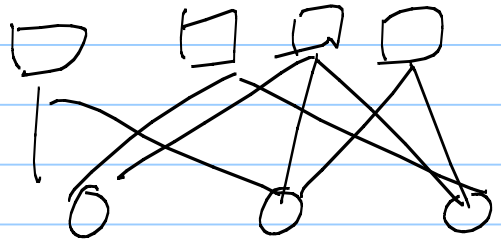
Lecture 25

Note Title 4/11/2012

Defn: A **stopping set** $S_{stopping}$ is a subset of variable nodes such that all neighbors of $S_{stopping}$ are connected to $S_{stopping}$ at least twice

(neighbors: check nodes)



Corollary: When the erasure prob ε is small. The asymptote of the frame error rate is

$$m_s \cdot \varepsilon^{d_s}$$

where d_s is the minimal stopping distance (the minimal size of non-empty stopping sets.)

& m_s is the multiplicity of the minimum stopping sets.

This asymptote is the cause of the error floor.

Q: What is the cause of $S_{stopping}$?

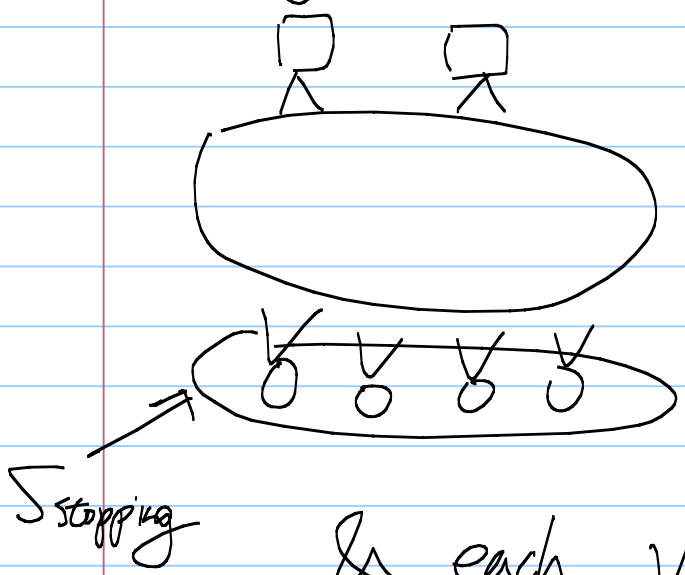
A: A heuristic answer is "circle"

Lemma: Suppose $\min_v d_v \geq 2$. Then any $S_{stopping}$ must contain a cycle. (But not vice-versa)

pf: we start from any $v \in S_{stopping}$

keep:

& moving between its check node neighbors & $S_{stopping}$.



Since each check node neighbor is connected to $S_{stopping}$ ≥ 2 times.

& each $v \in S_{stopping}$ has ≥ 2 neighbors. We can keep moving until we visit a node twice (it can be either a check or a variable)

$\Rightarrow$ We have a cycle.

⇒ Many finite LDPC code constructions focus on enlarge the "girth" of the code:

$$\text{maximize}_{\text{choose your interleaver}} \; \min_{\text{all cycles}} \; |\text{length of the cycle}|$$

$\triangleq$ girth

* Notheless, girth alone does not fully predict the performance.

Ex: For a regular (3,6) code with $n = 64$. The "girth" is upper bounded by 3 var. + 3 check nodes $= 6$.

Random construction gives you a code of minimum stopping distance $d_s =$ 2-4

An optimized (3,6) code can have $d_s = 8$ (8 variables)

* Unfortunately. given an arbitrary LDPC code, & an integer t,

deciding whether "$d_S \leq t$" is an NP complete problem. It is intrinsically hard to design a code with large d_S.

NP-complete vs feasibility.

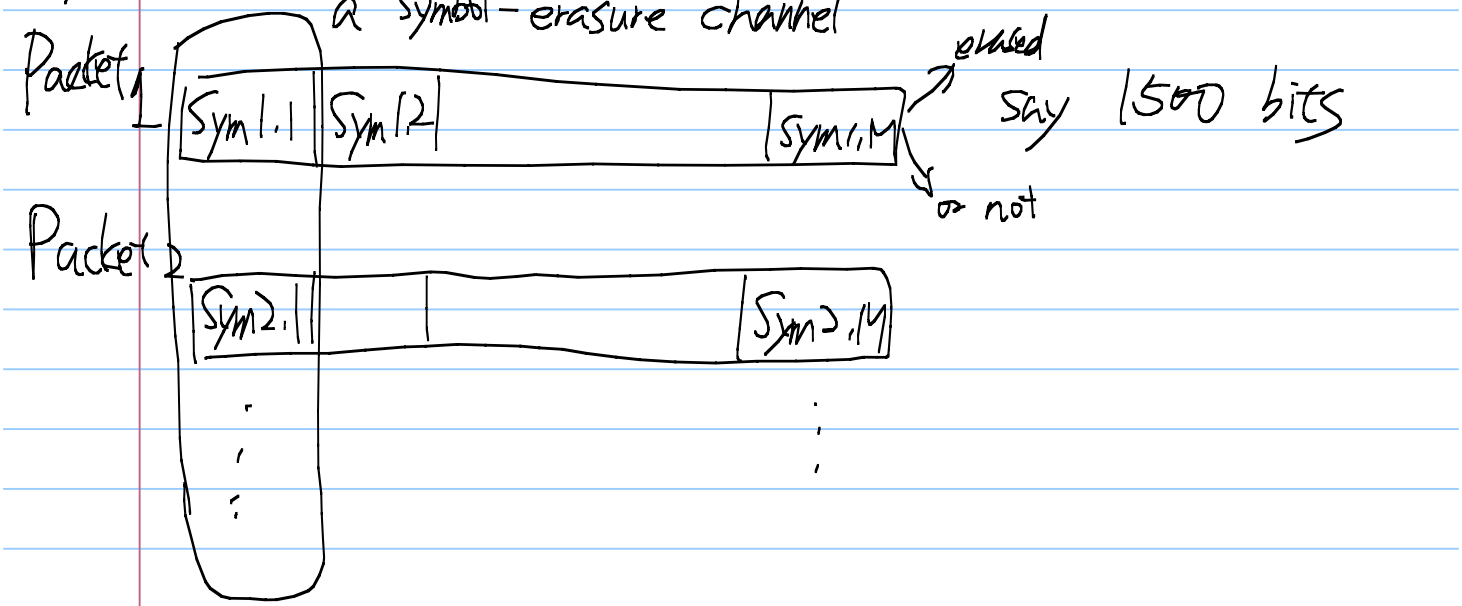
Ensemble-based approaches also have problems.

* Fountain Codes & Network coding
  - Coding for network communications.

* Network communication is a natural application of erasure channels. since a packet is either erased or not, which can be checked by CRC. (Cyclic Redundancy Check) corrupted

* Fortunately, for erasure channels, we have had most success of capacity-approaching codes.

* Different Views of a Pack-Erasure channel



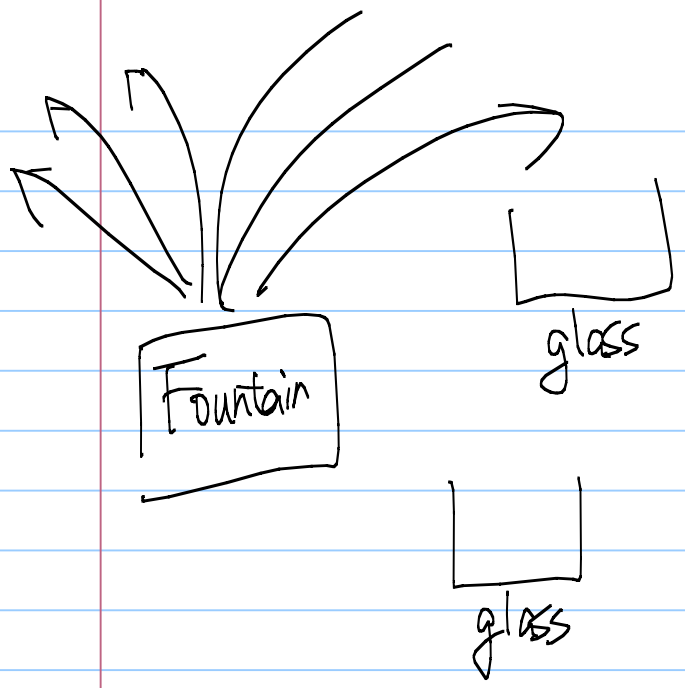
when each symbol is a bit, then we have converted PEC to BEC. If each symbol is GF(q), then we have GF(q) - erasure channel.

* Fountain Codes (or the Luby-Transform (LT) codes)

: A binary rate-less code for BECs.

Comparison

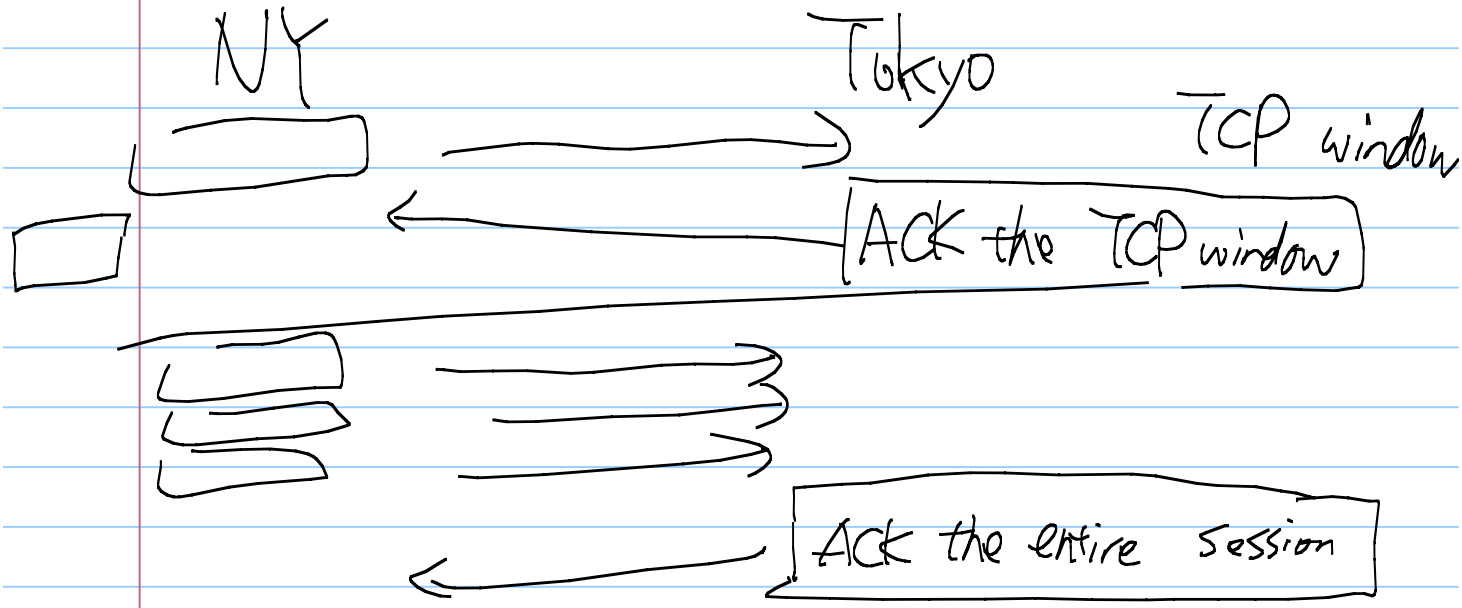
Rate $\frac{1}{2}$ code	Rateless LT code
k info bits	k-info bits.
$2k$ coded bits	an infinite stream of coded bits
If the destination receives $\geq k$ coded bits $\rightarrow$ decoding the k-info bits	k
If success prob $< \frac{1}{2}$, we are doomed	If success prob $< \frac{1}{2}$, keep transmitting until the desti. receives $\geq k$ coded bits



any glass that contains k received pkts (water) is sufficient.

Benefit: ① Universal design for any BEC of different parameters: ε

② Greatly reduce the feedback traffic & the round-trip waiting time for the ACK packet



* LT code construction:

① Start from k info bits, k is usually ≥ 1000.

② A prob distri $p_1 \cdots p_k$ is predefined $\sum_{i=1}^{k} p_i = 1$

③ Whenever we can transmit a bit (a packet)
choose independently & randomly a "d" value based on the distribution $\{p_i\}$.

④ Choose randomly d distinct bits.
$X_{i_1}, X_{i_2}, X_{i_3}, \cdots X_{i_d}$
& send $Y_t = X_{i_1} + X_{i_2} + \cdots + X_{i_d}$
$=$
↳ binary XOR

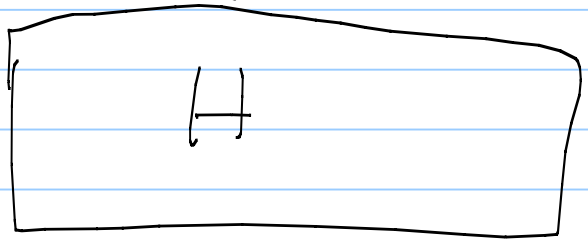
Our goal is when receiving $Y_1, \cdots, Y_k$ we can decode $X_1, \cdots, X_k$ efficiently.

* Remark: Due to the indepent construction it does not matter which Y_t that the destination really receives. That's why we assume that the first k coded bits $Y_1, \cdots, Y_k$ are received.

* The design parameters are $P_1, \cdots, P_k$

Q: How to decode?

A: Once we receive k coded bits, we have

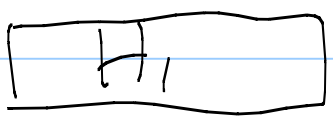


$$\vec{x} = \vec{y}$$

Decoding is equivalent to solving the equation:

Solution 1: Invesion, too complicated.

* Solution 2: Message passing (look for the rows of deg 1)



$$\vec{x}_{\text{erased}} = \vec{y} + H_0 \cdot \vec{x}_{\text{recovered}}$$