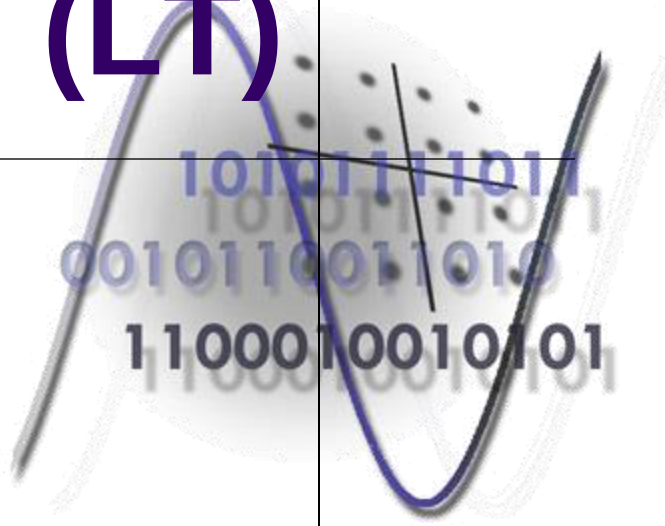
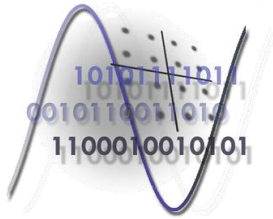
Codurile Luby-Transform (LT)

TACCFDRT Curs 2



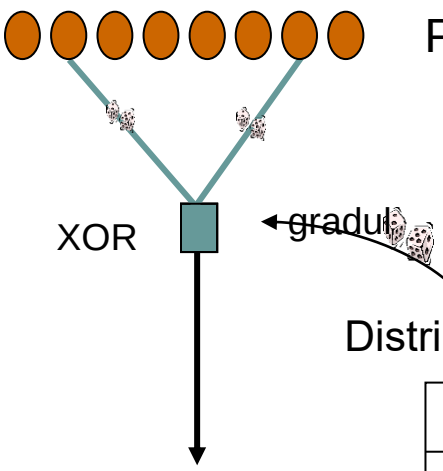
Codurile Luby-Transform (LT)



- Codurile LT reprezintă prima realizare practică a codurilor rateless; nu trebuie definită o rată fixă înainte de codare
- Se poate construi un flux infinit de pachete codate din pachetele informaționale

Principiul de codarea LT [3]

- Pentru pachetul codat care urmează să fie generat se alege aleator un grad *d*, conform unei distribuții predefinite
- Se aleg aleator *d* pachete din mulțimea pachetelor informaționale
- Pachetul codat se obține adunând modulo doi pachete selecționate la pasul anterior



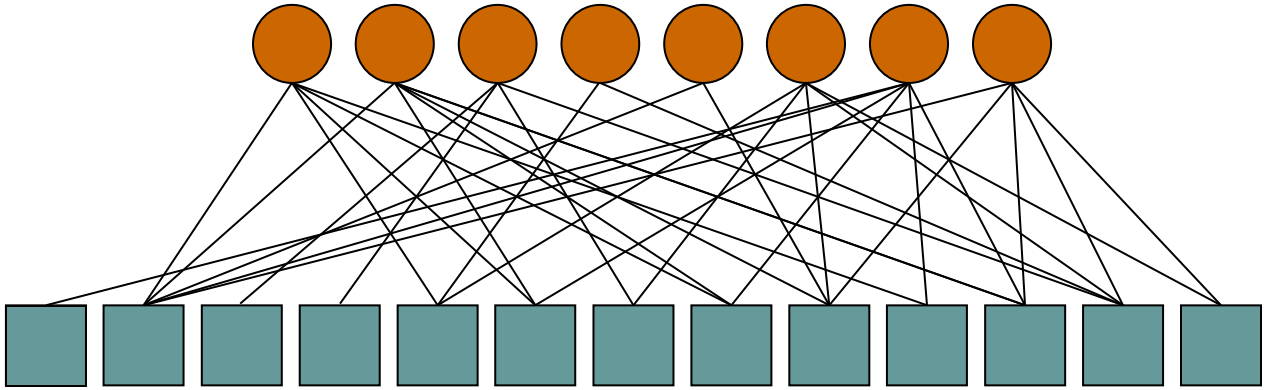
Distribuția gradurilor

grad	Prob
1	0.055
2	0.3
3	0.1
4	0.08
⋮	⋮
100000	0.0004

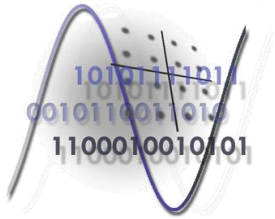
[3] Michael Luby; "LT Codes" -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; "Fountain Codes" EPFL and Digital Fountain, Inc.

Graful asociat codului

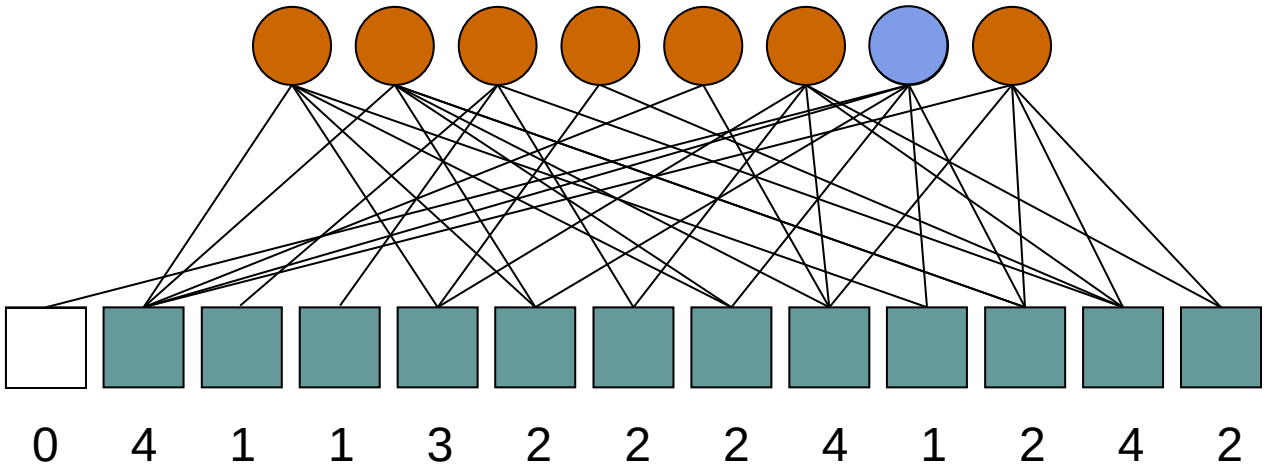


Principiul de decodare



- Regula de decodare
  - Dacă există minim un simbol codat care are un singur vecin, atunci valoarea vecinului respectiv este o copie a simbolului codat.
  - Valoarea simbolului de intrare recuperat va fi adunat modulo 2 cu toate simbolurile codate rămase care au ca și vecin acel simbol de intrare
  - gradul simbolurilor codate, la care a fost adunat simbolul informațional recuperat este redus cu unu, și acest simbol de intrare este eliminat din lista vecinilor

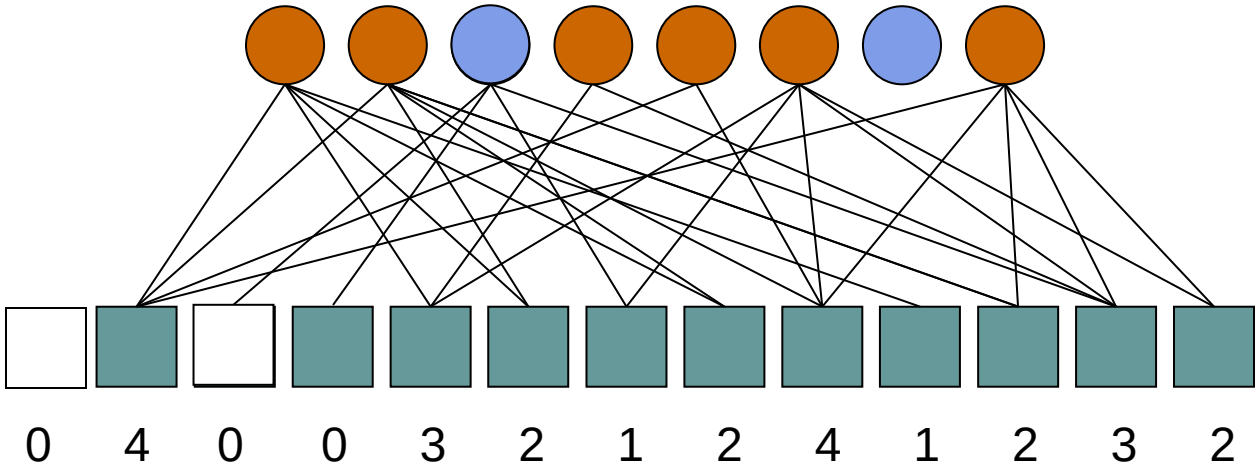
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

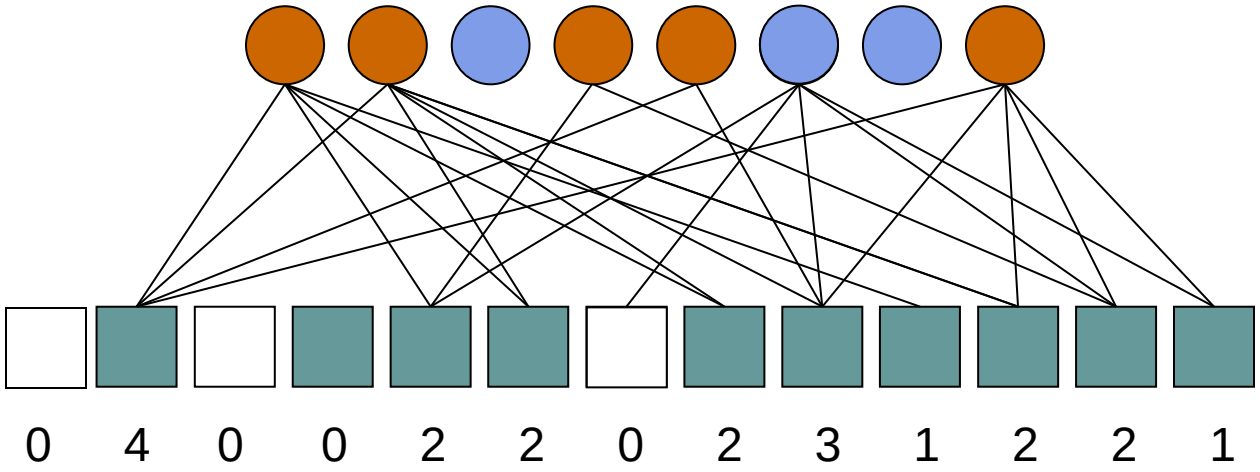
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

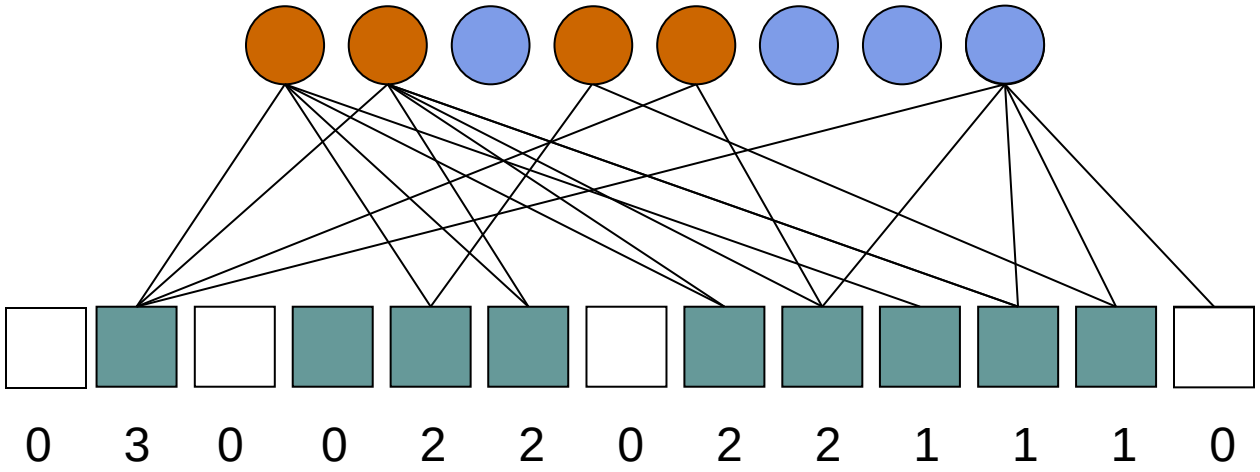
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

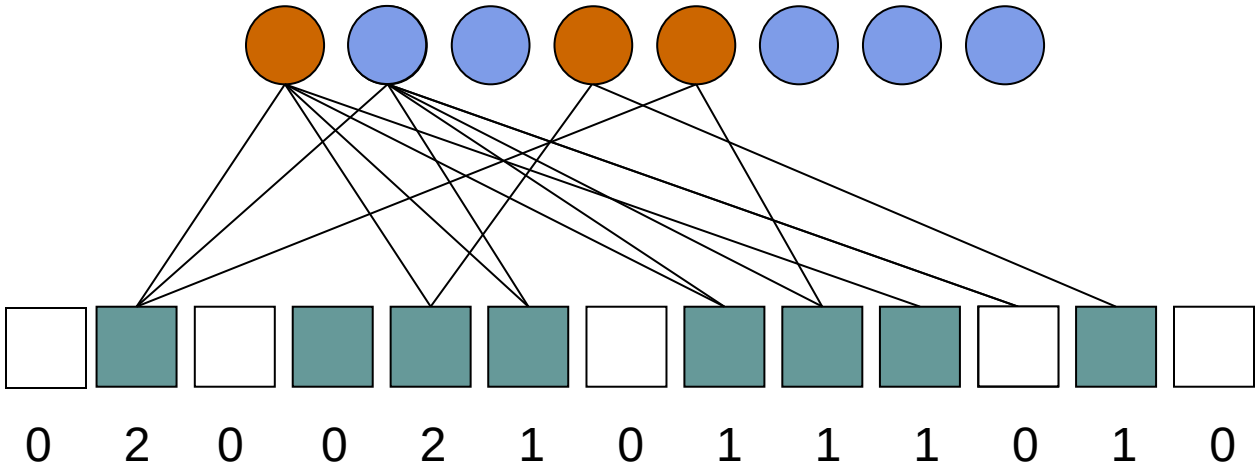
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

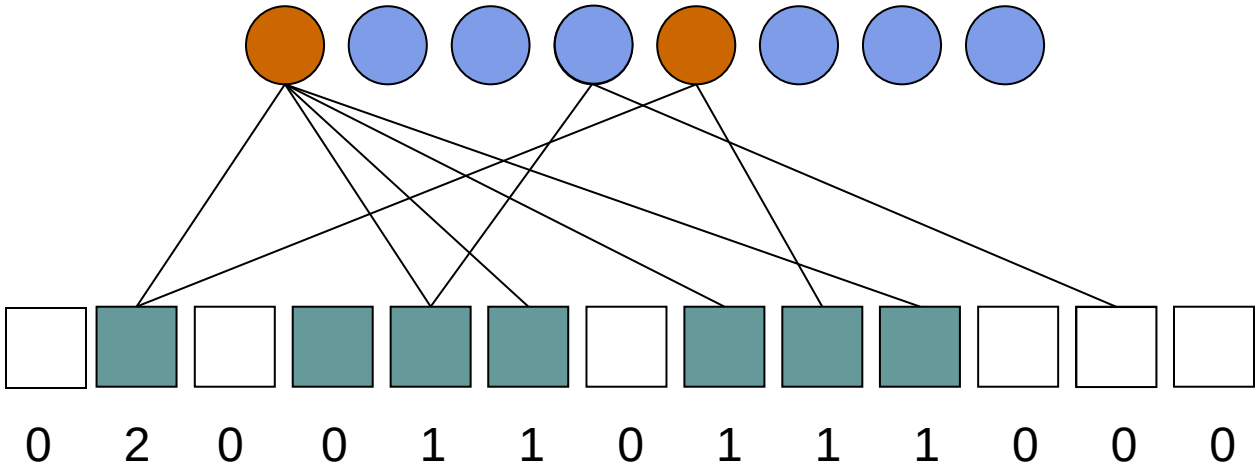
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

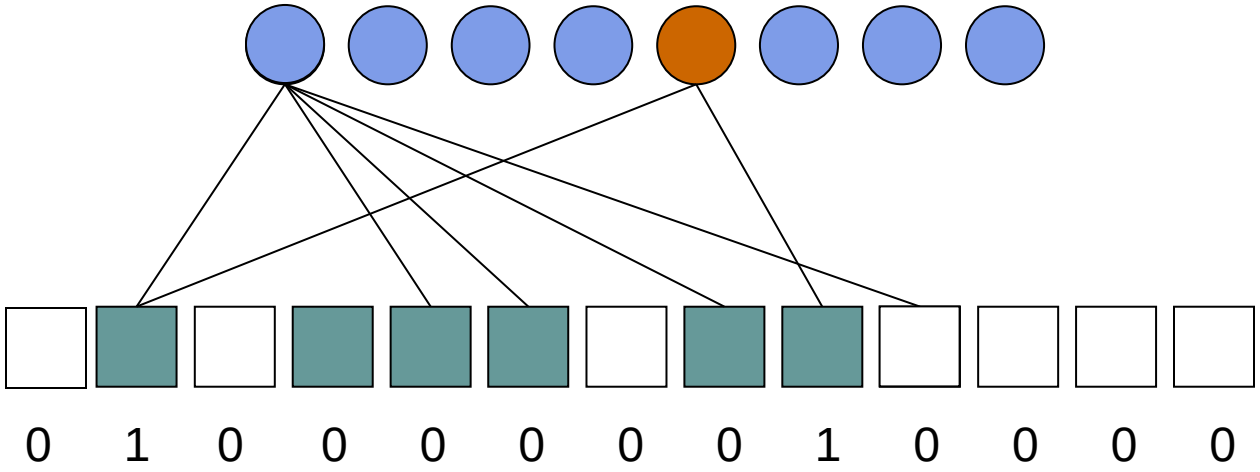
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

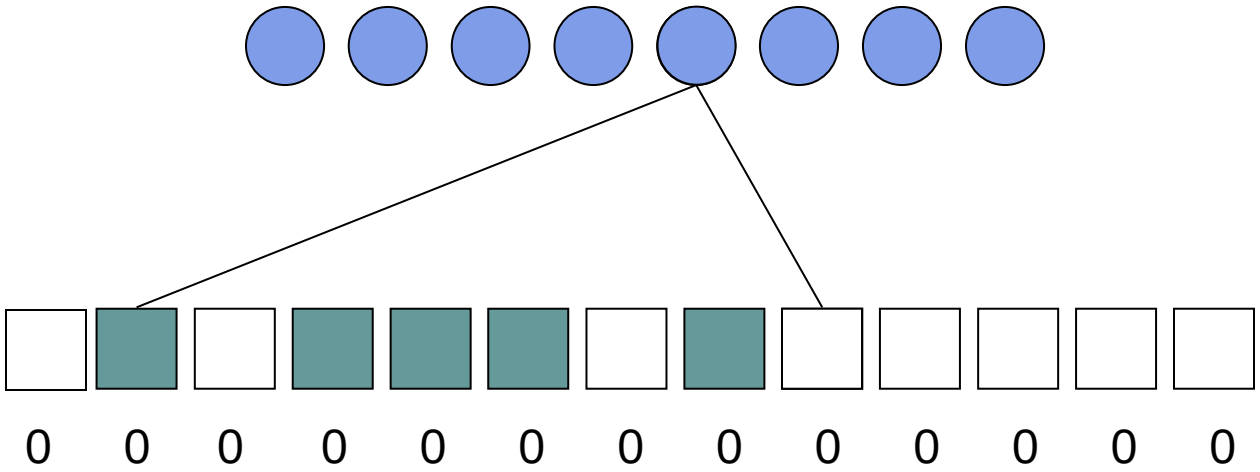
Decodare LT



[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

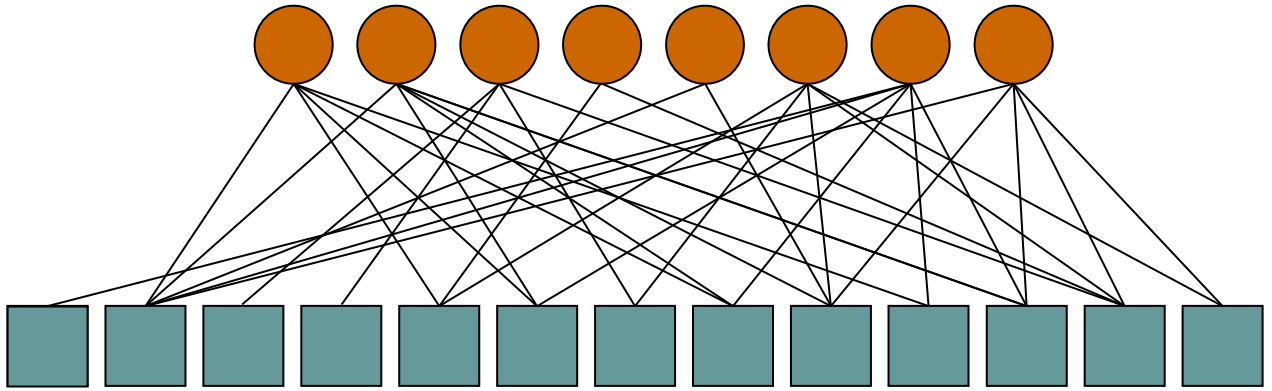
Decodare LT



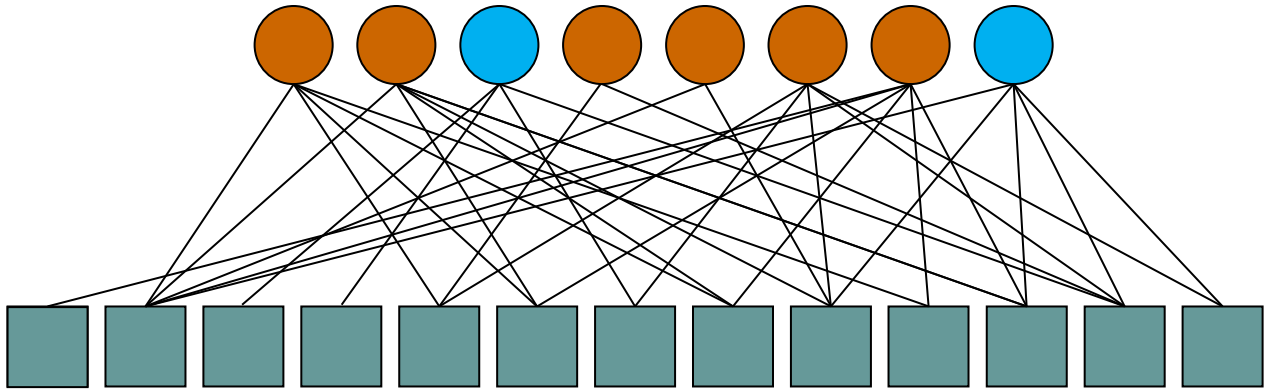
[3] Michael Luby; “LT Codes” -*Digital Fountain, Inc.*, Fremont, 2002

[4] Amin Shokrollahi; “Fountain Codes” EPFL and Digital Fountain, Inc.

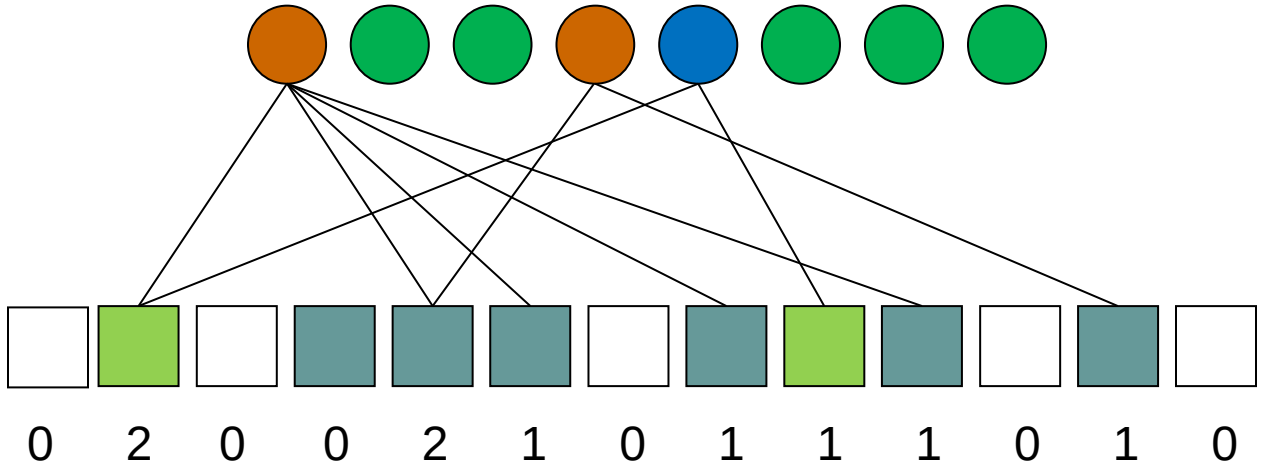
- Pt studierea distribuțiilor se reformulează procesul de decodare:
  - La început toate simbolurile informaționale sunt descoperite



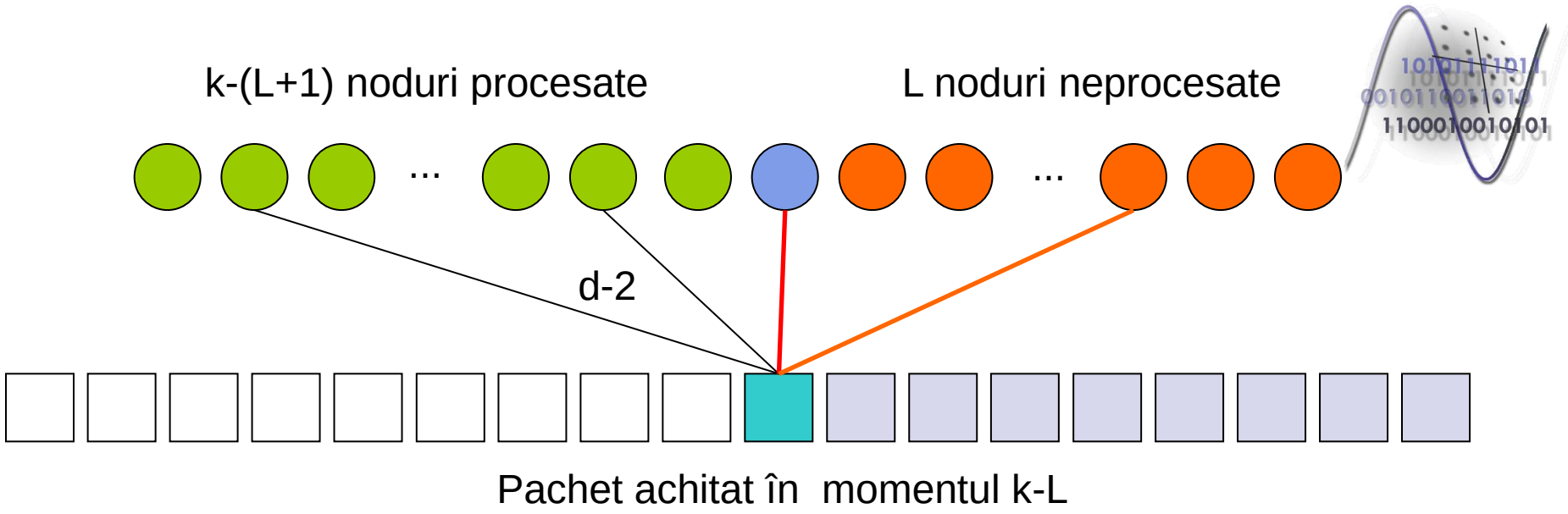
- În prima etapă toate simbolurile codate care sunt formate dintr-un singur simbol informațional, “acoperă” singurul lor vecin. Mulțimea formată din simbolurile acoperite, care încă nu au fost procesate, se numește “riplu”



- În pașii următor este luat câte un simbol informațional din riplu, este adunat la simbolurile codate la care este vecin și se reduce gradul acestor simboluri.
- Dacă un simbol codat astfel va avea grad 1, acest simbol va acoperi vecinul său, iar acest simbol codat astfel va fi "***achitat***" Dacă acest vecin acoperit nu a fost acoperit mai devreme, de un alt simbol codat, atunci dimensiunea riplu-lui crește.



- Procesul se termină când riplul se golește
  - Procesul de decodare este cu succes dacă la golirea riplului nu mai sunt simboluri de intrare neacoperite.



- Probabilitatea *q(d,L)* ca un pachet cu gradul inițial d să fie achitat când mai sunt L pachete informaționale neprocesate este:

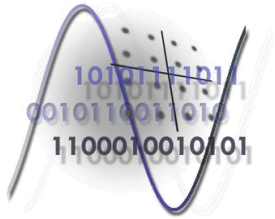
$$q(1,k)=1$$

$$\text{pentru} \quad d=2,\ldots,k \; si \; L=k-d+1,\ldots,1$$

$$q(d,L)=\frac{\binom{k-(L+1)}{d-2}\cdot\binom{L}{1}}{\binom{k-1}{d}}=\frac{d(d-1)\cdot L\prod_{j=0}^{d-3}k-(L+1)-j}{\prod_{j=0}^{d-1}k-j}$$

$$\text{pentru} \quad celelalte \; valori \; d \; si \; L$$

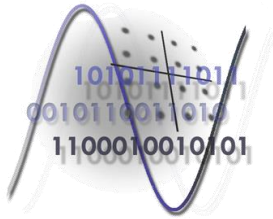
$$q(d,L)=0;$$



- Probabilitatea r(d,L) este probabilitatea ca un simbol codat să aibă gradul d, și să fie achitat când mai sunt L simboluri de intrare neprocesate
  - r(d,L)=p(d)q(d,L)
- Probabilitatea r(L) ca un pachet să fie achitat când mai sunt L pachete informaționale neprocesate este:

$$r(L)=\sum_{d} r(d,L)=\sum_{d=2}^{k} p(d)\frac{d(d-1)\cdot L\prod_{j=0}^{d-3} k-(L+1)-j}{\prod_{j=0}^{d-1} k-j}$$

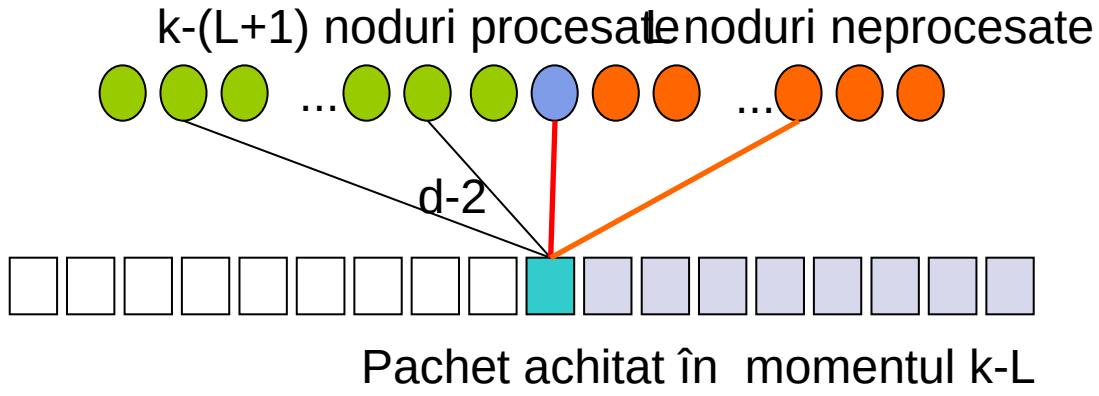
Distribuția *Soliton* ideală



- Cerințele impuse unei distribuții de graduri sunt:
  - Un număr mediu de simboluri codate cât mai mic posibil pentru a asigura succesul procesului LT.
  - Gradul mediu al simbolurilor cât mai mic posibil. Gradul mediu definește numărul operațiilor de simbol necesare pentru generarea unui simbol codat, iar k*(grad mediu) este numărul operațiilor necesare pentru recuperarea completă a datelor.

Distribuția *Soliton* ideală

- O proprietate elementară a unei distribuții "ideale" este ca la procesul de decodare, la procesarea unui simbol informațional la riplu să fie adăugat un simbol acoperit.
- Asta asigură că dimensiunea riplului să nu fie niciodată prea mică sau prea mare.
- r(L) –este probabilitatea ca la riplu să fie adunat un singur simbol la procesarea simbolului k-(L+1)



Distribuția *Soliton* ideală

- Ținând cont că:

$$r(L)=\sum_{d} r(d,L)=\sum_{d=2}^{k} p(d)\frac{d(d-1)\cdot L\prod_{j=0}^{d-3} k-(L+1)-j}{\prod_{j=0}^{d-1} k-j}$$

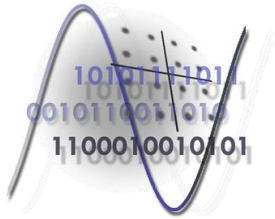
și

$$\sum_{d=2}^{k}\frac{L\prod_{j=0}^{d-3} k-(L+1)-j}{\prod_{j=0}^{d-1} k-j}=1 \quad pt\ orice\ L>1$$

Rezultă distribuția Soliton ideală:

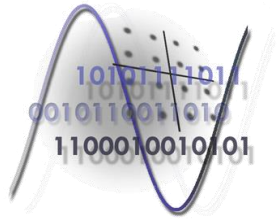
$$p(d)=\begin{cases}\frac{1}{k}; & pt\ d=1\\ \frac{1}{d(d-1)}; & pt\ d=2,\ldots,k\end{cases}$$

Distribuția *Soliton* ideală



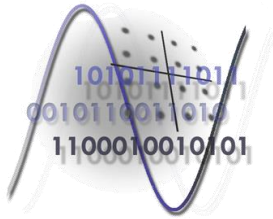
- Sunt necesare exact k simboluri codate pentru reconstruirea celor k simboluri de intrare
- Dimensiunea riplului este 1 pe toată durata decodări
- PERFORMAŢE FOARTE SLABE în practică
  - Deoarece riplul este foarte scurt, riplul poate să se golească înaintea decodării tuturor mesajelor informaționale

Distribuția Soliton Robust



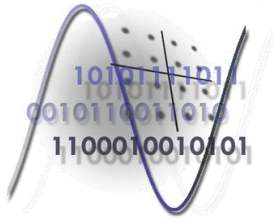
- Cu cât dimensiunea riplului este mai mare cu atât probabilitatea ca riplul să se golească înaintea recuperării tuturor simbolurilor informaționale este mai mică
- Pentru a minimiza numărul total de simbolul codate utilizate la recuperarea simbolurilor informaționale dimensiunea riplului trebuie să fie cât mai mic posibil

Distribuția Soliton Robust



- Cât trebuie să fie suma gradurilor pachetelor recepționate ca fiecare pachet informațional să fie acoperit cu probabilitate 1-δ?
  - Problema este echivalentă cu problema “clasică” coșuri și mingi:
    - avem k coșuri
    - se aruncă K mingi, fiecare minge intră într-unul dintre coșuri
  - Câte mingi trebuie aruncate ca probabilitatea ca în fiecare coș să intre cel puțin o minge să fie 1-δ

Distribuția Soliton Robust



- Câte mingi trebuie aruncate ca probabilitatea ca în fiecare coș să intre cel puțin o minge să fie 1-δ
- Probabilitatea ca o minge să intre în coșul i este 1/k
- Probabilitatea ca după N încercări să existe un coș gol este:

$$k\left(1-\frac{1}{k}\right)^{N}$$

- Probabilitatea ca după N încercări să existe un coș gol trebuie să fie δ

$$k\left(1-\frac{1}{k}\right)^{N}=\delta$$

- Logaritmând ecuația obținem:

$$\ln\left(1-\frac{1}{k}\right)^{N}=\ln\left(\frac{\delta}{k}\right)$$

Distribuția Soliton Robust

$$\ln\left(1-\frac{1}{k}\right)^{N}=\ln\left(\frac{\delta}{k}\right)$$

- Dar....

$$N\ln\left(1-\frac{1}{k}\right)^{\frac{k}{k}}=\ln\left(\frac{\delta}{k}\right)$$

- Adică....

$$N\frac{1}{k}\ln\left(1-\frac{1}{k}\right)^{k}=\ln\left(\frac{\delta}{k}\right)$$

- Dacă k este suficient de mare atunci:
- $\left(1-\frac{1}{k}\right)^{k}\approx\frac{1}{e}$ și $N\frac{1}{k}\ln\left(\frac{1}{e}\right)=\ln\left(\frac{\delta}{k}\right)$

Distribuția Soliton Robust

$$N\frac{1}{k}\ln\left(\frac{1}{e}\right)=\ln\left(\frac{\delta}{k}\right)$$

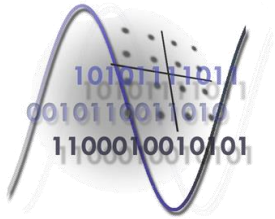
- Dar....

$$N\ln(e)=k\cdot\ln\left(\frac{k}{\delta}\right)$$

- Adică

$$N=k\cdot\ln\left(\frac{k}{\delta}\right)$$

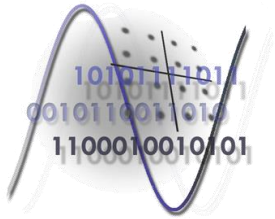
Distribuția Soliton Robust



- Dacă dimensiunea riplului este $\ln\left(\frac{k}{\delta}\right)\sqrt{k}$ atunci probabilitatea de golire a riplului înainte de terminarea decodării este maxim δ

  - Cu procesarea unui nod riplul poate să scadă sau poate să crească cu unu
  - Asta este echivalent cu un “random walk” unidimensional care are lungimea pasului egală cu unu
  - În cazul unui “random walk” de lungime k, o deviație față de valoarea medie mai mare ca $\ln\left(\frac{k}{\delta}\right)\sqrt{k}$ apare cu probabilitatea δ

Distribuția Soliton Robust



- Pentru un compromis se acceptă că din K pachete recepționate, decodorul LT nu reușește să determine informațiile originale cu probabilitatea δ
- pentru asigurarea ca probabilitatea de eroare să fie maxim δ, dimensiunea riplului trebuie să fie:

$$\ln\left(\frac{k}{\delta}\right)\sqrt{k}$$

- Iar numărul pachetelor codate necesare pentru decodare este:

$$K = k + O\left(\ln^2\left(\frac{k}{\delta}\right)\sqrt{k}\right)$$

Distribuția Soliton Robust

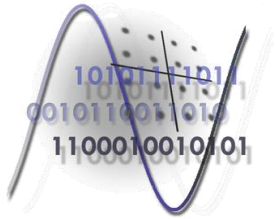
- Se alege lungimea riplului dorit ca fiind:

$$R = c \ln\left(\frac{k}{\delta}\right)\sqrt{k} \quad unde\ c > 0$$

- Se definește τ(d) ca fiind

$$\tau(d) = \begin{cases} \frac{R}{dk} & pt\ d = 1, \ldots, \frac{k}{R} - 1 \\ R\frac{\ln\left(\frac{R}{\delta}\right)}{k} & pt\ d = \frac{k}{R} \\ 0 & pt\ d = \frac{k}{R} + 1, \ldots, k \end{cases}$$

Distribuția Soliton Robust

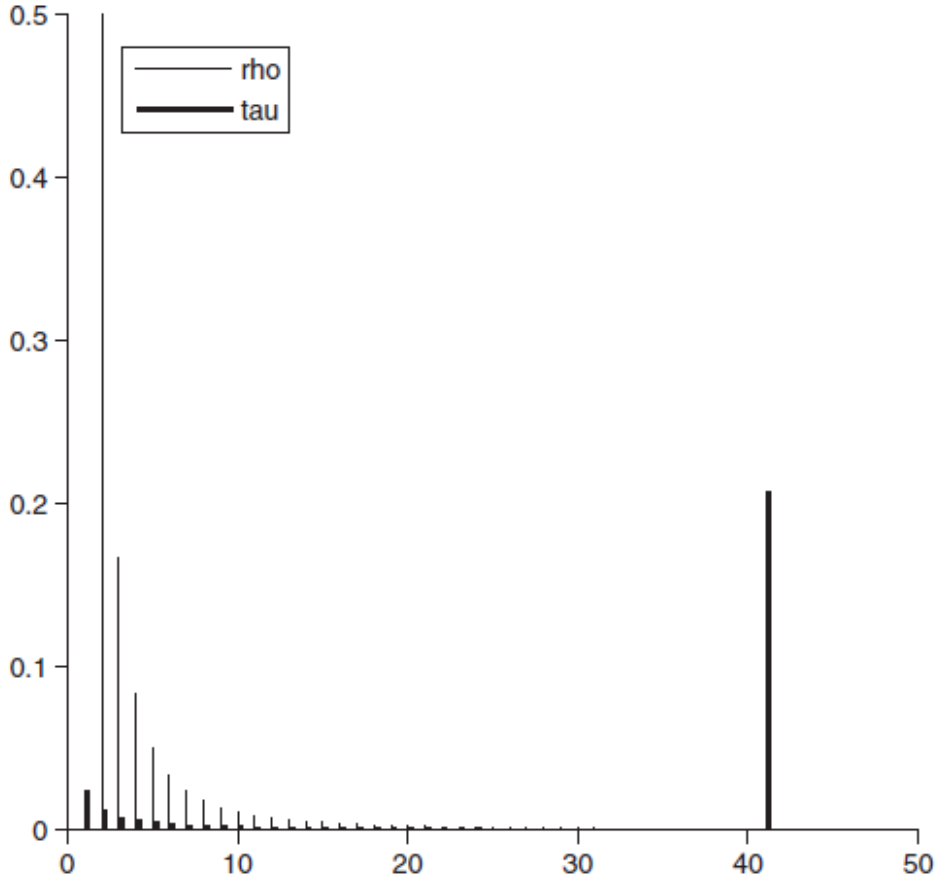


- Se adună distribuția ideală p(d) la τ(d), normalizând această sumă cu β se obține distribuția robustă μ(d)

$$\beta = \sum_{d=1}^{k} p(d) + \tau(d)$$

$$\mu(d) = \frac{p(d) + \tau(d)}{\beta}$$

  - Valoarea medie a gradurilor este c*ln*(k)
  - Ovearheadul necesar este proporțional cu K

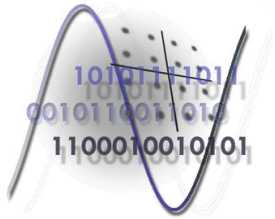


$$p(d)=\begin{cases}\frac{1}{k}; & pt\ d=1\\ \frac{1}{d(d-1)}; & pt\ d=2,\ldots,k\end{cases}$$

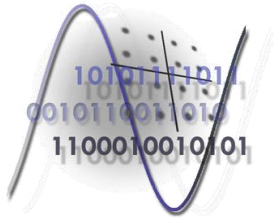
$$\tau(d)=\begin{cases}\frac{R}{dk} & pt\ d=1,\ldots,\frac{k}{R}-1\\ R\frac{\ln\left(\frac{R}{\delta}\right)}{k} & pt\ d=\frac{k}{R}\\ 0 & pt\ d=\frac{k}{R}+1,\ldots,k\end{cases}$$

- Distribuția soliton ideală și robustă pentru k=10000, c=0.2, δ=0.05, R=244, k/R=41, iar β≈1.3 [1]

[1] MacKay, D.J.C.: 'Information theory, inference, and learning algorithms' -Cambridge University Press, 2003



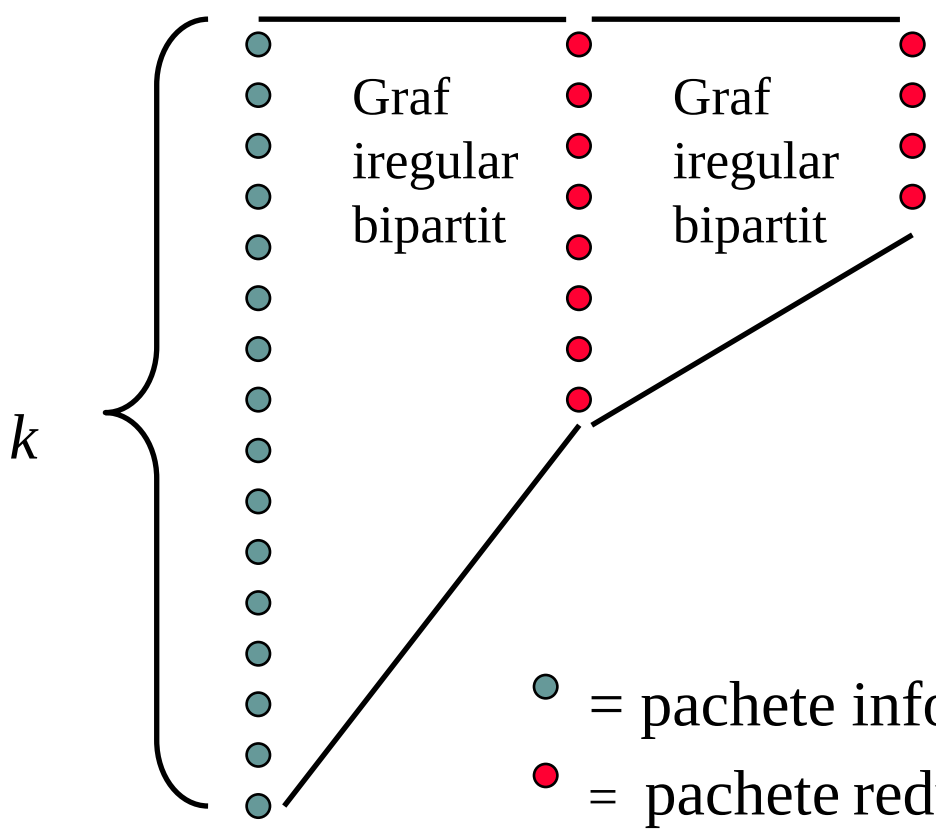
- În articolul lui Luby (2002) a demonstrat că primele valori a distribuției $\tau(d)$ (d mic) asigură pornirea procesului de decodare, adică asigură ca la începutul decodării riplul să "crească" la dimensiunea dorită
- iar valoarea relativ ridicată a funcției $\tau(d)$ pentru d = k/R este inclus pentru a asigura ca fiecare simbol (pachet) informațional are cel puțin un vecin între simbolurile recepționate, adică acest "vârf" asigură ca suma gradurilor să fie suficient de mare ca cele $K = k + O\left(\ln^2\left(\frac{k}{\delta}\right)\sqrt{k}\right)$ pachete recepționate să acopere toate cele k simboluri informaționale

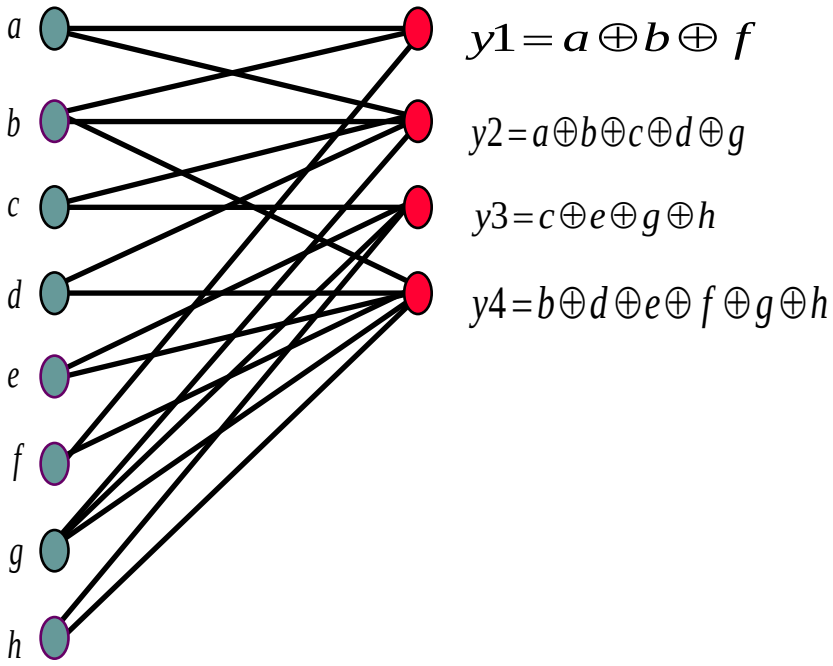


- Deoarece gradul pachetelor nu este constant
  - Timpul de codare/decodare nu este liniară
  - Probabilitatea de pierdere a pachetelor nu este uniformă
- Problema este rezolvată de codurile Raptor introduse de Amin Shokrollahi

Coduri Tornado

- Codurile Tornado sunt construite pe baza unor grafuri bipartite



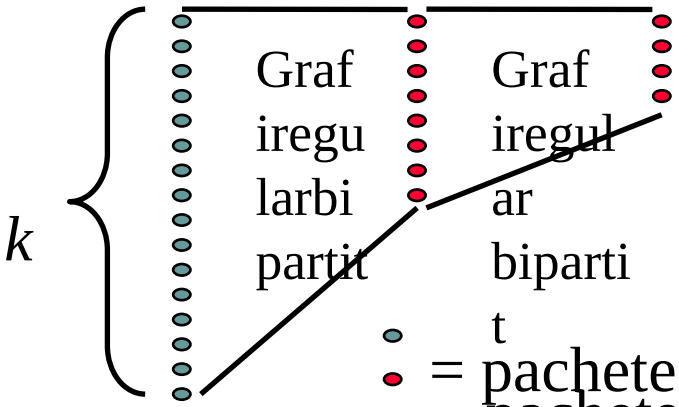


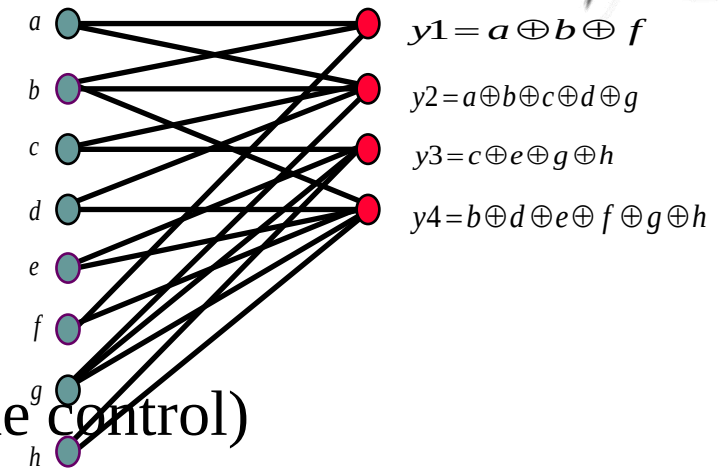
= pachete info

= pachete redundante (de control)

[2] John W.Byers, Michael Luby, Michael Mitzenmacher, Ashutosh Rege; “A Digital Fountain Approach to Reliable Distribution of Bulk Data” -Proceedings of the ACM SIGCOMM '98

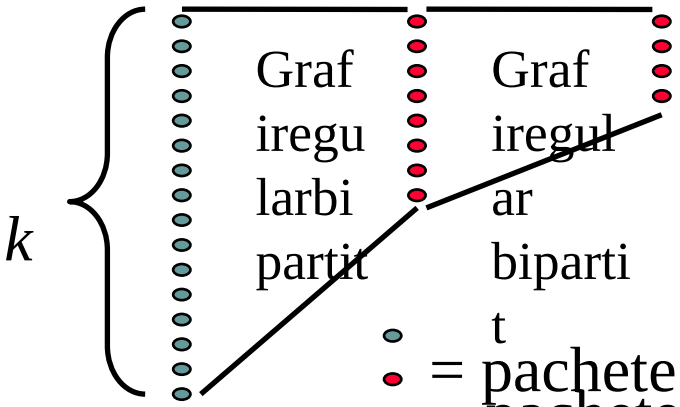
Coduri Tornado

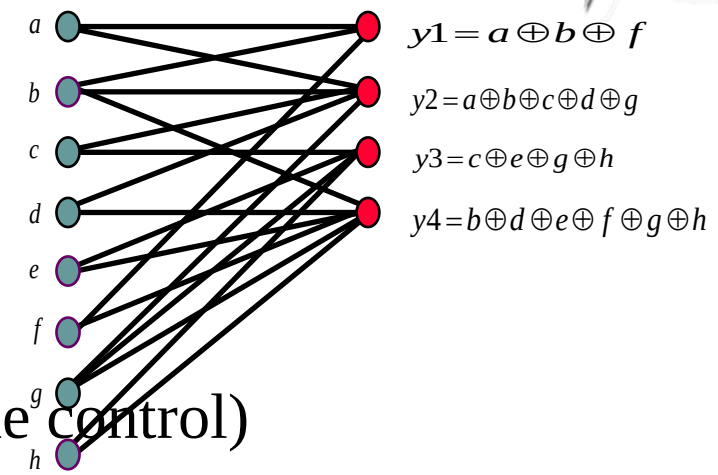




- în prima etapă din cele k pachete informaționale se obțin βk pachete de control (β număr pozitiv subunitar) conform grafului B1
- în a doua etapă pornind de la cele βk pachete de control obținute în etapa anterioară se obțin alte ββk pachete de control conform grafului B2

Coduri Tornado

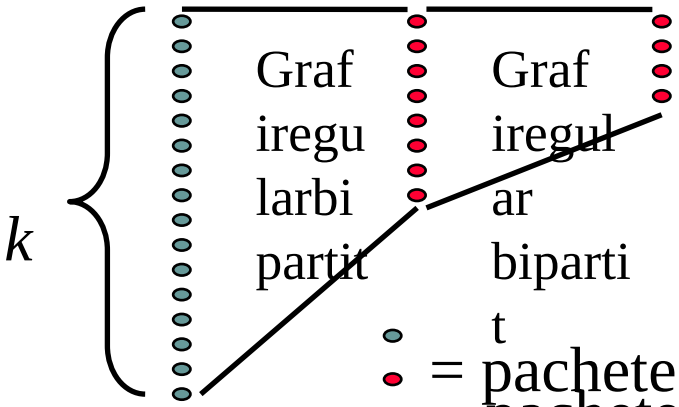


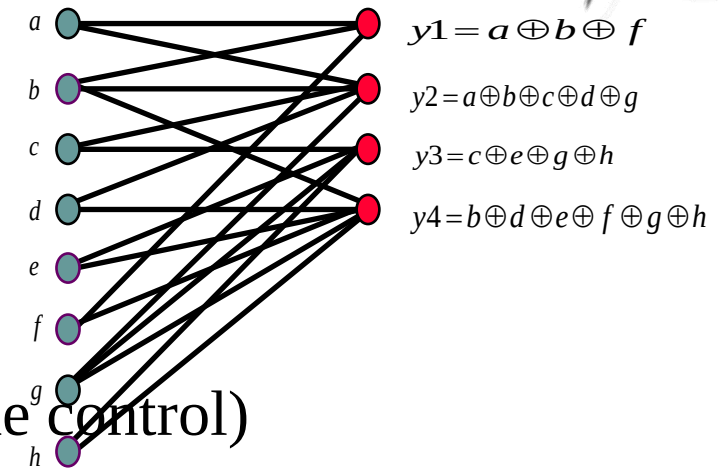


- în ultima (m-a) etapă se utilizează un cod corector de ștergeri "clasic", C, cu rata 1- β care poate să corecteze β ștergeri
- Codul C are $\beta^{m-1}k$ simboluri (pachete) la intrare, și generează β $\beta^{m-1}k/(1-\beta)$ simboluri de control adiționale
- Numărul simbolurilor de control generate în cele m etape este

$$\sum_{i=1}^{m-1}\beta^{i}k+\frac{\beta^{m}k}{1-\beta}=\frac{k\beta}{1-\beta}$$

Coduri Tornado

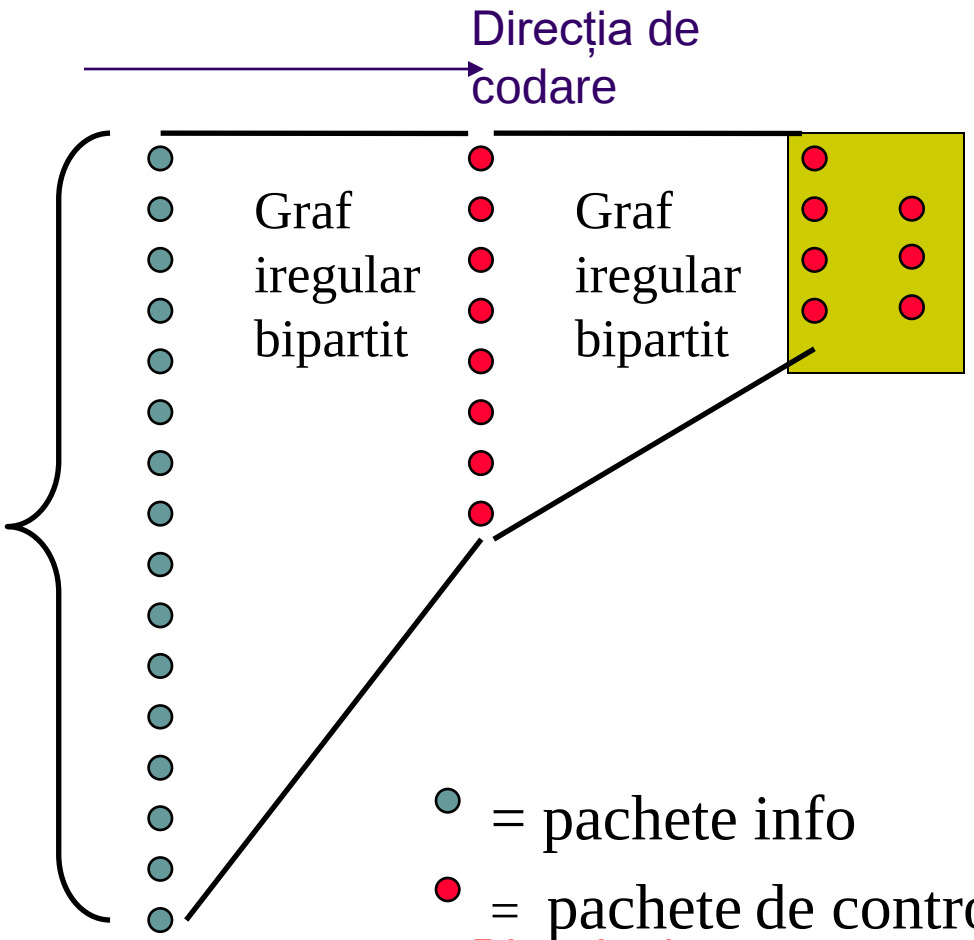


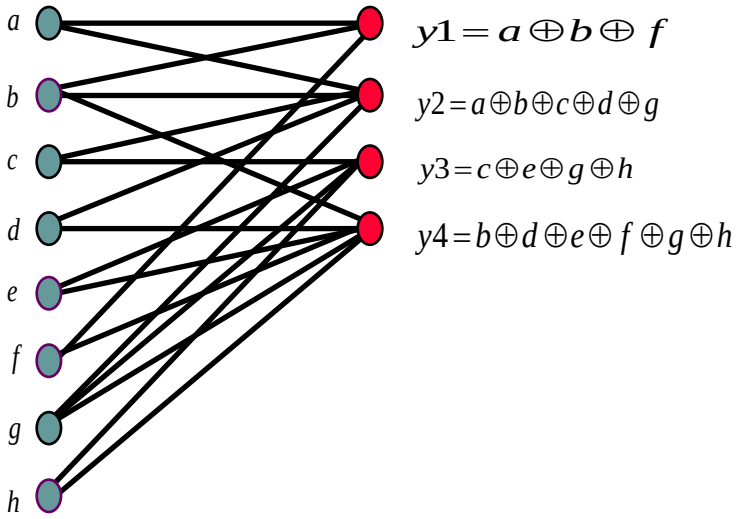


- rata globală va fi:

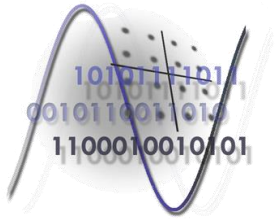
$$R = \frac{k}{k + \frac{k\beta}{1-\beta}} = \frac{k}{\frac{k - k\beta + k\beta}{1-\beta}} = 1 - \beta$$

- Codul global $C(B_1, B_2, \ldots B_{m-1}, C)$ este un corector de ștergeri cu rata 1-β, care poate să corecteze cu probabilitate mare orice ștergere până la lungime·β

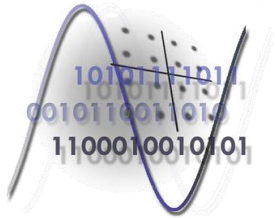

Direcția de codare
Graf iregular bipartit
Graf iregular bipartit
= pachete info
= pachete de control
Direcția de de-codare


a
b
c
d
e
f
g
h
y1 = a ⊕ b ⊕ f
y2 = a ⊕ b ⊕ c ⊕ d ⊕ g
y3 = c ⊕ e ⊕ g ⊕ h
y4 = b ⊕ d ⊕ e ⊕ f ⊕ g ⊕ h

Coduri Tornado



- Procesul de codare constă în sumarea modulo 2 a pachetelor conform grafului “generator”
- Pachetele recepționate sunt “substituite” în ecuațiile de control
- Prin înlocuirea variabilelor în ecuații de control, se pot reconstitui pachetele pierdute, utilizând adunări modulo 2
- De regulă primele k pachete sosite generează un număr redus de “rezolvări”, dar când numărul pachetelor recepționate este mai mare de k, un pachet recepționat generează o avalanșă de “rezolvări” permițând reconstrucția pachetelor informaționale



- Deoarece nodurile din graf au un număr redus de vecini (ecuații cu putini termeni) operațiunea de codare și decodare nu necesită multe operații
- Numărul de operații pentru obținerea pachetelor redundante depinde numai de gradul nodului respectiv
- Complexitatea decodării depinde tot de gradul nodurilor și de "poziția" pachetelor recepționate în graf
- Codurile Tornado sunt tot coduri cu lungime finită, deci pentru utilizarea lor ca DF pachetele codate se transmit întrețesute ciclic

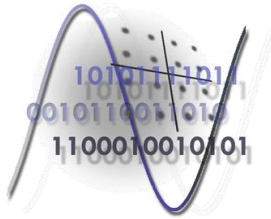
- Comparație între timpii de codare și decodare

Timpul de codare, pachete 1K		
Size	Reed-Solomon	Tornado
250 K	4.6 sec.	0.11 sec.
500 K	19 sec.	0.18 sec.
1 MB	93 sec.	0.29 sec.
2 MB	442 sec.	0.57 sec.
4 MB	30 min.	1.01 sec.
8 MB	2 hrs.	1.99 sec.
16 MB	8 hrs.	3.93 sec.

Timpul de decodare		
Size	Reed-Solomon	Tornado
250 K	2.06 sec.	0.18 sec.
500 K	8.4 sec.	0.24 sec.
1 MB	40.5 sec.	0.31 sec.
2 MB	199 sec.	0.44 sec.
4 MB	13 min.	0.74 sec.
8 MB	1 hr.	1.28 sec.
16 MB	4 hrs.	2.27 sec.

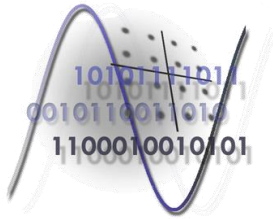
[1] John W.Byers, Michael Luby, Michael Mitzenmacher, Ashutosh Rege; “A Digital Fountain Approach to Reliable Distribution of Bulk Data” -Proceedings of the ACM SIGCOMM '98

Tornado vs LT



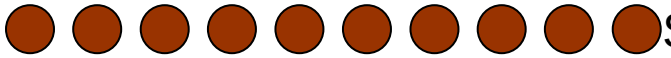
- Atât codurile RS cât și codurile Tornado sunt coduri sistematice, în schimb codurile LT sunt nesistematice
- memoria necesară pentru codarea decodarea codurilor tornado este mult mai mare decât în cazul codurilor LT
- codurile LT sunt coduri *rateless* iar codurile tornado au rata finită
- Codurile Tornado sunt generate pe baza grafurilor cu grad maxim constant, în schimb codurile LT au grafuri cu densitate logaritmică

Coduri Raptor(RAPide TORnado)



- Primul cod DF cu timp de codare și decodare liniară
- au fost inventate in 2000/2001 publicate in 2004
- Se acceptă ca o fracțiune maxim δ din simbolurile de intrare a unui cod LT să nu fie acoperite la terminarea procesului de decodare LT
- Aceste simboluri "pierdute" se pot recupera utilizând un cod corector de ștergeri clasic
  - Se cunoaște numărul maxim de ștergeri posibile
- Implică o precodare cu un cod corector de ștergeri clasic

[5] A. Shokrollahi,„Raptor Codes",http://www.cs.huji.ac.il/labs/danss/p2p/resources/raptor.pdf

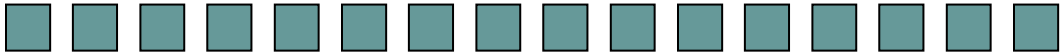


Simboluri de intrare

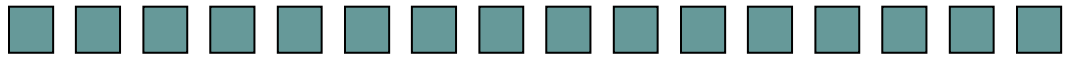
Codare cu cod corector de ștergeri

simboluri precodate

LT-”light”



Simboluri codate

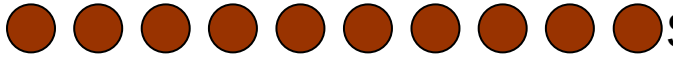


Simboluri recepționate

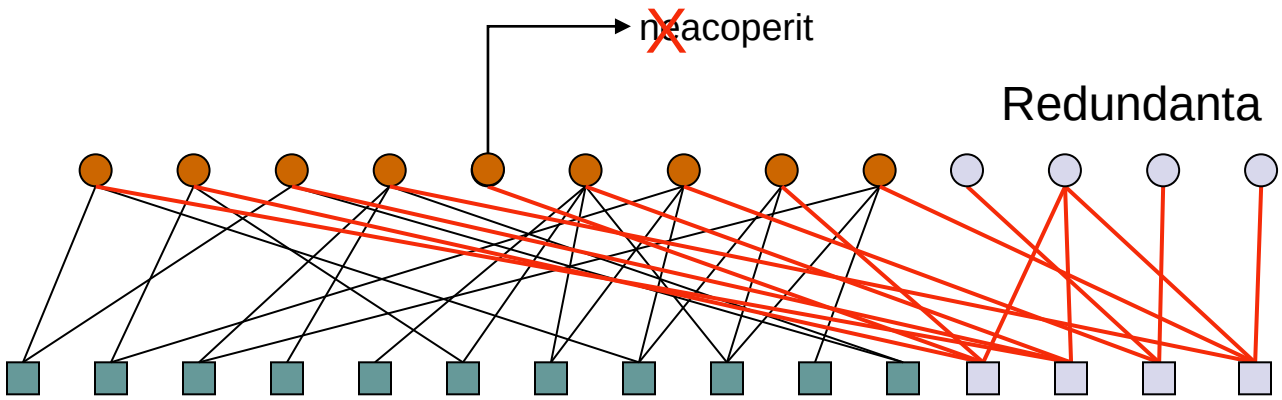
decodor LT

maxim δ simboluri neacoperite

Codare cu cod corector de ștergeri

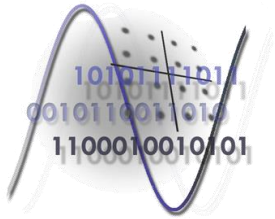


Simboluri decodate

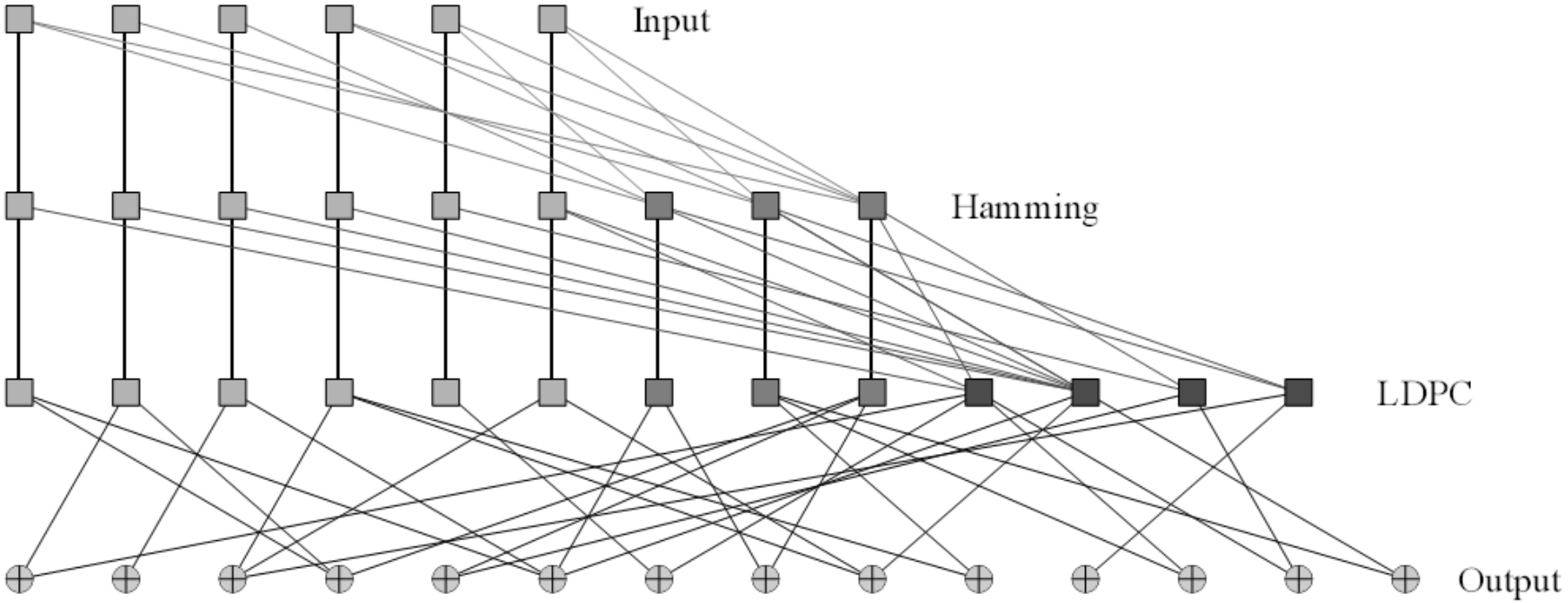


- Dacă precodorul este ales în mod corespunzător, atunci se poate utiliza un cod LT cu gradul mediu constant, care asigură timp de codare liniară
- Un cod Raptor (k,C, Ω(d))este definit de numărul de pachete informaționale k, codul corector de ștergeri C și distribuția gradurilor al codului LT Ω(d)

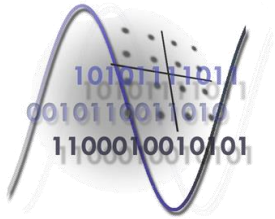
Coduri Raptor



- Parametrii principali de performanță ai codului Raptor sunt definite după cum urmează:
  - Spațiul de memorie: Codurile Raptor necesită spațiu de stocare pentru simbolurile intermediare, Consumul de spațiu al codurilor Raptor este k/*R*, unde *R* este rata pre-codului.
  - Overhead: Overheadul este o funcție a algoritmului de decodare folosit, și este definit ca numărul de simboluri de ieșire pe care trebuie să aibă decodorul pentru a recupera cu probabilitate mare simbolurile de intrare. Un overhead de $1+\varepsilon$ însemnă că trebuie recepționate $k(1+\varepsilon)$ simboluri de ieșire pentru a asigura o decodare cu succes cu o probabilitate mare.
  - Costul: Costul procesului de codare și de decodare.


Input
Hamming
LDPC
Output

Probleme legate de DF



- Implementarea oricărei aplicații, se poate face numai cu acordul DF inc.
- "…but networking people do not want to deal with developing codes." [1]
- datorită drepturilor de autor aplicațiile cu DF sunt foarte puțin cunoscute

[1] Michael Mitzenmacher-Digital Fountains: Applications and Related Issues