Al sham University
Faculty of Informatic

جامعة الشام الخاصة
كلية الهندسة المعلوماتية

التحكم ضمن طبقة وصلة المعطيات

Data Link Control

تهتم طبقة وصلة المعطيات بالنقل من عقدة إلى عقدة .لتحقيق ذلك تقوم بمجموعة من الوظائف مثل التأطير Framingوالتحكم بالأخطاء والتدفق وتحقيق البروتوكولات التي تؤمن نقل الأطر بين العقد المتجاورة نقلا موثوقًا

التأطير Framing

- فالتأطير على مستوى طبقة وصلة المعطيات يسمح بتمييز رسالة من مصدر محدد إلى وجهة محددة أو بالتمييز بين الرسالة وبين رسائل أخرى موجهة إلى وجهات مختلفة، وذلك عن طريق إضافة عنواني المصدر والوجهة.
- يفيد عنوان الوجهة بمعرفة إلى أين ستذهب ا لرسالة بينما تستفيد الوجهة من عنوان المصدر في حال أرادت إقرار الاستلام.
- مع أنه من الممكن إرسال كامل الرسالة ضمن إطار واحد إلا أن ذلك غير مقبول عمليًا،فتحريف أي بت من الرسالة، الذي يزداد احتمال وقوعه مع ازدياد طول الرسالة، يقتضي إعادة إرسال كل الرسالة من جديد، أضف إلى ذلك عدم فعالية التحكم بالتدفق وبالأخطاء.

البروتوكولات محرفية التوجه
Character-oriented protocols

- تكون المعطيات، وفق هذا النوع من البروتوكولات، عبارة عن محارف مؤلفة من ثمانية بتات تستخدم أحد أنظمة الترميز المعروفة مثل نظام ASCII،تكون الترويسة Header واللاحقة Trailerمن مضاعفات 8 bits، نضيف راية Flagمؤلفة من بايت واحد 8 bits للدلالة على بداية الإطار وراية أخرى للدلالة على نهايته؛ تنفع هذه الرايات في الفصل بين الأطر.

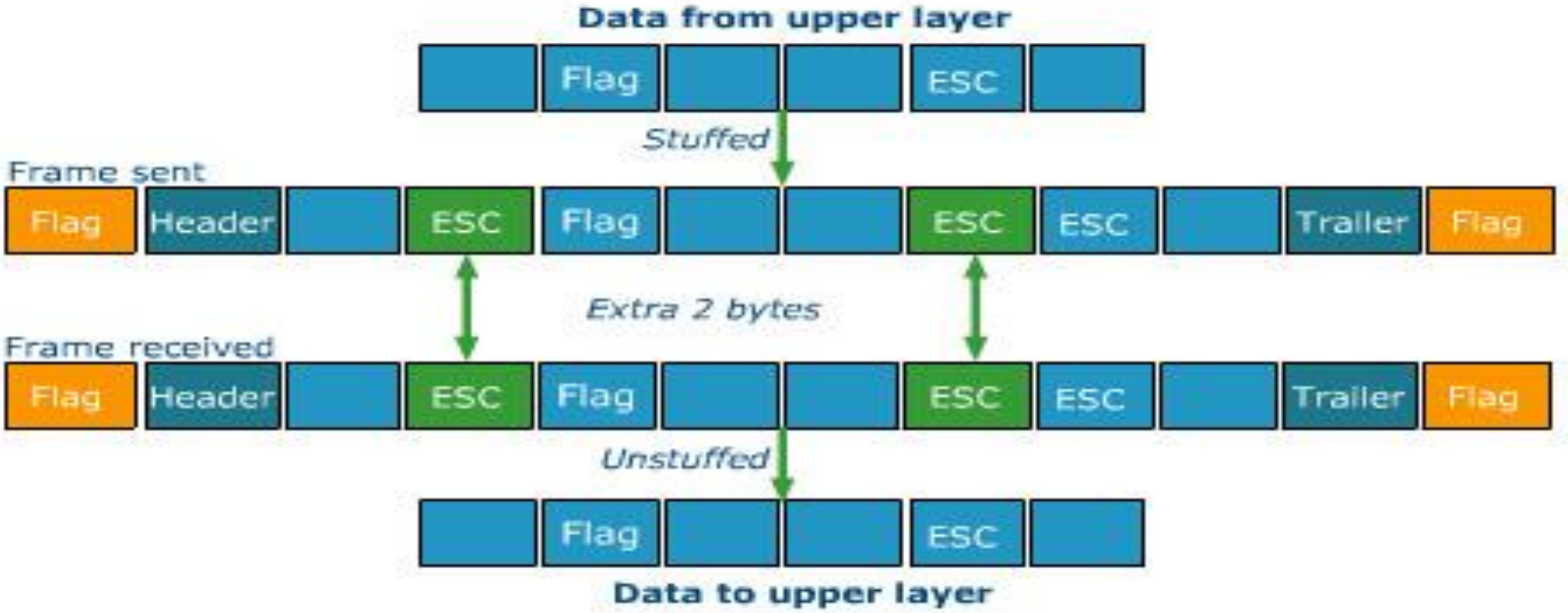
Flag	Traller	.....عدد صغير من المحارف	Header	flag	Flag

-
- يصلح هذا النوع من البروتوكولات عند نقل النصوص فقط لأنه في هذه الحالة يمكننا اختيار محرف غير مستخدم ضمن النص للدلالة على الراية . لكن مع تنوع طبيعة المعطيات المتناقلة حاليًا مثل الصور والفيديو والصوت، أصبح من غير الممكن ضمان عدم وجود الراية ضمن المعطيات .فإذا حدث ذلك فإن المستقبل عندما يقرأ حقل الراية في منتصف المعطيات فإنه يعتقد أنه وصل إلى نهاية الإطار .للتخلص من هذه المشكلة يتم استخدام طريقة حشو البايتات Byte stuffing

طريقة حشو البايتات Byte-stuffing

- يتم إضافة بايت خاص إلى جزء الإطار المخصص للمعطيات فقط عندما يوجد محرف الراية يدعى هذا المحرف عادةٍ بمحرف الإفلات ESC (Escape character)عندما يصادف المستقبل محرف الإفلات فإنه يحذفه ويعالج المحرف التالي على أنه محرف معطيات وليس محرف بداية أو نهاية الإطار.إضافة إلى ما سبق، فإنه يتم إضافة محرف الإفلات ESCعند وجود محرف ESCضمن المعطيات نفسها

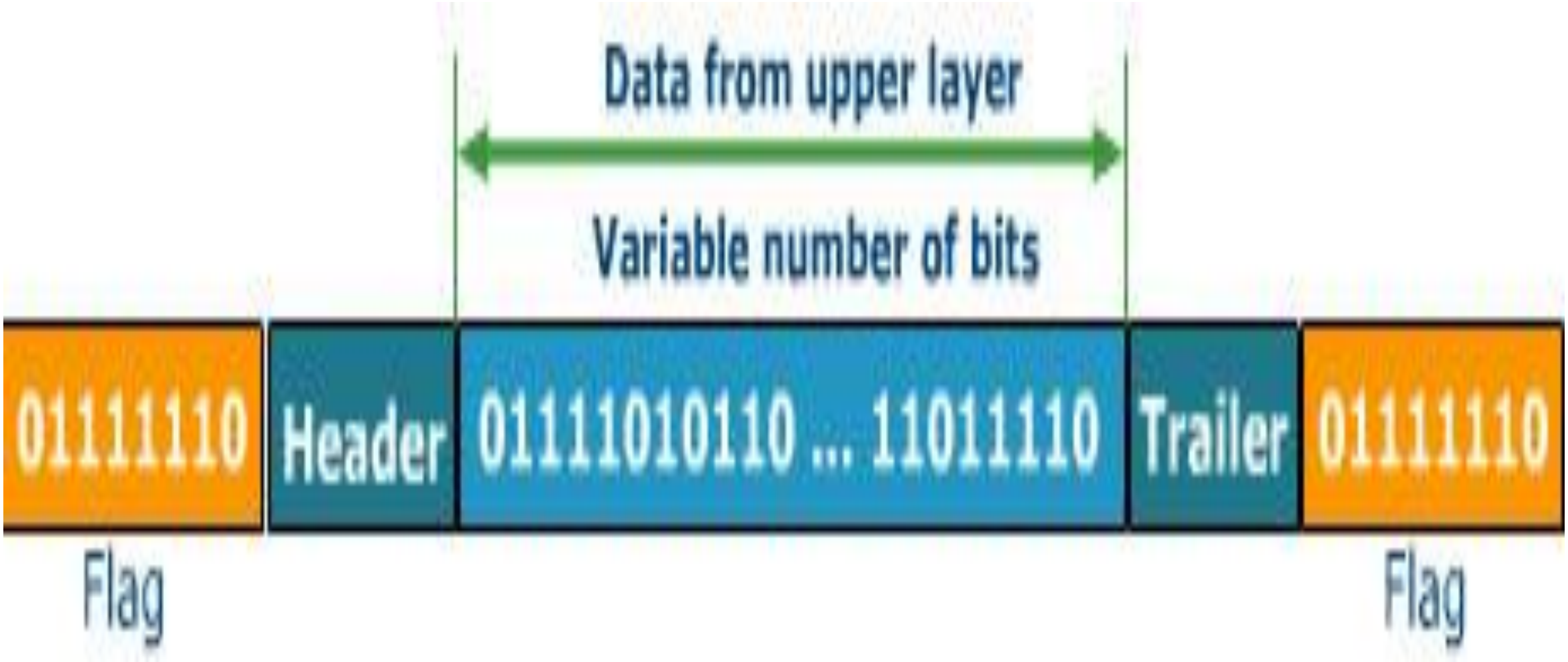
حشو البايتات وإزالة الحشو



البروتوكولات بتية التوجه
Bit-oriented Protocols

- يجري هنا ترميز جزء المعطيات من الإطار بسلسلة بتات التي يمكن أن تمثل الصوت والصورة والنصوص .تبقى الحاجة موجودة للفصل بين الأطر . تستخدم معظم البروتوكولات راية مكونة من ثمانية بتات **01111110** للدلالة على بداية أو نهاية كل إطار
- حتى نضمن عدم وجود راية البداية أو النهاية ضمن حقل المعطيات فإننا نقوم بتنفيذ طريقة الحشو البتيBit stuffing.

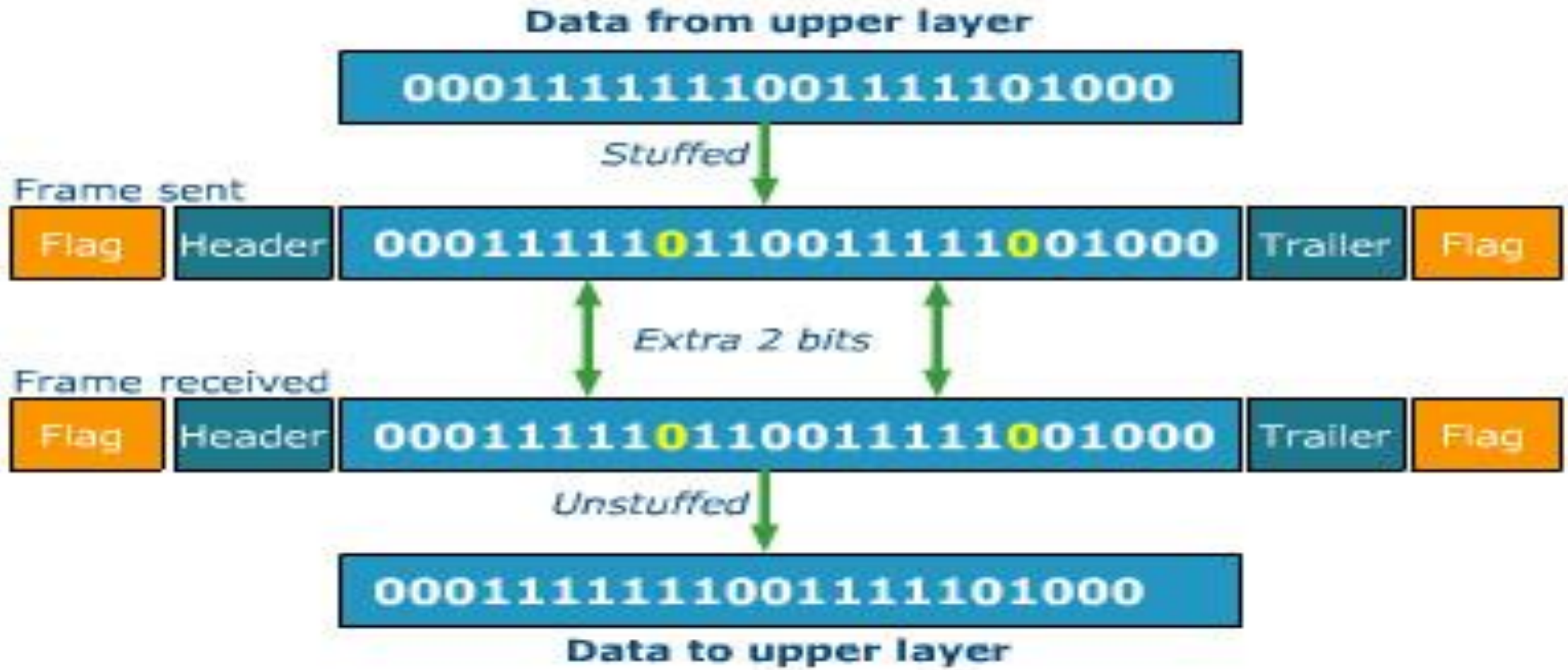
إطار وفق بروتوكول بتي التوجه



طريق الحشو البتي Bit-stuffing

- عندما يتم مصادفة صفر يليه خمس واحدات فإن المرسل يضيف صفر آخر بعدها .طبعًا يقوم المستقبل بحذف هذا البت الإضافي من المعطيات
- يبين الشكل التالي كيفية إضافة البت لدى المرسل وكيفية حذفه لدى المستقبل.

حشو البتات وإزالة الحشو



بروتوكولات التحكم بالتدفق وبالأخطاء

التحكم بالتدفق

- التحكم بالتدفق هو مجموعة الأجرائيات المستخدمة لتحديد كمية المعطيات التي يستطيع
- المرسل إرسالها قبل انتظار استلام إقرار من المستقبل
- يعتبر التحكم بالتدفق من أهم واجبات طبقة وصلة المعطيات حيث يسعى إلى تحديد كمية المعطيات التي يستطيع المرسل إرسالها قبل استلام إقرار من المستقبل . الهدف الأساسي للتحكم بالتدفق هو عدم إغراق المستقبل بمعطيات غير قادر على معالجتها بسبب سرعته المحدودة أو حجم الذاكرة المتوفرة له . لذلك يجب أن يستطيع المستقبل إعلام المرسل قبل الوصول إلى حدوده القصوى ليبطئ إرساله أو ليوقفه لفترة مؤقتة . بما أن سرعة المعالجة بشكل عام أقل من سرعة نقل المعطيات فيجب أن يتوفر لدى المستقبل ذاكرة دارئة Bufferيخزن فيها المعطيات قبيل معالجتها .فعندما تصبح الذاكرة شبه ممتلئة، يجب أن يكون باستطاعة المستقبل إعلام المرسل لتخفيف سرعة الإرسال

التحكم بالأخطاء

- يشمل التحكم بالأخطاء بشكل عام على آليات اكتشاف أخطاء النقل وتصحيحها .فهو يسمح للمستقبل بإعلام المرسل عن الأطر الضائعة أو المحرفة أثناء النقل وينسق آلية إعادة نقل هذه الأطر مع المرسل . أما على مستوى طبقة وصلة المعطيات فيتم التصحيح عن طريق إعادة النقل أو ما يعرف بطلب الإعادة الآليARQ (Automatic repeat request).

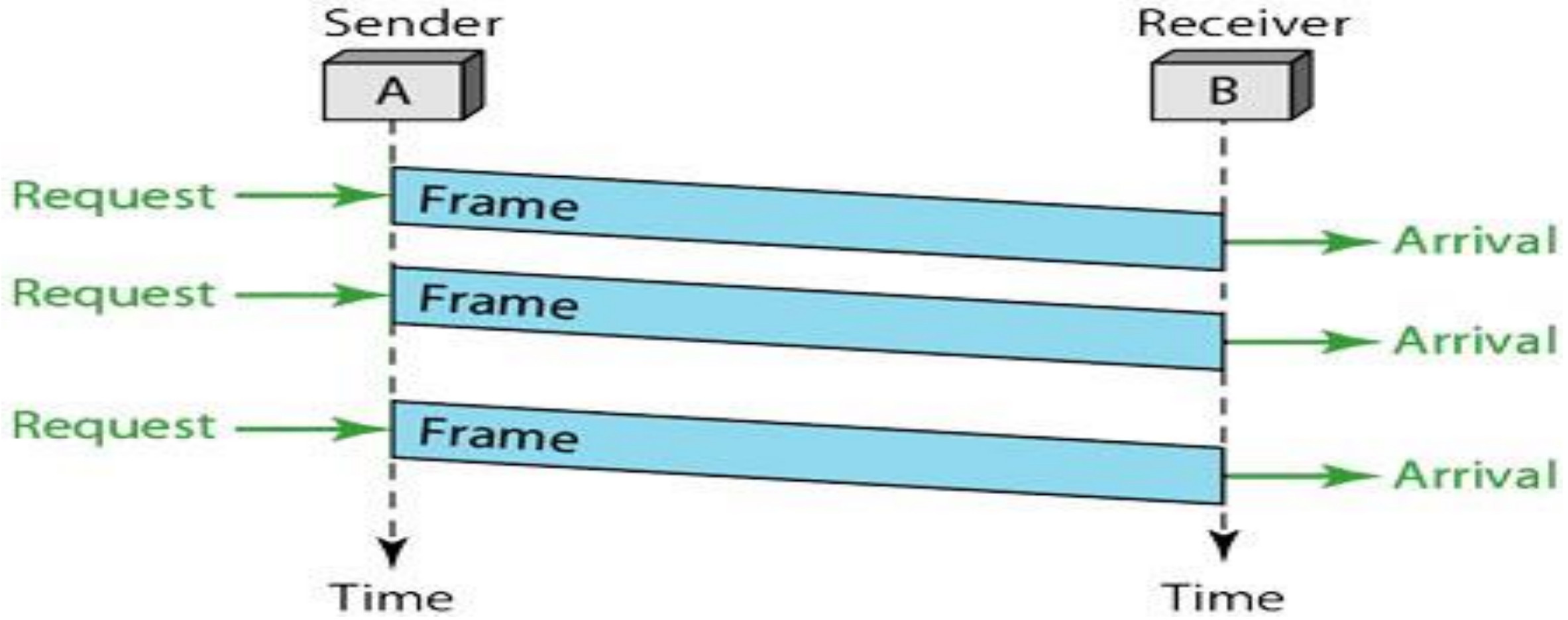
بروتوكولات القناة المثالية

- **١-البروتوكول المبسط**
- **٢-بروتوكول التوقف والانتظار Stop-and-wait**
- **٣-بروتوكولات القناة غير المثالية**
- **القناة المثالية :**هي القناة التي لا تضيع الأطر ولا تحرفها ولا تكررها.
- **تحريف الأطر** :تغيير بت أو أكثر في الإطار المرسل.
- **تكرار الأطر** :وصول نسختين من نفس الإطار

البروتوكول المبسط

- يتميز هذا البروتوكول بكونه لا يمتلك أية آلية للتعامل مع أخطاء النقل أو مع التدفق وبكونه
- وحيد الاتجاه .نفترض أيضًا أن المستقبل يستطيع معالجة أي إطار مستقبل بزمن معالجة مهمل.
- أي أن طبقة وصلة المعطيات عند المرسل تستلم المعطيات من طبقة الشبكة ومن ثم تبني الإطار
- وترسله بينما تقوم طبقة وصلة المعطيات عند المستقبل باستقبال الإطار القادم من الطبقة
- الفيزيائية وتستخلص منه المعطيات وتسلمها إلى طبقة الشبكة.
- لاحظ أن المرسل لا يستطيع الإرسال إذا لم يستلم طرد معطيات من طبقة الشبكة وكذلك الأمر
- بالنسبة للمستقبل الذي لا يستطيع تسليم طرد المعطيات إلى طبقة الشبكة إذا لم يستقبل إطار
- معطيات من الطبقة الفيزيائية . أو بشكل آخر، إذا قمنا بتحقيق البروتوكول عن طريق إجرائيات Procedures فإننا نحتاج لإدخال مفهوم الحدث Event داخل البروتوكول تنفذ الإجرائية لدى.
- المرسل بشكل مستمر وتكون بحالة انتظار دائم بحيث أنه لا يوجد أي فعل إلا عندما يصل طلب
- من طبقة الشبكة . كذلك الأمر بالنسبة للإجرائية لدى المستقبل التي تكون بحالة انتظار ورود
- إعلام Notification من الطبقة الفيزيائية بوصول إطار . أي أن الإجرائيتين تكونان في حالة
- عمل مستمر لأنهما لا يعرفان بشكل مسبق لحظة وقوع الحدث.

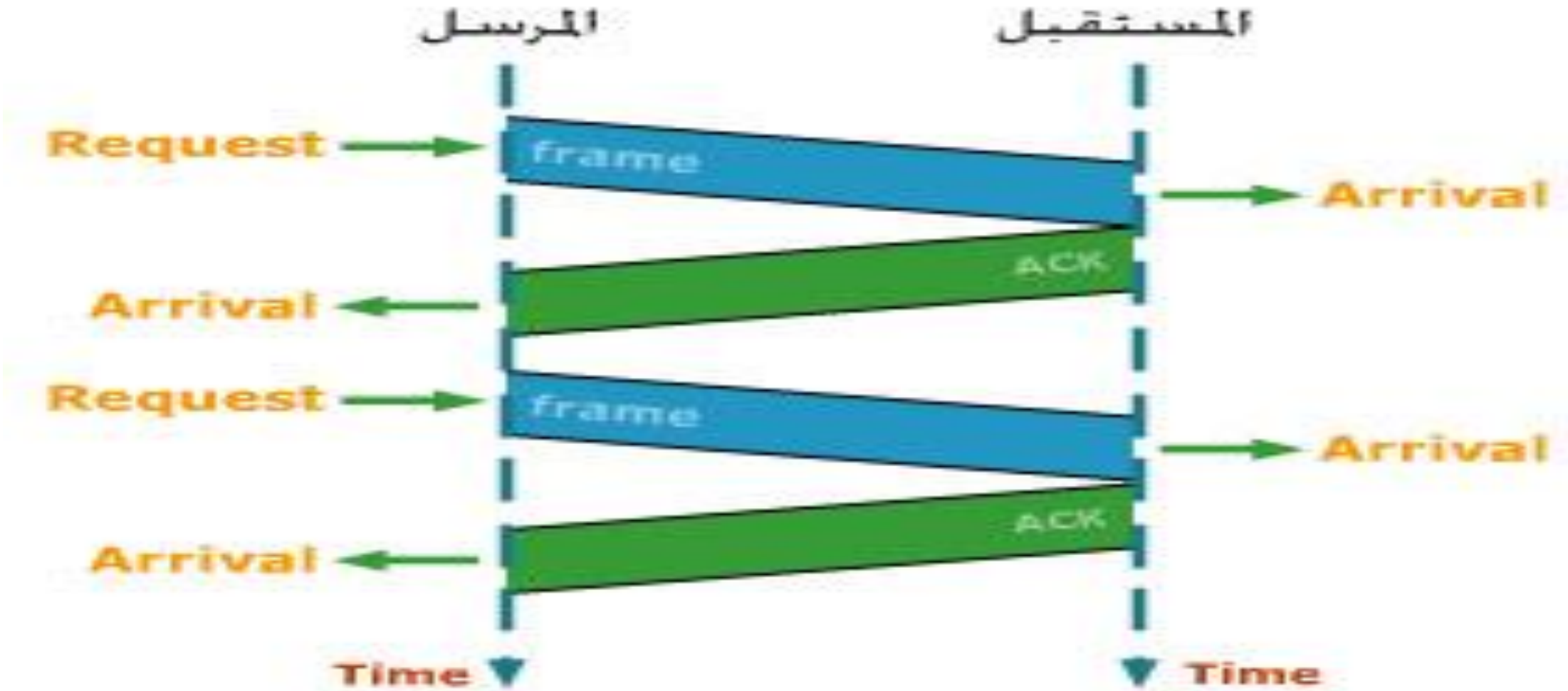
المخطط الزمني للبروتوكول المبسط Time Sequence Diagram



بروتوكول التوقف والانتظار Stop-and-wait

- يهدف هذا البروتوكول إلى التحكم بالتدفق بين المرسل والمستقبل حتى لا تمتلئ الذاكرة لدى المستقبل ويبدأ بإهمال المعطيات الجديدة.ولتحقيق ذلك يرسل المرسل إطار واحد ويتوقف حتى استلام تأكيد من المستقبل(جاهزية المستقبل لمتابعة الاستقبال)ومن ثم يرسل الإطار التالي وهكذا.
- لاحظ أن المعطيات تتدفق باتجاه واحد(من المرسل إلى المستقبل) لكننا نحتاج أيضًا إلى إطارات Ack frame. تحكم من المستقبل إلى المرسل

المخطط الزمني لبروتوكول الإرسال والتوقف



بروتوكولات القناة غير المثالية

- سنركز على ثلاثة أنواع من البروتوكولات:
- **١-التوقف والانتظار مع طلب الإعادة الآلي**

Stop-and-Wait Automatic Repeat Request

- **٢-طلب الإعادة الآلي بالعودة n- خطوة**

Go-Back-N ARQ

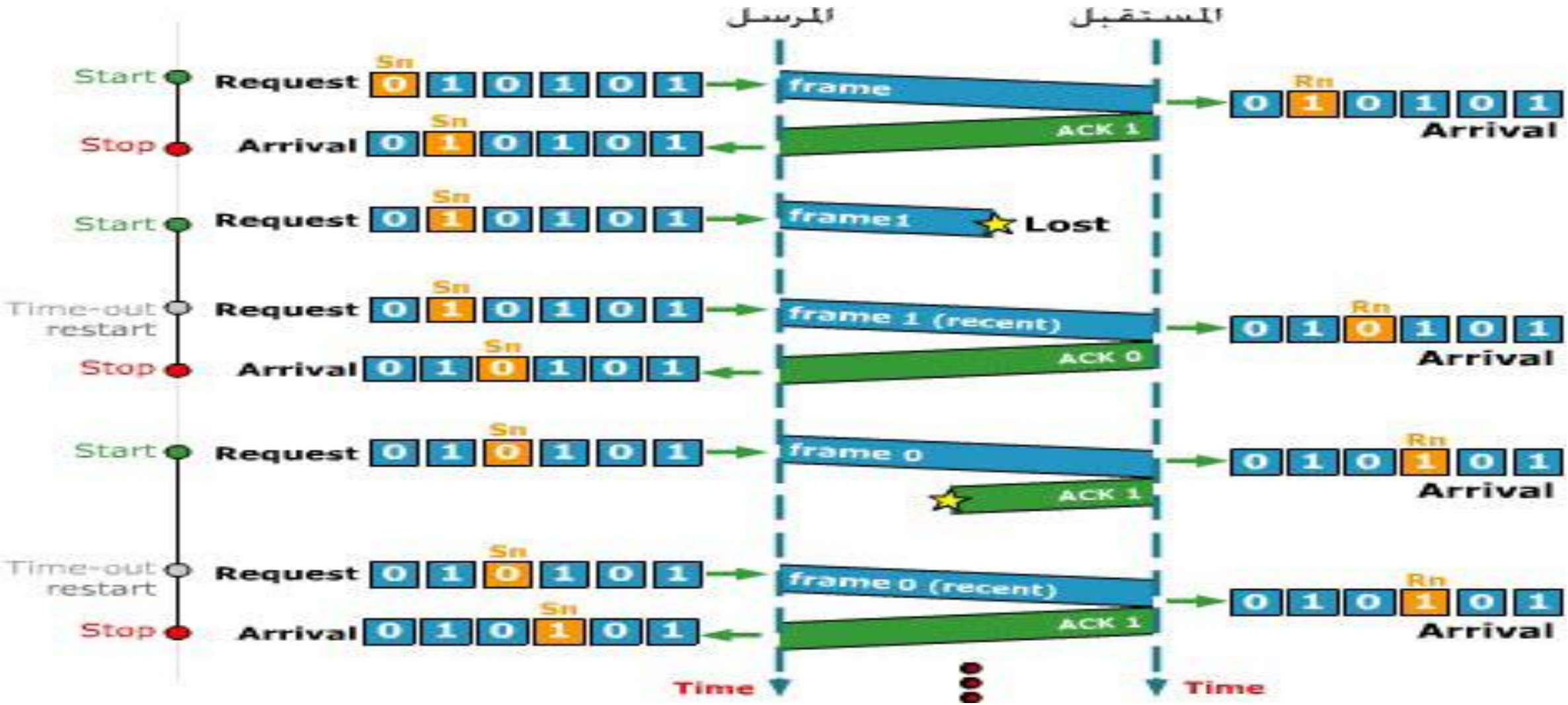
- **٣-بروتوكول تكرار الطلب الآلي مع تكرار انتقائي** **Selective Repeat Automatic Repeat Request**

التوقف والانتظار مع طلب الإعادة الآلي

Stop-and-Wait Automatic Repeat Request

- نفترض أولا أن المرسل يضيف بتات تكرارية تسمح للمستقبل باكتشاف الأخطاء .فعندما يستقبل
- المستقبل إطارًا ما فإنه يختبر كونه خالٍ من الأخطاء، فإذا كان يحوي خطأ ما فإنه يهمله.
- تبقى مشكلة الإطارات الضائعة أو المكررة أو المستلمة خارج الترتيب.
- عندما يتم إرسال عدة إطارات وفق ترتيب ما فيجب أن يتم استقبالها وفق نفس الترتيب وفي حال
- استلام إطار لا يحترم الترتيب فإننا نقول أن الإطار المستلم خارج الترتيبOut of order.
- إعادة إرسال الأطر الفاسدة أو الضائعة الأمر الذي يحتم على المرسل إبقاء نسخة من
- كل إطار مرسل وفي كل مرة يرسل فيها إطار ما فإنه يقلع مؤقت زمني Timerإذا انقضى.
- الوقت دون استلام إقرار تأكيد Ackمن المستقبل فإن المرسل يعيد إرسال النسخة المخزنة عنده
- ويعيد إقلاع المؤقت الزمني.بما أن إطار إقرار التأكيد Ack frameيمكن أن يضيع أيضًا أو يفسد أثناء الإرسال فيجب إضافة بتات تكرارية له ورقم تسلسلي . بنا ء على ما سبق، يقوم المرسل بإهمال إطار الإقرار إذا كان فاسدًا أو إذا كان خارج الترتيب.

بروتوكول التوقف والانتظار مع إعادة الطلب الآلي Stop-and-Wait ARQ



فعالية البروتوكول

- إن بروتوكول التوقف والانتظار مع إعادة الطلب الآلي غير فعال خاصة عندما تكون قناة الاتصال عريضة وطويلة . نقصد بقناة عريضة إمكانية إرسال عدد كبير من البتات في الثانية ونقصد بقناة طويلة كون تأخير الذهاب والإياب Round-trip delay طويل أي زمن تأخير البت لتحقيق رحلة ذهاب وإياب من وإلى المرسل.
- **تعريف :جداء عرض الحزمة Bandwidth-delay product**
- هو جداء عرض الحزمة في تأخير الذهاب والإياب
- Bandwidth (in bps) x Delay (second) =
- يعود سبب عدم الفعالية إلى أننا لا نستطيع استخدام عرض الحزمة المتوفر بشكل فعال حيث أن المرسل مضطر إلى انتظار الإقرار حتى يرسل إطار آخر

مثال

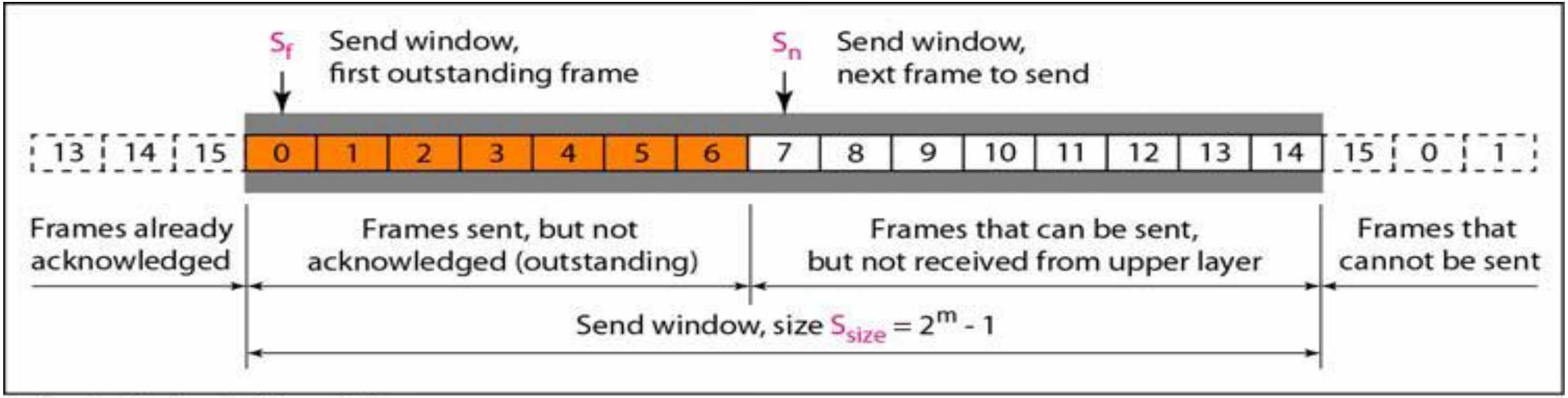
- افترض أن عرض الحزمة المتوفر هو 1 Mbpsوأن كل بت يستغرق 20 msلتحقيق رحلة ، ذهاب وإياب . ما هو جداء تأخير الحزمة ؟ وفي حال كان طول إطار المعطيات هو 1000 bits
- ما هي نسبة استخدام القناة؟
- **الحل:**
- جداء تأخير الحزمة هو:
- (1 x 106) x (20 x 10-3) = 20,000 bits
- هذا يعني أن النظام يستطيع إرسال 20,000 bitsخلال زمن إرسال المعطيات من قبل المرسل ووصولها إلى المستقبل وعودتها من جديد إلى المرسل .لكن نظامنا يرسل فقط1000 bitsمما يعني أن نسبة استخدام القناة هي:
- 1,000 / 20,000 = 5 %

بروتوكول طلب الإعادة الآلي بالعودة -nخطوة Go-Back-N ARQ

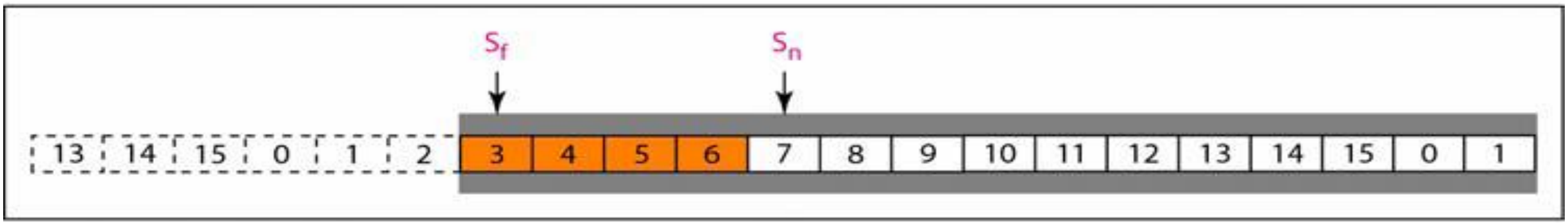
- نستطيع تحسين فعالية النقل إذا سمحنا للمرسل بإرسال عدة أطر .قبل انتظار الإقرار وهذا ما يفعله بروتوكول Go-Back-N ARQ
- يجري هنا ترقيم الأطر تسلسليًا إلى حد أعلى ومن ثم نعود إلى البداية .(0) فإذا افترضنا أن طول حقل الرقم التسلسلي هو m bitsفهذا يعني أننا نستطيع الترقيم من القيمة 0إلى القيمة 2 اس
- m-1. .أو بشكل آخر إن الأرقام التسلسلية هي مودولو 2اس m
- فإذا كان m=3فإن الأرقام التسلسلية ستكون:
- → 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, ... →

• النافذة المنزلقة Sliding Window

- تحدد النافذة المنزلقة للمرسل (أو للمستقبل) مجال الأرقام التسلسلية التي يستطيع أن يتعامل معها المرسل (أو المستقبل) في لحظة معينة . تحدد نافذة الإرسال الأرقام التسلسلية التي يمكن
- إرسالها .ضمن مكان ما من النافذة، تشير بعض الأرقام التسلسلية إلى الإطارات التي جرى إرسالها فعلا ويشير البعض الآخر إلى الأرقام التسلسلية التي يمكن إرسالها .سنعتبر أن قياس النافذة هو ثابت ويحدد الطول الأعظمي لها بالقيمة 2اس 1 -m.
- يبين الشكل التالي نافذة منزلقة ذات طول 15(أي أن 4 =m).



a. Send window before sliding



b. Send window after sliding

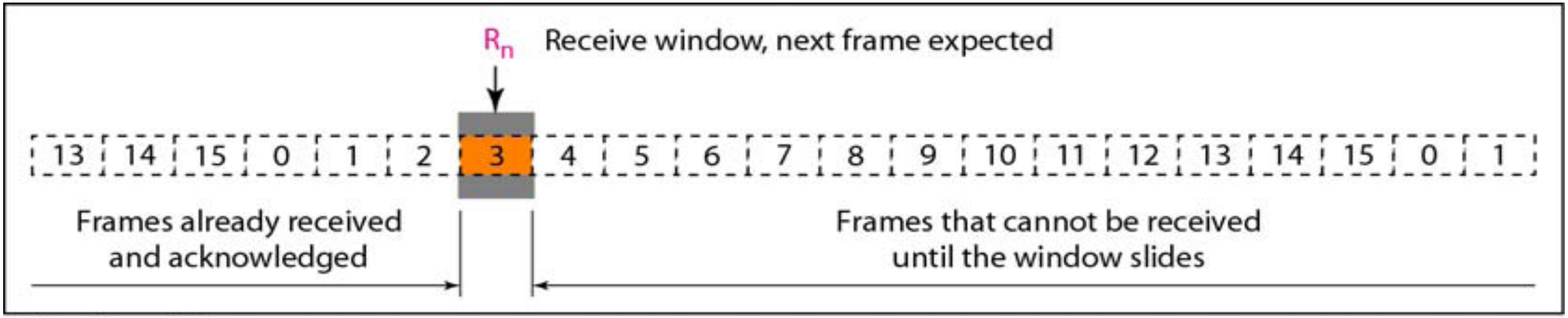
نافذة الإرسال لبروتوكول Go-Back-N ARQ

- تقسم النافذة، في لحظة معينة، الأرقام التسلسلية المحتملة إلى أربع مناطق . المنطقة الأولى،
- الممتدة من أقصى يسار الشكل وصولا إلى الجدار اليساري للنافذة، تحدد الأرقام التسلسلية للأطر
- التي جرى استلام إقرارات عنها .لا يعير المرسل أي اهتمام لهذه الأرقام ولا يحفظ أي نسخ عنها . المنطقة الثانية، الملونة باللون الأحمر من النافذة (تحمل الأرقام من 0إلى (6)، تحدد مجال
- الأرقام التسلسلية التابعة لأطر جرى إرسالها ولم يجري استلام أي إقرار عنها .يحتاج المرسل
- إلى الانتظار ليعرف فيما إذا كانت هذه الأطر أو بعضها وصلت بسلام أو ضاعت في الطريق.
- المنطقة الثالثة، البيضاء، تحدد مجال الأرقام التسلسلية للأطر التي يمكن إرسالها؛ لكن لم يتم
- استلامها بعد من الطبقات العليا .تحدد المنطقة الرابعة أرقام الأطر التي لا يمكن إرسالها قبل أن
- تنزلق النافذة.

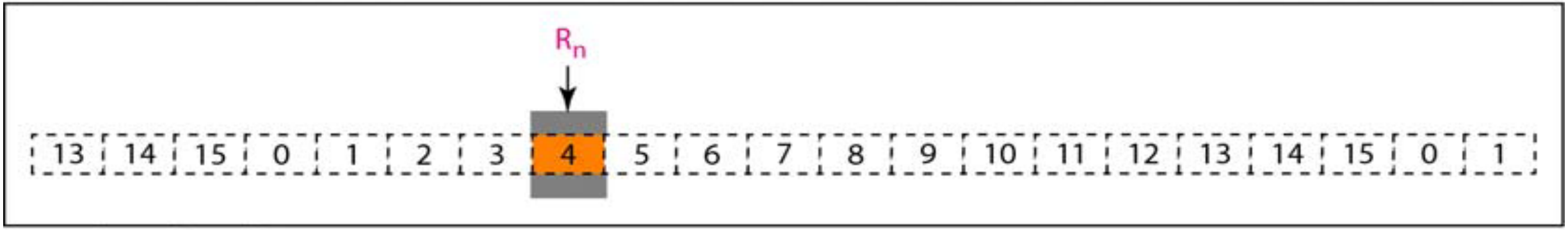
انزلاق النافذة

- يبين الشكل التالي كيف يمكن للنافذة أن تنزلق خانة واحدة أو أكثر إلى اليمين عندما يجري استلام إقرار من الطرف الأخر . لاحظ هنا أن الإقرارات تراكمية أي أننا نستطيع إقرار استلام أكثر من إطار واحد باستخدام إطار إقرار وحيد One ACK.بالنظر إلى الشكل جرى
- إقرار استلام الأطر المرقمة 0 , 1 and2مما أدى إلى انزلاق النافذة خانات لاحظ أن Sf =3. لأن الإطار رقم 3هو الإطار قيد الانتظار حاليًا.
- بالنسبة لنافذة المستقبل فهي تحاول التأكد من أن الأطر السليمة جرى استلامها وجرى إرسال الإقرارات المناسبة . يكون قياس نافذة المستقبل دومًا 1ويكون المستقبل دومًا في حالة انتظار إطار ما .عندما يجري استلام إطار خارج الترتيب فإنه يهمله ويجب إعادة إرساله .يبين الشكل التالي نافذة المستقبل.

شكل انزلاق النافذة



a. Receive window



b. Window after sliding

نـافذة الاستقبال لبروتوكول Go-Back-N ARQ

- لاحظ أننا نحتاج هنا إلى متحول واحد وهو الإطار التالي المنتظرReceive window, next)
- Rn (frame expected .تؤشر الأرقام التسلسلية الواقعة إلى يسار Rnعلى الأطر المستلمة
- والمرسل إقرار عنها بينما تؤشر الأطر الواقعة إلى يمين Rnعلى الأطر التي لا يمكن استلامها.
- لذلك فإن استلام أي إطار ذو رقم تسلسلي واقع ضمن هاتين المنطقتين يؤدي إلى إهماله ولا يتم
- قبول وإقرار استلام إلى الإطار الذي يحمل رقم تسلسلي مساوٍ للمتحولRn.
- تنزلق نافذة المستقبل أيضًا لكن خانة واحدة في كل مرة عندما يجري استلام الإطار المنتظر
- فقط.
- نستخدم في هذا البروتوكول مؤقت زمني وحيد وذلك لأن المؤقت المستخدم لأول إطار جرى
- إرساله تنقضي مدته أولا.

الإقرارات Acknowledgments

- يرسل المستقبل إقرارًا موجبًا Positive Ack إذا جرى استقبال إطار سليم وضمن الترتيب وإلا فإن المستقبل يهمل الإطار وجميع الأطر التالية ولا يفعل شيئًا) يصمت (حتى يستقبل الإطار المنتظر . إن صمت المستقبل يؤدي إلى انقضاء الفترة الزمنية المحددة لاستلام إقرار الإطار الأمر الذي يضطر المرسل إلى العودة Go-back وإعادة إرسال جميع الأطر من الإطار الذي انقضى مؤقته الزمني والأطر التالية.
- لا يحتاج المستقبل إلى إقرار استلام كل إطار سليم مستقبل وإنما يستطيع إرسال إقرار استلام تراكمي لمجموعة أطر معًا.

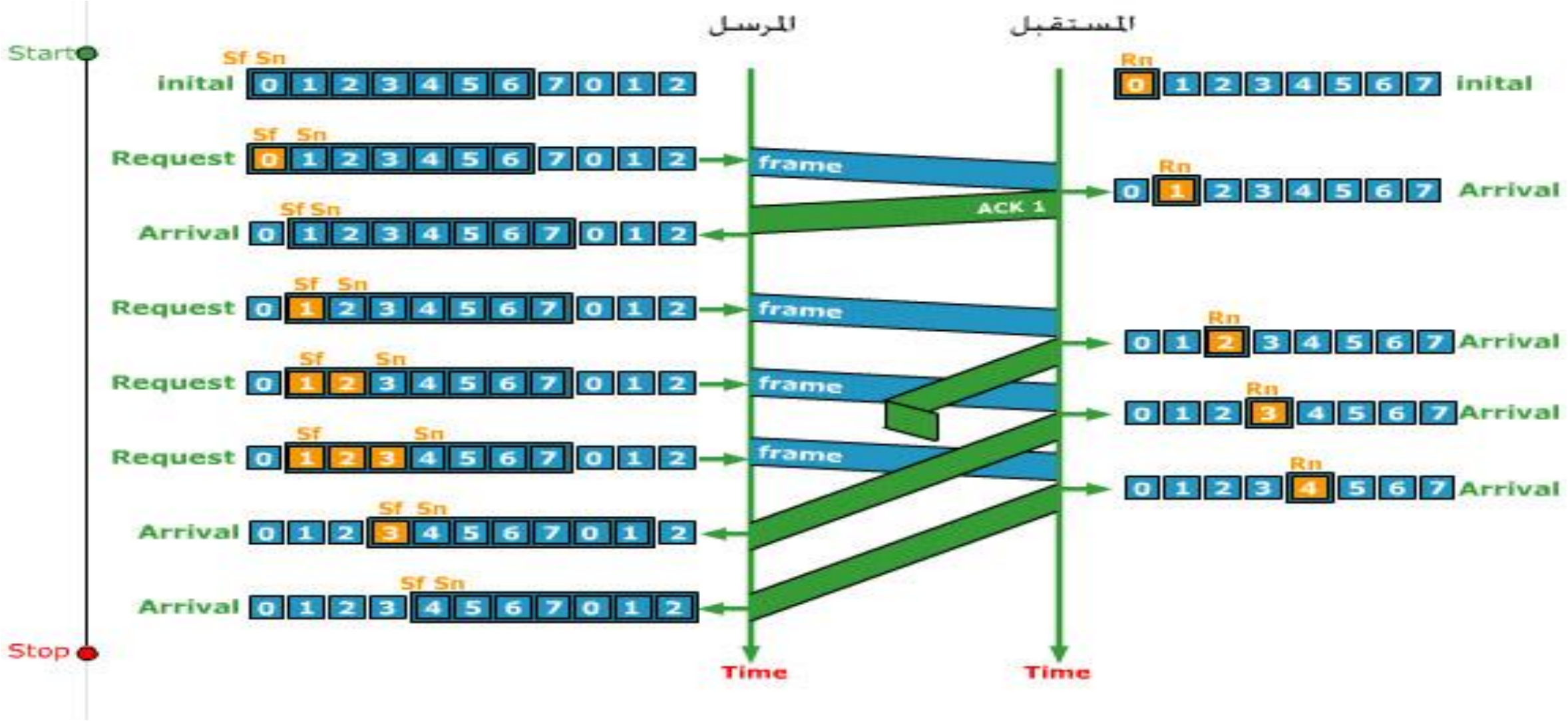
•إعادة إرسال إطار

- عندما ينقضي المؤقت الزمني فإن المرسل يعيد إرسال جميع الأطر قيد الانتظار . افترض أن المرسل أرسل الإطار رقم 6لكن المؤقت الزمني المتعلق بالإطار 3انقضى . الأمر الذي
- يعني أن الإطار رقم 3لم يجرِ إقرار استلامه؛ لذلك يعود المر سل ويرسل الأطر 4, 5, 6 3 من جديد لهذا السبب يدعى البروتوكول Go-back-N,.

مثال على بروتوكول GO-BACK-N

- يبين الشكل التالي مثالا على عمل بروتوكول Go-Back-N ARQ في حالة كون القناة من المرسل إلى المستقبل موثوقة بينما القناة العكسية غير موثوقة . أو، بشكل آخر، لا يجري إضاعة أطر المعلومات وإنما أطر الإقرارات هي التي يمكن أن تضيع أو تتأخر في الوصول.

مخطط التسلسل الزمني للمثال



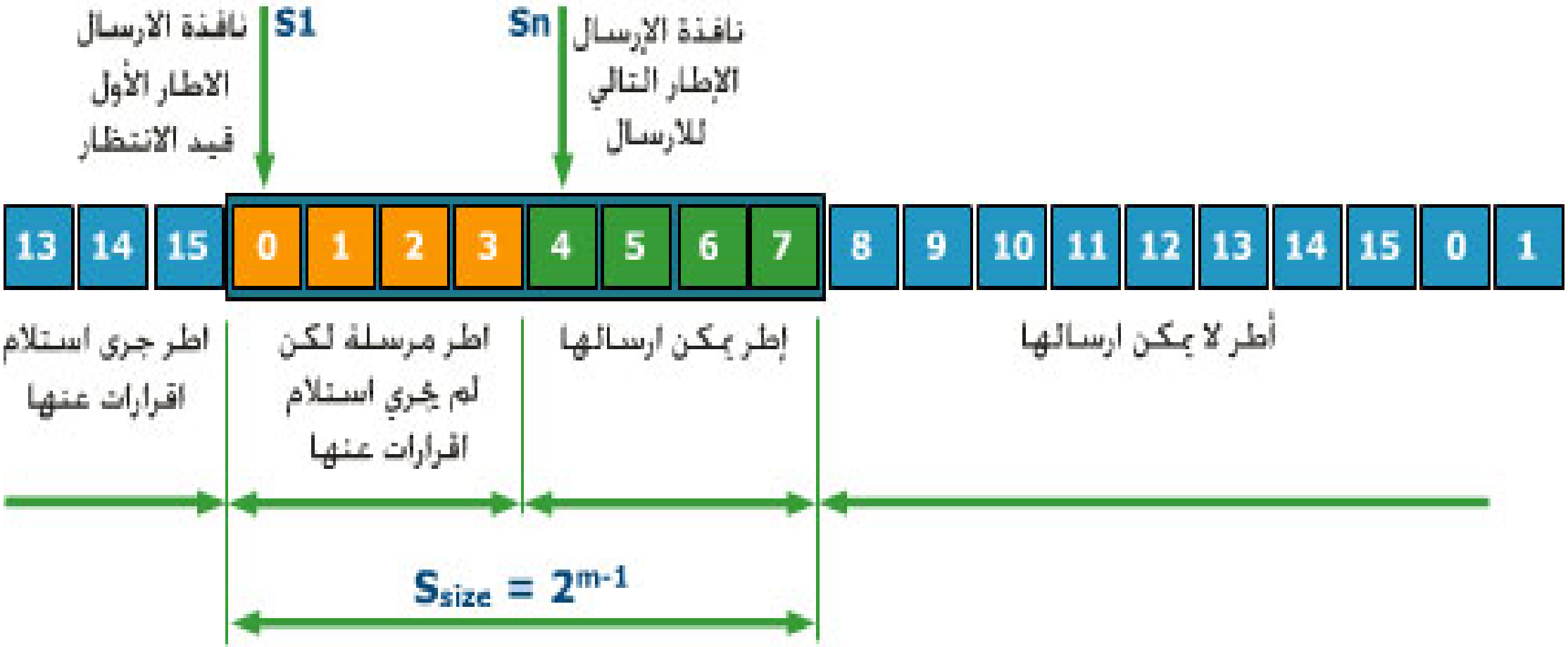
3-بروتوكول تكرار الطلب الآلي مع تكرار انتقائي
Selective Repeat Automatic Repeat Request

- يبسط بروتوكول Go-Back-N ARQ العمل لدى المستقبل لأنه يحتاج لمتابعة متحول واحد فقط دون الحاجة لتخزين الأطر المستقبلة خارج الترتيب . لكن هذا البروتوكول يصبح غير فعالا عند استخدامه مع وصلات ذات ضجيج كبير Noisy حيث يزداد احتمال تشوه الأطر وإعادة الإرسال الأمر الذي يستهلك حزمة المرور ويبطئ حركة النقل . يوجد لهذه الظروف بروتوكولا آخر لا يستدعي إعادة نقل عدة أطر وإنما الإطار المشوه فقط؛ يدعى هذا البروتوكول Selective Repeat ARQ يعتبر هذا أكثر فعالية عندما يوجد ضجيج كبير لكنه يتطلب معالجة أكبر لدى المستقبل.

النـافذة

- يستخدم بروتوكول التكرار الانتقائي نافذتين أيضًا :نافذة إرسال ونافذة استقبال . يختلف هذا البروتوكول عن البروتوكول السابق بما يلي:
- 1. يكون قياس نافذة الإرسال أصغر بكثير من البروتوكول السابق وهي 2اس m-1
- 2.يكون قياس نافذة المستقبل مساوٍ لقياس نافذة المرسل.
- يستخدم البروتوكول نفس متحولات Go-Back-N ARQ التي ناقشناها سابقًا .يبين الشكل التالي مثالا عن قياس النافذة المستخدم

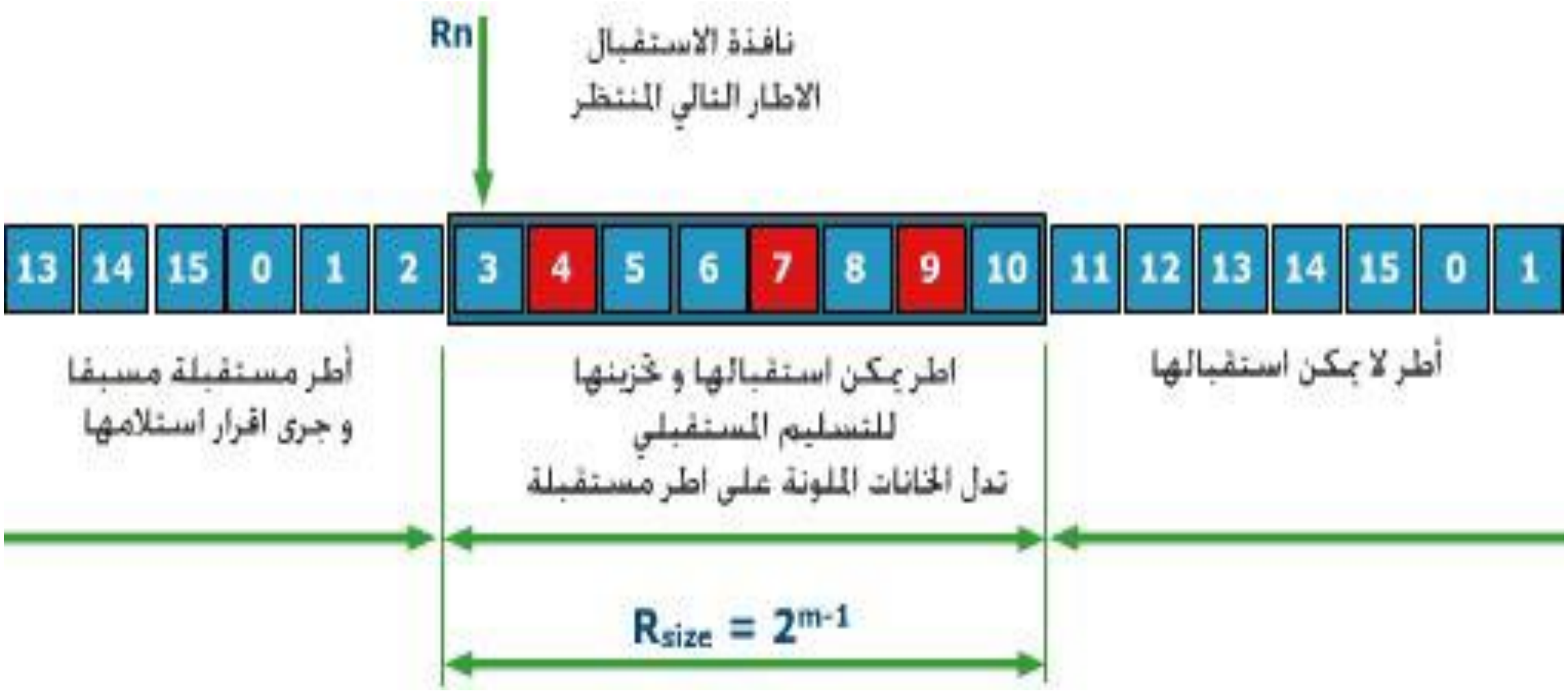
Selective Repeat ARQ نافذة الإرسال لبروتوكول



نافذة المستقبل في بروتوكول Selective Repeat ARQ

- أما بالنسبة لنافذة المستقبل فهي مختلفة تمامًا عن تلك المستخدمة في بروتوكول Go-Back-N ARQ أولا، قياس النافذة مساوٍ لنافذة المرسل . ثانيًا، يسمح بروتوكول التكرار الانتقائي بقبول أطر خارج الترتيب لكن حسب ما يحدده قياس نافذة المستقبل . فعندما يستطيع المستقبل ترتيب
- مجموعة من الأطر السابقة فإنه يرسلها إلى طبقة الشبكة.
- يبين الشكل التالي نافذة المستقبل المستخدمة ضمن البروتوكول

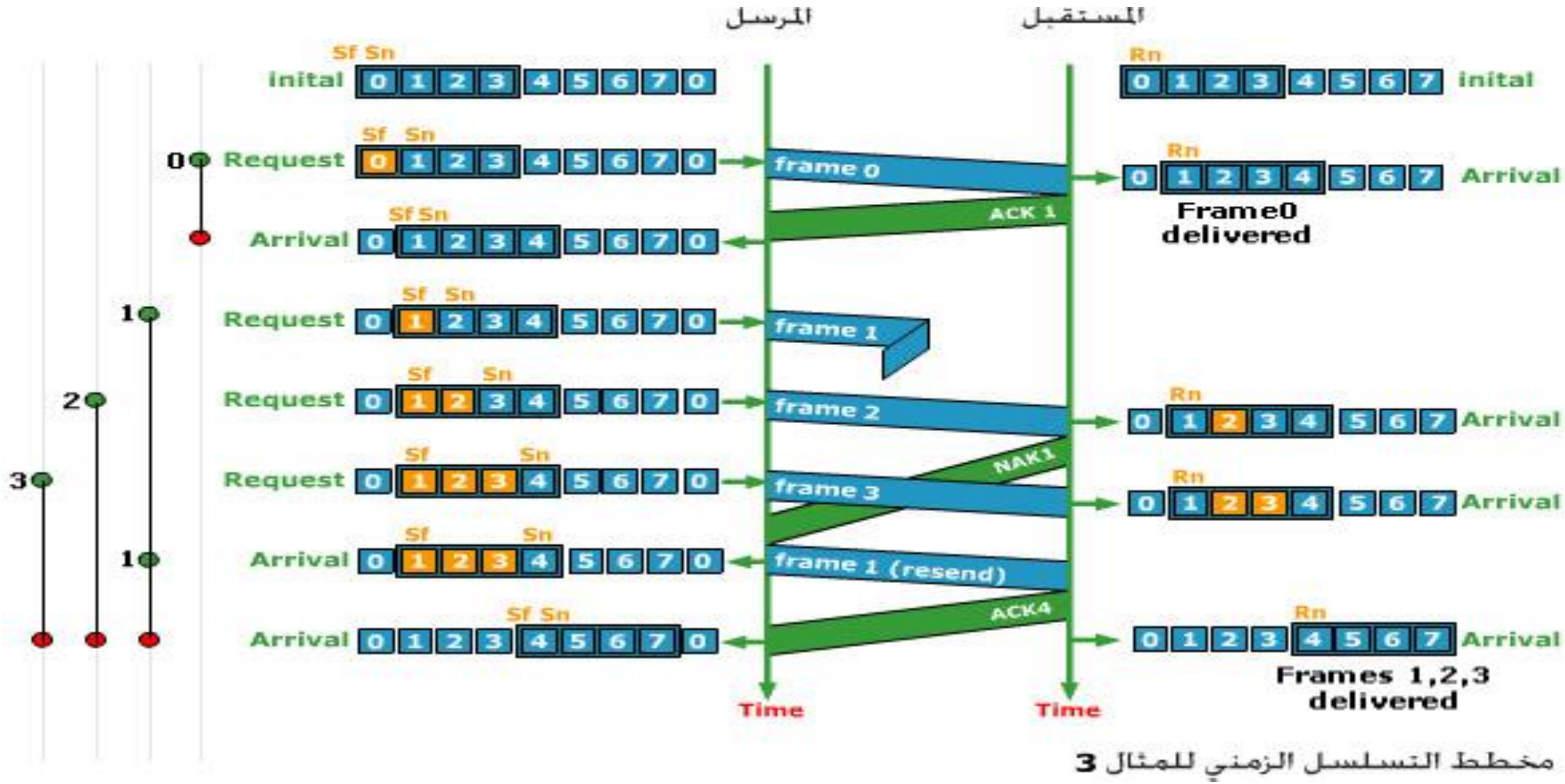
نافذة المستقبل في بروتوكول Selective Repeat ARQ



تحليل البروتوكول عند المرسل

- فيما يخص معالجة الأحداث المتعلقة بالطلبات Requestفهي مشابهة للبروتوكول السابق: ما عدى تخصيص مؤقت زمني لكل إطار مرسل .أما طلبات الوصول Arrival Events فهي أكثر تعقيدًا لأنه يمكن وصول إطار ACKأو NAK.إذا وصل إطار NAK سليم فإننا سنعيد إرسال الإطار المقابل، أما إذا وصل إطار ACKسليم فإننا سندخل في حلقة مؤلفة من تفريغ الذاكرة من الإطارات وتوقيف المؤقتات الزمنية المقابلة وزلق النافذة إلى
- اليسار .بالنسبة لانقضاء المؤقت الزمني فهو أبسط هنا لأننا نحتاج فقط لإعادة إرسال الإطار المقابل له.

يبين الشكل مثالا عن سلوك بروتوكول Selective repeat ARQ. عند ضياع الإطار رقم1



مخطط التسلسل الزمني للمثال 3

الشرح

- لاحظ أننا نحتاج هنا إلى مؤقت زمني لكل إطار يجري إرساله أو إعادة إرساله لذلك فإننا نحتاج الى ترقيم المؤقتات الزمنية حسب رقم الإطار . يقلع المؤقت الزمني للإطار رقم 0 عند وصول الطلب الأ ول Requestويتوقف عند وصول الإقرار Ackعن هذا الإطار بينما يقلع المؤقت الزمني للإطار 1عند وصول الطلب الثاني ويعاد إقلاعه من جديد عند وصول NAKويتوقف عند وصول آخرACK.
- يجب أن نفرق لدى المستقبل بين قبول إطار ما وتسليمه إلى الطبقة الأعلى . لدى الوصول الثاني، الإطار رقم 2يصل إلى المستقبل حيث يجري تخزينه وتعليمه (خانة ملونة) لكن لا يتم تسليمه لأننا لم نستقبل بعد الإطار رقم .1لدى الوصول التالي يجري استقبال وتخزين الإطار رقم 3دون التسليم إلى الطبقة الأعلى أيضًا .فقط لدى الوصول الأخير للإطار رقم 1 يجري تسليم الأطر 1,2,3إلى الطبقة الأعلى

اكتشاف وتصحيح الأخطاء
ضمن طبقة وصلة المعطياتData Link Layer

أنواع الأخطاء

- في كل مرة يجري فيها نقل المعطيات من جهاز إلى آخر فإنه يمكن أن تقع أخطاء في النقل أو تتشوه الطرود المنقولة نتيجة لضجيج خارجي أو تضيع بعض الأطر . يجب أن تضمن الشبكات مطابقة المعلومات المستقبلة لما تم إرساله فعلا .لذلك فإن غالبية التطبيقات تتطلب آليات لاكتشاف الأخطاء أو تصحيحها
- تكون البتات، عندما تنتقل من نقطة إلى أخرى، عرضة للتداخل interference.هذا التداخل يمكن أن يغير قيمة بت واحد أو مجموعة بتات.

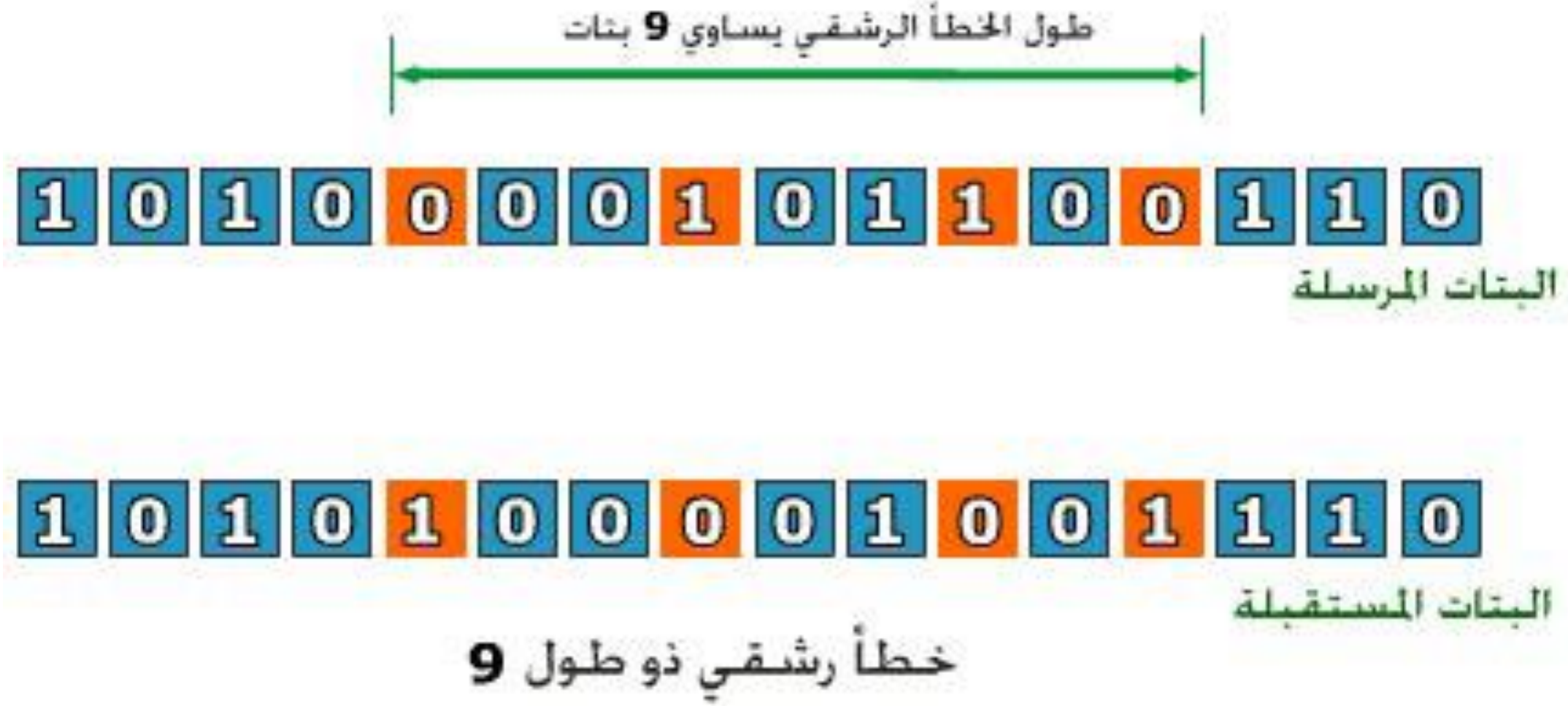
Single-Bit error أخطاء البت الواحد

- يجري هنا تغيير بت واحد فقط من سلسلة البتات المرسلة(بايت أو محرف أو أي وحدة معطيات أخرى)من الصفر المنطقي إلى الواحد أو بالعكس

مثال

- 1بافتراض أن سرعة نقل المعطيات هي 1Mbps وأن مصدر ضجيج خارجي يستمر لفترة100.µs
- .1احسب عدد البتات التي يمكن أن تتشوه أثناء النقل
- .2احسب فترة الضجيج اللازمة لوقوع خطأ ببت واحد فقط
- .3استنتج مما سبق إمكانية حدوث الخطأ ببت واحد.
- **الحل:**
- .1بما أن معدل نقل المعطيات هو 1Mbpsفهذا يعني أن كل بت يستمر لفترة 1µsوبما أن الضجيج الخارجي يستمر100 µs، ينتج من ذلك
- عدد البتات التي يمكن أن تتشوه أثناء النقل =100بت
- فترة الضجيج اللازمة لوقوع خطأ على بت واحد هي 1 µs
- إمكانية وقوع خطأ على بت واحدة نادرة جدًا) باحتمال 1%

الخطأ الرشقيBurst Error



خطأ رشقي ذو طول 9

الخطا الرشقي

- يعني تغيير قيمة بتين أو أكثر ضمن وحدة المعطيات المرسلة
- ويقاس طول الخطأ بالمسافة الفاصلة بين أول بت وأخر بت خضعا للتغيير.

التكرارRedundancy

- حتى نستطيع اكتشاف أو تصحيح الأخطاء فإننا نحتاج إلى استخدام بتات إضافية (تكرارية)مع المعطيات.
- تعتبر مسألة تصحيح الخطأ أكثر تعقيدًا من اكتشافه فقط، وذلك لأننا نحتاج لاكتشاف الخطأ جوابًا من نوع نعم أو لا فقط حتى أننا لا تحتاج للتمييز بين خطأ واحد أو رشقي، بينما نحتاج لتصحيح الخطأ إضافة إلى ما سبق تحديد المكان أو الأمكنة التي وقعت فيه الأخطاء لتصحيحها
- يوجد فعليًا طريقتين لتصحيح الأخطاء :الأول تدعى تصحيح الأخطاء المباشر Forward Error Correctionحيث يسعى المستقبل إلى معرفة مكان أو أمكنة الأخطاء وتصحيحها بينما يتم تحقيق النوع الثاني عن طريق إعادة الإرسال، أي اكتشاف الخطأ من قبل المستقبل وطلب إعادة الإرسال.

الرماز الكتليBlock Coding

- في الرماز الكتلي، نقوم بتقسيم الرسالة الأصلية إلى كتل يتألف كل منها منk bits وتدعى كلمات المعطياتDatawords.نضيف بتات تكرارية r bits إلى كل كتلة بحيث يصبح المجموعn=k + r.
- تدعى الكتل الناتجة كلمات الرماز Codewords.
- لاحظ هنا أنه يجري ترميز ٢ أسk باستخدام٢ أس k k data words k صالحين ويبقى لدينا ٢اس n ناقصا ٢ أس k k Codewords k 2n – 2. codeword's غير صالحين

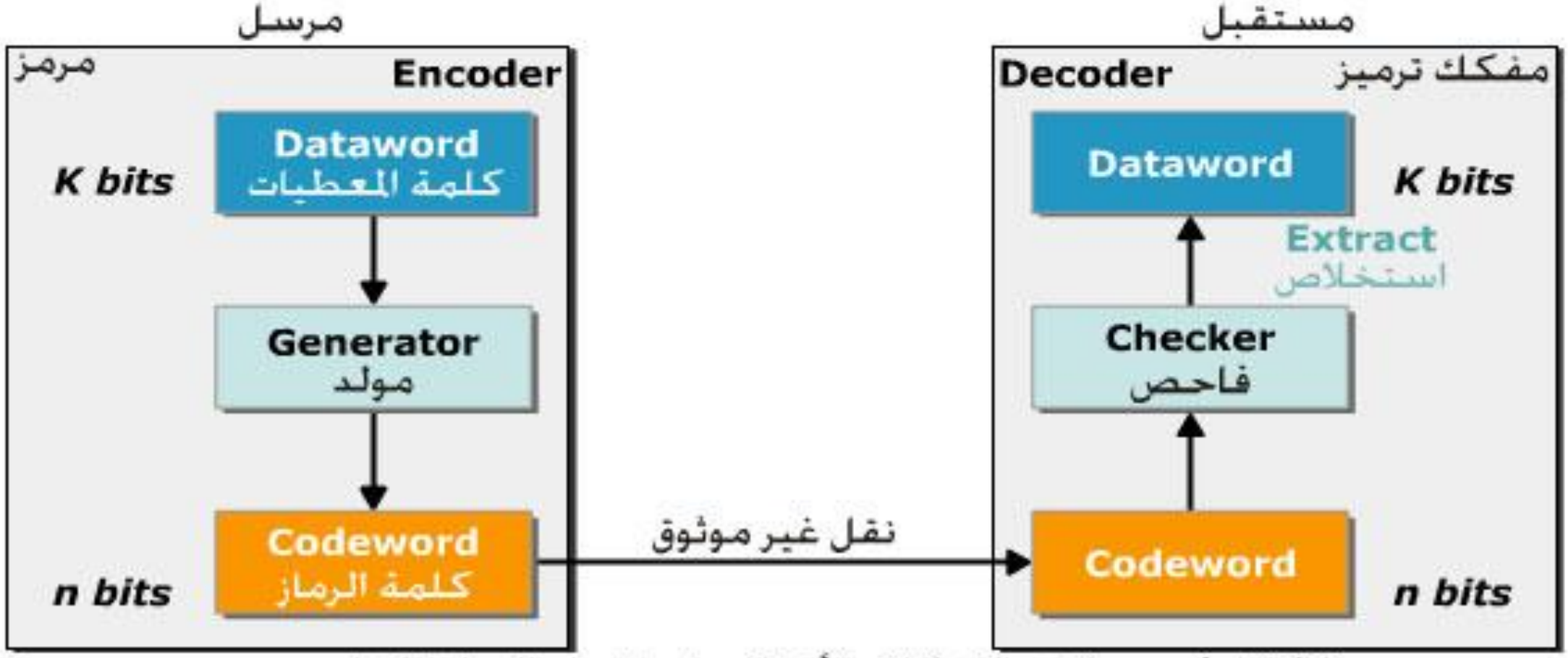
مثال

- لنفترض أن n = 5 and k = 4 هذا يعني أنه يمكننا استخدام ٢ أس ٤ , datawords=16وتعريف ٢ أس ٥=٣٢ .32 codewords أي أنه من بين 32كلمة رماز فإننا نستخدم 16لنقل الرسائل ونترك 16كلمة رماز لأغراض أخرى كالتحكم واكتشاف الأخطاء.

اكتشاف الأخطاء

- يمكن للمستقبل أن يكتشف حدوث تغيير في كلمة المعطيات إذا تحقق الشرطين التاليين:
- •يمتلك المستقبل(أو يمكن أن يحصل على)لائحة بكلمات المعطيات الصحيحة.
- •جرى تغيير كلمة الرماز الأصلية إلى كلمة رماز غير صالحة.
- يبين الشكل التالي طريقة ترميز المعطيات وتفكيك ترميزها.
-

عملية اكتشاف الأخطاء باستخدام الرماز الكتلي



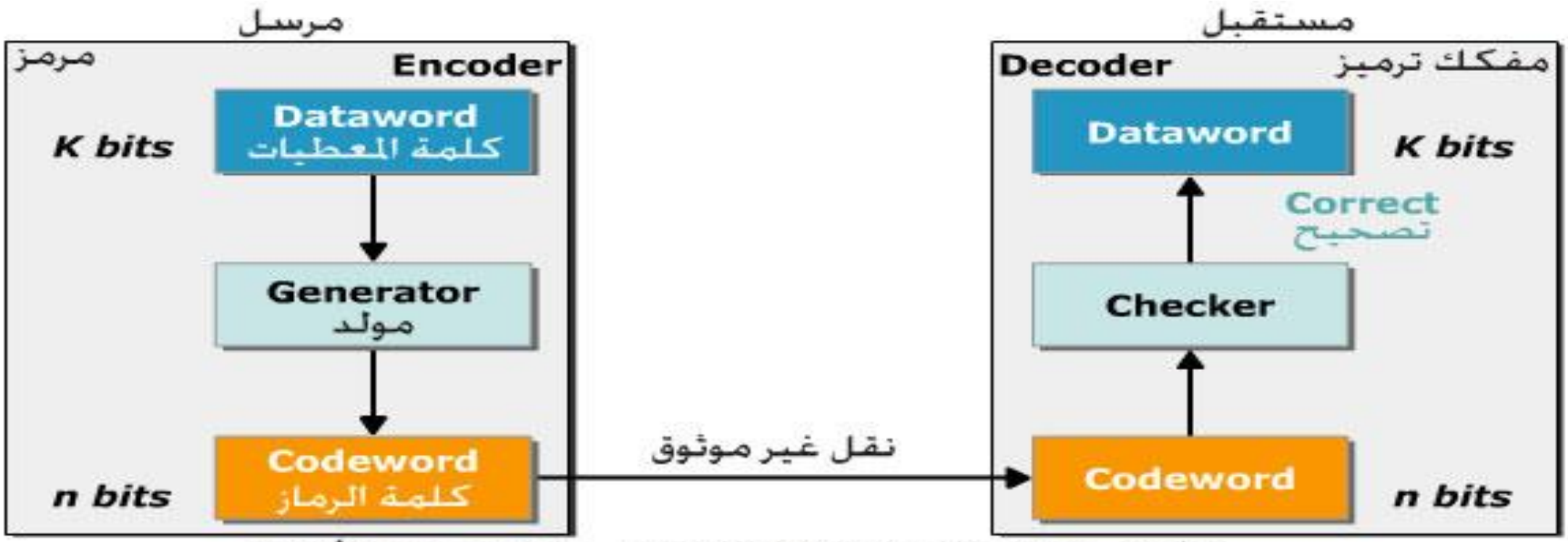
الشكل 4 - عمليات اكتشاف الأخطاء باستخدام الرماز الكتلي

تصحيح الأخطاء

- إن مسألة تصحيح الخطأ تقتضي اكتشاف حالة الخطأ أولا ومن ثم محاولة التعرف على البتات التي وقع عليها الخطأ .لذلك فإننا نحتاج على بتات تكرارية أكثر مما هو عليه الحال في اكتشاف الأخطاء فقط.

الشكل 5 - بنية المرمز ومفكك الرماز في حالة تصحيح الأخطاء

بنية المرمز ومفكك الترميز في حالة تصحيحي الأخطاء



الشكل 5 - بنية المرمز ومفكك الرماز في حالة تصحيح الأخطاء

مثال

- لنحاول إضافة بتات 3 بتات تكرارية على البتين المكونين لكلمة المعطيات لتصبح كلمة الرماز مكونة من 5بتات كما
- هو موضح بالجدول التالي:

Codewords	Datawords
00**000**	00
01**011**	0 1
10**101**	10
11**110**	11

- لنفترض أن المرسل يريد إرسال Dataword = 01 فهو يرمزها ب Codeword = 01011 وإذا افترضنا أيضًا أن الرسالة قد حرفت أثناء النقل ووصل المستقبل الرسالة = Codeword. 01001 يلاحظ المستقبل أولا أن الكلمة الرمازية غير موجودة ضمن جدول التقابلات مما يعني أن خطأ ما قد حدث . يقوم المستقبل عندئذ، مفترضًا أن بتًا واحدًا فقط قد حرف، باستخدام الطريقة التالية ليخمن مكان الخطأ:
- .1مقارنة الكلمة الرمازية المستقبلة مع أول كلمة رمازية ضمن القائمة (00000) مع(01001)، فيلاحظ أنها مختلفة عنها تمامًا ولا يمكن أن تكون هي الكلمة المطلوبة.
- .2بالطريقة نفسها يستنتج المستقبل أن الكلمة الرمازية لا يمكن أن تكون هي الثالثة أو الرابعة ضمن القائمة
- .3فإذًا يجب أن تكون الكلمة الرمازية هي الثانية لأنها الأقرب (تختلف عنها ببت واحد فقط)يقوم المستقبل بتغيير الكلمة الرمازية المستقبلة 01001 بالكلمة 01011 ويعود إلى . القائمة لاستخلاص كلمة المعطيات وهي 01
-

مسافة هامينغ Hamming Distance

- مسافة هامينغ بين كلمتين بطول متساوٍ هي عدد البتات المختلفة بين الكلمتين . وتكتب على الشكل التالي d(x, y) و حيث y هي الكلمة الثانية و x الكلمة الأولى ،يمكننا حساب مسافة هامينغ بين كلمتين بتطبيق XOR (⊕) على الكلمتين وحساب عدد الواحدات من النتيجة.

مسافة هامينغ الصغرى

- تعريف مسافة هامينغ الصغرى بين عدة كلمات رماز على أنها أصغر مسافة هامينغ بين جميع الأزواج الممكنة للكلمات .سنستخدم الاصطلاحdmin.
- أي مخطط رماز يجب أن يحوي على ثلاثة وسطاء وهم
- 1. **k**:طول كلمة المعطيات
- 2. **n**:طول كلمة الرماز
- 3. **dmin**مسافة هامينغ الصغرى

مثال

- إذا كان لدينا مخطط رمازي ذو مسافة هامنك صغرى تبلغ
- dmin = 4أوجد عدد الأخطاء ،القادر على اكتشافها وتلك القادر على تصحيحها.
- **الحل**
- S+1=4
- يضمن هذا المخطط اكتشاف حتى 3أخطاء s = 3 .لكنه يضمن تصحيح خطأ واحد فقط

Linear Block Codes الرماز الكتلي الخطي.

- ضمن الرماز الكتلي الخطي، تطبيق
- XOR (addition modulo- 2)على كلمتي رماز صالحتين يعطي كلمة رماز صالحة.
- **المسافة الصغرى للرماز الكتلي الخطي :** هو أقل عدد للواحدات الموجودة ضمن كلمة رماز لا تساوي الصفر.

بعض الرمازات الكتلية الخطية

- رماز اختبار الزوجية المبسط Simple parity-check code
- رمازات هامينغ
- الرمازات الدوَّارة Cyclic Codes

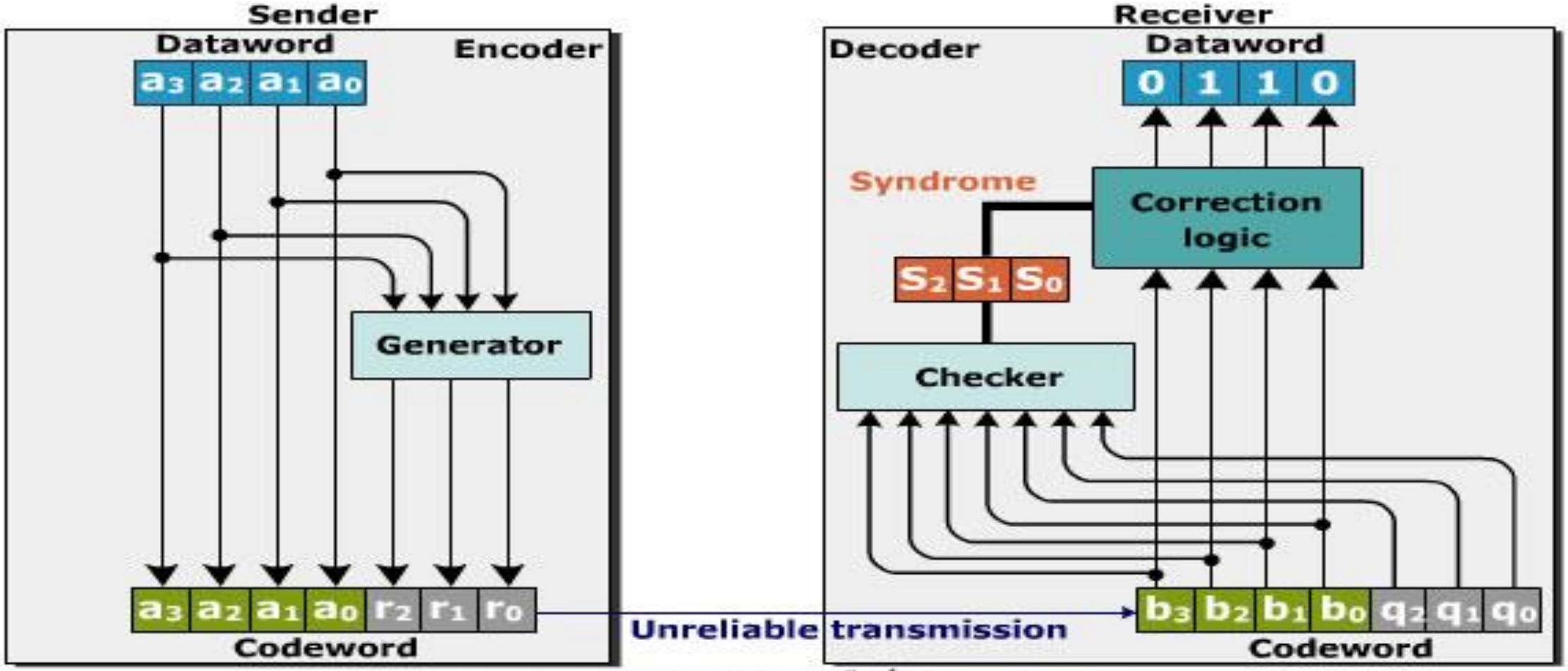
1رماز اختبار الزوجية المبسط Simple parity-check code

- يتم هنا استخدام كلمات معطيات ذات طول k وكلمات رماز ذات طول n = k +1
- يتم اختيار البت الإضافي بشكل يصبح عدد الواحدات الكلي ضمن كلمة الرماز زوجي.،
  تكون مسافة هامينغ الصغرى لهذا النوع من الرماز dmin = 2أي أنه يسمح باكتشاف خطأ على بت واحد ولا يسمح بتصحيح أي خطأ.
- يسمح رماز اختبار الزوجية المبسط باكتشاف الأخطاء على الأعداد الفردية من البتات.

رمازات هامينغ

- تسمح هذه الرمازات بتصحيح الأخطاء عن طريق استخدام مخطط رمازات يحقق dmin = 3أي أنه قادر على اكتشاف حتى خطأين وتصحيح خطأ واحد .لنبدأ بحساب العلاقة بين k وn وفق رماز هامينغ نحتاج إلى اختيار عدد طبيعي
- $m \geq 3$.نستطيع أن نستنتج من m قيمةn
- والتي تساوي $k = n - m$ و $n = 2m - 1$.
- لنأخذ m =3 مما يؤدي إلى كون n =7 هذا يعطينا رماز هامينغ C(7, 4) .حيثdmin =3.

بنية المرمز ومفكك الترميز لرماز هامنك



الشكل 10- بنية المُرمِّز ومفكك الرماز لرماز هامنك

مبدأ عمل المرمز

- لاحظ أن المولد Generatorيأخذ نسخة من البتات الأربعة a0, a1, a2, a3ويُنشأ 3بتات زوجية
- هي
- كما هو موضح فيما يلي r0, r1, r2 :
- modulo-2 r0 = a2 + a1 + a0
- modulo-2 r1 = a3 + a2 + a1
- modulo-2 r2 = a1 + a0 + a3
- و يقوم الفاحص checkerلدى المستقبل بتوليد مجموعة تدعى Syndromeمكونة من 3بتات
- حيث(s2s1s0) :
- modulo-2 s0 = b2 + b1 + b0 + q0
- modulo-2 s1 = b3 + b2 + b1 + q1
- modulo-2 s2 = b1 + b0 + b3 + q2
- تسمح البتات الثلاثة 0 s2s1s)بقيمها المختلفة بين 000و 111لنا بمعرفة حالات عدم وجود
- أخطاء أو وجود خطأ في بت محدد حسب الجدول.
- **Syndrome(s2s1s0)** 000 001 010 011 100 101 110 111
- **Error** None q0 q1 b2 q2 b0 b3 b1

- أي أنه إذا كانت نتيجة الفاحص s2s1s0 =0 0هذا يعني أنه لا توجد أخطاء في النقل أما إذا كانت s2s1s0 =001فهذا يعني أن يوجد خطأ في البت q0لكنه لا يهم بالنسبة للمستقبل لأنه بالأصل بت تكراري q، أما إذا كانت s2s1s0 =011فهذا يعني أنه يوجد خطأ في البت b2 مما يستوجب قلب قيمة هذا البت.

الرمازات الدوَّارة Cyclic Codes

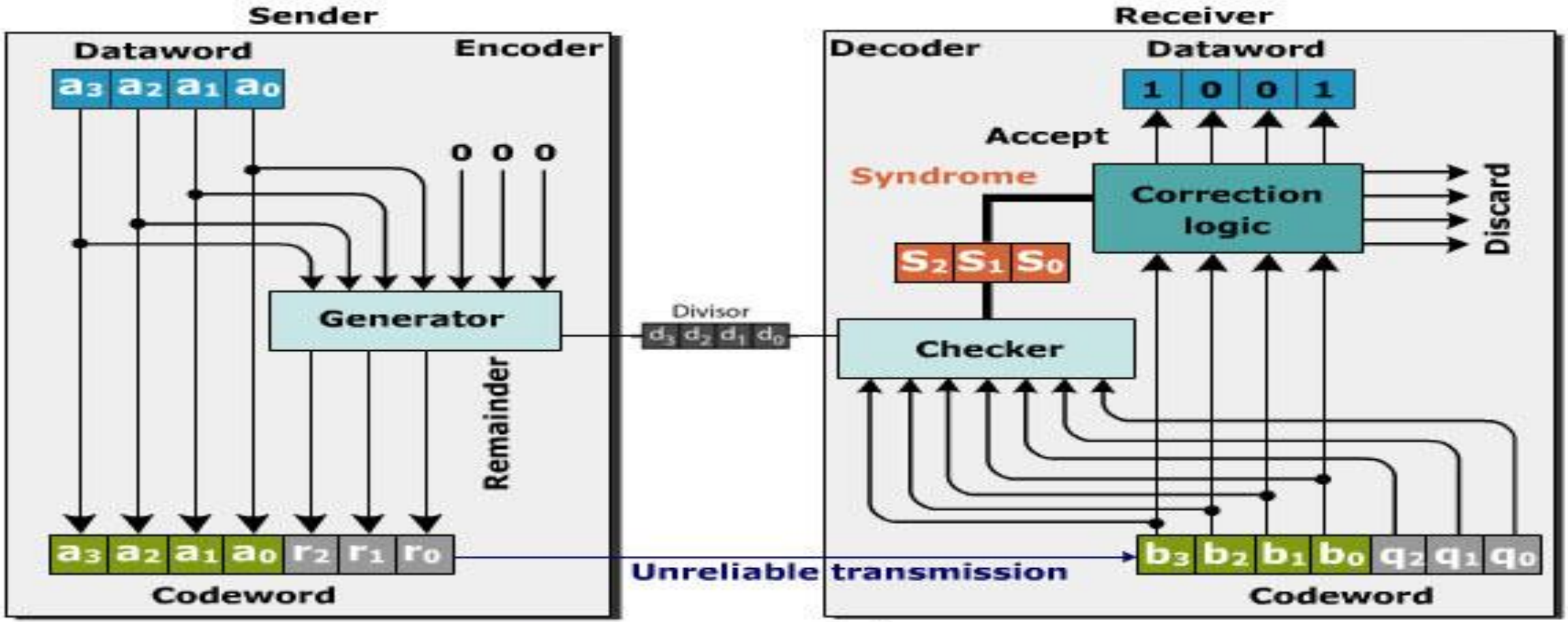
- تعتبر الرمازات الدوَّارة حالة خاصة من الرماز الكتلي الخطي مع وجود خاصة واحدة .إذا أزيحت(دُورت)كلمة رماز سنحصل على كلمة رماز أخرى . فإزاحة كلمة الرماز 1011000دائريًا إلى اليسار تنتج عنه الكلمة 0110001والتي هي أيضًا كلمة رماز . فإذا اعتبرنا أن كلمة الرماز الأصلية هي a0 to a 6والكلمة الناتجة عن الإزاحة الدائرية b0 to b6يتم تغيير البتات وفق الطريقة التالية:
- b1 = a0 b2 = a1 b3 = a2 b4 = a3 b5 = a4 b6 = a5

Check (CRC)Cyclic Redundancy اختبار التكرارية الدوارة

- يستخدم اختبار التكرارية الدوارة في الشبكات المحلية والواسعة . يبين الجدول مثالا عن رماز دوار.CRC

Dataword	Codeword	Dataword	Codeword
0000	0000**000**	1000	1000**101**
0001	0001**011**	1001	1001**110**
0010	0010**110**	1010	1010**011**
0011	0011**101**	1011	1011**000**
0100	0100**111**	1100	1100**010**
0101	0101**100**	1101	1101**001**
0110	0110**001**	1110	1110**100**
0111	0111**010**	1111	1111**111**

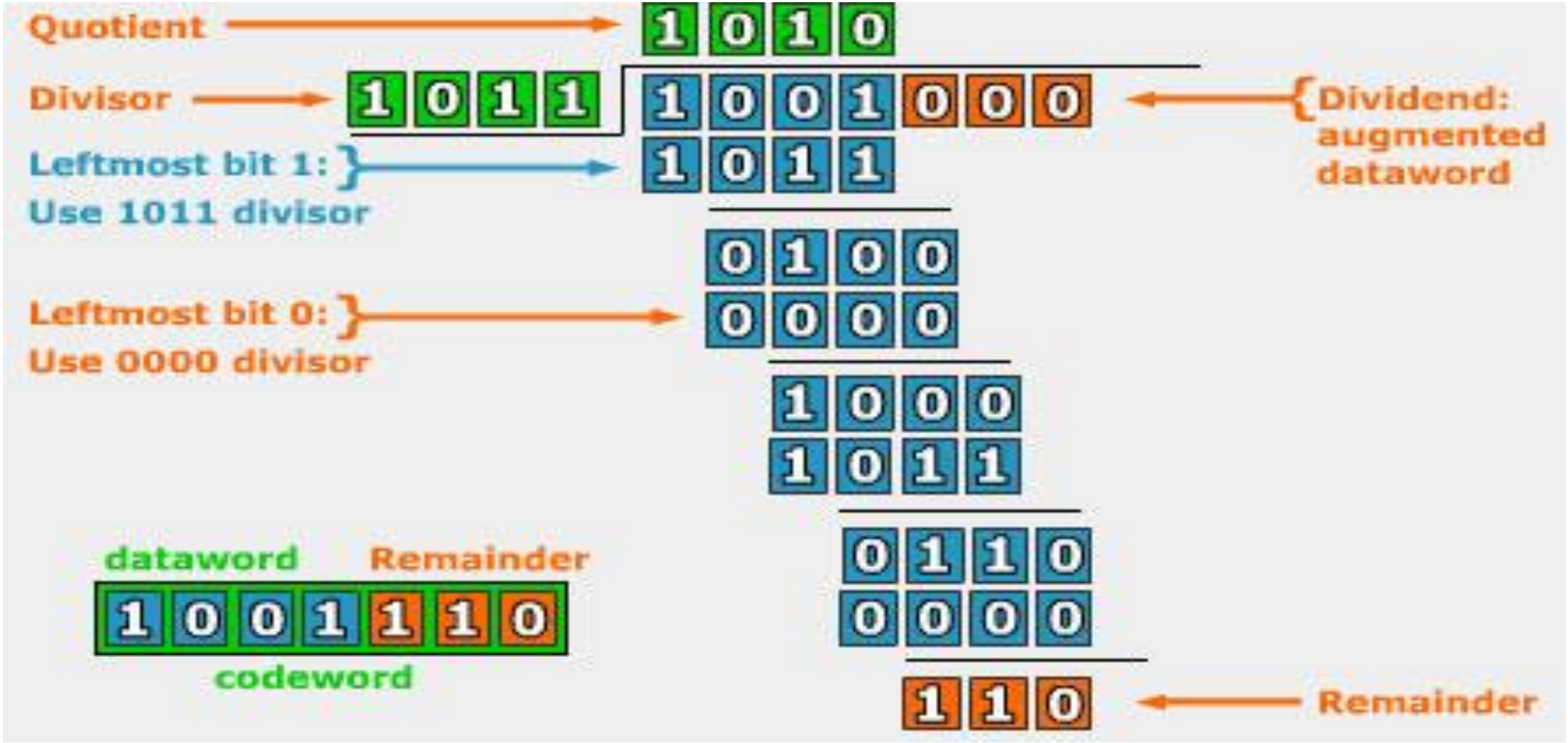
مرمز ومفكك ترميزCRC



الشكل 12- مُرمِّز ومفكك ترميز CRC

- تتألف كلمة المعطيات عند المرمز من k bits (٤في حالتنا هذه) ؛ أما كلمة الترميز فهي مؤلفة n bits(7في حالتنا هذه).تتم زيادة n-k bits أي 3من بتات ذات القيمة 0إلى الطرف اليميني من كلمة المعطيات ندخل الناتج (n bits)إلى المولد generator. يستخدم المولد قاسم divisorمتفق عليه مسبقًا بطول n – k + 1أي 4بتات هنا.(يقوم المولد بتقسيم كلمة الترميز الموسعة على القاسم قسمة مودولو ٢ ؛ نهمل حاصل القسمة ونضيف باقي القسمة(reminder (r2r1r 0 إلى كلمة المعطيات لتكوين كلمة الترميز.
- يستقبل مفكك الترميز كلمة المعطيات التي يمكن أن تكون محرفة .ندخل نسخة عن n bits إلى الفاحص الذي هو صورة طبق الأصل عن المولد .يحسب الفاحص باقي القسمة المكون من n – k bits(3هنا)وتنقل النتيجة إلى محلل القرار المنطقي .تتمثل وظيفة محلل القرار المنطقي بقبول البتات الأربعة اليسارية من كلمة الترميز إذا كانت أي S2S1S= 0 syndrome bits are all0s 000وإلا فإنه يتم إهمال كلمة الترميز

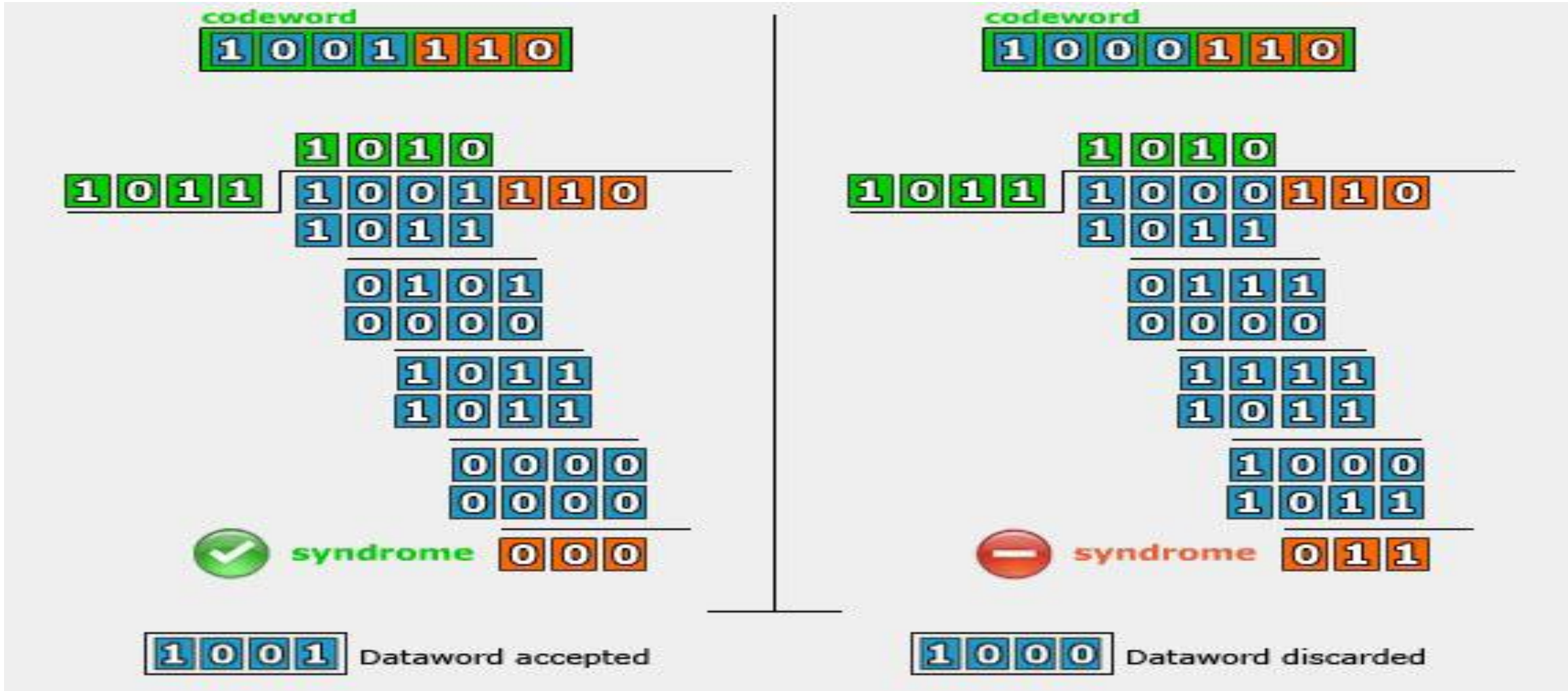
Encoder المرمز



- لاحظ هنا أن كلمة المعطيات الأصلية هي 1001التي يقوم المرمز بإضافة(3) بتات 000 إليها وبتقسيم الناتج 1001000على القاسم المتفق عليه 1011والذي يتألف من 4بتات . تتم القسم الثنائية مود ولو 2 - كما هو عليه الحال في القسمة العشرية لكن بما أن الجمع والطرح متكافئين .فإننا نستخدم XORلتحقيقهما. في كل خطوة من القسمة نأخذ نسخة من القاسم ونطبق عليها XOR مع المقسوم .تستخدم نتيجة XOR(3بتات في حالتنا هذه) ضمن الخطوة التالية بعض سحب بت واحد لجعل طوله 4بتات هناك نقطة أساسية يجب تذكرها في هذا النوع من القسمة : إذا كان البت اليساري من المقسوم 0فإننا نستخدم قاسم صفريًاall-0s divisor.عندما لا يعود هناك بتات تسحب إلى الأسفل، نحصل على نتيجة القسمة . تشكل بتات باقي القسمة(r2, r1, r 0) . بتات الاختبار حيث تتم إضافتها إلى كلمة المعطيات لتكوين كلمة الترميز

مفكك الترميز Decoder

- بما أن كلمة الترميز يمكن أن تتغير أثناء الإرسال، فإن مفكك الترميز يقوم بنفس عملية القسمة .مثله مثل المرمز . ويكون باقي عملية القيمة هو ما يعرف بالأعراض Syndrome إذا كان Syndrome = 0فلا يوجد أخطاء ويتم فصل كلمة المعطيات عن كلمة الترميز وقبولها .إذا كان Syndromeلا يساوي الصفر فتهمل كلمة الترميز وكلمة المعطيات . يبين الشكل 10حالتين:
- القسم اليساري يوضح عدم وجود أخطاء في النقل والقسم اليميني يوضح حالة وجود خطأ واحد حيث يكون باقي القسمة مختلف عن الصفر.


codeword
1 0 0 1 1 1 0
1 0 1 1
1 0 1 0
1 0 0 1 1 1 0
1 0 1 1
0 1 0 1
0 0 0 0
1 0 1 1
1 0 1 1
0 0 0 0
0 0 0 0
syndrome
0 0 0
1 0 0 1
Dataword accepted
codeword
1 0 0 0 1 1 0
1 0 1 1
1 0 1 0
1 0 0 0 1 1 0
1 0 1 1
0 1 1 1
0 0 0 0
1 1 1 1
1 0 1 1
1 0 0 0
1 0 1 1
syndrome
0 1 1
1 0 0 0
Dataword discarded