Ανάκληση Πληροφορίας

Information Retrieval

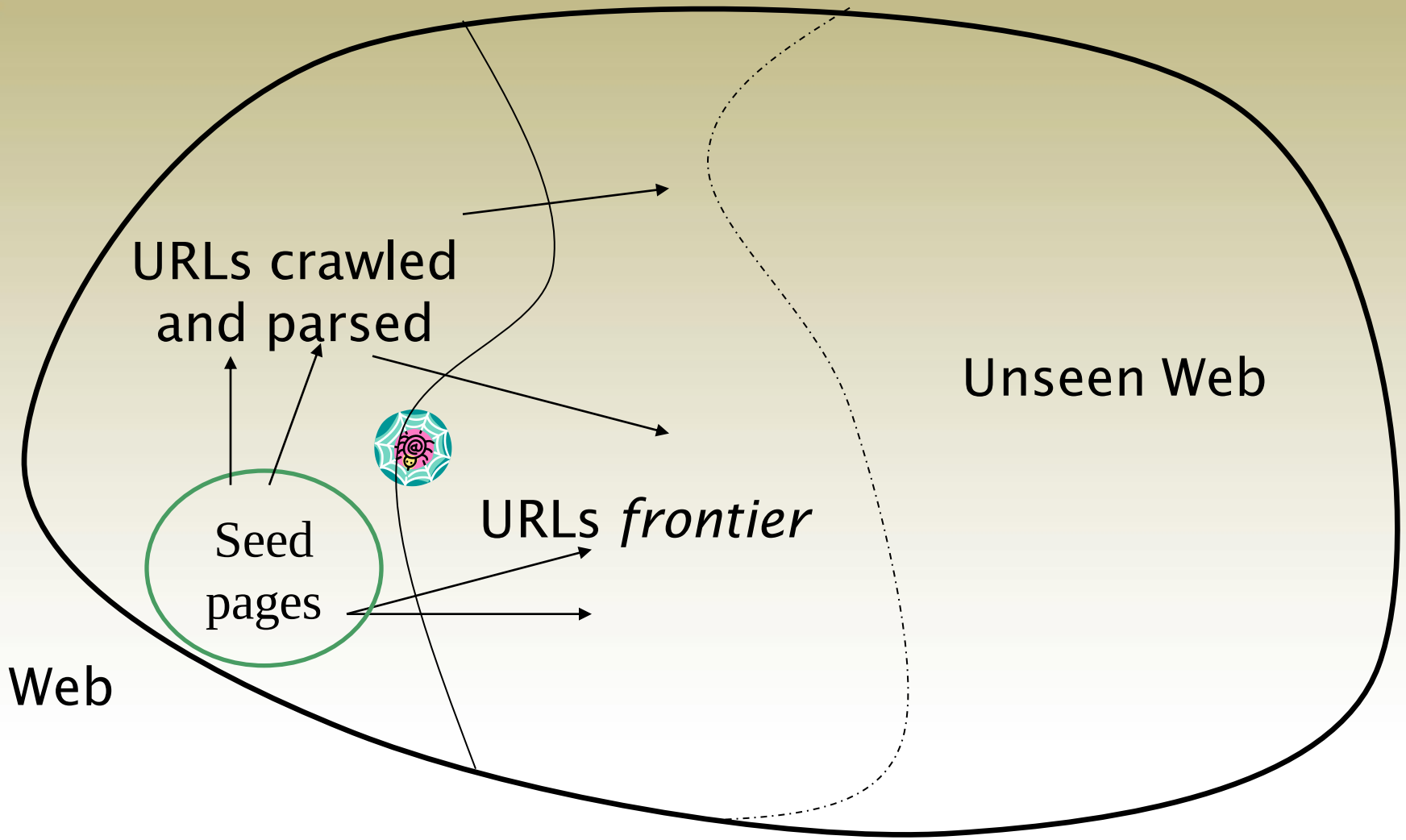
Διδάσκων –
Δημήτριος Κατσαρός

Ερπυστές στον Παγκόσμιο Ιστό

Βασική λειτουργία του crawler

- Αρχίζουμε με γνωστά “seed” URLs
- Τα φέρνουμε και τα κάνουμε parse
  - Εξάγουμε τα URLs στα οποία δείχνουν
  - Τοποθετούμε τα ηξηγμένα URLs σε μια ουρά
- Φέρνουμε κάθε URL της ουράς και επαναλαμβάνουμε

Αναπαράσταση crawling



Απλή εικόνα – επιπλοκές

- Το Web crawling δεν είναι εφικτό σε μια μηχανή
  - Τα προαναφερθέντα βήματα πρέπει να εκτελούνται παράλληλα και κατανεμημένα
- Κακόβουλες σελίδες
  - Spam σελίδες
  - Παγίδες των spiders – περιλαμβανόμενων και των δυναμικά παραγόμενων σελίδων
- Και οι μη κακόβουλες σελίδες παρουσιάζουν προκλήσεις
  - Latency/bandwidth των servers κυμαίνεται
  - Webmasters' συμβουλές
    - Πόσο "βαθιά" να γίνει το crawling;
  - Site mirrors και duplicate pages
- Ευγένεια στις επισκέψεις σε έναν server

Τι θα *πρέπει* να κάνει κάθε crawler

- Να είναι εύρωστος: Να είναι άνοσος στις spider traps και σε άλλες κακόβουλές συμπεριφορές από τους web servers

- Να είναι ευγενικός: Να σέβεται έμμεσες και άμεσες προσεγγίσεις ευγένειες

Άμεσες & έμμεσες ευγενικές προσεγγίσεις

- Άμεση ευγένεια: προσδιορισμοί από τους webmasters σχετικά με ποια τμήματα του site να γίνουν crawled
  - robots.txt
- Έμμεση ευγένεια: ακόμη και χωρίς προσδιορισμούς, αποφυγή πολύ συχνών επισκέψεων

Robots.txt

- Πρωτόκολλο για να παρέχει στους spiders ("robots") περιορισμένη πρόσβαση σε ένα website, από το 1994
  - www.robotstxt.org/wc/norobots.html
- Το Website ανακοινώνει το αίτημά του σχετικά με ποιες σελίδες μπορούν (δεν μπορούν) να γίνουν crawled
  - Για έναν server, δημιουργείται το αρχείο `/robots.txt`
  - Αυτό το αρχείο καθορίζει τους περιορισμούς πρόσβασης

Παράδειγμα Robots.txt

- Κανένα robot δεν θα επισκεφτεί κάποιο URL που ξεκινά με "/yoursite/temp/", εκτός από το robot με όνομα “searchengine":

```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

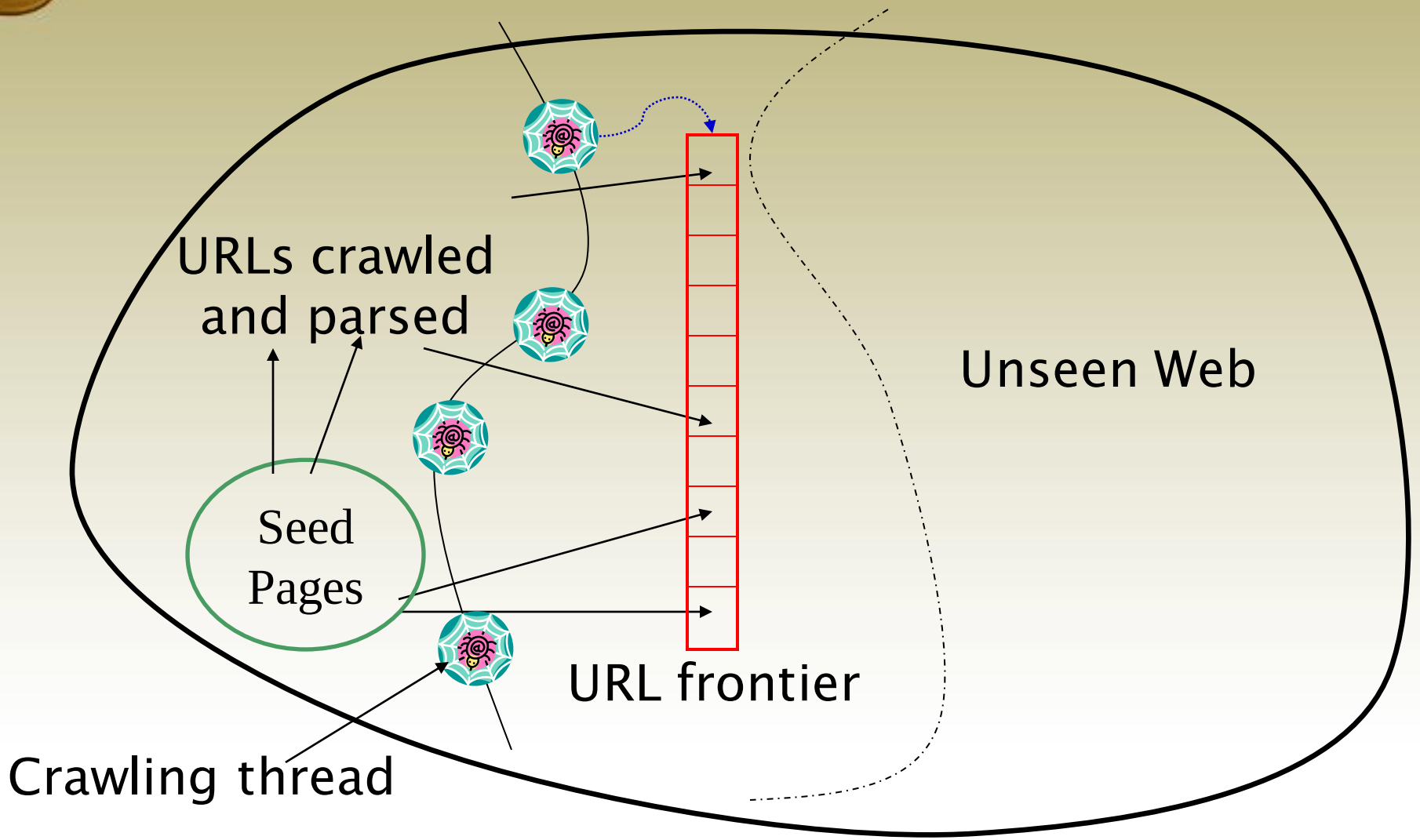
Τι θα πρέπει να *μπορεί να κάνει* κάθε crawler

- Να είναι ικανός για κατανεμημένη λειτουργία: να είναι σχεδιασμένος να εκτελείται σε πολλές κατανεμημένες μηχανές
- Να είναι κλιμακούμενος: να είναι σχεδιασμένος να αυξάνει τον ρυθμό crawling με προσθήκη επιπλέον μηχανών
- Επίδοση/αποδοτικότητα: να κάνει πλήρη χρήση των διαθέσιμων επεξεργαστικών και δικτυακών πόρων

Τι θα πρέπει να *μπορεί να κάνει* κάθε crawler

- Να φέρνει τις σελίδες με "υψηλότερη ποιότητα" πρώτα
- Συνεχής λειτουργία: Να συνεχίζει να φέρνει φρέσκα αντίγραφα μιας σελίδας που έφερε στο παρελθόν
- Να είναι επεκτάσιμος: Να προσαρμόζεται σε νέα data formats, protocols

Ενημερωμένη αναπαράσταση crawling



URL frontier

- Μπορεί να περιλάβει πολλές σελίδες από τον ίδιο host
- Πρέπει ν’ αποφεύγει να προσπαθεί να τα φέρει όλα μαζί την ίδια χρονική στιγμή
- Πρέπει να προσπαθεί να κρατάει όλα τα crawling threads απασχολημένα

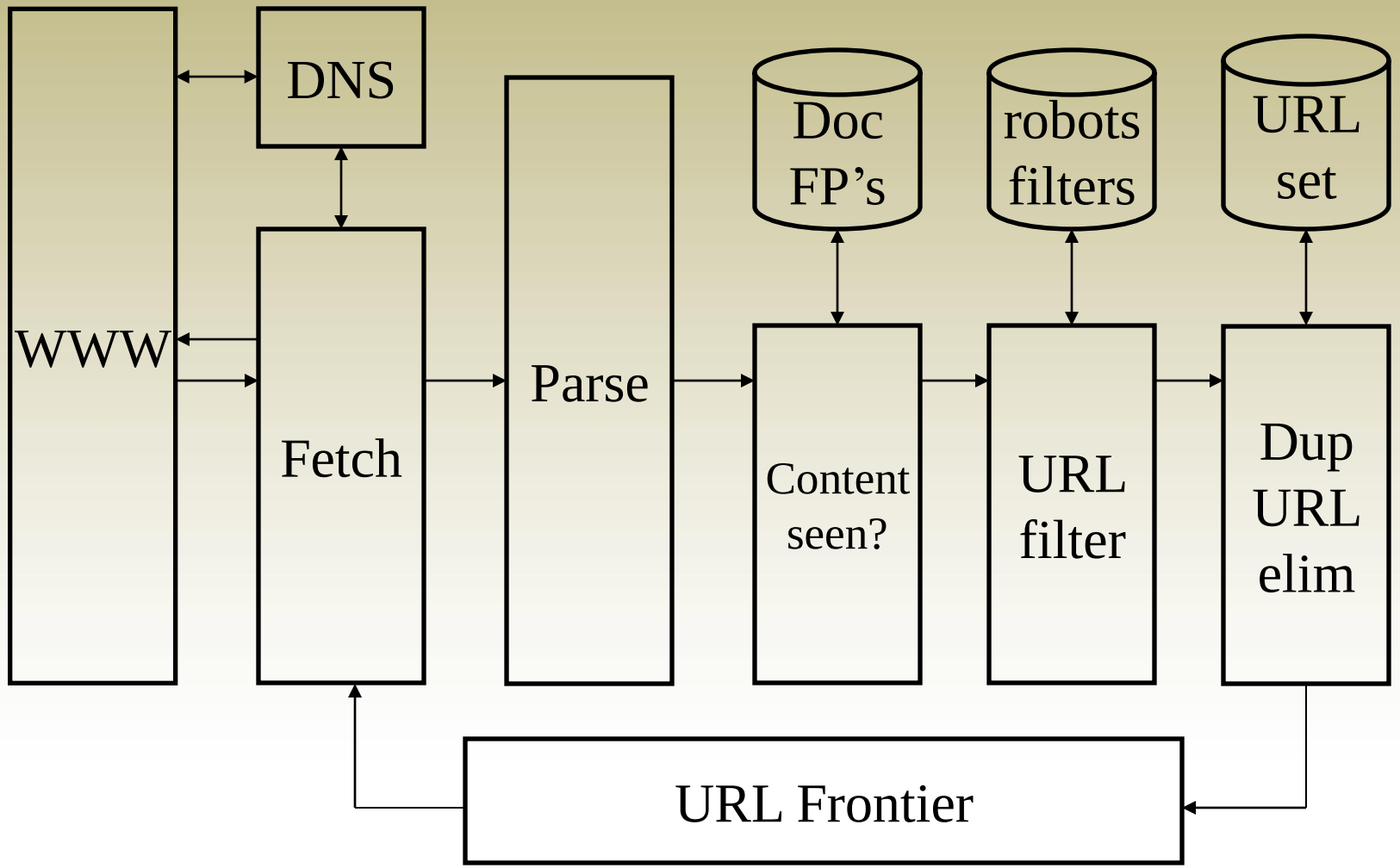
Βήματα επεξεργασίας στο crawling

- Επιλογή ενός URL από το frontier
- Κατέβασμα του εγγράφου που αντιστοιχεί στο URL
- Parse το URL
  - Εξαγωγή συνδέσμων του προς άλλα docs (URLs)
- Έλεγχος εάν το περιεχόμενου του URL το έχουμε καναδεί
  - Εάν όχι, προσθήκη του στους indexes
- Για κάθε εξηγμένο URL
  - Επιβεβαίωση ότι περνά κάποιους ελέγχους για URLs
  - Έλεγχος εάν είναι ήδη στο frontier (duplicate URL elimination)

Ποιο;

Π.χ., μόνο crawling .edu, obey robots.txt, etc.

Βασική αρχιτεκτονική crawling



DNS (Domain Name Server)

- Υπηρεσία lookup στο Διαδίκτυο
  - Δεδομένου ενός URL, ανάκτηση της IP address του
- Συνήθεις υλοποιήσεις από OS της DNS lookup είναι *blocking*: μόνο μια εκκρεμής αίτηση κάθε φορά
- Λύσεις
  - DNS caching
  - Batch DNS resolver – συλλέγει αιτήματα και τα στέλνει ομαδικά

Parsing: URL normalization

- Όταν ένα fetched document γίνεται parsed, κάποιοι από τους συνδέσμους είναι *relative* URLs
- E.g., http://en.wikipedia.org/wiki/Main_Page έχει relative link προς /wiki/Wikipedia:General_disclaimer που είναι το ίδιο με το απόλυτο absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer
- Κατά το parsing, πρέπει να κάνουμε normalize (expand) κάθε relative URL

Περιεχόμενο που έχουμε ήδη δει

- Duplication είναι εκτεταμένο στοWeb
- Εάν η σελίδα που μόλις κατέβηκε είναι ήδη στον index, να μην την επεξεργαστούμε περαιτέρω
- Αυτό γίνεται με fingerprints ή shingles

Filters και robots.txt

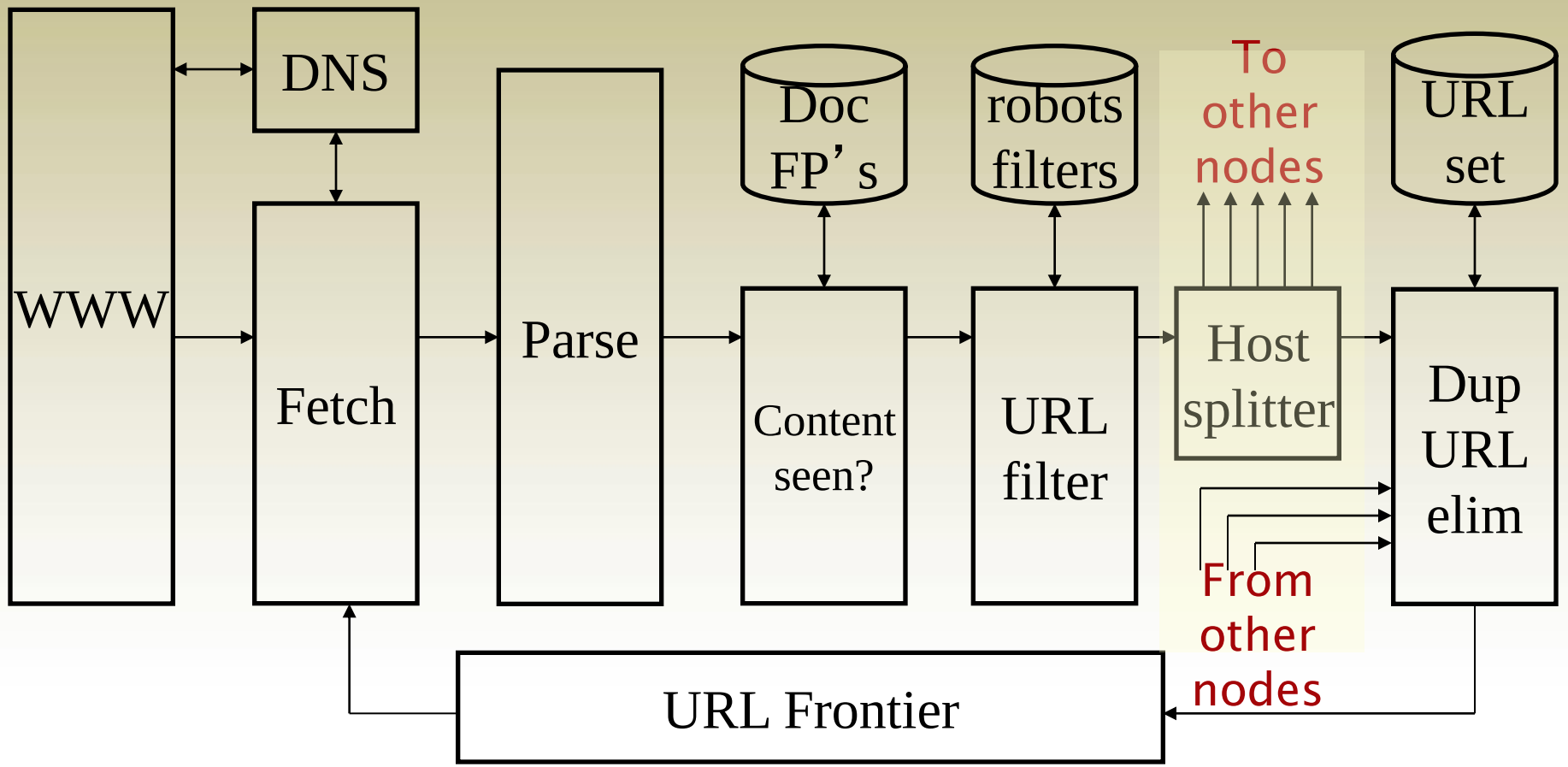
- Filters – regular expressions για URLs που θα γίνουν (ή όχι) crawled
- Αφού κατεβάσουμε το αρχείο robots.txt από ένα site, δεν χρειάζεται να το ξαναφέρουμε
- Cache των αρχείων robots.txt

Κατανεμημένος crawler

- Εκτέλεση πολλαπλών crawl threads, κάτω από διαφορετικές processes – εν δυνάμει σε διαφορετικούς κόμβους
  - Γεωγραφικά κατανημένους κόμβους
- Καταμερισμός των hosts που θα γίνουν crawled στους κόμβους
  - Χρησιμοποιείται hashing για τον καταμερισμό
- Πώς επικοινωνούν αυτοί οι κόμβοι και μοιράζονται τα URLs;

Επικοινωνία μεταξύ κόμβων

- Η έξοδος από κάθε URL filter σε κάθε κόμβο στέλνεται στο Dup URL Eliminator του κατάλληλου κόμβου



URL frontier: Δυο σημαντικά σημεία

- Politeness: Όχι συχνές επισκέψεις στον ίδιο Web server
- Freshness: crawl μερικές σελίδες πιο συχνά από κάποιες άλλες
  - Π.χ., σελίδες (όπως των News sites) των οποίων το περιεχόμενο αλλάζει πιο συχνά

Αυτοί οι στόχοι μπορεί να είναι αλληλοσυγκρουόμενοι

Προκλήσεις στην υλοποίηση ενός ερπυστή

- Ποιες σελίδες θα πρέπει να φέρει;
- Πώς θα πρέπει να ανανεώνει (ξαναεπισκέπτεται) τις σελίδες;
- Πώς θα ελαττώσει το φόρτο στα sites που επισκέπτεται;
- Πώς θα πρέπει να παραλληλοποιηθεί ο ερπυσμός;

Κριτήρια επιλογής σελίδων

- Μετρικές σημαντικότητας
  - Ομοιότητα προς υποβληθέν ερώτημα IS(p) (γνωστοί ως *Focused* ή *Topical* crawlers)
    - Με τις τεχνικές που είδαμε σε προηγούμενες διαλέξεις
  - Αριθμός εισερχόμενων συνδέσμων IB(p) (backlink count)
    - Εκτίμηση του αριθμού με βάση των γνωστών μέχρι στιγμής backlinks
  - PageRank IR(p)
    - Θα το εξερευνήσουμε αναλυτικότατα στις επόμενες διαλέξεις
  - Αριθμός εξερχόμενων συνδέσμων IF(p) (forward link count)
    - Ίσως πρόκειται για καλό directory (hub page)
  - Μετρική τοποθεσίας IL(p)
    - Οι .com (πιο χρήσιμες) ή πολλά slashes (λιγότερο χρήσιμες)
  - Συνδυαστική μετρική: $IC(p)= k_1*IS(p) + k_2*IB(p) + \ldots.$

Βασικός αλγόριθμος crawling

Algorithm 2.5.1 Crawling algorithm (backlink based)
Input: `starting_url`: seed URL
Procedure:

```
[1] enqueue(url_queue, starting_url)
[2] while (not empty(url_queue))
[3]   url = dequeue(url_queue)
[4]   page = crawl_page(url)
[5]   enqueue(crawled_pages, (url, page))
[6]   url_list = extract_urls(page)
[7]   foreach u in url_list
[8]     enqueue(links, (url, u))
[9]     if (u∉url_queue and (u,-)∉crawled_pages)
[10]      enqueue(url_queue, u)
[11]  reorder_queue(url_queue)
```

Function description:

`enqueue(queue, element)`: append `element` at the end of `queue`
`dequeue(queue)`: remove the element at the beginning of `queue` and return it
`reorder_queue(queue)`: reorder `queue` using information in links (refer to Figure 2.2)

reorder queue() για κάθε κριτήριο

(1) breadth first

```
do nothing (null operation)
```

(2) backlink count, $IB'(p)$

```
foreach u in url_queue
   backlink_count[u] = number of terms (-,u) in links
sort url_queue by backlink_count[u]
```

(3) PageRank $IR'(p)$

```
solve the following set of equations:
```

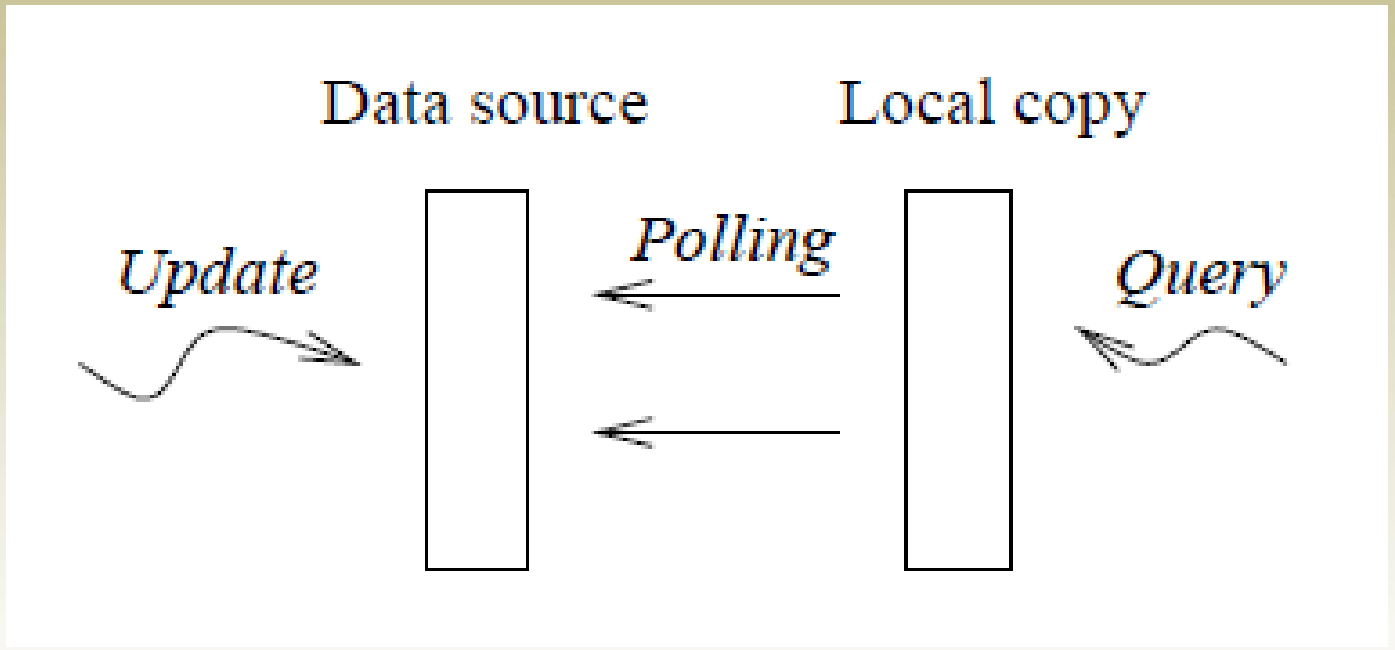
$IR[u] = (1 - 0.9) + 0.9 \sum_i \frac{IR[v_i]}{c_i}$, where

$(v_i, u) \in$ `links` and c_i is the number of links in the page v_i

```
sort url_queue by IR(u)
```

Το πρόβλημα της ανανέωσης σελίδων στον index από τους ερπυστές

Σχηματική απεικόνιση του προβλήματος



Freshness και Age

- Freshness (Φρεσκάδα)
  - Του τοπικού στοιχείου την χρονική στιγμή t

$$F(e_i;t) = \begin{cases} 1 & \text{if } e_i \text{ is up-to-date at time } t \\ 0 & \text{otherwise} \end{cases}$$

  - Του index την χρονική στιγμή t

$$F(S;t) = \frac{1}{N}\sum_{i=1}^{N} F(e_i;t)$$

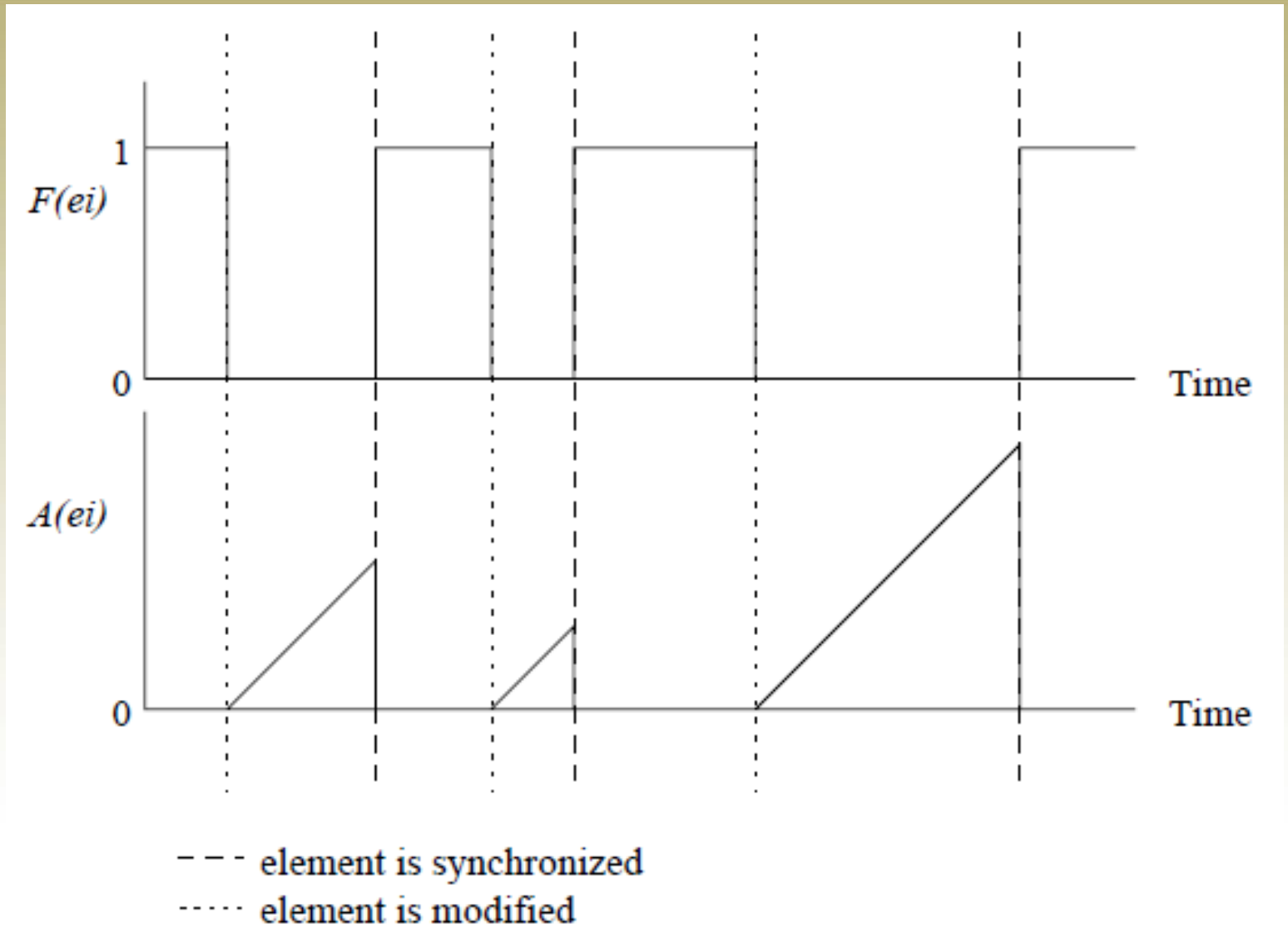
- Age (Ηλικία)
  - Του τοπικού στοιχείου την χρονική στιγμή t

$$A(e_i;t) = \begin{cases} 0 & \text{if } e_i \text{ is up-to-date at time } t \\ t - \text{modification time of } e_i & \text{otherwise} \end{cases}$$

  - Του index την χρονική στιγμή t

$$A(S;t) = \frac{1}{N}\sum_{i=1}^{N} A(e_i;t)$$

Χρονική εξέλιξη των freshness και age



Average freshness και average age

- Average freshness στοιχείου και του index

$$\overline{F}(e_i) = \lim_{t\to\infty} \frac{1}{t} \int_0^t F(e_i;t)dt \qquad \overline{F}(S) = \lim_{t\to\infty} \frac{1}{t} \int_0^t F(S;t)dt$$

- Average age στοιχείου και του index

$$\overline{A}(e_i) = \lim_{t\to\infty} \frac{1}{t} \int_0^t A(e_i;t)dt \qquad \overline{A}(S) = \lim_{t\to\infty} \frac{1}{t} \int_0^t A(S;t)dt$$

- Ισχύουν (απόδειξη στην επόμενη διαφάνεια) οι εξής σχέσεις:

$$\overline{F}(S) = \frac{1}{N} \sum_{i=1}^{N} \overline{F}(e_i) \qquad \overline{A}(S) = \frac{1}{N} \sum_{i=1}^{N} \overline{A}(e_i)$$

Απόδειξη average freshness και age

$$\overline{F}(S) = \lim_{t\to\infty} \frac{1}{t} \int_0^t F(S;t)dt$$

$$= \lim_{t\to\infty} \frac{1}{t} \int_0^t \Big(\frac{1}{N} \sum_{i=1}^{N} F(e_i;t)\Big)dt$$

$$= \frac{1}{N} \sum_{i=1}^{N} \lim_{t\to\infty} \frac{1}{t} \int_0^t F(e_i;t)dt$$

$$= \frac{1}{N} \sum_{i=1}^{N} \overline{F}(e_i)$$

$$\overline{A}(S) = \lim_{t\to\infty} \frac{1}{t} \int_0^t A(S;t)dt$$

$$= \lim_{t\to\infty} \frac{1}{t} \int_0^t \Big(\frac{1}{N} \sum_{i=1}^{N} A(e_i;t)\Big)dt$$

$$= \frac{1}{N} \sum_{i=1}^{N} \lim_{t\to\infty} \frac{1}{t} \int_0^t A(e_i;t)dt$$

$$= \frac{1}{N} \sum_{i=1}^{N} \overline{A}(e_i)$$

Πιθανοκρατική ανανέωση σελίδας

- Βρέθηκε (με ανάλυση των crawled σελίδων) ότι ο ρυθμός αφίξεων updates για ένα στοιχείο (σελίδα) του Web είναι Poisson process με ρυθμό αλλαγής λ_i, άρα ο χρόνος μέχρι το νέο γεγονός (δηλαδή, μια τροποποίηση) είναι εκθετικά κατανεμημένος
- "Συγχρονίζοντας" την σελίδα στην χρονική στιγμή t=0 και t=I, η expected value της freshness στον χρόνο t, η πιθανότητα ότι το συγκεκριμένο στοιχείο αλλάζει στο διάστημα (0,t] υπολογίζεται ως εξής:

$$Pr\{T \le t\} = \int_0^t f_T(t)dt = \int_0^t \lambda_i e^{-\lambda_i t} dt = \lambda_i \frac{-1}{\lambda_i} e^{-\lambda_i t}\Big|_0^t = e^0 - e^{-\lambda_i t} = 1 - e^{-\lambda_i t}$$

Expected freshness με Poisson αλλαγές

- Αφού το στοιχείο δεν συγχρονίζεται στο διάστημα $(0,I)$,

  θα είναι παροχημένο με πιθανότητα:

$$Pr\{T \leq t\} = 1 - e^{-\lambda_i t}$$

  άρα η expected freshness για $t \in (0,I)$ θα είναι:

$$E[F(e_i;t)] = 0 \times (1 - e^{-\lambda_i t}) + 1 \times e^{-\lambda_i t} = e^{-\lambda_i t}$$

Expected age με Poisson αλλαγές

- Αφού το στοιχείο τροποποιείται στο $s \in (0,I)$, η ηλικία του στον χρόνο $t \in (s,I)$ θα είναι t-s
- Αφού αλλάξει στον χρόνο s με πιθανότητα

$$\lambda_i e^{-\lambda_i s}$$

- και άρα (απόδειξη στην επόμενη διαφάνεια) η expected age θα είναι, για t στο $(0,I)$:

$$E[A(e_i;t)] = \int_0^t (t-s)(\lambda_i e^{-\lambda_i s})ds = t\left(1 - \frac{1-e^{-\lambda_i t}}{\lambda_i t}\right)$$

Υπόδειξη για τον υπολογισμό του ολοκληρώματος:
Η παράγουσα του $s*e^{-\lambda*s}$ είναι η $-[(\lambda*s+1)/\lambda^2]*e^{-\lambda*s}$

Αναλυτική εύρεση της expected age

$$E[A(e_i;t)] = \int_0^t (t-s)(\lambda_i e^{-\lambda_i s})ds = \int_0^t t\lambda_i e^{-\lambda_i s}ds - \int_0^t s\lambda_i e^{-\lambda_i s}ds$$

$$= t\lambda_i \int_0^t e^{-\lambda_i s}ds - \lambda_i \int_0^t s e^{-\lambda_i s}ds$$

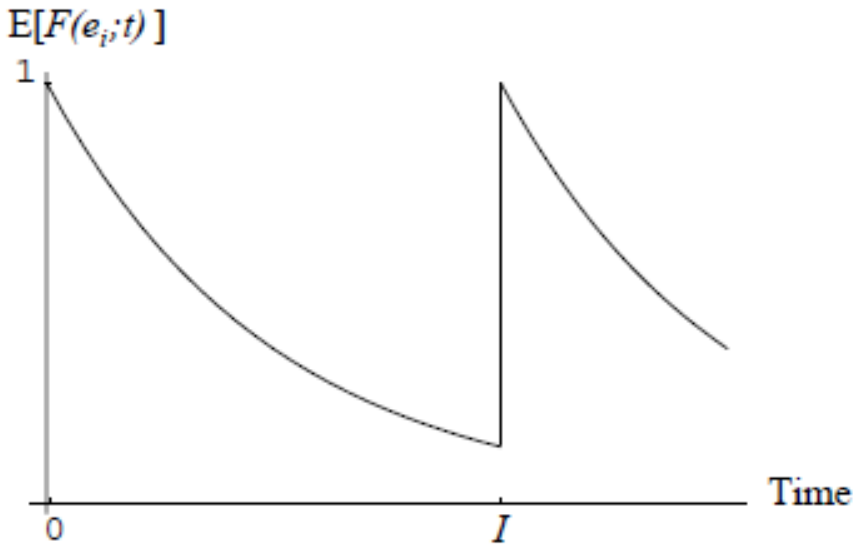
$$= t\lambda_i \frac{e^{-\lambda_i s}}{-\lambda_i}\Big|_0^t + \lambda_i \frac{\lambda_i s + 1}{\lambda_i^2} e^{-\lambda_i s}\Big|_0^t$$

$$= t(1 - e^{-\lambda_i t}) + \frac{1}{\lambda_i}\Big[(\lambda_i t + 1)e^{-\lambda_i t} - (1)\Big]$$

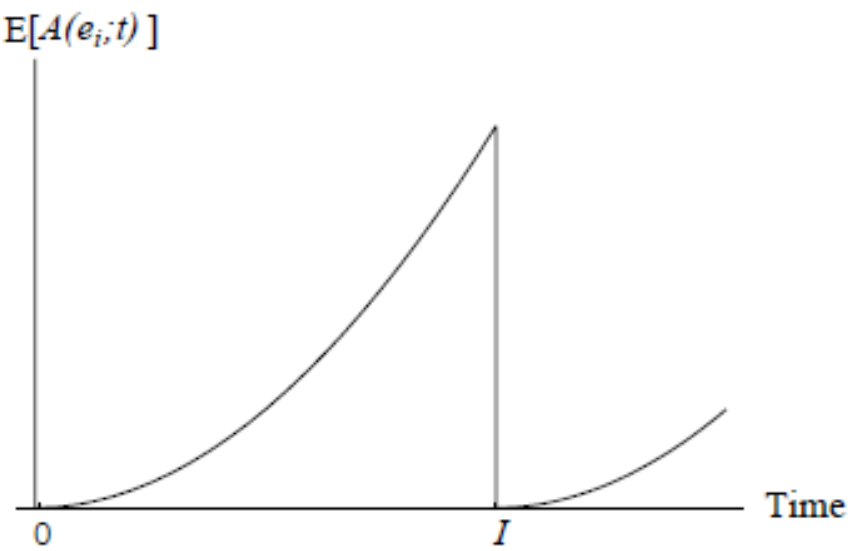
$$= t(1 - e^{-\lambda_i t}) + t\Big(e^{-\lambda_i t} + \frac{e^{-\lambda_i t}}{\lambda_i t} - \frac{1}{\lambda_i t}\Big)$$

$$= t\Big(1 - \frac{1 - e^{-\lambda_i t}}{\lambda_i t}\Big)$$

Χρονική εξέλιξη των expected freshness & expected age



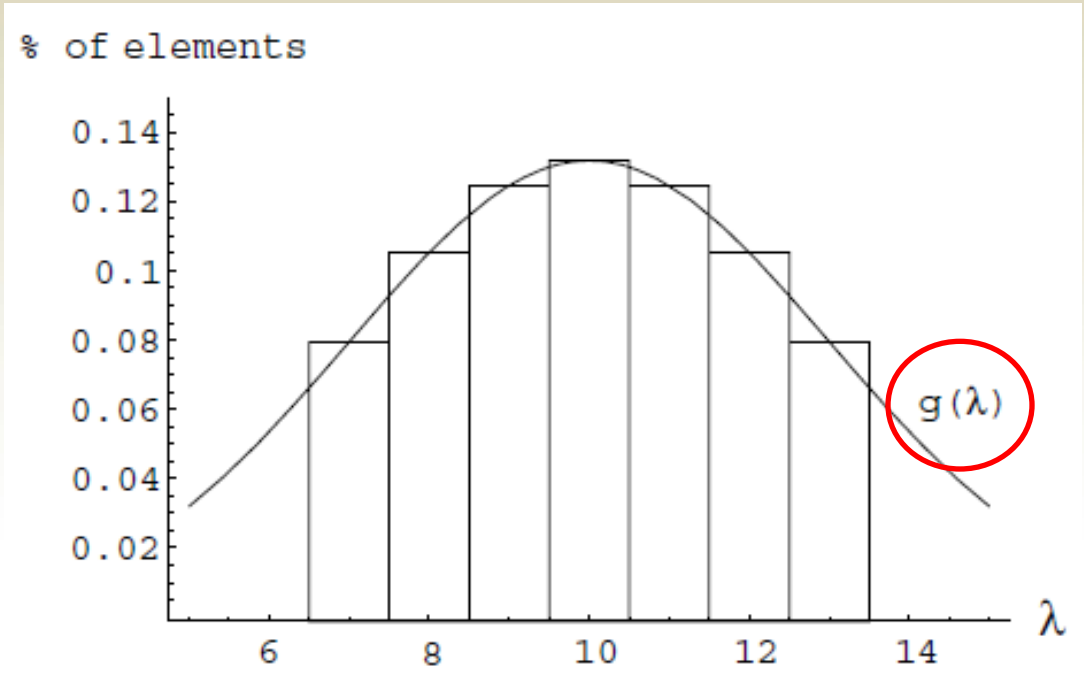
(a) $\mathrm{E}[F(e_i;t)]$ graph over time



(b) $\mathrm{E}[A(e_i;t)]$ graph over time

Μοντέλο εξέλιξης του Web

- Μοντέλο uniform change-frequency
  - Όλα τα στοιχεία αλλάζουν με την ίδια συχνότητα λ
- Μοντέλο non-uniform change-frequency
  - Κάθε στοιχείο αλλάζει με την δική του συχνότητα λ_i



Παράρτημα: Poisson και exponential

- Η Poisson παρέχει περιγραφή του αριθμού των εμφανίσεων ενός γεγονός στην μονάδα του χρόνου, και η exponential παρέχει την περιγραφή του μήκους του χρόνου μεταξύ διαδοχικών εμφανίσεων του γεγονότος
- Σε μια Poisson διαδικασία, εάν τα γεγονότα προκύπτουν κατά μέσο όρο με ρυθμό λ στην μονάδα του χρόνου, τότε θα έχουμε λt εμφανίσεις ανά t χρονικές μονάδες. Η κατανομή Poisson που περιγράφει αυτήν την διαδικασία είναι η εξής:

$$\boldsymbol{P(x) = e^{-\lambda t} (\lambda t)^x / x!}$$

από την οποία προκύπτει ότι $P(x=0) = e^{-\lambda t} (\lambda t)^0/0! = e^{-\lambda t}\ 1/1 = e^{-\lambda t}$ είναι η πιθανότητα μη εμφάνισης γεγονότος σε t χρονικές μονάδες

- Μια άλλη ερμηνεία του $P(x=0) = e^{-\lambda t}$ είναι ως η πιθανότητα ότι ο χρόνος T της πρώτης εμφάνισης γεγονότος είναι μεγαλύτερος από t, δηλαδή

$$P(T > t) = P(x=0 \mid \mu=\lambda t) = e^{-\lambda t}$$

- Αντίστροφα, η πιθανότητα ότι κάποιο γεγονός προκύπτει μέσα σε t χρονικές μονάδες δίνεται από την κάτωθι σχέση:

$$P(T \leq t) = 1 - P(x=0 \mid \mu=\lambda t) = 1 - e^{-\lambda t}$$

- Επισημαίνουμε ότι αυτή είναι η c.d.f., η οποία όταν παραγωγιστεί ως προς t μάς γίνει την p.d.f. της exponential κατανομής: $f(t) = \lambda e^{-\lambda t}$