

# Theory and Design of Turbo and Related Codes

## *Lecture 5*

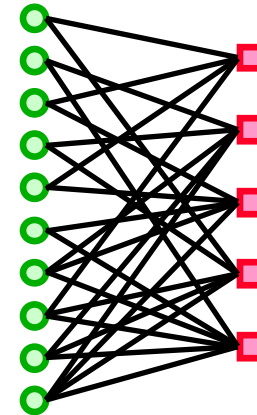
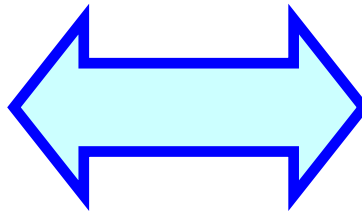
*Jossy Sayir & Gottfried Lechner*

<http://userver.ftw.at/~jossy/turbo/index.html>

# What we have learned...

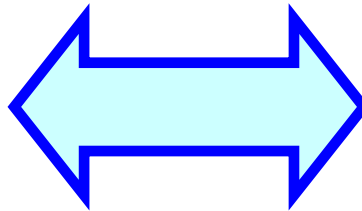
$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Parity-check matrix



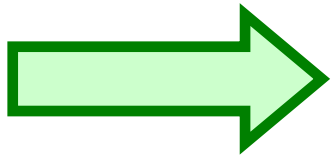
Bi-partite  
Factor Graph

Iterative  
Decoding



Message passing  
on graph

## What we have learned...



Iterative decoding of LDPC codes  
for general binary-input memoryless  
symmetric channels!

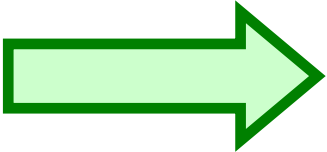
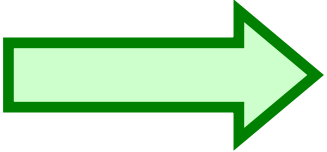
- Alternating variable & check node decoding
- Local decoding at variable nodes: repeat code
- Local decoding at check nodes: single parity-check code
- Message passing of L-values



... but how well does message-passing decoding  
perform in practice??

# Concentration theorems

(Luby, Mitzenmacher, Shokrollahi, Spielman / Richardson & Urbanke)

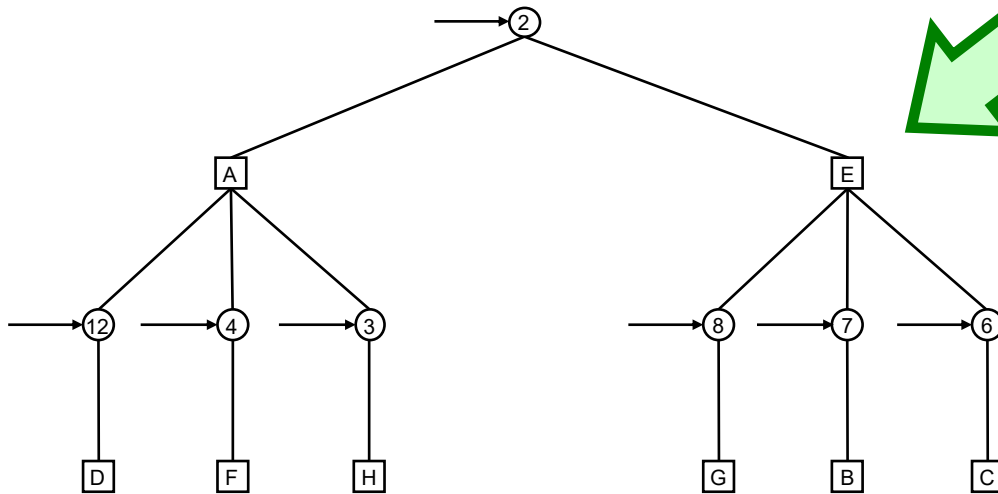
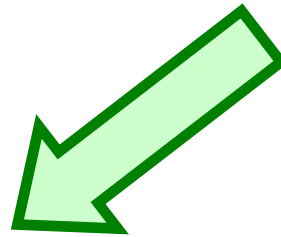
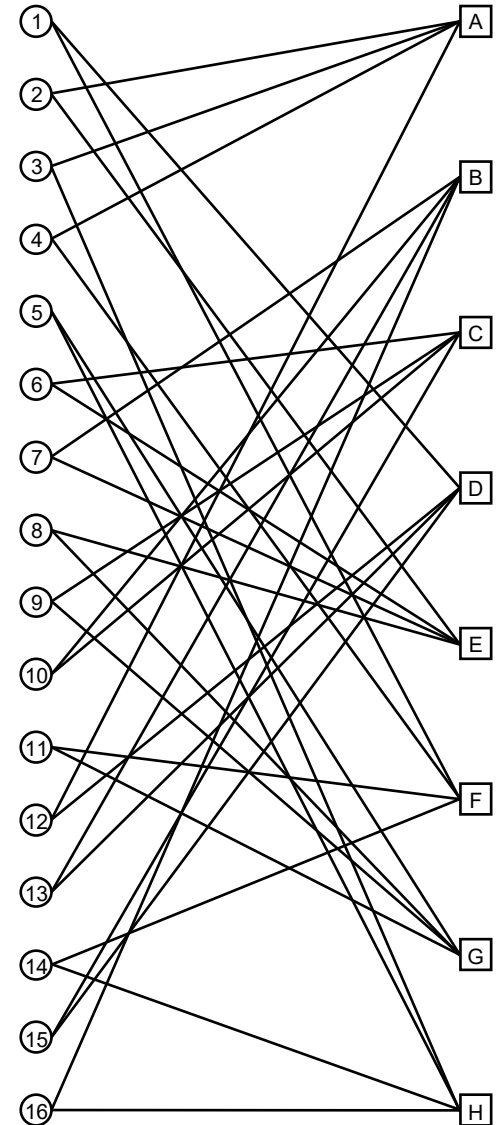
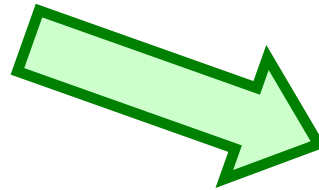
- We will only consider **expected decoding behavior** over all graphs of a given degree distribution (not the behavior for one specific graph/code)
- **Concentration**: the **probability** that the **performance** of a specific code **diverge by  $\varepsilon$**  from the expected performance over all graphs **converges to 0** exponentially in the code length  **$N$**   
 **expected performance suffices**
- **Cycle-free** behavior: for any iteration number  **$n_{it}$** , one can choose a code length  **$N$**  so that the expected **performance** at iteration  **$n_{it}$**  is **as near as desired** to the decoder performance under the **tree assumption**  
 **tree behavior suffices for  $N \rightarrow \infty$**

# Reminder: Matrix $\rightarrow$ Graph $\rightarrow$ Tree



$H =$

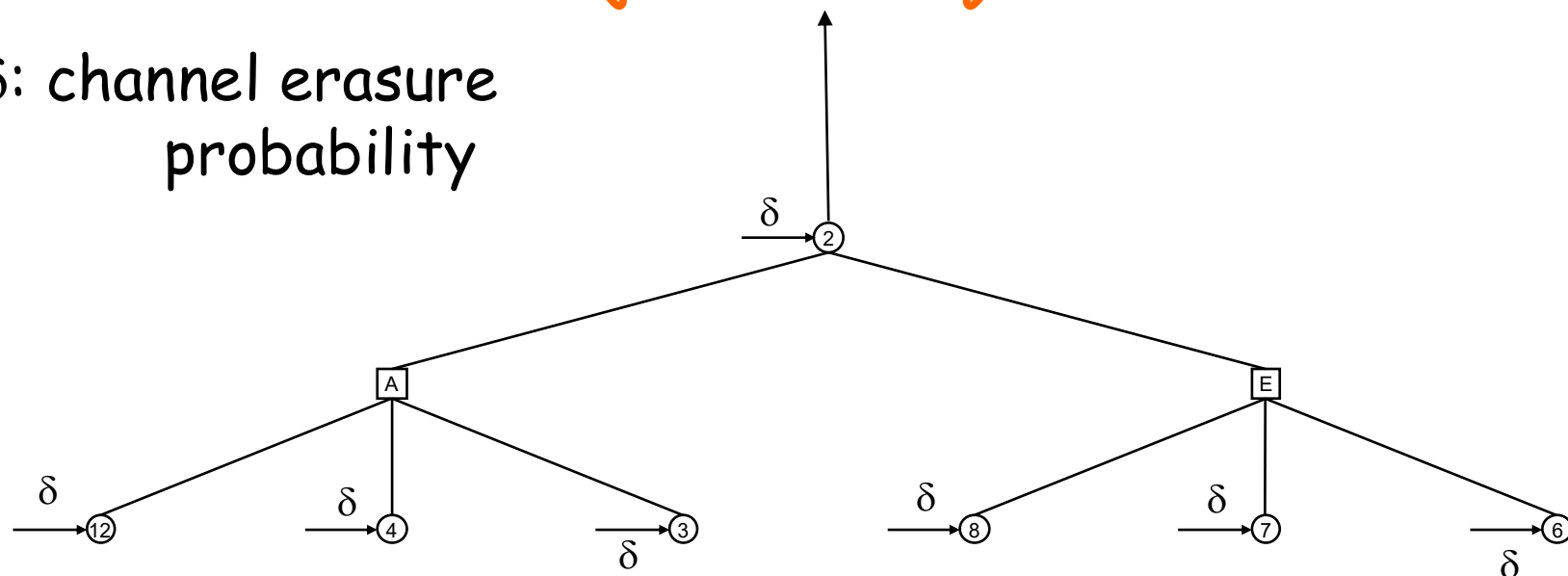
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	1	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1



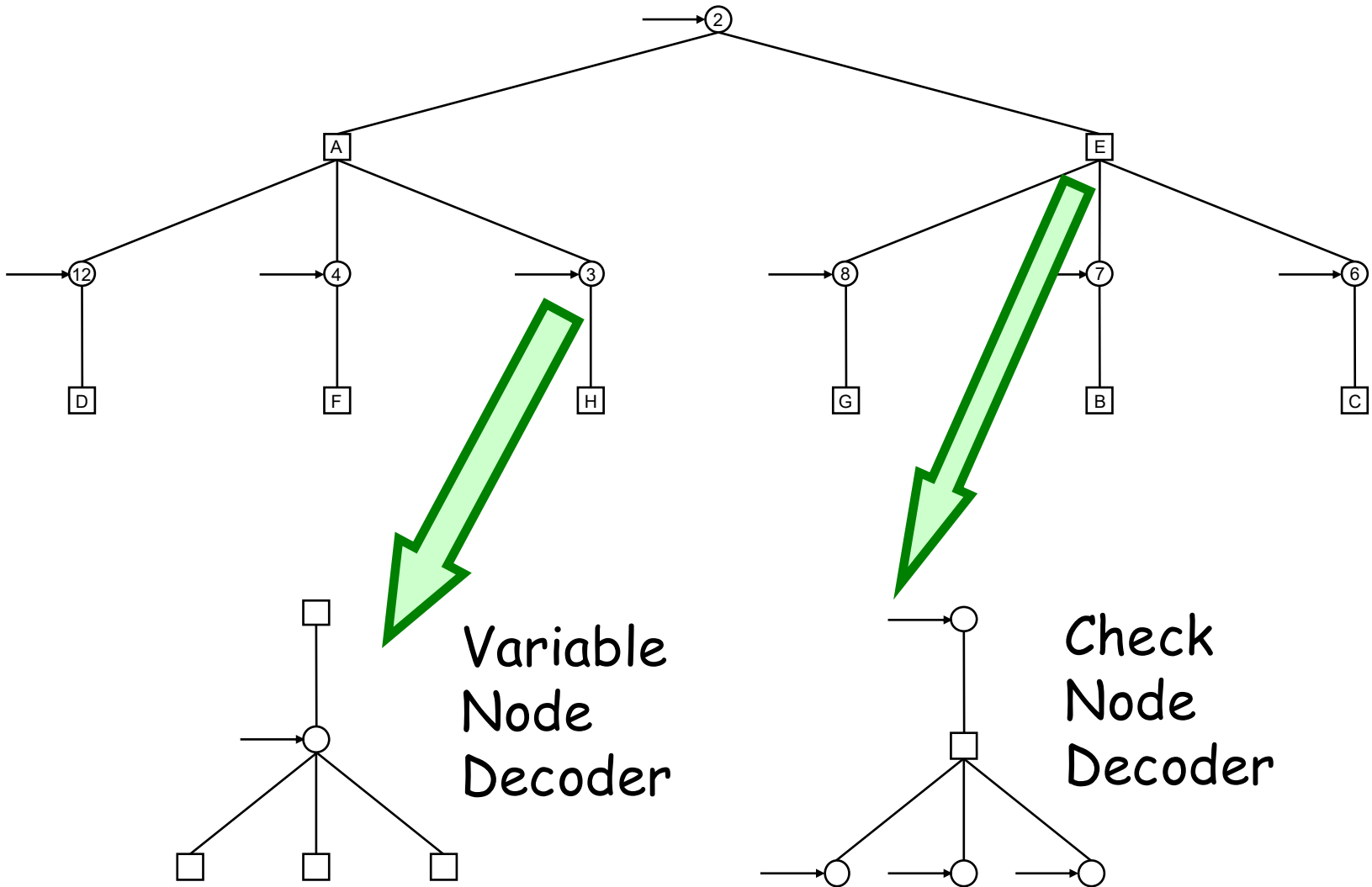
# Tree Decoding Performance for the BEC

$$P(\text{erasure}) = ?$$

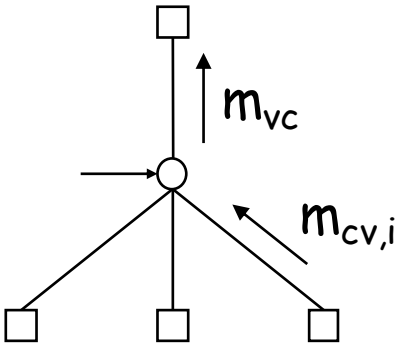
$\delta$ : channel erasure probability



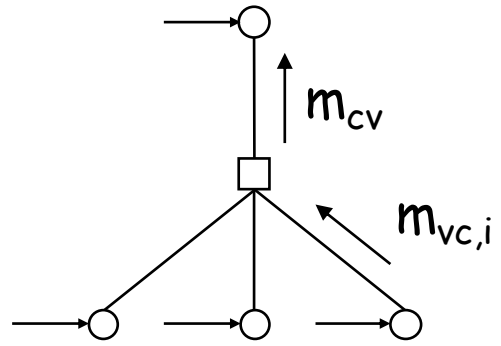
# Reminder: Elements of the Tree Perspective



# Reminder: Decoding for the BEC



Repeat Code



Single Parity-Check Code

$$m_{vc} = \begin{cases} 0, 1 & \text{if at least one } m_{cv,i} \text{ is no erasure} \\ & \text{or the node itself is no erasure} \\ \Delta & \text{if all } m_{cv,i} \text{ are erasures and the node} \\ & \text{itself is an erasure} \end{cases}$$

$$m_{cv} = \begin{cases} \sum_i m_{vc,i} & \text{if no } m_{vc,i} \text{ is an erasure} \\ \Delta & \text{if at least one } m_{vc,i} \text{ is an erasure} \end{cases}$$

# “Simulating Simulating”

(Tom Richardson, ISIT 2004, Chicago)

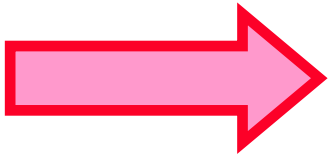
- **Simulation:** run message-passing decoding for codewords transmitted over a binary erasure channel and **measure the resulting error probability**
- **Simulating simulation:** compute **probability distributions of messages** passing through the decoder
- Instead of  $m_{vc}$  and  $m_{cv}$ , compute  $*m_{vc} = P(m_{vc} = 0, 1, \Delta)$  and  $*m_{cv} = P(m_{cv} = 0, 1, \Delta)$

What is  $*m_{vc}$  at iteration 0 ?

Notation:  $*m_{vc} = ( P(m_{vc} = 0), P(m_{vc} = 1), P(m_{vc} = \Delta) )$

Intuitively:  $*m_{vc}(0) = ( (1-\delta)/2, (1-\delta)/2, \delta )$

Why is this correct??



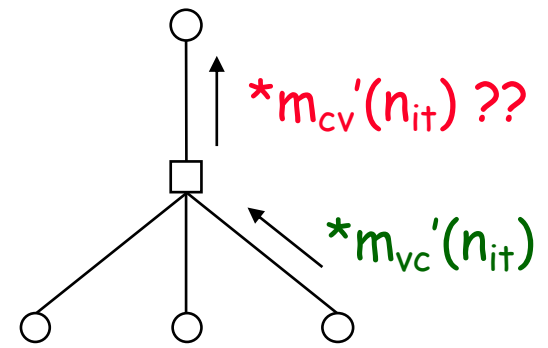
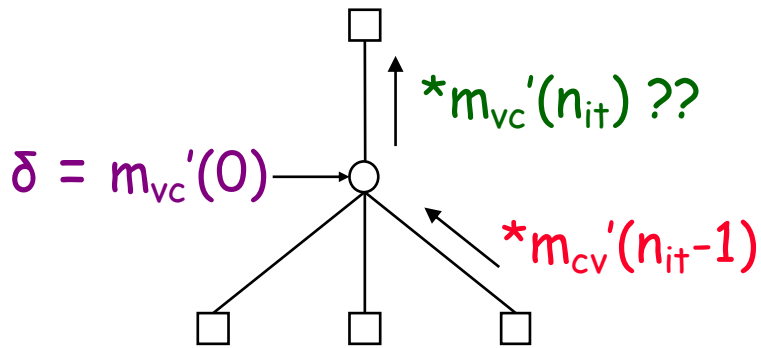
Linear codes... (explain)

It suffices to consider binary messages

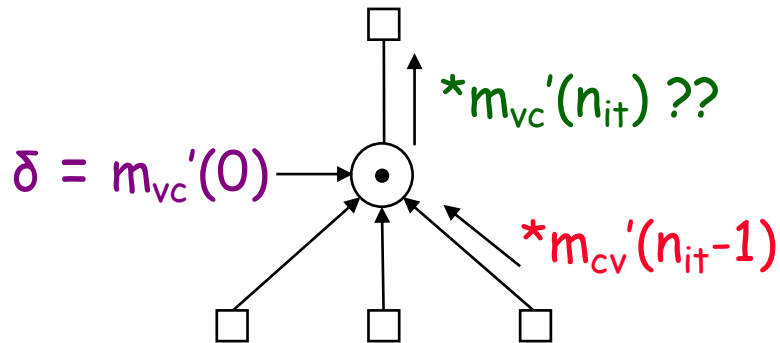
$m_{vc/cv}'(n_{it}) = 1$  if erasure,  $0$  if non-erasure

and track  $*m_{vc/cv}'(n_{it}) = P( m_{vc/cv}'(n_{it}) = \Delta )$

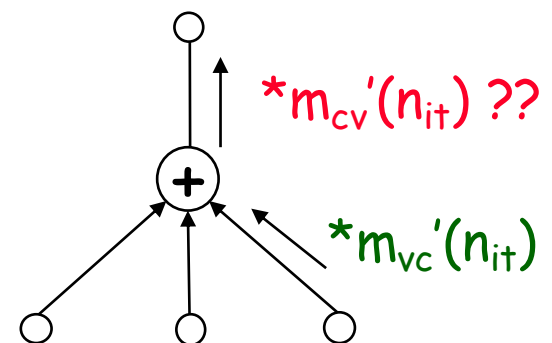
# Combining Erasure Probabilities



Equivalently:



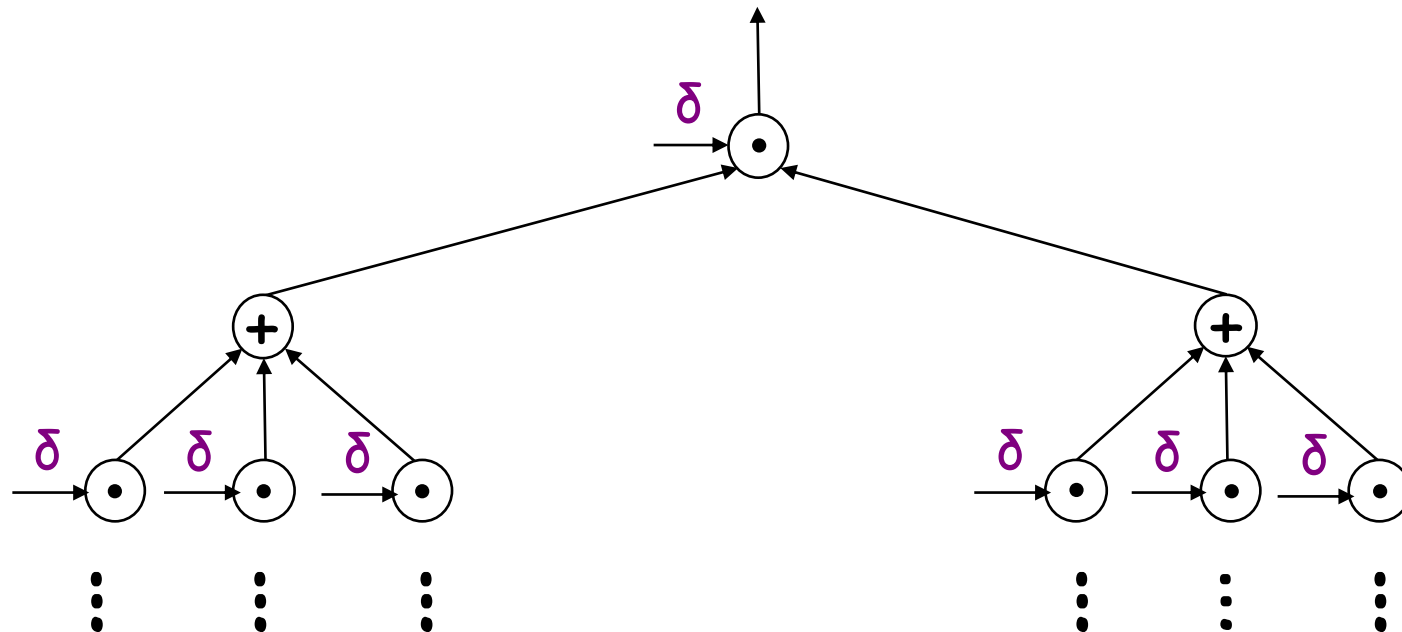
AND - operation



OR - operation

# AND-OR Tree

$$P(1) = ?$$



Let's simplify notation:

$$p_0 = \delta = m_{vc}'(0)$$

$$q_k = m_{cv}'(k),$$

$$p_k = m_{vc}'(k)$$

$$q_k = 1 - (1 - p_{k-1})^{d_c-1}$$

$$p_k = p_0 q_k^{d_v-1}$$

$$p_k = p_0 \left( 1 - (1 - p_{k-1})^{d_c-1} \right)^{d_v-1}$$

# Asymptotic Analysis for the Binary Erasure Channel

What is the probability of erasure for iterative decoding after  $k$  iterations?

The channel has a probability of erasure  $p_0$

The code has infinite length  $N \rightarrow \infty$

After iteration  $k$ , the probability of erasure is  $p_k$

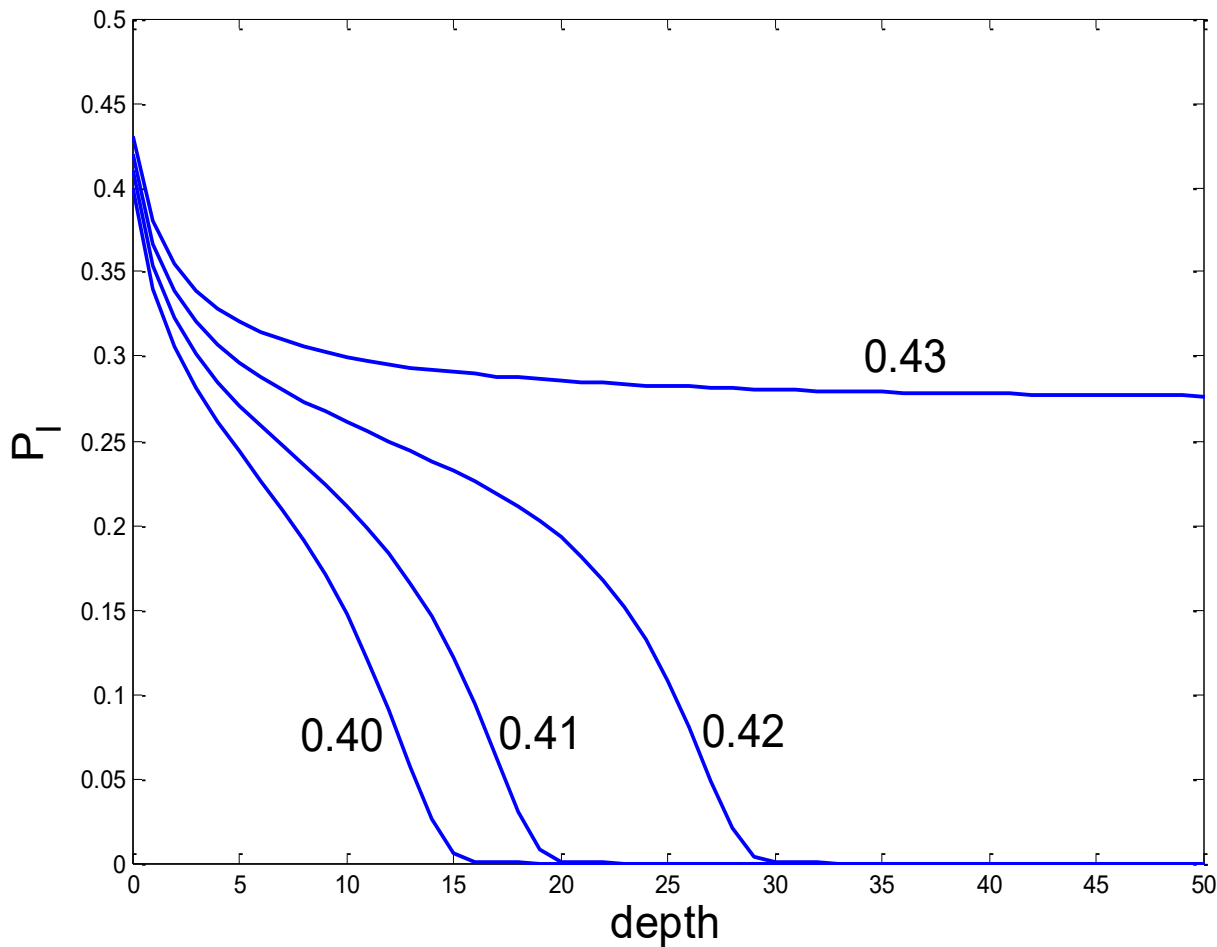
According to Luby, Mitzenmacher, Shokrollahi & Spielman, for regular  $(d_v, d_c)$  codes:

$$p_{k+1} = p_0 (1 - (1 - p_k)^{d_c - 1})^{d_v - 1}$$

# Applying the recursion

$$p_{k+1} = p_0 (1 - (1 - p_k)^{d_c - 1})^{d_v - 1}$$

$$d_v = 3 \quad d_c = 6$$



Threshold:  
0.42944

# BEC & Irregular LDPC Codes

Luby, Mitzenmacher, Shokrollahi & Spielman

Michael G. Luby was at Univ. of **Berkeley** and founded **Digital Fountain** after co-inventing irregular LDPC codes

Michael Mitzenmacher got his PhD from **Berkeley** in 1996, worked for DEC Research, then joined Harvard University in 1999 as Assistant Professor

Daniel A. Spielman got his PhD from MIT in 1995, did a post-doc in **Berkeley** 1995-96, then went back to MIT as Assistant Professor

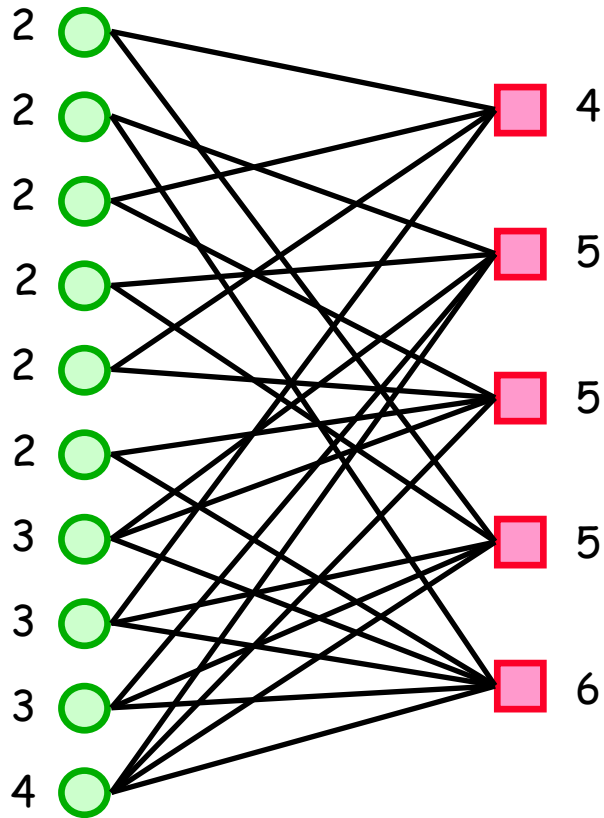
M. Amin Shokrollahi got his Dipl.-Ing. from the Univ. of Karlsruhe, his Dr. from the Univ. of Bonn. 1991, worked as a researcher in **Berkeley**, joined Bell Labs from 1998 to 2000, then Digital Fountain until 2003. He has been a Professor at EPF Lausanne since 2003.



Photo from shokrollahi.com

M. Amin Shokrollahi

# Irregular Graphs



$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} 4 \\ 5 \\ 5 \\ 5 \\ 6 \end{matrix}$$

2 2 2 2 2 2 3 3 3 4

$$(d_{v,1}, d_{v,2}, d_{v,3}, d_{v,4}, d_{v,5}, d_{v,6}, d_{v,7}, d_{v,8}, d_{v,9}, d_{v,10}) = (2, 2, 2, 2, 2, 2, 3, 3, 3, 4)$$

$$(d_{c,1}, d_{c,2}, d_{c,3}, d_{c,4}, d_{c,5}) = (4, 5, 5, 5, 6)$$

$\ell_i$ : proportion of left (variable) nodes with degree  $i$

$r_i$ : proportion of right (check) nodes with degree  $i$

In our example,  $\ell_2 = 6/10$ ,  $\ell_3 = 3/10$ ,  $\ell_4 = 1/10$   
 $r_4 = 1/5$ ,  $r_5 = 3/5$ ,  $r_6 = 1/5$

$\lambda_i$ : proportion of edges incident to left nodes with degree  $i$

$\rho_i$ : proportion of edges incident to right nodes with degree  $i$

In our example,  $\lambda_2 = 12/25$ ,  $\lambda_3 = 9/25$ ,  $\lambda_4 = 4/25$   
 $\rho_4 = 4/25$ ,  $\rho_5 = 15/25$ ,  $\rho_6 = 6/25$

# Degree Polynomials (continued)



Total number of edges:  $\#(\text{edges}) = \sum_i i r_i = \sum_i i \ell_i$

$$\lambda_i = i \ell_i / \#(\text{edges})$$

$$\rho_i = i r_i / \#(\text{edges})$$

Degree Polynomials:

$$\lambda(x) = \sum_{i=2}^{d_{v,max}} \lambda_i x^{i-1} \quad \rho(x) = \sum_{i=2}^{d_{c,max}} \rho_i x^{i-1}$$

$$p_{k+1} = p_0 \lambda(1 - \rho(1 - p_k))$$

With irregular LDPC codes, we have more flexibility and can achieve better performance than regular LDPC codes

For the Binary Erasure Channel,  
we can achieve a threshold of 0.5  
for rate  $R = 1/2$  codes!!