05 Κατανεμημένα Συστήματα

Επικοινωνία Αίτησης – Απάντησης (Μέρος Α: Request – Reply)

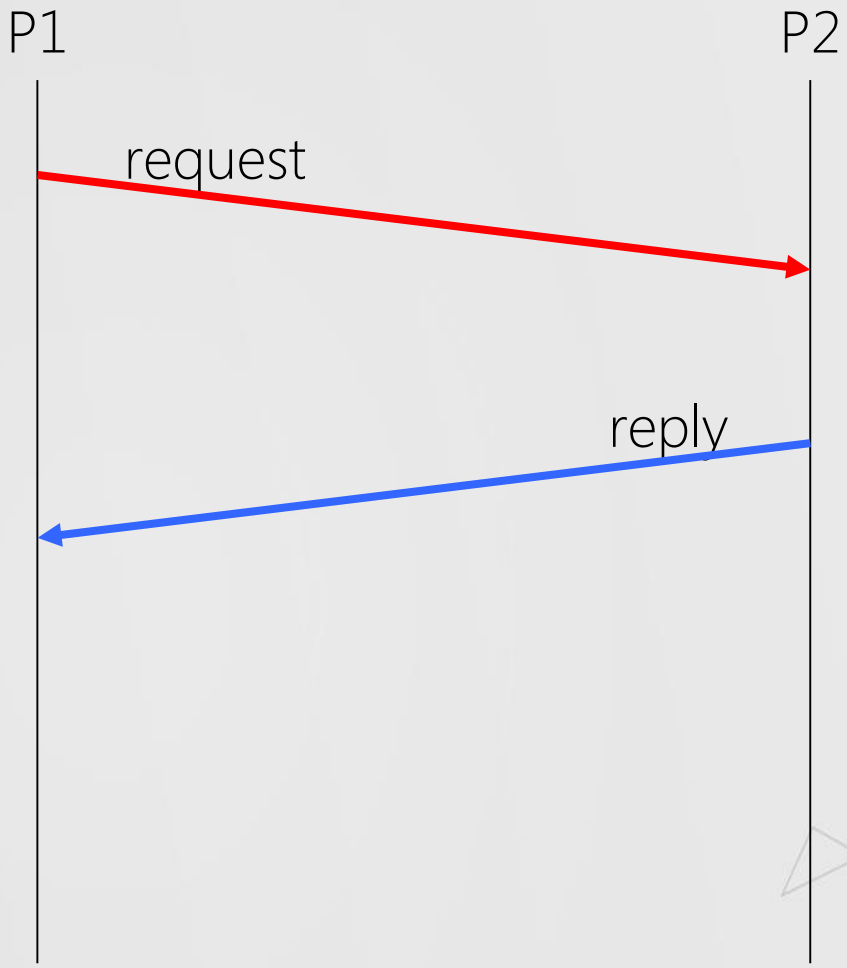
01

ΕΠΙΚΟΙΝΩΝΙΑ
REQUEST – REPLY

Σχήμα request-reply

- Ο αποστολέας στέλνει μια αίτηση (request).
- Ο παραλήπτης επεξεργάζεται την αίτηση, και στην συνέχεια στέλνει πίσω μια απάντηση (reply/response).
  - τα μηνύματα της αίτησης/απάντησης μπορεί να είναι μεγάλα οπότε χρειάζεται κατάλληλο fragmentation & reassembly.
- Το πλέον διαδεδομένο σχήμα επικοινωνίας στά κατανεμημένα συστήματα.
  - όπως άλλωστε και στα ανθρώπινα συστήματα.
- Τα περισσότερα (κατανεμημένα) συστήματα μπορεί να δομηθούν (με σχετικά φυσικό / εύκολο τρόπο) με βάση το σχήμα επικοινωνίας request – reply.

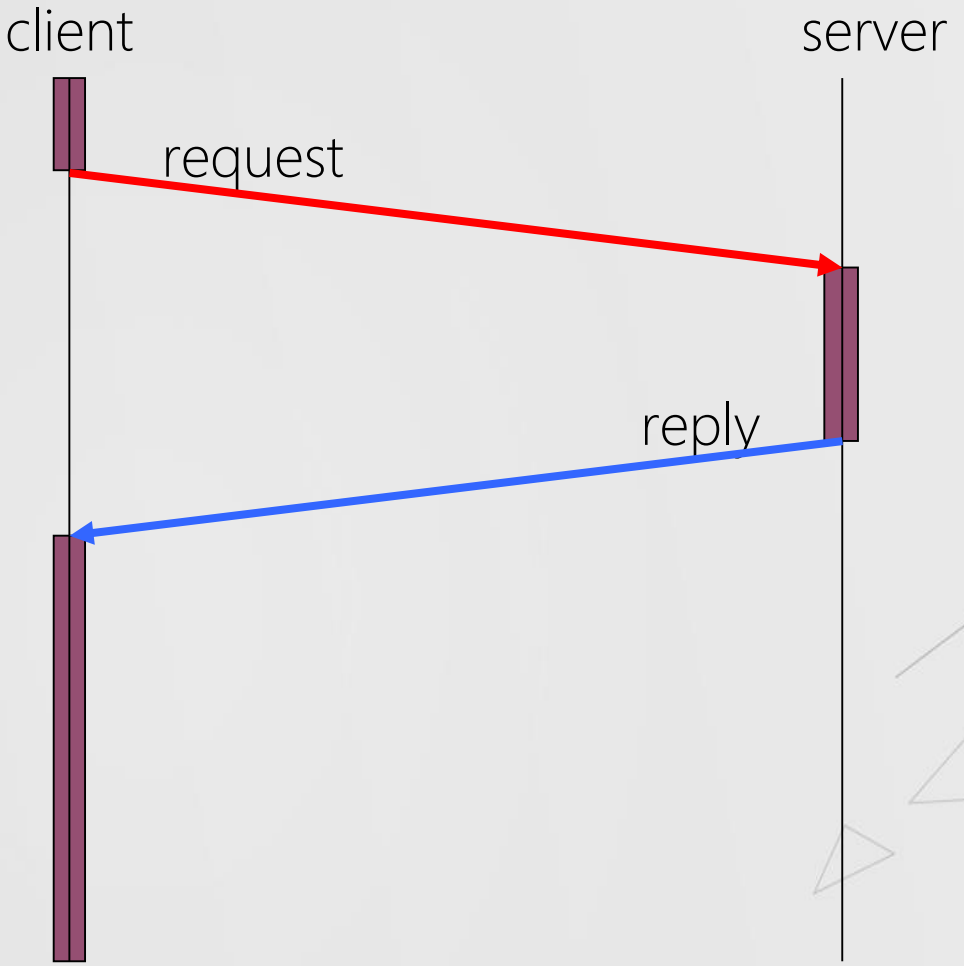
Σχεδιάγραμμα request – reply



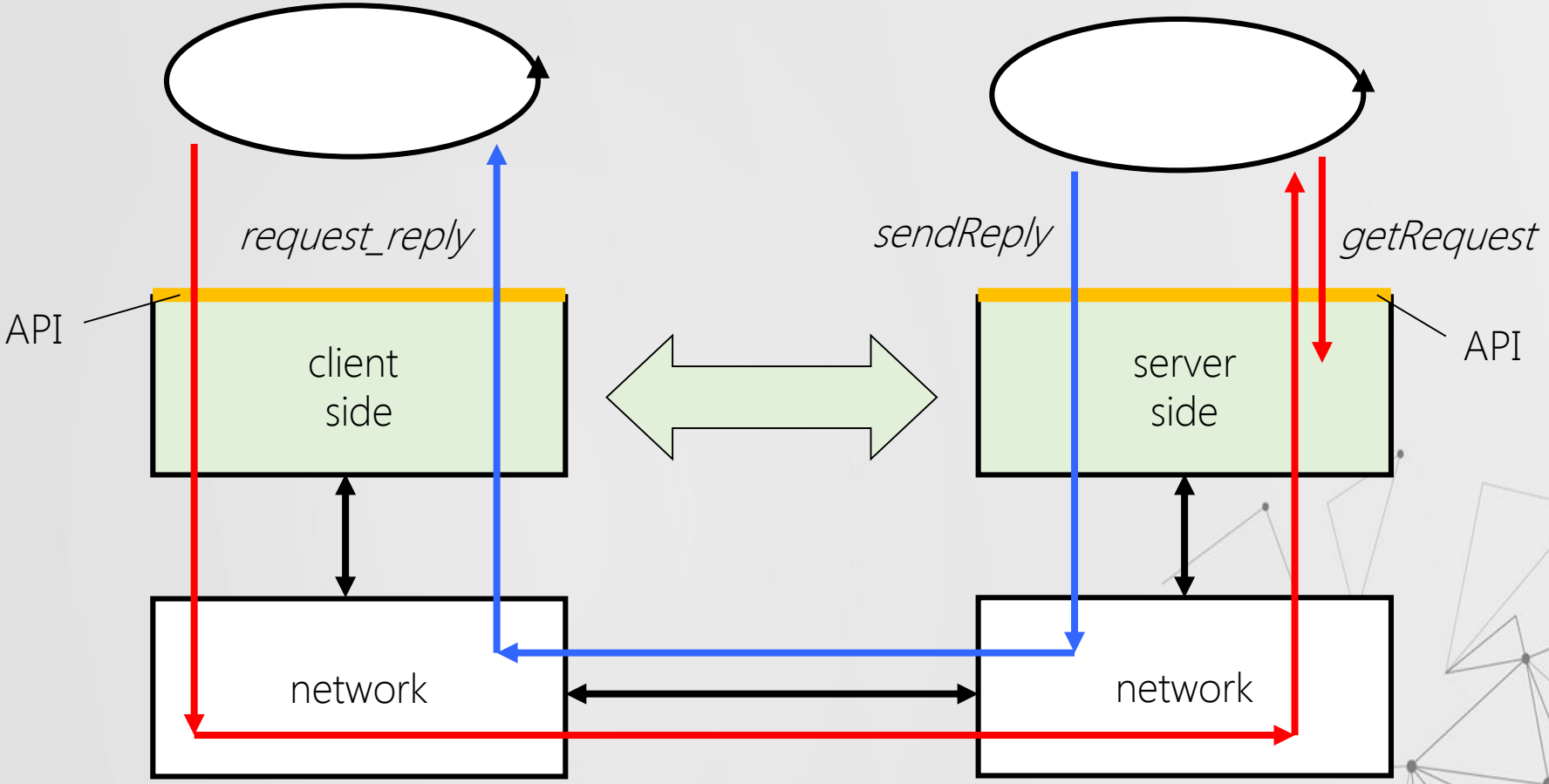
Σχήμα επικοινωνίας client-server

- Ακολουθεί το σχήμα επικοινωνίας request - reply.
  - με κάποιες εξειδικεύσεις / επεκτάσεις.
- Η διεργασία που στέλνει την αίτηση περιμένει (μπλοκάρεται) μέχρι να λάβει την απάντηση.
- Η διεργασία που δέχεται την αίτηση παραμένει ανενεργή όσο δεν λαμβάνει/επεξεργάζεται αιτήσεις.
  - υπάρχει για να δέχεται και να επεξεργάζεται τις αιτήσεις των πελατών (υλοποιεί μια απομακρυσμένη υπηρεσία).
- **Πελάτης:** η διεργασία που στέλνει την αίτηση.
- **Εξυπηρετητής:** η διεργασία που δέχεται και επεξεργάζεται τις αιτήσεις των πελατών.

Σχεδιάγραμμα client – server (1)



Σχεδιάγραμμα client – server (2)



Ενδεικτικό API

Client:

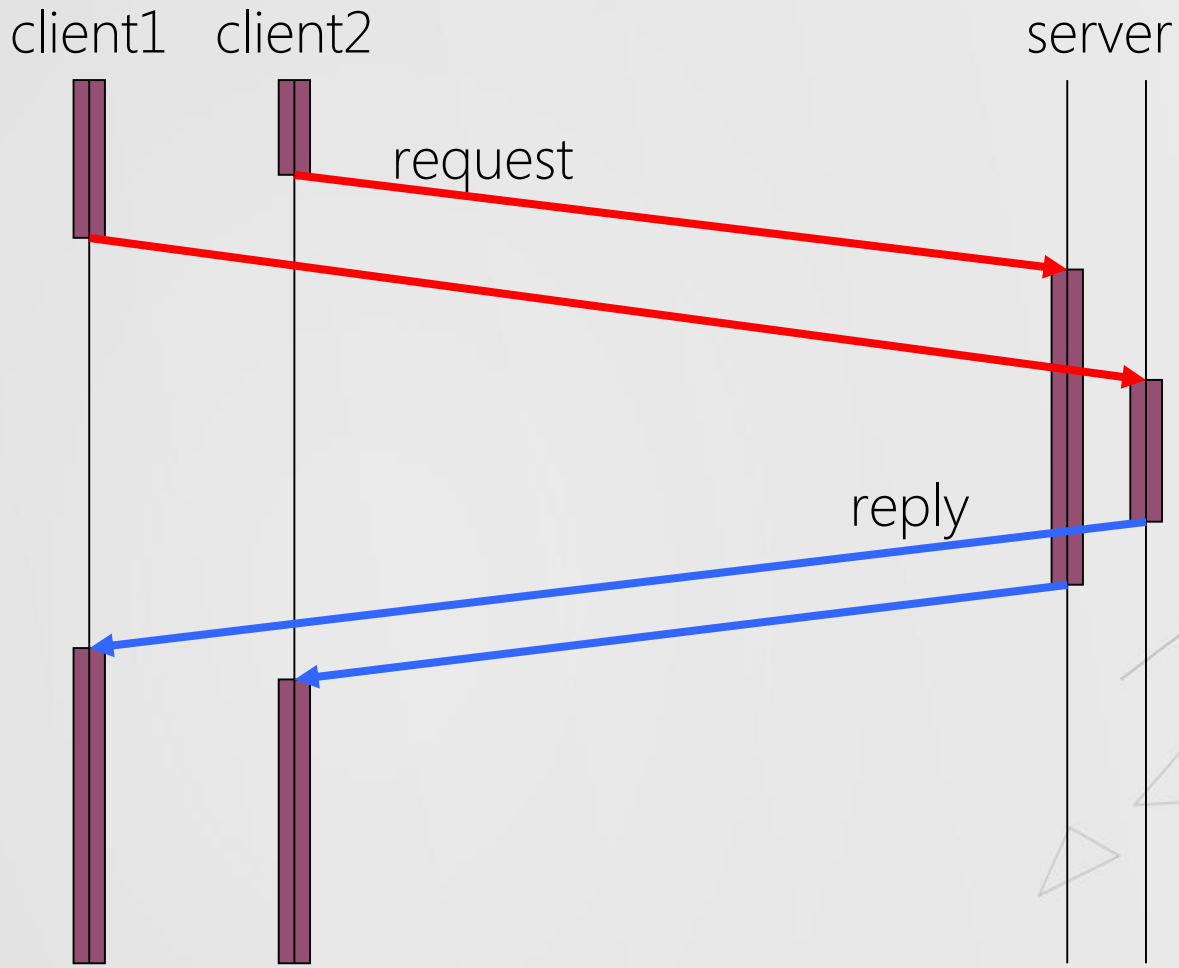
```
int request_reply(char *request, int requestlen,
                  char **reply, int *replylen);
```

Server:

```
ReqID getRequest(char **request, int *requestlen);
void sendReply(ReqID id, char *reply, int replylen);
```

Γιατί να υπάρχει ρητή αντιστοίχηση απάντησης σε αίτηση;

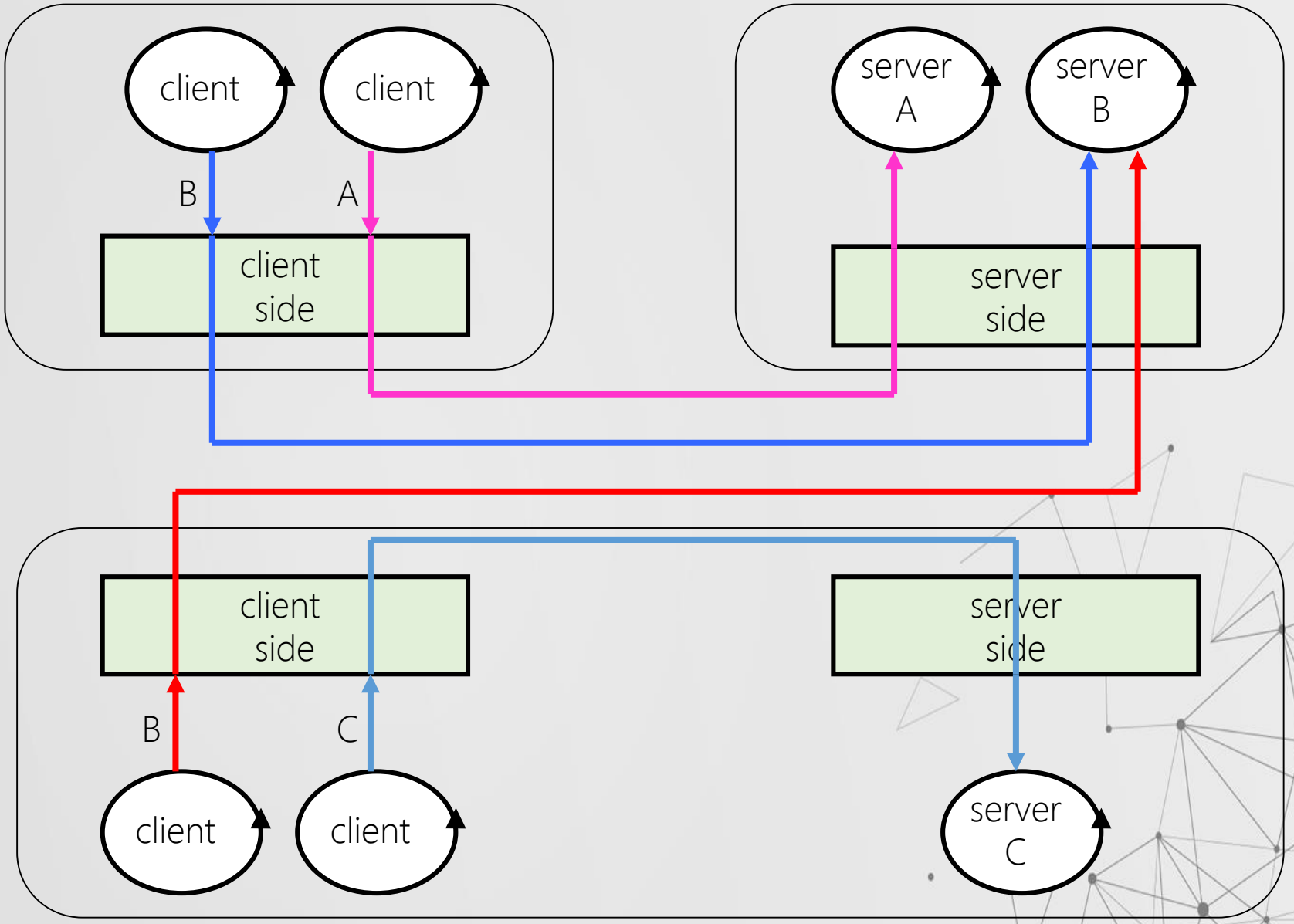
Σχεδιάγραμμα client – server (3)



Πολυπλεξία & ανακάλυψη εξυπηρετητών

- Συνύπαρξη πολλών υπηρεσιών στο ίδιο μηχάνημα.
- Χρειάζεται διαχωρισμός ανάμεσα στους server, μέσα από κατάλληλα (συμφωνημένα) αναγνωριστικά.
- Ο εξυπηρετητής πρέπει να δηλώνει ρητά ποια υπηρεσία υποστηρίζει/υλοποιεί.
- Ο πελάτης πρέπει να ανακαλύπτει τον «σωστό» εξυπηρετητή, για την υπηρεσία που επιθυμεί.
- Χρησιμοποιούνται οι γνωστές λύσεις ανακάλυψης.
- Προσυμφωνημένα στοιχεία επικοινωνίας ή ανακάλυψη τους με εκπομπή ή μέσω μιας υπηρεσίας καταλόγου.

Σχεδιάγραμμα client – server (4)



Ενδεικτικό API

Client:

```
int request_reply(SrvcID svc, char *request,
      int requestlen, char **reply, int *replylen);
```

Server:

```
int register(SrvcID svc);
void unregister(SrvcID svc);
ReqID getRequest(SrvcID svc, char **request,
int *requestlen)
void sendReply(ReqID id, char *reply, int replylen)
```

Γιατί να υπάρχει ρητή εγγραφή για το είδος της προσφερόμενης υπηρεσίας;

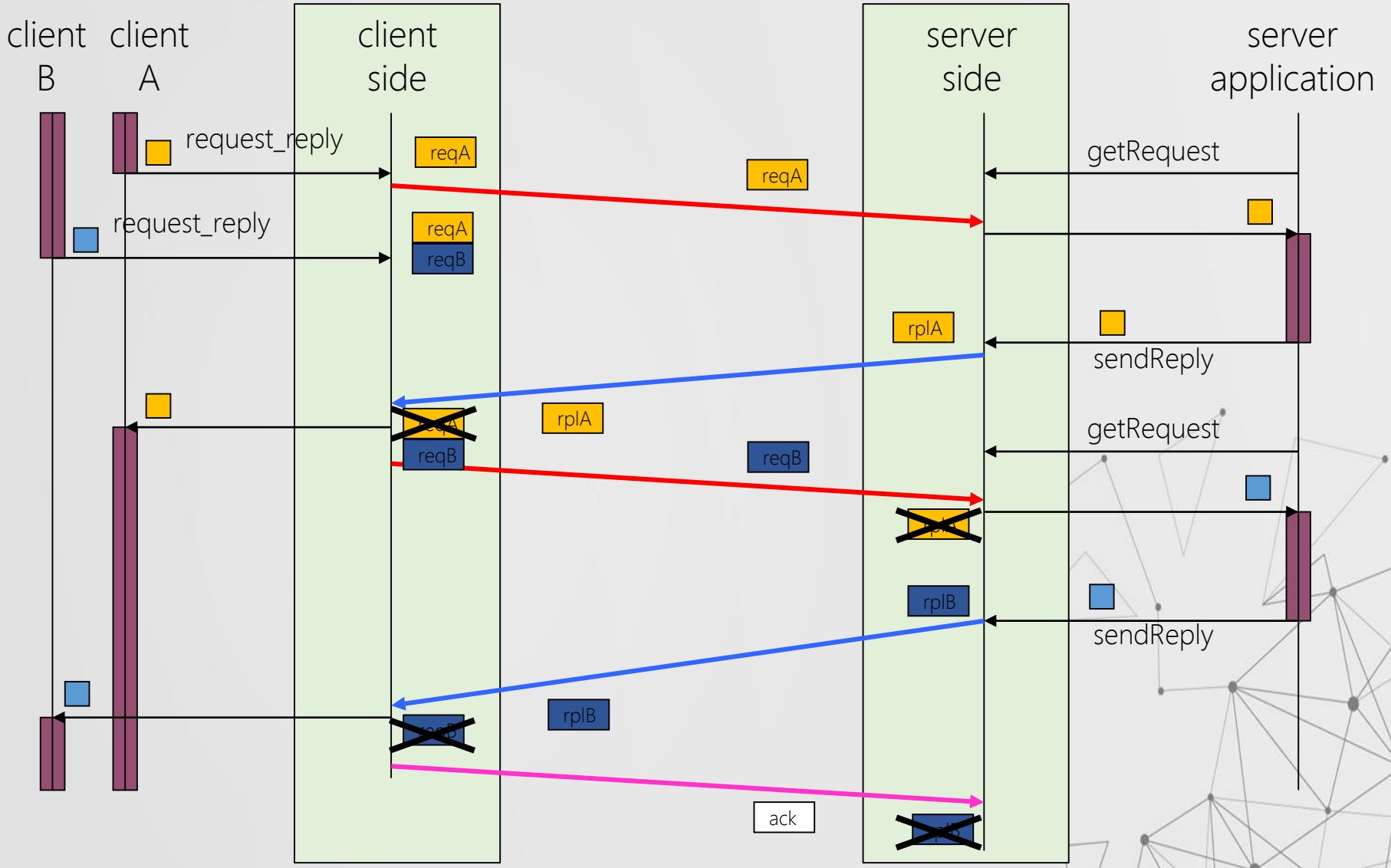
Πρωτόκολλο μετάδοσης αιτήσεων/απαντήσεων

- Πρέπει να οριστεί η αλληλεπίδραση ανάμεσα στην πλευρά του πελάτη και την πλευρά του εξυπηρετητή.
- Υπάρχουν 2 βασικές διαστάσεις.
- **Επιβεβαίωση** αίτησης του πελάτη και απάντησης του εξυπηρετητή.
- **Χειρισμός βλαβών** εξυπηρετητή και πελάτη.

Σειριακή αποστολή αιτήσεων (1)

- Η πλευρά του πελάτη στέλνει την επόμενη αίτηση αφού λάβει απάντηση για τη προηγούμενη.
  - αν δεν λάβει απάντηση, ξαναστέλνει την αίτηση.
- Η απάντηση του εξυπηρετητή ερμηνεύεται ως επιβεβαίωση της προηγούμενης αίτησης.
- Η πλευρά του εξυπηρετητή ερμηνεύει την επόμενη αίτηση ως επιβεβαίωση της τελευταίας απάντησης.
  - αν δεν υπάρχει επόμενη αίτηση, χρειάζεται ειδική επιβεβαίωση.
- Σειριακή επεξεργασία αιτήσεων στον εξυπηρετητή.
- Άσκοπες αποστολές της ίδιας αίτησης (αν αργεί να έρθει η απάντηση του εξυπηρετητή).

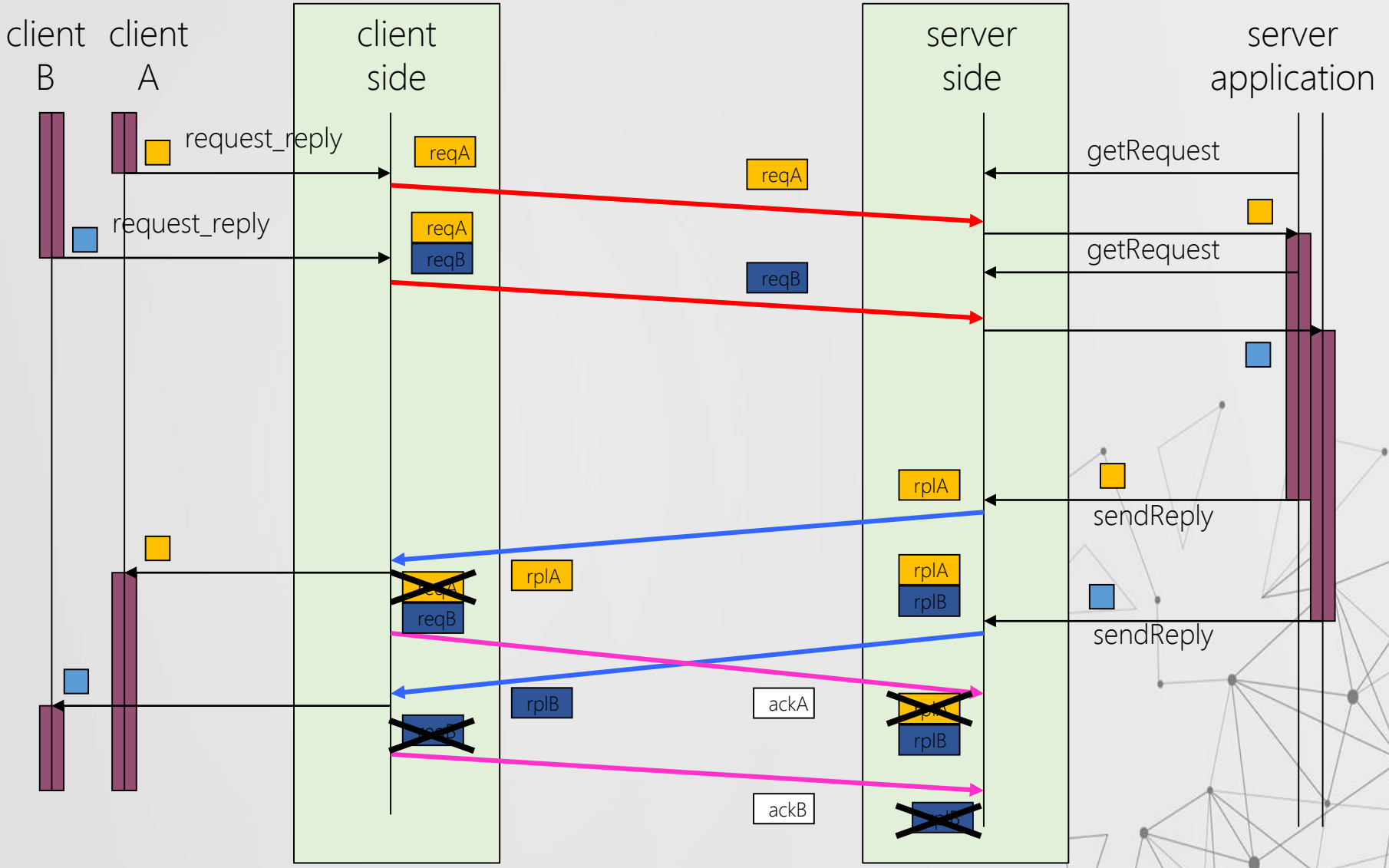
Σειριακή αποστολή αιτήσεων (2)



Ταυτόχρονη αποστολή αιτήσεων (1)

- Η πλευρά του πελάτη στέλνει νέες αιτήσεις χωρίς να περιμένει απάντηση για τις προηγούμενες.
  - η επόμενη αίτηση δεν χρησιμεύει ως επιβεβαίωση.
- Η απάντηση του εξυπηρετητή ερμηνεύεται ως επιβεβαίωση για την αντίστοιχη αίτηση.
- Χρειάζεται ξεχωριστή επιβεβαίωση της απάντησης.
  - η επόμενη αίτηση δεν χρησιμεύει ως επιβεβαίωση.
- Αυξημένες απαιτήσεις αποθήκευσης στον εξυπηρετητή.
- Αυξημένος φόρτος στο δίκτυο (μαζικές επιβεβαιώσεις).
- Άσκοπες αποστολές των ίδιων αιτήσεων (αν αργούν να έρθουν οι απαντήσεις του εξυπηρετητή).

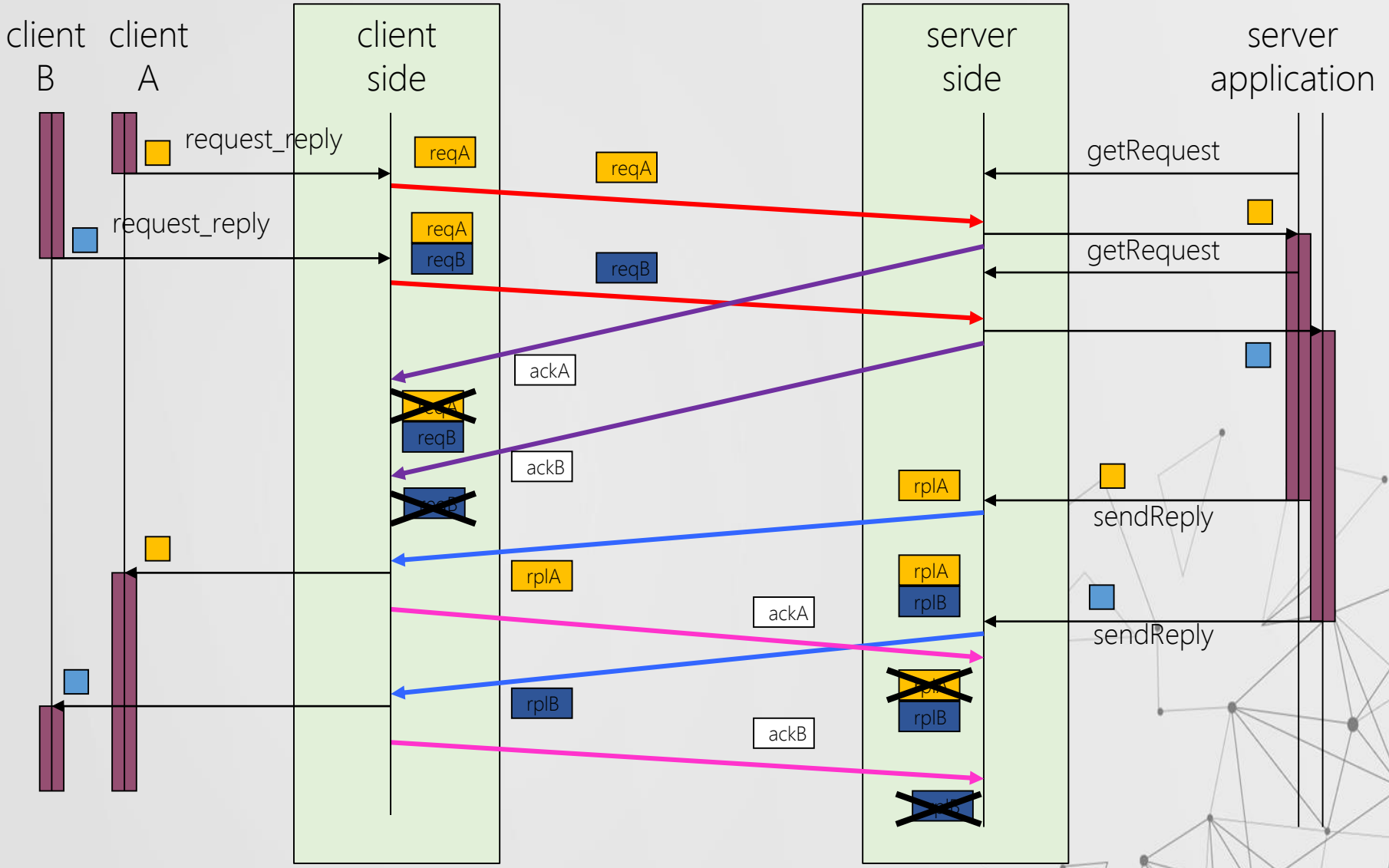
Ταυτόχρονη αποστολή αιτήσεων (2)



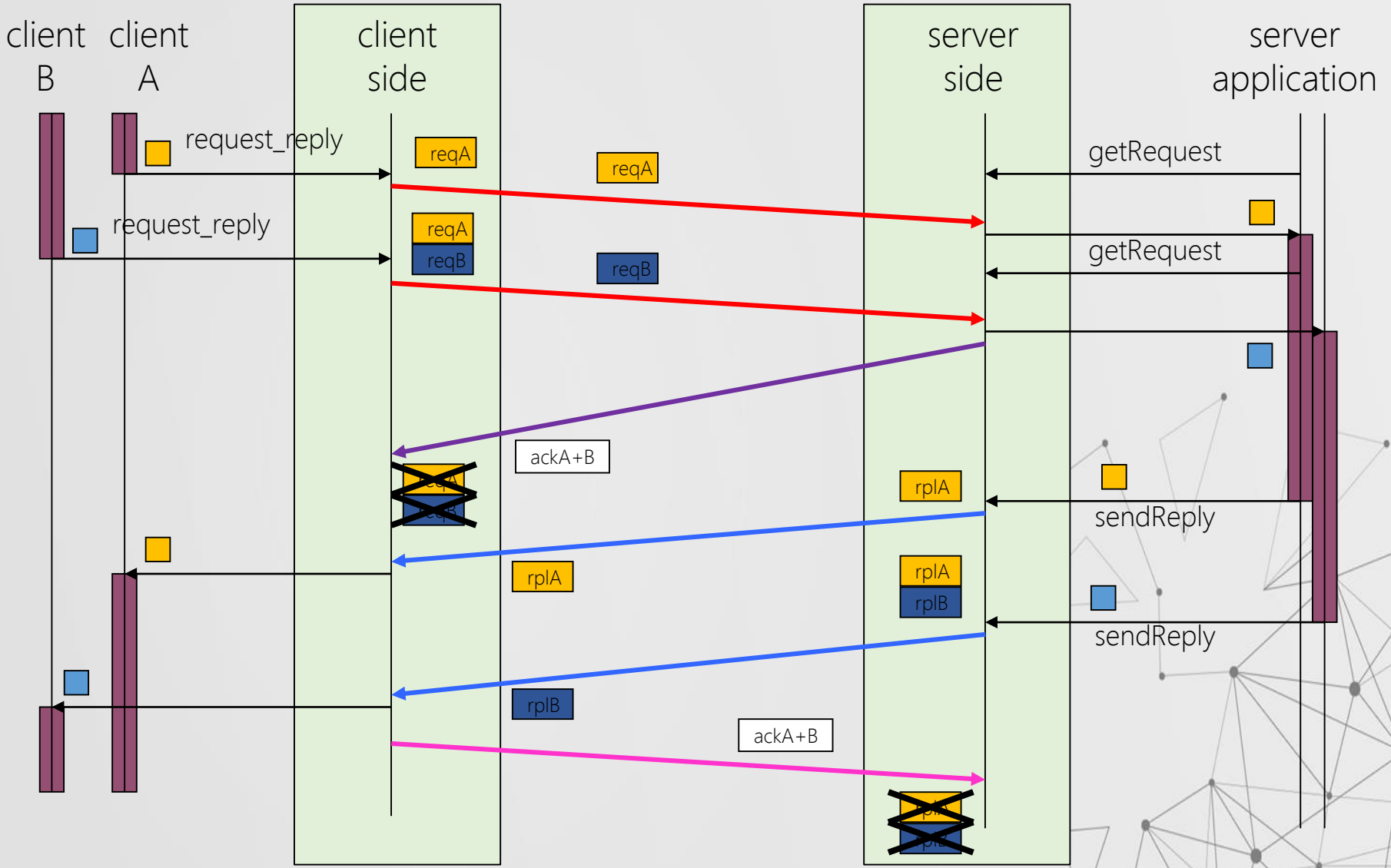
Ρητή επιβεβαίωση παραλαβής αιτήσεων (1)

- Πόσο πρέπει να περιμένει η πλευρά του πελάτη μέχρι να ξαναστείλει την αίτηση, σε περίπτωση που χάθηκε;
- Παρόμοιο πρόβλημα με αυτό του κλασικού timeout, αλλά με περισσότερους αστάθμητους παράγοντες.
  - χρόνος επεξεργασίας αίτησης, φόρτος υπηρεσίας.
- Η πλευρά του εξυπηρετητή επιβεβαιώνει την παραλαβή της αίτησης άμεσα, με ένα ξεχωριστό μήνυμα.
  - ανεξάρτητα από την απάντηση (που στέλνεται αργότερα).
- Όμως, έτσι στέλνονται ακόμα περισσότερα μηνύματα πάνω από το δίκτυο (και πάλι: μαζικές επιβεβαιώσεις).

Ρητή επιβεβαίωση παραλαβής αιτήσεων (2)



Ρητή επιβεβαίωση παραλαβής αιτήσεων (3)



Βασική υπηρεσία μεταφοράς

- TCP/IP:
- Κόστος σύνδεσης, αλλά η ίδια σύνδεση μπορεί να ξαναχρησιμοποιηθεί για συνεχόμενες ανταλλαγές.
- Εγγυημένη μεταφορά, αλλά (στην γενική περίπτωση) χρειάζονται επιβεβαιώσεις και σε πιο ψηλό επίπεδο

- UDP/IP:
- Ευέλικτα σχήματα επαναποστολής/επιβεβαίωσης.
- Ιδιαίτερα κατάλληλο αν δεν υπάρχουν απαιτήσεις πλήρους αξιοπιστίας και τα μηνύματα της εφαρμογής είναι σχετικά μικρά.

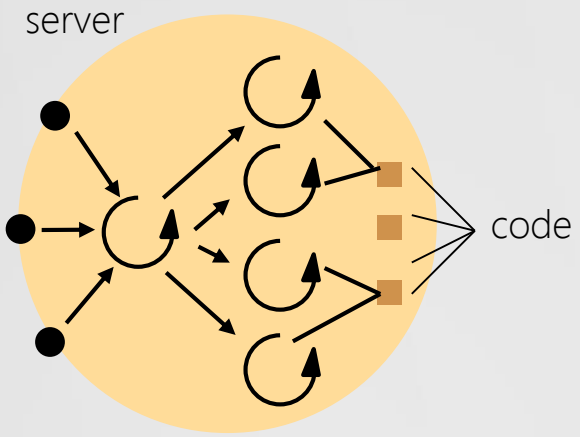
End-to-end argument

- Μια υπηρεσία επιπέδου *N* δεν μπορεί να προσφέρει σημασιολογία/εγγυήσεις πιο ψηλού επιπέδου *N+1*.
- Παράδειγμα request-reply πάνω από υπηρεσία αξιόπιστης μεταφοράς δεδομένων (π.χ. TCP/IP).
- Το γεγονός ότι η παραλαβή δεδομένων επιβεβαιώθηκε στο επίπεδο της μεταφοράς (TCP/IP), δεν επιβεβαιώνει κάτι για το παραπάνω επίπεδο (εφαρμογής).
- Π.χ., το μηχάνημα μπορεί να παρουσιάσει βλάβη αφού επιβεβαιωθεί η παραλαβή των δεδομένων στο επίπεδο μεταφοράς (TCP/IP), αλλά προτού η εφαρμογή παραλάβει τα δεδομένα.
- Στην γενικότερη περίπτωση, χρειάζεται επιβεβαίωση και στο επίπεδο εφαρμογής.

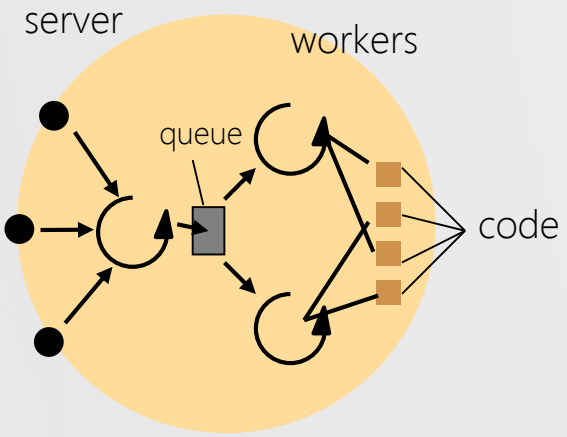
Ταυτόχρονη επεξεργασία αιτήσεων (1)

- Worker per request:
  - για κάθε αίτηση που καταφθάνει, δημιουργείται ένα νέο νήμα στο οποίο ανατίθεται η επεξεργασία της.
- Worker pool:
  - οι αιτήσεις μπαίνουν σε κοινή ουρά, από την οποία απομακρύνονται και επεξεργάζονται από νήματα εργάτες.
- Worker per connection:
  - για κάθε σύνδεση (ανά κόμβο) δημιουργείται ένα νήμα που επεξεργάζεται τις αντίστοιχες αιτήσεις.
- Worker per function group:
  - για κάθε ομάδα αιτήσεων υπάρχει ένα ξεχωριστό νήμα που επεξεργάζεται τις σχετικές αιτήσεις.
- Θέματα συγχρονισμού: «ανταγωνιστικές» κλήσεις.

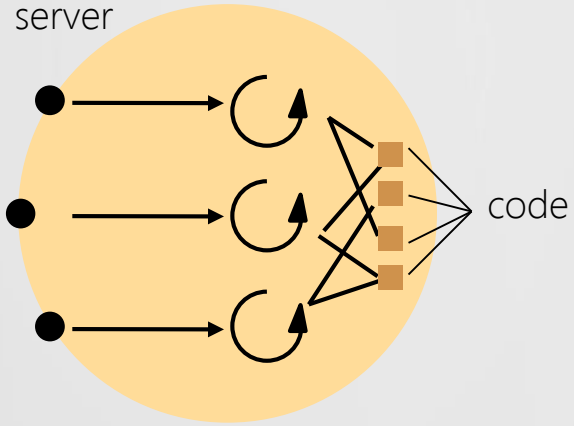
Ταυτόχρονη επεξεργασία αιτήσεων (2)



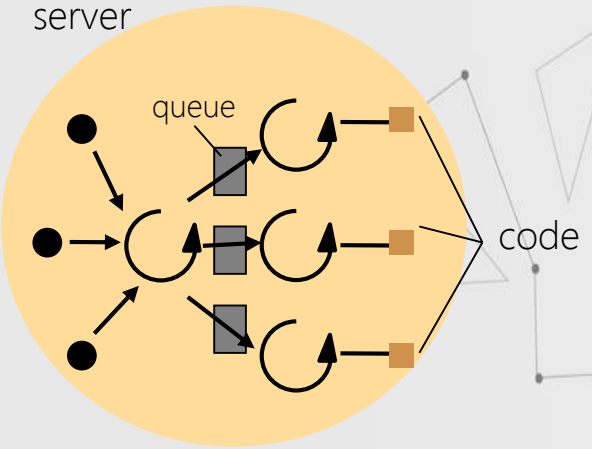
1) Worker per request



2) Worker pool



3) Worker per connection



4) Worker per function group

Ασύγχρονο σχήμα αίτησης-απάντησης

- Απεμπλοκή της διαδικασίας αποστολής της αίτησης από την διαδικασία παραλαβής της απάντησης.
- Ως αποτέλεσμα της αίτησης, ο πελάτης λαμβάνει μια απόδειξη/εισιτήριο, που μπορεί να χρησιμοποιήσει, αργότερα, για να παραλάβει το αποτέλεσμα.
- Δεν απαιτείται χρήση νημάτων σε επίπεδο εφαρμογής.
- Ούτε σε επίπεδο υπηρεσίας μεταφοράς.
- **Σημείωση:** η χρήση νημάτων σε επίπεδο εφαρμογής λύνει το πρόβλημα του μπλοκαρίσματος, αλλά δυσκολεύει την υλοποίηση και καταναλώνει πόρους

Ενδεικτικό API

Client:

έλεγχος και παραλαβή αποτελέσματος μέσω «απόδειξης»

```
ReqID request(SrvcID svc, char *request, int
requestlen);

int replyAvailable(ReqID id);

int getReply(ReqID id, char **reply, int *replylen);
```

Server:

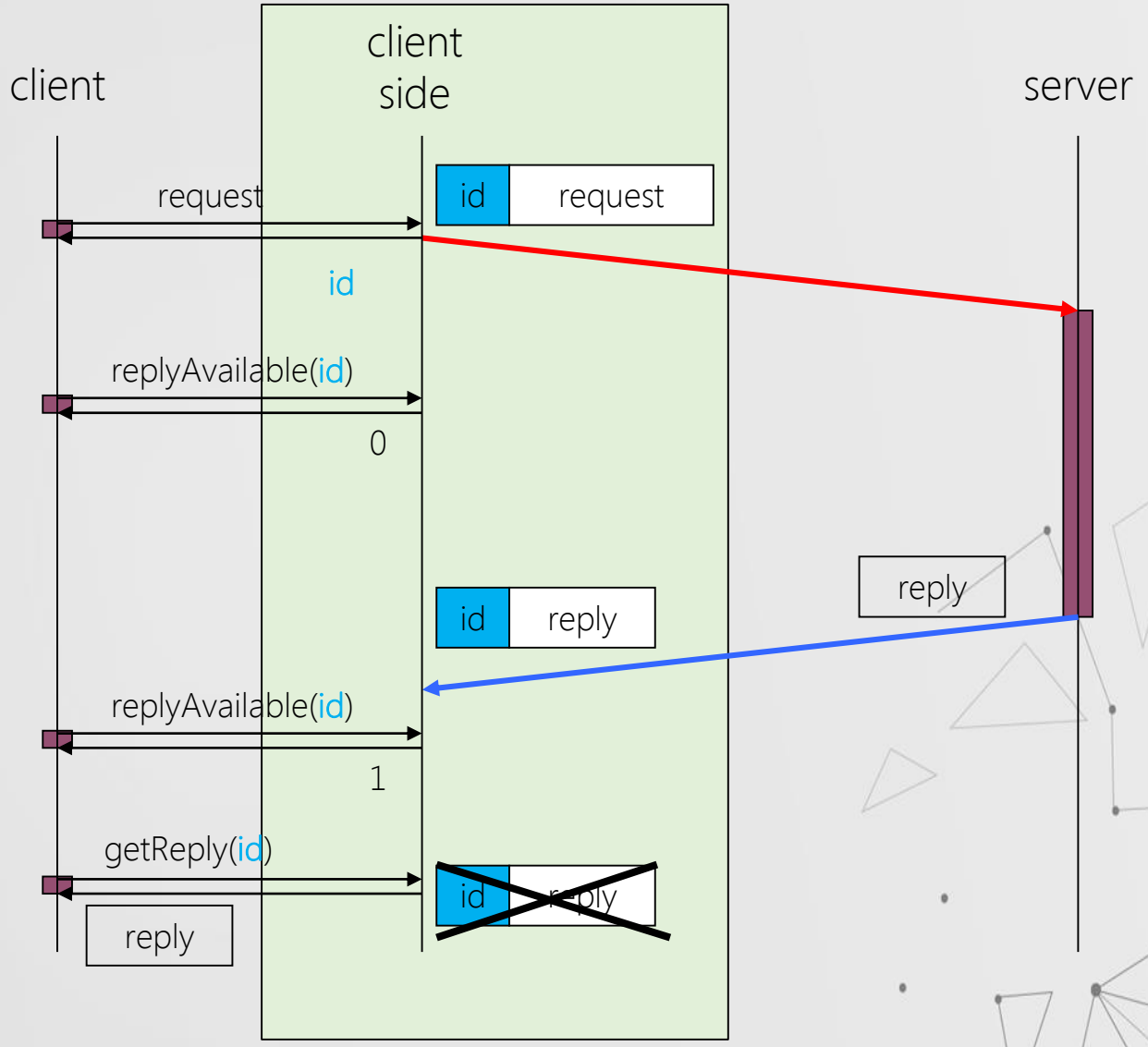
```
int register(SrvcID svc);
void unregister(SrvcID svc);

ReqID getRequest(SrvcID svc, char **request, int
*requestlen)

void sendReply(ReqID id, char *reply, int replylen)
```

παραμένει ίδιο

Ασύγχρονο σχήμα αίτησης-απάντησης



One-way requests

- Υπάρχουν αιτήσεις που δεν επιστρέφουν κάποια απάντηση σε επίπεδο εφαρμογής.
- Σε πιο χαμηλό επίπεδο, ο εξυπηρετητής πιθανώς εξακολουθεί να πρέπει να επιβεβαιώσει την ορθή επεξεργασία της αίτησης.
- Δεν υπάρχει λόγος να μπλοκάρεται ο πελάτης.
- Μπορεί να χρησιμοποιηθεί παρόμοιος μηχανισμός ασύγχρονης επικοινωνίας.
- Σε κάποιες περιπτώσεις, μπορεί να μην απαιτείται επιβεβαίωση ούτε σε επίπεδο μεταφοράς.

One-way requests