

**ΠΑΠΑΓΕΩΡΓΙΟΥ ΑΝΤΩΝΗΣ 3339 ετος 7**

**ΑΝΔΡΕΑΣ ΜΕΝΟΙΚΟΥ 3624 ετος 6**

**Επιβλέπων Καθηγητής: Μηνάς Δασυγένης**

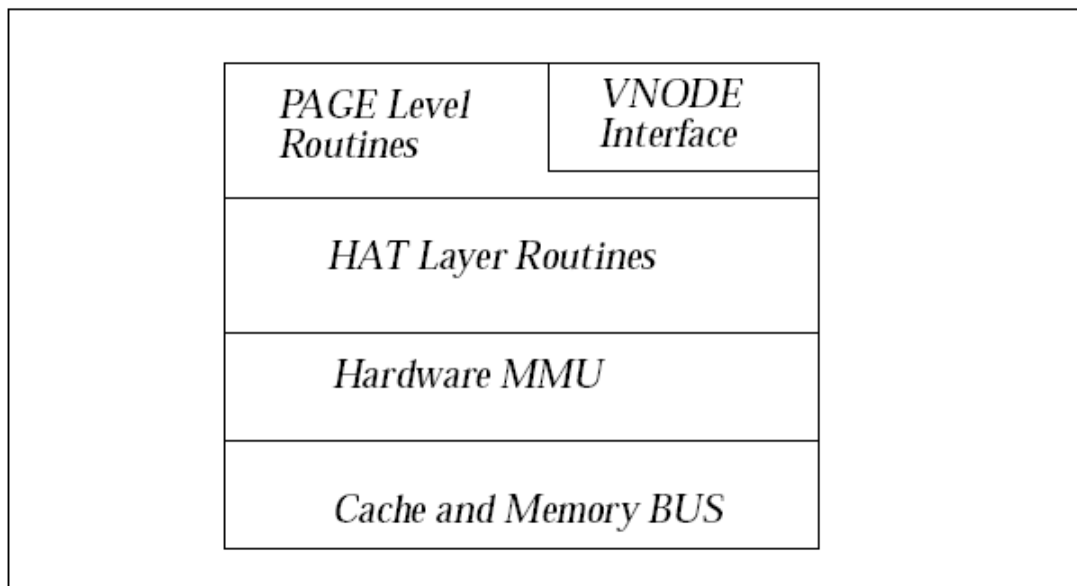
**ΕΡΓΑΣΙΑ** Να περιγράψατε τη διαχείριση μνήμης και τη διαδιεργασιακή επικοινωνία στο λειτουργικό σύστημα solaris . Να κατασκευάσατε σε αυτό το λειτουργικό σύστημα μια διεργασία η οποία θα στέλνει ένα μήνυμα σε μια άλλη διεργασία και θα τα αποθηκεύει σε ένα αρχείο μηνυμάτων.

### **ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ ΣΤΟ SOLARIS**

#### **Το επίπεδο HAT**

Τη σύνδεση μεταξύ της φυσικής μνήμης RAM και της δομής του page διαχειρίζεται το επίπεδο Hardware Address Translation layer (HAT layer). Το επίπεδο αυτό είναι ένα συγκεκριμένο σύνολο ρουτινών που διαχειρίζεται τους χάρτες και τη μετάφραση των διευθύνσεων μεταξύ της δομής του page και του MMU Hardware pages. Αυτές οι ρουτίνες καλούνται για να δημιουργήσουν pull down τις μεταφράσεις των σελίδων κάθε φορά που ένα page δημιουργείται η διαγράφεται .

Το επίπεδο HAT χειρίζεται ακόμα και τα traps , έτσι ώστε όταν γίνει μια αναφορά σε περιοχή της VIM που δεν έχει προς το παρόν φυσική σελίδα στον πυρήνα ξεκινά η ρουτίνα fault για να επαναφέρει τη σελίδα από την αποθηκευτική περιοχή.



*Pages as Vnode and Offset*

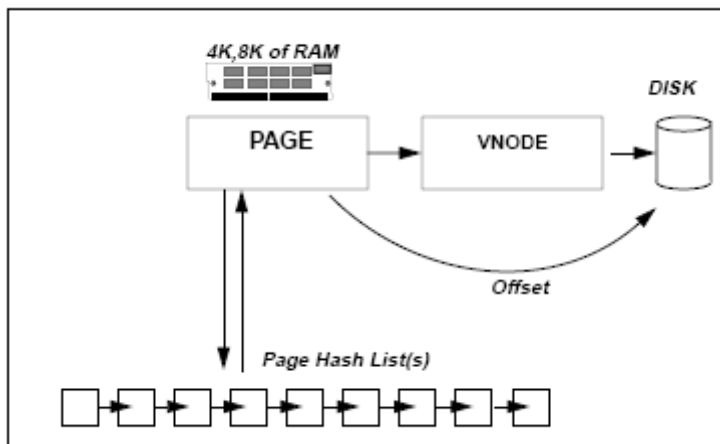
Στο solaris υπάρχει πάντα μια vnode με μια κατανεμημένη σελίδα

Φυσικής μνήμης. Κάθε σελίδα της μνήμης περιγράφεται από ένα vnode και ένα offset μέσα στο vnode. Ο σκοπός του vnode είναι για να μας δείξει τον αποθηκευτικό χώρο για αυτή τη σελίδα της μνήμης.

Αν η σελίδα εφαρμογή μνήμης, τότε το vnode για αυτή τη σελίδα είναι το SWAP. Για να βρεθεί μια συγκεκριμένη σελίδα στη μνήμη, σύστημα VM χρησιμοποιά το vnode και το offset σαν κλειδί hash για να βρει το δείκτη στη σελίδα. Το σύστημα VM χρησιμοποιεί την εντολή page\_find για να εντοπίσει τις σελίδες ψάχνοντας την λίστα hash.

Στον κατάλογο σελίδων hash, υπάρχουν ακόμα δυο άλλοι κατάλογοι σελίδων. Είναι ο ελεύθερος κατάλογος, και ο κατάλογος cache. Ο ελεύθερος κατάλογος είναι ένας hashed κατάλογος σελίδων που δεν έχουν χαρτογραφηθεί στο VIM. Ο κατάλογος cache είναι ένας κατάλογος σελίδων που είναι ελεύθερες αλλά είναι χαρτογραφημένες σε ένα συγκεκριμένο vnode και offset. Το συνολικό ποσό της ελεύθερης μνήμης είναι το άθροισμα του ελεύθερου καταλόγου σελίδων και του cache καταλόγου σελίδων.

Οι cache σελίδες καταλόγου μπορούν να ξαναχρησιμοποιηθούν αν το σύστημα VM χρειάζεται να δημιουργήσει ένα νέο χάρτη για μια σελίδα η οποία είναι ήδη στη μνήμη, αλλά ελευθερώθηκε από τον τελευταίο χρήστη. Η cache λίστα καταλόγου σταματά το σύστημα από το να διαγράψει και να δημιουργήσει τις ίδιες σελίδες ξανά.



Κάθε σελίδα έχει μια σημαία κατάστασης η οποία υποδεικνύει αν η σελίδα είναι ελεύθερη, αν έχει γίνει αναφορά σε αυτή η αν έχει τροποποιηθεί. η πληροφορία της σημαίας κατάστασης συγχρονίζεται από τους καταχωρητές μέσα στη MMU από το επίπεδο HAT κάθε φορά που η δομή της σελίδας καλείται ή όταν καλείται η εντολή hat\_sync() του επιπέδου HAT.

Η δομή της σελίδας έχει περισσότερα στοιχεία από αυτά που περιγράφονται στο σχήμα, τα περισσότερα από τα οποία είναι κλειδιά και καταστάσεις μεταβλητών τα οποία ενώ χρησιμοποιούνται στην επεξεργασία σημάτων μπορεί να περιμένουν για μια εφαρμογή I/O στη σελίδα.

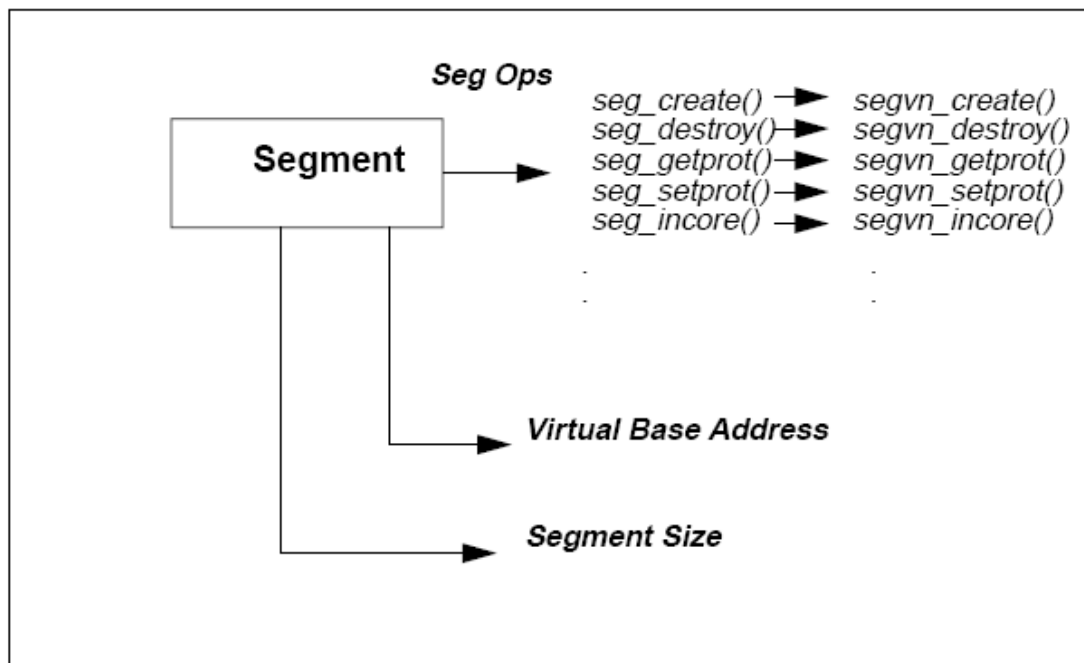
## Virtual Address Spaces

### Memory Segments

Γνωρίζουμε ότι οι σελίδες VIM χαρτογραφούνται σε φυσικές σελίδες μέσω του MMU και του επιπέδου HAT , και ότι κάθε σελίδα έχει κάποιας κατάστασης στήριξη αποθήκευσης. Ο δεσμός που λείπει είναι το πώς οι σελίδες σχετίζονται σε ένα γραμμικό χώρο διευθύνσεων , αυτό δηλαδή που οι εφαρμογές περιμένουν να δουν.

Η σχέση μεταξύ των σελίδων και του γραμμικού χώρου διευθύνσεων διαχειρίζεται από τμήματα της μνήμης.

Ένα τμήμα είναι μια χαρτογράφηση μιας συγκεκριμένης διεύθυνσης μνήμης και μήκους σε μια συσκευή. Υπάρχει επίσης ένα αντικείμενο προσανατολισμένο στη διασύνδεση του τμήματος το οποίο εξασφαλίζει μια συσκευή ανεξάρτητη από εκεί που είναι χαρτογραφημένο το τμήμα. Αυτά ονομάζονται οδηγοί τμημάτων.



Ο πιο κοινός οδηγός τμήματος στο Solaris είναι το τμήμα vnode , ή segvn , το οποίο χρησιμοποιείται για να χαρτογραφήσει ένα vnode σε μια συγκεκριμένη εικονική θέση και offset. Το τμήμα vnode χρησιμοποιείται για:

- (1) ανώνυμη εφαρμογή μνήμης
- (2) εκτελέσιμα αρχεία τα οποία έχουν το πρόγραμμα αρχείου στο σύστημα αρχείων

(3) κανονικά αρχεία , οπού ένα αρχείο έχει σελίδες στη μνήμη

Υπάρχουν επίσης άλλοι τύποι στη μνήμη που δεν συνεργάζονται με σελίδες ή nodes . Αυτοί οι τύποι τμημάτων συνεργάζονται με συσκευές όπως οι προσαρμογής γραφικών.

### **Segment Protection**

Κάθε τμήμα χαρτογραφείται με μια σχετική προστασία , η οποία είναι ένας συνδυασμός των :

- EXEC η χαρτογράφηση επιτρέπεται να έχει μηχανές κώδικα που εκτελούνται μέσα στην ακτίνα διευθύνσεων του
- READ η χαρτογράφηση επιτρέπεται να διαβαστεί , και το γράψιμο θα παράγει το SIGSEGV αν δεν επιτρέπεται η εγγραφή
- WRITE η χαρτογράφηση επιτρέπει την εγγραφή και η ανάγνωση θα δημιουργήσει το SIGSEGV αν δεν επιτρέπεται η ανάγνωση
- SHARED όλες οι εγγραφές σε αυτό το τμήμα μοιράζονται με άλλα τμήματα συμπεριλαμβάνοντας άλλες διεργασίες
- PRIVATE η εγγραφή σε αυτή τη περιοχή θα σταματήσει από το VIM και θα κατανεμηθεί μια ιδιωτική σελίδα με ανώνυμη μνήμη στη διεύθυνση εγγραφής . αυτό ονομάζεται Copy On Write (COW).

### **Process Address Spaces as Mapped Segments**

Τα τμήματα είναι τυπικά :

### ***Executable Text***

Μια χαρτογράφηση του εκτελέσιμου κομματιού του binary , χαρτογραφημένο ως read only .

### ***Executable Data***

Μια χαρτογράφηση των data του τμήματος του binary , που περιέχει αρχικές μεταβλητές από το binary . το τμήμα data χαρτογραφείται read, write, και private έτσι ώστε οι αλλαγές στο τμήμα binaries data δεν επηρεάζει άλλες διεργασίες. Αυτό συμβαίνει όταν ένα πρόγραμμα αλλάζει την τιμή μιας από τις αρχικές του μεταβλητές.

### ***Heap***

Το πρόγραμμα heap περιέχει όλα τα προγράμματα ανωνυμής μνήμης, που κατανομούνται συνήθως μέσω του malloc() or brk().

### ***Shared Library Text and Data***

Αυτά τα τμήματα χαρτογραφούνται με τις ίδιες προστασίες όπως τα εκτελέσιμα

### ***Optional mmap ()ed files.***

Τα αρχεία μπορούν να χαρτογραφηθούν σε ένα χώρο διευθύνσεων με την κλήση συστήματος mmap(). Μπορούν να χαρτογραφηθούν με οποιαδήποτε προστασία εκτός την EXEC

### ***Stack***

Ο σωρός προγράμματος είναι μια ξεχωριστή χαρτογράφηση ανώνυμης μνήμης η οποία χαρτογραφείται σαν read και write . ένα παράδειγμα επεξεργασίας του χώρου διευθύνσεων φαίνεται χρησιμοποιώντας την εντολή pmap.

```
# pmem 25888
or
# /usr/proc/bin/pmap -x 25888
```

25888: ksh

Addr	Size	Res	Shared	Priv	Prot	Segment-Name
00010000	184K	184k	184k	0k	read/exec	/bin/ksh
0004C000	8K	8k	8k	0k	read/write/exec	/bin/ksh
0004E000	40K	40k	0k	40k	read/write/exec	[ heap ]
EF5E0000	16K	16k	8k	8k	read/exec	/usr/lib/locale/en_AU.so.1
EF5F2000	8K	8k	0k	8k	read/write/exec	/usr/lib/locale/en_AU.so.1
EF600000	592K	568k	560k	8k	read/exec	/usr/lib/libc.so.1
EF6A2000	24K	24k	8k	16k	read/write/exec	/usr/lib/libc.so.1
EF6A8000	8K	8k	0k	8k	read/write/exec	
EF6B0000	8K	0k	0k	0k	read/write/exec	
EF6C0000	16K	16k	16k	0k	read/exec	/usr/lib/libc_psr.so.1
EF6D0000	16K	16k	16k	0k	read/exec	/usr/lib/libtmp.so.2
EF6E2000	8K	8k	8k	0k	read/write/exec	/usr/lib/libtmp.so.2
EF700000	448K	400k	400k	0k	read/exec	/usr/lib/libnsl.so.1
EF77E000	32K	32k	8k	24k	read/write/exec	/usr/lib/libnsl.so.1
EF786000	24K	8k	0k	8k	read/write/exec	
EF790000	32K	32k	32k	0k	read/exec	/usr/lib/libsocket.so.1
EF7A6000	8K	8k	8k	0k	read/write/exec	/usr/lib/libsocket.so.1
EF7A8000	8K	0k	0k	0k	read/write/exec	
EF7B0000	8K	8k	8k	0k	read/exec/shared	/usr/lib/libdl.so.1
EF7C0000	112K	112k	112k	0k	read/exec	/usr/lib/ld.so.1
EF7EA000	16K	16k	8k	8k	read/write/exec	/usr/lib/ld.so.1
EFFFC000	16K	16k	0k	16k	read/write/exec	
EFFFC000	16K					[ stack ]
-----		-----	-----	-----		
	1632K	1528k	1384k	144k		

## **The Pageout Process**

Ένα επιπρόσθετο κομμάτι του συστήματος VM είναι ο σαρωτής pageout. Είναι εγκατεστημένο σε boot-time σαν διεργασία του kernel . ο σκοπός του είναι να απελευθερώνει μνήμη όταν το ποσό της ελεύθερης μνήμης φτάνει σε επικίνδυνα επίπεδα.

## **Basis of Operation**

Η σάρωση pageout βασίζεται στο γενικό κώδικα που παρουσιάζεται στα unix. Και χρησιμοποια ένα πρότυπο του nru .Το pageout daemon ελέγχει 4 φορές το δευτερόλεπτο για να δει αν η ελεύθερη μνήμη πέφτει κάτω από το lotsfree, μια παράμετρος η οποία ελέγχει τη σάρωση pageout. Ο σαρωτής ξεκινά επίσης όταν γίνεται κλήση μνήμης και ο ελεύθερος κατάλογος είναι κάτω από ένα επίπεδο. Ο σαρωτής είναι υπεύθυνος είναι υπεύθυνος να καθορίσει ποιες σελίδες της μνήμης να απελευθερώσει.

### **Pageout Scanner**

Η φυσική μνήμη ram παρίσταται με τις 12 ώρες του ρολογιού . υπάρχουν 2 χεριά τα οποία περιστρέφονται γύρω από το ρολόι με την ίδια ταχύτητα , το ένα ελαφρώς πιο μπροστά από το άλλο . καθώς τα χεριά γυρίζουν , αυτό που είναι μπροστά καθαρίζει την σημαία αναφοράς στη σελίδα . το πίσω τότε ελέγχει τη σελίδα για να δει αν έχει γίνει αναφορά στη σελίδα από τότε που αυτό που είναι μπροστά καθάρισε τις σημαίες. Αν δεν έχει γίνει αναφορά η δεν έχει τροποποιηθεί τότε είναι υποψηφία για να διαγραφεί.

### **The Memory Scheduler**

Ο scheduler της cpu μπορεί να κάνει swap out ολόκληρες διεργασίες για να απελευθερώσει μνήμη . απομακρύνονται όλες οι διεργασίες και οι ιδιωτικές σελίδες από τη μνήμη ,και σημαίες καθορίζονται στον πίνακα διεργασιών για να καθορίσουν ότι αυτή η διεργασία έχει γίνει swap out. Αυτός είναι ένας φθηνός τρόπος για να διατηρηθεί μνήμη, αλλά επηρεάζει δραματικά την απόδοση των διεργασιών .

Ο scheduler μνήμης ξεκινά σε boot time και δεν κάνει τίποτα εκτός αν υπάρχει λιγότερη από την desfree μνήμη για ένα χρόνο (30 δευτερόλεπτα μέσο ορό). Σε αυτό το σημείο ξεκινά να ψάχνει για διεργασίες οπου μπορεί να διαγράψει εντελώς . ο σχεδιαστής μνήμης θα κάνει soft-swap out αν υπάρχει λίγο έλλειμμα , η hard-swap αν υπάρχει μεγαλύτερη έλλειψη μνήμης.

### **Soft Swapping**

Παρουσιάζεται όταν για 30 δευτερόλεπτα η μνήμη είναι κάτω από το desfree. Σε αυτό το σημείο ο scheduler της μνήμης θα ψάξει για μια διεργασία που είναι ανενεργοί για maxslp δευτερόλεπτα.

### **Hard Swapping**

Παρουσιάζεται όταν υπάρχουν τουλάχιστο δυο διεργασίες σε σειρά εκτέλεσης περιμένοντας τη cpu και όταν η ελεύθερη μνήμη είναι λιγότερη από τη desfree για τουλάχιστο 30 δευτερόλεπτα

Όταν ξεκινά το hard swapping η διαδικασία εύρεσης μνήμης είναι πολύ πιο επιθετική . το πρώτο βήμα είναι να κληθεί το kernel να αποδεσμεύσει όλες τις υπομονές του και τη μνήμη cache που είναι ανενεργοί ακολουθούμενη από τη σειριακό άδειασμα διεργασιών μέχρι η ελεύθερη μνήμη να φτάσει στο επιθυμητό επίπεδο..

### **Διαδιεργασιακή επικοινωνία με σωληνώσεις**

Η κλήση pipe() λαμβάνει ως παράμετρο έναν πίνακα fd δύο ακεραίων και θέτει τις τιμές τους σε περιγραφείς δύο άκρων μιας σωλήνωσης. Ο περιγραφέας fd[0] προσδιορίζει το

άκρο της σωλήνωσης από το οποίο μπορούμε να διαβάσουμε ενώ ο περιγραφέας fd[1] προσδιορίζει το άκρο της σωλήνωσης στο οποίο γράφουμε. Τα στοιχεία που γράφονται στο ένα άκρο της σωλήνωσης μπορούν να διαβαστούν από το άλλο. Επειδή οι περιγραφείς κληρονομούνται ανάμεσα σε διεργασίες η σωλήνωση μπορεί να χρησιμοποιηθεί για διαδιεργασιακή επικοινωνία.

## Διαδιεργασιακή Επικοινωνία

### Interprocess Communication

Το API (application programming interface) για τα Internet Protocols  
Χαρακτηριστικά της διαδιεργασιακής επικοινωνίας

□□ Για την επίτευξη της επικοινωνίας δυο διεργασιών μέσω μηνυμάτων αυτές θα πρέπει να υποστηρίζουν τις λειτουργίες επικοινωνίας μηνυμάτων send και receive. Μια διεργασία στέλνει (send) μήνυμα σε ένα προορισμό και μια διεργασία στον προορισμό λαμβάνει (receive) το μήνυμα. □ Σύγχρονη και ασύγχρονη επικοινωνία (synchronous and asynchronous communication).

Σε κάθε προορισμό υπάρχει μια ουρά στην οποία αυτός που στέλνει μηνύματα, προς τον αντίστοιχο προορισμό, τα εισάγει και μια διεργασία στον προορισμό λαμβάνει τα μηνύματα από την ουρά  
Synchronous communication: οι λειτουργίες send and receive είναι blocking

### Asynchronous communication:

η λειτουργία send είναι NON-blocking ενώ η receive μπορεί να είναι είτε blocking είτε NON-blocking.

Για την επίτευξη της επικοινωνίας δυο διεργασιών μέσω μηνυμάτων αυτές θα πρέπει να υποστηρίζουν τις λειτουργίες επικοινωνίας μηνυμάτων send και receive. Μια διεργασία στέλνει (send) μήνυμα σε ένα προορισμό και μια διεργασία στον προορισμό λαμβάνει (receive) το μήνυμα.

□□ Σύγχρονη και ασύγχρονη επικοινωνία (synchronous and asynchronous communication).

Σε κάθε προορισμό υπάρχει μια ουρά στην οποία αυτός που στέλνει μηνύματα, προς τον αντίστοιχο προορισμό, τα εισάγει και μια διεργασία στον προορισμό λαμβάνει τα μηνύματα από την ουρά

Synchronous communication: οι λειτουργίες send and receive είναι blocking

Asynchronous communication: η λειτουργία send είναι NON-blocking ενώ η receive



μπορεί να είναι είτε blocking είτε NON-blocking. Προορισμοί μηνύματος (message destinations) στο Internet Protocol (IP) τα μηνύματα στέλνονται στον προορισμό ο οποίος προσδιορίζεται από ένα ζεύγος της μορφής (internet address, local port). local port είναι ο προορισμός ενός μηνύματος μέσα σε ένα υπολογιστή και καθορίζεται από ένα

ακέραιο αριθμό. Ένα port έχει ένα (ή περισσότερους, στην περίπτωση του multicast) receivers και μπορεί να έχει πολλούς Senders. Μια διεργασία μπορεί να λαμβάνει από πολλά διαφορετικά Ports. Οποία διεργασία ξέρει τη διεύθυνση του υπολογιστή και τον αριθμό

του port όπου λαμβάνει μια άλλη διεργασία, τότε μπορεί να επικοινωνήσει μαζί της.

Οι Servers συνήθως δημοσιοποιούν τα ports numbers του στους Clients

Η Αξιοπιστία (Reliability) μπορεί να οριστεί σε σχέση με την εγκυρότητα (validity) ακεραιότητα (integrity). Ενώσω ισχύει η ιδιότητα της εγκυρότητας μπορούμε να καλούμε μια point-to-point message service αξιόπιστη ακόμα και αν ένας «εύλογος» αριθμός μηνυμάτων διακόπηκαν ή χάθηκαν.

καλούμε μια point-to-point message service Αναξιόπιστη (unreliable) αν τα μηνύματα δεν παραδίνονται εγγυημένα. Αναφορικά με την Ακεραιότητα των μηνυμάτων αυτά θα πρέπει να παραδίνονται αμετάβλητα και χωρίς επαναλήψεις (duplications) Σειρά (ordering). Σε μερικές εφαρμογές τα μηνύματα

θα πρέπει να παραδίνονται με την σειρά που έχουν αποσταλεί από τον αποστολέα.

Παράδοση των μηνυμάτων σε διαφορετική σειρά από αυτή του αποστολέα θεωρείται ΑΠΟΤΥΧΙΑ της εφαρμογής αυτής. Τα πρωτόκολλα επικοινωνίας UDP και TCP κάνουν χρήση socket abstraction η οποία παρέχει ένα τελικό σημείο για την επικοινωνία μεταξύ δύο διεργασιών.

□ □ Η διαδιεργασιακή επικοινωνία επιτυγχάνεται με την μετάδοση ενός μηνύματος ανάμεσα σε ένα socket της μιας διεργασίας και ένα socket της άλλης διεργασίας. Για να λάβει μια διεργασία ένα μήνυμα το socket της ΠΡΕΠΕΙ να είναι δεσμευμένο σε ένα local port και μια από τις Internet addresses (IA) του

υπολογιστή στον οποίο εκτελείται. Τα μηνύματα που αποστέλλονται σε συγκεκριμένη IA και συγκεκριμένο Port μπορούν να ληφθούν από τη διεργασία που το socket της είναι δεσμευμένο σε

αυτή την IA και post number. Κάθε υπολογιστής έχει πολλά post numbers (216). Μια διεργασία μπορεί να λαμβάνει από ΠΟΛΛΑ ports, αλλά \_EN μπορεί να μοιράζεται ports με άλλες διεργασίες, στον ίδιο υπολογιστή. Εξαίρεση αποτελεί η περίπτωση όπου οι διεργασίες χρησιμοποιούν IP multicast. Οποιοσδήποτε αριθμός διεργασιών μπορούν να στέλνουν μηνύματα σε ένα port (κάποιου υπολογιστή). Κάθε socket σχετίζεται με ένα συγκεκριμένο πρωτόκολλο, είτε το UDP ή TCP

Το API του TCP protocol παρέχει μια αφηρημένη μορφή ενός stream από bytes στο οποίο μπορούν να γραφτούν δεδομένα και από το οποίο μπορούν να διαβαστούν δεδομένα

□ □ Αυτή η αφηρημένη μορφή stream κρύβει τα ακόλουθα χαρακτηριστικά ενός δικτύου

Μέγεθος μηνύματος: η υλοποίηση του TCP stream αποφασίζει πόσα δεδομένα να συλλέξει πριν το στείλει το μήνυμα σε ένα ή περισσότερα πακέτα

Χαμένα μηνύματα: το TCP χρησιμοποιεί ένα acknowledgement σχήμα. Έλεγχος ροής: το TCP προσπαθεί να «ταιριάζει» την ταχύτητα της διαδικασίας που γράφει και αυτής που διαβάζει σε ένα stream

Σειρά μηνυμάτων και duplications: χρήση ταυτότητας μηνύματος (message identifier)

Προορισμός μηνύματος: το TCP εγκαθιδρύει σύνδεση μεταξύ των διαδικασιών που θα επικοινωνήσουν μέσω του stream. Χρήση connect request και accept request από τους client και server αντίστοιχα.

Το API για stream communication υποθέτει ότι, όταν για ένα ζεύγος διεργασιών έχει εγκαθιδρυθεί σύνδεση επικοινωνίας, μια από αυτές παίζει το ρόλο του πελάτη (client) και η άλλη το ρόλο του εξυπηρέτη (server)

□□ Ρόλος του client είναι να δημιουργήσει ένα stream socket δεσμευμένο σε οποιοδήποτε

(ελεύθερο) port κάνει ένα connect request ζητώντας σύνδεση με τον server στο server port

□□ Ρόλος του server είναι να: δημιουργήσει ένα listening socket δεσμευμένο σε ένα δεδομένο server

port περιμένοντας από clients requests για σύνδεση το listening socket διατηρεί μια ουρά από εισερχόμενες αιτήσεις όταν γίνει δεκτή (accept) μια αίτηση ενός client ένα δημιουργείται ένα

NEO stream socket για τον server για να επικοινωνεί με τον συγκεκριμένο client.

□ Στο μεταξύ το δικό του socket το χρησιμοποιεί όπως και πριν για να ακούει για τυχών νέες αιτήσεις από άλλους (ή τον ίδιο client)

Για την επικοινωνία δυο διεργασιών χρειάζεται ένα ζεύγος από sockets (όπως είπαμε πριν) και ένα

ζεύγος από streams, ένα για είσοδο (input stream) και ένα για έξοδο (output stream).

□ Μια διεργασία στέλνει πληροφορία στην άλλη γράφοντας στο output stream της και η άλλη

διεργασία παίρνει την πληροφορία διαβάζοντας από το input stream της.

□ Τι θα συμβεί όταν μια εφαρμογή κλείσει ('close') a socket;

TCP streams χρησιμοποιούν Checksums για ανίχνευση και απόρριψη corrupt packets

Sequence number για ανίχνευση και απόρριψη διπλών πακέτων Για την ιδιότητα της αξιοπιστίας του TCP

□ Σχετικά με την ιδιότητα της εγκυρότητας του TCP κάνει χρήση

Timeouts retransmissions=> Τα μηνύματα παραδίνονται εγγυημένα ακόμα και αν κάποιο από αυτά χαθεί.

Όταν μια σύνδεση ΔΙΑΚΟΠΕΙ ή χαλάσει, τότε η διεργασία που τη χρησιμοποιεί θα ενημερωθεί για αυτή την εξέλιξη. Οι επιπτώσεις που μπορεί να προκύψουν είναι οι εξής:

Η διεργασία που χρησιμοποιεί την σύνδεση ΔΕΝ καταλαβαίνει αν η διακοπή προέκυψε από βλάβη στο

δίκτυο ή από βλάβη στην άλλη διεργασία με την οποία επικοινωνούσε. Οι διεργασίες που ήταν σε επικοινωνία δεν μπορούν να ξέρουν αν τα μηνύματα που έστειλαν προς το τέλος λήφθηκαν από την άλλη διεργασία ή όχι. Πως αναπαριστάται η πληροφορία που θέλουμε να αποστείλουμε μέσω ενός μηνύματος; Η πληροφορία αυτή μετατρέπεται σε μια σειρά από bytes

(σε τι μορφή θα μπορούσε να είναι πριν την ενσωματώσουμε σε ένα μήνυμα; Ο τρόπος αποθήκευσης των βασικών τύπων δεδομένων είναι ο ίδιος σε όλους τους υπολογιστές; η απάντηση είναι ΟΧΙ. Πως θα μπορούσαμε να αντιμετωπίσουμε αυτό το πρόβλημα; Μέθοδοι μπορούν να χρησιμοποιηθούν έτσι ώστε να είναι δυνατή η ανταλλαγή δεδομένων (μεταξύ δικτυομένων) σε δυαδική μορφή

Οι τιμές των δεδομένων μετατρέπονται σε μια συμφωνημένη εξωτερική μορφή πριν την μεταφορά τους και όταν παραληφθούν μετατρέπονται κατάλληλα με βάση τα χαρακτηριστικά του συστήματος του παραλήπτη. Υπάρχει περίπτωση αυτή η διαδικασία να παραλειφθεί;

Οι τιμές αποστέλλονται με την μορφή που τις αναγνωρίζει ο αποστολέας μαζί με πληροφορίες για τη μορφή τους και ο παραλήπτης έχει την ευθύνη να τις μετατρέψει σε κατάλληλη για αυτόν μορφή

Προφανώς τα δεδομένα ΔΕΝ μεταβάλλονται κατά την μεταφορά τους

Για την υποστήριξη RPC και RMI πρέπει αν βρεθεί τρόπος αναπαράστασης οποιουδήποτε τύπου δεδομένων. Γιατί χρειάζεται αυτό;

Οι παράμετροι εισόδου των διαδικασιών/μεθόδων μπορεί να είναι οποιοδήποτε τύπου, όπως και οι τιμές εξόδου επίσης.

Μέθοδοι μπορούν να χρησιμοποιηθούν έτσι ώστε να είναι δυνατή η ανταλλαγή δεδομένων (μεταξύ δικτυομένων) σε δυαδική μορφή. Οι τιμές των δεδομένων μετατρέπονται σε μια συμφωνημένη εξωτερική μορφή

πριν την μεταφορά τους και όταν παραληφθούν μετατρέπονται κατάλληλα με βάση τα χαρακτηριστικά του συστήματος του παραλήπτη. Υπάρχει περίπτωση αυτή η διαδικασία να παραλειφθεί; Οι τιμές αποστέλλονται με την μορφή που τις αναγνωρίζει ο αποστολέας μαζί με πληροφορίες για τη μορφή τους και ο παραλήπτης έχει την ευθύνη να τις μετατρέψει σε κατάλληλη για αυτόν μορφή

\_ Προφανώς τα δεδομένα ΔΕΝ μεταβάλλονται κατά την μεταφορά τους

\_ Για την υποστήριξη RPC και RMI πρέπει αν βρεθεί τρόπος αναπαράστασης οποιουδήποτε τύπου δεδομένων. Γιατί χρειάζεται αυτό;

Οι παράμετροι εισόδου των διαδικασιών/μεθόδων μπορεί να είναι οποιοδήποτε τύπου, όπως και οι τιμές εξόδου επίσης.

Μαζί με την θεωρητικό μέρος της εργασίας στέλνουμε και 2 παραδείγματα σε μορφή rar όπου μια διεργασία η οποία στέλνει ένα μήνυμα σε μια

άλλη διεργασία και θα τα αποθηκεύει σε ένα αρχείο μηνυμάτων.

### 1<sup>η</sup> περίπτωση pipe.c

Εδώ έχουμε ένα parent όπου φτιαχνει child. Στην συνέχεια το child διαβάζει ένα μήνυμα και με pipe το στέλνει πίσω στο parent όπου αποθηκεύεται το μήνυμα στο αρχείο και εν συνεχεία αποδεσμεύει το child (καταστραφεί) έτσι ώστε να δημιουργήσει ένα νέο child όπου θα επαναληφτεί η ίδια διαδικασία

### 2<sup>η</sup> περίπτωση Queues

Εδώ έχουμε μια process1 όπου το μήνυμα μπαίνει σε μια ουρά με την εντολή `msend` , και παραλαμβάνετε από την process2 με την εντολή `mrecv` . το μήνυμα γράφετε σε ένα αρχείο και περιμένει το process1 να στείλει άλλο μήνυμα.

Και στις 2 περιπτώσεις έχουν κατασκευαστεί σε C  
Και λειτουργουν σε solaris