# Performance evaluation of TCP flows

Florian Hisch
Advisor: Fabien Geyer
Innovative Internet Technologies and Mobile Communication SS 2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: hisch@in.tum.de

## ABSTRACT
The behavior of different TCP algorithms has been subject to researchers over the past two decades. Various models have been developed to suit diverse network and traffic structures. This paper presents some of those analytic models and compares them with respect to their assumptions and validation techniques. It also classifies the given models and finally investigates their applicability on different network types.

## Keywords
TCP modeling, Fluid model, IEEE802.11, AQM, congestion control

## 1. INTRODUCTION
Several analytic TCP (Transmission Control Protocol) models, like [3], [5], [7], [12], [14], [15], [16] and [17], have been proposed in the last decades. This amount of different models is justified by the fact that over 90% of network traffic is transported by TCP.
The models are used to show and improve some TCP characteristic like the throughput, responsiveness or TCP friendliness. Models are also used to estimate the overall performance of a network topology. [10]

The large amount of different TCP models necessitate a comparison and survey on the assumptions and validation techniques. This paper reviews the outcome made by Khalifa et al.[10]. It updates their results on the basis of more recent developments and extends them with models for wireless networks. The rest of this paper is structured as follows: Section 2 gives an overview about TCP and Active Queue Management (AQM). The models are classified in Section 3 and discussed (in order of assumptions and network restrictions) in Section 4. A comparison of their applicability in different network topologies is made in Section 5. Conclusions are given in Section 6.

## 2. OVERVIEW OF TCP AND AQM
The *Transmission Control Protocol* (TCP) is one of the most used transport protocols (ISO/OSI level 4). In contrast to UDP (User Datagram Protocol) it's called a 'connection-oriented'-protocol, because it establishes a bidirectional connection between two end-points.
The protocols on network level (ISO/OSI level 3), such as IP, are packet oriented. These packets can arrive in any order, duplicated or can be even dropped by any point in the network. Latter will be simply called a packet *loss*. These losses are neither reported to the packet source nor to its destination.
A TCP connection is used to ensure the data exchange and comes therefore with several features like Three-Way-Handshake, Sequence-Numbers and Acknowledgments. It also tries to address two major issues in networking: flow control and congestion control.

The data to be send is organized in a Sliding Window, that limits the amount of data which can be send at once. All bytes (octets) get a consecutive sequence number which is also used to identify the packets on network level. The source sends WND (window size) bytes into the network. Those bytes are combined to segments which are acknowledged by the receiver with its sequence-number.

Flow control makes sure that the source never sends a higher amount of data than the receiver can handle. A field in the TCP header shows the current space in the receiver-buffer RWND.
Congestion control assumes that the probability of an IP-packet loss because of a transmission error is, thanks to channel-coding, very low (approx. $10^{-9}$ with 8b10b coding and Cat-5e cable) [18]. Most losses occur due to network congestion and thus induced packet drops. The congestion control uses a Congestion Window to determine how many bytes the network could handle. The size of this window is labeled CWND. [4]

The size of the Sliding Window WND is the minimum of the Receiver Window RWND and Congestion Window CWND. All reviewed papers assume that the Receiver Window RWND is as big as possible. That said, the Sliding Window size is only related to the Congestion Window CWND.

Many RFCs, like [9], [19] or [2], suggest different algorithms to calculate CWND. They have the most impact on the throughput of TCP flows and are therefore described in the following.

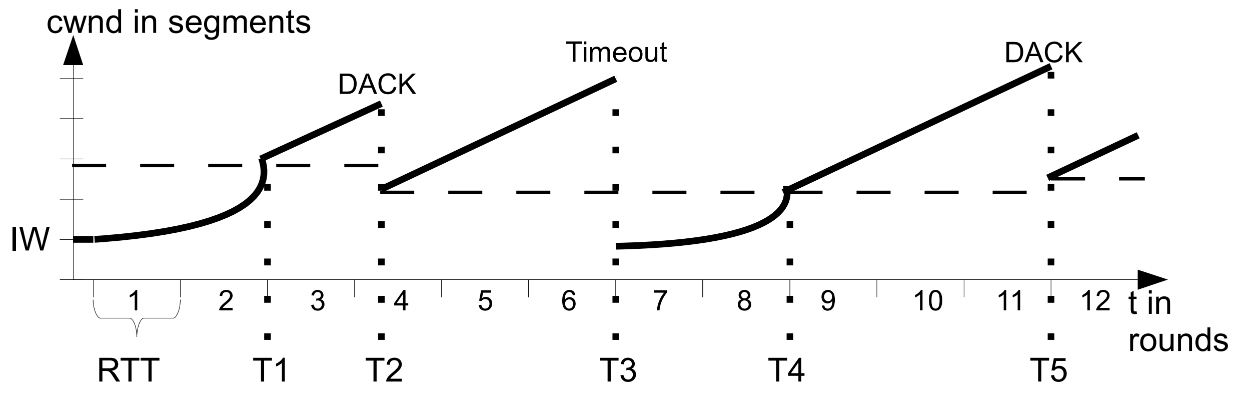

**Figure 1: Example: TCP Congestion Window during Initial Slow Start and Congestion Avoidance (TCP Reno like). Horizontal dashed line is the current threshold ssthresh. IW marks the initial window and RTT the round-trip-time. DACK stands for: triple Duplicated ACK arrived.**

### *TCP Slow Start*

The Congestion Window CWND is initially set to a small default value, e.g. $\text{IW} = \min(4 \cdot \text{SMSS}, \max(2 \cdot \text{SMSS}, 4380))$ in bytes [9]. The SMSS is the size of the largest segment that the sender can transmit.

For each acknowledged packet, the Congestion Window increases by one SMSS. Thus, the window grows roughly exponential by $\text{CWND}(t+1) = \text{CWND}(t) + \text{SMSS} \cdot \text{CWND}(t)$ after CWND bytes were send. The exact growing policy is subject to miscellaneous TCP versions like TCP Reno or TCP Vegas .

Slow Start ends and Congestion Avoidance begins when CWND exceeds a given threshold SSTHRESH or when congestion is detected. Latter is observed due to a Timeout or Duplicated Acknowledgments DACKs.

### *Congestion Avoidance*

During Congestion Avoidance the size of the Congestion Window increases only linear. TCP Reno for example increases with $\text{CWND} = \text{CWND} + \text{SMSS} \cdot \text{SMSS}/\text{CWND}$ for each received ACK. Doing so, the network load converges to its maximum in smaller steps to avoid a sudden overload.

SSTHRESH and CWND are reduced if congestion was detected. The exact reduction-behavior depends on the used TCP version. TCP Reno for example distinguishes between

| | |
|---|---|
| 3-Duplicated-ACKs: | $\text{SSTHRESH} = \text{CWND}/2$<br>$\text{CWND} = \text{CWND}/2$<br>start with Congestion Avoidance |
| Timeout : | $\text{SSTHRESH} = \text{CWND}/2$<br>$\text{CWND} = \text{IW}$<br>start with Slow Start |

Figure 1 shows an example developing of the Congestion Window size CWND over the time. At time $t = 0$ the window size is given by the initial window IW. In the section $0 < t < T1$ CWND grows exponential until CWND > SSTHRESH ($t = T1$). The window increases only linear until three Duplicated ACKs (DACKs) are received ($t = T2$) when CWND is halved and SSTHRESH is adjusted accordingly. TCP is in Congestion Avoidance phase until $t = T3$ where a Timeout (TO) occurred. Again, CWND and SSTHRESH are adapted, but the Timeout has triggered a Slow Start . After reaching the threshold ($t = T4$), TCP is back in Congestion Avoidance again.

### *Fast Retransmit / Fast Recovery*

A Duplicated ACK could be generated when the packets arrive out-of-order. This can be caused by full buffers, a re-ordering of network packets or a replication of the data packet or the ACK itself. RFC 5681 suggest to use a 'fast retransmit' algorithm to detect Duplicated ACKs and to perform an immediate retransmission of the lost data.

After this retransmission, the 'fast recovery' algorithm sends new data ignoring Duplicated ACKs until a non-duplicate ACK is received. [2] recommends, not to perform Slow Start but to artificially inflate the Congestion Window. This is legitimate because the Duplicated ACKs state that there were already packets received, which don't participate in the network traffic anymore. If ACKs for the previously missed packets arrive, the Congestion Window is deflated to its original size before Fast Retransmit was done.

### *Active Queue Management*

Buffers are used by routers to accommodate stress peaks in traffic and to keep a reserve, so that the link doesn't go idle [20]. Has the buffer reached his maximum amount of packets, further received packets have to be dropped. The method of simply fill and drop the rest is known as Taildrop algorithm.

Floyd et al. [8] show that the behavior of Taildrop leads to an oscillation between nearly idle and highly congested due to TCP flow synchronization. Active Queue Management (AQM) algorithms tries to avoid this synchronization by a more 'intelligent' dropping policy. RED (Random Early Detection) calculates for every incoming packet a drop probability. Doing so, a packet can be randomly dropped even if the buffer would have enough space to handle it. Enhancements of RED consider QoS (Quality of Service) factors to calculate the drop probability which prefer real-time flows like VoIP.

## 3. TCP MODEL CLASSIFICATION

Khalifa et al. [10] introduce three different classes for TCP models. The first one characterizes a TCP flow by its average throughput. The second class pays attention to the interaction between TCP flows, whereas the last class deals with TCP dynamics.

This paper presents a new classification, which distinguishes between Single connection models (SCM), Multiple connections models (MCM) and Fluid models.

### 3.1 Single connection models

This class observes a single TCP flow and its throughput in a single bottleneck network. Padhye et al. [15] give a model for the steady-state TCP Reno throughput of bulk transfers. This model was used by Parvez et al. [16] to develop a model for the more up-to-date TCP version NewReno. Cardwell et al. [5] introduce in contrast to the above ones, a model for a short-lived TCP flow. The papers [16] and [5] are reviewed more in detail below.

#### *Three-Way-Handshake*

The Three-Way-Handshake is used to establish the connection between two TCP endpoints. [5] et al. consider two phases:

- The initiating host sends $i \geq 0$ SYN packets until the $(i+1)$-th SYN arrives successfully. The probability of a successful packet transfer is $p_f$ (forward probability).
- The receiver of the SYN packets answers with $j \geq 0$ SYN/ACK packets until the $(j+1)$-th SYN/ACK arrives successfully. The probability of a successful packet transfer is $p_r$ (reverse probability).

Let $P_h(i,j)$ the probability of having a successful Three-Way-Handshake with exact $i$ respectively $j$ packet losses (see above).

$$P_h(i,j) = p_r^i \cdot (1-p_r) \cdot p_f^j \cdot (1-p_f)$$

The probability that the latency $D_h$, of a general Three-Way-Handshake, is $t$ seconds or less, is

$$P[D_h \leq t] = \sum_{D_h(i,j) \leq t} P_h(i,j)$$

where $D_h(i,j)$ is the latency for exact $i$ and $j$ losses. $D_h(i,j)$ is given as

$$D_h(i,j) = \text{RTT} + (2^i + 2^j - 2) \cdot D_s$$

RTT describes the average round trip time between the two points of the TCP flow. $D_s$ is the minimum waiting time for a answer during Three-Way-Handshake. This time doubles with every try.

Cardwell et al. [5] present an approximation of the expected handshake time for loss rates low enough that most Three-Way-Handshakes succeed before TCP gives up.

$$E[D_h] = \text{RTT} + D_s \cdot \left(\frac{1-p_r}{1-2p_r} + \frac{1-p_f}{1-2p_f} - 2\right) \quad (1)$$

#### *Initial Slow Start*

After a successful Three-Way-Handshake, TCP starts with an Initial Slow Start . Cardwell et al. [5] observe two cases: The first one lets the Congestion Window increase unlimited, whereas the second case gives a maximum window size $W_{\max}$. To compare this model to the other ones, $W_{\max}$ will be set to infinity.

Let $E[S_{\text{ss}}]$, the expected number of data segments send before a loss occurs, be

$$\begin{aligned} E[S_{\text{ss}}] &= \left(\sum_{k=0}^{S-1}(1-p)^k \cdot p \cdot k\right) + (1-p)^S \cdot S \\ &= \frac{(1-(1-p)^S(1-p)}{p} + 1 \end{aligned}$$

where $S$ is the total amount of data to be send in the whole TCP flow and $p$ the probability for a segment loss. $p$ does not distinguish between forward and reverse probability anymore.

CWND increases during Slow Start with

$$\begin{aligned} \text{CWND}_{i+1} &= \text{CWND}_i + \text{CWND}_i/b \\ &= \gamma \cdot \text{CWND}_i \end{aligned}$$

with $\gamma = (1 + 1/b)$ and $w_1$ as the initial window size.

The time needed to send all segments in a Congestion Window is further referred as a 'round'.

$$i = \log_\gamma\left(\frac{S_i(\gamma-1)}{w_1} + 1\right)$$

is the count of rounds needed to send $S_i$ segments.

The formulas for $i$ and $E[S_{\text{ss}}]$ lead to the expected time to send $E[S_{\text{ss}}]$ data segments during Slow Start:

$$E[D_{\text{ss}}] = \text{RTT} \cdot \log_\gamma\left(\frac{E[S_{\text{ss}}](\gamma-1)}{w_1} + 1\right) \quad (2)$$

#### *Congestion Avoidance / Fast Recovery*

TCP distinguishes two ways to detect a packet loss: Triple-Duplicate-ACKs (DACK) and Timeouts (TO). Parvez et al. [16] derive a model without Timeouts (NoTO) first and extends it with the missing behavior later.

The evolution of the Congestion Window during Congestion Avoidance and Fast Recovery can be modeled as statistically identical cycles. Those cycles consists of several rounds, like the ones described above. Each cycle (CAFR) combines a Congestion Avoidance period and a Fast Recovery period. The throughput of one of those cycles is given as

$$E[T_{\text{NoTO}}] = \frac{E[S_{\text{CAFR}}]}{E[D_{\text{CAFR}}]}$$

where $E[S_{\text{CAFR}}]$ is the expected number of successfully transmitted segments and $E[D_{\text{CAFR}}]$ the duration for this period.

Parvez et al. [16] use, besides the packet loss probability $p$, also a segment loss rate $q$ within a loss event. This indicates how many segments relative to all segments in a round are affected by a packet loss. During the Three-Way-Handshake and the initial Slow Start, $q$ was set to 1, meaning that if a packet loss occurs, all segments in that round are lost.

The expected amount $E[m]$ of uniformly spaced drops in a window of size $W$ can be obtained as

$$\begin{aligned} E[m] &= \sum_{m=1}^{W-3} m \cdot f(m-1, W-1, q) \\ &= \sum_{m=1}^{W-3} m \cdot \binom{W}{m} \cdot q^{m-1} \cdot (1-q)^{W-m} \\ &\approx 1 + (W-1) \cdot q \approx 1 + W \cdot q \end{aligned}$$

The upper border for the sum $W - 3$ is valid, as there has to be a DACK. $f(m-1, W-1, q)$ is obviously the Binomial probability mass function.

$E[S_{\text{CAFR}}]$ and $E[D_{\text{CAFR}}]$ can be split in a Linear Increase (LI), $\beta$ and Fast Recovery (FR) phase. For example $E[S_{\text{CAFR}}] = E[S_{\text{LI}}] + E[S_\beta] + E[S_{\text{FR}}]$.
The most interesting value is $E[S_{\text{FR}}]$, because of the inflation respectively the deflation of the Congestion Window . Each DACK increases CWND by one segment. This behavior leads to the possibility that during Fast Recovery not only missing (lost) data is transmitted, but also new data. If $m > \frac{W}{2}$, TCP will not transmit any new data. Parvez et al. [16] obtain

$$\begin{aligned} E[S_{\text{FR}}]^{m \le \frac{W}{2}} &= \sum_{j=1}^{m} \left(\frac{W}{2} - m + j - 1\right) = \frac{m}{2} \cdot (W - m - 1) \\ E[S_{\text{FR}}]^{m > \frac{W}{2}} &= \sum_{k=m-\frac{W}{2}+1}^{m-1} \left(\frac{W}{2} - m + k\right) = \frac{W^2}{8} - \frac{W}{4} \\ E[S_{\text{FR}}] &= \sum_{m=1}^{\frac{W}{2}} f(m-1, W-1, q) \cdot S_{\text{FR}}^{m \le \frac{W}{2}} \\ &+ \sum_{m=\frac{W}{2}+1}^{W-3} f(m-1, W-1, q) \cdot S_{\text{FR}}^{m > \frac{W}{2}} \end{aligned}$$

$$\approx \frac{W^2}{2}(q - q^2) + \frac{W}{2}(1 - 5q + 3q^2) - (1 - 2q + q^2)$$

where $f(m-1, W-1, q)$ is again the Binomial probability mass function.

The remaining expressions of those phases are:

$$\begin{aligned} E[S_{\text{LI}}] &= \sum_{i=\frac{W}{2}}^{W} i = \frac{3}{8}W^2 + \frac{3}{4}W \\ E[S_\beta] &= \frac{\text{W}}{2} \\ E[D_{\text{LI}}] &= \left(\frac{W}{2} + 1\right) \cdot \text{RTT} \\ E[D_\beta] &= \frac{\text{RTT}}{2} \\ E[D_{\text{FR}}] &= E[m] \cdot \text{RTT} \approx (1 + Wq) \cdot \text{RTT} \end{aligned}$$

With the expressions above the total terms $E[S_{\text{CAFR}}]$ and $E[D_{\text{CAFR}}]$ are:

$$E[S_{\text{CAFR}}] = \left(\frac{3}{8} + \frac{q}{2} - \frac{q^2}{2}\right) W^2 + \left(\frac{7}{4} - \frac{5q}{2} + \frac{3q^2}{2}\right) W - \left(1 - 2q + q^2\right) \quad (3)$$

$$E[D_{\text{CAFR}}] = \left(\frac{W}{2} + Wq + \frac{5}{2}\right) \text{RTT} \quad (4)$$

The formulas above need an approximation for the Congestion Window size $W$. The size can be expressed in terms of $p$ and $q$ as

$$W \approx \frac{10pq - 5p + \sqrt{p(24 + 32q + 49p}}{p(3 + 4q)} \quad (5)$$

To extend the NoTO model and to include also the Timeout case, Parvez et al. [16] adapt their throughput expression given in (6). $p_{\text{TO}}$ is the probability that a loss leads to a Timeout. $E[D_{\text{CA}}] + E[D_{\text{TO}}] + E[D_{\text{SS}}]$ combines a Congestion Avoidance (CA) phase with the Timeout itself and the consecutive Slow Start (SS) phase. From the NoTO model follows that $E[D_{\text{CA}}] = E[D_{\text{LI}}] + E[D_\beta]$ and $E[S_{\text{CA}}] = E[S_{\text{LI}}] + E[S_\beta]$. Since TCP forgets all outstanding packets after a Timeout, $E[S_\beta] = E[D_\beta] = 0$ and thus $E[S_{\text{CA}}] = E[S_{\text{LI}}]$ respectively $E[D_{\text{CA}}] = E[D_{\text{LI}}]$.

The derivation of $p_{\text{TO}}$ has to be split up in $p_{\text{TO}} = p_{\text{DTO}} + p_{\text{IFR}}$, a direct Timeout during Congestion Avoidance and a Timeout during Fast Recovery. First one is detected by more than $W - 3$ segments are lost. From the Binomial probability mass function $f$ follows

$$p_{\text{DTO}} = \sum_{m=W-2}^{W} f(m-1, W-1, q)$$

The latter case probability derivation is rather complex and includes several assumptions (see section 4). Therefore only the result is presented

$$p_{\text{IFR}} = \sum_{m=1}^{W-3} f(m-1, W-1, q) \left[1 - (1-p)^{\frac{mW}{2}}\right]$$

$$E[T_{\text{Full}}] = \frac{(1-p_{\text{TO}})E[S_{\text{CAFR}}] + p_{\text{TO}}(E[S_{\text{CA}}] + E[S_{\text{TO}}] + E[S_{\text{SS}}])}{(1-p_{\text{TO}})E[D_{\text{CAFR}}] + p_{\text{TO}}(E[D_{\text{CA}}] + E[D_{\text{TO}}] + E[D_{\text{SS}}])} \quad (6)$$

The total probability for a Timeout is given as

$$p_{\text{TO}} = 1 - \sum_{m=1}^{W-3} f(m-1, W-1, q)\left[(1-p)^{\frac{mW}{2}}\right] \quad (7)$$

Obviously TCP transmits no new data during a Timeout, thus $E[S_{\text{TO}}] = 0$. $T_0$ is the initial Timeout duration which doubles with each try. So, the Timeout of the $i$-th retransmission is $T_i = 2^{i-1}T_0$. The probability that an $i$-th retransmission is needed, is $\frac{p^{i-1}}{1-p}$ (geometric series). The total time spend during the Timeout is

$$E[D_{\text{TO}}] = T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p}$$

In the following Slow Start phase, the Congestion Window starts with size 1 and doubles each round until it reaches $\frac{W}{2}$. Similar to the derivation of the Initial Slow Start phase follows

$$E[S_{\text{SS}}] = 1 + 2 + 4 + \ldots + \frac{W}{4} = 2^{1+\log\frac{W}{4}} - 1$$

$$E[D_{\text{SS}}] = \left(\log\frac{W}{4} + 1\right)\text{RTT}$$

With the expressions above the total terms $S_{\text{Full}}$ and $D_{\text{Full}}$ are:

$$E[S_{\text{Full}}] = \frac{1}{p} + \frac{W^2 q}{1 + Wq} \quad (8)$$

$$E[D_{\text{Full}}] = E[N]\cdot\text{RTT} + p_{\text{TO}}\left((1 + 2p + 4p^2)T_0 + \left(1 + \log\frac{W}{4}\right)\text{RTT}\right) \quad (9)$$

where $E[N] = \left(\frac{W}{2} + \frac{3}{2} + (1-p_{\text{TO}})(1+Wq)\right)$.

### *Model for Short-Lived Transfers*

Many TCP flows in today's Internet, transport only a small amount (less than 10 KB) of data and will therefore never leave the Slow Start phase. Cardwell et al. [5] used this statistical information to derive a model which focuses on the Three-Way-Handshake and Slow Start.
The throughput of a Short-Lived TCP transfer is approximated by:

$$E[T_{\text{short}}] = \frac{S}{E[D_{\text{ss}}] + \frac{(S - E[S_{\text{ss}}])}{E[T_{\text{Full}}]}} \quad (10)$$

The exact expression given by Cardwell et al. [5] also consider the delays $E[D_{\text{loss}}]$ and $E[D_{\text{delack}}]$ which are caused by the first loss and delayed ACKs (Nagle algorithm).

## 3.2 Multiple connections models

This class observes many TCP flows and their throughput in a multiple bottleneck network. Firoiu et al. [7] are using their results in the single bottleneck case to derive also a model for the steady-state TCP throughput in larger networks. In contrast to Single connection models is a detailed analysis of startup effects (e.g. Three-Way-Handshake) due to the parallelism of the many TCP flows not possible. Though, heterogeneous traffic, like TCP and UDP, and Active Queuing Management (e.g. RED) can be modeled additionally.

Because of the similarity of Multiple connections models to Single connection models, a detailed derivation of the throughput is skipped. Nevertheless, the throughput at the $k$-th link in the network is given by

$$T_k\left(\left\{\{\overline{q}_m\}_{i,j}\right\}_k\right) = \sum_i^{\{f_i\}_k} t_i\left(\{\overline{q}_k\}\right) + \sum_j^{\{u_j\}_k} r_j\left(p'\right) \quad (11)$$

with the average queue length on link $k$ $\overline{q}_k$ and $p'$ the overall drop rate over multiple bottlenecks. $t_i(.)$ and $r_j(.)$ is the throughput of the $i$-th TCP flow and $j$-th UDP flow. The total throughput of the whole network is the sum over all $T_k\left(\left\{\{\overline{q}_m\}_{i,j}\right\}_k\right)$.

## 3.3 Fluid models

Fluid models use adjacency matrices to define a network. It distinguishes between Sources and Links, which both have a different view on the traffic. This can be used to model not only TCP's congestion control but also some Active Queuing Management (AQM).

### *Network model*

A typical approach is to define a set $\mathbb{L}$ of links with finite capacities $c = (c_l, l \in \mathbb{L})$. Sources are centralized in the set $\mathbb{S}$ and every of these sources can use $\mathbb{L}_s \subseteq \mathbb{L}$ of links. To connect the links with the sources a routing matrix

$$R_{ls} = \begin{cases} 1, & \text{if } l \in \mathbb{L}_s \\ 0, & \textit{otherwise} \end{cases} \quad (12)$$

is defined. Each source is associated with its transmission rate $x_s(t)$, whereas the links are associated with a congestion measure $p_l(t) \geq 0$ at time $t$. Using the routing matrix $R_{ls}$ the following expressions can be derived:

$$\begin{aligned} y_l &= \textstyle\sum_s R_{ls}x_s(t) && \text{aggregated source rate at link } l \\ q_s &= \textstyle\sum_l R_{ls}p_l(t) && \text{end-to-end congestion for sources} \\ y(t) &= Rx(t) \\ q(t) &= R^T p(t) \end{aligned}$$

Like in reality, a source $s$ can only observe its own traffic rate $x_s(t)$ and the end-to-end congestion $q_s(t)$. Link $l$ can only observe its own congestion $p_l(t)$ and the flow rate $y_l(t)$.

To model the adjustments on transmission rate and congestion measure Low [14] uses three state functions $F, G, H$. They are defined as

$$\begin{aligned} x_s(t+1) &= F_s(x_s(t), q_s(t)) && (13) \\ p_l(t+1) &= G_l(y_l(t), p_l(t), v_l(t)) && (14) \\ v_l(t+1) &= H_l(y_l(t), p_l(t), v_l(t)) && (15) \end{aligned}$$

The internal state vector $v_l(t)$ gives the opportunity to store information, such as queue lengths. A specific set of $F, G, H$ functions is the essential part of a Fluid model. It contains the whole behavior of TCP ($F$) and AQM ($G, H$).

To get a steady-state, the $F, G, H$ functions above shall reach an equilibrium point $(x, p)$ so that $x_s = F_s(x_s, q_s)$. To get the end-to-end congestion in the specific equilibrium point, an additional function $f_s$ is defined with $q_s = f_s(x_s) > 0$. This can be used to identify the equilibrium point without congestion $(x_s, 0)$. Obviously $F_s(x_s, 0) = x_s$ and $f_s(x_s) = 0$. The existence of these equilibrium points is subject to several assumption respectively conditions and not all Fluid models can comply with all [17].

To determine the overall performance of the network in a steady-state equilibrium point, a utility function for each source $s$ is defined as

$$U_s(x_s) = \int f_s(x_s)dx, \qquad x_s \geq 0$$

This utility function can be used to maximize the aggregated utility given by an equilibrium point $x_s$ by

$$\max_{x \geq 0} \sum_s U_s(x_s), \qquad \text{subject to } Rx \leq c \qquad (16)$$

under the premise that the flow rate $y = Rx(t)$ never exceeds the capacities $c$.

## *(F,G,H)-models*

Some definitions that are used in the following are adjusted from the original papers to suit the definitions made in previous models:
Let $W_s$ be the Congestion Window size of source $s$ and let $\mathrm{RTT}_s$ be the equilibrium round-trip time which is assumed to be constant. The average source rate $x_s(t)$ is defined by $x_s(t) = W_s(t)/\mathrm{RTT}_s$, since the time is discrete and is on the order of several round-trip times.

Exemplary derivation of a $F$ model for TCP Reno:
Since Fluid models can only observe the steady-state of the network, Three-Way-Handshake and Slow Start are no subject. One variant of TCP Reno halves the Congestion Window every time congestion is detected. This congestion detection is done by 'marks'. Let $p_l(t)$ be the current marking probability at link $l$, then the end-to-end loss probability is

$$q_s(t) = \sum_l R_{ls} p_l(t)$$

This definition is only valid if $0 \leq q_s(t) \leq 1$, so the probabilities $p_l$ have to be small enough. On average, source $s$ receives $x(t)(1 - q_s(t))$ positive acknowledgments per unit time. Each positive ACK increases $W_s$ by $1/W_s(t)$ (Congestion Avoidance ). The $x(t)q_s(t)$ negative ACKs (marks) per unit time halves the Congestion Window. In total, the Congestion Window changes on average by

$$x_s(t)(1 - q_s(t)) \cdot \frac{1}{W_s(t)} - x_s(t)q_s(t) \cdot \frac{1}{2} \cdot \frac{4W_s(t)}{3}$$

The TCP Reno algorithm in $F_s(x_s(t), q_s(t))$ shape is given as

$$\begin{aligned} x_s(t+1) &= F_s(x_s(t), q_s(t)) \\ &= \left[x_s(t) + \frac{1 - q_s(t)}{\mathrm{RTT}_s^2} - \frac{2}{3} q_s(t) x_s^2(t)\right]^+ \qquad (17) \end{aligned}$$

Using the $F$ function above, the $f_s(x_s) = q_s$ function for TCP Reno is

$$q_s = \frac{3}{2x_s^2 \mathrm{RTT}_s^2 + 3} =: f_s(x_s)$$

and the utility function $U_s(x_s)$ can be expressed as

$$U_s(x_s) = \frac{\sqrt{3/2}}{\mathrm{RTT}_s} \tan^{-1}\left(\sqrt{\frac{2}{3}} x_s \mathrm{RTT}_s\right) \qquad (18)$$

With only one internal state in the functions $G, H$, it is not possible to model abrupt changes in the queuing behavior. For example, there is only an approximation for DropTail since the dropping immediately starts if the buffer is full. Dropping algorithms like RED (Random Early Drop) and REM (Random Exponential Marking) which are more 'smooth', can be modeled without any discrepancy. Exemplarily derivation of a $G, H$ model for RED:

RED consists of two internal variables, the current queue length $b_l(t)$ and the average queue length $r_l(t)$, which is calculated with a factor $\alpha \in (0, 1)$:

$$\begin{aligned} b_l(t+1) &= [b_l(t) + y_l(t) - c_l]^+ \\ r_l(t+1) &= (1 - \alpha_l) r_l(t) + \alpha_l b_l(t) \end{aligned}$$

The marking (loss) probability of the link $l$ is given as a function of the average queue length $r_l(t)$. There are three additional RED characteristics: $\underline{b_l}$ and $\overline{b_l}$ being a lower / upper threshold and $m_l \in (0, 1)$ being a value to adjust the stringency. If $m_l$ is nearly 0, RED will start dropping some packets if the queue is longer than $\overline{b_l}$. If $m_l$ is nearly 1, RED will drop every packet if the queue length exceeds $\overline{b_l}$. The exact expression for the RED dropping behavior is:

$$p_l(t) = \begin{cases} 0, & r_l(t) \leq \underline{b_l} \\ \frac{m_l}{\overline{b_l} - \underline{b_l}}(r_l(t) - \underline{b_l}), & \underline{b_l} \leq r_l(t) \leq \overline{b_l} \\ \frac{1 - m_l}{\overline{b_l}}(r_l(t) - \overline{b_l}), & \overline{b_l} \leq r_l(t) \leq 2\overline{b_l} \\ 1, & \textit{otherwise} \end{cases}$$

To write the formulas above in the $G, H$ form, the internal state vector hast to be:

$$p_l(t+1) = G_l(y_l(t), p_l(t), v_l(t)) \tag{19}$$

$$v(t+1) = \begin{pmatrix} b_l(t+1) \\ r_l(t+1) \end{pmatrix} = H_l(y_l(t), p_l(t), v_l(t))$$

$$= \begin{pmatrix} [b_l(t) + y_l(t) - c_l]^+ \\ (1-\alpha_l)r_l(t) + \alpha_l b_l(t) \end{pmatrix} \tag{20}$$

## 4. ASSUMPTIONS AND VALIDATION

Before making any assumptions, there is not a generally accepted TCP implementation. The RFCs describing the behavior of TCP are full of recommendations and thus TCP implementations vary very much. Therefore, assumptions have to be made to define the TCP environment. These assumptions can be categorized in classes regarding the Data Transfer, End Points and Network.

### *Data Transfer*

The availability of the data to be sent is one of the first assumptions made during the modeling process. Usually, it is assumed that the application layer generates the data without any latency. This assumption is only true if the application can generate data faster than the TCP connection can send it. Compared to nowadays communications technologies (Ethernet, Wifi, ...), computer memory and CPUs are rarely the bottleneck in TCP transfers. Tough, for special cases, like the direct point-to-point high speed connection (e.g. via Gigabit LAN), the transmit queue may run empty.

Most of the paper propose a predefined amount of data or rather the Data transfer length $d$. Knowing this length, it is possible to estimate how much the initial TCP behavior (Three-Way-Handshake, Initial Slow Start) affects the over all throughput. Is $d$ rather small (around 10 KB), Short-Lived models like [5], which consider the initial TCP behavior, are more accurate than the Steady-State models. Those ignore the start-up effects and simplify their model by considering only Congestion Avoidance and Fast Recovery algorithms.

Fluid models are Steady-State models with an unlimited amount of data. They can give no statement, how much data is needed to reach the equilibrium point of the network. To get an approximation for the needed data length or the time to reach this point, expressions from the Single connection models can be used.

### *End Points*

TCP implementations and TCP versions vary, as already mentioned, very much. This affects especially the congestion control. Three different TCP versions are referred in the papers: TCP Tahoe (1988), TCP Reno (1990) respectively TCP NewReno (1999) and TCP Vegas (1995-2005). TCP Reno is the version, which is implemented in the most operating systems. [1]

Most papers give their model in different variations for the loss detection algorithms of some of the TCP versions. Additional assumptions regarding the End Points are

- the average number of segments acknowledged by a single ACK,
- the size of one segment,
- the size of the window limitation (the size of the Receiver Window ), which can be also set to infinity, and
- timing constraints (duration of initial Timeout, Three-Way-Handshake Timeout , ...).

A common assumption is to neglect additional algorithms like the Nagle algorithm and silly window syndrome.

### *Network*

Since TCP is theoretical independent from the network layer, it can be used on any network topology. Nevertheless TCP throughput is always a function of the loss probability $p$ which depends on the used network. Hence, models never specify the exact topology, but are more accurate in a certain network category. These categories can be distinguished by means of tethered / wireless connections and whether there are single or multiple bottlenecks.

A typical assumption (found in all reviewed models) is to neglect transmission and processing delays. Further, the round-trip time RTT is assumed to be independent from CWND and in most cases even constant. Doing so, the model becomes less complex, because a variable RTT would add an additional stochastic process. This assumption is however not justifiable, since nobody knows how the TCP packets are routed through the network, fragmented or delayed by Active Queuing Management. Even TCP itself has to measure the RTT during the whole connection time to adjust the Retransmission Timeout duration RTO to the current network situation. Hence, RTT cannot be assumed to be constant.
Wired networks, especially those with only direct links (no switches), have a small RTT with a little variance. Most network layers for wireless networks (e.g. 802.11x, Wireless LAN) use CSMA/CA, which contains a random waiting time after a collision has occurred. Thus, wireless connections have a bigger RTT with a higher variance.

Table 3 summarizes some of the characteristics of all reviewed models.

### *Model Validation*

All models have to be proofed, how accurately they reproduce the reality. They can be validated by comparing the calculated throughput with results from: simulations, controlled measurements under conditions that suit the assumptions and live measurements from the Internet. Table 1 shows some of the advantages and disadvantages of the said validation methods. Table 2 shows the validation methods used by the miscellaneous papers.

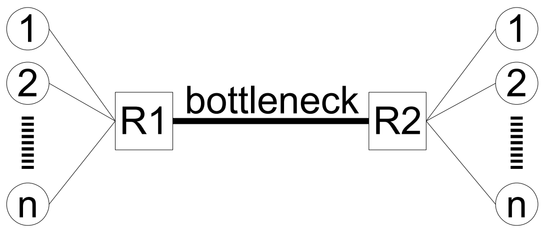

**Figure 2: Single bottleneck layout. $n$ sources and sinks are connected via a single link which has more incoming data than it can transmit.**

## 5. APPLICABILITY

None of the models is applicable to every network topology. The applicability can be defined on the basis of some assumptions. Table 4 shows the network restrictions that can be derived from the assumptions. For example, if the RTT is expected to be constant, the network topology has to be tethered or contains not more than one single wireless hop. This relation can be explained by the used physical layers (mostly Ethernet respectively 802.11x) and especially by the different CSMA variants (Collision Detection / Collision Avoidance).

Most models consider a single bottleneck. This is however given only in small networks. Firoiu et al. [7] define that a link is a bottleneck when the arrival rate of incoming packets is higher than service rate (departure rate), and the packets are enqueued. Thus, a network topology has to be like figure 5 to be called single bottleneck. Typical networks are more complex and hence contain multiple bottlenecks. The transition of a single to a multiple bottleneck model and their resulting (relative) error is shown in [7]. Fluid models consider multiple bottlenecks per default as there are all link capacities known.

## 6. CONCLUSION

This paper presents an extended overview of TCP models based on the previous summary made by Khalifa et al. [10]. It classified the models in Single connection models, Multiple connections models and Fluid models, compared them and surveyed their validation and applicability.

Single connection models (SCM) can reproduce TCP in a microscopic way (considering Three-Way-Handshake, Slow Start, Congestion Avoidance and Fast Recovery) but loose accuracy in complex networks with multiple bottlenecks. Multiple connections models (MCM) extend SCM and are used in complex networks with multiple bottlenecks. They reproduce the steady-state throughput of many TCP flows and can model heterogeneous traffic (e.g. TCP and UDP flows). In contrast, Fluid models observe only the macroscopic behavior of TCP but is accurate also in complex networks. Fluid models can be easily extended to consider Active Queuing Management.

The comparison showed that most of the models are only validated by simulation. The complex behavior of TCP, with its not standardized implementations, can generate side-effects which lower the simulated throughput. Only two models have been validated by live measurements in the Internet. Future models should be validated especially by live measurements and comparisons to existing models, to show their achievements.

The current extensive growth of wireless networks require new models for the TCP throughput. Most of the reviewed models assume characteristics of a tethered network which makes them inaccurate in wireless (especially Ad-Hoc) networks. Future models have to consider both network types.

## 7. REFERENCES

[1] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock. Host-to-host congestion control for TCP. *IEEE Communication Surveys and Tutorials*, 12(3):304–342, July 2010. (2012 The IEEE Communications Society Best Tutorial Paper Award).

[2] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (Draft Standard), Sept. 2009.

[3] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. *SIGCOMM Comput. Commun. Rev.*, 30(4):231–242, Aug. 2000.

[4] A. Badach and E. Hoffmann. *Technik der IP-Netze - TCP/IP incl. IPv6 - Funktionsweise, Protokolle und Dienste, 2. erw. Aufl.* Carl Hanser Verlag, München, 2007.

[5] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1742–1751 vol.3, Mar 2000.

[6] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang. Theories and models for Internet quality of service. *Proceedings of the IEEE*, 90(9):1565–1591, Sep 2002.

[7] V. Firoiu, I. Yeom, and X. Zhang. A framework for practical performance evaluation and traffic engineering in IP networks. *IEEE ICT 2001*, 2001.

[8] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, Aug 1993.

[9] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[10] I. Khalifa and L. Trajkovic. An overview and comparison of analytical TCP models. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 5, pages V–469–V–472 Vol.5, May 2004.

[11] X. Lan, S. Li, and S. Zhang. Network Coded TCP throughput Analysis in Wireless Mesh Networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 225–232, Dec 2013.

[12] X. Li, P.-Y. Kong, and K.-C. Chua. TCP Performance

in IEEE 802.11-Based Ad Hoc Networks with Multiple Wireless Lossy Links. *Mobile Computing, IEEE Transactions on*, 6(12):1329–1342, Dec 2007.

[13] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-scale IP Networks. *SIGMETRICS Perform. Eval. Rev.*, 31(1):91–101, June 2003.

[14] S. Low. A duality model of TCP and queue management algorithms. *Networking, IEEE/ACM Transactions on*, 11(4):525–536, Aug 2003.

[15] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, Apr. 2000.

[16] N. Parvez, A. Mahanti, and C. Williamson. An Analytic Throughput Model for TCP NewReno. *IEEE/ACM Trans. Netw.*, 18(2):448–461, Apr. 2010.

[17] Q. Peng, A. Walid, and S. H. Low. Multipath TCP Algorithms: Theory and Design. *SIGMETRICS Perform. Eval. Rev.*, 41(1):305–316, June 2013.

[18] E. Stein. *Taschenbuch Rechnernetze und Internet. Mit 94 Tabellen.* Fachbuchverl. Leipzig im Carl-Hanser-Verl., München, 3., neu bearb. aufl. edition, 2008.

[19] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001 (Proposed Standard), Jan. 1997. Obsoleted by RFC 2581.

[20] D. Wischik and N. McKeown. Part I: Buffer Sizes for Core Routers. *SIGCOMM Comput. Commun. Rev.*, 35(3):75–78, July 2005.

**Table 1: Advanteges and disadvanteges of model validiation methods**

| | advantages | disadvantages |
|---|---|---|
| simulation | full knowledge on the environment | not simulated side-effects can affect throughput in reality |
| | all characteristics observable (e.g. queue lengths on all routers) | simulation and model fails can compensate each other |
| | measurement recording does not falsify the results | TCP implementation can differ from real implementation in operation systems |
| controlled measurements | partial knowledge on the environment, but enough to ensure the assumptions | high effort to control the network environment |
| | real-world side-effects affect the results | measurement recordings partially falsify the result, because of additional traffic |
| live measurements | model can be validated against the reality | small knowledge of the environment (topology is rather complex) |

**Table 2: Model validation techniques.**

| Model | | Simulations | Controlled meas. | Live meas. | Comparison |
|---|---|---|---|---|---|
| Padhye | [15] | Yes | Yes | No | None |
| Cardwell | [5] | Yes | Yes | Yes | [15] |
| Firoiu | [7] | Yes | No | No | None |
| Low | [14] | Yes | No | No | None |
| Liu | [13] | Yes | No | No | None |
| Altman | [3] | No | Yes | No | [15] |
| Li | [12] | Yes | No | No | None |
| Parvez | [16] | Yes | Yes | Yes | [15] |
| Peng | [17] | Yes | No | No | [14] |

**Table 3: Summary of the models. The TCP algorithms are Three-Way-Handshake (TWH), Initial Slow Start (ISS), Congestion Avoidance (CA) and Fast Recovery (FRC). Losses are detected by Triple Duplicated ACKs (DACK) or Timeouts (TO). CA[1] and DACK[1] represent that Fluid models observe only the macroscopic behavior of TCP. [2]: Firoiu et al. introduce a SCM and a MCM, which means that TWH and ISS are only modled in the SCM.**

| Model | | Type | Based on | Lengths | TCP algorithm | Loss detection | Data/ACK Losses | AQM |
|---|---|---|---|---|---|---|---|---|
| Padhye | [15] | SCM | | Long-lived | CA | DACK, TO | Bursty/None | Taildrop |
| Cardwell | [5] | SCM | [15] | Short/Arbitrary | TWH, ISS, CA | DACK, TO | Bursty/SYN only | Taildrop |
| Firoiu | [7] | MCM[2] | [15], [5] | Arbitrary | TWH, ISS, CA | DACK, TO | Bursty/SYN only | RED |
| Low | [14] | Fluid | | Long-lived | CA[1] | DACK[1] | /None | RED, REM |
| Liu | [13] | Fluid | | Long-lived | CA[1] | DACK[1] | Bursty/None | RED |
| Altman | [3] | SCM | [15] | Long-lived | CA | DACK, TO | Stationary/None | Taildrop |
| Li | [12] | SCM | | Long-lived | CA, FRC | DACK, TO | Bursty/None | Taildrop |
| Parvez | [16] | SCM | [15] | Long-lived | CA, FRC | DACK, TO | Adjustable/None | Taildrop |
| Peng | [17] | Fluid | [14] | Long-lived | CA[1] | DACK[1] | Bursty/None | RED |

**Table 4: Assumptions and the validity on different network types. A X marks wether the RTT vaiance or the Loss probability variance is small or large.**

| | | RTT | | Losses | |
|---|---|---|---|---|---|
| | | small | large | small | large |
| single bottleneck | tethered | X | | X | |
| | wireless | | X | | X |
| multiple bottleneck | tethered | X | | | X |
| | wireless | | X | | X |