ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ

Φροντιστηριακή Διάλεξη | Άννα Κεφάλα

ΑΠΛΗ ΠΡΟΣΟΜΟΙΩΣΗ ΜΕΤΑΔΟΣΗΣ ΔΕΔΟΜΕΝΩΝ

Προσομοίωση, Δίκτυο, Μετάδοση δεδομένων, Πακέτα, Δρομολογητές

ΠΡΟΣΟΜΟΙΩΣΗ

- **Προσομοίωση** (Simulation): ένα ισχυρό εργαλείο για τη μελέτη πολύπλοκων συστημάτων.
- Ανάπτυξη **μοντέλου** ενός πολύπλοκου συστήματος & πειραματική χρήση του μοντέλου → **παρατήρηση αποτελεσμάτων**
- **Μοντέλο**: μία (*αφαιρετική*) αναλογία του πραγματικού συστήματος που βοηθάει να κατανοήσουμε τη λειτουργία του, ένα σύνολο υποθέσεων, δεδομένων και συμπερασμάτων ως μαθηματική περιγραφή μιας οντότητας ή μιας κατάστασης.
  Αναπαράσταση των οντοτήτων/αντικειμένων του συστήματος και των κανόνων που διέπουν την αλληλεπίδραση των οντοτήτων/αντικειμένων.

ΠΡΟΣΟΜΟΙΩΣΗ (2)

- **Ανάπτυξη Μοντέλου**: προσδιορισμός μικρού συνόλου χαρακτηριστικών και ιδιοτήτων, αρκετών ώστε να περιγραφεί η συμπεριφορά του συστήματος υπό μελέτη.

ΔΙΚΤΥΟ | NETWORK

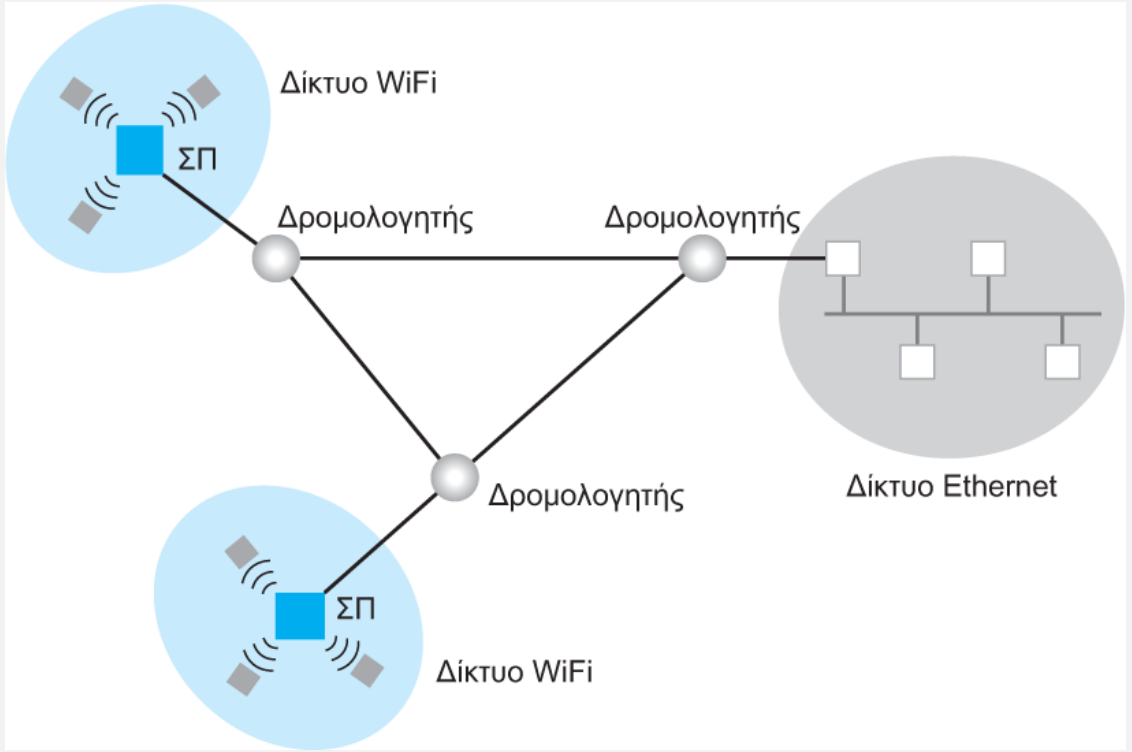
- **Δίκτυο**: διασυνδεδεμένο σύστημα υπολογιστών & άλλων συσκευών (hosts)
- **Internet** (Διαδίκτυο): διασυνδεδεμένα δίκτυα
- Κάθε host ταυτοποιείται στο Internet με μοναδική **IP διεύθυνση**

(IPv4) 32-bits αριθμός, μορφή dot-decimal (δεκαδική με τελείες)

π.χ. 195.251.255.142

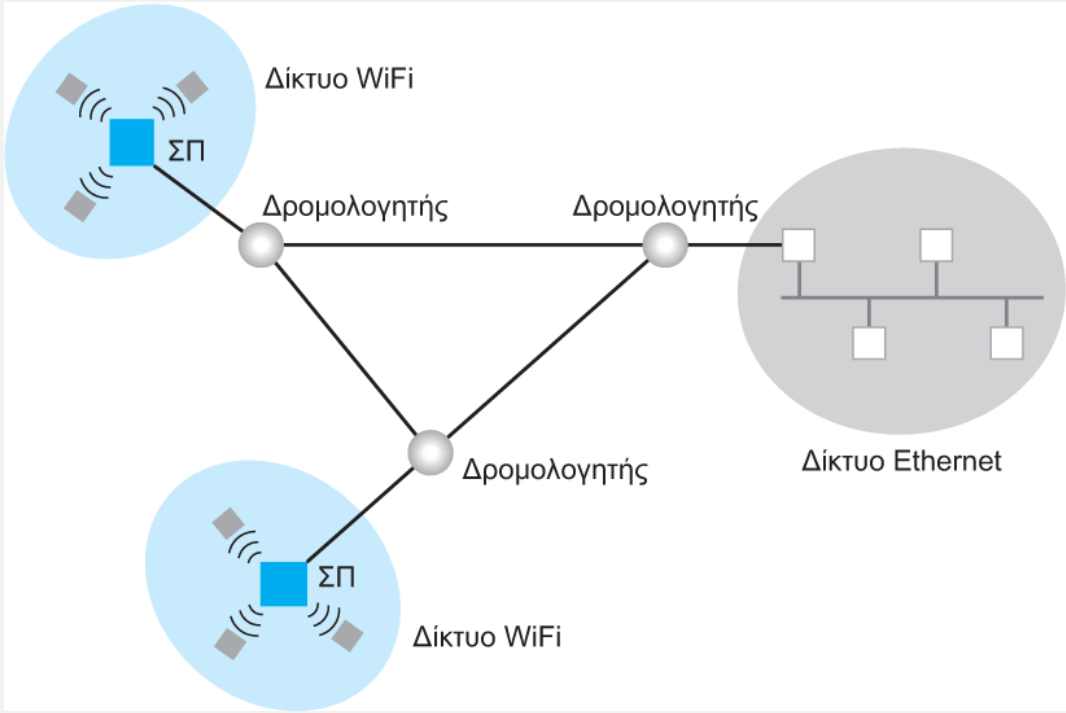
ΔΙΚΤΥΟ | INTERNET

- **Δρομολογητής** (Router): δικτυακή συσκευή, προωθεί/δρομολογεί την πληροφορία (πακέτα) μεταξύ δικτύων προς τον τελικό προορισμό.
- **Πακέτο** (Packet): μονάδα (αριθμός bits) μετάδοσης πληροφορίας /δεδομένων στο Internet, π.χ. 1 πακέτο = 1500 bytes



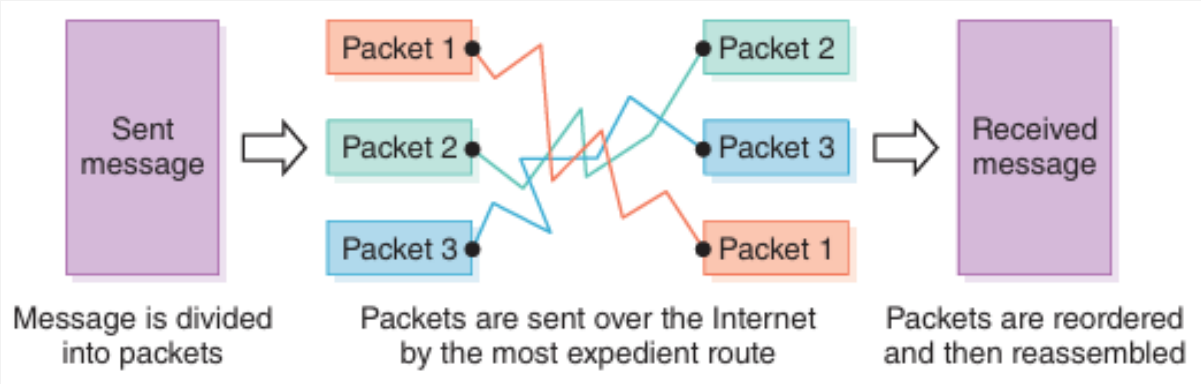
INTERNET | PACKET SWITCHING

- **Μεταγωγή πακέτου** (Packet Switching): κάθε πακέτο δρομολογείται αυτόνομα προς τον προορισμό του, το αρχικό μήνυμα συναρμολογείται στον προορισμό → βελτίωση απόδοσης μετάδοσης δεδομένων σε διαμοιραζόμενο δίαυλο
- Δικτυακές συσκευές → **δρομολογητές (routers)** αναλαμβάνουν την προώθηση των πακέτων στο δίκτυο, δεν σχεδιάζουν ολόκληρη τη διαδρομή, καθένας γνωρίζει το καλύτερο επόμενο βήμα προς τον προορισμό. Τελικά το πακέτο θα φτάσει σε δρομολογητή που «γνωρίζει» που συνδέεται ο προορισμός.



INTERNET | PACKET SWITCHING

- Κάθε πακέτο μπορεί να ακολουθήσει **διαφορετική** διαδρομή και να φτάσει **εκτός σειράς**. Στον προορισμό τα πακέτα μπαίνουν στη σειρά για την ανασύνθεση του αρχικού μηνύματος.
- Τα πακέτα μπορεί να περάσουν από διάφορους ενδιάμεσους κόμβους του δικτύου (δικτυακές συσκευές) πριν φτάσουν στον τελικό προορισμό.



ΔΟΜΗ DATA ΠΑΚΕΤΟΥ

- DATA ή Payload: bits πληροφορίας (χρήσιμα δεδομένα) που κουβαλάει το πακέτο
- Transmitter ID: διεύθυνση αποστολέα
- Receiver ID: διεύθυνση παραλήπτη

ΠΡΟΣΟΜΟΙΩΣΗ ΜΕΤΑΔΟΣΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕ PYTHON

Τμηματοποίηση, Πακετοποίηση & Ανασύνθεση Δεδομένων

1. ΣΕΝΑΡΙΟ & ΠΑΡΑΜΕΤΡΟΙ

- Ορισμός μηνύματος και βασικών παραμέτρων προσομοίωσης
  - ORIGINAL_MESSAGE: *Το μεγάλο μήνυμα που θέλουμε να στείλουμε.*
  - PACKET_SIZE: *Καθορίζει το μέγιστο μέγεθος του Payload (ωφέλιμα δεδομένα) σε κάθε πακέτο. Έστω 12 χαρακτήρες για την προσομοίωση.*
  - SENDER_IP & RECEIVER_IP: *Οι διευθύνσεις IP (ταυτοποίηση) του αποστολέα και του παραλήπτη.*

```
# --- 1. ΠΑΡΑΜΕΤΡΟΙ ---
ORIGINAL_MESSAGE = "Αυτό είναι ένα μεγάλο μήνυμα που πρέπει να ταξιδέψει στο δίκτυο και να φτάσει ακέραιο."
PACKET_SIZE = 12  # Μέγιστο μήκος δεδομένων (Payload) ανά πακέτο
SENDER_IP = "192.168.1.10"
RECEIVER_IP = "10.0.0.5"
```

ΣΤΟΧΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

- **Μετάδοσης δεδομένων στο Δίκτυο**
  - πώς ένα μεγάλο μήνυμα χωρίζεται σε μικρότερα κομμάτια (πακέτα)
  - ταξιδεύει στο Δίκτυο (όπου η σειρά των μηνυμάτων μπορεί να χαθεί)
  - ανασυντίθεται σωστά στον τελικό προορισμό

2. ΤΜΗΜΑΤΟΠΟΙΗΣΗ & ΔΗΜΙΟΥΡΓΙΑ ΠΑΚΕΤΩΝ

Μία συνάρτηση **create_packets(data, size)** *[στον αποστολέα]*

- Χωρίζει το (μεγάλο) μήνυμα σε μικρότερα κομμάτια (segments), μεγέθους ίσο με το PACKET_SIZE
- Σε κάθε κομμάτι δεδομένων (payload) προσθέτει μία κεφαλίδα (header) με εξτρά πληροφορίες για δρομολόγηση και ανασύνθεση
  - Source_IP & Dest_IP: για το στρώμα Δικτύου (π.χ. IP).
  - Payload_Length: το μήκος των δεδομένων.
  - Seq_Num (Αριθμός Σειράς): χρειάζεται για την ανασύνθεση. Δίνει ένα μοναδικό αύξοντα αριθμό σειράς (0, 1, 2, …) σε κάθε πακέτο.

2. ΤΜΗΜΑΤΟΠΟΙΗΣΗ & ΔΗΜΙΟΥΡΓΙΑ ΠΑΚΕΤΩΝ (PYTHON)

```
def create_packets(data, size):
  packets = []
   # 1. Τμηματοποίηση: Χωρίζουμε τα δεδομένα
   segments = []
   i = 0
   while i < len(data):
     # Παίρνουμε το κομμάτι μεγέθους 'size'
      segment = data[i:i + size]
      segments.append(segment)
      # Προχωράμε στην επόμενη αρχική θέση
      i += size
```

2. ΤΜΗΜΑΤΟΠΟΙΗΣΗ & ΔΗΜΙΟΥΡΓΙΑ ΠΑΚΕΤΩΝ(2) (PYTHON)

 # 2. Προσθήκη Κεφαλίδας (Αριθμός Σειράς)

διασχίζει τη λίστα segments, κάθε στοιχείο της αντιστοιχεί στο payload του πακέτου, enumerate(): παράγει έναν αυξανόμενο μετρητή για κάθε στοιχείο, που αποθηκεύεται στη μεταβλητή seq_num

```
for seq_num, payload in enumerate(segments):
    # ... Δημιουργία Πακέτου ως Λεξικό 'packet' με 'Header' και 'Payload'
    packet = {
        "Header": {
            "Source_IP": SENDER_IP,
            "Dest_IP": RECEIVER_IP,
            "Seq_Num": seq_num,  # Ο ΚΡΙΣΙΜΟΣ ΑΡΙΘΜΟΣ ΣΕΙΡΑΣ
            "Payload_Length": len(payload)
        },
        "Payload": payload
    }
    packets.append(packet)
return packets
```

2. ΤΜΗΜΑΤΟΠΟΙΗΣΗ & ΔΗΜΙΟΥΡΓΙΑ ΠΑΚΕΤΩΝ(3)

από μια **απλή λίστα** με κομμάτια κειμένου:

['Αυτό είναι ', 'ένα μεγάλο ', 'μήνυμα που '] ...

σε **λίστα δομημένων πακέτων**:

```
[
{ "Header":
        { "Dest_IP": "10.0.0.5", "Seq_Num": 0, ... },
        "Payload": "Αυτό είναι " },
{ "Header":
        { "Dest_IP": "10.0.0.5", "Seq_Num": 1, ... },
        "Payload": "ένα μεγάλο " }
// ... κ.ο.κ. ]
```

3. ΠΡΟΣΟΜΟΙΩΣΗ ΔΙΚΤΥΟΥ

- Μία συνάρτηση **transmit(packets)** *[στον αποστολέα]*
  - Προσομοιώνει την πραγματική λειτουργία του δικτύου (π.χ. Internet)
  - Τα πακέτα μπορούν να ακολουθήσουν διαφορετικές διαδρομές (routes) από τον αποστολέα στον παραλήπτη.
- Φτάνουν στον προορισμό με διαφορετική σειρά από αυτή που στάλθηκαν → **random.shuffle(packets)**

3. ΠΡΟΣΟΜΟΙΩΣΗ ΔΙΚΤΥΟΥ (PYTHON)

```
def transmit(packets):
    # Τα πακέτα ανακατεύονται για να προσομοιώσουν διαφορετικές διαδρομές
    random.shuffle(packets)
    print("\n--- ΔΙΚΤΥΟ: Τα πακέτα έφτασαν στον προορισμό με ΑΛΛΗ σειρά! ---")
    return packets
```

4. ΑΝΑΣΥΝΘΕΣΗ (REASSEMBLY) ΜΗΝΥΜΑΤΟΣ

- Μία συνάρτηση **reassemble_message(received_packets)** *[στον παραλήπτη]*
  - Το αντίστροφο της δημιουργίας των πακέτων, ο παραλήπτης ανασυνθέτει το αρχικό (μεγάλο) μήνυμα.
- **Α. Ταξινόμηση (sorting)** → απαραίτητος ο Αριθμός Σειράς!
  - Μέθοδος **sorted()** της Python
  - Παράμετρος key=lambda p: p['Header']['Seq_Num'] → ταξινόμηση ολόκληρης της λίστας πακέτων με βάση το πεδίο Seq_num μέσα στην κεφαλίδα κάθε πακέτου.
  - lambda p: ορίζει μία μικρή ανώνυμη συνάρτηση, παίρνει ως είσοδο ένα στοιχείο (p για όλο το πακέτο), επιστρέφει την τιμή στη θέση p['Header']['Seq_Num']

4. ΑΝΑΣΥΝΘΕΣΗ (REASSEMBLY) ΜΗΝΥΜΑΤΟΣ (2)

- Μία συνάρτηση **reassemble_message(received_packets)** *[στον παραλήπτη]*
- **Β. Ανασύνθεση (joining)**
  - Αφού τα πακέτα έχουν μπει στη σωστή σειρά, τα payload τους ενώνονται (concatenate) με τη μέθοδο "".join(…) για ανασύνθεση του αρχικού μηνύματος

4. ΑΝΑΣΥΝΘΕΣΗ (REASSEMBLY) ΜΗΝΥΜΑΤΟΣ (PYTHON)

```
def reassemble_message(received_packets):
    # ...
    # 2. Ταξινόμηση: Χρησιμοποιούμε τον Seq_Num για την ταξινόμηση
    sorted_packets = sorted(received_packets, key=lambda p:
p['Header']['Seq_Num'])

    # 3. Ανασύνθεση: Ενώνουμε τα Payload με τη σωστή σειρά
    reconstructed_data = "".join([p['Payload'] for p in sorted_packets])

    return reconstructed_data
```

ΕΠΙΣΗΜΑΝΣΕΙΣ

- Χωρίς τον Αριθμό Σειράς, η ανασύνθεση του μηνύματος θα γινόταν με τυχαία σειρά και θα ήταν ακατανόητο (όπως δείχνει η εμφάνιση του received_packets πριν την ανακατασκευή).
- Ο μηχανισμός του Αριθμού Σειράς είναι αυτός που επιτρέπει στα πρωτόκολλα όπως το TCP να διασφαλίζουν την αξιόπιστη (reliable) και με τη σωστή σειρά (in-order) παράδοση δεδομένων, ακόμα και σε ένα δίκτυο όπου η σειρά παράδοσης δεν είναι εγγυημένη.
- Κώδικας: net-sim.py

ΜΙΑ ΠΙΟ ΡΕΑΛΙΣΤΙΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΛΕΙΤΟΥΡΓΙΑΣ ΔΙΚΤΥΟΥ

Πολλαπλές διαδρομές (routing), απώλεια πακέτων (loss), και **καθυστέρηση (delay)**

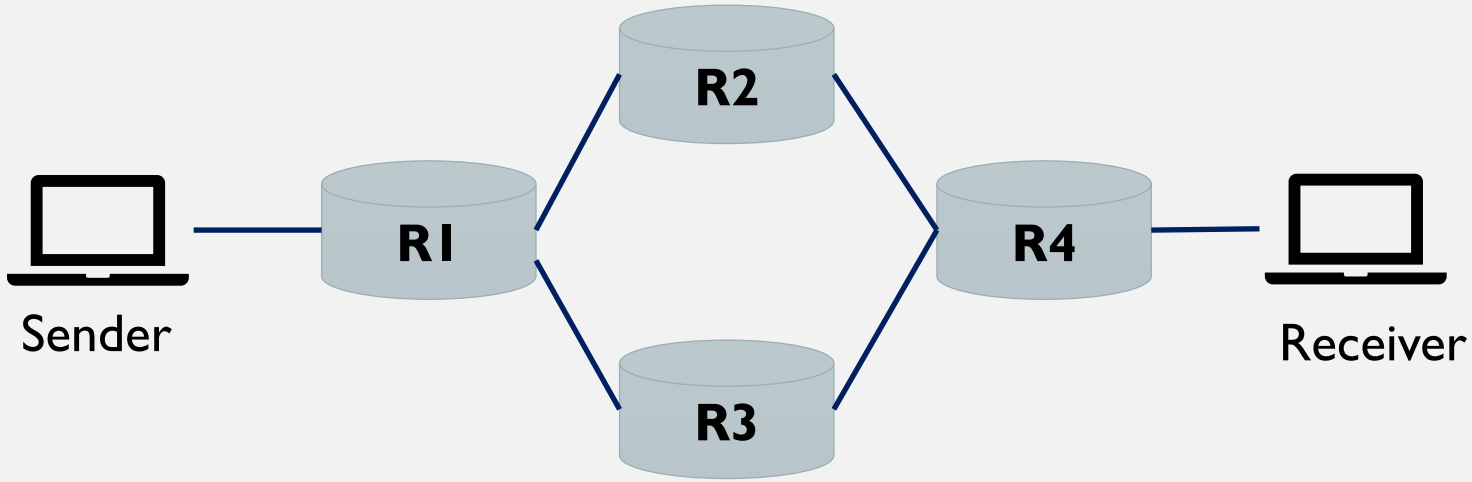
1. ΝΕΕΣ ΠΑΡΑΜΕΤΡΟΙ ΔΙΚΤΥΟΥ (ΕΠΕΚΤΑΣΗ)

- Εκτός από τις βασικές παραμέτρους (PACKET_SIZE, IPs), προσθέτουμε παραμέτρους που προσομοιώνουν ένα μη-ιδανικό δίκτυο
  - RANDOM_SEED: Διασφαλίζει ότι οι τυχαίες λειτουργίες (απώλεια, καθυστέρηση) είναι επαναλαμβανόμενες. *(Δηλαδή, η σειρά με την οποία ανακατεύονται τα πακέτα (random.shuffle()) θα είναι πάντα ίδια. Τα πακέτα που χάνονται (όταν εφαρμόζεται η random.random() < PACKET_LOSS_PROB) θα είναι πάντα τα ίδια. Οι καθυστερήσεις (delay = ... random.random()) θα είναι πάντα οι ίδιες*)
  - PACKET_LOSS_PROB: Η πιθανότητα απώλειας ενός πακέτου κατά τη μετάδοση (εδώ 5% ή 0.05).
  - MIN_DELAY / MAX_DELAY: Καθορίζουν το εύρος των καθυστερήσεων.

```
# --- 1. ΕΠΙΠΛΕΟΝ ΠΑΡΑΜΕΤΡΟΙ ---
# PACKET_LOSS_PROB = 0.05 (Πιθανότητα απώλειας)
# MIN_DELAY / MAX_DELAY (Εύρος καθυστέρησης)
```

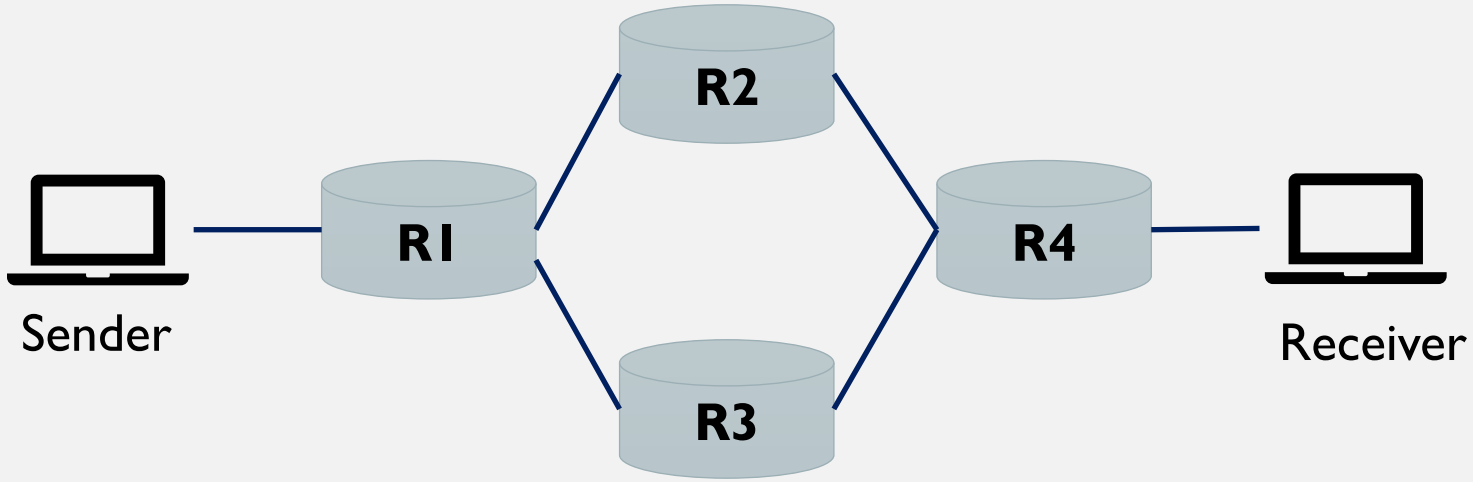
2. ΔΡΟΜΟΛΟΓΗΣΗ ΚΑΙ ΜΕΤΑΔΟΣΗ

- **Τοπολογία Δικτύου**
  - Αποστολέας (S) → Router 1 (R1)
  - R1 επιλέγει ανάμεσα σε R2 ή R3
  - R2/R3 → Router 4 (R4) → Παραλήπτης (R)



2. ΔΡΟΜΟΛΟΓΗΣΗ ΚΑΙ ΜΕΤΑΔΟΣΗ (2)

- **Στρατηγική δρομολόγησης** (από το δρομολογητή R1)
  Συνάρτηση route_from_R1: αποφασίζει ποια διαδρομή (R2 ή R3) θα πάρει κάθε πακέτο.
  - Πολιτική random: επιλέγει τυχαία τη διαδρομή
  - Πολιτική round_robin: εναλλάσσει συστηματικά τη διαδρομή (πακέτο 0 → R2, πακέτο 1 → R3, πακέτο 2 → R2, κ.ο.κ.)



2. ΔΡΟΜΟΛΟΓΗΣΗ ΚΑΙ ΜΕΤΑΔΟΣΗ (3)

- **Προσομοίωση Απώλειας και Καθυστέρησης**
  Συνάρτηση transmit_via_network
  - **Απώλεια**: έλεγχος if random.random() < loss_prob, *αν το πακέτο χαθεί, αφαιρείται από τη μετάδοση…*
  - **Καθυστέρηση:**
    - Κάθε διαδρομή (μέσω R2 ή R3) διαφορετικό χρόνο μετάδοσης (base)
    - Προστίθεται μία τυχαία καθυστέρηση (random.random() * (MAX_DELAY - base))
    - Τα πακέτα μπαίνουν σε ουρές (path_queues) με βάση το χρόνο άφιξης (arrival_time).
  - Συλλογή πακέτων: στον προορισμό όλα τα πακέτα συλλέγονται και ταξινομούνται βάσει χρόνου άφιξης

2. ΔΡΟΜΟΛΟΓΗΣΗ ΚΑΙ ΜΕΤΑΔΟΣΗ (PYTHON)

```
# Ελεγχος απώλειας
if random.random() < loss_prob:
    # ... το πακέτο ΧΑΝΕΤΑΙ
    continue


# Υπολογισμός καθυστέρησης και χρόνου άφιξης
delay = base + random.random() * (MAX_DELAY - base)
arrival_time = time.time() + delay
# ... ταξινόμηση βάσει arrival_time
```

3. ΟΠΤΙΚΟΠΟΙΗΣΗ ΔΙΑΔΡΟΜΗΣ

- Συναρτήσεις οπτικοποίησης διαδρομής που ακολούθησε το πακέτο
- visualize_packet_ascii (*ASCII Art*): χρησιμοποιεί απλούς χαρακτήρες ASCII για να σχηματίσει το διάγραμμα ροής, εμφανίζοντας την ακολουθία των κόμβων (π.χ., S → R1 → R3 → R4 → R).
- visualize_packet_grid (*Grid View*): χρησιμοποιεί ένα πλέγμα (grid) για να δώσει μια πιο δομημένη αναπαράσταση της τοπολογίας, όπου οι θέσεις (r, c) αντιστοιχούν στους κόμβους.

ΕΠΙΣΗΜΑΝΣΕΙΣ

- **Καθυστέρηση και Σειρά Άφιξης**: Λόγω των διαφορετικών διαδρομών (R2/R3) και των διαφορετικών καθυστερήσεων, τα πακέτα φτάνουν στον παραλήπτη σε **διαφορετική σειρά από αυτήν που στάλθηκαν**, ακόμη και αν χρησιμοποιήσαμε στρατηγική round_robin!
- **Απώλεια**: Η απώλεια πακέτων είναι ρεαλιστική. Στην πράξη (π.χ. TCP), αν ένα πακέτο χαθεί, ο αποστολέας θα πρέπει να λάβει επιβεβαίωση (ACK) για όλα τα πακέτα που έφτασαν και να **ξαναστείλει (retransmit)** το χαμένο πακέτο.
- **Ο Ρόλος του Seq_Num**: Για άλλη μια φορά, ο **Αριθμός Σειράς** είναι αυτός που επιτρέπει στον παραλήπτη:
  - Να αναγνωρίσει ποια πακέτα λείπουν (αν υπήρξε απώλεια).
  - Να ταξινομήσει σωστά τα πακέτα που έφτασαν (παρά την ανακατεμένη σειρά άφιξης λόγω καθυστέρησης).
- Κώδικας: net_routing_sim.py