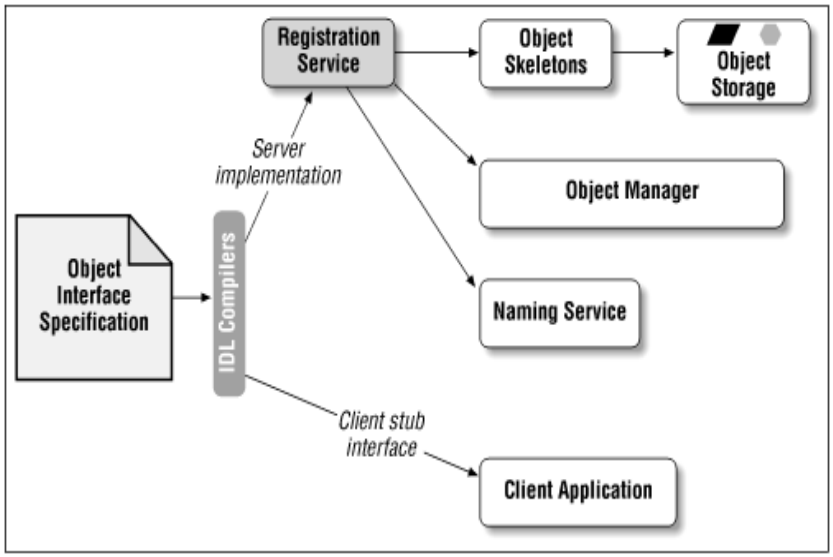
3.1.1 Γιατί κατανομή αντικειμένων;

Κατά την ανάπτυξη κατανεμημένων εφαρμογών, δημιουργείται η ανάγκη της κατανομής των δεδομένων και των λειτουργιών με τρόπο ελεύθερο και διάφανο, έτσι ώστε αυτά να επηρεάζονται μόνο από την δομή της εφαρμογής και όχι από την δομή του δικτύου. Αυτή την ανάγκη προσπαθούν να καλύψουν τα συστήματα κατανεμημένων αντικειμένων, δίνοντας στους προγραμματιστές την δυνατότητα να τοποθετούν και να εκτελούν τα αντικείμενα τους, όχι μόνο τοπικά, αλλά και σε απομακρυσμένους υπολογιστές. Κύριος στόχος αυτής της τεχνολογίας είναι να επιτρέψει στις εφαρμογές την πρόσβαση σε αντικείμενα τα οποία βρίσκονται στο δίκτυο, με τον ίδιο τρόπο που θα γινόταν η πρόσβαση εάν αυτά ήταν στον τοπικό υπολογιστή.

Η ανάγκη για κατανομή των αντικειμένων είναι περισσότερο προφανής όσο το μέγεθος των εφαρμογών μεγαλώνει. Σε μικρές εφαρμογές όπου χρειάζονται λίγοι τύποι αντικειμένων, είναι εύκολο να δημιουργηθεί ένας κατάλογος με μηνύματα, τα οποία θα επιτρέπουν σε απομακρυσμένους πράκτορες (remote agents) να επικοινωνήσουν. Σε μια μεγάλη εφαρμογή όμως, ο κατάλογος αυτός μεγαλώνει δραματικά, κάνοντας ο έργο του προγραμματιστή πολύπλοκο και λιγότερο αξιόπιστο, ενώ η επέκταση και η συντήρηση του προγράμματος με την προσθήκη νέων αντικειμένων γίνεται υπερβολικά σύνθετη.

3.1.2 Χαρακτηριστικά των συστημάτων κατανεμημένων αντικειμένων

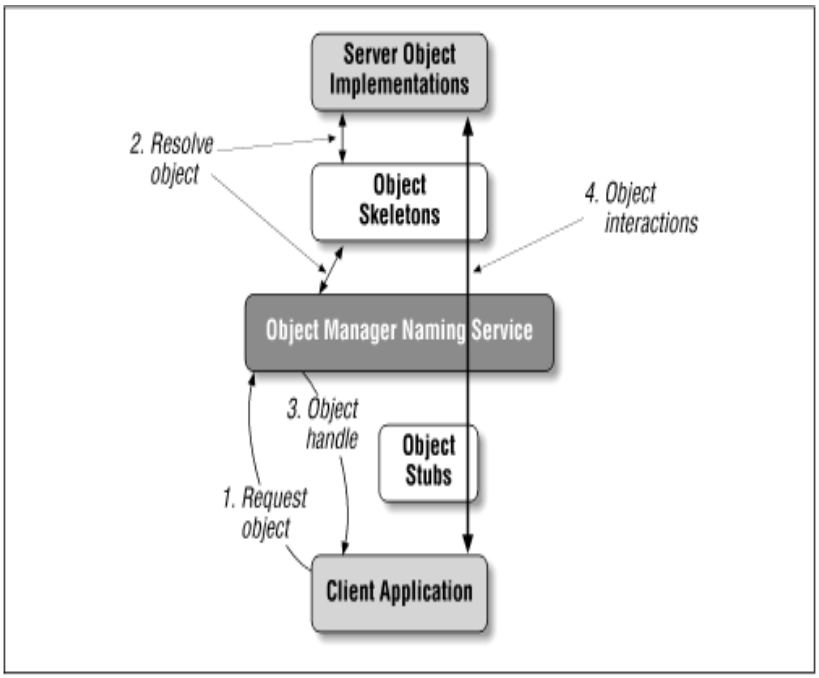
Η κύρια απαίτηση από ένα σύστημα κατανεμημένων αντικειμένων είναι η δυνατότητα της δημιουργίας απομακρυσμένων αντικειμένων και της επίδρασης πάνω σε αυτά. Στην Εικόνα (3.1) παριστάνεται τα χαρακτηριστικά δομικά στοιχεία που αποτελούν ένα τέτοιο σύστημα. Μια προδιαγραφή διασύνδεσης αντικειμένου (Object Interface Specification) χρησιμοποιείται για την υλοποίηση μιας κλάσης αντικειμένων: μιας διασύνδεσης μεταξύ της υλοποίησης του αντικειμένου και του διαχειριστή αντικειμένων (object manager) και μιας διασύνδεσης πελάτη (client interface). Η πρώτη καλείται συχνά σκελετός (skeleton) και η δεύτερη στέλεχος (stub). Ο σκελετός χρησιμοποιείται από τον εξυπηρετητή (server) για την δημιουργία στιγμιότυπων των αντικειμένων και για τη δρομολόγηση κλήσεων στην υλοποίηση του αντικειμένου. Το στέλεχος χρησιμοποιείται από τον πελάτη (client) για την δρομολόγηση αιτήσεων στο αντικείμενο του εξυπηρετητή. Στη πλευρά του εξυπηρετητή η υλοποιημένη κλάση καταχωρείται σε μια υπηρεσία εγγραφής (registration service), η οποία χρησιμοποιεί μια υπηρεσία ονοματοδοσίας (naming service) και ένα διαχειριστή αντικειμένων για την αποθήκευση του αντικειμένου.



Εικόνα (3.1)Χαρακτηριστικά των συστημάτων κατανεμημένων αντικειμένων

Μετά την καταχώρηση του αντικειμένου στον εξυπηρετητή, ο πελάτης μπορεί να ζητήσει ένα στιγμιότυπο της κλάσης χρησιμοποιώντας την υπηρεσία ονοματοδοσίας.

Κατά την διάρκεια του χρόνου εκτέλεσης (run-time) η υπηρεσία ονοματοδοσίας δρομολογεί τις αιτήσεις του πελάτη στον διαχειριστή αντικειμένων του εξυπηρετητή, ο οποίος με τη σειρά του αρχικοποιεί το νέο αντικείμενο χρησιμοποιώντας τον αποθηκευμένο σκελετό. Το νέο αντικείμενο αποθηκεύεται στην αποθήκη αντικειμένων (object storage), ενώ στον πελάτη παραδίδεται μια αντιστοίχηση για το αντικείμενο με τη μορφή μιας διασύνδεσης στελέχους. Το στέλεχος χρησιμοποιείται από τον πελάτη για να αλληλεπιδρά με το αντικείμενο. Οι επικοινωνίες που συμβαίνουν κατά το χρόνο εκτέλεσης (run-time transactions) παρουσιάζονται στην Εικόνα(3.2).



Εικόνα (3.2) run-time transactions

3.2 Εισαγωγή στο Κατανεμημένο COM

Η βάση του DCOM αποτελείται από την τεχνολογία αντικειμένων συστατικών στοιχείων της Microsoft, δηλαδή το COM. Στόχος του COM είναι η υποστήριξη της ανάπτυξης συστατικών στοιχείων τα οποία μπορούν να ενεργοποιούνται δυναμικά και να αλληλεπιδρούν μεταξύ τους. Ένα στοιχείο του COM αποτελεί ο εκτελέσιμος κώδικας, ο οποίος περιέχεται είτε σε μια βιβλιοθήκη (που μπορεί να συνδεθεί δυναμικά) είτε με τη μορφή εκτελέσιμου προγράμματος.

Το ίδιο το COM παρέχεται με τη μορφή βιβλιοθήκης η οποία συνδέεται σε μια διεργασία. Αρχικά αναπτύχθηκε για την υποστήριξη των λεγόμενων **σύνθετων εγγράφων** (compound documents). Ένα σύνθετο έγγραφο είναι ένα έγγραφο που αποτελείται από πολύ διαφορετικά μέρη, όπως (μορφοποιημένο) κείμενο, εικόνες, λογιστικά φύλλα κ.ο.κ. Ο χειρισμός κάθε μέρους γίνεται με την εφαρμογή με την οποία αυτό είναι σχετισμένο.

Έχοντας ως στόχο την υποστήριξη χιλιάδων σύνθετων εγγράφων, η Microsoft χρειαζόταν ένα γενικό τρόπο για να ξεχωρίσει τα διάφορα μέρη και να είναι σε θέση να τα ¨συγκολλάει¨. Αυτό οδήγησε αρχικά στην τεχνολογία **OLE**, που σημαίνει **Σύνδεση και Ενσωμάτωση Αντικειμένων** (Object Linking and Embedding). Η πρώτη έκδοση OLE χρησιμοποιούσε έναν πρωτόγονο και άκαμπτο τρόπο μεταβίβασης πληροφοριών μεταξύ των διαφορετικών μερών. Σύντομα η έκδοση αυτή αντικαταστάθηκε από μια νεότερη (που ονομάζεται επίσης OLE), η οποία δημιουργήθηκε όμως επάνω από ένα πιο ευέλικτο επίπεδο, το COM, με αποτέλεσμα την οργάνωση που μπορείτε να δείτε στην Εικόνα(3.3).

Η Εικόνα (3.3) περιέχει επίσης τον όρο **ActiveX**, ο οποίος χρησιμοποιείται πλέον για να καλύψει αυτό που προηγουμένως ονομαζόταν OLE, μαζί με κάποια νέα χαρακτηριστικά. Αυτά τα νέα χαρακτηριστικά περιλαμβάνουν κυρίως ευέλικτες δυνατότητες για την εκκίνηση συστατικών στοιχείων σε διάφορες διεργασίες, υποστήριξη για σενάρια, και μια λίγο-πολύ τυποποιημένη ομαδοποίηση αντικειμένων στα λεγόμενα χειριστήρια ActiveX. Φαίνεται ότι όλοι οι ειδικοί του DCOM (ακόμη και μέσα στη Microsoft) συμφωνούν ομόφωνα ότι δεν υπάρχει κάποια σαφής περιγραφή του τι ακριβώς είναι το ActiveX. Κατά συνέπεια, ούτε εμείς θα επιχειρήσουμε να το ορίσουμε.

Εικόνα (3.3) Η γενική οργάνωση του ActiveX, του OLE, και του COM.

Αυτό που προσθέτει το DCOM σε αυτή την οργάνωση είναι η δυνατότητα επικοινωνίας μια διεργασίας με συστατικά στοιχεία που είναι τοποθετημένα σε κάποιο άλλο μηχάνημα. Ωστόσο, οι βασικοί μηχανισμοί για την ανταλλαγή πληροφοριών μεταξύ στοιχείων όπως παρέχονται από το COM, συχνά είναι ακριβώς ίδιοι και για το DCOM. Με άλλα λόγια, η διαφορά μεταξύ COM και DCOM για έναν προγραμματιστή συχνά κρύβεται πίσω από διάφορες διασυνδέσεις. Το DCOM παρέχει κυρίως διαφάνεια πρόσβασης, λιγότερο εμφανείς είναι άλλες μορφές διαφάνειας κατανομής.

3.2.1 Επισκόπηση του DCOM

Ένα κατανεμημένο σύστημα που βασίζεται σε αντικείμενα είναι το **Κατανεμημένο COM** (Distributed COM, **DCOM**) της Microsoft. Όπως υποδηλώνει το όνομά του, το DCOM προήλθε από το COM, που σημαίνει **Μοντέλο Αντικειμένων Συστατικών Στοιχείων** (Component Object Model). Το COM αποτελεί την τεχνολογία στην οποία βασίζονται τα διάφορα λειτουργικά συστήματα Windows που δημιούργησε η Microsoft με αφετηρία τα Windows 95. Σε αντίθεση με το CORBA (θα αναφέρουμε παρακάτω), το DCOM δεν είναι έργο κάποιας επιτροπής. Δυστυχώς, το γεγονός ότι το DCOM κρατήθηκε μακριά από επιτροπές δεν οδήγησε σε μια καλά συγκροτημένη αρχιτεκτονική και σχεδιασμό, τα οποία θα αποτελούνται από ένα μικρό μόνο σύνολο βασικών στοιχείων που θα χρησιμοποιούνται για την κατασκευή συστατικών μερών και υπηρεσιών. Αντίθετα, το DCOM σήμερα αποτελεί ένα περίπλοκο σύστημα, στο οποίο πολλά παρόμοια πράγματα μπορούν να γίνουν με πολλούς διαφορετικούς τρόπους, και είναι τέτοιο ώστε η συνύπαρξη διαφορετικών λύσεων μερικές φορές είναι αδύνατη.

Δεν είναι δύσκολο να επικρίνει κάποιος το DCOM. Ωστόσο, σε σύγκριση με το CORBA, θα ήταν δίκαιο να υποστηρίξουμε ότι το DCOM είναι μια τεχνολογία η οποία σε ορισμένο βαθμό έχει αποδείξει την αξία της. Με δεκάδες εκατομμύρια ανθρώπους να χρησιμοποιούν τα Windows καθημερινά σε δικτυακά περιβάλλοντα, το DCOM έχει φθάσει να χρησιμοποιείται ευρέως. Με αυτήν την έννοια, το CORBA ή οποιοδήποτε άλλο κατανεμημένο σύστημα έχουν ακόμα πολύ δρόμο μπροστά τους.

3.2.2 Μοντέλο αντικειμένων DCOM

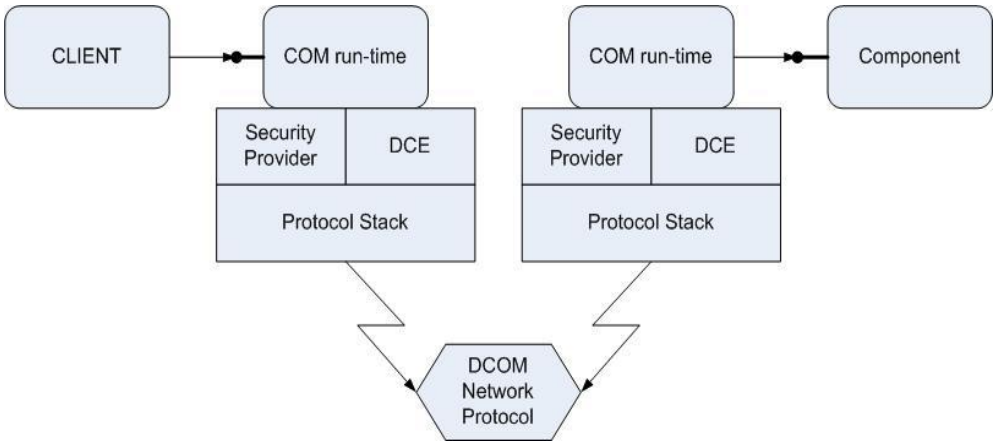
Όπως όλα σχεδόν τα άλλα κατανεμημένα συστήματα που βασίζονται σε αντικείμενα, το DCOM υιοθετεί το μοντέλο απομακρυσμένων αντικειμένων. Για την ακρίβεια, τα αντικείμενα στο DCOM μπορούν να τοποθετηθούν στην ίδια διεργασία με τον πελάτη, σε μια διεργασία στο ίδιο μηχάνημα, ή σε μια διεργασία σε απομακρυσμένο μηχάνημα.

Το μοντέλο αντικειμένων DCOM επικεντρώνεται στην υλοποίηση διασυνδέσεων. Χονδρικά, ένα αντικείμενο DCOM αποτελεί απλώς μια υλοποίηση κάποιας διασύνδεσης. Ένα μοναδικό αντικείμενο μπορεί να υλοποιεί πολλές διασυνδέσεις ταυτόχρονα. Σε αντίθεση με το CORBA, ωστόσο, το DCOM διαθέτει μόνο **δυαδικές συνδέσεις** (binary interfaces). Μια διασύνδεση τέτοιου είδους είναι ουσιαστικά ένας πίνακας με δείκτες πριν τις υλοποιήσεις των μεθόδων που αποτελούν μέρος αυτής της διασύνδεσης. Φυσικά, για τον ορισμό των διασυνδέσεων είναι βολική η χρήση μιας ξεχωριστής Γλώσσας Ορισμού Διασυνδέσεων (Interface Definition Language, IDL). Το DCOM διαθέτει IDL τέτοιου είδους, την **Microsoft IDL** (**MIDL**), από την οποία να δημιουργηθεί η τυπική διάταξη για δυαδικές διασυνδέσεις.

Το πλεονέκτημα της χρήσης δυαδικών διασυνδέσεων είναι ότι οι διασυνδέσεις είναι ανεξάρτητες από τη γλώσσα προγραμματισμού. Στην περίπτωση του CORBA, κάθε φορά που πρέπει να υποστηριχθεί μια άλλη γλώσσα προγραμματισμού, η αντιστοίχιση των προδιαγραφών IDL σε αυτή τη γλώσσα πρέπει να τυποποιείται. Μια τυποποίηση τέτοιου είδους δεν είναι απαραίτητη στην περίπτωση των δυαδικών διασυνδέσεων.

3.2.3 Distributed Component Object Model (DCOM)

Την επικοινωνία μεταξύ των διαδικασιών την έχει αναλάβει πλέον κάποιο πρωτόκολλο δικτύου, κάτι το οποίο δεν έχει αντιληφθεί κανένα από τα δύο άκρα της ζεύξης. Στην Εικόνα (3.4) φαίνεται η γενικότερη αρχιτεκτονική του DCOM: Το COM χρόνου εκτέλεσης (run-time COM) προσφέρει αντικειμενοστραφή υπηρεσίες σε πελάτες και τμήματα, ενώ χρησιμοποιεί το RPC και έναν security provider για να δημιουργήσει πακέτα δικτύου συμβατά με τις προδιαγραφές του πρωτοκόλλου του DCOM.



Εικόνα(3.4) Αρχιτεκτονική του DCOM

Οι εφαρμογές που βασίζονται στο DCOM μπορούν επιπλέον να εκμεταλλεύονται πλήρως τους μηχανισμούς ασφαλείας των Windows NT. Χρησιμοποιώντας το μοντέλο ασφαλείας (security model) που προσφέρει το API των NT, οι προγραμματιστές μπορούν να παρέχουν ασφαλείς και εξουσιοδοτημένες επικοινωνίες μέσω Windows Domain. Επίσης το εργαλείο DCOMCNF δίνει τη δυνατότητα του πλήρους καθορισμού του ελέγχου πρόσβασης για κάθε τμήμα του προγράμματος, επιτρέποντας έτσι τη δημιουργία πολύπλοκων, πολύ-χρηστικών (multi-user), πολύ-λειτουργικών (multi-role) εφαρμογών με πολλαπλά επίπεδα ασφαλείας. Οι υλοποιήσεις με DCOM παρουσιάζουν τρία σοβαρά μειονεκτήματα: πολύπλοκη διαδικασία ανάπτυξης, εξάρτηση από την πλατφόρμα (platform dependent) και προβλήματα ασφαλείας των ActiveX.

Ένα από τα μεγαλύτερα παράπονα γύρω από το DCOM είναι το γεγονός ότι, ανεξάρτητα από την γλώσσα προγραμματισμού, απαιτείται μεγάλη προσπάθεια και πολύ σύνθετη μεθοδολογία. Η Visual J++ απλοποιεί λίγο την κατάσταση με το να κάνει τα αντικείμενα DCOM να μοιάζουν με κλάσεις της Java. Παράλληλα οι αρχιτεκτονικές που βασίζονται στο DCOM παρουσιάζουν ισχυρούς δεσμούς με τις πλατφόρμες Windows NT. Αυτό είναι μεγάλο εμπόδιο για όποια αρχιτεκτονική δεν ακολουθεί πλήρως τις γραμμές της Microsoft, και γίνονται μεγάλες προσπάθειες να γεφυρωθεί το DCOM και με άλλα λειτουργικά συστήματα. Ενδεχομένως οι διαφορές αυτές θα απαλειφθούν και το DCOM θα υποστηρίζει πολλές πλατφόρμες, παρέχοντας αξιόπιστες λύσεις για την ανάπτυξη ετερογενών κατανεμημένων συστημάτων. Το τελευταίο αγκάθι στην πλευρά του DCOM είναι η κριτική που δέχεται για την έλλειψη ασφάλειας των τμημάτων ActiveX. Αυτά τα τμήματα υπό δεδομένες συνθήκες μπορούν να προκαλέσουν μεγάλα προβλήματα στους υπολογιστές-πελάτες (clients), αφού δεν περιέχονται σε κάποιο αυστηρά καθορισμένο πλαίσιο ασφαλείας όπως για παράδειγμα η εναλλακτική τους λύση, τα applets.

Ένας σημαντικός παράγοντας που εγγυάται το μέλλον του DCOM είναι η μετακίνηση του από την ιδιοκτησία της Microsoft σε έναν οργανισμό τυποποίησης, το The Open Group (TOG). Στόχοι αυτής της ομάδας είναι:

- η προώθηση της διαθεσιμότητας και συμβατότητας σε πολλές αρχιτεκτονικές και συστήματα της τεχνολογίας ActiveX
- ενίσχυση της διαλειτουργικότητας των ActiveX με τη χρήση του Distributed Computing Environment (DCE)
- επιτάχυνση της εξέλιξης της τεχνολογίας των ActiveX μέσω της διαδικασίας της συνεργαζόμενης ανάπτυξης.

Το - υπό την επίβλεψη του TOG πλέον - DCOM θα έχει μεγάλη επίδραση στην εξέλιξη της τεχνολογίας των κατανεμημένων αντικειμένων. Ο ανταγωνισμός του με την αρχιτεκτονική CORBA οδηγεί στην ανάγκη για ανάπτυξη περισσότερο πολύπλευρων και σταθερών τεχνολογιών, οι οποίες θα ανταποκρίνεται στις απαιτήσεις και τα οράματα μιας ευρύτερης κοινότητας.

3.3 Επισκόπηση του Globe

Το Globe είναι ένα κατανεμημένο σύστημα που βασίζεται σε αντικείμενα, στο οποίο η επεκτασιμότητα παίζει κεντρικό ρόλο. Κατά το σχεδιασμό του Globe λήφθηκαν υπόψη όλες οι πλευρές που αφορούν την κατασκευή ενός μεγάλης κλίμακας συστήματος ευρείας περιοχής, το οποίο μπορεί να υποστηρίξει πολύ μεγάλους αριθμούς χρηστών και αντικειμένων. Θεμελιώδης στην προσέγγιση αυτή είναι ο τρόπος με τον οποίο γίνονται αντιληπτά τα αντικείμενα. Όπως συμβαίνει και σε άλλα συστήματα με βάση αντικείμενα, τα αντικείμενα του Globe θα πρέπει να ενθυλακώνουν πληροφορίες κατάστασης, καθώς και τις λειτουργίες που εφαρμόζονται σε αυτές.

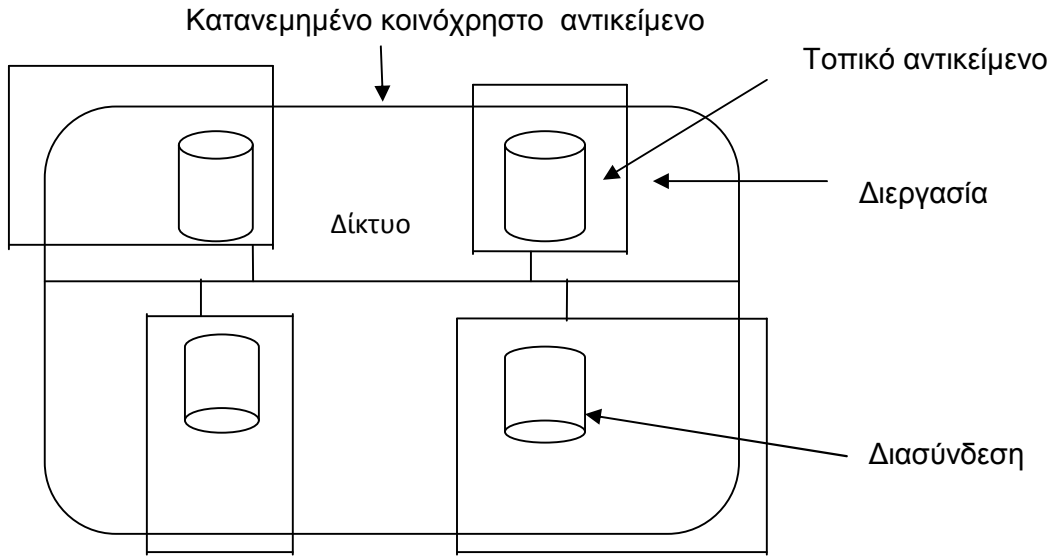
Μια σημαντική διάφορα από αλλά σύστημα που βασίζονται σε αντικείμενα, ιδιαίτερα από εκείνα που στοχεύουν επίσης σε επεκτασιμότητα ευρείας περιοχής, όπως το Legion [17], είναι ότι τα αντικείμενα θα πρέπει επίσης να ενθυλακώνουν την υλοποίηση πολιτικών οι οποίες υπαγορεύουν την κατανομή της κατάστασης των αντικειμένων σε πολλά μηχανήματα. Με άλλα λόγια, κάθε αντικείμενο καθορίζει πως θα κατανεμηθεί η κατάσταση του στα διάφορα αντίγραφά του. Κάθε αντικείμενο ελέγχει επίσης τις δικές του πολιτικές σε άλλες περιοχές.

Σε γενικές γραμμές, ο έλεγχος στο Globe ανατίθεται όσο το δυνατό περισσότερο στα αντικείμενα. Για παράδειγμα, ένα αντικείμενο αποφασίζει πώς, πότε, και πού θα πρέπει να μεταναστεύσει η κατάστασή του. Επίσης, ένα αντικείμενο αποφασίζει αν η κατάστασή του θα πρέπει να αναπαραχθεί, και αν ναι, πώς θα πρέπει να πραγματοποιηθεί η αναπαραγωγή. Ακόμα, ένα αντικείμενο μπορεί επίσης να καθορίσει την πολιτική ασφαλείας του και την υλοποίησή της. Στη συνέχεια θα περιγράψουμε τον τρόπο που επιτυγχάνεται μια τέτοια ενθυλάκωση.

3.3.1 Μοντέλο αντικειμένων Globe

Σε αντίθεση με τα περισσότερα από τα άλλα κατανεμημένα συστήματα που βασίζονται σε αντικείμενα, το Globe δεν υιοθέτει το μοντέλο απομακρυσμένων αντικειμένων. Αντίθετα, τα αντικείμενα του Globe μπορούν να είναι κατανεμημένα από φυσική άποψη, με την έννοια ότι η κατάσταση ενός αντικείμενου μπορεί να κατανεμηθεί και να αναπαραχθεί σε πολλές διεργασίες. Αυτή η οργάνωση παρουσιάζεται στην Εικόνα (3.5), η όποια δείχνει ένα αντικείμενο που έχει κατανεμηθεί σε τέσσερις διεργασίες, κάθε μία από τις όποιες εκτελείται και σε διαφορετικό μηχάνημα. Τα αντικείμενα στο Globe αναφέρονται ως **κατανεμημένα κοινόχρηστα αντικείμενα** (distributed shared objects), ώστε να φαίνεται το γεγονός ότι ένα αντικείμενο κανονικά είναι κοινόχρηστο μεταξύ πολλών διεργασιών.

GLOBE



Εικόνα (3.5) Η οργάνωση ενός κατανεμημένου κοινόχρηστου αντικείμενου στο Globe.

Μια διεργασία που συνδέεται με ένα κατανεμημένο κοινόχρηστο αντικείμενο έχει στη διάθεσή της μια τοπική υλοποίηση των διασυνδέσεων που παρέχονται από το αντικείμενο αυτό. Αυτές οι τοπικές υλοποιήσεις λέγονται **τοπικοί αντιπρόσωποι** (local representatives) ή απλώς **τοπικά αντικείμενα.** Βασικά, το αν είναι τοπικό αντικείμενο διαθέτει ή όχι κατάσταση είναι εντελώς διάφανες στη συνδυασμένη διεργασία. Όλες οι λεπτομέρειες υλοποίησης ενός αντικείμενου κρύβονται πίσω από τις διασυνδέσεις που αυτό παρέχει στις διεργασίες.

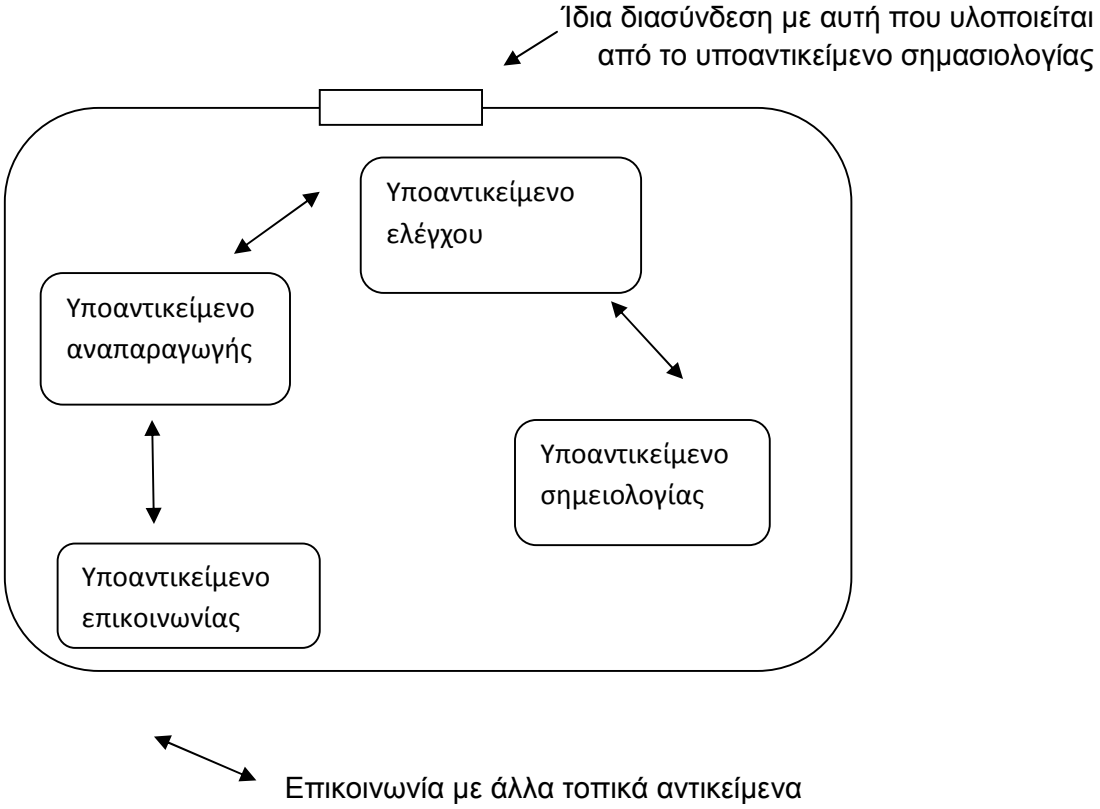
Κάθε τοπικό αντικείμενο υλοποιεί μια τυπική διασύνδεση αντικειμένων με όνομα SOInf, η οποία είναι παρόμοια με τη διασύνδεση IUnkown που χρησιμοποιείται στο DCOM. Πιο συγκεκριμένα η SOInf παρέχει μια μέθοδο, η όποια δέχεται ως είσοδο το αναγνωριστικό μιας διασύνδεσης και επιστρέφει ένα δείκτη προς αυτήν, με την προϋπόθεση ότι η διασύνδεση υλοποιείται από το αντικείμενο. Μεταξύ των τοπικών αντικειμένων του Globe και των αντικειμένων του DCOM υπάρχουν και άλλες ομοιότητες. Για παράδειγμα, κάθε τοπικό αντικείμενο θεωρείται ότι έχει ένα σχετικό αντικείμενο κλάσης, από το οποίο μπορούν να δημιουργηθούν νέα τοπικά αντικείμενα.

Τα τοπικά αντικείμενα θεωρείται ότι υλοποιούν δυαδικές διασυνδέσεις οι οποίες στην ουσία αποτελούνται από πίνακες δεικτών συναρτήσεων. Ο καθορισμός των διασυνδέσεων γίνεται με μια Γλώσσα Ορισμού Διασυνδέσεων, η όποια γενικά μοιάζει με αυτές που χρησιμοποιούνται στο CORBA και το DCOM, χωρίς ωστόσο να λείπουν και σημαντικές διαφορές σε κάποια σημεία.

Τα οπτικά αντικείμενα Globe είναι δυο ειδών. Ένα **στοιχειώδες τοπικό αντικείμενο** δεν περιέχει κανένα άλλο τοπικό αντικείμενο. Αντίθετα, ένα **σύνθετο τοπικό αντικείμενο** αποτελείται από πολλά (πιθανώς σύνθετα) τοπικά αντικείμενα. Η σύνθεση αντικειμένων πραγματοποιείται μέσω πινάκων διασύνδεσης των τοπικών αντικειμένων Globe, οι όποιοι αποτελούνται από ζεύγη δεικτών (κατάσταση, μέθοδος). Κάθε δείκτης κατάστασης αναφέρεται σε δεδομένα που ανήκουν σε ένα συγκεκριμένο τοπικό αντικείμενο.

Για τη διασύνδεση ενός στοιχειώδους τοπικού αντικείμενου, όλοι οι δείκτες κατάστασης αναφέρονται στα ίδια δεδομένα που αναπαριστούν την κατάσταση αυτού του αντικείμενου. Ενώ στην περίπτωση της σύνθεσης, οι δείκτες κατάστασης αναφέρονται στην κατάσταση των διαφορετικών αντικειμένων τα οποία αποτελούν τη σύνθεση.

Η σύνθεση χρησιμοποιείται για την δημιουργία τοπικών αντικειμένων τα οποία είναι απαραίτητα για την υλοποίηση κατανεμημένων κοινόχρηστων αντικειμένων. Ένα τέτοιο τοπικό αντικείμενο, το οποίο αποτελείται από τέσσερα τουλάχιστον υποαντικείμενα, φαίνεται στην Εικόνα (3.6).



Εικόνα (3.6). Η γενική οργάνωση ενός τοπικού αντικείμενου για κατανεμημένα κοινόχρηστα αντικείμενα στο Globe.

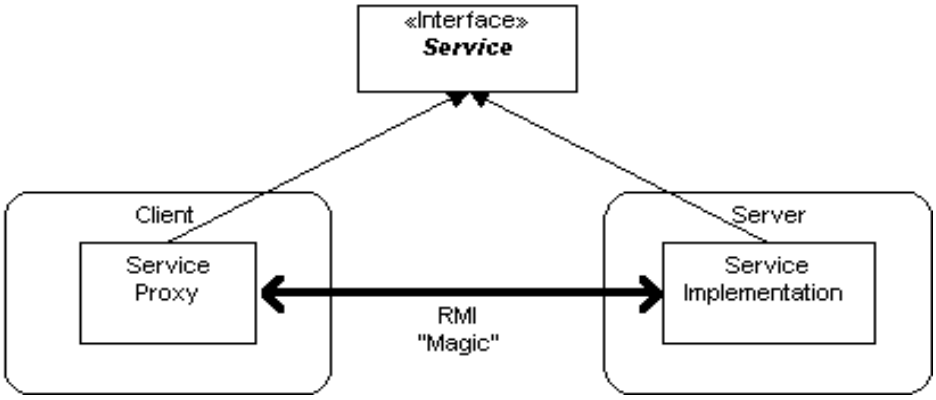
Το **υποαντικείμενο σημασιολογίας** (semantics subobject) υλοποιεί τη λειτουργικότητα που παρέχεται από ένα κατανεμημένο κοινόχρηστο αντικείμενο. Για παράδειγμα, το Globe έχει χρησιμοποιηθεί για τη δημιουργία κατανεμημένων τοποθεσιών Ιστού, στις οποίες μια συλλογή από λογικά συσχετισμένες ιστοσελίδες, εικονίδια, εικόνες, κ.ο.κ. ομαδοποιούνται σε ένα μοναδικό έγγραφο. Ένα έγγραφο τέτοιου είδους, που ονομάζεται **GlobeDoc** (van Steen κ.α., 1999b), υλοποιείται τοπικά μέσω ενός υποαντικειμένου σημασιολογίας. Κάθε αρχείο που χρησιμοποιείτε ως μέρος ενός εγγράφου Ιστού επιστρέφεται ως στοιχείο ενός αντικείμενου GlobeDoc. Τα στοιχεία μπορούν να προστίθενται και να διαγράφονται μέσω μιας διασύνδεσης έγγραφου. Αυτή η διασύνδεση παρέχει επίσης μια μέθοδο η οποία επιστρέφει μια λίστα όλων των στοιχείων. Τα έγγραφα Ιστού πρέπει να είναι οργανωμένα με τη μορφή εγγράφων με ρίζα. Το στοιχείο ρίζας, το οποίο αντιστοιχεί στο τυπικό αρχείο index.htm πολλών εφαρμογών Ιστού, μπορεί να οριστεί και να αναφερθεί μέσω ξεχωριστών μεθόδων. Στην πράξη, αυτές οι μέθοδοι συνήθως είναι κατάλληλες διασυνδέσεις.

Κάθε στοιχείο αναπαριστάτε απλώς ως ένας πινάκας από byte. Η διασύνδεση περιεχομένου παρέχει μεθόδους για τη λήψη του τρέχοντος περιεχομένου ενός στοιχείου, και την αντικατάσταση αυτού του περιεχομένου μέσω ενός τέτοιου πινάκα. Η διασύνδεση ιδιοτήτων παρέχει μεθόδους για τη συσχέτιση των στοιχείων με μεταδεδομένα. Τα μεταδεδομένα ενός στοιχείου έχουν τη μορφή ζευγών (γνώρισμα, τιμή). Για παράδειγμα, κάθε στοιχείο σε ένα αντικείμενο GlobeDoc διαθέτει ένα σχετικό τύπου MIME.

ΚΕΦΑΛΑΙΟ 4: RMI ΚΑΙ CORBA

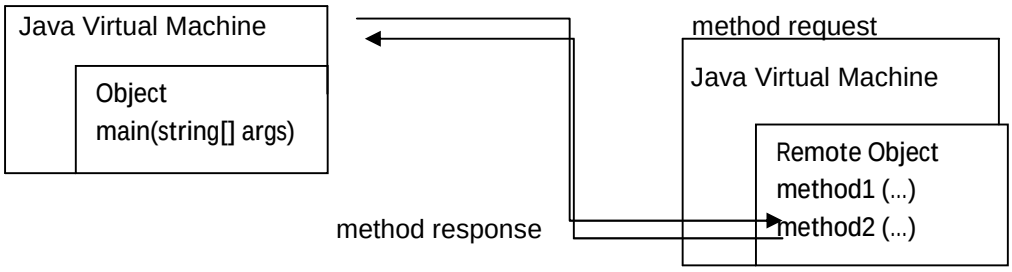
4.1 Remote Method Invocation (RMI)

Αυτό είναι ένα αντικειμενοστραφές μοντέλο όπου Java αντικείμενα μπορούν και επικοινωνούν μεταξύ τους σε ένα κατανεμημένο περιβάλλον με ετερογενή μηχανήματα. Έχει αναπτυχθεί από τη Sun και είναι μέρος της Java από την έκδοση 1.1 του Java Development Kit. Η RMI αποτελεί ένα μηχανισμό που επιτρέπει τις κλήσεις μεταξύ αντικειμένων σε διαφορετικά εικονικά μηχανήματα Java. Από τη στιγμή που αποκτήθηκε μια αναφορά σε ένα απομακρυσμένο αντικείμενο τότε αυτό μπορεί να χρησιμοποιηθεί σαν να ήταν τοπικό αντικείμενο. Η RMI εκτελεί όλες τις απαραίτητες λειτουργίες που αφορούν το απομακρυσμένο αντικείμενο με διαφανή τρόπο, ώστε να μην γίνονται αντιληπτές από τον προγραμματιστή κάνοντας έτσι την ανάπτυξη κατανεμημένων εφαρμογών ιδιαίτερα εύκολη υπόθεση Εικόνα (4.1).



Εικόνα(4.1) Τυπικό παράδειγμα Java RMI

Η Remote Method Invocation (RMI) είναι ένα βασικό API της Java (και βιβλιοθήκη κλάσεων) το οποίο αποτελεί μια εναλλακτική προσέγγιση για την ανάπτυξη κατανεμημένων εφαρμογών. Στην ουσία πρόκειται για μια τεχνολογία η οποία επιτρέπει στην Java Virtual Machine ενός υπολογιστή να καλεί μεθόδους αντικειμένων οι οποίες και εκτελούνται σε JVM άλλων απομακρυσμένων υπολογιστών Εικόνα (4.2).



Εικόνα (4.2) Remote Method Invocation

Οι επιστρεφόμενες τιμές και οι παράμετροι μπορούν να μεταφερθούν και προς τις δύο κατευθύνσεις, κάνοντας την παραπάνω επικοινωνία πλήρως αμφίδρομή. Έτσι τόσο ο client, όσο και ο server μπορούν να εκτελέσουν απομακρυσμένες μεθόδους. Ακόμα πιο γενικά, δίνει την δυνατότητα σε κομμάτια του ίδιου προγράμματος να εκτελούνται στην τοπική JVM, ενώ τα υπόλοιπα τμήματα του να βρίσκονται σε απομακρυσμένους hosts. Ένα σημείο που γίνεται αμέσως αντιληπτό είναι ότι η επικοινωνία αυτή είναι εφικτή μόνο όταν τα αντικείμενα είναι υλοποιημένα σε Java και επομένως καθιστά την χρήση αυτής της γλώσσας απαραίτητη για την κατασκευή της κατανεμημένης εφαρμογής με αυτή την τεχνολογία.

4.1.1 Στόχοι του RMI

Οι προδιαγραφές του RMI θέτουν σαν στόχους του πακέτου RMI τα παρακάτω:

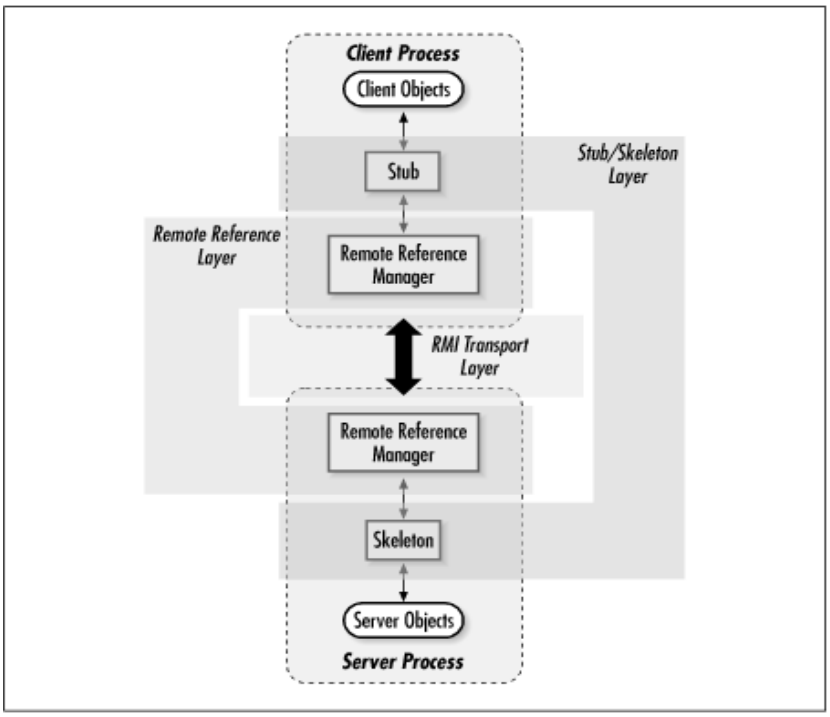
- υποστήριξη απομακρυσμένων κλήσεων σε αντικείμενα τοποθετημένα σε διαφορετικές JVM
- υποστήριξη κλήσεων επιστροφής (callbacks) από server σε client
- ενοποίηση του μοντέλου κατανεμημένων αντικειμένων με το μοντέλο αντικειμένων της Java, με τρόπο φυσικό, διατηρώντας τις ήδη υπάρχουσες έννοιες της Java.
- οι διαφορές μεταξύ των δύο παραπάνω μοντέλων δεν πρέπει να είναι εμφανείς
- η ανάπτυξη αξιόπιστων κατανεμημένων εφαρμογών οφείλει να είναι όσο απλή γίνεται
- διατήρηση της ασφάλειας που παρέχεται από το περιβάλλον του χρόνου εκτέλεσης της Java (Java run-time environment).

Επιπλέον το σύστημα του RMI επιδιώκει την ευελιξία και την επεκτασιμότητα:

- πολλαπλοί μηχανισμοί απομακρυσμένων κλήσεων όπως unicast και multicast
- υποστήριξη διαφόρων τρόπων μεταφοράς
- κατανεμημένο μάζεμα απορριμμάτων (garbage collection)

4.1.2 Αρχιτεκτονική RMI

Στα συστήματα RMI υπάρχουν τρία, πλήρως ανεξάρτητα, στρώματα: το στρώμα στελέχους/σκελετού (stub/skeleton layer), το στρώμα απομακρυσμένης διασύνδεσης (remote interface layer) και το στρώμα μεταφοράς (transport layer), όπως φαίνονται στο Εικόνα (4.3). Η ανεξαρτησία των στρωμάτων τους δίνει την δυνατότητα να αντικατασταθούν από εναλλακτικές υλοποιήσεις, χωρίς αυτό να δημιουργήσει τυχόν προβλήματα συμβατότητας στο σύστημα RMI.

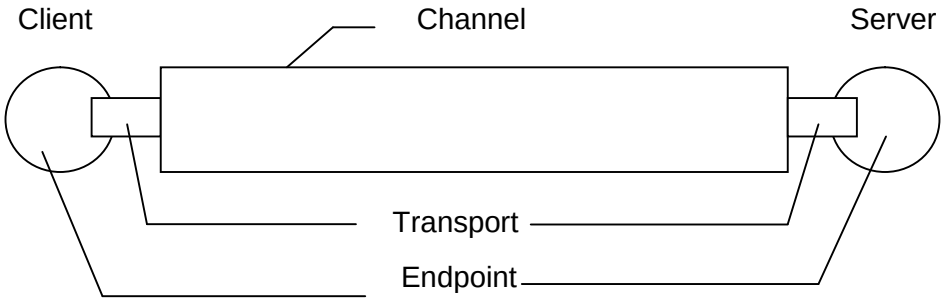


Εικόνα(4.3) Αρχιτεκτονική RMI

Stub/Skeleton Layer. Αυτό το στρώμα αποτελεί τη σύνδεση μεταξύ της εφαρμογής και του υπόλοιπου RMI συστήματος. Τα στελέχη και οι σκελετοί είναι κλάσεις πληρεξούσιοι (proxy classes), οι οποίες δημιουργούνται κατά την ανάπτυξη της εφαρμογής όταν μεταγλωττίσουμε την διασύνδεση του αντικειμένου και την υλοποίηση του εξυπηρετητή με τον rmic, τον μεταγλωττιστή του RMI. Το στέλεχος (stub) αποτελεί το τμήμα του client και είναι υπεύθυνο για πολλές εργασίες όπως την εκκίνηση των απομακρυσμένων κλήσεων, την συγκέντρωση των παραμέτρων που θα σταλούν, την ενημέρωση του επόμενου στρώματος για την πραγματοποίηση της κλήσης, απόδοση των τιμών (ή εξαιρέσεων) και ενημέρωση του remote reference στρώματος για το τέλος της κλήσης. Από την άλλη, ο σκελετός είναι το τμήμα του server και είναι υπεύθυνο για την απόδοση των εισερχομένων παραμέτρων από τον client, την κλήση του επιθυμητού αντικειμένου και συγκέντρωση και επιστροφή των τιμών (ή εξαιρέσεων) πίσω στον client.

Remote Reference Layer. Το στρώμα απομακρυσμένης αναφοράς είναι το στρώμα που μεσολαβεί μεταξύ των στελεχών/σκελετών και του στρώματος μεταφοράς. Κύρια του ευθύνη είναι υποστήριξη πολλαπλών απομακρυσμένων αναφορών ή πρωτόκολλα κλήσεων, ανεξάρτητα από τα στελέχη και τους σκελετούς. Τα πρωτόκολλα αυτά μπορούν να υποστηρίζουν επικοινωνία από σημείο σε σημείο (point-to-point) ή και αναφορά σε αντικείμενα αντίγραφα. Το σύστημα του RMI εξασφαλίζει ότι η κλήση σε απομακρυσμένο αντικείμενο το οποίο έχει πολλαπλά αντίγραφα θα προωθηθεί και στα υπόλοιπα αντίγραφά του.

Transport Layer. Το χαμηλότερο από τα τρία στρώματα, το οποίο αναλαμβάνει τη μεταφορά των δημιουργημένων ροών δεδομένων στην σωστή διεύθυνση. Το στρώμα αυτό καθορίζει και χειρίζεται τις συνδέσεις με απομακρυσμένες διευθύνσεις, περιμένει για εισερχόμενες κλήσεις, κρατάει ένα πίνακα με τα διαθέσιμα απομακρυσμένα αντικείμενα, εντοπίζει την τοποθεσία του καλούντα για λογαριασμό του καλούμενου και παραδίδει την σύνδεση στον καλούντα. Το στρώμα μεταφοράς αποτελείται από τέσσερα τμήματα: τα αντικείμενα, τον χώρο μεταξύ τοπικών και απομακρυσμένων διευθύνσεων, την φυσική υποδοχή και το πρωτόκολλο μεταφοράς Εικόνα (4.4).



Εικόνα (4.4) Transport Layer

Τα RMI συστήματα χρησιμοποιούν γα τη μεταφορά ένα πρωτόκολλο βασισμένο στο TCP, αλλά εφόσον είναι δυνατή η πολλαπλή μεταφορά ανά διεύθυνση, είναι εφικτή η χρήση ενός πρωτοκόλλου βασισμένου στο UDP.

4.1.3 Λειτουργία του RMI

Όταν ένας client καλέσει κάποιο απομακρυσμένο αντικείμενο που βρίσκεται σε ένα server τότε αναλαμβάνουν δράση τα τρία στρώματα της αρχιτεκτονικής του RMI.

Τα συστήματα που χρησιμοποιούν την RMI για την επικοινωνία τους αποτελούνται συνήθως από το κομμάτι του server, το οποίο παρέχει την υπηρεσία RMI και το κομμάτι του client, το οποίο γίνεται χρήστης αυτής της υπηρεσίας.

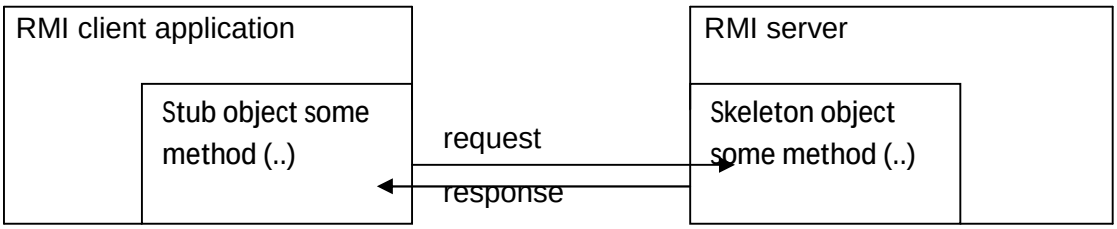
Οι RMI servers πρέπει να καταχωρηθούν σε μια υπηρεσία αναζήτησης (lookup service), ώστε να μπορούν οι clients να τους εντοπίζουν. Συγκεκριμένα στην Java υπάρχει μια τέτοια υπηρεσία, η rmiregistry, η οποία εκτελείται ως ξεχωριστή διαδικασία και επιτρέπει στις εφαρμογές να καταχωρούν RMI υπηρεσίες ή να αποκτούν αναφορά σε μια δηλωμένη υπηρεσία. Από τη στιγμή που κάποιος server καταχωρηθεί εκεί, περιμένει πλέον για εισερχόμενες αιτήσεις.

Οι RMI clients στέλνουν μηνύματα RMI για να καλέσουν κάποια απομακρυσμένη μέθοδο. Πριν όμως συμβεί κάτι τέτοιο, πρέπει πρώτα ο client να αποκτήσει μια αναφορά στο απομακρυσμένο αντικείμενο. Αυτό μπορεί να γίνει με την υπηρεσία αναζήτησης που προσφέρει το μητρώο του RMI. Η εφαρμογή του πελάτη ζητάει το όνομα μιας συγκεκριμένης υπηρεσίας και δέχεται ένα URL για τον απομακρυσμένο πόρο. Οι αναφορές σε απομακρυσμένα αντικείμενα συνήθως αναπαριστώνται από την RMI ως εξής:

rmi://hostname:port/servicename

όπου hostname μπορεί να είναι IP διεύθυνση ή όνομα server, port η τοποθεσία της υπηρεσίας στον συγκεκριμένο υπολογιστή και servicename η περιγραφή της υπηρεσίας.

Αφού αποκτήσει την αναφορά, ο client μπορεί πλέον να αλληλεπιδράσει με την απομακρυσμένη υπηρεσία. Οι λεπτομέρειες που αφορούν το δίκτυο, όσον αφορά τις αιτήσεις, είναι τελείως διαφανείς για τον προγραμματιστή, ο οποίος έχει την εντύπωση ότι δουλεύει με τοπικά αντικείμενα. Αυτό είναι ένα από τα σημαντικά πλεονεκτήματα που προσφέρουν τα στελέχη και οι σκελετοί, τα οποία δρουν ως πληρεξούσιοι των αντικειμένων που αλληλεπιδρούν. Το στέλεχος υλοποιεί κάποια συγκεκριμένη διασύνδεση RMI, την οποία μπορεί να χρησιμοποιεί η εφαρμογή του client. Το στέλεχος δεν πραγματοποιεί τις απαραίτητες εργασίες το ίδιο, μόνο περνάει τις αιτήσεις στην επιθυμητή υπηρεσία και περιμένει για απάντηση την οποία επιστρέφει στην καλούσα μέθοδο. Στην μεριά του server, ο σκελετός είναι υπεύθυνος για την απάντηση στις εισερχόμενες αιτήσεις και τη μεταφορά τους στην υπηρεσία RMI. Όπως και το στέλεχος, έτσι και ο σκελετός δεν προσφέρει καμία υλοποίηση για τις παρεχόμενες υπηρεσίες. Μετά από τη δημιουργία των διασυνδέσεων, ο προγραμματιστής οφείλει να τις υλοποιήσει. Αυτή η υλοποίηση θα κληθεί από τον σκελετό, ο οποίος θα ενεργοποιήσει τις απαραίτητες μεθόδους και θα περάσει τα αποτελέσματα πίσω στο στέλεχος και κατά συνέπεια στον client. Η όλη διαδικασία φαίνεται στην Εικόνα (4.5).



Εικόνα (4.5) Αίτηση-Απάντηση κλήσεων αντικειμένων

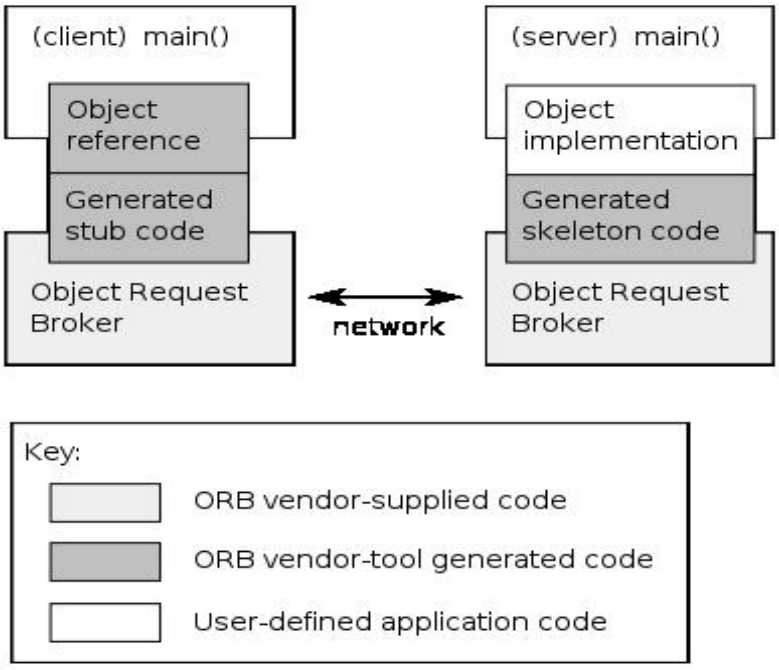
Καταλήγοντας βλέπουμε ότι η RMI είναι μια τεχνολογία κατανεμημένων συστημάτων, η οποία επιτρέπει την απομακρυσμένη κλήση αντικειμένων σε απομακρυσμένες Java Virtual Machines. Αυτός είναι συχνά ο απλούστερος τρόπος επικοινωνίας μεταξύ δύο εφαρμογών, ακόμα και από την απευθείας επικοινωνία μέσω υποδοχών (sockets). Η αλληλεπίδραση με τα αντικείμενα γίνεται με μεθόδους ορισμένες σε μια διασύνδεση RMI. Μόλις μια αναφορά σε ένα απομακρυσμένο αντικείμενο γίνει γνωστή, αυτό μπορεί να χρησιμοποιηθεί σαν τοπικό αντικείμενο, γεγονός που απλοποιεί σημαντικά την δημιουργία εφαρμογών δικτύου. Τα παραπάνω έχουν βέβαια αναγκαία προϋπόθεση την χρήση της JAVA ως κοινή γλώσσα προγραμματισμού, ένας περιορισμός που δεν τίθεται από την αρχιτεκτονική CORBA, την κύρια εναλλακτική λύση για την RMI και με την οποία θα ασχοληθούμε παρακάτω.

4.2 CORBA (Common Object Request Broker Architecture)

Στη μελέτη των κατανεμημένων συστημάτων με βάση αντικείμενα εξετάζουμε το πρότυπο **Κοινής Αρχιτεκτονικής Διαχείρισης Αιτήσεων για τη Διάθεση Αντικειμένων** (Common Object Request Broker Architecture), το οποίο αναφέρεται απλώς ως **CORBA**. Όπως υποδηλώνει το όνομά του, το CORBA δεν αποτελεί τόσο ένα κατανεμημένο σύστημα όσο τις προδιαγραφές ενός τέτοιου συστήματος. Αυτές οι προδιαγραφές έχουν καταρτιστεί από την **Ομάδα Διαχείρισης Αντικειμένων** (Object Management Group, **OMG**) μια μη κερδοσκοπική οργάνωση με περισσότερα από 800 μέλη τα οποία προέρχονται κυρίως από το χώρο της βιομηχανίας.

Όσον αφορά το CORBA, σημαντικό στόχο της OMG αποτέλεσε ο ορισμός ενός κατανεμημένου συστήματος το οποίο θα μπορούσε να υπερπηδήσει πολλά από τα προβλήματα διαλειτουργικότητας κατά το συνδυασμό δικτυακών εφαρμογών. Κάθε υλοποίηση του CORBA έχει τις δικές της επεκτάσεις, αφού κάθε κατασκευαστής αισθάνεται πάντοτε ότι υπάρχει κάτι που δεν μπορεί να παραλειφθεί αλλά δεν περιέχεται στις προδιαγραφές. Το CORBA αποτελεί ένα ακόμα καλό παράδειγμα του γεγονότος ότι η δημιουργία ενός απλού κατανεμημένου συστήματος μπορεί να αποτελέσει μια εξαιρετικά δύσκολη προσπάθεια.

Το πρότυπο αυτό αναπτύσσεται από τον οργανισμό OMG και αν και εξελίσσεται ακόμη είναι το πιο ολοκληρωμένο από πλευράς αρχιτεκτονικής. Η CORBA δεν είναι μια υλοποίηση ενός προτύπου αλλά το ίδιο το πρότυπο. Ο OMG αναπτύσσει μόνον το πρότυπο και οι διάφορες υλοποιήσεις του προέρχονται από άλλους, συνήθως ανεξάρτητους φορείς. Κύριο στοιχείο αυτού του προτύπου αποτελεί ο ORB (Object Request Broker). Ο OMG εντάσσει την CORBA σε ένα ευρύτερο μοντέλο που είναι το OMA. Με βάση το OMA κάθε κομμάτι λογισμικού είναι ένα αντικείμενο. Το αντικείμενο ως οντότητα επικοινωνεί με το ORB και μέσω αυτού μπορεί να παρέχει ή να ζητά υπηρεσίες. Το πρότυπο αυτό αναφέρεται και ως CORBA 2.x για να ξεχωρίζει από το CORBA 3.x που είναι η επέκτασή του υποστηρίζοντας συνιστώσες. Στην Εικόνα (4.6) φαίνεται ο ρόλος του ORB.



Εικόνα (4.6) Η αρχιτεκτονική CORBA 2.x

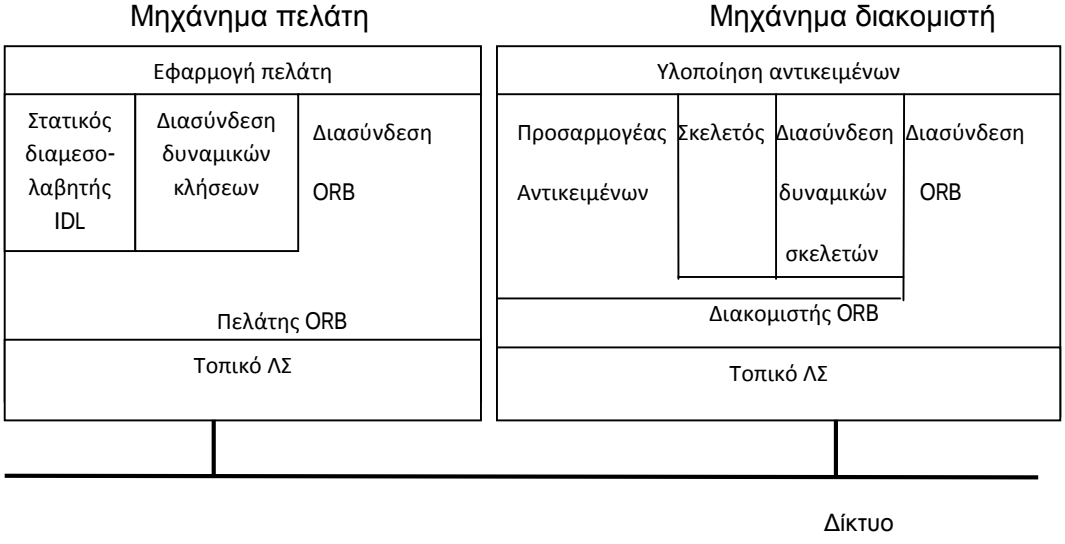
4.2.1 Μοντέλο αντικειμένων

Το CORBA χρησιμοποιεί το μοντέλο απομακρυσμένων αντικειμένων. Στο συγκεκριμένο μοντέλο, η υλοποίηση ενός αντικειμένου βρίσκεται στο χώρο διευθύνσεων ενός διακομιστή. Αντικείμενα και υπηρεσίες καθορίζονται στη **Γλώσσα Ορισμού Διασυνδέσεων** (Interface Definition Language, **IDL**)

του CORBA. Η IDL του CORBA είναι παρόμοια με άλλες γλώσσες ορισμού διασυνδέσεων, με την έννοια ότι παρέχει ένα ακριβές συντακτικό για την έκφραση των μεθόδων και των παραμέτρων τους. Η σημασιολογία της IDL του CORBA δεν είναι δυνατό να περιγραφεί. Μια διασύνδεση αποτελεί μια συλλογή μεθόδων, και τα αντικείμενα καθορίζουν ποιες διασυνδέσεις υλοποιούν.

Οι προδιαγραφές διασυνδέσεων μπορούν να δοθούν μόνο μέσω της IDL. Όπως είδαμε νωρίτερα, σε συστήματα όπως το DCOM οι διασυνδέσεις καθορίζονται σε ένα χαμηλότερο επίπεδο με τη μορφή πινάκων. Αυτές οι αποκαλούμενες **δυαδικές διασυνδέσεις** είναι από τη φύση τους ανεξάρτητες από οποιαδήποτε γλώσσα προγραμματισμού. Στο CORBA όμως, είναι απαραίτητο να παρασχεθούν ακριβείς κανόνες όσον αφορά την αντιστοίχιση των προδιαγραφών IDL σε υπάρχουσες γλώσσες προγραμματισμού. Προς το παρόν τέτοιοι κανόνες έχουν δοθεί για μια σειρά γλωσσών, μεταξύ των οποίων οι C, C++, Java και COBOL.

Με δεδομένο ότι το CORBA είναι οργανωμένο ως ένα σύνολο πελατών και διακομιστών αντικειμένων στην Εικόνα (4.7) παρουσιάζεται η γενική οργάνωση ενός συστήματος CORBA. Πίσω από οποιαδήποτε διεργασία στο CORBA, είτε είναι πελάτη είτε διακομιστή, βρίσκεται ο Διαχειριστής Αιτήσεων για τη Διάθεση Αντικειμένων (ORB). Ο ORB μπορεί να θεωρηθεί ως το σύστημα χρόνου εκτέλεσης που είναι υπεύθυνο για το χειρισμό της βασικής επικοινωνίας μεταξύ ενός πελάτη και ενός αντικειμένου. Αυτή η βασική επικοινωνία πρέπει να διασφαλίσει ότι μια κλήση θα σταλεί στο διακομιστή του αντικειμένου, και ότι η απάντηση θα μεταβιβαστεί πίσω στον πελάτη.



Εικόνα (4.7) Η γενική οργάνωση ενός συστήματος CORBA.

Από την άποψη μιας διεργασίας, ο ίδιος ο ORB παρέχει ελάχιστες μόνο υπηρεσίες. Μία από αυτές είναι ο χειρισμός των αναφορών των αντικειμένων. Τέτοιες αναφορές εξαρτώνται γενικά από ένα συγκεκριμένο ORB. Για τον λόγο αυτόν, ένας ORB παρέχει λειτουργίες για την παράταξη (marshal) και την από-παράταξη (un-marshal) αναφορών αντικειμένων, έτσι ώστε να είναι δυνατή η ανταλλαγή τους μεταξύ διεργασιών, καθώς επίσης και λειτουργίες για τη σύγκριση αναφορών.

Άλλες λειτουργίες που παρέχονται από έναν ORB αφορούν το αρχικό εντοπισμό των υπηρεσιών που είναι διαθέσιμες σε μια διεργασία. Γενικά, ο ORB παρέχει τα μέσα για την απόκτηση μιας αρχικής αναφοράς σε ένα αντικείμενο υλοποιώντας μια συγκεκριμένη υπηρεσία CORBA. Για παράδειγμα, προκειμένου να μπορέσει μια διεργασία να χρησιμοποιήσει μια υπηρεσία ονομασίας, πρέπει να γνωρίζει τον τρόπο αναφοράς σε αυτή την υπηρεσία. Αυτά τα ζητήματα προετοιμασίας έχουν εφαρμογή και σε άλλες υπηρεσίες.

Εκτός από τη διασύνδεση του ORB, οι πελάτες και οι διακομιστές δεν μπορούν να δουν σχεδόν τίποτε άλλο από αυτόν. Αντίθετα, γενικά βλέπουν μόνο στελέχη για το χειρισμό κλήσεων μεθόδων σε συγκεκριμένα αντικείμενα. Μια εφαρμογή πελάτη συνήθως διαθέτει ένα διαμεσολαβητή, ο οποίος υλοποιεί την ίδια διασύνδεση με κάθε αντικείμενο που χρησιμοποιεί. Η διασύνδεση μεταξύ ενός διαμεσολαβητή και του ORB δε χρειάζεται να είναι τυποποιημένη. Επειδή το CORBA προϋποθέτει ότι όλες οι διασυνδέσεις δίνονται σε IDL, οι υλοποιήσεις του παρέχουν στους υπεύθυνους ανάπτυξης εφαρμογών ένα μεταγλωττιστή IDL, ο οποίος δημιουργεί τον απαραίτητο κώδικα για το χειρισμό της επικοινωνίας ανάμεσα στον πελάτη και το διακομιστή ORB.

4.3 Επισκόπηση του μοντέλου συνιστωσών CORBA

Για να υπερκεράσει τους περιορισμούς του CORBA μοντέλου αντικειμένου που θα δούμε παρακάτω ο οργανισμός OMG υιοθέτησε το CORBA μοντέλο συνιστωσών (CORBA component model CCM) έτσι ώστε να επεκτείνει το CORBA μοντέλο αντικειμένων.

Το CCM συμπεριλήφθηκε στο πρότυπο CORBA 3.0 και διατέθηκε από τον OMG το 2001. Το CCM επεκτείνει το CORBA μοντέλο αντικειμένων με το να ορίζει δυνατότητες και υπηρεσίες που επιτρέπουν στους προγραμματιστές να υλοποιούν, διαχειρίζονται, παραμετροποιούν και εξαπλώνουν συνιστώσες όπου περιλαμβάνουν τις συχνά χρησιμοποιούμενες CORBA υπηρεσίες. Τέτοιες υπηρεσίες είναι οι υπηρεσίες δοσοληψιών, ασφάλειας, μόνιμης κατάστασης και ειδοποίηση συμβάντων. Επιπροσθέτως το πρότυπο CCM επιτρέπει σε μεγαλύτερο βαθμό την επαναχρησιμοποίηση του λογισμικού για εξυπηρετητές και προσφέρει μεγαλύτερη ευελιξία για δυναμική παραμετροποίηση CORBA εφαρμογών. Με την αυξανόμενη αποδοχή της CORBA σε ένα ευρύ τομέα εφαρμογών το CCM κατέχει υψηλή θέση στην κατάταξη για χρήση κρίσιμων εφαρμογών πελάτη εξυπηρετητή. Σήμερα το μοντέλο αυτό χρησιμοποιείται σε συστήματα που απαιτούν υψηλή σταθερότητα και εύκολη επεκτασιμότητα. Ιδιαίτερο ρόλο παίζει στο μοντέλο αυτό ότι μπορούν να δημιουργηθούν συνιστώσες βασισμένες σε διαφορετικές γλώσσες, που σημαίνει επαναχρησιμοποίηση κώδικα που έχει γραφτεί σε πιο παλιού τύπου γλώσσες (COBOL, FORTRAN) και επέκτασή του. Τέτοια συστήματα είναι τα τραπεζικά δίκτυα, οι βιομηχανικές εφαρμογές και τα αμυντικά συστήματα.[8]

Το μοντέλο CCM ορίζει δύο ειδών συνιστώσες. Τις βασικές και τις επεκταμένες. Οι βασικές συνιστώσες είναι μια ειδική κατηγορία όπου υλοποιούν ένα μικρό μέρος του προτύπου. Συγκεκριμένα δεν μπορούν να κληρονομούν άλλες συνιστώσες και δεν μπορούν να προσφέρουν ή να χρησιμοποιούν διεπαφές. Επίσης δεν παρέχεται κάποιος μηχανισμός υποστήριξης γεγονότων. Αυτά κυρίως χρησιμοποιούνται για αντιστοίχηση με τα Java Beans.

4.4 Υπάρχουσες υλοποιήσεις του CCM

Σήμερα υπάρχουν και άλλες υλοποιήσεις του προτύπου συνιστωσών CORBA. Οι περισσότερες υλοποιήσεις που υπάρχουν σήμερα είναι ανοιχτού κώδικα. Σε όλες τις περιπτώσεις για τη λειτουργία μιας από τις παρακάτω πλατφόρμες απαιτείται ξεχωριστός ORB και δεν περιέχεται σε αυτές. Δηλαδή όλες οι υλοποιήσεις δεν προσφέρουν πλήρες περιβάλλον CORBA αλλά μόνο την διερμήνευση από CORBA σε CORBA 2x. Ακόμα καμία δεν βρίσκεται σε θέση να υποστηρίξει ολόκληρο το πρότυπο. συνήθως ικανοποιούν μόνο ένα μικρό μέρος του και μόνο δυο συγκεκριμένες υπηρεσίες, την ονομασία (naming) και την ανταλλαγή (trading). Από τις υλοποιήσεις που υπάρχουν σήμερα η μοναδική που είναι εμπορική είναι η K2 Container. Όσον αφορά τις γλώσσες πάνω στις οποίες γίνεται η χαρτογράφηση (mapping) όλα εκτός του OpenCCM χρησιμοποιούν την C++. Το OpenCCM είναι το μόνο που χρησιμοποιεί την Java.

- **K2-CCM Container**. Αυτή είναι μια εμπορική υλοποίηση του προτύπου από την εταιρία iCMG. Καλύπτει μεγάλο μέρος του προτύπου CCM και είναι το μόνο που για την επικοινωνία χρησιμοποιεί αποκλειστικά διαδικτυακές υπηρεσίες (web services). Επίσης έχει υλοποιήσει επικοινωνία μεταξύ CORBA components και συνιστωσών Enterprise Java Beans. Από πλευράς υπηρεσιών έχει υλοποιήσει τα transaction (συναλλαγή), trader (ανταλλαγή), persistent (επιμονή) και security (ασφάλεια). Σήμερα χρησιμοποιείται αρκετά από εταιρίες οι οποίες θέλουν να συνεχίσουν να χρησιμοποιούν κώδικα από FORTRAN και COBOL.

- **OpenCCM**. Αυτό είναι μια υλοποίηση ανοιχτού λογισμικού από τον οργανισμό ObjectWeb. Είναι το μόνο που χρησιμοποιεί την Java σαν γλώσσα για την δημιουργία του τελικού κώδικα. Ακόμα έχει αρκετά βοηθητικά γραφικά εργαλεία. Αν και αναφέρει ότι θα υποστηρίξει στο μέλλον χαρακτηριστικά πραγματικού χρόνου, προς το παρόν δεν έχει. Το OpenCCM συνεργάζεται με το JacOrb, το Orbacus και το OpenOrb.

- **CIAO**. Η υλοποίηση αυτή είναι για C++. Είναι ανοιχτού λογισμικού υλοποίηση, και αναπτύσσεται από το DOC Group που είναι μια κοινή προσπάθεια από πανεπιστήμια. Αυτή η υλοποίηση σαν προτέρημά της έχει ότι ενσωματώνει δυνατότητες για υποστήριξη χαρακτηριστικών πραγματικού χρόνου. Το CIAO χρησιμοποιεί σαν ORB κυρίως το TAO (The Ace Orb).

- **MICO CCM**. Αυτή η υλοποίηση είναι ανοιχτού λογισμικού και υποστηρίζεται από την τηλεπικοινωνιακή εταιρία Alcatel. Συνεργάζεται με το MICO ORB. Εκτός από το session container που υλοποιούν τα άλλα το MICO CCM υλοποιεί και το service container. Διαθέτει πολύ καλό γραφικό περιβάλλον για τη δημιουργία των περιγραφέων.

- **EJCCM**. Η υλοποίηση αυτή είναι ανοιχτού λογισμικού σε Java, αλλά έχει εγκαταλειφθεί. Ικανοποίει πολύ μικρό μέρος του προτύπου και έχει πολλά χρόνια να ανανεωθεί.

- **StarCCM**. Η υλοποίηση αυτή είναι ανοιχτού κώδικα. Σαν γλώσσα χρησιμοποιεί την C++. Οι υπηρεσίες που υλοποιεί είναι το transaction (συναλλαγή) και το naming (ονομασία) και ακόμα υλοποιεί το Persistent State Service.

4.5 Περιορισμοί του CORBA

Το CORBA σήμερα είναι μια τεχνολογία που προσφέρει μηχανισμούς για την δημιουργία λογισμικού που αποκρύπτει την ετερογένεια των συστημάτων. Όμως αυτή η απόκρυψη αφορά μόνο χαμηλού επιπέδου ετερογένεια. Σήμερα υπάρχει ανάγκη για λογισμικό που είναι πλήρως μεταφέρσιμο και οι λειτουργίες του είναι πλήρως ενθυλακωμένες. Αυτό έφερε την ανάγκη για προτυποποίηση και των εξωτερικών μηχανισμών που προσφέρεται σε ένα κομμάτι λογισμικού. Συγκεκριμένα πρέπει να προσπεραστούν οι παρακάτω περιορισμοί [9][10][11][12][13]:

1. Έλλειψη τυποποίησης για γενικού τύπου εξυπηρετητές: Το πρότυπο CORBA αντικειμένων προσφέρει ένα δυνατό σετ μηχανισμών για τη δημιουργία εξυπηρετητών. Παρόλα αυτά από τους προγραμματιστές μόνο ένα μικρό υποσύνολο τελικά χρησιμοποιείται. Ακόμα υπάρχει ανάγκη για υποστήριξη μηχανισμών όπου ο προγραμματιστής θα επικεντρώνεται μόνο στο λειτουργικό κώδικα που θέλει να κατασκευάσει και όχι στο να παραμετροποιήσει τους μηχανισμούς της CORBA. Για παράδειγμα στο μοντέλο CORBA 2.x ορίζεται το POA (Portable Object Adapter). Αυτός είναι ο μηχανισμός που αναλαμβάνει την προώθηση των αιτημάτων από ένα αντικείμενο σε ένα άλλο. Διαθέτει πλήθος μηχανισμών για την σύνδεση ενός αντικειμένου με το ORB, και για την πολιτική διαχείρισής του. Όμως κάθε προγραμματιστής χρησιμοποιεί μόνο ένα υποσύνολο αυτών των μηχανισμών αν και χρειάζεται να τους γνωρίζει όλους αν θέλει να έχει τα επιθυμητά αποτελέσματα.

2. Έλλειψη τυποποίησης διασύνδεσης και εγκατάστασης: Το πρότυπο CORBA 2.x δεν διαθέτει μηχανισμό για την διασύνδεση των αντικειμένων μεταξύ τους. Ακόμα δεν διαθέτει μηχανισμό για την αυτοματοποιημένη μεταφορά του από ένα σημείο εγκατάστασης σε ένα σύστημα στόχο με σκοπό την αυτόματη εγκατάστασή του. Ο προγραμματιστής πρέπει με δικό του τρόπο να δημιουργήσει τα στιγμιότυπα, να τα τοποθετήσει το σύστημα που επιθυμεί και να τα εκκινήσει. Περισσότερο αυτό δημιουργεί προβλήματα όταν τα αντικείμενα μεταξύ τους έχουν εξαρτήσεις, όπου αυξάνεται η πολυπλοκότητα.

3. Έλλειψη μηχανισμού επέκτασης των δυνατοτήτων ενός αντικειμένου: Ο μηχανισμός επέκτασης της λειτουργικότητας ενός αντικειμένου γίνεται μόνο μέσω της κληρονομικότητας. Αν ο προγραμματιστής θέλει να επεκτείνει ένα αντικείμενο τότε πρέπει:

a. Να γράψει με τη γλώσσα IDL τη νέα διεπαφή και αυτή να κληρονομεί όλες τις άλλες που χρειάζονται.
b. Να υλοποιήσει τον κώδικα για τη νέα διεπαφή.
c. Να εξαπλώσει το νέο αντικείμενο ξανά σε όλους τους εξυπηρετητές.

Παρόλα αυτά όμως η διαδικασία της κληρονομικότητας είναι μια δύσκολη υπόθεση. Στην IDL δεν υποστηρίζεται η υπερφόρτωση μεθόδων. Αυτό γιατί υπάρχουν γλώσσες που δεν την υποστηρίζουν. Όμως οι προγραμματιστές ορισμένες φορές απαιτούν να έχουν την ίδια διεπαφή σε ένα σύστημα διαθέσιμη πολλές φορές. Αυτό γιατί θέλουν μια υπηρεσία να είναι διαθέσιμη ως στιγμιότυπο πολλαπλές φορές από το ίδιο σημείο.

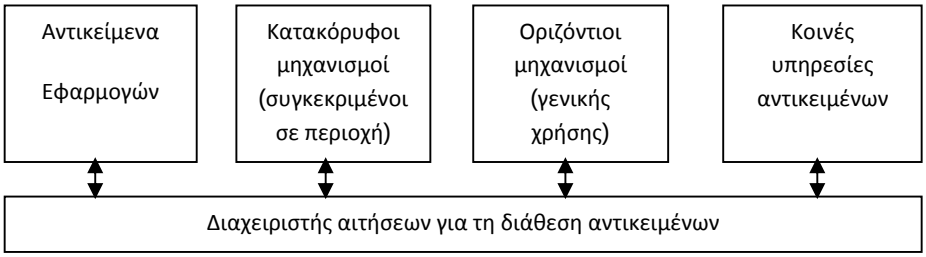
4. Έλλειψη διαθεσιμότητας των υπηρεσιών του προτύπου στο χρόνο εκτέλεσης: Ο προγραμματιστής δεν γνωρίζει ποιες υπηρεσίες είναι διαθέσιμες κατά την εκτέλεση της εφαρμογής του. Έτσι χρειάζεται να αναπτύξει δικές του στρατηγικές για την ενεργοποίησή και παραμετροποίησή τους κατά τη φάση σχεδιασμού της εφαρμογής. Αυτό οδηγεί σε μη μεταφέρσιμες υλοποιήσεις.

5. Έλλειψη προτύπου διαχείρισης κύκλου ζωής: Αν και το πρότυπο CORBA ορίζει την υπηρεσία κύκλου ζωής (Lifecycle) η χρήση της δεν είναι υποχρεωτική. Έτσι πολλές φορές η διαχείριση του κύκλου ζωής των αντικειμένων επιβάλει στρατηγικές τύπου adhoc.

Συνοπτικά η ανεπάρκεια του CORBA προτύπου που περιγράψαμε παραπάνω πριν την έκδοση 2.4 συνήθως περιλάμβανε την επί τούτου υλοποίηση αντικειμένων που είναι δύσκολο να σχεδιαστούν, επαναχρησιμοποιηθούν, συντηρηθούν και επεκταθούν.

4.6 Σύνοψη του CORBA

Η καθολική αρχιτεκτονική του CORBA βασίζεται σε ένα μοντέλο αναφορών της ομάδας OMG. Αυτό το μοντέλο αναφορών φαίνεται στην Εικόνα (4.8), και αποτελείται από τέσσερις ομάδες αρχιτεκτονικών στοιχείων συνδεδεμένων με αυτό που ονομάζεται **Διαχειριστής Αιτήσεων για τη Διάθεση Αντικειμένων** (Object Request Broker, **ORB**). Ο ORB αποτελεί τον πυρήνα οποιουδήποτε κατανεμημένου συστήματος CORBA. Είναι υπεύθυνος για την επίτευξη της επικοινωνίας ανάμεσα στα αντικείμενα και τους πελάτες τους, κρύβοντας παράλληλα ζητήματα που σχετίζονται με την κατανομή και την ανομοιογένεια. Σε πολλά συστήματα, ο ORB υλοποιείται όπως οι βιβλιοθήκες που συνδέονται με εφαρμογές πελάτη ή διακομιστή, και παρέχει βασικές υπηρεσίες επικοινωνίας.



Εικόνα (4.8) Η καθολική αρχιτεκτονική του CORBA

Εκτός από τα αντικείμενα που δημιουργούνται ως μέρος συγκεκριμένων εφαρμογών, το μοντέλο αναφορών διακρίνει επίσης τους **μηχανισμούς CORBA** (CORBA facilities). Οι μηχανισμοί αυτοί κατασκευάζονται ως συνθέσεις υπηρεσιών CORBA και χωρίζονται σε δύο διαφορετικές ομάδες. Οι **οριζόντιοι μηχανισμοί** αποτελούνται από υπηρεσίες υψηλού επιπέδου γενικής χρήσης, οι οποίες είναι ανεξάρτητες από περιοχές εφαρμογών. Στις υπηρεσίες τέτοιου είδους σήμερα περιλαμβάνονται υπηρεσίες διασύνδεσης χρήστη, διαχείρισης πληροφοριών, διαχείριση συστήματος, και διαχείρισης εργασιών (η οποία χρησιμοποιείται για τον ορισμό συστημάτων ροής εργασιών). Οι **κατακόρυφοι μηχανισμοί** αποτελούνται από υπηρεσίες υψηλού επιπέδου οι οποίες στοχεύουν σε μια συγκεκριμένη περιοχή εφαρμογών, όπως το ηλεκτρονικό εμπόριο, οι τραπεζικές συναλλαγές, η βιομηχανία κλπ.

ΚΕΦΑΛΑΙΟ 5: ΣΥΓΚΡΙΣΗ ΤΕΧΝΟΛΟΓΙΩΝ

5.1 Σύγκριση CORBA, DCOM, και GLOBE

Το CORBA, το DCOM και το GLOBE είναι τρία διαφορετικά συστήματα που βασίζονται σε αντικείμενα, το καθένα με τα δικά του πλεονεκτήματα και μειονεκτήματα. Αφού περιγράψαμε λεπτομερώς τις αρχές του κάθε συστήματος, θα προχωρήσουμε τώρα σε μια σύγκριση των τριών συστημάτων ως προς αυτές τις αρχές, ξεκινώντας από τους γενικούς στόχους σχεδιασμού τους. [15]

5.1.1 Φιλοσοφία

Το CORBA, το DCOM και το GLOBE έχουν αναπτυχθεί με διαφορετικούς στόχους σχεδιασμού. Το CORBA αποτελεί κυρίως το αποτέλεσμα μιας προσπάθειας για την παροχή ενός τυπικού περιβάλλοντος ενδιάμεσου λογισμικού, το οποίο θα επιτρέπει σε εφαρμογές πολλών διαφορετικών κατασκευαστών να διαλειτουργούν. Με εκατοντάδες μέλη διάφορων επίτροπων να προσπαθούν να συμφωνήσουν ως προς το πώς θα πρέπει να φαίνονται οι διασυνδέσεις προς τα διάφορα στοιχεία CORBA, το ότι υπάρχει έστω κάποια συμφωνία είναι όντως εκπληκτικό.

Ο σημαντικότερος στόχος σχεδιασμού του DCOM φαίνεται να είναι η βελτίωση της λειτουργικότητας με παράλληλη διατήρηση της συμβατότητας με προηγούμενες εκδόσεις που ήταν ενσωματωμένες ακόμα και στα πρώτα συστήματα Windows. Τα καλά νέα είναι ότι οι παλιές εφαρμογές μπορούν να συνεχίσουν να χρησιμοποιούνται. Τα κακά νέα είναι ότι, από τη στιγμή που λαμβάνεται μια κακή απόφαση σχεδιασμού, γενικά δεν μπορεί να διορθωθεί. Το DCOM είναι υπερβολικά σύνθετο με δεδομένο ότι δεν αποτελεί το αποτέλεσμα της δουλειάς μιας επιτροπής, όπως στην περίπτωση του CORBA.

Το Globe αποτελεί το αποτέλεσμα μιας ερευνητικής προσπάθειας έχει αναπτυχθεί από μια μικρή ομάδα η όποια προτίμησε την απλότητα από τη λειτουργικότητα, και που ήταν σε θέση να σχεδιάσει το σύστημα από το μηδέν. Δεν υπήρχε λόγος ανησυχίας για ξένα προβλήματα, ποσό μάλλον για την ανάγκη συμβατότητας με προηγούμενες εκδόσεις. Όπως ισχύει για πολλά αλλά ερευνητικά προγράμματα, ο σχεδιασμός του Globe είναι στην ουσία σαφής και απλός. Ωστόσο, το Globe εξακολουθεί να παραμένει ατελές, και ο σχεδιασμός του χρειάζεται να επεκταθεί σε πολλά σημεία. Ο κύριος στόχος σχεδιασμού του Globe ήταν η παροχή επεκτασιμότητας.

Ανάμεσα στα μοντέλα αντικειμένων που υποστηρίζει το κάθε σύστημα υπάρχουν πολύ μεγάλες διαφορές. Το CORBA και το DCOM χρησιμοποιούν και τα δύο το μοντέλο απομακρυσμένων αντικειμένων. Το CORBA παρέχει ένα ευέλικτο και συνεπές μοντέλο αντικειμένων. Η ισχύς της προσέγγισης του CORBA είναι ότι παρέχει μεγάλο βαθμό διαφάνειας όσον αφορά την πρόσβαση και τον εντοπισμό αντικειμένων. Ένας πελάτης είναι δύσκολο να διακρίνει τη διάφορα μεταξύ ενός τοπικού και ενός απομακρυσμένου αντικείμενου. Τα αντικείμενα διαθέτουν κατάσταση, μπορούν να αναγνωριστούν καθολικά, και οι αναφορές μπορούν να μεταβιβαστούν εύκολα μεταξύ πελατών διαφορετικών μηχανημάτων. Επιπλέον υποστηρίζονται τόσο παροδικά όσο και μόνιμα αντικείμενα.

Το μοντέλο DCOM είναι πολύ πιο απλό από το μοντέλο που χρησιμοποιείται στο CORBA, και μάλιστα έχει επικριθεί για την υπερβολική του απλότητα. Τα αντικείμενα CORBA είναι παροδικά, δε διαθέτουν καθόλου αναγνωριστικό, και σε μερικές περιπτώσεις θεωρούνται μη καταστασιακά. Αυτό το μοντέλο παραβιάζει πολλές από τις θεμελιώδεις αρχές που βρίσκονται πίσω από την τεχνολογία των κατανεμημένων αντικειμένων.

Το Globe παρέχει και αυτό αληθινά αντικείμενα. Ωστόσο, ένα διακριτικό γνώρισμα του μοντέλου αντικειμένων του είναι ότι τα αντικείμενα στο Globe μπορούν να αναπαραχθούν και να κατανεμηθούν από φυσική άποψη σε πολλά μηχανήματα. Επιπλέον, το αντικείμενο είναι αυτό που καθορίζει τον τρόπο με τον όποιο πραγματοποιείται η κατανομή και η αναπαραγωγή της κατάστασής του. Με αλλά λόγια, ένα αντικείμενο ενθυλακώνει τη δική του πολιτική κατανομής και υλοποίησης. Επιπλέον, μπορεί επίσης να ενθυλακώνει τις πολιτικές και τις υλοποιήσεις για ασφάλεια, χειρισμό αστοχιών, κ.ο.κ.

Πολύ μεγάλες διαφορές προκύπτουν επίσης κατά τη σύγκριση των υπηρεσιών. Το CORBA παρέχει ένα πλούσιο σύνολο υπηρεσιών τόσο πλούσιο, που πολλές από τις υπηρεσίες συχνά αλληλεπικαλύπτονται. Από την άλλη πλευρά, το DCOM παρέχει ένα μείγμα δικών του υπηρεσιών, αλλά βασίζεται επίσης σε υπηρεσίες που παρέχονται από το περιβάλλον του όπως υπηρεσίες ονομασίας και καταλόγου. Το Globe παρέχει ελάχιστες μόνο υπηρεσίες μια απλή υπηρεσία ονομασίας και μια προηγμένη υπηρεσία εντοπισμού. Η απλότητα αυτή ταιριάζει με τη φιλοσοφία σχεδιασμού του Globe, αλλά είναι προφανές ότι, για να αρχίσει πραγματικά το Globe να λειτουργεί ως κατανεμημένο σύστημα γενικής φύσης, θα χρειαστούν περισσότερες υπηρεσίες.

Μια άλλη σημαντική διάφορα ανάμεσα στα τρία συστήματα είναι το πώς βλέπουν τις διασυνδέσεις. Το CORBA ακολουθεί μια προσέγγιση με βάση τις γλώσσες προγραμματισμού παρέχοντας μια τυπική IDL η όποια χρησιμοποιείται για την αντιστοίχιση των προδιαγραφών σε διάφορες γλώσσες. Για λόγους φορητότητας και διαλειτουργικότητας αυτές οι αντιστοιχίσεις γλωσσών είναι σημαντικό να είναι τυποποιημένες. Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι το CORBA είναι ανεξάρτητο από ό,τι παράγουν οι μεταγλωττιστές. Στην ουσία, το CORBA ορίζει τη διαλειτουργικότητα στο επίπεδο των γλωσσών προγραμματισμού.

Από την άλλη πλευρά, το DCOM και το GLOBE υποστηρίζουν δυαδικές διασυνδέσεις. Από αυτή την προσέγγιση, οι διασυνδέσεις σε αντικείμενα ορίζονται ανεξάρτητα από γλώσσες προγραμματισμού. Το πλεονέκτημα των δυαδικών διασυνδέσεων στο DCOM έχει αποδειχθεί από τις πολυάριθμες εφαρμογές που είναι γραμμένες σε ένα μείγμα γλωσσών όπως η JAVA και η Visual Basic. Το GLOBE ακολουθεί την ίδια προσέγγιση.

5.1.2 Επικοινωνία

Τώρα θα εξετάσουμε τον τρόπο με τον οποίο τα συστήματα χειρίζονται την επικοινωνία. Το CORBA αρχικά διέθετε μόνο άπλες σύγχρονες κλήσεις μεθόδων για επικοινωνία. Αυτό το αρχικό μοντέλο παρείχε επίσης ασύγχρονη επικοινωνία με τη μορφή μονόδρομων και καθυστερημένων σύγχρονων αιτήσεων. Τώρα πλέον το μοντέλο επικοινωνίας του CORBA δεν είναι καθόλου απλό, και παρέχει ένα πλούσιο σύνολο δυνατοτήτων. Υπάρχει ασύγχρονη επικοινωνία με τη μορφή συμβάντων και ειδοποιήσεων, με πρόσθετες δυνατότητες για έκδοση, κατανάλωση, και φιλτράρισμα συμβάντων. Επιπλέον, το CORBA παρέχει μια υπηρεσία μηνυμάτων η όποια επιτρέπει συνδυασμούς επανακλήσεων (callbacks) και σταθμοσκοπήσεων (polling) μηνυμάτων.

Από αυτήν την άποψη, το DCOM μοιάζει με το CORBA. Παρέχει σύγχρονες κλήσεις απομακρυσμένων μεθόδων, αλλά επίσης διάφορες (και ασύμβατες μεταξύ τους) μορφές μηχανισμών επανάκλησης. Όπως το CORBA, διαθέτει επίσης υποστήριξη ανταλλαγής μηνυμάτων.

Το Globe διαθέτει μόνο σύγχρονες κλήσεις μεθόδων. Δε διαθέτει συμβάντα, επανακλήσεις, υπηρεσία μηνυμάτων, κ.ο.κ. Επειδή τα αντικείμενα στο Globe μπορεί να έχουν αναπαραχθεί, ένα αντικείμενο δεν έχει ούτε τη δυνατότητα να καλεί άλλα αντικείμενα. Μέχρι τώρα, πάντως δεν έχει χρειαστεί υποστήριξη για αλυσίδες κλήσεων. Σημειώστε ότι αυτό το πρόβλημα δεν είναι μοναδικό στο Globe, θα το αντιμετωπίσουν τόσο το CORBA όσο και το DCOM από τη στιγμή που τα αντικείμενα θα αρχίσουν να αναπαράγονται.

5.1.3 Διεργασίες

Κάθε σύστημα παρέχει τη δική του έκδοση ενός διακομιστή αντικειμένων. Οι διακομιστές αντικειμένων διαφέρουν πολύ μεταξύ τους. Ο διακομιστής αντικειμένων του CORBA είναι με πολλούς τρόπους ο πιο πολύπλευρος και ευέλικτος, και κάνει μια σαφή διάκριση μεταξύ του τι υλοποιεί ένα αντικείμενο όσον αφορά τις μεθόδους, και του τρόπου με τον οποίο γίνεται η διαχείριση των αντικειμένων και των κλήσεων μεθόδων μέσω του προσαρμογέα αντικειμένων του CORBA. Αυτή η σαφής διάκριση συμβάλλει στη δημιουργία κατανεμημένων αντικειμένων και τη χρήση τους σε ένα κατανεμημένο σύστημα.

Ο διακομιστής αντικειμένων του DCOM αποτελεί τη συμβολή της Microsoft στην ανάπτυξη κατανεμημένων αντικειμένων. Δεν είναι τόσο ευέλικτος όσο ο διακομιστής αντικειμένων του CORBA που χρησιμοποιεί POA, και έχει κυρίως τη διαχείριση αντικειμένων και την κλήση μεθόδων ενσωματωμένες στο σύστημα χρόνου εκτέλεσης. Επιπλέον, οι προγραμματιστές αντικειμένων μένουν ως ένα βαθμό αβοήθητοι όσον αφορά την υποστήριξη νημάτων. Η ενεργοποίηση JIT που παρέχεται από τους διακομιστές DCOM μπορεί να ενισχύσει την απόδοση, αλλά λειτούργει μόνο σε μη καταστασιακά αντικείμενα. Υπάρχουν σοβαρές αμφιβολίες σχετικά με το αν μια τέτοια προσέγγιση συμβαδίζει με οποιαδήποτε τεχνολογία αντικειμένων.

Ο διακομιστής αντικειμένων του Globe εξακολουθεί να είναι μάλλον πρωτόγονος, αλλά παρέχει λειτουργικότητα όμοια με αυτήν του POA στο CORBA. Αυτή η λειτουργικότητα προέρχεται κυρίως από το γεγονός ότι τα αντικείμενα μπορούν να έχουν τη δική τους συγκεκριμένη υλοποίηση ενός υποαντικειμένου έλεγχου και ενός υποαντικειμένου αναπαραγωγής. Ο διακομιστής αντικειμένων του Globe θα κάνει απλώς ότι του υπαγορεύσουν τα αντικείμενα που φιλοξένει. Δεν υπάρχει κανένας λόγος διευθέτησης του διακομιστή όπως στην περίπτωση του POA στο CORBA. Ωστόσο, καθώς τα αντικείμενα του Globe μπορούν κυρίως να αποφασίζουν μόνο για τον τρόπο κατανομής, αναπαραγωγής, και ασφάλειάς τους, υπάρχουν και αλλά είδη πολιτικών που απαιτούν υποστήριξη από το διακομιστή αντικειμένων. Για παράδειγμα, προς το παρόν υπάρχει μόνο περιορισμένος αριθμός τρόπων για την υποστήριξη της μονιμότητας.

5.1.4 Ονομασία

Το CORBA, το DCOM, και το Globe παρέχουν υποστήριξη για την ονομασία το καθένα με πολύ διαφορετικό τρόπο από τα υπόλοιπα.

Το CORBA διαθέτει πολλούς μηχανισμούς για την ονομασία και την αναφορά σε αντικείμενα. Πιο συγκεκριμένα, υπάρχουν ξεχωριστές υπηρεσίες ονομασίας, ιδιοτήτων, και ανταλλαγής. Στην ουσία, καθεμία από αυτές τις υπηρεσίες επιστρέφει μία ή περισσότερες αναφορές αντικειμένων. Μια αναφορά αντικείμενου στο CORBA ‘’δείχνει’’ απευθείας στο διακομιστή αντικείμενων που φιλοξενεί το αναφερόμενο αντικείμενο. Εναλλακτικά, ‘’δείχνει’’ σε μια αποθήκη υλοποιήσεων, η οποία παρακολουθεί τον τρέχοντα διακομιστή του αντικείμενου η ακόμα και ενεργοποιεί το αντικείμενο, αν υπάρξει ανάγκη.

Με αλλά λόγια, οι αναφορές αντικειμένων στο CORBA εξαρτώνται από τη θέση. Κατά συνέπεια, κάθε φορά που ένα αντικείμενο μετακινείται, η αναφορά του καθίσταται άκυρη. Αυτή η προσέγγιση γενικά είναι συμβατή με τοπικά κατανεμημένα συστήματα, στα όποια η υλοποίηση μιας αποδοτικής υπηρεσίας εντοπισμού είναι σχετικά εύκολη. Ωστόσο, ως γενική λύση δε θα μπορούσε ποτέ να επεκταθεί σε μεγάλης κλίμακας συστήματα ευρείας περιοχής.

Το DCOM δε διαθέτει από μόνο του υπηρεσία ονομασίας, αλλά βασίζεται στην ύπαρξη υπηρεσιών που παρέχονται στο περιβάλλον του, όπως ο Ενεργός Κατάλογος. Αν μια υπηρεσία τέτοιου είδους δεν είναι διαθέσιμη, η αναζήτηση ενός αντικείμενου με το όνομά του είναι αδύνατη.

Ένα είδος αναφορών αντικειμένων στο DCOM είναι οι δείκτες διασύνδεσης. Η παράταξη ενός δείκτη τέτοιου είδους γενικά απαιτεί την παράταξη και μεταφορά του διαμεσολαβητή του πελάτη στον προορισμό. Η χρήση τυπικής παράταξης απλοποιεί λίγο τα πράγματα, αλλά γενικά η μεταβίβαση ενός δείκτη διασύνδεσης ως αναφοράς αντικείμενου σε ένα άλλο πελάτη είναι μια δύσκολη δουλειά, ειδικά όταν απαιτείται κάτι ειδικό.

Μόνιμες αναφορές αντικειμένων παρέχονται με τη μορφή των ψευδωνύμων (monikers), τα όποια οι Orfali κ.α. (1996) αποκαλούν ‘’μπαλώματα’’ (kludges). Θα πρέπει να υπογραμμίσουμε ότι η κατανόηση των ψευδωνύμων δεν είναι εύκολη. Λύνουν πολλά προβλήματα που σχετίζονται με την αναφορά αντικειμένων, αλλά παραμένει το ερώτημα για το αν αποτελούν την καλύτερη λύση.

Το Globe διαφέρει ριζικά από το CORBA και το DCOM ως προς το ότι διαχωρίζει την ονομασία από τη διευθυνσιοδότηση αντικειμένων. Η χρήση αναγνωριστικών αντικειμένων με μεγάλη διάρκεια ζωής και ανεξάρτητων θέσης, επιτρέπει την τροποποίηση των ονομάτων των αντικειμένων χωρίς να επηρεάζεται η αντιστοίχιση ονομάτων προς διευθύνσεις, η όποια συνήθως φιλοξενείται μέσω παραδοσιακών υπηρεσιών ονομασίας. Παρόμοια η αντιστοίχιση ονομάτων προς διευθύνσεις δεν επηρεάζεται ούτε από την αλλαγή μια διεύθυνσης. Το κύριο μειονέκτημα της προσέγγισης στο Globe είναι ότι απαιτείται μια ξεχωριστή, παγκόσμιας κλίμακας επεκτάσιμη υπηρεσία εντοπισμού. Η ανάπτυξη μιας υπηρεσίας τέτοιου είδους κάθε άλλο παρά εύκολη υπόθεση αποτελεί.

Η ονομασία αντικειμένων στο Globe μέσω αλφαριθμητικών χαρακτήρων είναι σχετικά απλή. Αυτό που λείπει είναι υπηρεσίες καταλόγου οι οποίες θα δίνουν σε έναν πελάτη τη δυνατότητα να αναζητά αντικείμενα με βάση ζητούμενες ιδιότητες. Μια υπηρεσία τέτοιου είδους είναι απαραίτητη για πολλά κατανεμημένα συστήματα.

5.1.5 Συγχρονισμός

Μηχανισμοί συγχρονισμού παρέχονται με τη μορφή παραδοσιακών υπηρεσιών κλειδώματος και συναλλαγών, οι οποίες υλοποιούνται μόνο για το CORBA και το DCOM. Το Globe δε διαθέτει άλλους αποκλειστικούς μηχανισμούς συγχρονισμού εκτός από εκείνους που υλοποιούνται για συγχρονισμούς στο εσωτερικό αντικειμένων, μέσω των υποαντικειμένων αναπαραγωγής του.

5.1.6 Χρήση κρυφής μνήμης και αναπαραγωγή

Τα τρία συστήματα έχουν επίσης σημαντικές διαφορές όσον αφορά τη χρήση κρυφής μνήμης και την αναπαραγωγή.

Η προσθήκη αναπαραγωγής CORBA αποτέλεσε μια προσπάθεια που αναβαλλόταν συνεχώς, κυρίως εξαιτίας των δυσκολιών του συνδυασμού της με την έννοια των απομακρυσμένων αντικειμένων. Η προσέγγιση που υιοθετείται σήμερα είναι ότι ένας ξεχωριστός διακομιστής αναπαραγωγής δημιουργεί την ψευδαίσθηση ότι μια ομάδα αντικειμένων εμφανίζεται ως ένα ενιαίο απομακρυσμένο αντικείμενο. Με αλλά λόγια, για τον πελάτη ουσιαστικά δεν υπάρχει διάφορα ανάμεσα σε ένα αναπαρηγμένο και ένα μη αναπαρηγμένο αντικείμενο.

Ο διακομιστής αναπαραγωγής καθορίζει πώς, ποτέ, και πού πραγματοποιείται η αναπαραγωγή. Σήμερα υπάρχουν αρκετές πολιτικές αναπαραγωγής που μπορούν να συσχετιστούν με μια ομάδα αντικειμένων. Όλες αυτές οι πολιτικές υλοποιούν σειριακή συνέπεια. Μια τυπική υλοποίηση βασίζεται σε ένα μηχανισμού πολυεκπομπής ολικής διάταξης.

Όσον αφορά τη χρήση κρυφής μνήμης δεν υπάρχει κάποια ειδική πρόβλεψη, αλλά η λειτουργία αυτή μπορεί να προστεθεί εύκολα σε μια υπάρχουσα υλοποίηση με τη χρήση υποκλοπέων. Η υλοποίηση ενός υποσυστήματος χρήσης κρυφής μνήμης επαφίεται, ωστόσο, ολοκληρωτικά στον προγραμματιστή της εφαρμογής. Το πρόβλημα με αυτή την προσέγγιση είναι ότι μπορεί να απαιτεί σημαντική προσπάθεια.

Το DCOM δεν παρέχει καμία υποστήριξη για χρήση κρυφής μνήμης και αναπαραγωγή. Αυτά τα ζητήματα εξαρτώνται ολοκληρωτικά από τις εφαρμογές.

Το Globe υιοθετεί μια διαφορετική προσέγγιση από το CORBA, ενσωματώνοντας ένα υποαντικείμενο αναπαραγωγής σε κάθε τοπικό αντικείμενο ενός κατανεμημένου κοινόχρηστου αντικείμενου. Επειδή η διασύνδεση με τα υποαντικείμενα αναπαραγωγής είναι τυποποιημένη, με ένα αντικείμενο μπορούν να συσχετιστούν εύκολα διαφορετικές υλοποιήσεις διαφόρων στρατηγικών αναπαραγωγής. Το αποτέλεσμα είναι υψηλός βαθμός ευελιξίας πολύ υψηλότερος από τη περιορισμένη λίστα στρατηγικών που υποστηρίζει το CORBA. Ο ίδιος μηχανισμός μπορεί να χρησιμοποιηθεί για τη δημιουργία λειτουργιών κρυφής μνήμης.

Παρόλο που το Globe παρέχει ένα γενικό μηχανισμό για το συσχετισμό στρατηγικών αναπαραγωγής με αντικείμενα, ακόμα δε διαθέτει μεγάλη συλλογή υλοποιημένων υποαντικειμένων αναπαραγωγής. Με αλλά λόγια, ένας προγραμματιστής προς το παρόν συχνά θα πρέπει να γράφει τη δική του υλοποίηση. Με αυτή την έννοια, τα υποαντικείμενα αναπαραγωγής του Globe είναι συγκρίσιμα με τους υποκλοπείς του CORBA ως προς το ότι παρέχουν μόνο μηχανισμούς αλλά όχι και υποσυστήματα που θα μπορούσαν να χρησιμοποιηθούν άμεσα για την ανάπτυξη συστημάτων κατανεμημένων αντικειμένων.

5.1.7 Ανοχή σε βλάβες

Η ανοχή σε βλάβες στο CORBA υποστηρίζεται κυρίως μέσω της αναπαραγωγής. Επιπλέον, οι υπηρεσίες του CORBA για συναλλαγές και έλεγχο ταυτοχρονισμού βοηθούν στην ανάπτυξη ανεκτικών σε βλάβες υπηρεσιών.

Το DCOM παρέχει παρόμοια υποστήριξη για ανοχή σε βλάβες μέσω των αυτομάτων συναλλαγών του, οι οποίες υποστηρίζονται επιπλέον από έναν ξεχωριστό συντονιστή συναλλαγών.

Το Globe δεν παρέχει συναλλαγές η υποστήριξη του για ανοχή σε βλάβες γίνεται κυρίως μέσω της αναπαραγωγής. Δεν παρέχεται σχεδόν κανένας μηχανισμός για ανάκαμψη από κατάρρευση, εκτός από τη δυνατότητα εγγραφής της κατάστασης σε κάποιο τοπικό μόνιμο αποθηκευτικό μέσο.

5.1.8 Ασφάλεια

Τέλος, η σύγκριση της ασφάλειας στο CORBA, το DCOM, και το Globe, έχει και πάλι ως αποτέλεσμα πολύ μεγάλες διαφορές.

Το CORBA ορίζει μια ολοκληρωμένη αρχιτεκτονική ασφάλειας για απομακρυσμένα αντικείμενα. Έχει γίνει προσπάθεια να διαχωριστούν με σαφή τρόπο οι μηχανισμοί από τις πολιτικές, με την παροχή μιας πληθώρας μηχανισμών για τον καθορισμό του πότε και πού θα πρέπει να εφαρμόζονται οι λειτουργίες ασφάλειας. Η ίδια η ασφάλεια έχει τη μορφή αντικειμένων πολιτικής που εξαρτώνται από την εκάστοτε εφαρμογή, και τα οποία

καλούνται όταν γίνεται επεξεργασία μιας αίτησης ή απάντησης από έναν ORB. Ένα αντικείμενο πολιτικής μπορεί να διευθετηθεί από μια εφαρμογή, και γενικά χρησιμοποιεί υπηρεσίες ασφάλειας που είναι διαθέσιμες τοπικά. Για παράδειγμα, ένα αντικείμενο πολιτικής για πιστοποίηση ταυτότητας μπορεί να βασίζεται στο Kerberos πρωτόκολλο ασφάλειας.

Ο μηχανισμός που χρησιμοποιείται για την κλήση αντικειμένων πολιτικής είναι και πάλι η υποκλοπή κλήσεων. Για να γίνει αυτό με ασφαλή τρόπο, το CORBA παρέχει ξεχωριστούς ασφαλείς υποκλοπείς για την κλήση ενός αντικείμενου πολίτικης και τον έλεγχο πρόσβασης.

Το DCOM ακολουθεί μια διαφορετική προσέγγιση. Η ασφάλεια μπορεί να υποστηριχθεί με δηλωτικό τρόπο, μέσω του καθορισμού στο μητρώο των απαιτήσεων που έχει ένα αντικείμενο όσον αφορά την ενεργοποίηση, τον έλεγχο πρόσβασης, και την πιστοποίηση ταυτότητας. Το λειτουργικό σύστημα σε συνδυασμό με το DCOM θα φροντίσουν κατόπιν ώστε να καλύπτονται αυτές οι απαιτήσεις. Μια εναλλακτική προσέγγιση είναι να μπορούν να καθορίζουν οι εφαρμογές τις υπηρεσίες ασφάλειας που θέλουν, μέσω της κλήσης μιας ειδικής λειτουργίας κατά την προετοιμασία των αντικειμένων.

Στο Globe, η διαχείριση της ασφάλειας γίνεται εν μέρει από ένα ξεχωριστό υποαντικείμενο ασφάλειας. Αυτό το υποαντικείμενο επικοινωνεί με υπηρεσίες τοπικής ασφάλειας, όπως το Kerberos. Ακόμα, χειρίζεται ζητήματα ασφάλειας που αφορούν την επικοινωνία, την αναπαραγωγή, και την κλήση του υποαντικειμένου σημασιολογίας. Ωστόσο, η αρχιτεκτονική ασφάλειας στο Globe είναι ακόμα ατελής. Για παράδειγμα, δεν έχει προσδιοριστεί ακόμα πως ένα τοπικό αντικείμενο λαμβάνει πραγματικά το υποαντικείμενο ασφάλειας που θέλει, αντί για ένα κακοπροαίρετο υποαντικείμενο.

Στην Εικόνα (5.1) συνοψίζονται τα διάφορα ζητήματα που έχουμε περιγράψει σε αυτή την ενότητα συγκρίνοντας το CORBA, το DCOM, και το Globe.

Ζήτημα	**CORBA**	**DCOM**	**Globe**
Στόχοι σχεδιασμού	Διαλειτουργικότητα	Λειτουργικότητα	Επεκτασιμότητα
Μοντέλο αντικειμένων	Απομακρυσμένα αντικείμενα	Απομακρυσμένα αντικείμενα	Κατανεμημένα αντικείμενα
Υπηρεσίες	Πολλές δικές της	Από το περιβάλλον	Λίγες
Διασυνδέσεις	Με βάση IDL	Δυαδικές	Δυαδικές
Σύγχρονη επικοινωνία	Ναι	Ναι	Ναι
Ασύγχρονη επικοινωνία	Ναι	Ναι	Όχι
Επανακλήσεις	Ναι	Ναι	Όχι
Συμβάντα	Ναι	Ναι	Όχι
Υπηρεσία μηνυμάτων	Ναι	Ναι	Όχι
Διακομιστής αντικειμένων	Ευέλικτος (POA)	Ενσωματωμένος	Ανάλογα με το αντικείμενο
Υπηρεσία καταλόγου	Ναι	Ναι	Όχι
Υπηρεσία ανταλλαγής	Ναι	Όχι	Όχι
Υπηρεσία ονομασίας	Ναι	Ναι	Ναι
Υπηρεσία εντοπισμού	Όχι	Όχι	Ναι
Αναφορά Αντικειμένων	Θέση αντικείμενου	Δείκτης διασύνδεση	Πραγματικό αναγνωριστικό
Συγχρονισμός	Συναλλαγές	Συναλλαγές	Μόνο εσωτερικό αντικειμένων
Υποστήριξη αναπαραγωγής	Ξεχωριστός διακομιστής	Καμία	Ξεχωριστό υποαντικείμενο
Συναλλαγές	Ναι	Ναι	Όχι
Ανοχή σε βλάβες	Μέσω αναπαραγωγής	Μέσω συναλλαγών	Μέσω αναπαραγωγής
Υποστήριξη ανάκαμψης	Ναι	Μέσω συναλλαγών	Όχι
Ασφάλεια	Διάφοροι μηχανισμοί	Διάφοροι μηχανισμοί	Χρειάζεται βελτίωση

Εικόνα (5.1) Μια σύγκριση των CORBA, DCOM, και Globe.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ

Το ενδιάμεσο λογισμικό χρησιμοποιείται στα κατανεμημένα συστήματα κυρίως ως διαμεσολαβητής μεταξύ των εφαρμογών, αιτήσεων και των κατώτερων επιπέδων μεταφοράς. Ακόμη καλύπτει την ετερογένεια μεταξύ των υπολογιστικών συστημάτων και «κρύβει» από τους χρήστες τυχόν προβλήματα σε κάποιο μέρος του δικτύου. Κάποιες τεχνολογίες είναι το Java RMI που κάνει ακριβώς αυτό, δηλαδή καλύπτει την ετερογένεια και δείχνει το σύστημα σαν ενιαίο στον χρήστη.

Τα κατανεμημένα συστήματα που βασίζονται σε αντικείμενα χρησιμοποιούν σχεδόν πάντοτε το μοντέλο απομακρυσμένων αντικειμένων. Αυτή η προσέγγιση απαιτεί τη λήψη ειδικών μέτρων όταν χρειάζεται προσαρμογή, για την υποστήριξη, λόγω κρυφής μνήμης ή αναπαραγωγής. Το CORBA χρησιμοποιεί υποκλοπείς για την πραγματοποίηση εξερχόμενων και εισερχόμενων αιτήσεων κλήσεων. Στο DCOM η προσαρμοσμένη παράταξη επιτρέπει την προσαρμογή ενός διαμεσολαβητή στην πλευρά του πελάτη ανάλογα με τις ανάγκες. Το Globe αντιμετωπίζει την προσαρμογή επιτρέποντας σε διαφορετικά υποαντικείμενα, το καθένα από τα όποια βρίσκεται σε διαφορετικό μηχάνημα, να συνυπάρξει στο ίδιο κατανεμημένο αντικείμενο.

Εκτός από τις σύγχρονες κλήσεις μεθόδων, τα κατανεμημένα συστήματα με βάση τα αντικείμενα όπως το CORBA και το DCOM παρέχουν εναλλακτικές μορφές κλήσεων, μεταξύ των οποίων είναι τα συμβάντα και οι ασύγχρονες κλήσεις μεθόδων. Το Globe δεν υποστηρίζει αυτές τις εναλλακτικές λύσεις, και ουσιαστικά παρέχει μόνο σύγχρονες κλήσεις.

Η οργάνωση των διακομιστών αντικειμένων είναι παρεμφερής, ως ένα βαθμό, σε όλα τα συστήματα. Σε όλες τις περιπτώσεις, ένας διακομιστής έχει δυνατότητα να υποστηρίξει περισσότερα από ένα αντικείμενα. Υπάρχουν, ωστόσο, διαφορές όσον αφορά την ευέλικτη προσαρμογή ενός διακομιστή. Στο CORBA η ευελιξία επιτυγχάνεται μέσω προσαρμογής αντικειμένων. Το DCOM παρέχει έναν τυπικό διακομιστή αντικειμένων, ο οποίος μπορεί να προσαρμοστεί για συγκεκριμένες εφαρμογές. Οι διακομιστές αντικείμενων του Globe είναι σχετικά απλοί, καθώς τυχόν συγκεκριμένες λειτουργιές θεωρείται ότι θα πρέπει να υλοποιούνται ως μέρος ενός κατανεμημένου αντικείμενου.

Όσον αφορά την ονομασία των κατανεμημένων συστημάτων που βασίζονται σε αντικείμενα, διαφορές εμφανίζονται στο επίπεδο των αναφορών αντικειμένων. Η υποστήριξη με ονόματα φιλικά προς τον άνθρωπο γενικά είναι ίδια παντού, σε μεγάλο βαθμό. Τόσο στο CORBA όσο και στο DCOM οι αναφορές αντικειμένων με εύρος σε όλο το σύστημα διευθετούνται από τη θέση. Αυτές οι αναφορές περιέχουν πληροφορίες για τη

θέση του διακομιστή όπου βρίσκετε το αντικείμενο. Αντίθετα, το Globe χρησιμοποιεί αναφορές, βασιζόμενο όμως σε μια παγκόσμια υπηρεσία εντοπισμού για την ανάκτηση των αναφορών σε διευθύνσεις επικοινωνίας. Μια διεύθυνση επικοινωνίας δηλώνει την θέση ενός αντικείμενου και τον τρόπο επικοινωνίας μαζί του.

Το CORBA υποστηρίζει αναπαραγωγή μόνον όσον αφορά την ανοχή σε βλάβες. Το DCOM δεν παρέχει καθόλου αναπαραγωγή, ένας προγραμματιστής θα πρέπει να χρησιμοποιεί εξειδικευμένους διακομιστές αναπαραγωγής. Το Globe υποστηρίζει αναπαραγωγή ανά αντικείμενο, μέσω ενός υποαντικειμένου αναπαραγωγής που υλοποιεί ένα συγκεκριμένο πρωτόκολλο για το αντικείμενο του οποίου αποτελεί μέρος.

Σχετική με την αναπαραγωγή είναι η υποστήριξη για ανοχή σε βλάβες. Το CORBA έχει ανοχή σε βλάβες μέσω μιας υπηρεσίας αναπαραγωγής, η όποια βασίζεται σε μια υποτιθέμενη αξιόπιστη υπηρεσία πολυεκπομπής. Στο DCOM, η ανοχή σε βλάβες υποστηρίζεται μέσω (αυτόματων) συναλλαγών. Το Globe υποστηρίζει ανοχή σε βλάβες μόνο μέσω αναπαραγωγής, χωρίς όμως να παρέχει μηχανισμούς ανάκαμψης. Τέλος, καθένα από τα κατανεμημένα συστήματα που περιγράφονται παραπάνω διαθέτουν μηχανισμούς ασφάλειας.

Αναλογιζόμενοι όλα αυτά τα στοιχεία διαπιστώνουμε ότι το ενδιάμεσο λογισμικό (middleware) θα είναι η καρδιά, στο μέλλον, των κατανεμημένων συστημάτων ώστε να υποστηρίζεται η καλύτερη χρήση των πόρων ανεξαρτήτως της φυσική τους τοποθεσίας και να δίνει στο χρήστη την εικόνα ενός ενιαίου συστήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] A. Sinha, Client-Server Computing, Magazine Communications of the ACM, 1992

[2] M. D. McIlroy, Mass produced software components, Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, pp. 88-98, 1968

[3] S. M. Lewandowski, Frameworks for component based client/server computing, ACM Computing Surveys, 1998

[4] Ed. Robert Orfali, Dan Harkey, Client / Server Programming with Java and Corba 2nd, John Wiley & sons Inc, 1998

[5] Χρήστος Γαρίδης, Enterprise Java Beans, Πανεπιστήμιο Μακεδονίας, 2005

[6] www.wikipedia.org, λήμμα middleware

[7] http://www.infosoc.gr/infosoc/el-GR/services/elibrary/reports_list/prodiagrafes_dialeitourg_plirofor_systim/eld.htm

[8] Object Management Group, Inc., CORBA Success Stories, 2007

[9] N. Wang, D. C. Schmidt, C. O' Ryan, Overview of the CORBA Component Model, Component-based software engineering: putting the pieces together, Addison-Wesley, 2001

[10] Σ. Κάντας, «Αξιοποίηση της τεχνολογίας συνιστωσών στην ανάπτυξη λογισμικού ενσωματωμένων συστημάτων πραγματικού χρόνου», Διπλωματική εργασία, 2004

[11] Need for CORBA Component Architecture, http://www.icmgworld.com K2-CCM Container

[12] Henning and Vinoski, Advanced CORBA Programming with C++ , Addison-Wesley Professional Computing Series, 1999

[13] OMG Standards, CORBA 2.4 , OMG 2000, http://www.omg.org/

[14] Δούδαλης Γιώργος, << Ανάπτυξη στρώματος μεσισμικού CORBA και ενσωμάτωση του στο λογισμικό κατανεμημένης/παράλληλης επεξεργασίας WebXPVM>>, Διπλωματική εργασία ΕΜΠ, 2004

[15] Tanenbaum, Andrew S. Van Steen, Maarten, Κατανεμημένα συστήματα: Αρχές και υποδείγματα, Εκδόσεις Κλειδάριθμος, 2005

[16]http://el.wikipedia.org**,** λήμμα Παράλληλα και κατανεμημένα συστήματα

[17] Grimshaw Jane, Argument structure, MIT Press, Cambridge: MIT Press, vii 201p. 1994