Real-Time Transport Protocol (RTP)

Real-Time Transport Protocol, vgl. RFC3550, RFC3551

Das Designziel von RTP ist die Unterstützung von Multimedia Konferenzen mit mehreren (vielen) Teilnehmern.

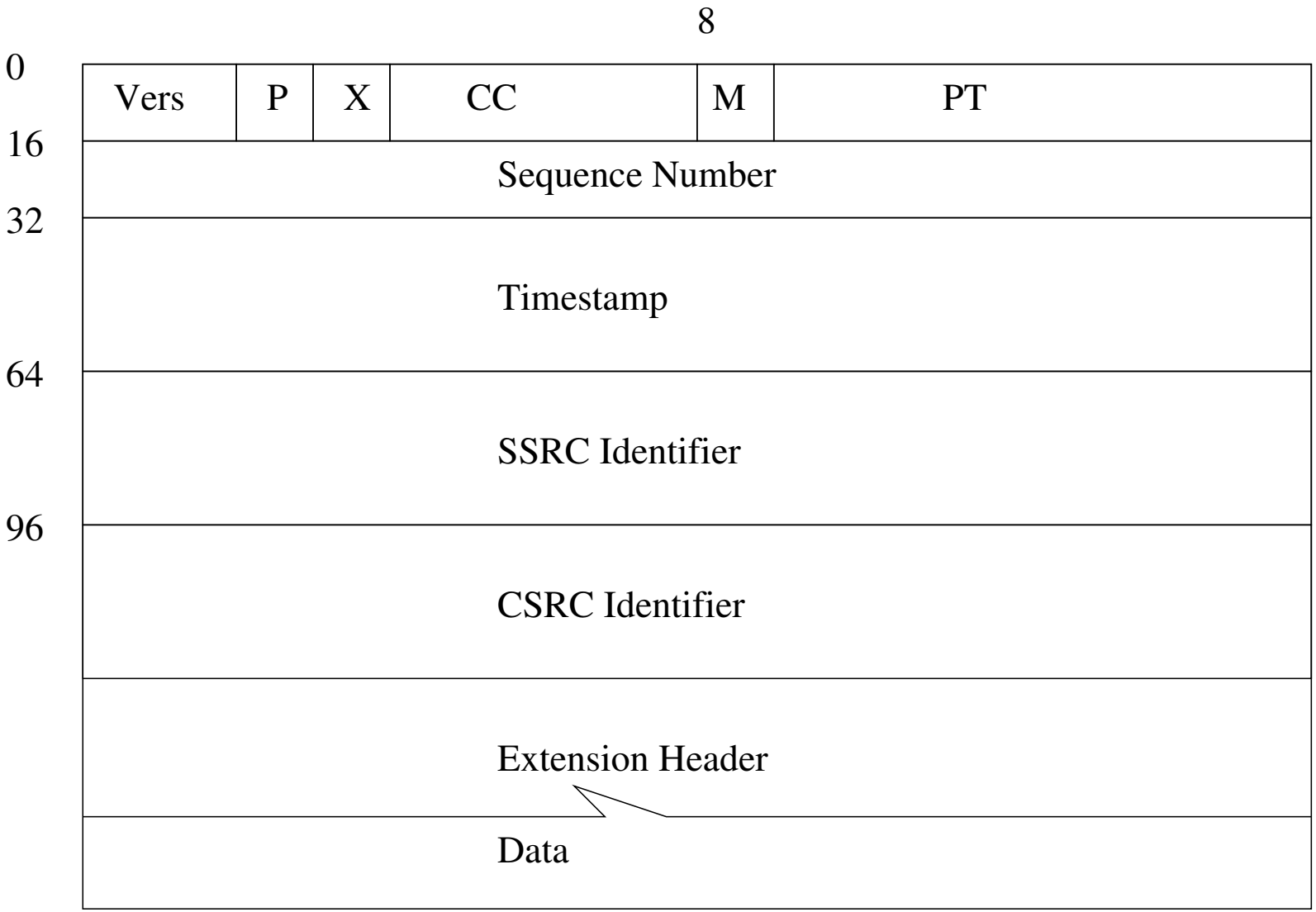
RTP ist das Transfer Protokoll zur Übertragung von Daten mit Zeitanforderungen.

- kurze Rahmenheader erlauben kleine Pakete
- Zeitstempel im Paket zur Berechnung von Laufzeitschwankungen
- unterstützt Aufteilen und Zusammenmischen von Datenströmen
- unterstützt beliebige Medientypen

Transport

- RTP benutzt (üblicherweise) UDP als Transport, dadurch:
  - keine automatischen Wiederholungen
  - keine garantierte Auslieferung
  - unterstützt Multicast
  - keine QoS Garantien
- Port Nummern sind nicht spezifiziert, aber üblicherweise gerade “Ephemeral Ports”
- Die folgende ungerade Portnummer wird üblicherweise für den zugehörigen RTCP Strom benutzt.
- Portnummern zwischen verschiedenen Stationen werden über andere Protokolle (z.B.: SIP, H.323) ausgetauscht.

RTP Rahmen


8
0
Vers
P
X
CC
M
PT
16
Sequence Number
32
Timestamp
64
SSRC Identifier
96
CSRC Identifier
Extension Header
Data

Felder im Rahmen

- **Vers.**: Version, aktuell 2
- **P**: Padding, 1, wenn das Paket Füllbytes enthält, das letzte Byte enthält dann deren Anzahl
- **X**: Protokollerweiterungen werden benutzt, dann ist genau ein Erweiterungsheader enthalten
- **CC**: Anzahl CSRCs
- **M**: Marker, dessen Bedeutung vom Profil abhängt
- **PT**: (Payload Type) Art der Daten im Rahmen
- **Sequence Number**: Zahl, die pro Paket um 1 erhöht wird. Der Startwert ist zufällig.

Felder im Rahmen

- **Timestamp**: Zeitstempel der Erzeugung des ersten Bytes der Daten, der Startwert ist zufällig, die Auflösung muß zur Synchronisation ausreichen.
- **SSRC**: (Synchronization Source) Eindeutige Kennung einer RTP Datenquelle
- **CSRC**: (Contributing Source) Eindeutige Kennung einer RTP Datenquelle, wenn mehrere Quellen zu einem Strom gemischt wurden.

Erweiterungsheader

Ist das Feld **X** im RTP Rahmen 1, wird genau ein Erweiterungsheader verwendet. Das Format ist

- 16 Bit private Daten (abhängig vom Profil)
- 16 Bit Länge (kann 0 sein) in 4 Byte Einheiten
- Daten

Sinn des Erweiterungsheaders ist, daß neue RTP Implementationen profilabhängige Zusatzfunktionen implementieren können, ohne daß bestehende Implementationen beeinträchtigt werden (die Payload bleibt gleich).

Profile

Allein die Informationen im RTP Rahmen reichen nicht aus, um die Daten sinnvoll zu nutzen.

- Die Aufteilung und Interpretation eines RTP Rahmens findet auf Anwendungsebene statt.
- Die Protokollspezifikation muß neben den RTP Feldern die unterstützten Datenformate und deren Interpretation festlegen (Profile)
- Profile für Audio und Video Konferenzen finden sich z.B. in RFC3551.

Application Level Framing

Die Aufteilung der Daten in Pakete/Rahmen geschieht bei RTP nicht in der Transportschicht oder Sicherungsschicht, sondern durch die Anwendung. Gründe ergeben sich aus der möglichen Interaktion mit dem Zeitverhalten der unteren Schichten:

- Zusammenfassen von Daten zu einem Paket erhöht die Latenz.
- Automatische Fehlerkorrektur (z.B. durch ARQ) verzögert ggfs. den ganzen Datenstrom.

Daher muß ein verbindungsloser Transport gewählt werden. Das heißt aber, daß die Anwendungsschicht für Fehlerbehandlung zuständig ist:

- Paketverlust
- Mehrfachübertragung
- Veränderung der Reihenfolge

RTP Payload (vgl. RFC2736)

Der Datenstrom wird festgelegt durch die verwendete Quellkodierung (z.B. iLBC, RFC3951).

- Die Aufteilung auf einzelne RTP Pakete sollte sicherstellen, daß der Datenstrom auch bei moderatem Paketverlust noch verwendbar ist.
- Idealerweise sollten einzelne Pakete unabhängig von ihren Vorgängern dekodierbar sein.

Hieraus ergeben sich Anforderungen an die Rahmenbildung:

- Die Payload eines Paketes sollte vollständige Rahmen des Codecs enthalten.
- Minimale Datengrößen des Codecs sollten bei der Payload nicht unterschritten werden, es sei denn, sie überschreiten die MTU.
- Fragmentierung sollte auf Anwendungsebene unter Berücksichtigung des Codecs implementiert sein.

Beispiel: MPEG-1 Video (ISO/IEC 11172-2)

MPEG-1 (Group Of Pictures) benutzt 3 verschiedene Rahmentypen:

- **I-Frame**: Ein Bild, das unabhängig vom Rest (ähnlich JPEG) kodiert worden ist.
- **P-Frame**: Enthält die Unterschiede zum vorausgegangenen I-Frame oder P-Frame.
- **B-Frame**: Enthält Korrekturdaten zur Interpolation zwischen vorausgegangenem und folgenden I-Frame oder P-Frame.

Offensichtlich kann der Codec bei jedem I-Frame erneut starten, allerdings überschreitet deren Größe üblicherweise die angenommene UDP MTU (512 Byte). MPEG-1 unterteilt Bilder allerdings in 16×16 Pixel Makroblocks, die unabhängig voneinander behandelt werden können. Für Details vgl. RFC2250.

Identifikation der Payload

Das Feld **PT** Payload Type enthält eine 16Bit Zahl zur Festlegung des Codecs. Diese Zahl muß von beteiligten Anwendungen (Sender / Empfänger / Mixer / Translator) interpretiert werden können.

- Eine Liste standardisierter Quellkodierungen und deren **PT** findet sich unter `http://www.iana.org/assignments/rtp-parameters`
- RFC3551 gibt die Standardprofile für Audio- und Videoübertragung an. Ferner wird beschrieben, wie **PT** dynamisch und anwendungsspezifisch vergeben werden kann.
- RFC3555 beschreibt für die Payload Typen in RFC3551 die Werte für den (z.B. HTTP/SDP) Content-Type Header.

Umwandlung der Daten: Mixer

In Konferenzen ist es nicht immer möglich/sinnvoll, daß alle Teilnehmer Daten in identischer Kodierung bekommen. Andererseits ist es bei großer Teilnehmerzahl nicht möglich, daß der Sender abhängig vom Empfänger den Datenstrom gesondert kodiert und überträgt.

Ist z.B. ein Teil einer Gruppe über eine Verbindung zu geringer Bandbreite angebunden, kann ein **Mixer** die Datenströme der Sender für diese Teilnehmer bündeln, mit niedriger Datenrate kodieren und dann weiterleiten.

Inzwischen werden auch hierarchische Verfahren verwendet. Dabei werden die Daten in aufeinander aufbauenden Schichten (Layern) kodiert. Höhere Schichten können bei der Übertragung bei Bedarf weggelassen werden, was natürlich einen Qualitätsverlust zur Folge hat.

Umwandlung der Daten: Translator

Ist ein Teil der möglichen Empfänger nicht über den üblichen RTP Transport (z.B. UDP Multicast) erreichbar, kann ein **Translator** den Datenstrom für diese Verbindung in ein geeignetes Format umwandeln und an ein Gegenstück auf der anderen Seite der Verbindung weiterleiten.

Einleitung

RTCP (RTP Control Protocol) ist das Protokoll, mit dem die Teilnehmer einer RTP Session Statusinformationen austauschen. Aufgaben sind:

- Rückmeldung zur Übertragungsgüte
  Auf Basis dieser Informationen kann ein beteiligter Knoten z.B. die Quellkodierung anpassen.
- Identifikation der Teilnehmer über den “Canonical Name” (CNAME), da sich die SSRC während einer Session ändern können (z.B. zur Konfliktauflösung). Ferner dient der CNAME zur Korrelation verschiedener RTP Ströme (z.B. Audio / Video).
- Anpassung der eigenen Senderate von RTCP Paketen
- Trivialversion einer Sessionkontrolle, die zumindest die Liste der Teilnehmeridentifikationen anbietet.

RTCP Paket Typen

RTCP definiert die folgenden Pakettypen:

- **SR** (Sender Report) Sende- und Empfangsstatistiken von aktiven Sendern der Session
- **RR** (Receiver Report) Empfangsstatistiken von Teilnehmern, die aktuell nicht senden.
- **SDES** (Source Description) Informationen über die Quelle, vor allem CNAME
- **BYE** Teilnehmer verlässt sie Konferenz.
- **APP** Anwendungsspezifische Daten

Transport

- RTCP benutzt wie RTP den verbindungslosen Dienst der Transportschicht, üblicherweise UDP, wobei die Portnummern auf die Portnummern des RTP Stroms folgen.
- Mehrere RTCP Pakete können in einem Paket der Transportschicht übertragen werden.
- Die Reihenfolge der RTCP Pakete in einem zusammengesetzten Paket ist festgelegt.
- Jedes zusammengesetzte Paket muß eine Empfangsstatistik (SR oder RR) enthalten.
- Jeder zusammengesetzte Paket muß den CNAME enthalten, falls möglich.
- Pro Sendeintervall soll nur ein zusammengesetztes Paket pro Teilnehmer gesendet werden.

Zusammengesetzte RTCP Pakete

Jedes RTCP Paket hat den folgenden Aufbau:

- **Encryption Prefix**: 4 Byte Zufallszahl, falls Verschlüsselung benutzt wird.
- **SR/RR**: gegebenenfalls leere Sende- oder Empfangsstatistik
- **RR**: Folge weiterer Empfangsstatistiken bei großen Konferenzen
- **SDES**: mindestens der CNAME des Teilnehmers, möglicherweise weitere anwendungsspezifische Informationen
- **BYE/APP**: weitere Informationen in beliebiger Reihenfolge

RTCP Sendeintervall

Die Frequenz, mit der RTCP Pakete gesendet werden, muß der Anzahl Teilnehmer einer Konferenz angepasst werden. Wird z.B. für einen Link eine Datenrate reserviert, würde diese sonst bei steigender Teilnehmerzahl nicht mehr ausreichen.

- Die für eine Session (RTP + RTCP) genutzte Bandbreite wird im voraus festgelegt.
- Bandbreitenberechnung schließt Transport- und Netzwerkschicht (z.B. UDP über IP) ein.
- Die Bandbreite für RTCP soll einen festen, bekannten Prozentsatz der Gesamtbandbreite ausmachen (z.B. 5%, RFC3550).
- Aufteilung der Bandbreite auf RTP und RTCP, sowie in RTCP auf Sender und Empfänger wird im Profil festgelegt.

RFC3550 RTCP Interval Berechnung

RFC3550 schlägt einen Algorithmus mit folgenden Eigenschaften vor:

- Das berechnete Intervall wächst proportional zur Anzahl Teilnehmer.
- Die Zeit zwischen der Erzeugung zweier RTCP Pakete liegt zufallsgesteuert zwischen dem 0.5- und 1.5-fachen des berechneten Intervalls.
- Die (Netzwerkschicht-)Paketgröße wird permanent mitgeschätzt.
- Kommen (fast) gleichzeitig viele neue Teilnehmer hinzu, werden deren RTCP Pakete gesondert verzögert.
- Wird ein BYE empfangen, werden die berechneten Intervalle schnell verkleinert.
- RTCP Pakete, die ein BYE enthalten, können vorzeitig gesendet werden.

Sender Report

Header	V	P	RC	PT	Length
	SSRC of Sender				
Sender Info	NTP Timestamp				
	RTP Timestamp				
	Sender Packet Count				
	Sender Byte Count				
Report Block	SSRC of Source 1				
	Fraction Lost	Number of Lost Packets			
	Highest Sequence Number				
	Interarrival Jitter				
	Last SR				
	Delay since Last SR				
	Next Report Block				
	Profile Specific Extension				

Felder im SR

- **V**: Version=2
- **P**: Padding, wie bei RTP
- **RC**: Anzahl Report Blocks im Rahmen
- **PT**: 200 für RTCP SR
- **Length**: Länge des Rahmens in 4Byte Einheiten gezählt ab Sender Info
- **SSRC**: Identifikation des Senders des RTCP Paketes

SR Sender Info

- **NTP Timestamp**: Uhrzeit der Erzeugung des SR im Format des Network Time Protocols (RFC1305).
- **RTP Timestamp**: Gleiche Zeit wie NTP Timestamp, aber in Skalierung und Offset des zugehörigen RTP Stroms.
- **Sender Packet Count**: Anzahl gesendeter Pakete seit Start der Session unter der aktuellen SSRC.
- **Sender Byte Count**: Anzahl gesendeter Bytes der Payload, d.h. keine RTP Header oder Padding

SR Report Blocks

- **SSRC**: Identifikation des Senders, zu dem die Statistik gehört
- **Fraction Lost**: Verhältnis von verlorenen Paketen zu erwarteten Paketen multipliziert mit 256
- **Number of Lost Packets**: Insgesamt verlorene Pakete von dieser SSRC im Verlauf der Session
- **Highest Sequence Number**: Höchste empfangene Sequenznummer von der SSRC, erweitert durch 2 Byte für Überlauf.
- **Interarrival Jitter**: Varianz der Zwischenankunftszeiten
- **Last SR**: 0, falls noch kein SR empfangen wurde, sonst die mittleren 4 Byte des NTP Zeitstempel des letzten SR
- **Delay since Last SR**: Anzahl 1/65536-tel Sekunden seit Empfang des letzten SR; 0, falls undefiniert.

SR/RR Interarrival Jitter

Der Jitter muß kontinuierlich geschätzt werden. Der Algorithmus dazu ist festgeschrieben, um unabhängig vom Profil des RTP Stroms interpretiert werden zu können.

Sei S_i der RTP-Zeitstempel des i-ten RTP Paketes, R_i die Ankunftszeit in derselben Einheit. Setze

$$D(i,j) := (R_j - S_j) - (R_i - S_i)$$

$$J(i) := J(i-1) + \frac{|D(i-1,i)| - J(i-1)}{16}$$

Gesendet wird jeweils $J(i)$ für das aktuelle Paket i.

Receiver Report

Der Receiver Report hat zwei Funktionen,

1. als Erweiterung des SR, falls die 5 Bit des RC für mehr als 31 Empfänger nicht ausreicht.
2. als Benachrichtigung eines reinen Empfängers.

Das Datenformat ähnlich dem SR:

- Wert des **PT** ist 201
- Es gibt keinen Sender Info Block.

Source Description (SDES)

32

Header	V	P	SC	PT	Length
Chunk 1	SSRC/CSRC_1				
	SDES Items				
Chunk 2	SSRC/CSRC_2				
	SDES Items				

- **SC**: Source Count, d.h. Anzahl Chunks
- **PT**: SDES = 202

SDES Chunks

Chunks haben immer die TLV Kodierung:

- **Type**: 1 Byte, Typ der folgenden Daten
- **Length**: 1 Byte, Länge der Daten
- **Data**: Nutzdaten in UTF-8, auf 4 Byte Grenze aufgefüllt

Die folgenden SDES Typen sind in RFC3550 definiert:

1	CNAME	2	NAME
3	EMAIL	4	PHONE
5	LOC	6	TOOL
7	NOTE	8	PRIV

SDES CNAME

Der CNAME unterliegt besonderen Anforderungen, da er unabhängig vom Profil für die Funktion des Protokolls notwendig ist:

- Da die SSRC Kennung zu Konfliktauflösung und bei Programmneustarts verändert werden kann, dient der CNAME zur Identifikation des Senders.
- Über den CNAME können unterschiedliche Medienströme in verschiedenen RTP Sessions synchronisiert werden.
- Der CNAME dient zur Identifikation der Quelle durch Prozesse aber auch durch Menschen z.B. für Überwachungsaufgaben.

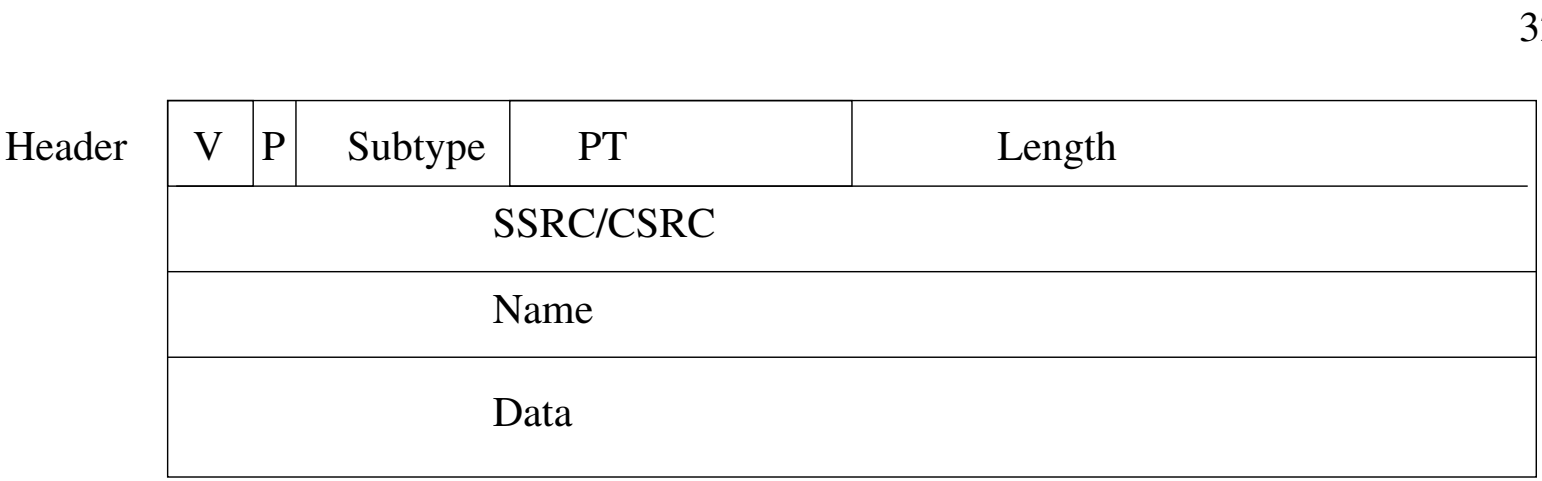
Daher muß der CNAME für den Verlauf der Session eindeutig und konstant sein. RFC3550 schlägt `user@host` vor.

RTCP Goodbye (BYE)

RTCP Paket BYE hat denselben Header wie SDES, darauf folgt die Liste der SSRC/CSRC, die die Gruppe verlassen.

- PT ist 203 für BYE.
- SC gibt die Anzahl SSRC/CSRC im Paket an.
- Optional kann am Ende des Paketes ein Grund für das Verlassen der Gruppe angegeben werden.

Application-Defined (APP)



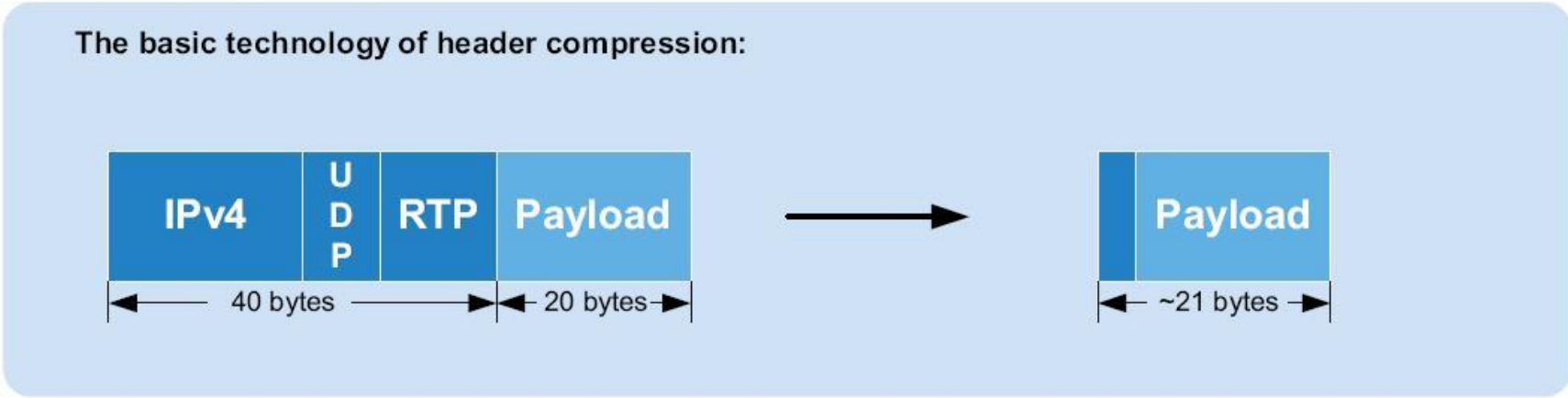
- **Subtype**: Feld zur Unterscheidung verschiedener APP Pakete
- **PT**: APP = 204
- **Name**: 4 Byte ASCII Text
- **Data**: Daten, je nach Profil

Robust Header Compression (ROHC)

Robust Header Compression (RFC 3095)

Bei der Übertragung vieler kleiner Datenpakete z.B. in einem Stream kann ein großer Teil der tatsächlich übertragenen Daten aus identischer Header Information bestehen.

Idee der Header Kompression ist es diese identische Header Information so weit wie möglich einzusparen, d.h., z.B. nach Übertragung der initialen Werte nur die Änderungen zu übertragen.



(c) Effnet

ROHC: Anwendungsbeispiele

Mögliche Anwendungsgebiete sind z.B.:

- Voice over IP (VoIP)
- Audio/Video Streaming
- interaktive Spiele
- ...

Ein Einschränkung besteht darin, das auch nach der Kompression die Verbindung eindeutig definiert bleibt, z.B. durch den Header einer tieferen Schicht oder durch physikalische Gegebenheiten.

Häufige Anwendung wenn IP Pakete über nicht original IP (Teil-)Netze übertragen werden, z.B. analoges Modem oder Mobilfunknetz.

ROHC: Einsparpotential

The header compression gains:

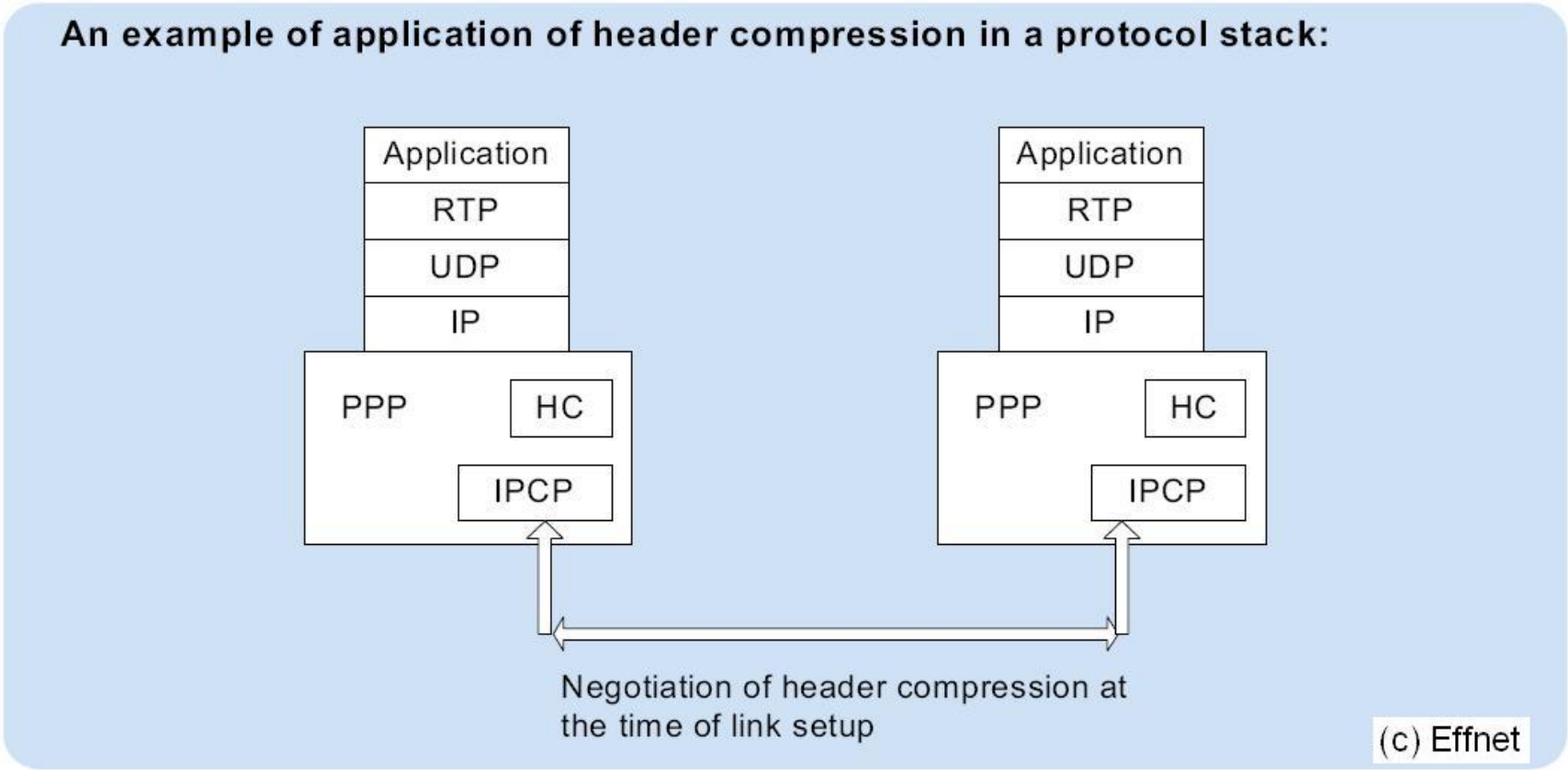
Protocol headers	Total header size (bytes)	Min. compressed header size (bytes)	Compression gain (%)
IP4/TCP	40	4	90
IP4/UDP	28	1	96.4
IP4/UDP/RTP	40	1	97.5
IP6/TCP	60	4	93.3
IP6/UDP	48	3	93.75
IP6/UDP/RTP	60	3	95

(c) Effnet

In typischen Anwendungen können mit ROHC mehr als 90% der Header Information eingespart werden.

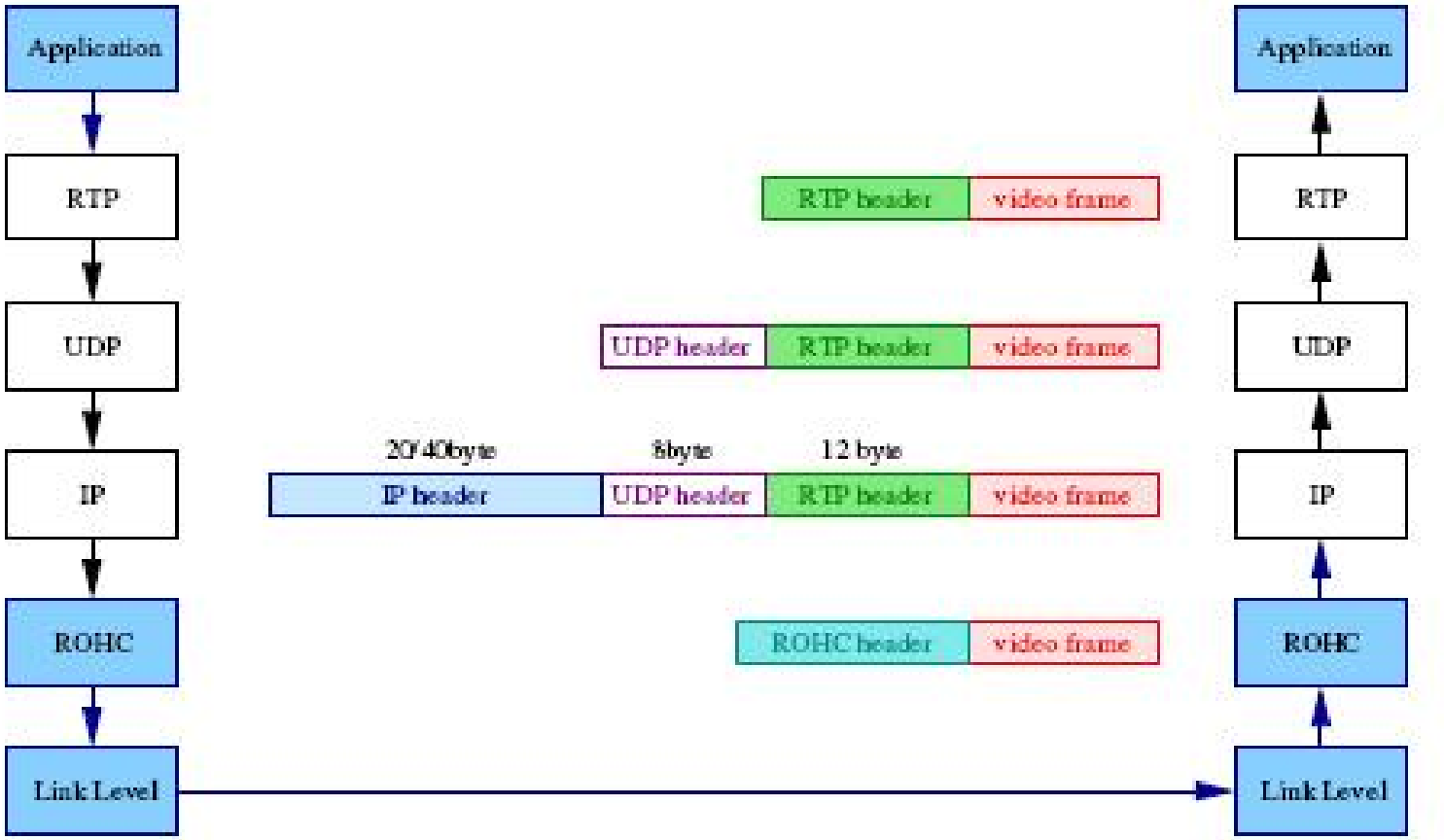
“Robust” bedeutet mit (mindestens) der gleichen Qualität wie ohne Header Kompression.

ROHC Beispiel: Protokollstapel für PPP



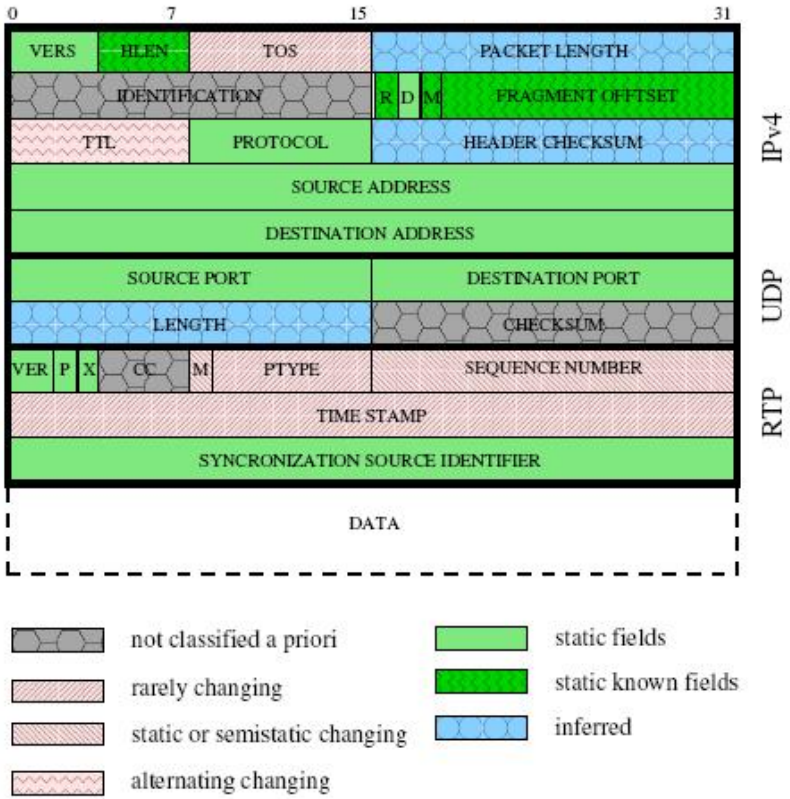
HC: Header (De-)Compression, IPCP: IP control protocol

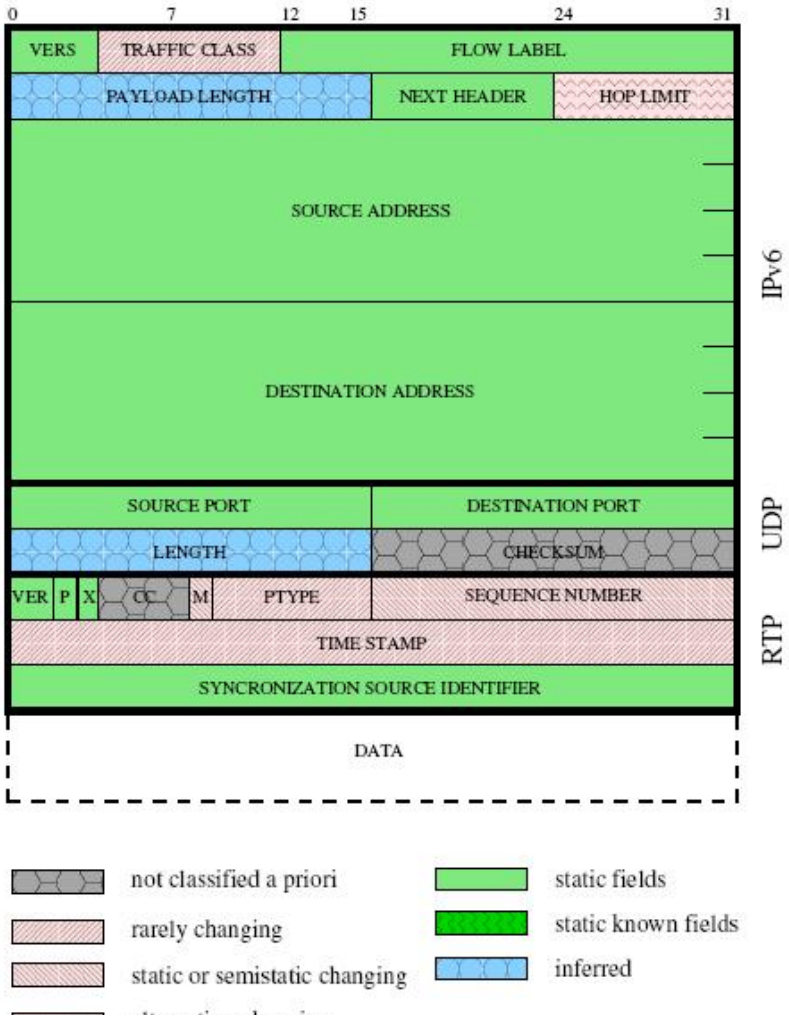
ROHC Beispiel: Aufbau der Header



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

ROHC Beispiel: Struktur der unkomprimierten Header

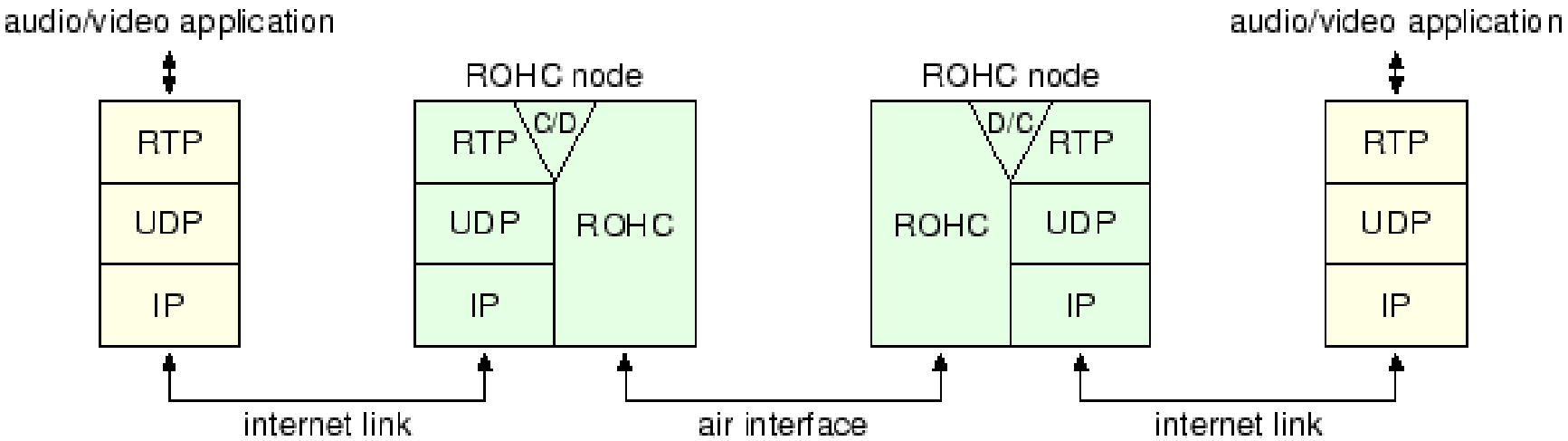




from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

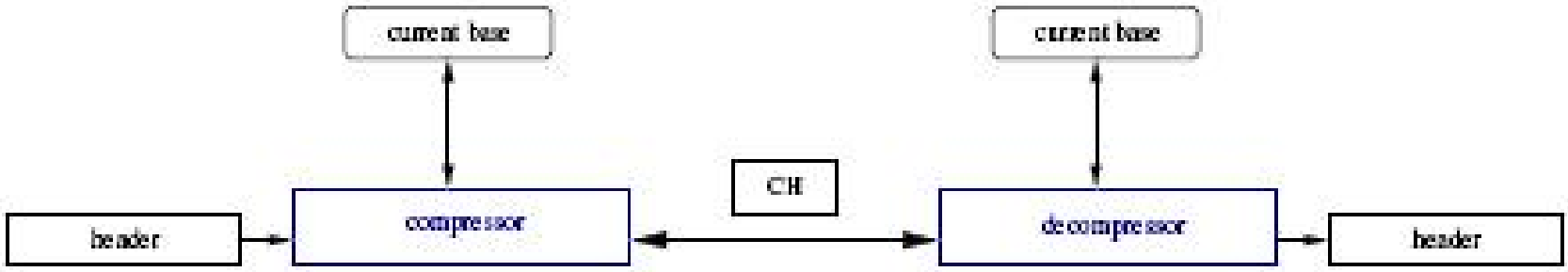
ROHC Beispiel 2: Funkschnittstelle

Endpunkte verarbeiten/benutzen eine IP Schnittstelle. Dazwischen befindet sich eine bandbreitenbegrenzte Funkverbindung. Hier wird für den Endnutzer transparent der Header komprimiert um Bandbreite zu sparen.



ROHC Konzept

Sender und Empfänger speichern und aktualisieren beide die komplette Header Information für die Verbindung. Diese Header Information, auch Kontext genannt, dient als Basis.



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

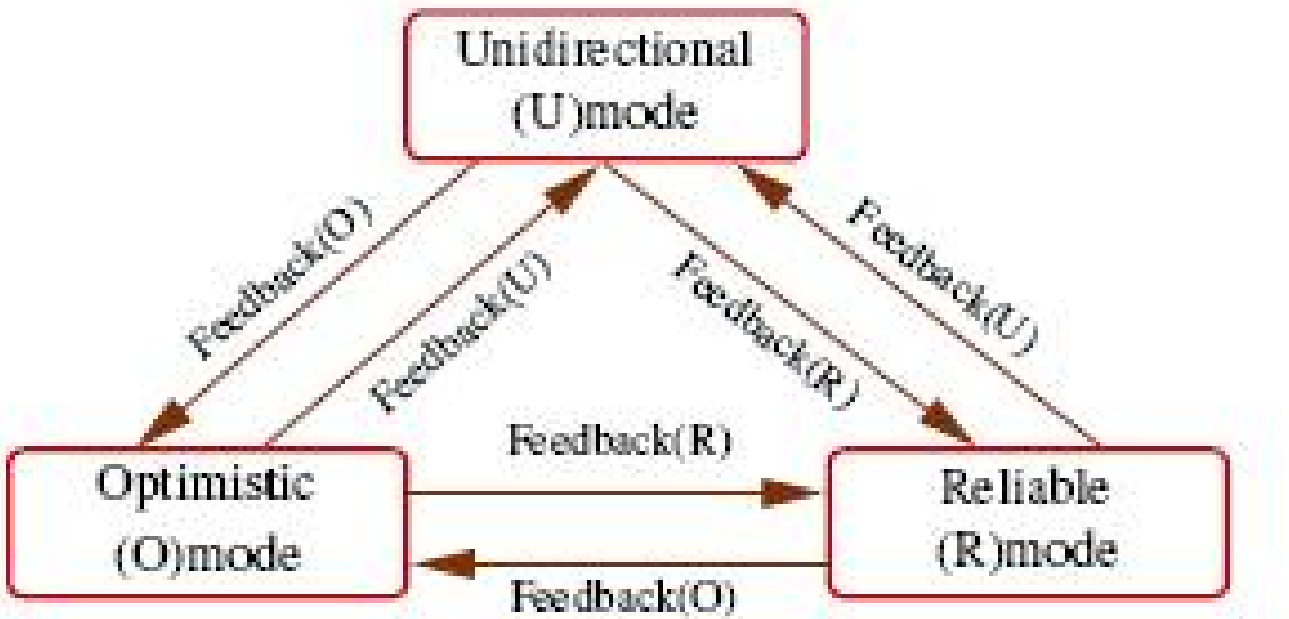
Relevante Änderungen im Header des Senders werden an den Empfänger übertragen, damit dieser seinen Kontext aktualisieren kann. Vorhersehbare Änderungen wie z.B. ein Inkrement der Sequenznummer werden nicht übertragen.

Modi der ROHC

- **Unidirectional Mode (U-Mode)**
  Pakete werden nur in eine Richtung verschickt, kein Feedback
- **Bidirectional Optimistic Mode (O-Mode)**
  Feedback Kanal vorhanden, der aber nur für Fehlermeldungen und Acknowledgements für wichtige Änderungen des Kontexts benutzt wird
- **Bidirectional Reliable Mode (R-Mode)**
  Intensive Nutzung des Feedback Kanals für höchste Qualitätsanforderungen (Ziel: Kein Synchronisationsverlust, niedrige Restfehlerrate)

Modi der ROHC

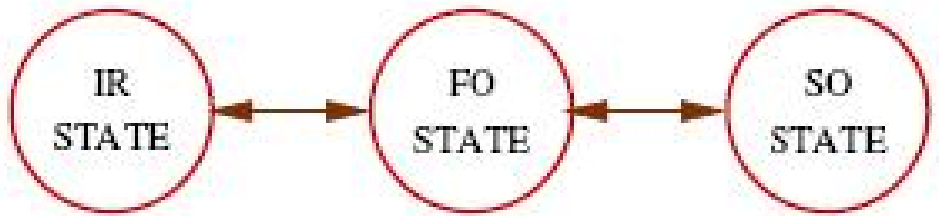
Wechsel der Modi geschieht durch einen Request vom Decompressor im Empfänger auf dem Feedback Kanal



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

Zustände des Compressors

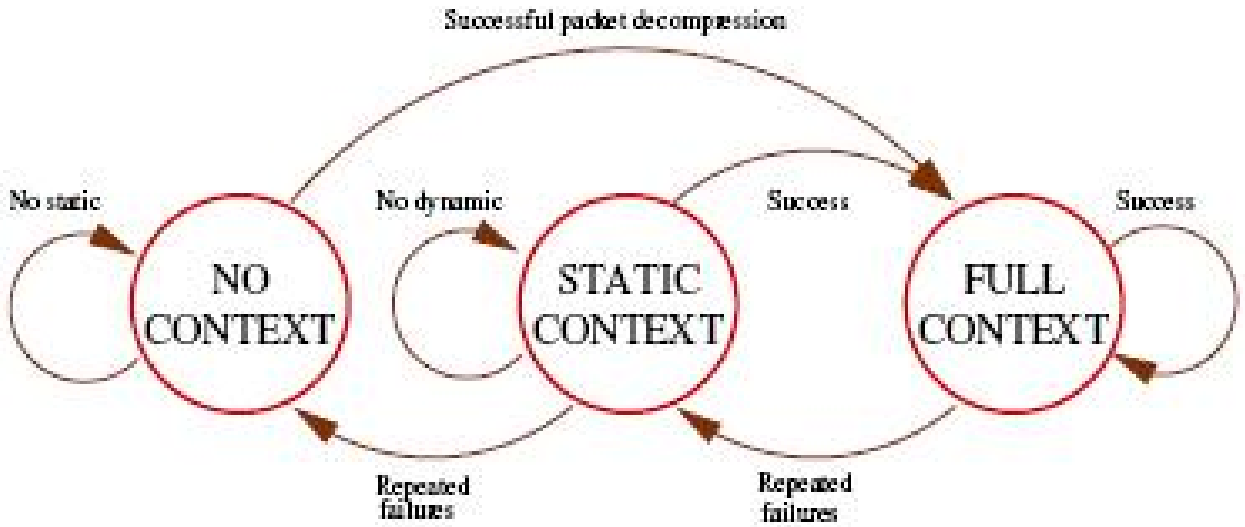
- **Initialization and Refresh (IR):** Startzustand oder nach Reset, gesamter Header wird gesendet.
- **First-Order (FO):** Compressor und Decompressor haben beide die statischen Headerelemente gespeichert, dynamische Headerelemente werden verschickt.
- **Second-Order (SO):** Auch dynamische Headerelemente werden nicht mehr gesendet, sondern nur noch eine Sequenznummer und eine Checksum (1-2 Byte). Der Decompressor muß die dynamischen Headerelemente berechnen.



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

Zustände des Decompressors

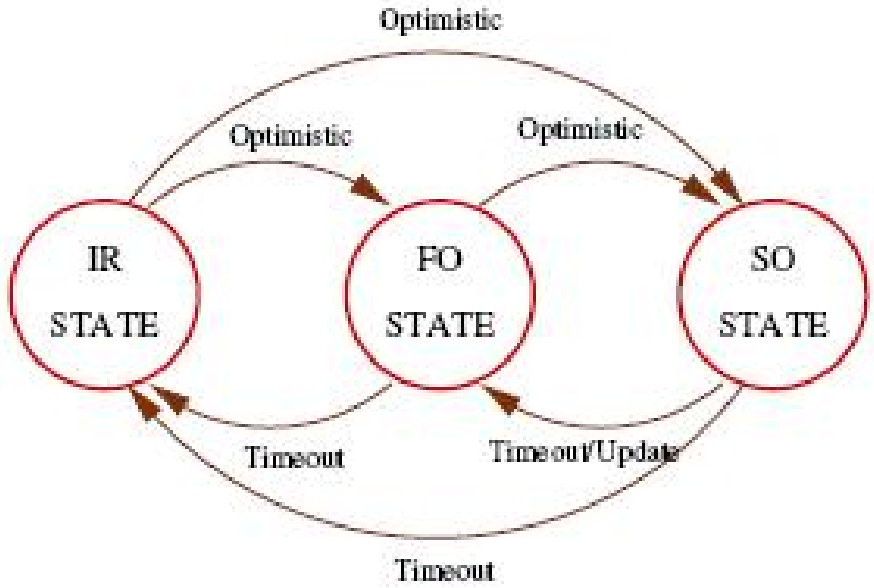
- **No Context:** keine Header Information vorhanden, gesamter Header wird benötigt
- **Static Context:** statische Header Information vorhanden
- **Full Context:** gesamte Header Information vorhanden



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

Zustandsübergänge im Unidirectional Mode

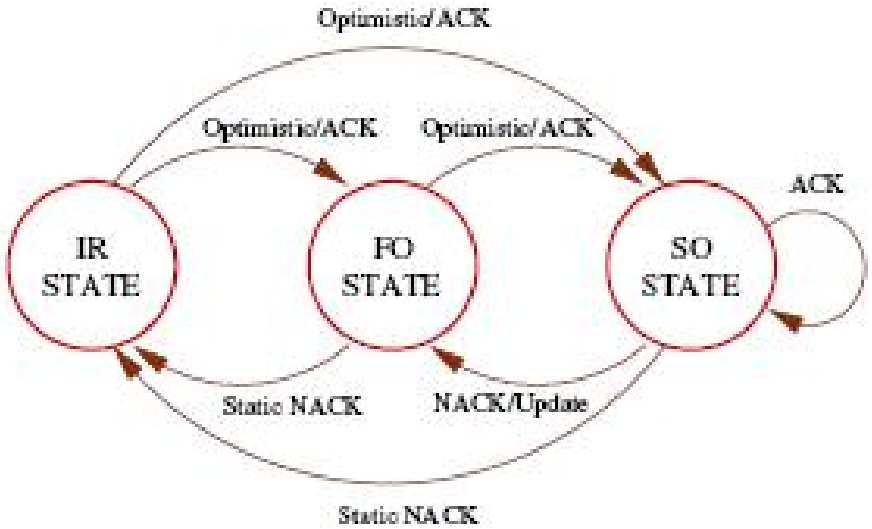
- Rückkehr vom Zustand SO nach IR und FO durch timeouts.
- Zum Schutz vor Fehlern in wichtigen Headerelementen werden in den Zuständen IR und FO mehrere Pakete gesendet bevor in den nächsten Zustand gewechselt wird.



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

Zustandsübergänge im Optimistic Mode

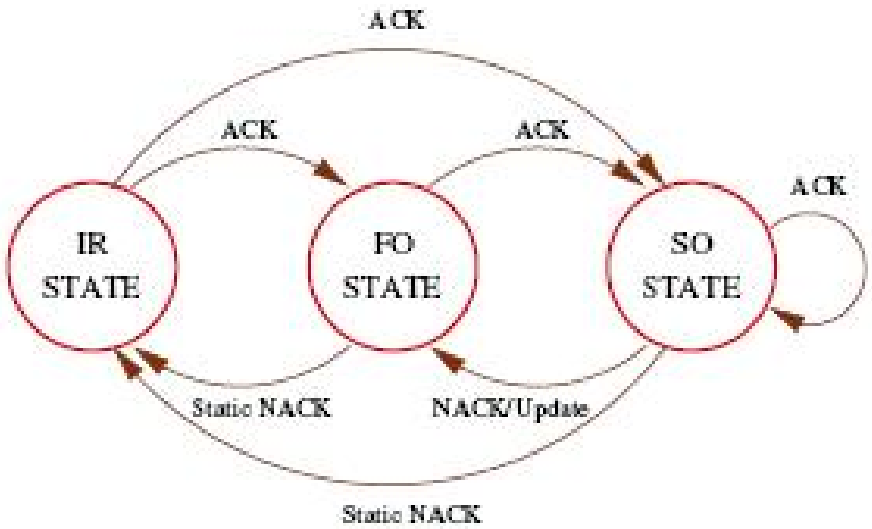
- Decompressor kann ACKs für erfolgreich empfangene Headerelemente versenden
- Decompressor kann mit NACKs Fehler melden.
  - einfaches NACK, neuer Zustand FO
  - static NACK, neuer Zustand IR



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.

Zustandsübergänge im Reliable Mode

- Decompressor sendet ACK für jede erfolgreich empfangene Änderung
- Compressor wechselt nur nach ACK in FO oder SO Zustand
- Durch die ACKs wird sichergestellt, daß Compressor und Decompressor immer im gleichen Zustand sind.



from Header Compression Schemes fo Wireless Internet Access by Fitzek et al.