

# Staukontrolle durch Active Queue Management

## Teil 2



Thomas Fischer

Betreuer: Martin Metzker  
03.07.2014



# Gliederung

- Einführung und Motivation
- Staukontrolle in Netzen
- Definition und Anwendung von AQM
- Drei Beispiele für AQM Algorithmen
  - RED
  - BLUE
  - AVQ
- Vergleich der vorgestellten Algorithmen
  - BLUE vs. RED
  - AVQ vs. RED
- Zusammenfassung



# Gliederung

- Einführung und Motivation
- Staukontrolle in Netzen
- Definition und Anwendung von AQM
- Drei Beispiele für AQM Algorithmen
  - RED
  - BLUE
  - AVQ
- Vergleich der vorgestellten Algorithmen
  - BLUE vs. RED
  - AVQ vs. RED
- Zusammenfassung



# RED



Random Early Detection, Floyd und Van Jacobson 1993

# RED



Random Early Detection, Floyd und Van Jacobson 1993

Prinzip:

Ankommende Pakete werden mit bestimmter Wahrscheinlichkeit markiert, die sich proportional zum Anteil der Übertragungsrate verhält, welche diese Verbindung belegt.



Random Early Detection, Floyd und Van Jacobson 1993

Prinzip:

Ankommende Pakete werden mit bestimmter Wahrscheinlichkeit markiert, die sich proportional zum Anteil der Übertragungsrate verhält, welche diese Verbindung belegt.

„Markieren“ kann dabei Fallenlassen des Pakets oder setzen des ECN-Bits sein

# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

$$Q_{avg} = (1 - w_q) Q_{avg} + w_q \cdot q$$

, mit Queuelänge  $q$  und Gewicht der Queue  $w_q$



# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

$$Q_{avg} = (1 - w_q) Q_{avg} + w_q \cdot q$$

, mit Queuelänge  $q$  und Gewicht der Queue  $w_q$

Vergleichsparameter  $Q_{min}$  und  $Q_{max}$ :

# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

$$Q_{avg} = (1 - w_q) Q_{avg} + w_q \cdot q$$

, mit Queuelänge  $q$  und Gewicht der Queue  $w_q$

Vergleichsparameter  $Q_{min}$  und  $Q_{max}$ :

- $Q_{min} > Q_{avg}$ : keine Aktion

# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

$$Q_{avg} = (1 - w_q) Q_{avg} + w_q \cdot q, \quad \text{mit Queuelänge } q \text{ und Gewicht der Queue } w_q$$

Vergleichsparameter  $Q_{min}$  und  $Q_{max}$ :

- $Q_{min} > Q_{avg}$ : keine Aktion
- $Q_{min} < Q_{avg} < Q_{max}$ : markieren mit Wahrscheinlichkeit  $p_a$

# RED



Messgröße: durchschnittliche Queuelänge  $Q_{avg}$ :

$$Q_{avg} = (1 - w_q) Q_{avg} + w_q \cdot q \quad , \text{ mit Queuelänge } q \text{ und Gewicht der Queue } w_q$$

Vergleichsparameter  $Q_{min}$  und  $Q_{max}$ :

- $Q_{min} > Q_{avg}$ : keine Aktion
- $Q_{min} < Q_{avg} < Q_{max}$ : markieren mit Wahrscheinlichkeit  $p_a$
- $Q_{avg} > Q_{max}$ : immer markieren

# RED



Markierungswahrscheinlichkeit  $p_b$ :

# RED



Markierungswahrscheinlichkeit  $p_b$ :

$$p_b = \max_b \cdot \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}}, \text{ mit } \max_b, \text{ dem Maximum für } p_b$$

# RED



Markierungswahrscheinlichkeit  $p_b$ :

$$p_b = max_b \cdot \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}}, \text{ mit } max_b, \text{ dem Maximum für } p_b$$

finale Markierungswahrscheinlichkeit  $p_a$ :



Markierungswahrscheinlichkeit  $p_b$ :

$$p_b = \max_b \cdot \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}}, \text{ mit } \max_b, \text{ dem Maximum für } p_b$$

finale Markierungswahrscheinlichkeit  $p_a$ :

$$p_a = \frac{p_b}{1 - z \cdot p_b}, \text{ mit Zähler } z$$



# RED



RED kann auch Bytelänge (anzahl an Bytes eines Pakets) anstatt Queuelänge in Paketen nutzen. Dafür Modifikation von  $p_b$  zu

# RED



RED kann auch Bytelänge (anzahl an Bytes eines Pakets) anstatt Queuelänge in Paketen nutzen. Dafür Modifikation von  $p_b$  zu

$$p_b = p_b \cdot \frac{\text{Paketbytes}}{\text{maximale Paketbytes}}$$

# RED



Algorithmus:

# RED



Algorithmus:

**for** jedes ankommende Paket **do**

Berechne  $Q_{avg}$ ;

**if**  $Q_{min} < Q_{avg} < Q_{max}$  **then**

Berechne  $p_a$ ;

Markiere ankommendes Paket mit Wahrscheinlichkeit  $p_a$ ;

**else if**  $Q_{max} < Q_{avg}$  **then**

Markiere ankommendes Paket

**end**

# BLUE



1999, Feng et.al., University of Michigan mit IBM

# BLUE



1999, Feng et.al., University of Michigan mit IBM

entwickelt, um Schwachstellen von RED zu verbessern:



1999, Feng et.al., University of Michigan mit IBM

entwickelt, um Schwachstellen von RED zu verbessern:

- RED benötigt viele Parameter welche konfiguriert werden müssen



1999, Feng et.al., University of Michigan mit IBM

entwickelt, um Schwachstellen von RED zu verbessern:

- RED benötigt viele Parameter welche konfiguriert werden müssen
- RED funktioniert nur gut, wenn richtig konfiguriert und ausreichend Pufferplatz



# BLUE



1999, Feng et.al., University of Michigan mit IBM

entwickelt, um Schwachstellen von RED zu verbessern:

- RED benötigt viele Parameter welche konfiguriert werden müssen
- RED funktioniert nur gut, wenn richtig konfiguriert und ausreichend Pufferplatz

➡ BLUE als neues Verfahren

# BLUE



Kennt nur eine globale Markierungswahrscheinlichkeit  $p_m$



Kennt nur eine globale Markierungswahrscheinlichkeit  $p_m$

Nutzt Paketverlust und Verbindungsauslastung zur Berechnung von  $p_m$

# BLUE



Kennt nur eine globale Markierungswahrscheinlichkeit  $p_m$

Nutzt Paketverlust und Verbindungsauslastung zur Berechnung von  $p_m$

Kann Pakete Fallen lassen oder ECN-Bit setzen

# BLUE



Ablauf:

# BLUE



Ablauf:

- Jedes ankommende Paket wird mit Wahrscheinlichkeit  $p_m$  markiert



## Ablauf:

- Jedes ankommende Paket wird mit Wahrscheinlichkeit  $p_m$  markiert
- $p_m$  ändert sich auf Basis verloren gegangener Pakete bzw. ungenutzter Verbindungen:



## Ablauf:

- Jedes ankommende Paket wird mit Wahrscheinlichkeit  $p_m$  markiert
- $p_m$  ändert sich auf Basis verloren gegangener Pakete bzw. ungenutzter Verbindungen:
  - Router erfährt, dass Paket verloren:  $p_m \rightarrow p_m + d_1$





## Ablauf:

- Jedes ankommende Paket wird mit Wahrscheinlichkeit  $p_m$  markiert
- $p_m$  ändert sich auf Basis verloren gegangener Pakete bzw. ungenutzter Verbindungen:
  - Router erfährt, dass Paket verloren:  $p_m \rightarrow p_m + d_1$
  - Router erkennt ungenutzte Verbindung:  $p_m \rightarrow p_m - d_2$



## Ablauf:

- Jedes ankommende Paket wird mit Wahrscheinlichkeit  $p_m$  markiert
- $p_m$  ändert sich auf Basis verloren gegangener Pakete bzw. ungenutzter Verbindungen:
  - Router erfährt, dass Paket verloren:  $p_m \rightarrow p_m + d_1$
  - Router erkennt ungenutzte Verbindung:  $p_m \rightarrow p_m - d_2$
- Zusätzlich *freeze\_time*: Zeitintervall, dass zwischen Änderungen an  $p_m$  gewartet werden muss, damit Änderungen wirksam werden können

# BLUE



Algorithmus:



Algorithmus:

**for** jedes ankommende Paket **do**

**if** Paketverlust **&&** (*now* – *last\_update*) < *freeze\_time* **then**

$$p_m = p_m + d_1;$$

$$last\_update = now;$$

**if** Verbindung frei **&&** (*now* – *last\_update*) < *freeze\_time*  
    **then**

$$p_m = p_m - d_2;$$

$$last\_update = now;$$

**end**



# BLUE



Wahl der Parameter:



Wahl der Parameter:

- $d_1$  (Erhöhung von  $p_m$ ) sollte deutlich größer als  $d_2$  (Reduzierung von  $p_m$ ) sein, da auf Staus sehr schnell reagiert werden muss



## Wahl der Parameter:

- $d_1$  (Erhöhung von  $p_m$ ) sollte deutlich größer als  $d_2$  (Reduzierung von  $p_m$ ) sein, da auf Staus sehr schnell reagiert werden muss
- *freeze\_time* wurde von Autoren in Versuchen konstant gehalten; sollte aber zufällig gewählt werden, um globale Synchronisation zu vermeiden

# AVQ

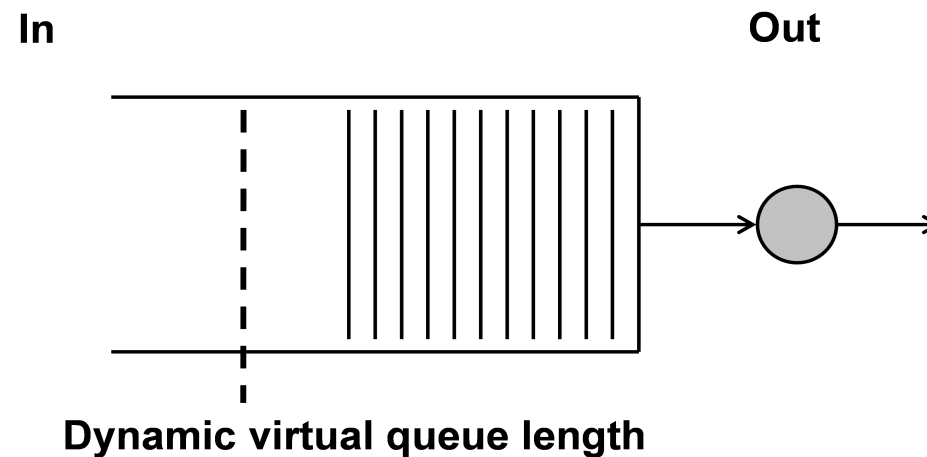


Addaptive Virtual Queue, Kunniyur und Srikant, 2001





## Addaptive Virtual Queue, Kunniyur und Srikant, 2001



Prinzip: nutze virtuelle Queue, deren Größe dynamisch angepasst wird, um bessere Leistungsgrenzen zu erhalten



Keine Markierungswahrscheinlichkeiten; Entscheidung über Markieren wird anhand der Kapazität der virtuellen Queue getroffen



Keine Markierungswahrscheinlichkeiten; Entscheidung über Markieren wird anhand der Kapazität der virtuellen Queue getroffen

Unterstützt Fallenlassen von Paketen und das Setzen des ECN-Bits



Virtuelle Queue mit Kapazität  $C_v \leq C$ ,  $C$  ist Kapazität der tatsächlichen Verbindung, zu Beginn  $C_v = C$



Virtuelle Queue mit Kapazität  $C_v \leq C$ ,  $C$  ist Kapazität der tatsächlichen Verbindung, zu Beginn  $C_v = C$

Überprüfe für ankommende Pakete, ob virtuelle Queue Paket aufnehmen könnte:



Virtuelle Queue mit Kapazität  $C_v \leq C$ ,  $C$  ist Kapazität der tatsächlichen Verbindung, zu Beginn  $C_v = C$

Überprüfe für ankommende Pakete, ob virtuelle Queue Paket aufnehmen könnte:

- Falls ja: Paket in tatsächliche Queue einreihen



Virtuelle Queue mit Kapazität  $C_v \leq C$ ,  $C$  ist Kapazität der tatsächlichen Verbindung, zu Beginn  $C_v = C$

Überprüfe für ankommende Pakete, ob virtuelle Queue Paket aufnehmen könnte:

- Falls ja: Paket in tatsächliche Queue einreihen
- Falls nein: Paket markieren



Kapazität der virtuellen Queue wird bei jedem ankommenden Paket angepasst gemäß

$$\dot{C}_v = \alpha (\gamma \cdot C - \lambda)$$





Kapazität der virtuellen Queue wird bei jedem ankommenden Paket angepasst gemäß

$$\dot{C}_v = \alpha (\gamma \cdot C - \lambda)$$

wobei

- $\alpha$  ein Glättungsparameter



Kapazität der virtuellen Queue wird bei jedem ankommenden Paket angepasst gemäß

$$\dot{C}_v = \alpha (\gamma \cdot C - \lambda)$$

wobei

- $\alpha$  ein Glättungsparameter
- $\gamma$  die angestrebte Auslastung der Verbindung



Kapazität der virtuellen Queue wird bei jedem ankommenden Paket angepasst gemäß

$$\dot{C}_v = \alpha (\gamma \cdot C - \lambda)$$

wobei

- $\alpha$  ein Glättungsparameter
- $\gamma$  die angestrebte Auslastung der Verbindung
- $\lambda$  die Ankunftsrate der Verbindung



Kapazität der virtuellen Queue wird bei jedem ankommenden Paket angepasst gemäß

$$\dot{C}_v = \alpha (\gamma \cdot C - \lambda)$$

wobei

- $\alpha$  ein Glättungsparameter
- $\gamma$  die angestrebte Auslastung der Verbindung
- $\lambda$  die Ankunftsrate der Verbindung

Da keine Pakete in virtuelle Queue eingereiht werden ist lediglich die Kapazität von Interesse

# AVQ



Algorithmus:

# AVQ



Algorithmus:

**for** jedes ankommende Paket **do**

**if**  $VQ = \max(VQ - C_v(t - s), 0)$  **then**

        Paket markieren;

**else**

$VQ = VQ + b$ ;

$C_v = \max(\min(C_v + \alpha \cdot \gamma \cdot C(t - s), C) - \alpha \cdot b, 0)$ ;

$s = t$ ;

**end**

# AVQ



Algorithmus:

**for** jedes ankommende Paket **do**

**if**  $VQ = \max(VQ - C_v(t - s), 0)$  **then**

        Paket markieren;

**else**

$VQ = VQ + b$ ;

$C_v = \max(\min(C_v + \alpha \cdot \gamma \cdot C(t - s), C) - \alpha \cdot b, 0)$ ;

$s = t$ ;

**end**

---

B: Puffergröße, s: Ankunftszeit des letzten Pakets,

t: aktuelle Zeit, b: Paketgröße, VQ: Bytes in virt. Queue



# Gliederung

- Einführung und Motivation
- Staukontrolle in Netzen
- Definition und Anwendung von AQM
- Drei Beispiele für AQM Algorithmen
  - RED
  - BLUE
  - AVQ
- Vergleich der vorgestellten Algorithmen
  - BLUE vs. RED
  - AVQ vs. RED
- Zusammenfassung



# BLUE vs. RED



Von den Autoren von BLUE

# BLUE vs. RED



Von den Autoren von BLUE

Aufbau:

- ECN aktiviert

# BLUE vs. RED



Von den Autoren von BLUE

Aufbau:

- ECN aktiviert
- Messen der Auslastung und Paketverluste nach 100s  
Übertragung + 100s Warten

# BLUE vs. RED



Von den Autoren von BLUE

Aufbau:

- ECN aktiviert
- Messen der Auslastung und Paketverluste nach 100s Übertragung + 100s Warten
- RED:  $Q_{min} = 20\%$ ,  $Q_{max} = 80\%$

# BLUE vs. RED



Von den Autoren von BLUE

Aufbau:

- ECN aktiviert
- Messen der Auslastung und Paketverluste nach 100s Übertragung + 100s Warten
- RED:  $Q_{min} = 20\%$ ,  $Q_{max} = 80\%$
- BLUE:  $d_1 = 10 \cdot d_2$

# BLUE vs. RED



Von den Autoren von BLUE

Aufbau:

- ECN aktiviert
- Messen der Auslastung und Paketverluste nach 100s Übertragung + 100s Warten
- RED:  $Q_{min} = 20\%$ ,  $Q_{max} = 80\%$
- BLUE:  $d_1 = 10 \cdot d_2$
- Variation der Buffergröße von 100 KB bis 1000 KB, entspricht Verzögerung von 17,8 ms bis 178 ms

# BLUE vs. RED



1000 Quellen:

# BLUE vs. RED



1000 Quellen:

Auslastung bei beiden 100%

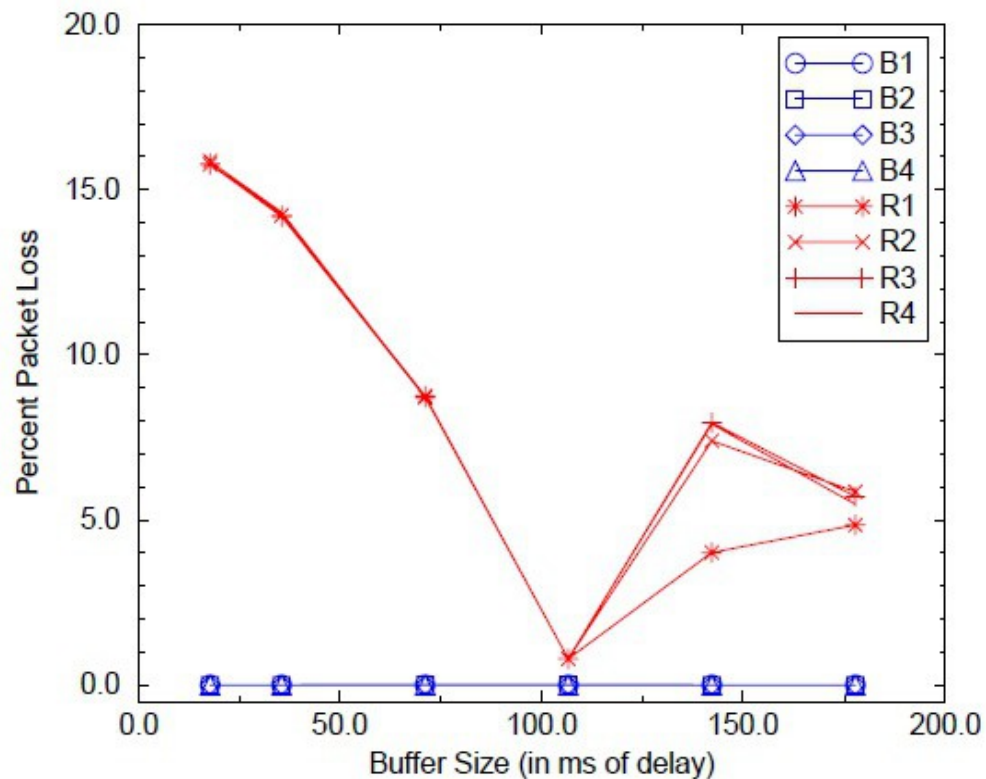


# BLUE vs. RED



1000 Quellen:

Auslastung bei beiden 100%



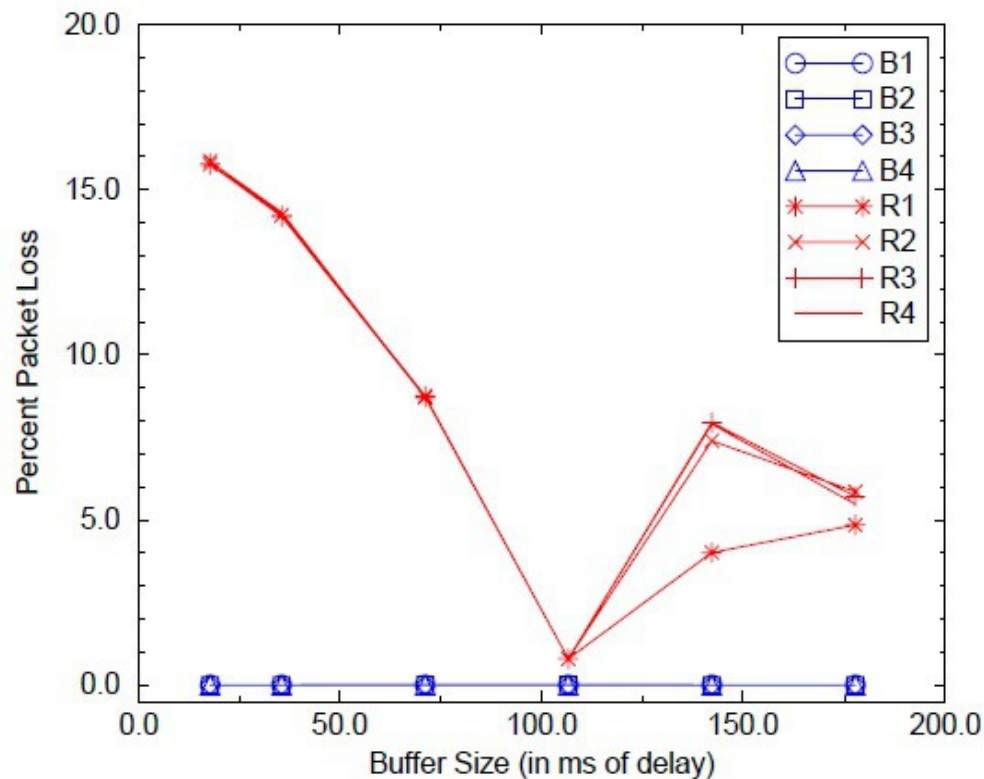
# BLUE vs. RED



1000 Quellen:

4000 Quellen:

Auslastung bei beiden 100%



# BLUE vs. RED

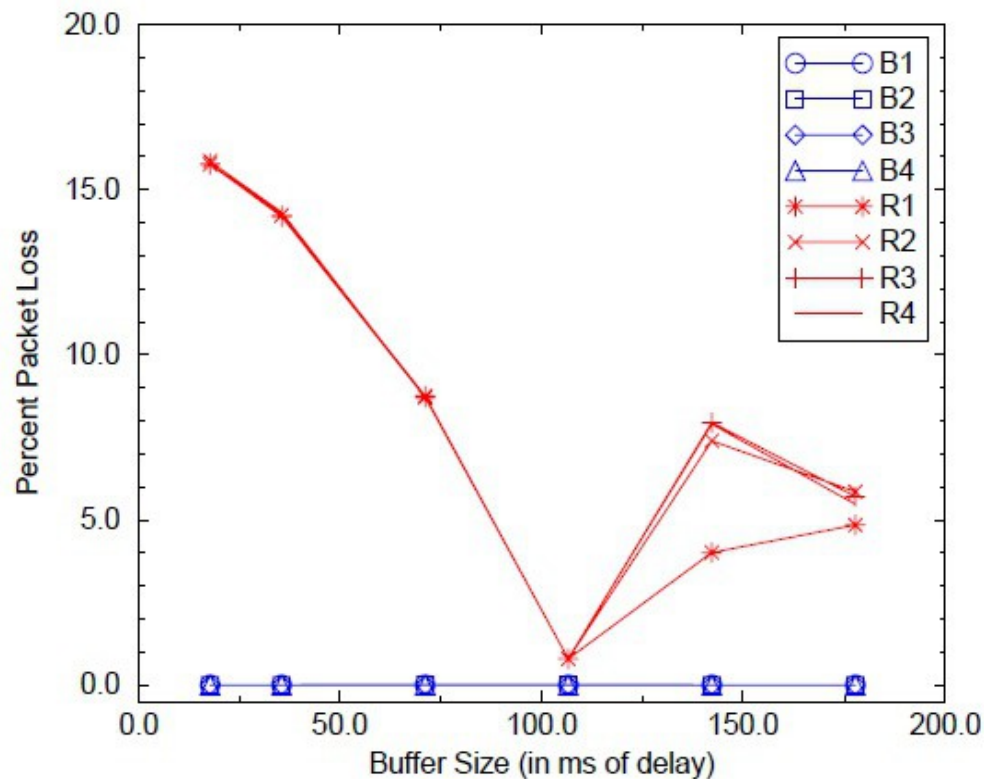


1000 Quellen:

Auslastung bei beiden 100%

4000 Quellen:

Auslastung bei beiden 100%

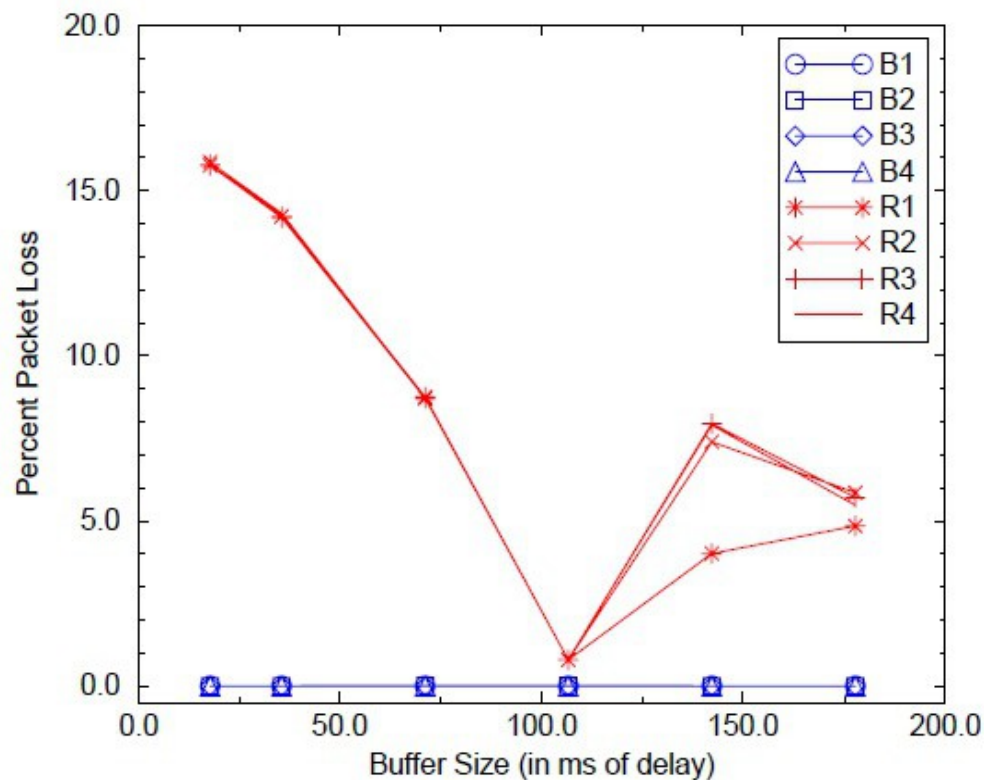


# BLUE vs. RED



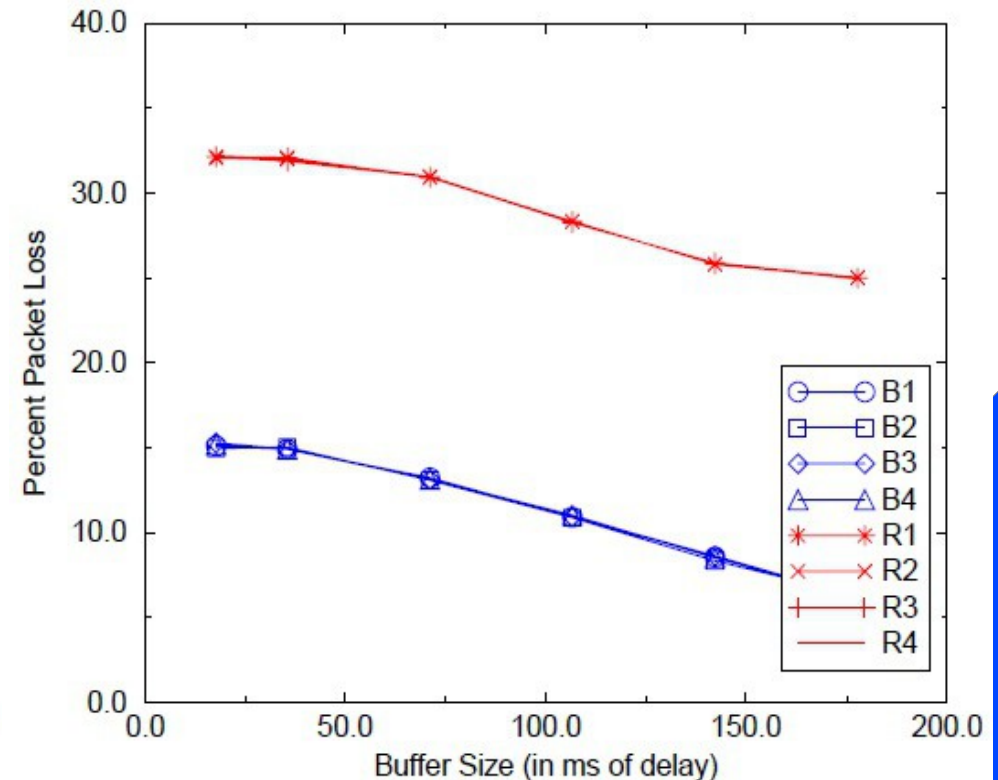
1000 Quellen:

Auslastung bei beiden 100%



4000 Quellen:

Auslastung bei beiden 100%



# AVQ vs. RED



Von den Autoren von AVQ

# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert

# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert
- Messen der Auslastung und Paketverluste nach 30 – 60 ms

# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert
- Messen der Auslastung und Paketverluste nach 30 – 60 ms
- Flaschenhals Queuelänge: 100 Pakete bzw. 1000 bytes



# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert
- Messen der Auslastung und Paketverluste nach 30 – 60 ms
- Flaschenhals Queuelänge: 100 Pakete bzw. 1000 bytes
- RED:  $Q_{min} = 37\%$ ,  $Q_{max} = 75\%$

# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert
- Messen der Auslastung und Paketverluste nach 30 – 60 ms
- Flaschenhals Queuelänge: 100 Pakete bzw. 1000 bytes
- RED:  $Q_{min} = 37\%$ ,  $Q_{max} = 75\%$
- AVQ: A:  $\gamma = 98\%$  B:  $\gamma = 100\%$ ;  $\alpha = 0,15$

# AVQ vs. RED



Von den Autoren von AVQ

Aufbau:

- Versuch A: ECN aktiviert, B: ECN deaktiviert
- Messen der Auslastung und Paketverluste nach 30 – 60 ms
- Flaschenhals Queuelänge: 100 Pakete bzw. 1000 bytes
- RED:  $Q_{min} = 37\%$ ,  $Q_{max} = 75\%$
- AVQ: A:  $\gamma = 98\%$  B:  $\gamma = 100\%$ ;  $\alpha = 0,15$
- A: Variation der FTP Verbindungen von 20 bis 180;  
B: 40 FTP Verbindungen, steigende Anzahl an short-flows

# AVQ vs. RED



A (FTP Variation):

# AVQ vs. RED



A (FTP Variation):

Auslastung RED: 90% - 85%

Auslastung AVQ: 95% - 98%

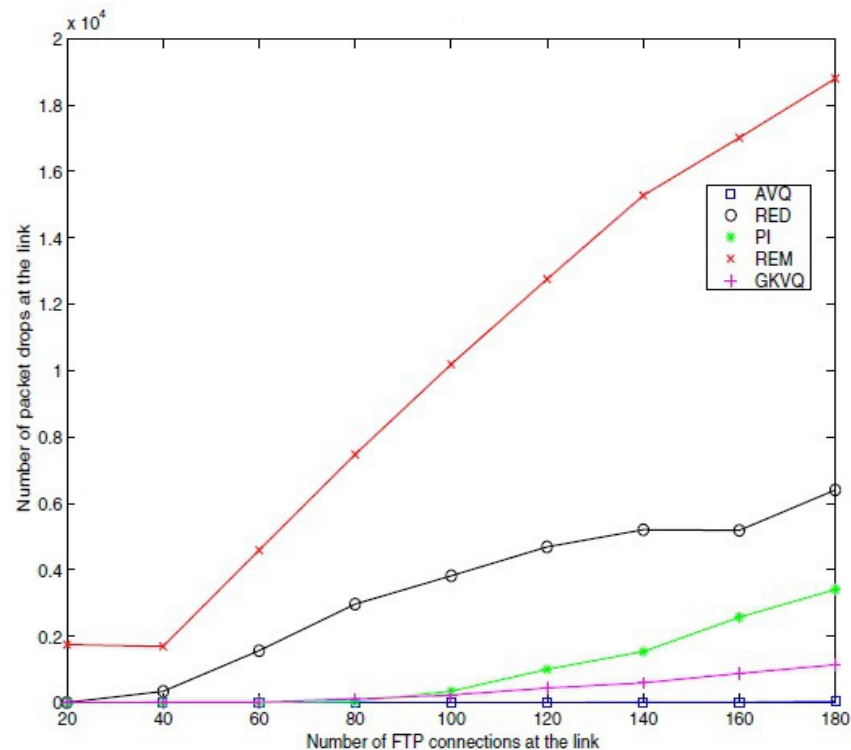
# AVQ vs. RED



A (FTP Variation):

Auslastung RED: 90% - 85%

Auslastung AVQ: 95% - 98%



# AVQ vs. RED

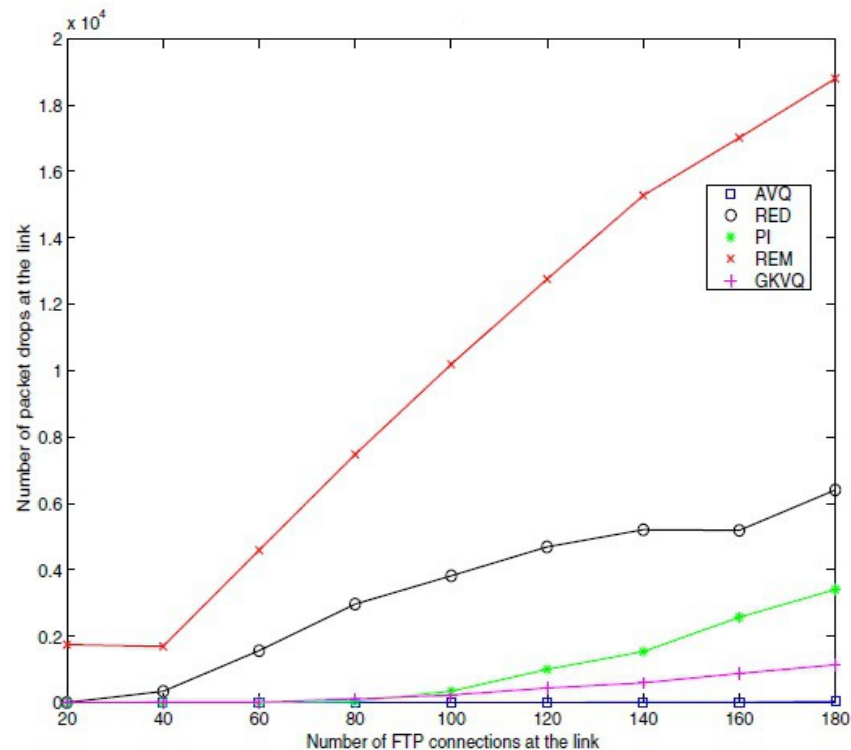


A (FTP Variation):

Auslastung RED: 90% - 85%

Auslastung AVQ: 95% - 98%

B (short flows Variation):



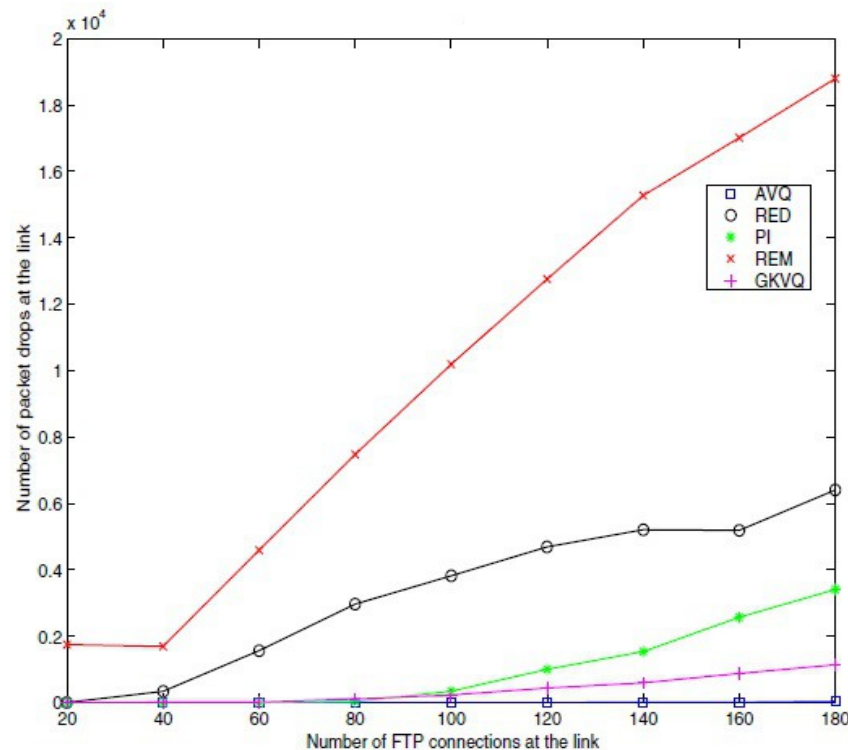
# AVQ vs. RED



## A (FTP Variation):

Auslastung RED: 90% - 85%

Auslastung AVQ: 95% - 98%



## B (short flows Variation):

Auslastung RED: 94% - 99 %

Auslastung AVQ: 100%



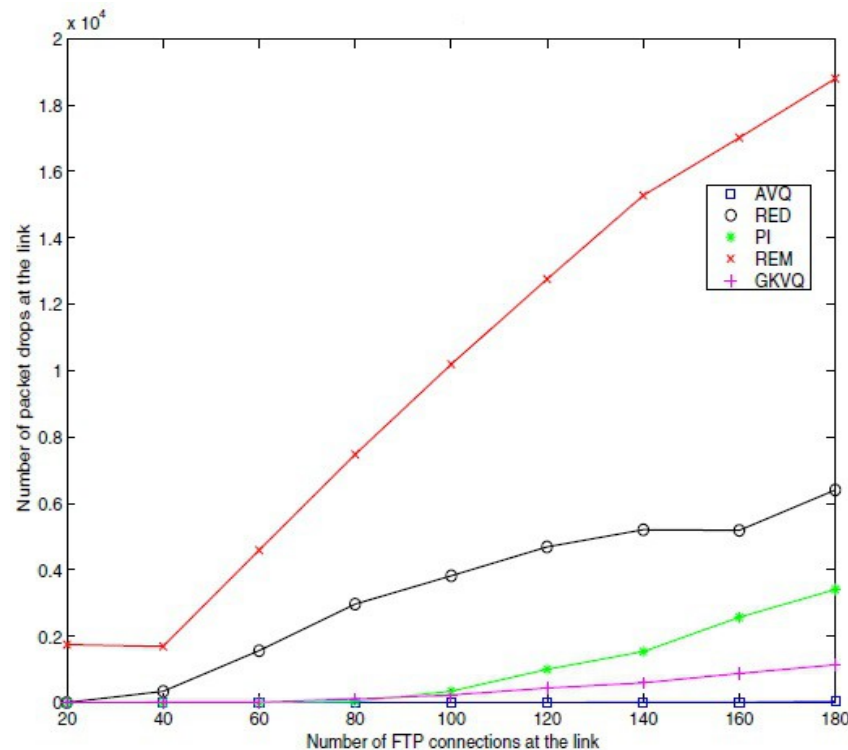
# AVQ vs. RED



## A (FTP Variation):

Auslastung RED: 90% - 85%

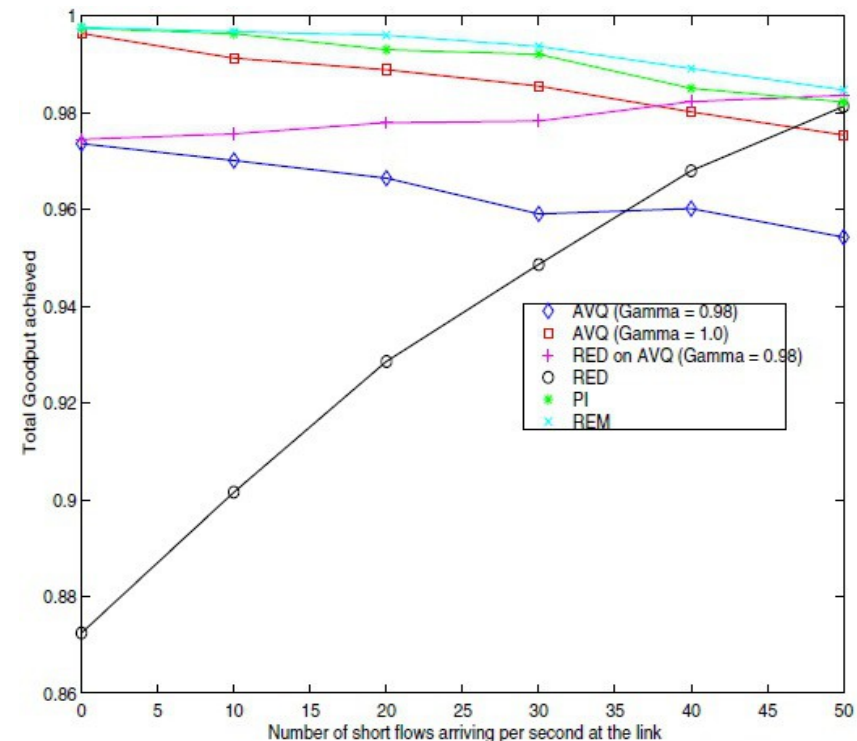
Auslastung AVQ: 95% - 98%



## B (short flows Variation):

Auslastung RED: 94% - 99 %

Auslastung AVQ: 100%





# Gliederung

- Einführung und Motivation
- Staukontrolle in Netzen
- Definition und Anwendung von AQM
- Drei Beispiele für AQM Algorithmen
  - RED
  - BLUE
  - AVQ
- Vergleich der vorgestellten Algorithmen
  - BLUE vs. RED
  - AVQ vs. RED
- Zusammenfassung

# Zusammenfassung



AQM Algorithmen notwendig

# Zusammenfassung



AQM Algorithmen notwendig

Es gibt zahlreiche, weitere Algorithmen

# Zusammenfassung



AQM Algorithmen notwendig

Es gibt zahlreiche, weitere Algorithmen

Wichtig für die Zukunft: Einführung von AQM im Internet auf allen Routern (RED bereits 1998 in RFC 2309 empfohlen, noch vor ECN, welches 1999 in RFC 2481 erwähnt)

# Zusammenfassung



AQM Algorithmen notwendig

Es gibt zahlreiche, weitere Algorithmen

Wichtig für die Zukunft: Einführung von AQM im Internet auf allen Routern (RED bereits 1998 in RFC 2309 empfohlen, noch vor ECN, welches 1999 in RFC 2481 erwähnt)

Alternative Verfahren zur Staukontrolle: z.B. Zugangssteuerung oder Routing unter Verkehrsberücksichtigung

Vielen Dank für Ihre Aufmerksamkeit!



Fragen?