Technical University of Braunschweig  
Institute of Operating Systems and Computer Networks

Seminar on Communication and Multimedia

TCP-friendly Rate Control (TFRC)

written by  
Sven Jaap

supervised by  
Xiaoyuan Gu

Braunschweig, 23. January 2004
Abstract

This paper is intended to introduce in the *TCP-friendly Rate Control Protocol (TFRC)* and to show some current developments that make use of the protocol. Additionally, it should be looked at further developments of the protocol.

TFRC is a congestion control mechanism designed for unicast flows operating in an Internet environment and competing with TCP traffic. It is designed to be reasonably fair when sharing bandwidth with TCP flows, where a flow is "reasonably fair" if its sending rate is generally within a factor of two of the sending rate of a TCP flow under the same conditions. However, TFRC has a much lower variation of throughput over time compared with TCP, which makes it more suitable for applications such as telephony or streaming media where a relatively smooth sending rate is of importance.

This paper starts with an introduction, which explains the purpose of TCP-friendly rate control mechanisms in the Internet. Afterwards, the functioning of the TFRC protocol is depicted and its behaviour is compared with TCP. Then, some examples of use for TFRC are presented. Furthermore, it is dealt with future research directions of TFRC. Finally, a conclusion is drawn and the paper closes with some outlooks.
## Contents

1 **Introduction** ............................................................................................................................................................................. 1

2 **TFRC protocol specification in brief** ............................................................................................................................................................................. 2

   2.1 Throughput Equation ............................................................................................................................................................................. 3

   2.2 Functionality of the data sender ......................................................................................................................................................... 3

   2.3 Functionality of the data receiver ......................................................................................................................................................... 4

3 **Comparison of TFRC and TCP** ......................................................................................................................................................... 8

4 **Current development and future research directions** ............................................................................................................................................................................................................................................. 9

   4.1 Datagram Congestion Control Protocol (DCCP) ......................................................................................................................................................... 10

   4.2 MPEG-4 Video Transfer with TFRC ......................................................................................................................................................... 10

   4.3 TCP-friendly Multicast Congestion Control (TFMCC) ......................................................................................................................................................... 11

   4.4 Adaptive TCP Friendly Rate Control (ATFRC) ......................................................................................................................................................... 12

5 **Conclusions and Outlook** ......................................................................................................................................................... 13

References ............................................................................................................................................................................................................................................. 14

## List of Figures

1-1: Congestion scenario ............................................................................................................................................................................. 1

2-1: Control flow of the TFRC protocol ......................................................................................................................................................... 2

2-2: Loss intervals and the accompanying weights ......................................................................................................................................................... 5

2-3: Effect of History Discounting on the weights of the loss intervals ......................................................................................................................................................... 7

3-1: TCP throughput while sharing the bandwidth with TFRC flows ......................................................................................................................................................... 8

3-2: Comparison of TCP’s and TFRC’s transmission rate ......................................................................................................................................................... 9

3-3: Responsiveness of TFRC compared with TCP ......................................................................................................................................................... 9

4-1: Example of MPEG-4 video structure ......................................................................................................................................................... 11

4-2: Methods for video rate adjustment ......................................................................................................................................................... 11
1 Introduction

Nowadays, the Internet is one of the most important communications systems in the world. The number of Internet users yearly reaches all-time highs. More and more people are using this medium to gather information or to stay in contact to people all over the world. Most applications in the Internet use the Transmission Control Protocol (TCP). TCP is a reliable, connection-oriented protocol and has mechanisms to recover from congestions. The TCP-receiver sends acknowledgements back to the sender to confirm the data’s receipt. The transmission of unacknowledged data packets is repeated until a confirmation arrives at the sender.

TCP is said to hold the Internet together, especially because of its congestion control mechanisms. Figure 1-1 shows an example of a congestion scenario.

![Figure 1-1: Congestion scenario](image)

The connection between Router A and Router B is not able to take all data packets coming from Server A and B. Thus the buffers at router B start to fill and packets even have to be thrown away if the buffers run over. This situation is called “congestion”. The packets suffer from a high transmission delay. Using reliable data transfer (like TCP does) which guarantees the correct delivery of all data packets, lost packets have to be retransmitted later. That is why the congestion may continue to exist or even gets worse when refraining from the use of any mechanisms that try to prevent or recover from congestion.

When the TCP-sender detects lost data packets, it assumes that there is congestion at any of the network components on the way to the receiver. Thus, it reduces its sending rate very fast in order to adapt to the congestion. If no further losses are detected, TCP tries to increase the sending rate again.

Through the availability of high speed Internet access (e.g. DSL) to end-users, applications like multimedia streaming, online gaming or voice-over-IP get more import. Those applications transmit a constant data stream over long periods in contrast to the short-term connections of most “typical” TCP applications. The information transmitted by those applications is critical of the times. Therefore it is useless to repeat the transmissions of lost packets because their content might by out of date when arriving late at the receiver. In addition, the applications depend on a smooth data rate to transmit their continuous packet stream. TCP is not able to meet these conditions thus the User Datagram Protocol (UDP) is used instead.
UDP is an unreliable protocol, which is not interested in the fact whether a packet reaches its goal or not. Due to the absence of any mechanisms for e.g. control messages or acknowledgements, UDP is well suited for applications with the need of fast data delivery. On the other hand, the main disadvantage of UDP is its congestion-unawareness. When congestion occurs, UDP is not able to detect it and thus does not reduce its sending rate as TCP does. According to [1], this leads to unfairness towards competing protocols that are congestion-aware. Due to the increased demand on UDP-traffic, congestion control mechanisms for unreliable data transmission gets important. As described in [8], the use of congestion control mechanisms would not only have advantages for the network but also for applications (e.g. real-time applications) that suffer from high delays due to filled up buffers. One possible mechanism to provide congestion control to applications that need a smooth sending rate is “TCP-friendly Rate Control” (TFRC). The major purpose of TFRC is to share TCP-flows fairly with other data flows that use an unreliable transmission. Fairly means that it not has to fit perfectly to the behaviour of TCP, but TFRC’s long-term throughput has to be similar to that of TCP. Furthermore, the protocol considers the special demands of streaming applications. The following subsection deals with the protocol’s mode of operation. Afterwards, TFRC is compared with TCP to find out whether it competes fairly with TCP-flows. Finally, new directions of TFRC’s development are presented.

## 2 TFRC protocol specification in brief

The mechanisms of the TFRC protocol are specified in RFC 3448 [4]. TFRC is no real transport protocol but a congestion control mechanism that can be used by a transport protocol.

In [8], the main components of the protocol architecture are described. On the one hand a method is needed to ensure the TCP-friendly behaviour of the protocol. To be able to determine the conditions on the transmission channel, some measurements have to be taken as well at the receiver as at the sender. The values that are measured at the receiver have to be sent to the data sender whose job is to calculate the allowed sending rate. Therefore a reporting mechanism from the receiver to the sender is indispensable. Finally, the sender needs any kind of regulations on how to change the transmission rate to the calculated value. Figure 2-1 shows the general functionality of the TFRC protocol.

The sender transmits data packets with a specific data rate to the receiver. Those data packets may be subject to e.g. packet loss or transmission delay on the network. At the receiver, the loss event rate is calculated immediately and fed back to the sender. By receiving those “receiver reports”, the sender is able to determine the round-trip-time (RTT) between sender and receiver. Both parameters, the estimated loss event rate and the round-trip-time, are used
to calculate a suitable transmission rate according to the actual network conditions. Afterwards, the sender adapts its transmission rate to the calculated value. In the following sections, some aspects of the TFRC protocol are examined more closely.

### 2.1 Throughput Equation

One of the most important goals of TFRC is reaching TCP-friendly behaviour. According to [8] a TCP-compatible flow is defined by using no more bandwidth than a TCP flow under same conditions, in steady state. In most cases, this demand will not be perfectly fulfilled, but even if the transmission rate keeps in a factor of two of TCP’s rate, the algorithm will be reasonably fair. Therefore, the TCP throughput function is suitable in helping to describe a TCP-compatible flow. This equation describes an upper bound to TFRC’s steady-state sending rate and is given by:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO} (3\sqrt{\frac{32}{3}}) p (1 + 32p^2)}$$

(2-1)

The sending rate is a function of the packet size \(s\), the round-trip time \(R\), the TCP retransmit timeout \(t_{RTO}\) and the loss event rate \(p\). The packet size, round-tip-time and the loss event rate have to be measured at the sender and receiver respectively. For simplicity reasons, \(t_{RTO}\) is set to \(4*R\). According to [4], this approximation is reasonable for reaching TCP-friendliness.

### 2.2 Functionality of the data sender

As mentioned above, the sender sends a stream of data packets to the receiver, which processes the packets and takes some measurements. Those results are returned to the sender in a so called “receiver report”. Now, it is the responsibility of the sender to calculate the round-trip-time (RTT). In order to be able to assign the feedback packet to the correct data packet, a mechanism with sequence numbers and timestamps is used. Data packets that are sent to the receiver contain the following additional information:

- A sequence number which identifies the packet’s position within the entire stream.
- A timestamp carrying the time information when the actual packet was sent.
- The present value of the round-trip-time that was determined by the sender.

In turn, the receiver informs the sender of the loss event rate and parameters to calculate the round-trip time. Therefore, the receiver report contains the following parameters:

- The timestamp \(t_{\text{recvdata}}\) of the last data packet that was received before sending the report.
- The delay \(t_{\text{delay}}\) between receiving the last packet and sending the receiver report.
- The receiving rate \(T_{\text{rec}}\) that was determined at the receiver since the last receiver report was sent.
• The actual value of the loss event rate $p$ as an indicator of the congestion level

When receiving a feedback packet, the sender first determines the RTT for the last packet by using equation 2-2.

$$R_{\text{sample}} = t_{\text{now}} - t_{\text{recvdata}} - t_{\text{delay}} \quad (2-2)$$

In order to achieve a smooth changing sending rate and to prevent unstable protocol behaviour, the new average RTT $R$ is calculated by weighted addition of the new round-trip-time sample and the present average RTT.

$$R = q \cdot R + (1 - q)R_{\text{sample}} \quad (2-3)$$

According to [4] the weighting factor $q$ is recommended to be 0.9. Next, the timeout value $t_{\text{RTO}}$ has to be updated. As mentioned above, this value is approximated by:

$$t_{\text{RTO}} = 4 \cdot R \quad (2-4)$$

Afterwards, the sender is able to determine the new sending rate $T$ by the means of equation 2-1. If the current sending rate is lower than $T$, the sender is allowed to increase its sending rate. Otherwise, it has to decrease the sending rate to $T$.

2.3 Functionality of the data receiver

The main tasks of the data receiver are calculating the loss event rate, and providing feedback to the sender.

As mentioned in the previous subsection, the receiver periodically sends feedback reports to the sender, containing the estimated value of the loss event rate and the information that allows the sender to calculate the round-trip-time. Those reports should be sent at least once per RTT, unless the sending rate is lower than one packet per RTT.

According to [2] the estimation of the loss event rate is one of the most critical and important parts of the TFRC-protocol. The difference between the loss rate and the loss event rate is that a loss event may consist of several packet losses during one round-trip-time. This is due to getting a more stable and smooth behaviour of TFRC and is equivalent to TCP which only reduces its congestion window on half during a single round-trip time.

A packet is assumed to be lost if at least three subsequent packets with higher sequence number than the lost packet have arrived. This procedure corresponds to TCP and is robust to the reordering of packets during their way through the network.

[2] describes some demands on the method for calculating the loss event rate:

• Loss events in several successive round-trip times should have a strong effect on the loss event rate.
• The loss event rate should only increase if new loss events occur.
• The number of packets between two loss events is called loss interval. The loss event rate should only decrease if the new loss interval is longer than the average of the previous loss intervals.
TFRC uses the “Average Loss Interval” method to calculate the loss event rate. This method fulfills all requirements made above. It calculates the weighted average loss event rate over the last n loss intervals by giving the n/2 most recent loss intervals equal weights. Figure 2-2 shows the intervals and the accompanying weights used to calculate the loss event rate.

The number of loss intervals $n$ that are used to calculate the loss event rate determine the sensitivity of TFRC towards changes in the intensity of congestion. It should not increase a value much greater than 8. The set of the specific intervals that slip in the calculation is called Loss History. The weights $w_i$ for the intervals of the Loss History can be determined by the following equation 2-5.

$$w_i = \begin{cases} 1 & 1 \leq i \leq \frac{n}{2} \\ 1 - \frac{i - \frac{n}{2}}{\frac{n}{2} + 1} & \frac{n}{2} < i \leq n \end{cases} \quad (2-5)$$

Now, the average loss interval $I_{(1,n)}$ is given by the weighted average of the last n intervals $I_i$ as follows:

$$I_{(1,n)} = \frac{\sum_{i=1}^{n} w_i I_i}{\sum_{i=1}^{n} w_i} \quad (2-6)$$
Equation 2-6 is not dealing with the interval I_0 which contains all packets that arrived at the receiver since the last lost event. This interval is different from all other because it is not yet completed. According to [2] it is important to ignore I_0 until the interval is large enough to increase the average loss interval. This way, TFRC behaves more smoothly in the presence of a stable loss event rate.

To decide whether I_0 should be taken into consideration or not, an additional average loss interval I_{(0,n-1)} has to be calculated as described in equation (2-7).

\[
I_{(0,n-1)} = \frac{\sum_{i=0}^{n-1} w_{i+1} I_i}{\sum_{i=1}^{n} w_i}
\]  

(2-7)

The final value of the average loss interval is determined by:

\[
I_{\text{mean}} = \max(I_{(1,n)}, I_{(0,n-1)})
\]  

(2-8)

Now, the loss event rate p is given as follows:

\[
p = \frac{1}{I_{\text{mean}}}
\]  

(2-9)

If the average loss interval I_{(0,n-1)} is larger than I_{(1,n)}, the last interval I_0 is larger than the average. This means, that the period of time without packet loss is larger than in the average case. Therefore, the interval I_0 is included in the calculation and the loss event rate is reduced. On the other hand, if I_0 is shorter than the average, it is ignored and the loss event rate is kept steady, which contributes to a more smooth behaviour of TFRC.

[2] and [8] explain that the Average Loss Interval method reacts nearly immediately to increased congestion by a higher loss event rate, but is slow in response to a sudden decrease of the packet loss rate. If no packet losses occur for a long period, the last interval I_0 gets very large. Thus, the estimated loss event rate may not reflect the actual situation any more. Therefore, an optional mechanism called “History Discounting” is used to improve TFRC’s reaction to a fast decrease in the level of congestion.

History Discounting offers the possibility to adjust the weights of the intervals when the most recent interval I_0 gets very large. It is only employed if I_0 is more than twice as large as the estimated average loss interval. A Discounting Factor DF_i for each loss interval is used to perform a proportional deweighting of the single intervals. The Discounting Factor can be calculated as shown in [3] and [4]:

\[
DF_i = \begin{cases} 
\max(0.5, \frac{I_{\text{mean}}}{I_0}) & 0 < i < n \\
1 & i = 0 
\end{cases}
\]  

(2-10)

When using History Discounting, the average loss interval is determined by:
\[ I = \frac{\sum_{i=0}^{n-1} DF_i w_{i+1} l_i}{\sum_{i=1}^{n} DF_i w_i} \]  

(2-11)

The Figure 2-3 shows the effect of History Discounting on the weighting of the intervals.

Limiting the Discounting Factor to a minimum value of 0.5 ensures, that never all weights are concentrated on the most recent loss interval. The old intervals keep a certain influence on the calculation of the average loss interval and are not neglected because of the deweighting. All Discounting Factors are updated each time a new packet is received.

When a new packet loss is detected, all loss intervals are shifted down (e.g. \( I_0 \) gets the new interval \( I_1 \)) and the last loss interval is forgotten. Furthermore, the Discounting Factors need to be shifted down corresponding to the loss intervals. The Discounting Factor of the most recent loss interval is set to 1. Even if the packet loss rate increases again and therefore no further History Discounting is performed, the Discounting Factor of a loss interval is never increased again.
3 Comparison of TFRC and TCP

In this section it should be examined, how TFRC behaves in comparison with TCP. One main goal during the development of TFRC was reaching TCP-friendliness while having a smoother sending rate than TCP.

Firstly, it has to be mentioned, that a major difference between TFRC and TCP is the fact that TCP is a complete transport protocol which support features like flow and congestion control as well as connection establishment, while TFRC just takes care of congestion control and is intended to be used by a transport protocol that provides an unreliable data transmission. Moreover, TFRC is not intended to replace the TCP protocol. It is rather an alternative to the use of TCP and enables congestion control to applications which cannot use the Transmission Control Protocol.

Experiments have shown that TFRC is well suited to compete fairly with TCP under same conditions. In the following, some of these results are presented. More detailed results can be found in [3] and [7].

To be fair when competing with TCP, TFRC needs to have nearly the same throughput in steady-state. Therefore, TFRC uses the TCP response function as described in the previous section, to determine its sending rate. Figure 3-1 (taken from [2]) shows that TFRC’s throughput is almost equivalent to the one of TCP. Especially if many data flows compete against each other, fair sharing of the bandwidth is reached.

The figure presents the average throughput of a TCP-flow competing with other TCP- and TFRC-flows. The throughput is presented in a normalized way, where a value of 1 means a fair share of bandwidth between the different flows.

These results indicate, that TFRC performs reasonably well when sharing the bandwidth with TCP flows.

Another important goal is that the transmission rate of TFRC connections changes over time much slower than the rate of TCP connections. Figure 3-2 is also taken from [2] and depicts the throughput of competing TCP and TFRC flows at a bottleneck bandwidth over time.
It is obvious that the transmission rates of the TFRC flows are less fluctuating than the rates of the TCP flows. Thus, TFRC may offer advantages to applications like multimedia streaming.

Finally, an important difference between TFRC and TCP has to be mentioned. While both protocols react to a sudden decrease of the bandwidth in nearly the same time, TFRC needs much longer to recover when more bandwidth gets available. Figure 3-3 describes this fact.

It might take clearly more than 20 seconds until the fair sharing of the bandwidth is reached again. This could be a problem for applications like video telephony because the video quality would be bad over a quite long period. An improvement of this behaviour can be reached by the use of History Discounting as described above.

In summary it may be said that simulations as well as real-world experiments showed that TFRC works well and all in all fulfils the demands made on it.

## 4 Current development and future research directions

TFRC is mainly intended to be used with applications that prefer a nearly constant transmission rate and may be delay sensitive. Therefore they are not able to make use of TCP. As already mentioned earlier, voice-over-IP, video on demand and video telephony are some examples. It is to be expected that the importance of these applications will increase drastically in the future. Thus, new mechanisms are needed to cope with this trend. In the
following, some developments using TFRC are presented. Afterwards two further developments of TFRC are presented in brief.

4.1 Datagram Congestion Control Protocol (DCCP)

The Data Congestion Control Protocol (DCCP) is a new transport protocol which is able to provide a congestion-controlled and unreliable flow of datagrams. A detailed specification of DCCP can be found in [5]. According to [6], some of the main features of DCCP are:

- Unreliable transport of datagrams with acknowledgements
- Mechanisms for reliable connection establishment and termination
- Usage of TCP-like congestion control or TCP-friendly rate control
- Reliable negotiation of different protocol features

DCCP is developed to be used with applications that attach great importance to a fast packet delivery rather than the correct order of the packets. The data of such applications may become outdated very fast, thus TCP is not suitable in this case. Today, those applications would use UDP and either their own (proprietary) congestion control mechanism or refrain from using any congestion control.

DCCP offers a standardized end-to-end congestion control and the negotiation of different parameters to the applications. Furthermore, it is designed only to cause minimum overhead. Due to the use of TFRC or TCP-like congestion control (AIMD) respectively, DCCP is able to compete fairly with other TCP flows e.g. in the Internet.

At present, the DCCP specification reached the state of a draft standard. It is expected that DCCP has good prospects to become the most applied transport protocol for future streaming applications.

4.2 MPEG-4 Video Transfer with TFRC

In the last years, the Internet traffic caused by video transmission has drastically increased. Especially the advent of high speed connections to the Internet enables more and more “home-users” to get access to these services. The data transmission of most multimedia application is based on UDP, whose nature is to gather as much bandwidth as it can and may cheat other competing connections. This proceeding might lead to an increased level of congestion in the network.

[9] proposes a possibility to transmit MPEG-4 data in a way that those data streams share the bandwidth fairly with other flows of TCP connections. The MPEG-4 source coding standard is supposed to play an important role in the coding multimedia data in future. Firstly, some basics of MPEG-4 coding have to be introduced.

A MPEG-4 video stream is divided into several “Visual Objects” (VO), which may be a whole video frame or just part of it e.g. a single person. Each Visual Object consists of many “Visual Object Planes” (VOP) that are arranged to “Groups of VOPs” (GOV) as depicted in Figure 4-1. There are three kinds of VOPs (I,B,P) that differ in the way on how the VOP is coded.
As described in [9], this transfer mechanism uses TFRC to determine a target sending rate. In order to be able to cooperate with TFRC, applications need mechanisms to adjust the video rate according to guideline of TFRC. This is done by controlling the amount of video data that is transmitted to the receiver. Thus, the video rate can be adjusted by changing the coding parameters, which affects the video quality.

In this case, Fine Granular Scalability (FGS) is used for video coding. FGS divides the coded data into two parts, the Base Layer (BL) and the Enhanced Layer (EL). The Base Layer information is absolute necessary for decoding the video at the receiver, while adding Enhanced Layer information improves the video quality. This way, the video rate can be controlled.

Figure 4-2 shows three different methods, how the video rate can be adjusted according TFRC’s target rate. In the first case (a), the target rate is determined for every single VOP. The resulting transmission rate consists of the BL information and as much as EL information as permitted by TFRC. The second method (b) calculates the new target rate always at the beginning of a GOP. Now, this rate is valid for all VOPs of this GOP. The last possibility (c) is to average the video rate over the period of one GOP. Therefore, the rate of some VOPs may exceed the target rate but on average the boundary is kept by adding the same amount of EL information to every VOP. This enables a smoother change of the video rate.

[9] comes to a close that the third method is just right to support a stable video quality. Although there are still some problems of this procedure which have to be solved, it shows a possibility for the usage of TFRC.

### 4.3 TCP-friendly Multicast Congestion Control (TFMCC)

A problem of TFRC is that it is only intended to be used in unicast environments. Future applications like e.g. live video streaming over IP-networks may benefit from the use of
multicast. Therefore, TFRC is developed further to TFMCC which extends TFRC’s basic mechanisms to cope with multicast connections. The TFMMC protocol reaches the state of an Internet Draft and is described in [10].

According to [11], the main problem of using TFRC in a multicast environment is the large number of receivers that would send feedback reports to the sender. For this reason, some changes of this mechanism are needed.

One the one hand, the target sending rate is not determined at the sender but at the receivers. This is due to the fact that the receivers might be scattered all over the Internet and suffer from different network conditions. To be able to calculate this rate, the receiver needs to measure the round-trip-time between itself and the sender additionally. This is done with the help of packets that are echoed by the sender containing a timestamp when they where sent. Now, the receivers have to inform the sender on their results. To limit the amount of messages that arrive at the sender, a feedback message is only sent, if the estimated sending rate is lower than the actual rate. This mechanism works fine while a receiver gives instructions to the sender to decrease its sending rate, but does not deal with the case of an increase of the sending rate. To cope with this problem, a so called “Current Limiting Receiver” (CLR) is selected within the multicast group. The receiver that seems to have the lowest throughput of all other members of the group becomes CLR. Thus, it is allowed to send feedback messages immediately. Now, the sender is able to determine whether to increase the sending rate or not. A new CLR has to be chosen, as soon as the throughput of a further receiver seems to be lower than the throughput of the CLR or the current CLR leaves the multicast group. Furthermore, the sender needs to determine the round-trip-time to the CLR.

Simulations with TFMMC showed that it behaves similar fair to TCP like TFRC does. It is intended to work with multicast groups consisting of several thousand members.

### 4.4 Adaptive TCP Friendly Rate Control (ATFRC)

The ATFRC protocol is purposed in [12]. It is based on the mechanisms of TFRC, but ATFRC’s throughput is intended to be closer to the one of TCP than TFRC’s throughput. In addition, it reacts faster than TFRC to a decrease in the level of congestion.

ATFRC extends TFRC with an “adaptive sending rate control”. The calculation of the estimated throughput is split up into two phases. On the one hand, periods of triple duplicate ACKs without timeouts and on the other hand periods after a timeout occurred. This distinction is made in order to get a more accurate value of the allowed sending rate. Both events might reflect different network conditions. While triple duplicate ACKs are an indication for a slight congestion or an exceeding of the fair share sending rate, timeouts might be a sign of a serious congestion.

To be able to separate both events, the receiver keeps track of two different loss histories. For both histories, the loss event rates are calculated and fed back to the sender. The calculation and the feedback mechanisms are equal to those that are used in TFRC.

At the sender, two different equations are used to calculate the new sending rate. In the case of a timeout event, the actual sending rate is cut in half immediately. The next determination of a new sending rate is done with the throughput equation that is known from TFRC. In the other case and when the effect of the timeout has disappeared, a less conservative throughput equation is used, in order to approach to TCP’s sending rate better than TFRC does.

The fact that ATFRC reacts faster when bandwidth gets available is due to the more precise modelling of TCP’s behaviour.

Simulations in [12] showed that ATFRC’s throughput is almost as smooth as the throughput of TFRC.
5 Conclusions and Outlook

TCP-friendly Rate Control is a congestion control mechanism for unreliable datagram transmission. It offers many advantages to applications that are not able to use TCP because they need a stable transmission rate. TCP’s congestion control mechanism reacts very fast to changes in the level of congestion, while TFRC prevents those fluctuations by calculating a loss event rate, instead of regarding the packet loss rate. This loss event rate is determined at the receiver. Therefore TFRC uses a feedback mechanism between receiver and sender, to provide the sender with the actual loss rate. To improve the slower reaction of TFRC on a fast decrease of congestion compared with TCP, History Discounting is introduced.

Another important fact of TFRC is its fairness compared with other TCP-flows. Due to the use of the same throughput equation as TCP, TFRC has the same long-term throughput than TCP. That way, TFRC is well suited for e.g. multimedia transmissions without increasing the level of congestion in the network significantly.

Although TFRC is not a perfect solution, the prospects might be good that TFRC will play an important role in the future. The amount of applications like video streaming or Internet telephony will increase drastically in the next years. There are already many approaches to use TFRC with these applications.

In summary it may be said that the use of such rate control mechanisms will be essential when trying to cope with the increasing amounts of data that will be send through the Internet in the future.
References

[1] Floyd, S.; Fall, K.
“Promoting the Use of End-to-End Congestion Control in the Internet”,
IEEE/ACM Transactions on Networking
August 1999

“Equation Based Congestion Control for Unicast Applications”
SIGCOM 2000, Stockholm
Stockholm (Sweden), May 2000

“Equation Based Congestion Control for Unicast Applications: the
Extended Version”
Technical Report TR-00-03
International Computer Science Institute, Berkeley, March 2000

“TCP Friendly Rate Control (TFRC): Protocol Specification”
Request for Comments (RFC) 3448, The Internet Society
January 2003

„Datagram Congestion Control Protocol (DCCP)”
Internet Draft
October 2003

„Datagram Congestion Control Protocol (DCCP) Overview”
Berkeley, July 2003

[7] Padhye, J.
“Model-based Approach to TCP-friendly Congestion Control”
Ph.D. thesis
University of Massachusetts at Amherst, March 2000

[8] Widmer, Jörg
“Equation-Based Congestion Control”
Diploma Thesis
University of Mannheim, February 2000

“MPEG4-Video Transfer with TCP-friendly Rate Control”
E.S. Al-Shaer and G. Pacifici (Eds.): MMNS 2001, LNCS 2216 pp. 29-42
2001
[10] Widmer, J.; Handley, M.
Internet Draft
July 2003

“Extending Equation-based Congestion Control to Multicast Applications
Proc. ACM SIGCOMM 2001
San Diego (Cal.), August 2001

“ATFRC: Adaptive TCP Friendly Rate Control Protocol”
In Proceedings of ICOIN 2003
Jeju, Korea, Feb 2003