وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ - زنجان

بازیابی بسته‌های آسیب‌دیده در کدگذاری شبکه

پایان‌نامه کارشناسی ارشد علوم کامپیوتر

حسین کنگاورنظری

استاد راهنما: دکتر پیمان پهلوانی

شهریور ۱۴۰۰

به نام مهربان

تشکر و قدردانی

با تقدیر و تشکر از تمامی استادان محترم دانشکده علوم‌کامپیوتر که از محضر علمی و اخلاقی‌شان بهرمند شدم. به ویژه آقای دکتر پیمان پهلوانی که به حق در طول دوره‌ی کارشناسی و کارشناسی ارشد صبورانه و مشفقانه بنده را راهنمایی کرده‌اند.

ماحصل آموخته‌هایم را به خانواده‌ام تقدیم می‌کنم که همیشه پشتیبان و مشوق من بوده‌اند.

چکیده

برنامه‌های نوظهور شبکه‌های کامپیوتری بیش از هر زمان دیگری به دنبال افزایش اطمینان و کاهش تاخیر در ارتباطات هستند. آسیب دیدن بسته‌ها در یک کانال ارتباطی خطادار اجتناب ناپذیر است، به همین دلیل افزایش اطمینان برای شبکه‌های بی‌سیم چالش برانگیز بوده است. در ابتدای سال ۲۰۰۰، کدگذاری شبکه به عنوان یک راهکار مناسب برای افزایش قابلیت اطمینان و استفاده بهینه از منابع شبکه مطرح شد. در این روش فرستنده با ترکیب‌کردن بسته‌ها به صورت خطی آنها را کدگذاری می‌کند. گیرنده نیز پس از دریافت تعداد کافی از بسته‌های کدشده، قادر به کدگشایی آنها است. به طور معمول در شبکه‌های کدگذاری شده بسته‌های آسیب‌دیده دور ریخته می‌شوند. با توجه به آنکه قسمت قابل توجه‌ای از بسته‌های آسیب‌دیده سالم هستند، دور ریختن آنها استراتژی بهینه‌ای نیست. به همین منظور روش‌های برای بازیابی این بسته‌ها به عنوان استراتژی مکمل ارائه شده‌اند. متاسفانه اغلب روش‌های ارائه شده دارای پیچیدگی محاسباتی بالا و باعث تاخیر زیادی در کدگشایی می‌شوند. در این مطالعه، دو روش برای بازیابی بسته‌های آسیب‌دیده معرفی می‌کنیم. ابتدا روش F-PRNC با هدف بهبود سرعت بازیابی و کاهش تاخیر ارائه می‌شود. به کمک شبیه‌سازی، روش پیشنهادی را با سایر روش‌های ارائه شده مقایسه کرده‌ایم. نتایج نشان می‌دهد که در یک کانال ارتباطی پرخطا، روش پیشنهادی مقدار گذردهی را بیش از ۴ برابر افزایش و تاخیر کدگشایی را تا ۴/۵ برابر کاهش می‌دهد. در ادامه روش دیگری بنام Fly-PRAC را معرفی می‌کنیم. روش پیشنهادی با ارائه تکنیکی امکان تخمین دقیق‌تر و بازیابی سریعتر بسته‌های آسیب‌دیده را فراهم می‌آورد. همچنین، Fly-PRAC اولین روشی است که قادر به بازیابی بسته‌ها در گره‌های میانی می‌باشد. به کمک شبیه‌سازی نشان می‌دهیم که در یک کانال ارتباطی پرخطا، روش ارائه شده قادر است زمان تکمیل بازیابی و کدگشایی را ۱۶ درصد کاهش و مقدار گذردهی را تا ۳/۸ برابر افزایش دهد.

کلمات کلیدی: بازیابی بسته‌های آسیب‌دیده، کدگذاری خطی تصادفی، کدگذاری شبکه

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

۱.۱ ارتباط مطمئن در شبکه‌های بی‌سیم

نرخ استفاده از شبکه‌های بی‌سیم در دهه‌های گذشته به طرز خیره‌کننده‌ای افزایش داشته است. این نوع از کانال ارتباطی به علت انعطاف‌پذیری بالا، برای طیف گسترده‌ای از ارتباطات مورد استفاده است. به عنوان مثال، شبکه‌های بی‌سیم با برد کوتاه مانند بلوتوث[۱] [۱]، زیگبی[۲] [۲]، و وای‌فای[۳] [۳] ؛ شبکه‌های با برد متوسط مانند نسل‌چهارم[۴] و پنجم[۵] ارتباطات و یا شبکه‌های دوربرد مانند LPWAN[۶] [۴] که برای اتصال میلیون‌ها دستگاه اینترنت‌اشیا[۷] در سراسر دنیا مورد استفاده هستند، نمونه‌های از کاربردهای ارتباطات بی‌سیم می‌باشند.

[1] Bluetooth
[2] Zigbee
[3] Wifi
[4] Fourth generation broadband cellular networks
[5] Fifth generation broadband cellular networks
[6] Low power wide area network
[7] Internet of thing

انعطاف‌پذیری شبکه‌های بی‌سیم باعث شده تا برنامه‌های متنوعی بر این بستر توسعه بیابند. این برنامه‌های متفاوت به علت تنوع در کاربرد می‌توانند به دنبال ویژگی‌های متنوع و گاهی متناقض باشند. به عنوان مثال ارتباطات بین ماشین‌های خودران از طریق شبکه‌های بی‌سیم صورت می‌گیرد. این نوع ارتباطات نیازمند تاخیر بسیار کم و قابلیت اطمینان بالا هستند. عموما با ایجاد یک ارتباط مطمئن[۸]، تاخیر در آن افزایش پیدا می‌کند، به همین دلیل افزایش اطمینان و کاهش تاخیر به طور همزمان چالش برانگیز می‌باشد.

طیف گسترده‌ای از برنامه‌ها نیازمند یک ارتباط مطمئن می‌باشند. یک ارتباط مطمئن، ارتباطی است که در آن بسته‌ها با ترتیب، غیرتکراری و صحیح دریافت شوند[۵]. هر کانال ارتباطی، بسته به ویژگی‌های فیزیکی آن رسانه، دارای یک ظرفیت گذردهی[۹] مشخصی می‌باشد. این ظرفیت به واسطه تضعیف،ازدحام سیگنال و انتشار چندمسیری[۱۰] می‌تواند کاهش پیدا کند[۶]. این مشکلات باعث آسیب بسته‌ها شده و ارتباط را نامطمئن می‌کند.

به طور سنتی، روش‌های برقراری ارتباط مطمئن به دو گروه درخواست ارسال مجدد[۱۱] یا تصحیح‌خطای پیشرو[۱۲] تقسیم می‌شوند. در روش درخواست ارسال مجدد، گیرنده با ارسال پیام‌های تصدیق از فرستنده درخواست می‌کند تا بسته‌های آسیب‌دیده را دوباره ارسال کند. در تصحیح‌خطای پیشرو، فرستنده اطلاعات تکراری هوشمندانه‌ای به هر بسته می‌افزاید. سپس در سمت گیرنده، در صورت آسیب دیدن بسته، از اطلاعات افزوده شده برای بازیابی آن استفاده می‌کند. هرکدام از این گروه‌ها دارای معایبی هستند. به عنوان مثال، روش درخواست ارسال مجدد برای شبکه‌های پرخطا صرفا باعث ازدحام بیشتر بسته‌ها شده و ظرفیت ارتباط را کاهش می‌دهد. روش تصحیح‌خطای پیشرو، با توجه به میزان اطلاعات افزوده، توانایی اصلاح محدودی داشته و با تغییر شرایط کانال یا باعث هدر رفت ظرفیت کانال شده، یا امکان بازیابی بسته‌ها را از دست می‌دهد[۷] [۸].

[8] Reliable

[9] Throughput

[10] Multipath propagation

[11] Automatic repeat request

[12] Forward error correction

کدگذاری شبکه[۱۳] به عنوان روشی برای استفاده بهینه از منابع شبکه و افزایش قابلیت اطمینان در سال ۲۰۰۰ معرفی شد. در این روش، کدگذار [۱۴] با ترکیب بسته‌ها با یکدیگر آنها را کد می‌کند. سپس، در سمت گیرنده پس از دریافت مقدار مناسبی بسته کدشده به کمک کدگشا[۱۵] عملیات کدگشایی صورت می‌گیرد. یکی از مهمترین مزیت‌های کدگذاری شبکه نسبت به سایر روش‌های کدگذاری، قابلیت کدگذاری مجدد[۱۶] بسته‌های کدشده در گره‌های میانی[۱۷] است.

روش کدگذاری خطی تصادفی[۱۸] [۹] یا به اختصار RLNC، از رایج ترین پیاده‌سازی‌های کدگذاری شبکه است. در این روش اطلاعات به بلوک‌های از داده به نام نسل[۱۹] تقسیم می‌شود. هر نسل شامل تعدادی بسته است. تعداد بسته‌های موجود در یک نسل، اندازه آن نسل را مشخص کرده و با نماد g نمایش داده می‌شود. هر بسته نیز شامل l نماد[۲۰] هم‌اندازه است که همگی آنها اعضای یک مجموعه‌ی متناهی[۲۱] هستند. این مجموعه متناهی را با $GF(۲^q)$ نمایش می‌دهیم.

در این روش برای کد کردن بسته‌ها، g بسته‌ی یک نسل با یکدیگر به صورت خطی ترکیب می‌شوند. برای این منظور از یک ماتریس ضرایب (C) استفاده می‌شود. برای ایجاد n بسته‌ی کدشده، ماتریس ضرایبی با g ستون و n ردیف نیاز خواهیم داشت. تمامی المان‌های این ماتریس به صورت تصادفی و مستقل از هم از $GF(۲^q)$ انتخاب می‌شوند. سپس برای ایجاد بسته‌های کدشده، ماتریس ضرایب با ماتریسی شامل g بسته‌ی یک نسل ضرب می‌شود. همچنین در سمت گیرنده برای کدگشایی کافی است تا معکوس ماتریس ضرایب را با بسته‌های دریافتی ضرب کنیم.

[13] Network coding
[14] Encoder
[15] Decoder
[16] Recoding
[17] Intermediate node
[18] Random Linear Network Coding
[19] Generation
[20] Symbol
[21] Finite field

۲.۱ بازیابی بسته‌های آسیب‌دیده

در کدگذاری شبکه، کدگشایی صرفا با بسته‌های بدون خطا صورت می‌گیرد. به همین منظور، عموما هر بسته‌ی کدشده با یک کدافزونگی چرخشی[۲۲]یا به اختصار CRC همراه می‌باشد. گیرنده می‌تواند با استفاده از CRC، صحت اطلاعات دریافتی را بررسی نماید. گیرنده بسته‌ای که صحت آن توسط CRC رد شود را دور می‌ریزد. مطالعات اخیر نشان می‌دهد که قسمت عمده‌ای از بسته‌ها در شبکه‌های پرخطا آسیب‌دیده هستند. به عنوان مثال، برای شبکه‌ی استاندارد وای‌فای[۲۳] نشان داده شده ۵۵ درصد کل بسته‌های دریافتی آسیب‌دیده هستند[۱۰]. همچنین محققان نشان داده‌اند که تا ۹۵ درصد اطلاعات موجود در یک بسته‌ی آسیب‌دیده، بدون خطا و کاربردی هستند[۱۱]. به این دلایل، دور ریختن بسته‌های آسیب‌دیده استراتژی مناسبی نیست.

در سالهای گذشته، چندین روش بازیابی بسته‌های آسیب‌دیده[۲۴] یا به اختصارPPR برای کدگذاری خطی تصادفی معرفی شده‌اند. این روش‌ها عموما از دو گام تخمین و اصلاح بخش‌های آسیب‌دیده تشکیل شده‌اند. برای گام اول، اغلب این روش‌ها از تکنیکی به نام قانون پایداری جبری [۲۵] یا به اختصار ACR برای تخمین نقاط آسیب‌دیده بهره می‌برند.

روش‌های PPR ارائه شده، در تئوری برای بازیابی بسته‌ها موثر هستند و می‌توانند ظرفیت گذردهی شبکه را بهبود ببخشند. هرچند از معایب مشترکی نیز رنج می‌برند. به عنوان مثال، روش‌های برپایه ACR عملیات بازیابی را صرفا بعد از دریافت $g + ۱$ بسته شروع کرده که منجر به تاخیر زیادی در تکمیل بازیابی و کدگشایی می‌شود. این روش‌ها در کانال‌های پرخطا سرعت بازیابی داشته و عملیات بازیابی را صرفا در گیرنده انجام می‌دهند. با توجه به آنکه بسته‌های آسیب‌دیده نمی‌توانند در فرآیند کدگذاری مجدد شرکت کنند، این روش‌ها قادر نیستند تا از ظرفیت کدگذاری شبکه به طور کامل بهره ببرند.

[22]Cyclic redundancy code

[23]IEEE 802.11a/b/g

[24]Partial packet recovery

[25]Algebraic consistency rule

۳.۱ سهم علمی پایان‌نامه

در این مطالعه، ۲ روش برای بازیابی بسته‌های آسیب‌دیده ارائه می‌شوند. در ابتدا روش F-PRNC را با هدف کاهش زمان بازیابی و افزایش گذردهی برای شبکه‌ای با خطاهای قطاری[۲۶] معرفی می‌کنیم. در این روش به هر بسته‌کد شده تعدادی نماد افزونگی[۲۷] اضافه می‌شود تا هنگام بازیابی مورد استفاده قرار بگیرد. در روش پیشنهادی، فرآیند بازیابی در دو گام صورت می‌گیرد. گام اول بدون نیاز به تخمین نقاط آسیب‌دیده و پس از دریافت اولین بسته‌ی آسیب‌دیده آغاز می‌گردد. گام دوم پس از دریافت $g + ۱$ بسته شروع می‌شود. ابتدا با استفاده از ACR تخمینی از نقاط آسیب‌دیده تهیه شده و عملیات بازیابی به کمک این اطلاعات ادامه می‌یابد. نتایج آزمایش‌ها و شبیه‌سازی‌ها نمایانگر آن است که در شرایط پرخطا و در مقایسه با روش‌های پیشین، F-PRNC می‌تواند منجر به بهبود گذردهی و کاهش سربار[۲۸] شبکه شود.

در ادامه، روش دیگری به نام Fly-PRAC را معرفی می‌کنیم. در این روش تکنیک جدیدی برای تخمین نقاط آسیب‌دیده بر پایه روابط جبری گروهی از بسته‌های وابسته‌ی خطی[۲۹] معرفی می‌شود. این روش قادر است تا تخمین نقاط آسیب‌دیده را قبل از دریافت $g + ۱$ بسته و برای گروه‌ی از بسته‌های وابسته فراهم آورد. همچنین Fly-PRAC، اولین روشی است که قادر است در گره‌های میانی بسته‌های آسیب‌دیده را بازیابی کند. Fly-PRAC می‌تواند با بسته‌های بازیابی شده در گره‌میانی، عملیات کدگذاری مجدد را برای تعداد بسته‌های بیشتری انجام داده و از ظرفیت کدگذاری شبکه بیشتر بهره‌مند گردد. این روش به کمک شبیه‌سازی با آخرین PPR موجود به نام S-PRAC [۱۲] مقایسه شده است. نتایج شبیه‌سازی نشان می‌دهد که روش ارائه شده می‌تواند در نرخ خطای‌بیت[۳۰] بالا مانند ۵×۱۰^{-۵}، زمان تکمیل بازیابی را تا حدود ۴/۷ برابر کاهش دهد. همچنین نتایج بیانگر آن است که برای بسته‌ای با اندازه ۹۰۰ بایت، مقدار گذردهی تا ۳/۸

[26] Burst error

[27] Redundancy symbol

[28] Overhead

[29] Linearly dependent packets

[30] Bit error rate

برابر افزایش می‌یابد. این روش می‌تواند تا بسته‌های بیشتری را بازیابی کرده و با کاهش تعداد ارسال‌ها، سربار شبکه را نیز کاهش دهد.

فصل ۲

مرور کارهای پیشین

روش‌های متنوعی برای بازیابی بسته‌های آسیب‌دیده تاکنون معرفی شده‌اند. یکی از مهمترین بخش‌های هر روش PPR، نحوه تشخیص قسمت‌های آسیب‌دیده بسته‌ی دریافتی است. طیفی از روش‌ها مانند[۱۳]، این اطلاعات را از لایه‌ی فیزیکی شبکه دریافت می‌کنند. با توجه به اینکه وظیفه هرلایه در معماری استاندار ۷ لایه‌ای شبکه [۱] از پیش مشخص شده و دریافت چنین اطلاعاتی از لایه فیزیکی منظور نشده است، برای پیاده‌سازی چنین روش‌های نیاز به تغییر در سطح سخت‌افزار است[۱۴]. گروه دیگری از روش‌ها به اطلاعات موجود در لایه‌ی پیوند داده [۲] یا بالاتر اکتفا کرده و از آنها برای شناسایی بخش‌های آسیب‌دیده استفاده می‌کنند. روش‌های PPR موجود برای کدگذاری شبکه در دسته دوم قرارگرفته و از ارتباطات جبری میان بسته‌های کدشده برای تخمین بخش‌های آسیب‌دیده استفاده می‌کنند.

در این بخش به صورت خلاصه، روش‌های PPR موجود برای کدگذاری شبکه را که از قانون پایداری جبری استفاده می‌کنند معرفی کرده و به نکات مثبت و منفی آنها اشاره خواهیم کرد.

[1] Open systems interconnection model

[2] Data link layer

۱.۲ پایداری جبری بسته‌های بدون نرخ

اولین روش معرفی شده برای بازیابی بسته‌های آسیب‌دیده در کدگذاری خطی در [۱۵] معرفی شد. نام این روش پایداری جبری بسته‌های بدون نرخ[3] یا به اختصار PRAC می‌باشد. این روش برای تشخیص قسمت‌های آسیب‌دیده بسته‌های دریافتی، از روش ACR که بر پایه روابط جبری بین بسته‌های کدشده می‌باشد، استفاده می‌کند. در این روش برای شروع عملیات بازیابی، ابتدا نیاز است تا گیرنده حداقل $g+1$ بسته (سالم یا آسیب‌دیده) دریافت نماید. سپس، با انتخاب g بسته، عملیات کدگشایی را انجام داده و بسته‌های حاصل شده را بسته‌های پیش-کدگشایی‌شده [4] می‌نامد. این بسته‌ها تخمینی از بسته‌های کدگشایی شده می‌باشند و صرفا برای بررسی پایداری اطلاعات موجود در بسته کاربرد دارند. سپس، بسته‌ی دریافت شده‌ی دیگری که در ایجاد بسته‌های پیش-کدگشایی‌شده دخیل نبوده، انتخاب شده و به کمک وکتور ضرایب [5] آن بسته و نمادهای موجود در بسته‌های پیش-کدگشایی‌شده، بسته را کدگذاری مجدد کرده و آن را بسته تخمینی[6] می‌نامد. سپس بسته‌تخمینی با نسخه دریافتی آن بسته را نماد به نماد مقایسه شده و در صورتی که بین نمادها تفاوتی باشد، آن قسمت به عنوان بخش ناپایدار معرفی می‌شود. برای بازیابی بسته‌های آسیب‌دیده، مقادیر نمادهای موجود در قسمت‌های ناپایدار تغییر داده شده و پس از هر تغییر به همین روش صحت اطلاعات آن بررسی می‌شود. این تغییرات تا زمانی که پایداری اطلاعات برای تمامی بخش‌های بسته تایید گردد، ادامه پیدا می‌کند.

این روش، سربار اضافی برای شبکه ندارد و در تئوری قادر است تا ۹۰ درصد بسته‌های آسیب‌دیده را اصلاح نماید [۱۰]. از مهمترین معضلات این روش می‌توان به سرعت پایین در بازیابی و تاخیر قابل توجه برای تخمین نقاط آسیب‌دیده اشاره کرد. زمان صرف شده برای بازیابی بسته‌ها با افزایش نرخ‌خطای شبکه، افزایش تعداد نمادهای یک بسته و یا افزایش تعداد بسته‌های یک نسل، به طرز قابل توجه‌ای افزایش می‌یابد. همچنین در فرآیند ACR امکان آن وجود دارد که بخشی آسیب‌دیده از

[3] Packetized rateless algebraic consistency

[4] Pre-decododed

[5] Coefficient vector

[6] Estimated packet

بسته‌ها به اشتباه تشخیص داده نشود که از به عنوان رویداد مثبت-کاذب[7] یاد می‌کنیم.

۲.۲ روش بازیابی DAPRAC

دومین روشی که بر پایه ACR سعی در بازیابی بسته‌های آسیب‌دیده دارد در [۱۶] معرفی شد. این روش DAPRAC نام دارد و هدف آن کاهش زمان فرآیند بازیابی است. به این منظور، ابتدا برای تمامی بسته‌های یک نسل، یک CRC محاسبه شده و به آن افزوده می‌شود. سپس، پس از کدگذاری بسته‌ها، برای هر بسته کدشده نیز یک CRC محاسبه و به آن افزوده می‌شود. فرآیند بازیابی بسته‌ها، پس از دریافت $g+۱$ بسته شروع می‌شود. در این روش، با هر ترکیب g-تایی ممکن از $g+۱$ بسته دریافتی عملیات کدگشایی صورت می‌گیرد. به ازای هر ترکیب، یک گروه از بسته‌های پیش-کدگشایی شده حاصل می‌شود. به عنوان مثال، برای یک نسل با دو بسته، سه بسته‌کد شده دریافت شده و ۶ گروه از بسته‌های پیش-کدگشایی شده حاصل می‌شود. در این مثال، در هر گروه از بسته‌های پیش-کدگشایی ۲ بسته وجود دارد که تخمینی از بسته‌های کدگشایی شده هستند. برای مشخص کردن نقاط آسیب‌دیده بسته‌ها، بسته‌های تخمینی موجود در هر گروه، نماد به نماد با بسته‌های متناظر سایر گروه‌ها مقایسه می‌شود. در صورتی که نمادها با یکدیگر برابر نباشد، آن نمادها به عنوان نماد ناپایدار معرفی می‌شوند. پس از اتمام عملیات شناسایی نقاط ناپایدار، یک گروه از بسته‌های پیش-کدگشایی انتخاب می‌شود. سپس، به ازای تمامی بسته‌های موجود در گروه که حداقل دارای یک نماد ناپایدار هستند، عملیات اصلاح بسته انجام می‌شود. در این فرآیند، نمادهای ناپایدار بسته، با نمادهای متناظر از گروه‌های دیگر جایگزین شده و پس از آن صحت اطلاعات بسته به کمک CRC موجود در بسته بررسی می‌گردد. این فرآیند جایگزنی نمادها ادامه یافته تا زمانی که صحت داده‌های بسته توسط CRC تایید شود.

روش DAPRAC درمقایسه با PRAC، در نرخ خطای متوسط قادر است تا سرعت بازیابی را بهبود دهد. البته فرض‌های در این روش وجود دارد که دامنه عملکرد آن را محدود می‌سازد. به عنوان مثال، آسیب‌دیدن نمادهای یکسان در

[7] False positive

بسته‌های دریافتی، فرآیند بازیابی را مختل می‌کند [۱۶]. هرگونه خطا در CRCها نیز عملیات بازیابی را غیرممکن می‌سازد. در نرخ خطای کم، عملیات بازیابی به دلیل اینکه باید کدگشایی را بارها انجام دهد، بسیار زمان‌بر بوده و هزینه محاسباتی بالایی نیز دارد.

۳.۲ روش بازیابی S-PRAC

آخرین روش ارائه شده برای بازیابی‌بسته‌های آسیب‌دیده در کدگذاری خطی تصادفی تا زمان نگارش این مطالعه، S-PRAC می‌باشد[۱۲]. در این روش، هر بسته کدشده به s قسمت[۸] برابر تقسیم شده و برای هر قسمت یک CRC محاسبه و افزوده می‌شود. هر قسمت شامل یک یا چند نماد کدشده است. سپس یک CRC دیگر برای کل نمادهای کدشده محاسبه و به بسته افزوده می‌شود. در سمت گیرنده پس از دریافت $g + ۱$ بسته، ACR بخش‌های آسیب‌دیده هر بسته را مشخص می‌نماید. سپس برای اصلاح بسته‌های آسیب‌دیده، گیرنده سعی می‌کند با تغییر مقدار بخش‌های معرفی شده توسط ACR آنها را بازیابی کند. این عملیات به صورت جداگانه برای هر قسمت انجام شده و برای بررسی صحت اطلاعات تغییر داده شده از CRC متعلق به همان قسمت استفاده می‌شود. انجام تغییرات و بررسی هر قسمت از بسته به صورت جداگانه باعث می‌شود تا عملیات اصلاح بخش‌های آسیب‌دیده با تغییرات کمتری به نتیجه رسیده و عملیات بازیابی با سرعت بیشتری انجام شود. به عبارت دیگر، این روش می‌تواند سرعت بازیابی را به نسبت PRAC و DAPRAC در شرایط پرخطا بهبود دهد. هر چند، در این روش نیز حداقل $g + ۱$ بسته برای شروع فرآیند بازیابی نیاز است. به همین دلیل، تاخیر در بازیابی و کدگشایی نیز در این روش نسبتا بالا است. مشکل دیگر، امکان رخ دادن رویداد مثبت-کاذب در فرآیند اصلاح بسته است. در این فرآیند امکان آن وجود دارد که CRC صحت داده را برای اطلاعات دارای خطا تایید کند و فرآیند بازیابی را با مشکل مواجه کند. چنین رویدادی در نرخ خطای بالاتر، بیشتر رخ داده و باعث کاهش گذردهی کانال ارتباطی می‌شود[۱۷].

[8] Segment

۴.۲ جمع‌بندی و نتیجه‌گیری

نقاط ضعف مشترکی در روش‌های PPR ارائه شده برای کدگذاری شبکه وجود دارد. یکی از مهمترین این مشکلات، تاخیر زیاد در فرآیند بازیابی بسته‌ها است. مورد دیگر، رویدادهای مثبت-کاذب در فرآیند ACR و اصلاح بسته با CRC می‌باشد. چنین رویدادهایی می‌تواند عملیات بازیابی را با مشکل مواجه کند. احتمال این رویدادها نیز با افزایش نرخ خطا بیشتر شده و درصد کمتری از بسته‌های آسیب‌دیده در این شرایط بازیابی می‌شوند. همچنین، تمامی روش‌های ذکر شده عملیات بازیابی را صرفا در گره گیرنده انجام داده و نمی‌توانند از تمام ظرفیت کدگذاری شبکه بهرهمند شوند.

فصل ۳

روش‌های پیشنهادی

در این بخش به صورت خلاصه دو روش برای PPR در شبکه‌های کدگذاری شده ارائه می‌شود. ابتدا روشی به نام F-PRNC با هدف کاهش تاخیر در کدگشایی و مقابله با خطاهای قطاری معرفی می‌کنیم. مراحل کدگذاری، بازیابی و کدگشایی این روش را به صورت خلاصه بررسی خواهیم کرد. سپس روش Fly-PRAC را معرفی می‌کنیم. این روش با هدف شروع سریعتر فرآیند بازیابی و برای مقابله با تک خطاهای بیتی [1] معرفی می‌شود. فرآیند کدگذاری، بازیابی و کدگشایی این روش را نیز به صورت خلاصه در این فصل بررسی خواهیم کرد.

۱.۳ روش F-PRNC

۱.۱.۳ کدگذاری

فرآیند کدگذاری برای یک نسل شامل g بسته $\{p_۱, p_۲, ..., p_g\}$ صورت خواهد گرفت. هر بسته خود شامل l نماد است. به عنوان مثال، بسته p_i دارای نمادهای $\{s_{i,۱}, s_{i,۲}, ..., s_{i,l}\}$ است. تمامی نمادها دارای q بیت داده بوده و عضوی از

[1] Single Bit Error

مجموعه‌ی متناهی $GF(۲^q)$ می‌باشند. ابتدا یک ماتریس شامل بسته‌های یک نسل ایجاد شده که آن را با نماد P نشان می‌دهیم. همچنین ماتریس دیگری به نام ماتریس ضرایب (C) ایجاد می‌شود که تمامی اعضای این ماتریس به صورت مستقل از هم و تصادفی از مجموعه‌ی متناهی $GF(۲^q)$ انتخاب می‌شوند. سپس، برای تولید n بسته کدشده از فرمول ۱.۳ استفاده می‌شود. ماتریس حاصل شده (P') حاوی بسته‌های کدشده می‌باشد.

$$P' = C_{(n\times g)} \times P_{(g\times l)}. \tag{۱.۳}$$

تصور کنید p'_i نمایانگر بسته کدشده i-ام باشد. به ازای هر بسته کدشده، l نماد کدشده وجود دارد. به عنوان مثال، p'_i شامل نمادهای $\{s'_{i,۱}, s'_{i,۲}, ..., s'_{i,l}\}$ می‌باشد. پس از تولید بسته‌های کدشده، برای هر یک از آنها R' نماد افزونگی تولید می‌شود. هر نماد افزونگی در واقع ترکیبی خطی از نمادهای کدشده درون یک بسته کدشده می‌باشد. به این منظور، یک ماتریس ضرایب برای بسته‌های کدشده نیز در نظر گرفته می‌شود که آن را با C' نشان می‌دهیم. هر یک از المان‌های این ماتریس نیز به صورت مستقل از هم و تصادفی از مجموعه‌ی متناهی $GF(۲^q)$ انتخاب می‌شوند. برای تولید نمادهای افزونگی از فرمول ۲.۳ استفاده می‌شود. مجموعه $\{r_{i,۱}, r_{i,۲}, ..., r_{i,R'}\}$ بیانگر نمادهای افزونگی بسته i-ام می‌باشند.

$$C'_{(R'\times l)} \times \begin{pmatrix} s'_{i,۱} \\ s'_{i,۲} \\ \vdots \\ s'_{i,l} \end{pmatrix}_{(l\times ۱)} = \begin{pmatrix} r_{i,۱} \\ r_{i,۲} \\ \vdots \\ r_{i,R'} \end{pmatrix}_{(R'\times ۱)} . \tag{۲.۳}$$

سپس، برای هر بسته کدشده یک CRC به نام CRCs برای نمادهای کدشده درون هر بسته محاسبه شده و به آن افزوده می‌شود. ضرایب مربوط به هر بسته یا به آن بسته افزوده شده یا گیرنده و فرستند تابع تولید کننده ضرایب تصادفی را با

یکدیگر هم‌میزان[۲] می‌کنند. در این مطالعه، حالت دوم در نظر گرفته شده است.

۲.۱.۳ فرآیند بازیابی و کدگذاری

فرآیند کدگذاری شامل دو فرآیند بازیابی به نام‌های LR[۳] و HR[۴] می‌باشد. در ادامه این دو فرآیند به صورت خلاصه معرفی خواهند شد.

۱.۲.۱.۳ فرآیند بازیابی **LR**

پس از دریافت هر بسته کدشده، به کمک CRCs درون بسته، صحت داده دریافتی بررسی می‌شود. در صورتی که صحت نمادهای کدشده در بسته تایید شود، آن بسته به بافر[۵] بسته‌های سالم افزوده می‌شود. در صورتی که بسته آسیب‌دیده تشخیص داده شود، مراحل بازیابی برای آن آغاز می‌گردد. برای بازیابی یک بسته آسیب‌دیده، کافی است تا بتوان l نماد سالم از میان نمادهای کدشده و افزونگی یافته شود. با توجه به آنکه نمادهای افزونگی ترکیبی خطی از نمادهای کدشده در هر بسته هستند، می‌توان به کمک اطلاعات ضرایب آنها نمادهای کدشده را بازیابی کرد. در واقع در هر مرحله، یک زیر مجموعه l تایی از $l + R'$ نماد دریافتی انتخاب می‌شود. در صورتی که در این زیر مجموعه، تعدادی از نمادهای انتخابی از نمادهای افزونگی باشند، به کمک اطلاعات ضرایب آنها و سایر نمادهای کدشده، می‌توان نمادهای کدشده ای را که در زیرمجوعه‌ی انتخابی حضور ندارند، محاسبه نمود. سپس، صحت اطلاعات به دست آمده به کمک CRCs درون بسته سنجیده می‌شود. در صورتی که هیچ یک از نمادها در زیرمجموعه انتخابی آسیب‌دیده نباشند، صحت اطلاعات تایید خواهد شد و بسته بازیابی شده است. در غیر این صورت، زیر مجموعه‌ی دیگری انتخاب خواهد شد و تمامی فرآیندهای ذکر شده تکرار می‌گردد. در LR، این فرآیند حداکثر t بار با زیرمجموعه‌های متفاوت تکرار می‌شود. در صورتی که در t بار تلاش بسته بازیابی نشود، بسته به بافر بسته‌های آسیب‌دیده اضافه خواهد شد.

[2] Synchronize

[3] Limited recovery phase

[4] Hinted recovery phase

[5] Buffer

۲.۲.۱.۳ تخمین قسمت‌های آسیب‌دیده با **ACR**

پس از آنکه $g+۱$ بسته دریافت شد، فرآیند تخمین نمادهای کدشده آسیب‌دیده آغاز می‌شود. روش پیشنهادی برای انجام این کار از ACR استفاده می‌کند. در گام اول، یک ماتریس شامل $g+۱$ بسته دریافتی به نام U' ایجاد می‌گردد. این عملیات به ازای هر ستون U' به صورت جداگانه انجام می‌شود. در صورتی که صحت نمادهای کدشده آن ستون تایید نشود، به این معنی است که حداقل یکی از نمادهای کدشده حاضر در آن ستون آسیب‌دیده است. به عنوان مثال، برای بررسی ستون j-ام U'، وکتوری شامل نمادهای کدشده حاضر در آن ستون انتخاب می‌شود. سپس، معکوس ماتریس ضرایب C متناظر با بسته‌های دریافتی را با این وکتور ضرب می‌کنیم. نمادهای کدشده حاصله را $\{s^*_{i,۱}, s^*_{i,۲}, ..., s^*_{i,l}\}$ نمادهای پیشبینی‌شده می‌نامیم.

$$\begin{pmatrix} s^*_{۱,j} \\ s^*_{۲,j} \\ \vdots \\ s^*_{g,j} \end{pmatrix} = \begin{pmatrix} c_{۱,۱} & c_{۱,۲} & \cdots & c_{۱,g} \\ c_{۲,۱} & c_{۲,۲} & \cdots & c_{۲,g} \\ \vdots & \vdots & \ddots & \vdots \\ c_{g,۱} & c_{g,۲} & \cdots & c_{g,g} \end{pmatrix}^{-۱} \times \begin{pmatrix} s'_{۱,j} \\ s'_{۲,j} \\ \vdots \\ s'_{g,j} \end{pmatrix}. \tag{۳.۳}$$

سپس، بسته $g+۱$ را به کمک ضرایب آن و نمادهای پیشبینی‌شده کدگذاری کرده و آن را با $s''_{(g+۱),j}$ نمایش می‌دهیم.

$$s''_{(g+1),j} = \begin{pmatrix} c_{(g+1),1} & \cdots & c_{(g+1),g} \end{pmatrix} \times \begin{pmatrix} s^*_{1,j} \\ s^*_{2,j} \\ \vdots \\ s^*_{g,j} \end{pmatrix} \tag{۴.۳}$$

باتوجه به آنکه بسته پیشبینی‌شده و بسته‌دریافتی $g+1$-ام، هر دو از یک مجموعه ضرایب برای کدشدن استفاده کرده‌اند، مقدار آنها باید باهم برابر باشد. پس اگر $s''_{(g+1),j} = s'_{(g+1),j}$ برقرار باشد، صحت ستون j-ام تایید می‌شود. در غیر این صورت، تمامی نمادهای کدشده در این ستون به عنوان نمادهای آسیب‌دیده تخمین‌زده می‌شود. پس از آنکه این فرآیند برای تمامی ستون‌های شامل نمادهای کدشده در U' انجام شد، لیستی از موقعیت ستون‌های ناپایدار به مرحله بازیابی بعدی تحویل داده می‌شود.

۳.۲.۱.۳ فرآیند بازیابی **HR**

در این مرحله، بازیابی برای بسته‌های آسیب‌دیده‌ای که در مرحله پیشین بازیابی نشده‌اند انجام می‌شود. برای این منظور، ابتدا یک بسته از بافر بسته‌های آسیب‌دیده انتخاب می‌شود. مشابه مرحله بازیابی قبلی، باید l نماد سالم از میان تمامی نمادهای موجود انتخاب شود. درگام اول، تمامی نمادهای کدشده‌ای که در ستون‌های تایید شده هستند، انتخاب می‌شوند. فرض کنید N_{vs} نماینگر تعداد ستون‌های تاییدشده است. در این گام، گیرنده تلاش می‌کند تا $l - N_{vs}$ نماد دیگر را از ستون‌های آسیب‌دیده و نمادهای افزنگی پیدا کند. به همین منظور، زیرمجموعه‌های $l - N_{vs}$ عضوی از ستون‌های آسیب‌دیده و نمادهای افزنگی انتخاب می‌کند. مشابه LR نمادهای کدشده را به دست آورده و صحت آنها را با CRC بررسی می‌کند. این فرآیند تا زمانی که صحت زیرمجموعه انتخابی تایید نشود، ادامه پیدا می‌کند.

بدیهی است که برای بسته‌های دارای بیش از $R'+1$ نماد آسیب‌دیده، هیچ زیرمجموعه‌ای تایید شده‌ای وجود ندارد. این

بسته‌ها قابل بازیابی نیستند و دورریخته خواهندشد. سپس گیرنده برای بسته‌های دور ریخته‌شده درخواست بسته‌ی جایگزین می‌کند.

۲.۳ روش Fly-PRAC

۱.۲.۳ کدگذاری

فرآیند کدگذاری برای یک نسل شامل g بسته $\{p_۱, p_۲, ..., p_g\}$ صورت خواهد گرفت. هر بسته خود شامل l نماد است. به عنوان مثال، بسته p_i دارای نمادهای $\{s_{i,۱}, s_{i,۲}, ..., s_{i,l}\}$ است. هر نماد دارای q بیت داده و عضوی از مجموعه‌ی متناهی $GF(۲^q)$ می‌باشد. ابتدا یک ماتریس به نام P شامل بسته‌های یک نسل ایجاد می‌شود.

$$P = \begin{pmatrix} s_{۱,۱} & s_{۱,۲} & \cdots & s_{۱,l} \\ s_{۲,۱} & s_{۲,۲} & \cdots & s_{۲,l} \\ \vdots & \vdots & \ddots & \vdots \\ s_{g,۱} & s_{g,۲} & \cdots & s_{g,l} \end{pmatrix}. \tag{۵.۳}$$

برای تولید هربسته کدشده نیاز به یک وکتور ضرایب است. هر وکتور ضرایب شامل g المان [۶] است. هر المان به صورت مستقل از هم و تصادفی از مجموعه‌ی متناهی $GF(۲^q)$ انتخاب می‌شود. بسته‌های کدشده در قالب گروه‌های بسته‌های وابسته ایجاد می‌شود. در هر گروه R بسته کدشده وجود دارد و برای یک نسل از بسته‌ها، حداقل $\frac{g}{R-۱}$ گروه ایجاد می‌شود.

برای تولید یک گروه وابسته، ابتدا $R - ۱$ وکتور ضرایب ایجاد می‌شود. c_j^i نمایانگر وکتور ضرایب بسته‌ی j-ام از گروه i-ام است. سپس ماتریسی شامل تمامی این وکتورهای ضرایب تولید می‌شود به صورتی که هر سطر این ماتریس دارای

[6] Element

المان‌های یکی از این وکتورها است. C^i نمایانگر ماتریس ضرایب گروه i-ام است.

$$C^i = \begin{pmatrix} c^i_1 \\ c^i_2 \\ \dots \\ c^i_{R-1} \end{pmatrix}. \tag{۶.۳}$$

برای تولید $R-1$ بسته کدشده اول، ماتریس ضرایب آن گروه در ماتریس P ضرب می‌شود.

$$P'_i = C^i_{(R-1\times g)} \times P_{(g\times l)}. \tag{۷.۳}$$

سپس برای تولید بسته آخر گروه، وکتورهای ضرایب بسته‌های متعلق به یک گروه به کمک فرمول ۸.۳ با یکدیگر ترکیب می‌شوند. بسته‌ای که به کمک وکتور ضرایب حاصل شده تولید می‌شود، درواقع به سایر بسته‌های کدشده آن گروه به صورت خطی وابسته است.

$$c^i_R = c^i_1 \oplus c^i_2 \oplus \dots \oplus c^i_{R-1}. \tag{۸.۳}$$

برای تولید بسته‌ی کدشده آخر، وکتور ضرایب حاصل شده با ماتریس P ضرب می‌شود.

$$p'_{iR} = c^i_R \times P, \tag{۹.۳}$$

به صورتی که p'_{iR} نماینگر آخرین بسته‌ی کدشده‌ی گروه i-ام است. فرض کنید p'_i نمایانگر بسته‌ی کدشده‌ی i-ام باشد. هر p'_i شامل l نماد کدشده $\{s'_{i,1}, s'_{i,2}, ..., s'_{i,l}\}$ می‌باشد. سپس، هربسته کدشده به s قسمت مساوی تقسیم می‌شود. برای هر قسمت، یک CRC به نام ICRC محاسبه شده و به بسته افزوده می‌شود. همچنین یک CRC به نام OCRC برای تمامی نمادهای کدشده یک بسته محاسبه شده و به بسته افزوده می‌شود. به عبارت دیگر، در هر بسته کدشده به ازای هر قسمت یک ICRC و یک OCRC برای کل بسته محاسبه خواهدشد.

همچنین هر بسته کدشده باید وکتور ضرایب خود را با خود حمل کند یا تولید کننده ضرایب تصادفی بین گیرنده و فرستنده باید همزمان شوند. در این مطالعه، حالت دوم را در نظر گرفته‌ایم.

۲.۲.۳ بازیابی

پس از دریافت هر بسته، صحت نمادهای کدشده دریافتی به کمک OCRC بررسی می‌شود. در صورتی که OCRC آن بسته را تایید کند، بسته به بافر بسته‌های سالم افزوده می‌شود. در صورتی که OCRC صحت نمادها را تایید نکند، آن بسته به بافر بسته‌های آسیب‌دیده افزوده می‌شود. با توجه به آنکه در هر گروه یک بسته به سایر بسته‌ها وابسته‌ی خطی می‌باشد، پس از دریافت تمامی بسته‌های متعلق به یک گروه، اگر تنها یک بسته از آن گروه در بافر بسته‌های آسیب‌دیده باشد، آن بسته دور ریخته می‌شود. در صورتی که دو بسته یا بیشتر در یک گروه آسیب‌دیده باشند، فرآیند بازیابی برای آنها به جریان می‌افتد. این فرآیند شامل دو گام تخمین نمادهای آسیب‌دیده و اصلاح آنها می‌باشد که در ادامه آنها را به اختصار شرح می‌دهیم.

۱.۲.۲.۳ تخمین نمادهای آسیب‌دیده

در گام اول بازیابی، تلاش می‌شود تا به کمک روابط جبری و خطی موجود بین بسته‌های یک گروه وابسته، تخمینی از محل های آسیب‌دیده‌ی فراهم گردد. این فرآیند پس از دریافت تمامی بسته‌های یک گروه آغاز می‌گردد. برای این کار، دو ماتریس با نام‌های C و S تشکیل می‌شوند. هر سطر از ماتریس C، المان های مربوط به یک بسته‌ی کدشده را در خود جای داده‌است.

$$C = \begin{pmatrix} c_{i,1} & c_{i,1} & \cdots & c_{i,g} \\ c_{i+1,1} & c_{i,1} & \cdots & c_{i+1,g} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i+R,1} & c_{i,1} & \cdots & c_{i+R,g} \end{pmatrix}. \tag{۱۰.۳}$$

همچنین هر سطر از ماتریس S دارای نمادهای کدشده یک بسته می‌باشد.

$$S = \begin{pmatrix} s'_{i,1} & s'_{i,2} & \cdots & s'_{i,l} \\ s'_{i+1,1} & s'_{i+1,2} & \cdots & s'_{i+1,l} \\ \vdots & \vdots & \ddots & \vdots \\ s'_{i+R,1} & s'_{i+R,2} & \cdots & s'_{i+R,l} \end{pmatrix}. \tag{۱۱.۳}$$

ابتدا این دو ماتریس را به یکدیگر الحاق می‌کنیم و آن را Y می نامیم. به عبارت دیگر، در هر سطر g المان اول وکتور ضرایب، و سایر المان‌های آن سطر نمادهای کدشده حاصل‌شده از آن وکتور ضرایب می‌باشند.

در ادامه، ماتریس Y را به کمک الگوریتم گاوس-جردن[۷] ،به صورت سطری پلکانی کاهش یافته[۸] در می‌آوریم. با توجه به آنکه در ماتریس C یکی از ضرایب وابسته خطی می‌باشد، پس از انجام این عملیات بر روی ماتریس Y، در یکی از دریف‌های آن g المان اول صفر خواهدبود. اگر تمامی المان‌های یک وکتور ضرایب همگی صفر باشند، تمامی نمادهای کدشده حاصل از آن نیز صفر هستند. درصورتی که برخی از نمادهای کدشده آسیب‌دیده باشند، امکان آن وجود دارد که برخی از المان‌ها در

[7] Gauss Jordan

[8] Reduced Row Echelon Form

آن سطر غیر صفر باشند. به عبارت دیگر، این نشان دهنده آن است که حداقل یک نماد کدشده آسیب‌دیده در آن ستون از ماتریس Y می‌باشد. ستون‌های با چنین ویژگی، به عنوان ستون‌های ناپایدار معرفی شده و نمادهای حاضر در این ستون‌ها به عنوان نمادهای تخمینی آسیب‌دیده معرفی می‌شوند.

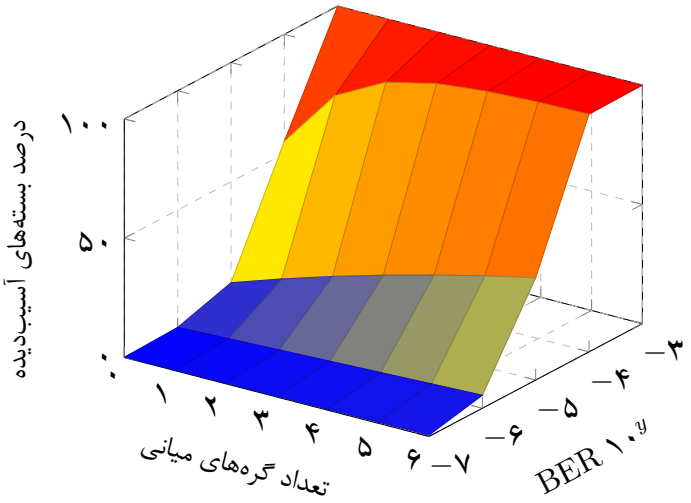
۲.۲.۲.۳ اصلاح نمادهای آسیب‌دیده

هدف از این قسمت، بازیابی بسته‌های آسیب‌دیده به کمک تغییر در نمادهای تخمینی آسیب‌دیده آنها می‌باشد. ابتدا یک بسته آسیب‌دیده انتخاب می‌گردد. هر بسته شامل s قسمت هم‌اندازه می‌باشد که صحت اطلاعات آن به کمک ICRC سنجیده می‌شود. به ازای هر قسمت آسیب‌دیده، مقادیر نمادهای کدشده‌ای که در ستون های ناپایدار وجود دارند تغییر داده می‌شود. پس از هر تغییر با محاسبه دوباره ICRC صحت اطلاعات بررسی می‌گردد. این فرآیند تکرار شده تا زمانی که صحت داده تایید شود. پس از انجام این عملیات بر روی تمامی قسمت‌های آسیب‌دیده یک بسته، به کمک OCRC صحت کل نمادهای کدشده بررسی می‌شود. در صورت تایید، بسته بازیابی شده و نسخه قبلی آن از بافر بسته‌های آسیب‌دیده حذف شده و سپس، نسخه بازیابی‌شده بسته به بافر بسته‌های سالم افزوده می‌شود. در صورتی که صحت نمادهای کدشده تایید نگردد، آن بسته دور ریخته خواهد شد.

زمانی که تعداد بسته‌های موجود در بافر بسته‌های سالم به عدد g رسید، فرآیند بازیابی متوقف می‌گردد. سپس، بسته های موجود در بافر بسته‌های سالم برای کدگشایی به کدگشا تحویل می‌شود.

۳.۲.۳ بازیابی در گره‌های میانی

قابلیت کدگذاری مجدد بسته‌های کدشده بدون نیاز به کدگشایی در گره‌های میانی یکی از مهمترین ویژگی‌های کدگذاری شبکه است. در صورت کدگذاری مجدد یا ترکیب بسته‌های آسیب‌دیده با سایر بسته‌ها، نتیجه قطعا یک بسته‌ی آسیب‌دیده خواهد بود. برای چنین بسته‌ای، CRC ها را به دلیل عدم دسترسی به اطلاعات صحیح نمی‌توان محاسبه نمود. درنتیجه، بسته‌های آسیب‌دیده نمی‌توانند در فرآیندکدگذاری مجدد شرکت کنند. شکل ۱.۳ نشان دهنده‌ی درصد بسته‌های آسیب‌دیده



شکل ۱.۳: درصد بسته‌های آسیب‌دیده هنگام عبور از تعداد متفاوتی از گره‌های میانی، درصورتی که اندازه نمادهای کدشده هربسته ۱۰۰۰ بایت می‌باشد.

هنگام عبور از چند گره متوالی است. به عنوان مثال، بعد از عبور از ۲ گره میانی در صورتی که نرخ خطای بیت $۱۰^{-۴}$ باشد، تقریبا ۹۰ درصد بسته‌ها آسیب‌دیده خواهند بود. اغلب روش‌های PPR، اصلاح بسته‌های آسیب‌دیده را صرفا در گیرنده انجام می‌دهند. به همین دلیل، بسته‌های آسیب‌دیده در گره‌های میانی را به صورت آسیب‌دیده به گره‌های بالاتر جریان ارسال می‌کنند یا دور می‌ریزند. این روش‌ها از ظرفیت کدگذاری شبکه بهره‌مند نمی‌شوند.

در Fly-PRAC، گره‌های میانی در فرآیند بازیابی بسته‌های آسیب‌دیده شرکت می‌کنند. هنگام عبور بسته‌های یک گروه وابسته از یک گره میانی، بسته‌های سالم به بافر بسته‌های سالم و بسته‌های آسیب‌دیده به بافر بسته‌های ناسالم افزوده می‌شوند. پس از دریافت هر بسته متعلق به گروه وابسته، یک بسته با ترکیب تصادفی بسته‌های موجود کدگذاری مجدد شده و ارسال می‌گردد. همچنین، پس از دریافت R بسته (معادل دریافت تمامی بسته‌های یک گروه وابسته)، فرآیند تخمین و اصلاح بسته‌های آسیب‌دیده برای بسته‌های موجود در بافر بسته‌های ناسالم شروع می‌شود. برای بازیابی این بسته‌ها از Fly-PRAC استفاده می‌شود. پس از اصلاح بسته‌های آسیب‌دیده، به ازای هر بسته‌ی آسیب‌دیده دریافتی در آن گروه وابسته، یک بسته کدگذاری مجدد شده با ترکیب تصادفی بسته‌های سالم و بازیابی شده ایجاد و ارسال می‌گردد.

فصل ۴

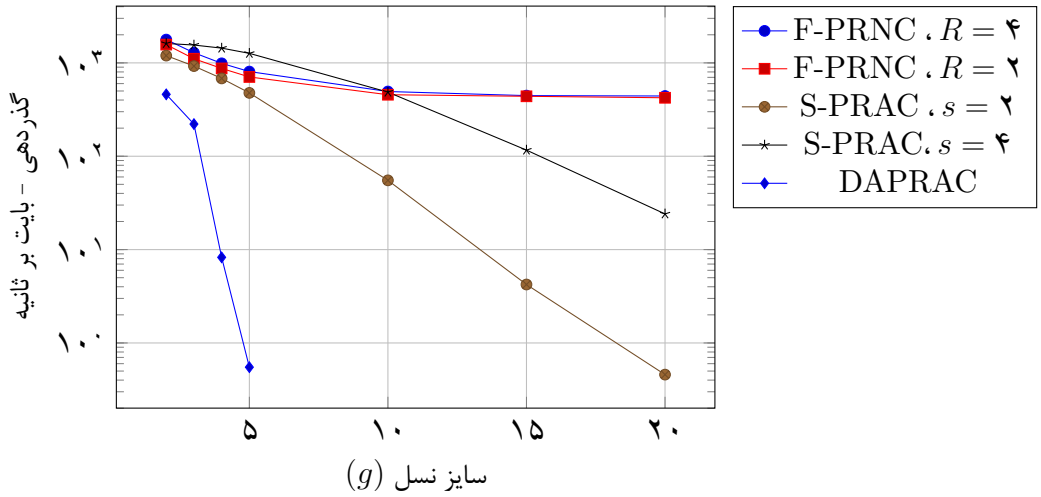
خلاصه آزمایش‌ها و نتایج شبیه‌سازی

در این فصل به کمک شبیه‌سازی، روش‌های پیشنهادی با آخرین PPR های معرفی‌شده برای کدگذاری شبکه مقایسه خواهند شد. در ابتدا روش F-PRNC را با S-PRAC و DAPRAC مقایسه می‌کنیم. در این آزمایش‌ها بسته‌ها، در مقابل نرخ خطای‌بایت یا به اختصار ByER های متفاوت قرار گرفته و از جهت میزان گذردهی با یکدیگر مقایسه می‌شوند. در ادامه روش Fly-PRAC با روش S-PRAC در نرخ خطای‌بیت یا به اختصار BER متفاوت از جهت گذرهی، سربار، تاخیر بازیابی و کدگشایی مقایسه می‌شوند. پیاده‌سازی روش‌ها در این دو بخش تفاوت اندکی دارد که در شرایط شبیه‌سازی به آن اشاره کرده‌ایم.

۱.۴ روش F-PRNC

۱.۱.۴ شرایط شبیه‌سازی

تمامی روش‌ها به کمک کتابخانه KODO [۱۸] پیاده‌سازی شده‌اند. همچنین تمامی عملیات بر پایه‌ی مجموعه متناهی $GF(۲^۸)$ صورت گرفته است. هر شبیه‌سازی ۵۰۰۰ بار تکرار شده و مقدار متوسط نتایج حاصله گزارش شده‌است. تمامی



شکل ۱.۴: مقایسه F-PRNC (با R' نماد افزونگی) با S-PRAC (با s قسمت) و DAPRAC، برای بسته‌ای به اندازه ۸ بایت و نرخ خطای‌بایت۰/۱.

روش‌ها از یک CRC-32 (با اندازه افزونگی ۳۲ بیت) استفاده می‌کنند. همچنین یک CRC-8 (با اندازه افزونگی ۸ بیت) به ازای هر قسمت S-PRAC در نظر گرفته شده‌است. در این بخش منظور از گذردهی، نرخ کدگشایی موفق در سمت گیرنده می‌باشد. به ازای هر درخواست ارسال دوباره در روش F-PRNC، ۳۶ میلی‌ثانیه تاخیر در ارسال در نظر گرفته‌ایم. همچنین، فرض کرده‌ایم که در صورت رخ دادن رویداد مثبت-کاذب در فرآیند بازیابی S-PRAC، فرستند باید بسته‌های آن نسل را از ابتدا ارسال کند. همچنین، در این بخش از اندازه بسته، منظور مجموع اندازه نمادهای کدشده یک بسته می‌باشد.

۲.۱.۴ مقایسه گذردهی F-PRNC با S-PRAC و DAPRAC

شکل ۱.۴ میزان گذردهی F-PRNC با S-PRAC و DAPRAC را در نرخ خطای‌بایت ۰/۱ و اندازه بسته ۸ بایت مقایسه می‌کند. این شکل نشان می‌دهد که افزایش اندازه نسل، گذردهی S-PRAC و DAPRAC را به طرز قابل توجه‌ای کاهش می‌دهد. علت این امر برای DAPRAC رخ دادن خطا در نقاط یکسان از بسته‌های متفاوت و برای S-PRAC افزایش رویدادهای مثبت کاذب است. البته با افزایش اندازه نسل سرعت گذردهی F-PRNC نیز به دلیل افزایش پیچیدگی کدگشایی کاهش پیدا می‌کند. این افزایش پیچیدگی در کاهش گذردهی سایر روش‌ها نیز تاثیرگذار است.

جدول ۱.۴: مقایسه F-PRNC (با R' نماد افزونگی) با S-PRAC (با s قسمت) و DAPRAC از نظر تاخیر ارسال(RD) و تاخیر کدگشایی (DD) به میلی‌ثانیه در کانالی با نرخ خطای بایت ۰/۱۴.

اندازه بسته (بایت)			۸		۱۶	
g			۴	۱۶	۴	۱۶
F-PRNC	$R'=۲$	RD	۶/۶	۷۲/۳۶	۳۴/۹	۵۱۱/۲
		DD	۷۱/۲	۴۸۶/۵	۲۵۶/۴	۴۲۸۶/۲
	$R'=۴$	RD	۰/۸	۸/۲۸	۱۱/۲۳	۹۵/۴
		DD	۶۰/۸	۴۹۰/۹	۴۰۶/۵	۱۱۶۱۸/۲
S-PRAC	$s=۲$	RD	۱۵۰۷/۳	$> ۱۰^۶$	۱۴۳۸۵۶	$> ۱۰^۶$
		DD	۱۴۳/۲	$> ۱۰^۶$	۱۳۳۴۹/۷	$> ۱۰^۶$
	$s=۴$	RD	۳۶۷/۸	$> ۱۰^۶$	۲۱۰۳۲/۵	$> ۱۰^۶$
		DD	۴۳/۶	۳۴۳۷۲/۶	۱۹۴۴/۴	$> ۱۰^۶$
DAPRAC		RD	۴۲۱۹/۶	$> ۱۰^۶$	$> ۱۰^۶$	$> ۱۰^۶$
		DD	۸۰۲۴۰/۳	$> ۱۰^۶$	$> ۱۰^۶$	$> ۱۰^۶$

جدول۱.۴ تاثیر استفاده از نمادهای افزونگی بر تاخیر کدگشایی و ارسال را برای کانالی با نرخ خطای‌بایت ۰/۱۴ نشان می‌دهد. این جدول نشان می‌دهد که روش F-PRNC نسبت به سایر روش‌ها از تاخیر کدگشایی و ارسال کمتری برخوردار است. به عنوان مثال، F-PRNC با ۴ نمادافزونگی، اندازه نسل ۱۶ و اندازه بسته‌ی ۸ بایت، دارای تاخیر ارسال ۸/۲۸ میلی‌ثانیه‌ای و تاخیر کدگشایی ۴۹۰/۹ میلی‌ثانیه‌ای می‌باشد. همین مقادیر برای S-PRAC (با ۲ قسمت) و DAPRAC، بالای ۱ ثانیه است.

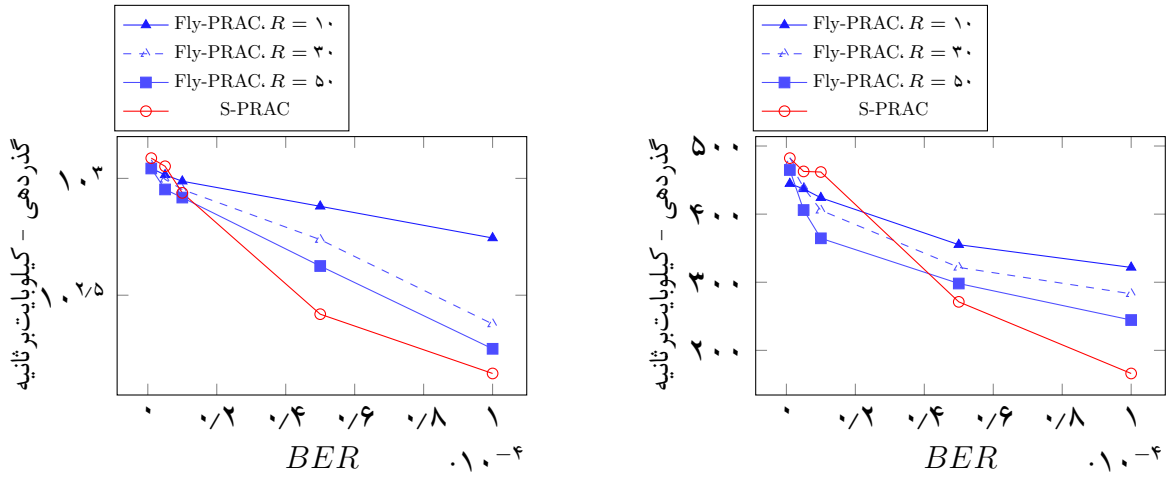
این جدول همچنین نشان می‌دهد که افزایش تعداد نمادهای افزونگی تاثیر به سزای بر روی تاخیرها می‌گذارد. به عنوان مثال، برای بسته‌ای با اندازه ۸ بایت و اندازه نسل ۴، افزایش تعداد نمادهای افزونگی از ۲ به ۴، تاخیر ارسال را از ۶/۶ میلی‌ثانیه به ۰/۸ میلی‌ثانیه کاهش می‌دهد. البته با توجه به آنکه اندازه تمامی بسته‌ها با افزایش نمادهای افزونگی افزایش می‌باد، می‌تواند تاخیر کدگشایی را تا حدودی افزایش دهد. به عنوان مثال، برای بسته‌ای با اندازه ۱۶ بایت و اندازه نسل ۱۶، افزایش تعداد نمادهای افزونگی از ۲ به ۴، تاخیر ارسال را از ۲۵۶/۴ میلی‌ثانیه به ۴۰۶/۵ میلی‌ثانیه افزایش می‌دهد.

۲.۴ روش Fly-PRAC

۱.۲.۴ شرایط شبیه‌سازی

تمامی روش‌ها به کمک کتابخانه KODO پیاده‌سازی شده‌اند. همچنین تمامی عملیات بر پایه‌ی مجموعه متناهی $GF(۲^۸)$ صورت گرفته است. هر شبیه‌سازی ۱۰۰۰۰ بار تکرار شده و مقدار متوسط آن گزارش شده‌است. همچنین در صورت رخ‌دادن رویداد مثبت-کاذب در روش S-PRAC، صرفا همان بسته دورریخته شده و گیرنده درخواست ارسال بسته‌ی اضافی می‌کند. S-PRAC و Fly-PRAC هر دو ، به ازای هر بسته از یک CRC-32 و به ازای هر قسمت از بسته‌ی کدشده یک CRC-8 استفاده می‌کنند. در این شبیه‌سازی فرض شده که تمامی CRCها در برابر خطا مقاوم هستند. همچنین، در این بخش از اندازه بسته، منظور مجموع اندازه نمادهای کدشده یک بسته می‌باشد.

۲.۲.۴ تاثیر اندازه نسل و بسته بر بازیابی



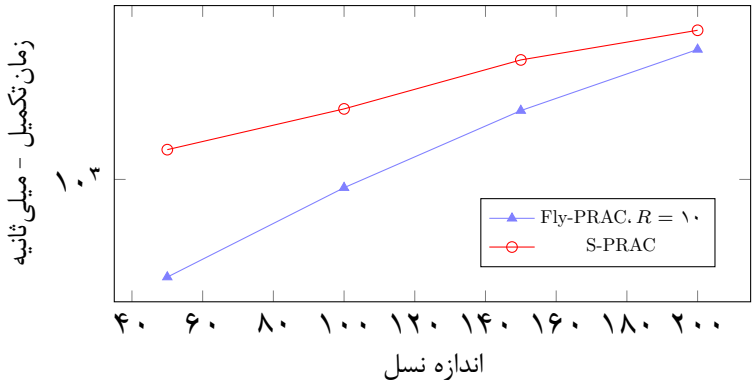
الف- اندازه نمادها ۳۰۰ بایت

ب- اندازه نمادها ۹۰۰ بایت

شکل ۲.۴: تاثیر مقدار نرخ خطای‌بیت و اندازه نمادهای یک بسته بر گذردهی شبکه برای یک نسل به اندازه ۱۰۰.

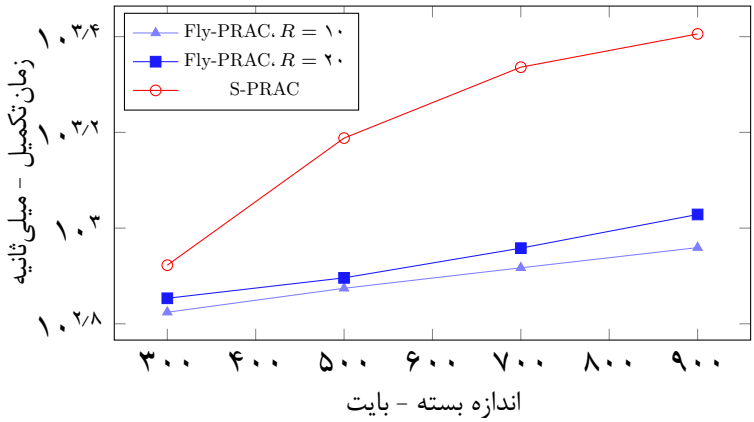
در این بخش Fly-PRAC با S-PRAC از نظر گذردهی در نرخ خطای‌بیت متفاوت مقایسه شده‌اند. همچنین تاثیر اندازه نسل و بسته بر زمان تکمیل کدگشایی و بازیابی بررسی شده‌است.

شکل ۲.۴ نشان می‌دهد که در شرایط پرخطا روش پیشنهادی مقدار گذردهی بیشتری را دارد. به عنوان مثال، برای بسته‌ای با اندازه ۳۰۰ بایت، گذردهی برای Fly-PRAC ($R = ۱۰$) و S-PRAC به ترتیب ۳۵۵/۱ و ۲۷۱ کیلوبایت بر ثانیه می‌باشد. همچنین گذردهی برای نرخ خطای‌بیت $۱۰^{-۵}$، در روش‌های Fly-PRAC ($R = ۱۰$) و S-PRAC به ترتیب ۹۷۲ و ۸۸۶ کیلوبایت بر ثانیه می‌باشد.



شکل ۳.۴: مقایسه زمان‌تکمیل بازیابی و کدگشایی برای Fly-PRAC و S-PRAC با ۴ قسمت و سایز بسته ۸۰۰ بایت در کانالی با نرخ خطای‌بیت ۵×۱۰^{-۵}.

شکل ۳.۴ نشان می‌دهد که برای اندازه نسل بزرگتر، مقدار گذردهی به دلیل افزایش پیچیدگی کدگشایی کاهش یافته‌است. در این شکل مشاهده می‌شود برای Fly-PRAC ($R = ۱۰$) زمان تکمیل کدگشایی و بازیابی در اندازه نسل ۵۰ و ۱۰۰ به ترتیب ۳۰۳ و ۹۰۳/۷ میلی‌ثانیه می‌باشد. به طور کلی با افزایش تعداد یا اندازه نمادها، زمان بازیابی افزایش می‌یابد. در شکل ۴.۴ مشاهده می‌شود این افزایش برای S-PRAC چشمگیرتر می‌باشد. دلیل این امر، افزایش بیشتر تعداد ستون‌های ناپایدار در S-PRAC است. به عنوان مثال، با افزایش اندازه بسته از ۳۰۰ به ۹۰۰ بایت، زمان کدگشایی و بازیابی برای S-PRAC از ۸۳۶/۵ به ۲۵۴۴/۸ میلی‌ثانیه افزایش پیدا می‌کند. همین میزان افزایش اندازه بسته برای روش Fly-PRAC ($R = ۱۰$) این زمان را از ۶۶۷/۱ به ۹۱۰/۳میلی‌ثانیه افزایش می‌دهد.



شکل ۴.۴: مقایسه زمان‌تکمیل بازیابی و کدگشایی برای Fly-PRAC و S-PRAC با ۴ قسمت و سایز نسل ۱۰۰ در کانالی با نرخ خطای‌بیت ۵×۱۰^{-۵}.

جدول ۲.۴: مقایسه تعداد کل ارسال‌ها و زمان‌تکمیل بازیابی و کدگشایی به میلی‌ثانیه (CT) برای Fly-PRAC و S-PRAC با بسته‌هایی با ۴ قسمت در شرایطی که نرخ خطای‌بیت برابر ϵ و یک گره‌میانی بین گیرنده و فرستنده قرار دارد.

نام روش	R	$\epsilon = ۵ \times ۱۰^{-۵}$		$\epsilon = ۱۰^{-۴}$	
		تعداد ارسال	CT	تعداد ارسال	CT
Fly-PRAC	۵	۱۴۱/۳	۱۱۲۶/۲	۱۸۱/۹	۱۵۰۲/۵
	۱۰	۱۳۶/۶	۱۲۲۱/۹	۲۰۲	۲۳۴۴
S-PRAC	-	۱۴۶/۱	۱۴۷۶/۴۵	۲۱۲/۳	۳۰۶۹/۹۷

۳.۲.۴ تاثیر بازیابی در گره‌های میانی

در این قسمت، Fly-PRAC را با S-PRAC در شبکه‌ای که از طریق یک گره‌میانی متصل شده‌اند، مقایسه کرده‌ایم.کانال‌های ارتباطی فرستنده-گره‌میانی و گره‌میانی-گیرنده دارای نرخ خطای‌بیت برابر می‌باشند. گره‌میانی در Fly-PRAC قادر به بازیابی بسته‌های آسیب‌دیده می‌باشد. اما در روش S-PRAC گره‌میانی صرفا تمامی بسته‌های دریافتی را به همان حالت به فرستنده ارسال می‌کند. جدول ۲.۴ نشان می‌دهد روش پیشنهادی می‌تواند تعداد ارسال‌ها و زمان تکمیل کدگشایی و

جدول ۳.۴: مقایسه تعداد کل ارسال‌ها و قابلیت بازیابی برای Fly-PRAC و S-PRAC با بسته‌هایی با ۴ قسمت و نرخ خطای‌بیت ۵×۱۰^{-۵}.

نام روش	سایز نسل	R	سایز بسته (بایت)			
			۳۰۰		۷۰۰	
			تعداد کل ارسال	تعداد بازیابی شده	تعداد کل ارسال	تعداد بازیابی شده
Fly-PRAC	۱۰۰	۵	۱۲۴/۳	۵/۵	۱۲۵	۲۰/۴
		۱۰	۱۱۱/۲	۸/۵	۱۱۲/۸	۲۲/۹
		۱۵	۱۰۵/۹	۱۰	۱۱۱/۶	۱۹/۸
	۲۰۰	۵	۲۴۹/۴	۱۰/۸	۲۵۰/۵	۴۱
		۱۰	۲۲۲/۴	۱۶/۵	۲۲۵/۵	۴۶/۳
		۱۵	۲۱۱/۷	۱۹/۷	۲۲۳	۴۰
S-PRAC	۱۰۰	-	۱۱۲/۳	۰/۹	۱۳۱/۷	۱/۶
	۲۰۰	-	۲۲۴/۴	۱/۳	۲۶۴	۱/۵

بازیابی را بهبود دهد. به عنوان مثال، در نرخ خطای‌بیت ۵×۱۰^{-۵}، تعداد کل ارسال‌ها در Fly-PRAC ($R = ۱۰$) و S-PRAC به ترتیب ۱۳۶/۶ و ۱۴۶/۱ می‌باشد. همچنین برای نرخ خطای‌بیت $۱۰^{-۴}$، زمان تکمیل کدگشایی و بازیابی در Fly-PRAC ($R = ۵$) و S-PRAC به ترتیب ۱۵۰۲/۵ و ۳۰۶۹ میلی‌ثانیه می‌باشد.

۴.۲.۴ مقایسه قابلیت بازیابی و سربار

جدول ۳.۴ تعداد کل ارسال‌ها تا کدگشایی موفق یک نسل و تعداد بسته‌های بازیابی شده را در Fly-PRAC و S-PRAC مقایسه می‌کند. به طور کلی این جدول نشان می‌دهد که روش Fly-PRAC قادر است تا تعداد بسته‌های بیشتری را بازیابی کرده و همچنین کدگشایی را با تعداد ارسال‌های کمتری انجام دهد. به عنوان مثال، برای اندازه نسل ۱۰۰ و اندازه بسته ۳۰۰ بایت، تعداد بسته‌های بازیابی شده در Fly-PRAC ($R = ۱۰$) و S-PRAC به ترتیب ۱۰۵/۹ و ۱۱۲/۳ می‌باشد. این امر نشان می‌دهد ارسال بسته‌های وابسته در زمان مناسب می‌تواند به بازیابی کمک کرده و حتی سربار شبکه را کاهش دهد. این جدول نشان می‌دهد، تعداد کل ارسال‌ها برای روش پیشنهادی در بسته‌های با اندازه ۳۰۰ و ۹۰۰ بایت، به

ترتیب ۱۲/۰۹ و ۴۰/۹۸ ارسال نسبت به S-PRAC کمتر می‌باشد. در واقع با افزایش اندازه‌ی بسته فاصله عملکرد بین S-PRAC و Fly-PRAC افزایش می‌یابد. هرچند ارسال تعداد زیادی بسته وابسته لزوما شرایط ارتباط را بهبود نمی‌دهد. در روش پیشنهادی، تعداد کل ارسال‌ها برای بسته‌ای با اندازه ۳۰۰ بایت، برای گروه‌های وابسته با $R = ۵$ و $R = ۱۰$ به ترتیب ۱۲۴/۳ و ۱۰۵/۹ می‌باشد. این امر نشان می‌دهد که اندازه گروه‌های وابسته باید مطابق با شرایط کانال انتخاب شود.

فصل ۵

نتیجه‌گیری و کارهای آتی

با توجه به طبیعت ارتباطات بی‌سیم، آسیب‌دیدن بسته‌ها هنگام عبور از یک کانال ارتباطی خطادار اجتناب‌ناپذیر است. به ویژه در کانال‌های ارتباطی پرخطا، تعداد قابل توجه‌ای از بسته‌های دریافتی آسیب‌دیده هستند. به طورکلی در شبکه‌های کدگذاری شده بسته‌های آسیب‌دیده دورریخته می‌شوند. با توجه به اینکه اغلب قسمت‌های یک بسته آسیب‌دیده دارای اطلاعات بدون خطا و کاربردی است، دور ریختن آنها استراتژی بهینه‌ای نیست.

در این مطالعه، دو روش برای بازیابی بسته‌های آسیب‌دیده در شبکه‌های کدگذاری شده معرفی شد. ابتدا روشی به نام F-PRNC را معرفی کردیم. در این روش به بسته‌های کدشده چندین نماد افزونگی اضافه می‌شود. سپس در سمت گیرنده فرآیند بازیابی بسته‌های آسیب‌دیده در دو گام پیگیری می‌شود. گام اول پس از دریافت اولین بسته آسیب‌دیده شروع شده و سعی می‌کند به کمک نمادهای افزونگی، بازیابی بسته را انجام دهد. گام دوم پس از دریافت $g+1$ بسته شروع می‌شود. در این گام بازیابی، ابتدا ACR تخمینی از نمادهای آسیب‌دیده تهیه کرده و سپس عملیات بازیابی برای بسته‌های آسیب‌دیده ادامه پیدا می‌کند. این روش به کمک شبیه‌سازی با دو روش DAPRAC و S-PRAC مقایسه شد. به کمک نتایج شبیه‌سازی نشان دادیم که F-PRNC می‌تواند در شرایط ارتباطی پرخطا مقدار گذردهی را افزایش، تاخیر ارسال و کدگشایی را کاهش

دهد. در ادامه، روش Fly-PRAC را معرفی کردیم. در این روش، فرستنده پس از ارسال چند بسته کدشده، یک بسته وابسته خطی را عامدانه ارسال می‌کند. سپس به کمک تکنیک معرفی شده برای تخمین نقاط آسیب‌دیده، عملیات بازیابی را پیش از دریافت $g+1$ بسته آغاز می‌کند. Fly-PRAC اولین روشی است که امکان بازیابی بسته را در گره‌های میانی دارد و می‌تواند از ظرفیت کدگذاری شبکه بهره بیشتری ببرد. به کمک شبیه‌سازی روش پیشنهادی را با S-PRAC مقایسه کرده و نشان دادیم که این روش در شرایط پرخطا مقدار گذردهی را افزایش و زمان تکمیل کدگشایی و بازیابی را کاهش می‌دهد. این روش همچنین با کاهش میزان رویدادهای مثبت-کاذب در بازیابی قادر است تا بسته‌های بیشتری بازیابی کرده و با کاهش تعداد کل ارسال‌ها سربار شبکه را کاهش دهد.

روش های PPR موجود را می‌توان از چند جنبه بهبود داد. تمامی PPRهای معرفی شده برای کدگذاری شبکه، فرض می‌کنند که وکتور ضرایب و CRCها در برابر خطا مقاوم هستند. در شرایط واقعی، هیچ یک از بخش‌های بسته مصون از خطا نیست. یک روش PPR بهبودیافته باید قادر باشد تا تمامی نقاط آسیب‌دیده در یک بسته را بازیابی کند. همچنین تمامی روش های PPR ارائه شده بر پایه ACR، صرفا شبکه‌های نقطه به نقطه که در آن صرفا یک گیرنده می‌باشد را درنظر گرفته‌اند. در نظر گرفتن شبکه‌های چند-پخشی[1] یا همه-پخشی[2] در مطالعات آتی و مقایسه این روش‌ها در این نوع شبکه‌ها می‌تواند ارزیابی بهتری از عملکرد آنها بدهد.

[1] Multicast

[2] Broadcast

کتاب‌نامه

[1] N. Todtenberg and R. Kraemer, "A survey on bluetooth multi-hop networks," *Ad Hoc Networks*, vol.93, p.101922, 2019.

[2] D. Gislason. *Zigbee wireless networking*. Newnes, 2008.

[3] K. Pahlavan and P. Krishnamurthy, "Evolution and impact of wi-fi technology and applications: a historical perspective," *International Journal of Wireless Information Networks*, vol.28, no.1, pp.3–19, 2021.

[4] F. Muteba, K. Djouani, and T. Olwal, "A comparative survey study on lpwa iot technologies: Design, considerations, challenges and solutions," *Procedia Computer Science*, vol.155, pp.636–641, 2019. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019).

[5] A. F. Behrouz and C. Sophia, "Data communications and networking," *Forouzan with Sophia Chung Fegan*, 2007.

[6] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," *SIGCOMM Comput. Commun. Rev.*, vol.26, p.243–254, Aug. 1996.

[7] S. L. Howard, C. Schlegel, K. Iniewski, and K. Iniewski, "Error control coding in low-power wireless sensor networks: When is ecc energy-efficient?," *EURASIP Journal on Wireless Communications and Networking*, vol.2006, pp.1–14, 2006.

[8] Z. H. Kashani, "Power optimised channel coding in wireless sensor networks using low-density parity-check codes," *IET Communications*, vol.1, pp.1256–1262(6), December 2007.

[9] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE Int. Symp. Inf. Theory, 2003. Proceedings.*, p.442, June 2003.

[10] J. A. Cabrera, R. Steppert, F. Gabriel, and F. H. P. Fitzek, "Do not waste the waste: Packetized rateless algebraic consistency for ieee 802.11 networks," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp.1–6, 2021.

[11] J. Xie, W. Hu, and Z. Zhang, "Revisiting partial packet recovery in 802.11 wireless lans," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, (New York, NY, USA), p.281–292, Association for Computing Machinery, 2011.

[12] K. D. Irianto, J. A. Cabrera, G. T. Nguyen, H. Salah, and F. H. Fitzek, "S-PRAC: Fast partial packet recovery with network coding in very noisy wireless channels," in *2019 Wireless Days (WD)*, (Manchester, United Kingdom), pp.1–7, Apr. 2019.

[13] G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi, "Beyond the bits: Cooperative packet recovery using physical layer information," in *Proc. 13th Ann. ACM Int. Conf. Mobile Comput. and Netw.*, (Montréal, Québec, Canada), pp.147–158, Sept. 2007.

[14] K. D. Irianto, G. T. Nguyen, H. Salah, and F. H. Fitzek, "Partial packet in wireless networks: a review of error recovery approaches," *IET Communications*, vol.14, no.2, pp.186–192, 2020.

[15] G. Angelopoulos, M. Médard, and A. P. Chandrakasan, "Harnessing partial packets in wireless networks: Throughput and energy benefits," *IEEE Trans. Wireless Commun.*, vol.16, pp.694–704, Nov. 2017.

[16] J. Cabrera, G. Nguyen, D. E. Lucani, M. Pedersen, and F. H. P. Fitzek, "Taking the trash back in: Practical joint channel and network coding for improving IEEE 802.11 networks," in *Eur. Wireless 2017; 23th Eur. Wireless Conf.*, pp.1–5, May 2017.

[17] H. K. Nazari, K. Ghassabi, P. Pahlevani, and D. E. Lucani, "Improving the decoding speed of packet recovery in network coding," *IEEE Commun. Lett.*, vol.25, pp.351–355, Feb. 2021.

[18] M. V. Pedersen, J. Heide, and F. H. P. Fitzek, "Kodo: An open and research oriented network coding library," in *Networking 2011 Workshops* (V. Casares-Giner, P. Manzoni, and A. Pont, eds.), (Berlin, Heidelberg), pp.145–152, 2011.