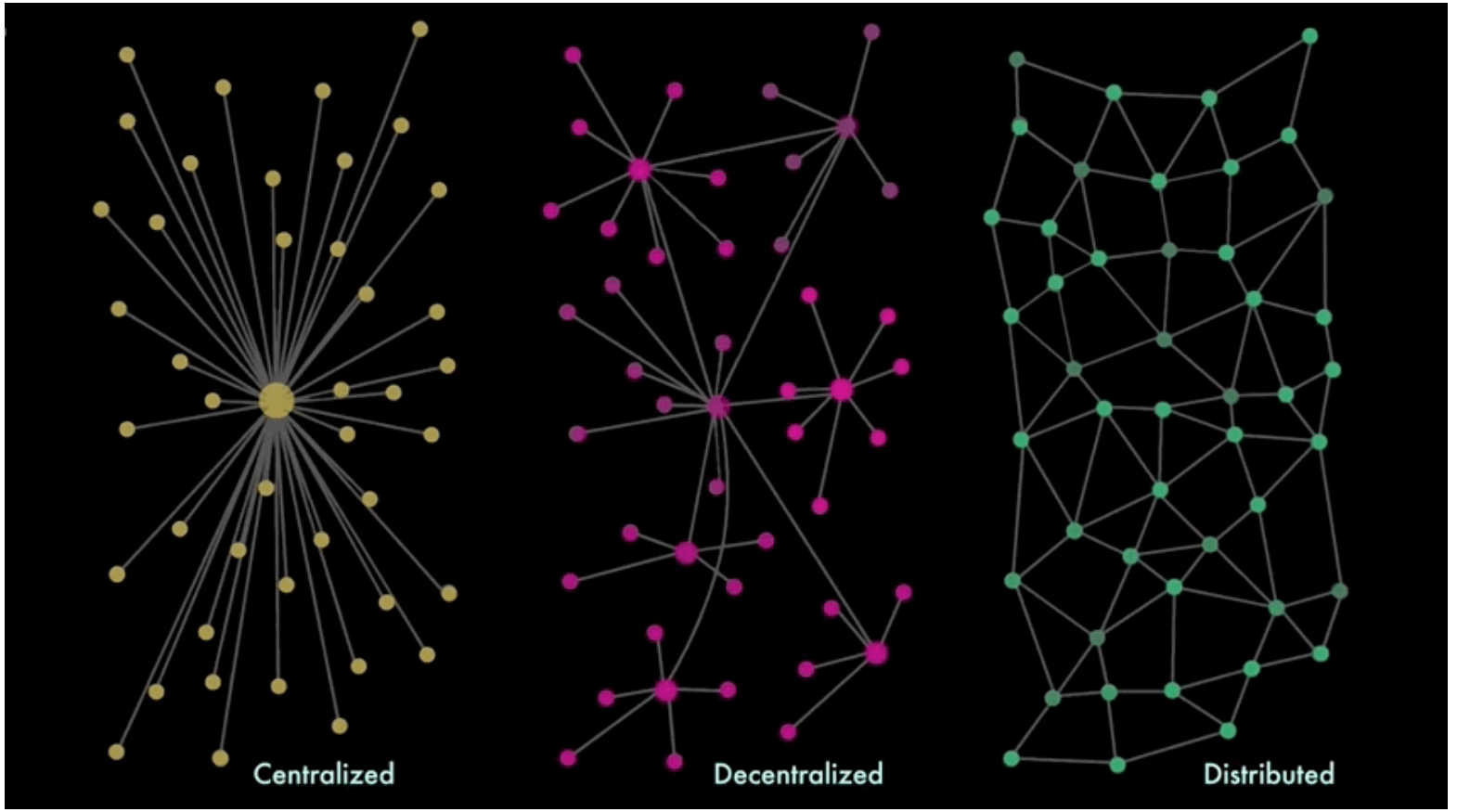
Αρχιτεκτονικές Πληροφοριακών Συστημάτων

Κατανεμημένα Συστήματα

Κεντρικοποιημένα, Αποκεντρωμένα και Κατανεμημένα Συστήματα

Centralised, decentralized and distributed systems

Κεντρικοποιημένα, αποκεντρωμένα και κατανεμημένα συστήματα

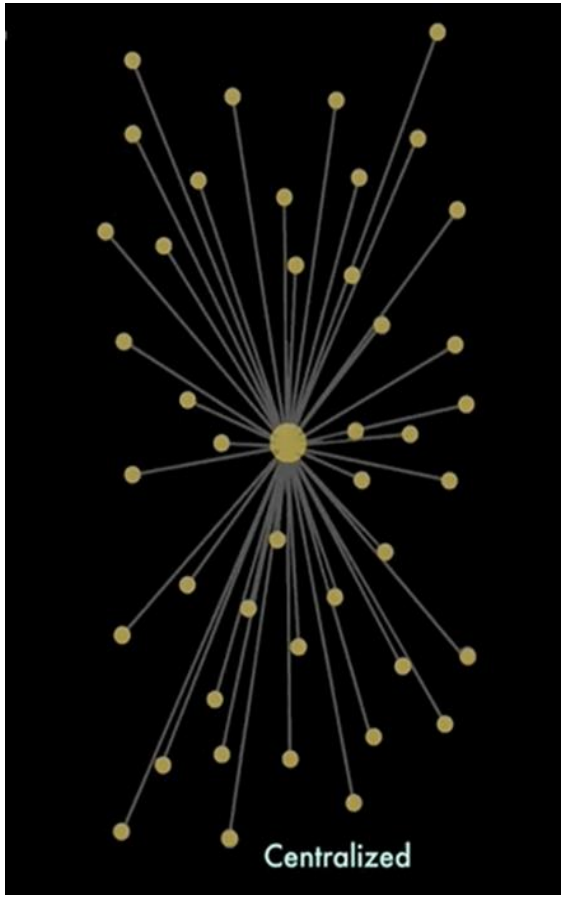


Κεντρικοποιημένα Συστήματα

- Το σύστημα ελέγχεται από **μία κεντρική οντότητα**
- Η επεξεργασία γίνεται (και ελέγχεται) κεντρικά από την κεντρική οντότητα (κεντρικός εξυπηρετητής

(οικονομικό ανάλογο: μονοπώλιο)

- **Θετικά**
  - Είναι απλή στη σύλληψη και τη λειτουργία με βάση το μοντέλο εξυπηρετητή-πελάτη
- **Αρνητικά**
  - Η κεντρική οντότητα είναι αναντικατάστατη, και αποτελεί «κεντρικό σημείο αστοχίας» του συστήματος
  - Είναι δύσκολη η προσαρμογή του συστήματος σε μεταβλητές απαιτήσεις (π.χ. αύξηση της ζήτησης υπολογιστικής ισχύος)



Αποκεντρωμένα Συστήματα

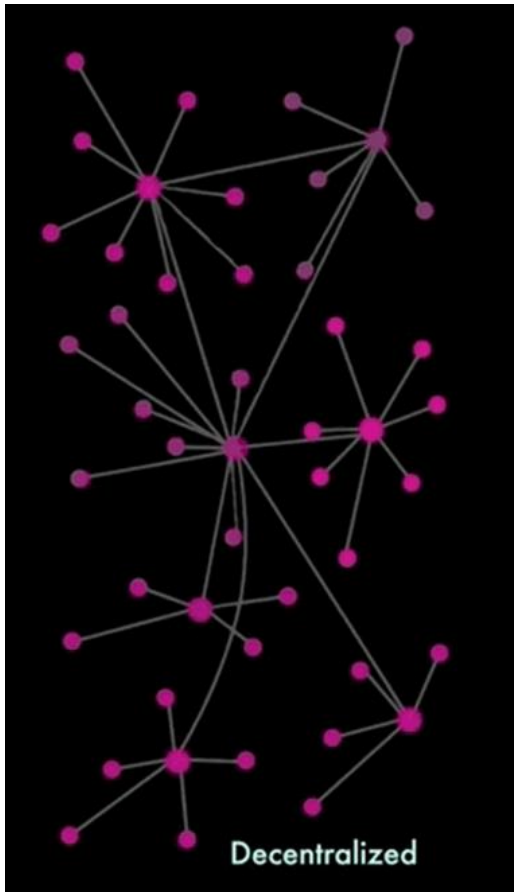
- Το σύστημα ελέγχεται από **πολλές κεντρικές οντότητες** που «μιλάνε» μεταξύ τους
- Οι κεντρικές οντότητες «συνεργάζονται» και ο υπολογιστικός φόρτος «κατανέμεται»

(οικονομικό ανάλογο: ολιγοπώλιο)

Παράδειγμα:

προκειμένου να μπορούμε να παρέχουμε video σε ολόκληρο τον πλανήτη, μπορούμε π.χ. να έχουμε ένα video server ανά γεωγραφική ενότητα

- **Θετικά**
  - Είναι περισσότερο ανθεκτική σε αστοχίες μονού σημείου
- **Αρνητικά**
  - Εξακολουθεί να υπάρχει ένα σύνολο οντοτήτων που ελέγχει τη διαδικασία στο σύνολό της.



Κατανεμημένα Συστήματα

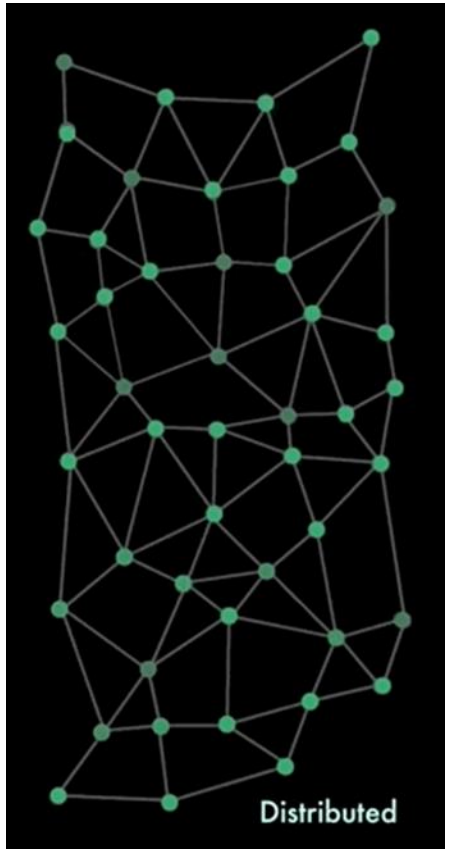
- Όλοι οι κόμβοι ενός ομότιμου κατανεμημένου συστήματος είναι «ίσοι» μεταξύ τους, είναι ταυτόχρονα **και πελάτες και εξυπηρετητές**

(Οικονομικό ισοδύναμο: «Ομότιμα» νομίσματα βασισμένα στο **Blockchain**)

- Στα κατανεμημένα συστήματα, η ίδια εφαρμογή έχει τμήματα που λειτουργούν ως **εξυπηρετητής** και τμήματα που λειτουργούν ως **πελάτης**. Πολλές φορές τα τμήματα αυτά εκτελούνται παράλληλα (σε πολλαπλά «νήματα»)

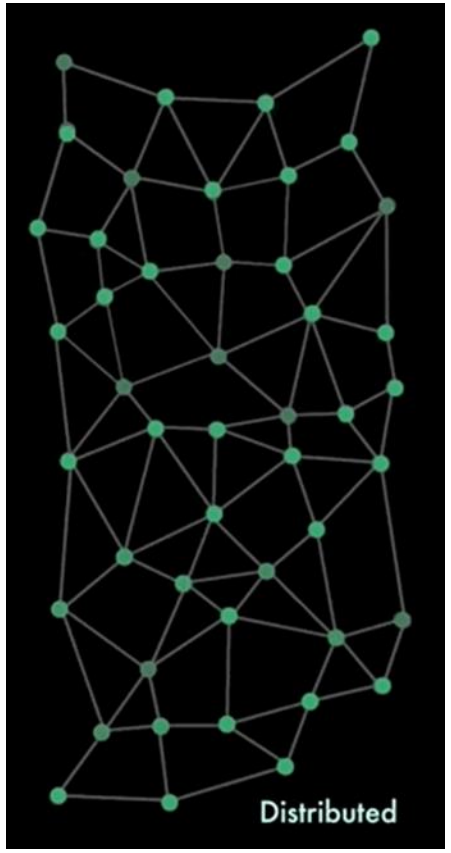
- Το τμήμα της εφαρμογής που παίζει το ρόλο του εξυπηρετητή, «**ακούει**», περιμένοντας από κάποιο πελάτη να της ζητήσει να κάνει κάτι

- Το τμήμα της εφαρμογής που παίζει το ρόλο του πελάτη, όταν αυτό χρειαστεί (π.χ. σε απόκριση ενός κλικ του χρήστη) «**ζητάει**» δεδομένα από τους διαθέσιμους κάθε φορά κόμβους του δικτύου



Κατανεμημένα Συστήματα

- Γνωστές εφαρμογές:
  - Ομότιμα (p2p) δίκτυα τύπου torrent
  - Blockchain
  - Smart contracts
  - Νέες άυλες ψηφιακές τράπεζες
  - Νέα κρυπτογραφικά νομίσματα



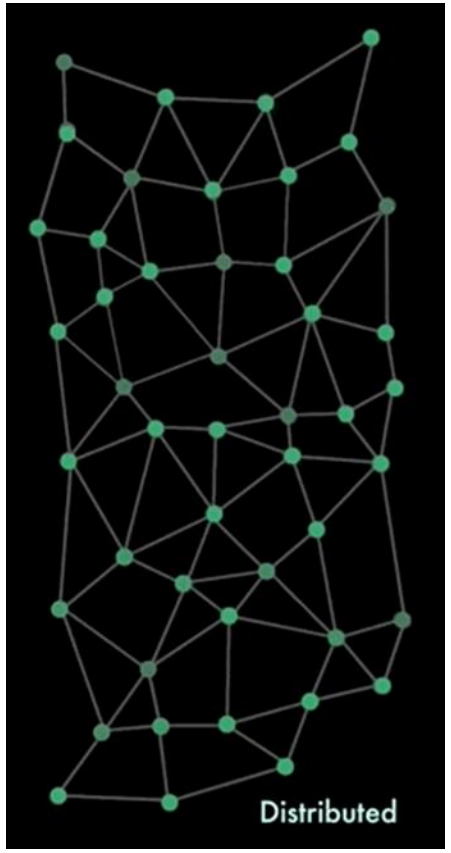
Κατανεμημένα Συστήματα

- Η **ανθεκτικότητα** των κατανεμημένων συστημάτων σε περιβάλλοντα υψηλής αβεβαιότητας μελετάται εκτενώς και **βελτιώνεται διαρκώς**
- Τα κατανεμημένα συστήματα παρουσιάζουν αξιοσημείωτη σταθερότητα και ανθεκτικότητα σε κακόβουλες επιθέσεις, ακόμα **και σε περιβάλλοντα με σημαντικό ποσοστό σφαλμάτων**

- **Θετικά**
  - Είναι **εύρωστη** και **ανθεκτική** σε φυσικές καταστροφές και κακόβουλες επιθέσεις
  - Είναι απόλυτα **διαφανής**, (όλοι οι κόμβοι εκτελούν τον ίδιο κώδικα)

- **Αρνητικά**
  - Νέα φιλοσοφία ανάπτυξης λογισμικού, με **αρκετούς τεχνικούς περιορισμούς** και δυσκολίες
  - Η επίτευξη συμφωνίας στο δίκτυο των κόμβων (consensus) απαιτεί σεβασμό σε συγκεκριμένους πλειοψηφικούς κανόνες, χωρίς τους οποίους δεν γίνεται να υπάρξει το σύστημα.

Θέμα για προβληματισμό:
Πώς επιτυγχάνεται συναίνεση στα κατανεμημένα συστήματα?

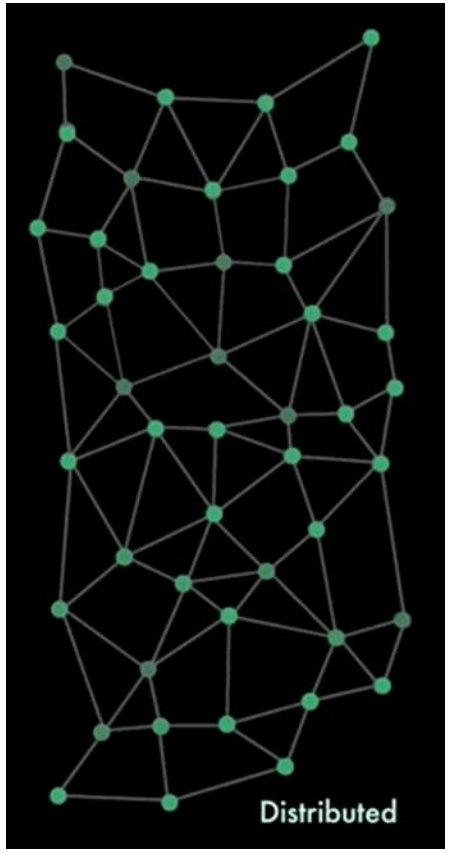


Επικοινωνία μεταξύ των υπολογιστών-κόμβων

Πως «μιλάνε» οι κόμβοι μεταξύ τους;

Πώς μιλάνε οι εφαρμογές μεταξύ τους;

- Η διαδικασία της «συνομιλίας» μεταξύ των εφαρμογών είναι «διάφανη» στον τελικό χρήστη
- Απαραίτητη προϋπόθεση είναι η ύπαρξη αξιόπιστου δικτύου επικοινωνίας* μεταξύ τους



"Το δίκτυο είναι ο υπολογιστής"
*SUN microsystems 1992

Πώς μιλάνε οι εφαρμογές μεταξύ τους;

- Οι εφαρμογές επικοινωνούν μεταξύ τους **μέσω καλά δομημένων μηνυμάτων**
- Η δομή και ο τύπος του περιεχομένου των μηνυμάτων **είναι γνωστά από την αρχή**, από τη φάση του σχεδιασμού του συστήματος
- Κάθε εφαρμογή* παρέχει στις υπόλοιπες ένα σετ από μηνύματα τα οποία μπορεί να εξυπηρετήσει. Αυτά είναι γνωστό με τον όρο **Application Programmable Interface (API)**
- Το API περιγράφεται σε κάποια μορφή «μεταγλώσσας» (XML, json, XMLNS, XHTML,…)

Αίτημα πελάτη:

http://sampleserver.gr/?Request=GetCapabilities

Απόκριση εξυπηρετητή:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DPIWS SYSTEM "http://localhost/DPIWS/schemas/1.0.0/DPIWS.dtd">
<DPIWS_Capabilities version="1.0.0">
  <Service>
    <Name>DPIWS</Name>
    <Title>Digital Pathology Information Web Service</Title>
    <Abstract>A compliant implementation of DPIWS</Abstract>
    <ContactInformation>
      <ContactPersonPrimary>
        <ContactPerson>Administrator</ContactPerson>
        <ContactOrganization>Digital Pathology INC</ContactOrganization>
      </ContactPersonPrimary>
      <ContactPosition>Pathologist</ContactPosition>
      <ContactAddress>
        <AddressType>Cloud</AddressType>
        <Address/>
        <City>Anywhere</City>
        <StateOrProvince/>
        <PostCode/>
        <Country>Somewhere</Country>
      </ContactAddress>
      <ContactVoiceTelephone/>
      <ContactFacsimileTelephone/>
      <ContactElectronicMailAddress>claudius.ptolomaeus@gmail.com</ContactElectronicMailAddress>
    </ContactInformation>
    <AccessConstraints>NONE</AccessConstraints>
  </Service>
  <Capability>
</DPIWS_Capabilities>
```

Πώς μιλάνε οι εφαρμογές μεταξύ τους;

Παράδειγμα API

```
DateTimeAPI {
    /*
     *  Abstract API for interfacing with the DateTime contract
     */
    function isLeapYear(uint16 year) constant returns (bool);
    function getYear(uint timestamp) constant returns (uint16);
    function getMonth(uint timestamp) constant returns (uint8);
    function getDay(uint timestamp) constant returns (uint8);
    function getHour(uint timestamp) constant returns (uint8);
    function getMinute(uint timestamp) constant returns (uint8);
    function getSecond(uint timestamp) constant returns (uint8);
    function getWeekday(uint timestamp) constant returns (uint8);
    function toTimestamp(uint16 year, uint8 month, uint8 day) constant returns (uint timestamp);
    function toTimestamp(uint16 year, uint8 month, uint8 day, uint8 hour) constant returns (uint timestamp);
    function toTimestamp(uint16 year, uint8 month, uint8 day, uint8 hour, uint8 minute) constant returns (uint timestamp);
    function toTimestamp(uint16 year, uint8 month, uint8 day, uint8 hour, uint8 minute, uint8 second) constant returns (uint timestamp);
```

Κλήση συνάρτησης (τι θέλουμε να κάνει ο εξυπηρετητής για εμάς)

Παράμετροι (τι πρέπει να ξέρει από εμάς ο εξυπηρετητής για να μας δώσει αυτό που θέλουμε)

Απάντηση (τι μας επιστρέφει ο εξυπηρετητής σε απόκριση στην κλήση μας)

*Timestamp: μέθοδος του Linux που μετράει το χρόνο

Πώς μιλάνε οι εφαρμογές μεταξύ τους;

Η επικοινωνία μεταξύ των κόμβων μπορεί να είναι:

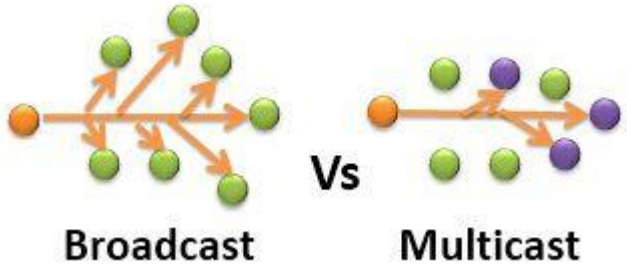
- Σημείου προς σημείο (Point to Point). Κάθε κόμβος ανταλλάσσει πληροφορία με ένα κόμβο (π.χ. εφαρμογή τηλεφωνίας)

- Multicasting / Anycasting

  (ο εξυπηρετητής «σερβίρει» περιεχόμενο σε μια ομάδα επιλεγμένων κόμβων - π.χ. τηλεδιάσκεψη πολλαπλών μερών)

- Broadcasting

  (ο εξυπηρετητής «εκπέμπει» περιεχόμενο σε όλους τους κόμβους ενός δικτύου – π.χ. YouTube)



Η επικοινωνία των συστημάτων, ξεκινάει πάντα από δύο κόμβους, έναν πελάτη και έναν εξυπηρετητή.

Πού τρέχει το πρόγραμμα;
Πού είναι αποθηκευμένα τα δεδομένα μου;

Οι ερωτήσεις «του 1.000.000 €» κατά το σχεδιασμό ενός Πληροφοριακού Συστήματος

Πού «τρέχουν» οι εφαρμογές;

Περίπτωση 1: αυτόνομα συστήματα

- Στα “stand-alone” συστήματα η εφαρμογή εγκαθίσταται “μια φορά” και τρέχει τοπικά στο σταθμό εργασίας του χρήστη
- Δεν είναι απαραίτητη η διαρκής ύπαρξη σύνδεσης στο διαδίκτυο, παρά μόνο αν αυτό χρειάζεται προκειμένου η εφαρμογή να πάρει είσοδο από άλλες εφαρμογές (ή να δώσει έξοδο σε αυτές)

Πού «τρέχουν» οι εφαρμογές;

Περίπτωση 2:

- Στα **κεντρικοποιημένα** συστήματα το κύριο μέρος του προγράμματος εκτελείται στον κεντρικό εξυπηρετητή, και το αποτέλεσμα αυτής της επεξεργασίας απεικονίζεται στον «thin client», συνήθως μέσα από μια εφαρμογή πελάτη η οποία (*υποχρεωτικά) τρέχει στον τοπικό υπολογιστή

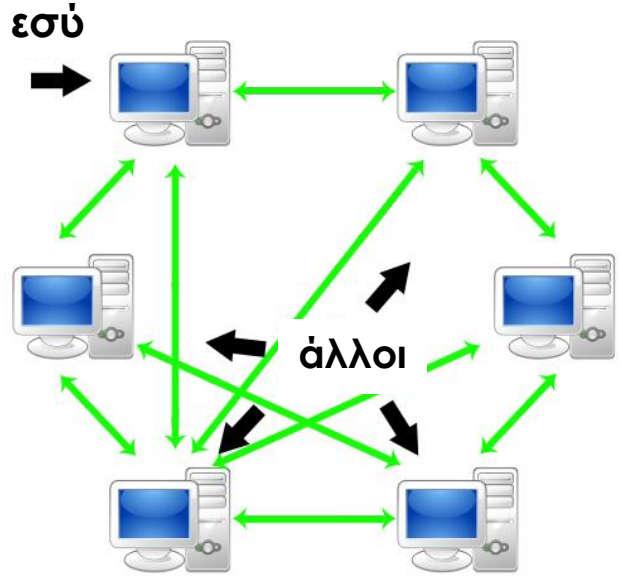
- Στα **αποκεντρωμένα** συστήματα ξανά το κύριο μέρος της επεξεργασίας γίνεται σε κάποιο κεντρικό εξυπηρετητή. Υπάρχουν ωστόσο περισσότεροι από ένας εξυπηρετητές (π.χ. ένας ανά γεωγραφική περιοχή)

- Είναι απαραίτητη η σύνδεση στο δίκτυο κατά το μεγαλύτερο μέρος της εργασίας. Ωστόσο, κάποια τμήματα των δεδομένων αποθηκεύονται συνήθως και τοπικά, δίνοντας έτσι μια μικρή ανοχή στις σύντομες διακοπές του δικτύου.

Πού «τρέχουν» οι εφαρμογές;

Περίπτωση 3:
Κατανεμημένα συστήματα

- Στην περίπτωση των «ομότιμων» κατανεμημένων συστημάτων η αποθήκευση και η επεξεργασία των δεδομένων γίνεται «από λίγο» σε όλους τους υπολογιστές (κόμβους) που μετέχουν στο δίκτυο
- Κάθε κόμβος είναι ταυτόχρονα και εξυπηρετητής και πελάτης
- Είναι απαραίτητη διαρκής σύνδεση στο δίκτυο.



Πού «τρέχουν» οι εφαρμογές;

Το «internet των πραγμάτων»
(Internet of Things - IoT)

- Ακόμα και οι πιο «χαζές» συσκευές είναι εξοπλισμένες με επεξεργαστές που τους επιτρέπουν να εκτελούν προγράμματα
- Στο «Internet των πραγμάτων» είναι εδώ. Οι συσκευές είναι διαρκώς συνδεδεμένες στο δίκτυο, εκτελούν προγράμματα και ανταλλάσσουν δεδομένα