Σ 4,5 /5

Rechnernetze-Media Networking
Uebung 03
Gruppe 15
17.11.2023

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

Rechnernetze – Media Networking (WiSe 2023/2024)

Übung 03

Aufgabe 1) In zuverlässigen Multicast-Transport-Protokollen wird nicht selten sichergestellt, dass die Nachrichten bei den Gruppenmitgliedern in Consistent Time Ordering ankommen. Skizziert je ein (fiktives) Beispiel für eine Gruppenkommunikation mit mehreren Sendern, in dem Consistent Time Ordering der Nachrichten

a) sinnvoll wäre (und ein schwächeres Verfahren nicht ausreichen würde),
b) nicht ausreichen würde (also ein noch stärkeres Ordnungskritierium der Nachrichten erforderlich wäre).

a) Consistent Time Ordering (CTO) ist ein zuverlässiges Multicast-Transport-Protokoll bei dem alle Nachrichten, bei allen noch aktiven Gruppenmitgliedern, in derselben Reihenfolge ausgeliefert werden, die aber nicht der globalen Senderreihenfolge entsprechen muss.
Das ist wichtig, wenn es um Anwendungen geht, bei denen die korrekte Reihenfolge von Ereignissen entscheidend ist, wie zum Beispiel in synchronisierten Anwendungen.
Gelöst wird das dadurch, dass es in der Gruppe einen ausgezeichneten Knoten gibt (Kann der Empfänger sein oder Sonderverwaltung Knoten), der seine Empfangs-Reihenfolge an die anderen Teilnehmer weiter meldet und somit bei allen Knoten dieselbe Reihenfolge umgesetzt werden kann.

Das garantiert CTO gerade nicht

Im Folgenden ist ein fiktives Beispiel für eine Gruppenkommunikation mit mehreren Sendern, in dem Consistent Time Ordering der Nachrichten sinnvoll wäre. Angenommen, es gibt eine Gruppe von Empfängern (Gruppenmitglieder), die alle eine gemeinsame Aufgabe in der gleichen Reihenfolge ausführen müssen, und es gibt mehrere Sender, die Daten an die Gruppe senden.

Beispiel ohne das Consistent Time Ordering (CTO):

- Sender A sendet Nachricht 1 (m1) an die Gruppe G
- Sender B sendet Nachricht 2 (m2) an die Gruppe G
- Aufgrund von Netzwerkverzögerungen kommt m2 vor m1 bei G an, obwohl Sender A zuerst seine Nachricht schickte (siehe Abb. 1).

Die Verzögerung könnte jedoch auch bei B stattfinden (oder es läuft alles gewöhnlich ohne unerwartete Verzögerung ab). Dann würde die Reihenfolge der Nachrichten bei bestimmten Gruppenmitgliedern anders ankommen als bei anderen.

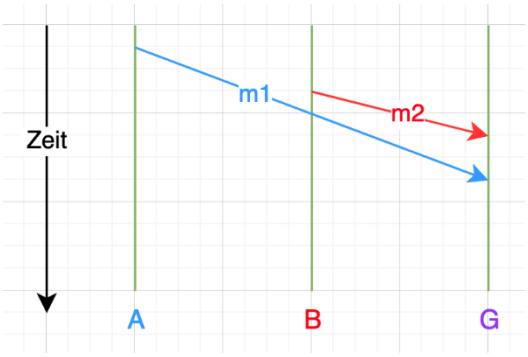

Abbildung 1 Ohne CTO: Empfang bei Gruppe G: [m2, m1] (oder [m1, m2], wenn die Verzögerung bei B stattfindet und hinreichend groß ist).

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

Ohne CTO könnten also einige Gruppenmitglieder die Nachrichten in unterschiedlicher Reihenfolge erhalten, was zu Unstimmigkeiten in der Ausführung der gemeinsamen Aufgabe führen könnte.

Hinweis: Die Skizze ist sehr vereinfacht dargestellt, Sender A und Sender B würden mehrere Nachrichten an G senden. Bei den einzelnen Mitgliedern von G ist die Reihenfolge der jeweiligen Nachrichten unterschiedlich, obwohl manchmal A oder B zuerst sendet (z.B. wegen der Verzögerung).

Eine etwas ausführlichere Skizze ist in Abbildung 2 zu erkennen.

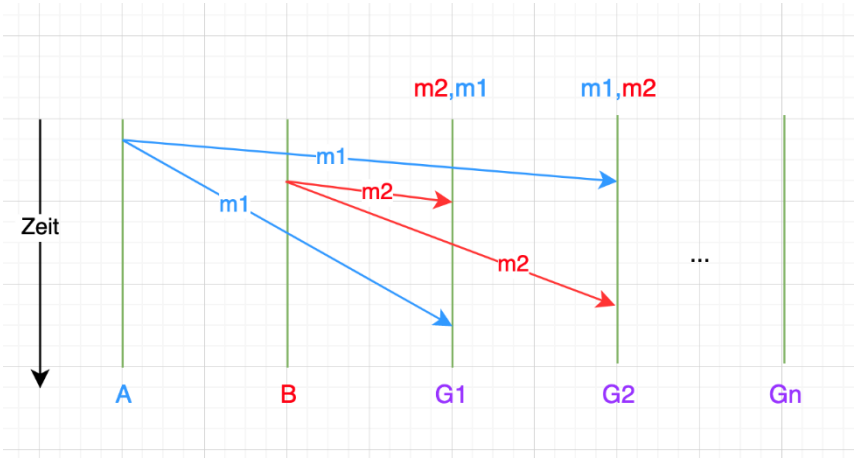

Abbildung 2 hier sind die Mitglieder der Gruppe G genauer abgebildet. Gruppe G hat bis zu n Mitglieder. G1 erhält die Nachrichten in der Reihenfolge [m2, m1], G2 hingegen in der Reihenfolge [m1,m2].

Mit Consistent Time Ordering (CTO):

- Es gibt einen ausgezeichneten Knoten (zum Beispiel ein Sonderverwaltungsknoten C), der die empfangene Reihenfolge der Nachrichten festhält
- Sender A sendet Nachricht 1 (m1) und Sender B sendet Nachricht 2 (m2) an den Sonderverwaltungsknoten C

- Der Sonderverwaltungsknoten C stellt sicher, dass alle Gruppenmitglieder G die Nachrichten in derselben Reihenfolge erhalten, unabhängig von der tatsächlichen Übertragungsreihenfolge

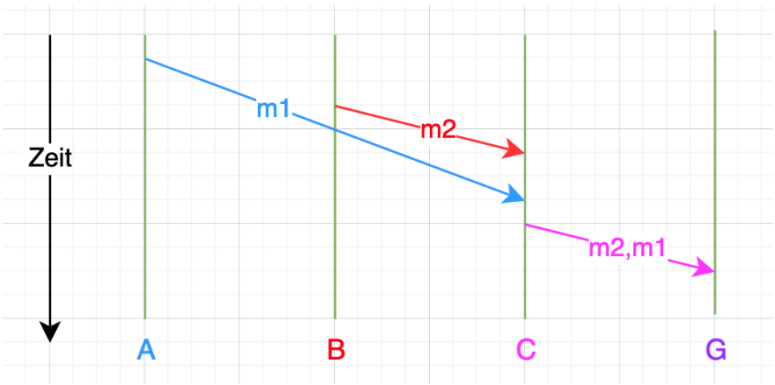

Abbildung 3 Mit CTO: Empfang bei Sonderverwaltungsknoten: [m2, m1] = Empfang bei G: [m2, m1]. Alle Gruppenmitglieder erhalten die gleiche Reihenfolge von C.

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

Durch die Implementierung von CTO wird sichergestellt, dass alle Gruppenmitglieder die Nachrichten in der gleichen Reihenfolge empfangen, selbst wenn die Netzwerkübertragung die Reihenfolge der tatsächlichen Sendungen verändert. Denn der Sonderverwaltungsknoten stellt sicher, dass die zeitliche Reihenfolge der empfangenen Nachrichten gemerkt und diese Reihenfolge dann an alle Gruppenmitglieder weitergeleitet wird.
Bezüglich anderer Ansätze wie z.B. dem "Global Time Ordering" wird vorausgesetzt, dass alle Nachrichten in einer globalen Senderreihenfolge geliefert werden, was möglicherweise nicht immer die flexibelste Lösung ist. Insbesondere in Systemen mit vielen Sendern könnte dies zu einer begrenzten Skalierbarkeit führen. Zudem wird eine zeitsynchrone Umgebung, also möglicherweise Zeitstempel, in den Nachrichten benötigt, um eine genaue Reihenfolge im Empfängersystem sicherzustellen. Die Umsetzung kann sehr umständlich sein.

b) Ein Szenario, in dem ein noch stärkeres Ordnungskriterium als Consistent Time Ordering (CTO) erforderlich ist, könnte auftreten, wenn die genaue zeitliche Abhängigkeit zwischen den Nachrichten entscheidend ist oder eventuell eine Priorität besteht, welche Nachrichten zuerst erreicht werden soll.
In solchen Fällen könnte CTO möglicherweise nicht ausreichen, da es sich nur auf die Reihenfolge des Empfangs bezieht, nicht jedoch auf die absolute Zeit, zu der die Ereignisse stattfinden oder sonstige Faktoren berücksichtigt werden, die wichtig sein könnten. Angenommen, es gibt zwei Sender A und B. Diese schicken ihre Nachrichten über das CTO an ein Sonderverwaltungsknoten, welcher eine feste Reihenfolge empfängt und diese an den Empfänger schickt (siehe Abb. 4).

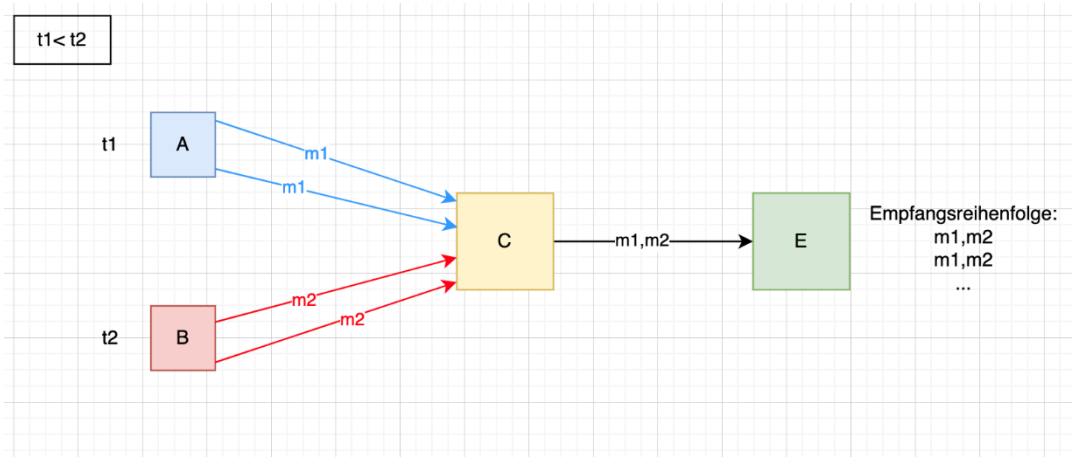

Abbildung 4 Mit CTO, t1 < t2

Für den Fall, dass die Empfangsreihenfolge zeitabhängig ist, wäre CTO keine geeignete Lösung, da die Reihenfolge immer gleich ist. Also zu bestimmten Zeiten soll zuerst m1 dann m2

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

empfangen werden und zu anderen zuerst m2 dann m1. Es könnte sein, dass der Empfänger die Nachrichten in der Reihenfolge m1,m2 empfangen muss, nur wenn t1< t2 ist.
Falls wir ein zeitabhängiges System betrachten und t1 zu einem späteren Zeitpunkt größer als t2 ist, dann sollte gelten, dass die Reihenfolge m2,m1 gelten muss.

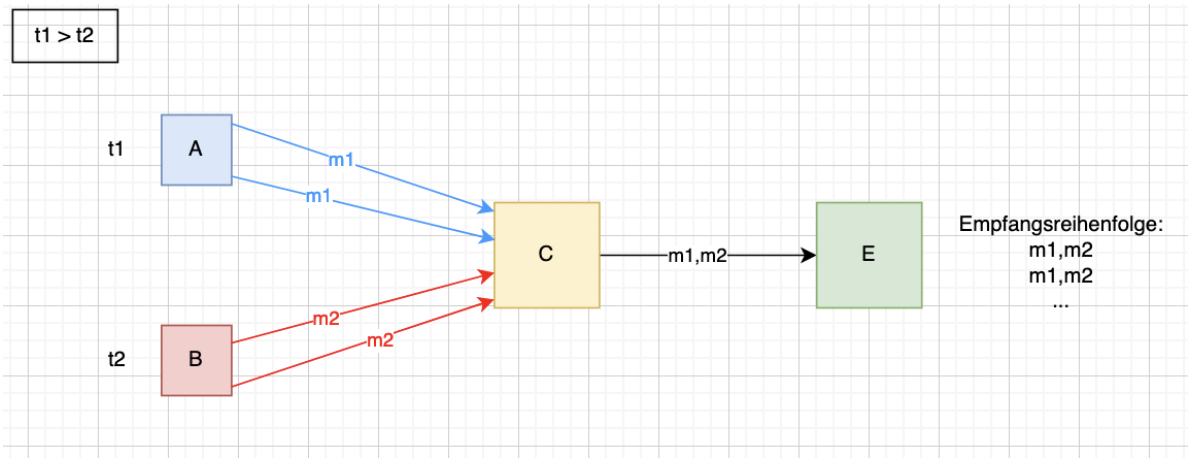

Abbildung 5 Mit CTO, t1>t2

Im Fall von CTO wird aber die zeitliche Abhängigkeit nicht berücksichtigt und die Nachrichten kommen weiterhin in derselben Reihenfolge an.

Ein konkretes Beispiel dafür sind zwei Online-Streaming-Dienste A und B. Diese senden ihren Inhalt an einen zentralen Server, welcher den dann weiter an die Benutzer teilt.
In diesem Beispiel soll die Reihenfolge der empfangenen Inhalte zeitabhängig sein.

Für (t1 < t2):

- Sender A streamt zum Zeitpunkt t1 eine Live-Übertragung (m1).
- Sender B startet seine Übertragung zum Zeitpunkt t2 (m2).

Der Empfänger soll die Nachrichten in der Reihenfolge m1, m2 empfangen, wenn die Zeit t1 vor t2 liegt.

Für (t1 > t2):

- Sender A beginnt seine Übertragung zum Zeitpunkt t1 (m1).
- Sender B startet zum Zeitpunkt t2 (m2).

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

In diesem Fall soll der Empfänger die Nachrichten in der Reihenfolge m2, m1 empfangen, wenn die Zeit t1 nach t2 liegt. Mit dem bestehenden CTO, bei dem die Reihenfolge unabhängig von der Zeit immer gleichbleibt, erhalten die Empfänger jedoch in beiden Fällen die Nachrichten in der Reihenfolge m1, m2 erhalten.

2 Punkte

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

Aufgabe 2) In einer fiktiven Anwendung wird MTP als zuverlässiges Multicast-Transport-Protokoll verwendet. Station A sendet eine MTP-Nachricht an ihre Gruppenmitglieder. Wie erfährt A, dass die Nachricht komplett verloren gegangen ist?

Nachdem Station A eine MTP-Nachricht an ihre Gruppenmitglieder gesendet hat, muss der Empfänger ermitteln können, was fehlt. Das Prinzip von MTP ist, dass grundsätzlich jede Nachricht in mehrere Pakete zerlegt wird, und diese Pakete werden in abgegrenzten Zeitintervallen gesendet, "Heartbeats", welche nicht bei allen Gruppenmitgliedern synchron sein müssen. Es wird allerdings gefordert, dass in jedem Heartbeat mindestens ein Paket der aktuellen Nachricht versendet und entsprechend empfangen werden muss. Die Pakete einer Nachricht werden durchnummeriert und es existiert ein explizit markiertes letztes Paket einer Nachricht. Dies hilft einem Empfänger zu erkennen, ob ein Paket einer Nachricht fehlt. Er kann erkennen, ob es eine Lücke in der Nummerierung gibt, in dem Paketstrom, den er bekommt, und kann prüfen, ob kein Paket der Nachricht im Intervall kam und auch das Endpaket der Nachricht dann nicht empfangen wurde. In solch einem Fall reagiert er am Ende des Heartbeats mit einem NAK. Erhält der Sender nun solch ein NAK-Paket mit der Information, was der Empfänger vermisst hat, kann der Sender die entsprechenden Pakete wiederholen.
Das bisher beschriebene Verfahren funktioniert nur, wenn der Empfänger den Anfang der Nachricht mitbekommen hat, denn dann erwartet er in den folgenden Heartbeats
weitere Pakete dieser Nachricht bis zum entsprechend markierten Ende-Paket.
Wenn jetzt aber schon das erste Paket nicht angekommen ist, dann weiß er ja gar nicht, dass diese Nachricht begonnen wurde und erwartet dementsprechend auch nichts (...).
Das Senden von Zustandsinformationen, insbesondere die Message-Acceptance-Records (MAR), werden auch in leeren Heartbeats verteilt, also Heartbeats, in denen gar keine Nachrichten geschickt werden. Dies erlaubt es allen Knoten im MTP-Web ihren eigenen Empfangszustand regelmäßig zu überprüfen. Kommt also in einem Heartbeat gar nichts an, dann ist entweder der Master ausgefallen oder man selber ein Problem mit der Empfangsfähigkeit (z.B. von der Gruppe abgeschnitten, durch defekte Kommunikationsleitung). Das heißt, dass alle Nachrichten auf n-Heartbeats ausgeweitet werden musste, egal ob sie wirklich so lang waren oder nicht (empty packets). Dadurch konnte man die Existenz einer Nachrichtenübertragung sicher erkennen (da davon ausgegangen wurde, dass kein temporärer Ausfall länger als n-1 Heartbeats lang ist - aber kann auch aus dem MAR erkannt werden). Des Weiteren wird dadurch geprüft, ob der eigene Empfangszustand identisch ist mit dem Masterzustand. Wenn dies wahr ist, dann ist die Gruppenzugehörigkeit per Definition noch gegeben und man ist synchron mit dem Master. Empfängt man MAR, aber der Master meldet, dass er eine Nachricht schon komplett bekommen hat, und der unten markierte Knoten z.B. nicht, dann wäre es an

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

der Zeit, einen NAK zu senden. Wenn dennoch keine Pakete ankommen, dann ist der Empfänger möglicherweise aus der Gruppe rausgefallen. Zusammenfassend erfolgen Retransmissions auf der Grundlage von NAKs. Es gibt ein Ratenkontrollverfahren und es wird CTO sichergestellt. Dies wird durch eine Master-Station realisiert, in der die Vergabe von Token stattfindet und dass pro Heartbeat 1 bis k Pakete einer Nachricht gesendet werden und der Master in jedem Heartbeat seinen MAR an die Gruppenmitglieder verteilt. Dies wird ersichtlich in Abbildung 6 visualisiert – Beantragung und Gewährung des Tokens (links) und Multicast Verteilung an die Gruppenmitglieder, sowie das Senden des NAK-Pakets an den Sender, falls ein Paket verloren gegangen ist (rechts).

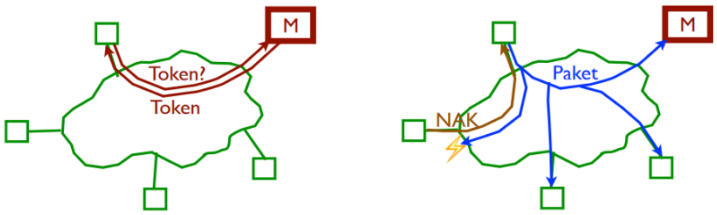

Abbildung 6 Token-Master (links), Multicast Verteilung an Gruppenmitglieder (rechts)

Als Ergänzung dazu, könnte eine Timeout-Funktion implementiert werden: Nachdem eine Nachricht gesendet wurde, wird seitens A ein Timer gestartet. Wenn bis zum Ablauf des Timers keine Bestätigungen eingehen, könnte A davon ausgehen, dass die Nachricht verloren gegangen ist und diese Nachricht erneut senden.

0,5 Punkte

In MTP werden aber genau keine ACKs verschickt, sondern NAKs.

Aufgabe 3) In einem fiktiven Anwendungsprotokoll wird eine Folge von ASCII-Schriftzeichen (Bit8=0) als Paketstrom mit jeweils vier Byte Nutzinformation pro Paket übertragen. Nach vier solchen Paketen folgt ein weiteres mit Redundanz-Information (der bitweise Exklusiv- Oder-Wert der Nutzpakete). Der Empfänger erhält die folgende Paketfolge (die Bitfolgen sind — entgegen der typischen Übertragungsreihenfolge — mit MSB-first angegeben worden). a) Was hat der Sender wohl ursprünglich mitteilen wollen? Kurze Begründung.

Paket2	01111001	00100000	01101101	01100001
Paket3	01100111	00100000	01010000	01101001
Paket4	01111010	01111010	01100001	00101110
Red.Paket	00101100	00011011	00101110	01010100

c) Unter welchen Bedingungen ist das verwendete Redundanzverfahren einsetzbar?

Hier wird davon ausgegangen, dass es sich um Forward Error Correction (FEC) handelt. Bei diesem Verfahren werden zusätzliche Redundanzinformationen übertragen, um Fehler, die während der Übertragung auftreten, zu korrigieren, ohne dass eine separate Rückfrage an den Absender erforderlich ist. Die FEC greift auf mathematische Operationen wie die bitweise Exklusiv-Oder (XOR) zurück, um Redundanzinformationen zu erzeugen.

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

In diesem Beispiel stellen wir zunächst fest, dass insgesamt 5 Pakete verschickt wurden, wobei das letzte Paket ein Redundanzpaket ist. Außerdem ist das erste Paket verloren gegangen. Um den Inhalt des ersten Pakets zu bestimmen, müssen wir die empfangenen Pakete übereinanderstapeln und eine spaltenweise XOR-Operation durchführen. Wenn die Anzahl der 1en in einer Spalte gerade ist, ist das Ergebnis 0, sonst 1.

Paket	Daten
Paket 1	0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 1 1 0 0 1 0
Paket 2	0 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0 1
Paket 3	0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1
Paket 4	0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 0
Red.Paket	0 0 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0

Jetzt können wir die einzelnen Bytes eines Nutzdatenpakets mithilfe einer ASCII-Tabelle in Zeichen umwandeln:

Paket 1 = 01001000 (‘H’) 01100001 (‘a’) 01110010 (‘r’) 01110010 (‘r’)

Paket 2 = 01111001 (‘y’) 00100000 (‘ ‘) 01101101 (‘m’) 01100001 (‘a’)

Paket 3 = 01100111 (‘g’) 00100000 (‘ ’) 01010000 (‘P’) 01101001 (‘i’)

Paket 4 = 00101100 (‘z’) 00011011 (‘z’) 00101110 (‘a’) 01010100 (‘.’)

Wenn wir nun die Zeichen aneinanderhängen, erhalten wir die ursprüngliche Nachricht:

Harry mag Pizza.

3b) Beim Forward Error Correction (FEC) schickt man seine Informationen in einer Folge von n Paketen, wobei nur k Pakete die Daten enthalten. n-k Pakete enthalten also Redundanz Informationen. Das FEC Verfahren muss so aufgebaut sein, dass alle Daten rekonstruierbar sind, sobald aus der Menge von n Paketen k Pakete angekommen sind.
Bei einer einfachen FEC Nutzung wird beispielsweise jeder Gruppe von k Paketen ein weiteres als Redundanz Wert gegeben, also gilt für n = k+1

Hier ist eine wichtige Bedingung, dass das Redundanzpaket korrekt mit bitweise XOR gebildet wird. Weiterhin gilt, dass man immer nur ein Nutzdatenpaket mit einem Redundanzpaket rekonstruieren kann, und alle dabei xor-verknüpften Nutzdatenpakete müssen den Empfänger

Cihasev Adam
Erdogan Ibrahim
Jaber Muhamed

einwandfrei erreicht haben. Also pro Gruppe von k Paketen kann es hier nur ein Redundanz-paket geben und somit darf nur ein Paket verloren gehen.

Deshalb kann es sinnvoll sein, k möglichst klein zu wählen, um damit einen guten Schutz vor Paketverlusten zu realisieren.

In dem einfachen Fall dürfen nicht mehrere Pakete hintereinander verloren gehen. Ansonsten muss man das FEC-Verfahren auf einer anderen Basis aufbauen. Wenn mehrere Redundanz-pakete vorgesehen werden, also mehrere Paketverluste erwartet werden, dann kann man das FEC-Verfahren auf den gleichen Grundprinzipien aufbauen, wie bei einem Gleichungssystem. Dabei haben wir so viele unabhängige Gleichungen mit der gleichen Lösung wie unbekannte Variablen. Bei einem Paketverlust kann der Sender einfach eine weitere Gleichung mit gleicher Lösung an den Empfänger mitteilen

So kann die Anzahl der Redundanz Pakete (n-k) in Abhängigkeit von er Verlustrate gewählt werden.

2 Punkte