**Luby Transform Codes (LT Codes)**

1. **Đặt vấn đề**

Hiện nay trong quá trình dữ liệu đều có sử dụng kênh xoá (erasure channel). Với kênh xoá, dữ liệu truyền qua các kênh này sẽ chỉ nhận 2 giá trị: hoặc là 1 (đã nhận được gói tin hoàn chỉnh) hoặc là ? (packet lỗi). Thông thường khi sử dụng kênh xoá để truyền dữ liệu người ta sẽ sử dụng kĩ thuật backward error correction (BEC), tức là sẽ sử dụng thêm 1 kênh feedback gửi ACK (acknowledgement) từ bên nhận (receiver) về bên gửi (sender) để thông báo những packet nào chưa nhận được để được gửi lại. Kĩ thuật BEC được sử dụng rộng rãi vì nó sử dụng băng thông ít hơn kĩ thuật forward error correction (FEC), nhưng nhược điểm của nó là thời gian của một vòng truyền dữ liệu (round trip time – RTT) lâu hơn, và trong trường hợp broadcast thì việc feedback có thể dẫn đến việc broadcaster phải gửi lại toàn bộ dữ liệu. Thêm nữa, với việc truyền dữ liệu qua các kênh truyền không tin cậy hoặc có độ nhiễu cao với khoảng cách lớn thì rõ ràng kĩ thuật BEC không khả quan vì tỉ lệ lỗi cao, thời gian cho việc truyền nhận dữ liệu là quá lớn. Với việc truyền dữ liệu qua các khoảng cách lớn hay kênh truyền không có feedback (mạng di động) thì FEC lại là một lựa chọn tốt. Với kĩ thuật FEC, bên gửi sẽ gửi rất nhiều gói tin và bên nhận có thể tự phát hiện và sửa lỗi dữ liệu với một lượng packet nhất định.

Một trong những cách tiếp cận FEC cổ điển nhưng hiệu quả đó là Luby Transform Code - kĩ thuật truyền tin rateless đầu tiên với độ hiệu quả tỉ lệ thuận với độ lớn của dữ liệu. Chính vì khả năng xử lí hiệu quả với lượng dữ liệu lớn nên nhóm chúng em quyết định nghiên cứu LT code.

1. **Mục tiêu đề tài**

Nghiên cứu, tìm hiểu về LT Codes

1. **Giới thiệu về LT Codes**

Quy tắc hoạt động của LT Codes: chúng ta hình dung sender là một vòi phun còn các receiver là các thùng hứng, sender sẽ phun nước (dữ liệu) liên tục, còn các receiver chỉ việc hứng sao cho đầy thùng là hoàn tất.

LT Codes được giới thiệu bởi Michael Luby vào năm 1998 và được xuất bản thành báo vào năm 2002.

Các khái niệm:

* Input symbol: từ 1 file dữ liệu ta chia ra thành k phần nhỏ gọi là input symbol dùng cho việc mã hóa
* Encoding symbol: từ các input symbol ta tiến hành mã hóa chúng để tạo thành encoding symbol
* Bậc d của encoding symbol: là số lượng d input symbol đã được sử dụng để mã hóa thành encoding symbol đó

Đặc điểm của LT Codes:

* LT Codes truyền dữ liệu rateless, tức là lượng dữ liệu được mã hoá (encoding symbols) từ dữ liệu gốc là vô hạn, có thể được phát sinh thêm tuỳ ý. Cần định nghĩa như thế nào là rate của 1 code, tu đó mới giải thích được tại sao LT code la reateless
* Receiver có thể giải mã được dữ liệu gốc từ bất kì tập các encoding symbols nào mà nó đã nhận chỉ với 1 lượng thừa encoding symbols không đáng kể. (Chưa định nghĩa như thế nào là encoding symbol)
* Độ nhiễu của kênh truyền (channel) có lớn thì với việc encoding symbols được phát sinh liên tục đến khi receiver nhận đủ để decode là quá trình truyền hoàn tất => đảm bảo truyền dữ liệu trong các môi trường có độ nhiễu cao. (Có phần lặp ý với rateless, nên gộp chung với điểm 1)
* Thời gian mã hoá và giải mã sẽ càng tối ưu hơn khi xử lý các dữ liệu lớn.

1. **Quá trình mã hoá và giải mã**

Quá trình phát sinh 1 encoding symbol:

1. Chọn ngẫu nhiên bậc *d* cho encoding symbol từ hàm phân phối bậc (sẽ đề cập chi tiết ở phần sau)
2. Chọn ngẫu nhiên *d* input symbols (dữ liệu đầu vào) phân biệt từ dữ liệu gốc
3. XOR các input symbols với nhau để tạo thành encoding symbol

RANDOM

d

Chọn ngẫu nhiên d input symbol phân biệt từ dữ liệu gốc

XOR các input symbol lại ta thu được 1 encoding symbol

Quá trình giải mã dữ liệu gốc: receiver chỉ nhận được các encoding symbol.Để dễ hình dung cho việc giải mã thì chúng ta có 1 tập input symbols chưa được phủ song song với các encoding symbol.

Tập input symbols chưa được phủ

1

⊕

2

1

2

⊕

3

⊕

4

2

⊕

3

List các encoding symbol

1. Đầu tiên các input symbol đều chưa được phủ
2. Tất cả các encoding symbol có bậc 1 sẽ được giải mã và phủ input symbol kề với nó. Các input symbol được phủ nhưng chưa được xử lý thì được gọi là ripple.

2

2

⊕

3

⊕

4

2

⊕

3

ripple

giải mã

1. Mỗi vòng lặp ta xử lý 1 input symbol trong ripple, ta xoá cạnh nối nó với các encoding symbol kề với nó, dẫn đến bậc của encoding symbol sẽ giảm đi 1, nếu encoding symbol đó có bậc mới là 1 thì sẽ tiếp tục giải mã để phủ input symbol kề với nó. Khi encoding symbol được giải mã thì đôi khi các input symbol được phủ không nằm trong ripple => kích cỡ ripple tăng trong khi với các input symbol được phủ vốn nằm trong ripple thì không làm thay đổi kích cỡ của ripple.

X

X

X

X

X

X

3

⊕

4

3

X

X

4

X

1. Quá trình kết thúc khi ripple rỗng sau 1 số bước nhất định. Quá trình thất bại nếu còn ít nhất 1 input symbol chưa được phủ vào cuối quá trình. Ngược lại quá trình thành công nếu tất cả input symbol đều được phủ vào cuối quá trình.
2. **Tối ưu hàm phân phối bậc**
3. *Ý tưởng:*

Vì chúng ta sử dụng phương pháp belief propagation nên cần phải thiết kế hầm phân phối bậc sao cho hiệu quả do đó phải có các encoding symbol có bậc 1 để quá trình giải mã có thể diễn ra.

1. *Phân phối đều (All At Once Distribution).*

Ý tưởng đầu tiên là tại sao chúng ta không thiết kế cho tất cả các bậc đều là *d = 1* tức là sẽ đảm bảo được việc giải mã thành công nếu như chúng ta nhận đủ các encoding symbol. Ta gọi xác suất để encoding symbol có bậc d trong phân phối này là *p(d) = 1*

Tuy nhiên nếu ta dùng hàm phân phối đều này thì để có thể phủ được *k* input symbols thì chúng ta phải cần tới ít nhất *k.ln(k/δ)* encoding symbols. ( kết quả này có được do việc phân tích cần ném ít nhất bao nhiêu trái banh vào *k* rổ sao cho mỗi rổ đều có banh ) với *1 – δ* là xác xuất để k input symbols được phủ hết.

Điều này cũng đồng nghĩa việc tổng các bậc của các encoding symbol là *k.ln(k/δ).*Do đó khi dùng phân phối đều này thì tuy số phép tính trên symbol là nhỏ nhưng phải cần tới *k.ln(k/δ)* encoding symbols mới có thể phủ hết được *k* input symbol tức là phải gấp tới *ln(k/δ)* lần. Đây là con số khó có thể chấp nhận được.

1. *Phân phối lý tưởng Soliton*
   1. *Ý tưởng*

Khi giải mã encoding *sy*mbol điều ta mong muốn là lượng encoding symbol phủ lên những input symbol đã có trong ripple là càng thấp càng tốt. Vì nếu encoding symbol phủ lên những symbol đã nằm trong ripple thì đó là những encoding symbol dư thừa. Ngược lại, nếu ripple quá nhỏ dẫn đến bị biến mất trước khi tất cả input symbol được xử lý thì sẽ dẫn đến quá trình thất bại. Từ đó ta thiết lập phân phối lý tưởng Soliton sao cho kích thước của ripple không quá nhỏ cũng không quá to.

* 1. *Xác suất giải mã bậc*

Ta gọi *q(i, L)* là xác suất giải mã của 1 encoding symbol bậc i khi còn L input symbol chưa được xử lý

Rõ ràng là ở bước đầu tiên quá trình giải mã.Những encoding symbol mang bậc 1 chắc chắn sẽ được giải mã nên hiển nhiên là q(1, k) = 1.

Do mỗi bước ta chỉ xử lý 1 input symbol nhằm giảm bậc của các encoding symbol có liên quan đến nó nên ta luôn có ở bước thứ m thì tất cả những encoding symbol có bậc ít nhất là m+2 tại thời điểm ban đầu quá trình giải mã sẽ không bao giờ giải mã được nên ta có dòng thứ 3 trong công thức ở trên.

Đối với những encoding symbol có bậc i không thuộc vào những trường hợp trên thì để nó có thể giải mã được khi còn L input symbol chưa được xử lý với L = k-i+1,…,1 thì nó phải phủ i – 1 imput symbol không nằm trong L input symbol chưa được xử lý và phủ lên 1 input symbol nằm trong L input symbol chưa được xử lý.Từ đó ta sẽ có xác xuất giải mã của những encoding symbol có bậc i khi còn L input symbol chưa được xử lý là

* 1. *Xác suất giải mã tổng quát*

Ta gọi *r(i, L)* là xác suất để encoding symbol được chọn có bậc *i* được giải mã khi còn *L* input symbol chưa được xử lý

*r(i, L) = p(i)⋅ q(i, L)*

Ta có *r(L)* là xác suất tổng quát để 1 encoding symbol được giải mã khi còn L input symbol chưa được xử lý

* 1. *Phân phối lý tưởng Soliton*

Ta có

Hiển nhiên = 1 thỏa điều kiện là hàm phân phối xác xuất.Ta cũng tính được kì vọng bậc của hàm phân phối trên là = H(k) ln(k).

Với hàm phân phối xác suất trên, ta thế vào công thức xác suất giải mã tổng quát thu được (với *r(L) = 1/k cho mọi L = k,…,1*)

* 1. *Nhược điểm*

Như vậy để có thể phủ mỗi input symbol trong *k* input symbol đúng 1 lần thì ta chỉ cần đúng k encoding symbol và theo công thức ở trên về hàm phân phối bậc ta chỉ có chính xác 1 encoding symbol bậc 1 vào thời điểm giải mã ban đầu và sau đó đúng 1 encoding symbol sẽ được giải mã đồng thời khi 1 input symbol được xử lý. Điều này tương đương với kích thước của ripple là 1, khi input symbol này được xử lý thì encoding symbol được giải mã sẽ lại phủ 1 input symbol không nằm trong ripple và tái lập lại hiện trạng.

Tuy nhiên chính điều này khiến cho phân phối Ideal Soliton không khả thi vì kích thước của ripple là 1, chỉ cần thay đổi nhỏ cũng có thể khiến cho ripple biến mất, từ đó quá trình thất bại.

Chú ý rằng tổng tất cả các bậc của k encoding symbol xấp xỉ k.ln(k).Quay lại với phân phối đều ta cũng cần k.ln(k) bậc để có thể phủ hết k input symbol.Trực giác cho thấy ứng với bất kì hàm phân phối bậc ta cũng đều cần phải có trung bình k.ln(k) bậc cho tất cả encoding symbol để có thể phủ hết k input symbol.

1. *Phân phối tối ưu hóa Soliton (Robust Soliton Distribution)*
   1. *Ý tưởng*

Do chính những nhược điểm của phân phối Ideal Soliton nên người ta phải tìm ra một hướng đi mới để có thể giải quyết được vấn đề của Ideal Soliton. Từ đó người ta đưa ra hàm phân phối Robust Soliton. Khác hẳn với phân phối Ideal Soliton, phân phối Robust Soliton đảm bảo kích cỡ mong muốn của ripple là đủ lớn để tại mọi thời điểm trong quá trình xác suất để ripple không bao giờ bị biến mất là đủ lớn.Bên cạnh đó, để giảm thiểu lượng encoding symbol sử dụng, ta cần chú ý đến việc giảm kích cỡ ripple mong muốn để không giải phóng quá nhiều encoding symbol phủ lại nhiều lần các input symbol vốn đã nằm trong ripple.

* 1. *Phân phối Robust Soliton*

Gọi *δ* là xác suất thất bại của quá trình giải mã dữ liệu từ *K* encoding symbol. Ta thiết kế hàm phân phối sao cho kích cỡ mong muốn của ripple tầm khoảng trong suốt quá trình. Dựa theo cảm tính ta cho 1 random walk có độ dài k không bị lệch quá với xác suất tối đa là *δ*. Như sẽ trình bày bên dưới, điều này có thể đạt được với encoding symbol.

Hàm phân phối Robust Soliton được định nghĩa như sau: gọi   
 với hằng số c > 0. Ta có

Thêm phân phối Ideal Soliton vào và chuẩn hoá nó ta thu được

Ban đầu đảm bảo khởi tạo ripple có kích thước vừa phải. Trong quá trình, giả sử ta đang xử lý 1 input symbol khi còn *L* input symbol chưa được xử lý. Vì kích cỡ ripple giảm đi 1 mỗi lần xử lý 1 input symbol, trung bình kích cỡ của ripple lại phải được tăng thêm 1 để bù lại. Nếu kích cỡ của ripple là *R* thì xác suất 1 encoding symbol được giải phóng thêm vào ripple chỉ là *(L-R)/L*, suy ra cần trung bình *L/(L-R)* encoding symbol được giải phóng để thêm vào ripple.Mặt khác từ công thức giải mã bậc ta có thể biến đổi được rằng xác xuất giải mã 1 encoding symbol có bậc là i với i xấp xỉ với giá trị k/L được khởi tạo tại thời điểm ban đầu khi còn L input symbol chưa được xử lý.Từ đó để ripple có thể duy trì ở mức R thì mật độ của những encoding symbol có bậc i = k/L phải tỉ lệ với

Với p(i) là xác xuất encoding symbol có bậc là i khi khởi tạo ban đầu.Thay từ công thức phân phối lý tưởng Soliton ta được:

Với i = 2,…,k/R – 1

đảm bảo rằng tất cả những input symbol chưa được xử lý khi L = R phải được phủ hết.Điều này đồng nghĩa với việc giải mã đồng thời Rln(R/δ) encoding symbol để phủ hết toàn bộ R input symbol chưa được xử lý.Như vậy, chi phí dư ra cho việc phủ lên mỗi input symbol ít nhất 1 lần bằng cách giải mã đủ số lượng encoding symbol chỉ là 1 phần rất nhỏ trong tổng số k input symbol.