

Analysis and Design of an Adaptive Virtual Queue Algorithm for Congestion Notification in the Internet

Srisankar Kunniyur **R. Srikant**

Coordinated Science Lab and

Department of Electrical and Computer Engineering

University of Illinois at Urbana-Champaign

Roadmap for this presentation

- Goal
- System Model
- Congestion-control algorithms
- Adaptive Virtual Queue Algorithm
 - Analysis in the absence of feedback delays
 - Analysis with feedback delays
 - Comparison with other AQM schemes
- Conclusions

Goal

To design low-loss, low-queueing delay Internet

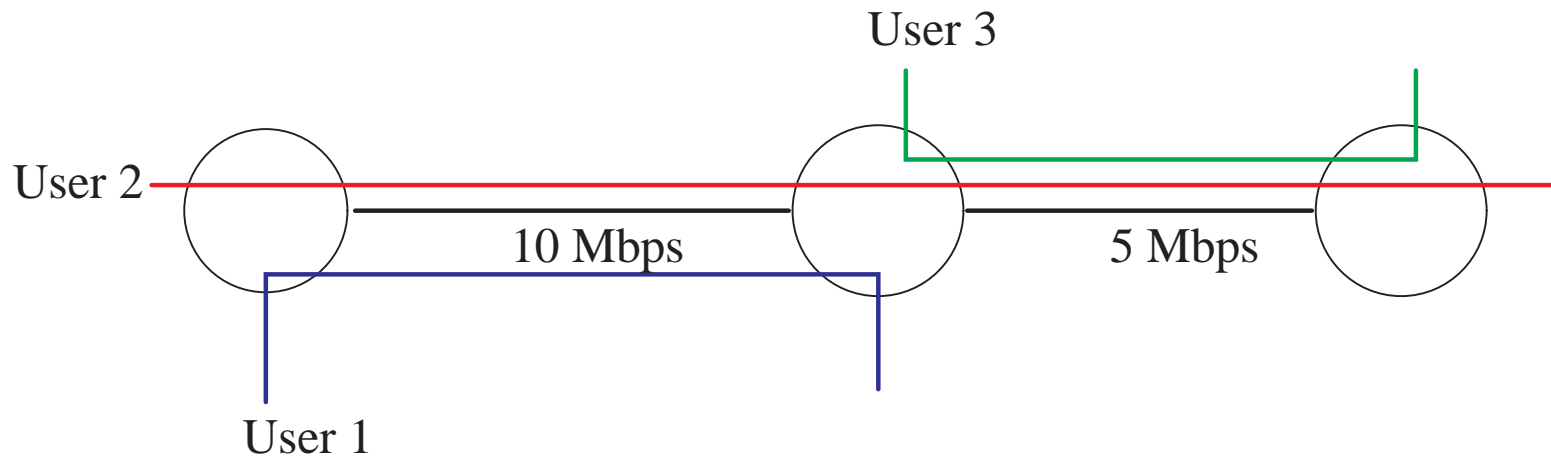
Issues to be addressed:

- Fair Resource Allocation Among Competing Users
- Congestion control (to adapt to varying available bandwidth) using end-end measurable quantities
- **When to signal congestion?**

System Model (Kelly)

- L - set of links l with capacity C_l
- γ_l - desired utilization at the link
- User r - Subset of L
- R - Set of all routes
- $U_r(x_r)$ - Utility of rate x_r to User r

What is a fair allocation of resource?



Rate Allocation 1

User 1 = 5 Mbps

User 2 = 5 Mbps

User 3 = 0 Mbps

Rate Allocation 2

User 1 = 7.5 Mbps

User 2 = 2.5 Mbps

User 3 = 2.5 Mbps

Rate Allocation 3

User 1 = 8 Mbps

User 2 = 2 Mbps

User 3 = 3 Mbps

System Model (Kelly) ... (contd.)

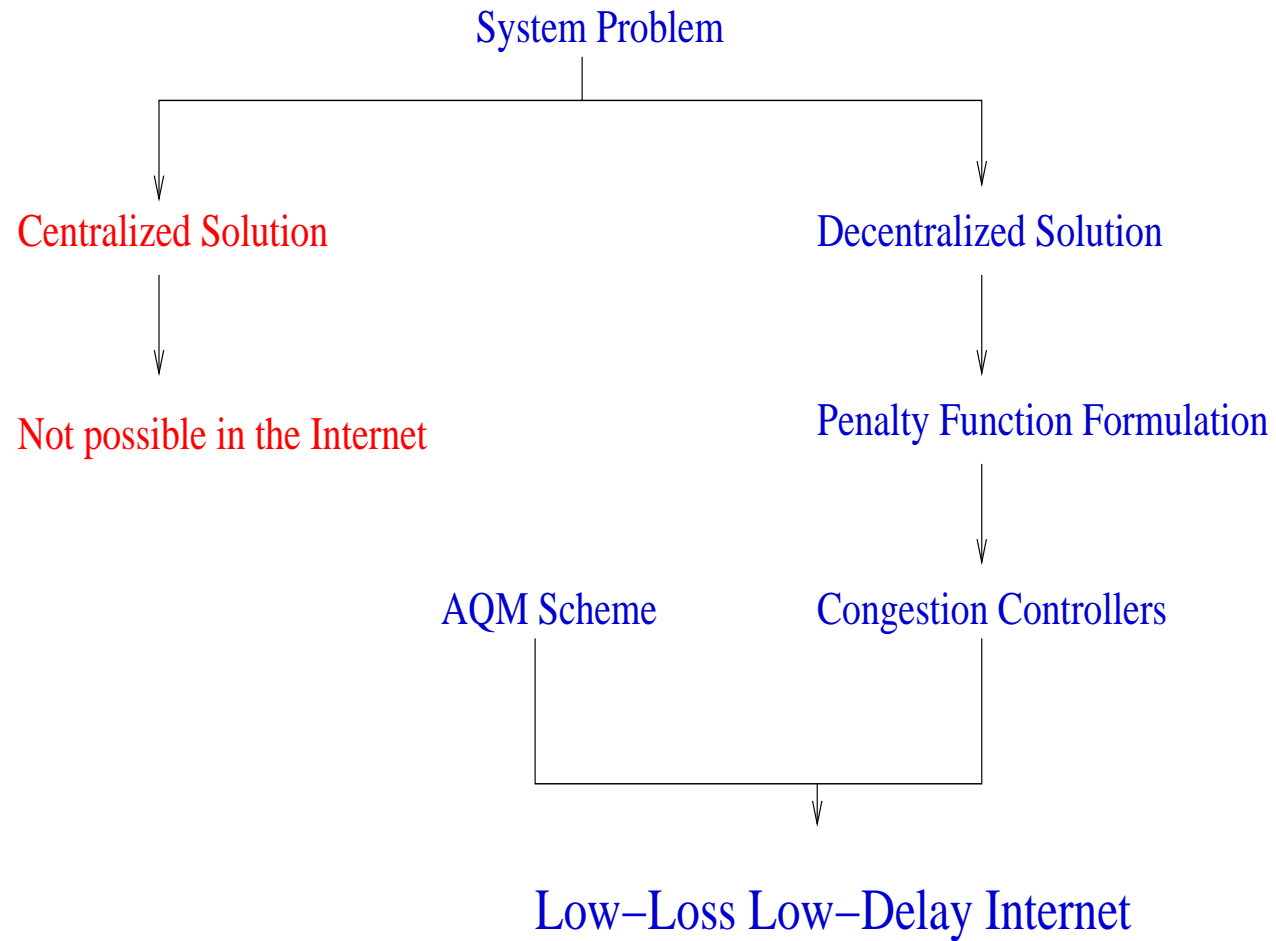
SYSTEM(U, C, γ)

$$\max_x \sum_r U_r(x_r)$$

subject to

$$\begin{aligned} \sum_{r:l \in r} x_r &\leq \gamma_l C_l && \forall l \in L \\ x_r &\geq 0 && \forall r \in R \end{aligned}$$

Maximize aggregate utility subject to capacity and non-negativity constraints



Penalty Function Formulation (Kelly)

- Penalty function approach to the system problem:

$$\max_{\{x_r\}} \sum_r U_r(x_r) - \beta \sum_{l \in L} \int_0^{\sum_{i:l \in i} x_i} p_l(z) dz$$

- $\beta p_l(x)$ is the penalty for exceeding the capacity at link l
- As penalty increases, solution arbitrarily close to the system optimal solution

Congestion-control algorithms (Kelly)

- If $U_r(x_r) = -\Delta_r \frac{1}{x_r}$, the congestion-control scheme for user r is:

$$\frac{dx_r}{dt} = \Delta_r - x_r^2 \beta \sum_{l:l \in r} p_l \left(\sum_{j:l \in j} x_j \right)$$

- RHS is the first derivative of the objective
- p_j 's are the **feedback signals** from each link
- Congestion controllers converge to the solution of the penalty function problem (the objective with the penalty function is a Lyapunov function for this system!)

Penalty Function revisited

- The penalty function approach to the system problem:

$$\max_{\{x_r\}} \sum_r U_r(x_r) - \beta \sum_{l \in L} \int_0^{\sum_{i:l \in i} x_i} p_l(z) dz$$

- p_l 's are the drop/mark probability at link l
- As β increases, solution arbitrarily close to the system optimal solution
- Not possible in practice

Can we achieve the system optimal solution for a finite β ?

Penalty Function revisited ... (contd.)

- Penalty function can also be written as:

$$\max_{\{x_r\}} \sum_r U_r(x_r) - \beta \sum_{l \in L} \int_0^{\sum_{i:l \in i} x_i} p_l(z, \tilde{C}_l) dz$$

\tilde{C}_l can be thought of as a parameter that determines when congestion feedback is provided

Example: \tilde{C}_l can be thought of as a virtual capacity:

$$p_l(z, \tilde{C}_l) = \left(\frac{z - \tilde{C}_l}{z} \right)^+.$$

Goal: Design the parameters $\{\tilde{C}_l\}$ such that the penalty function formulation solves $\text{SYSTEM}(U, C, \gamma)$

Estimating $\{\tilde{C}_l\}$: Adaptive Algorithm

How to estimate $\{\tilde{C}_l\}$ in a heterogeneous network?

- Update \tilde{C}_l at each link as follows:

$$\begin{aligned}\dot{\tilde{C}}_l &= \alpha(\gamma_l C_l - \lambda_l) & \forall l \in L \\ \lambda_l &= \sum_{j:l \in j} x_j & \forall l \in L\end{aligned}$$

where $0 < \alpha < 1$.

- Entire system can be written as:
 - Congestion-controllers at the sources:

$$\dot{x}_r = 1 - \beta U_r'^{-1}(x_r) \sum_{l:l \in r} p_l(\lambda, \tilde{C}_l) \quad \forall r \in R$$

- Adaptation of the virtual capacities \tilde{C}_l at the links:

$$\dot{\tilde{C}}_l = \alpha(\gamma_l C_l - \lambda_l) \quad \forall l \in L$$

Convergence

Theorem 1 For α sufficiently small, the system given by:

$$\begin{aligned}\dot{x}_r &= 1 - \beta U_r'^{-1}(x_r) \sum_{l:l \in r} p_l(\lambda, \tilde{C}_l) \quad \forall r \in R \\ \dot{\tilde{C}}_l &= \alpha \frac{1}{\left| \frac{\partial p_l(\lambda_l, \tilde{C}_l)}{\partial \tilde{C}_l} \right|} (\gamma_l C_l - \lambda_l) \quad \forall l \in L\end{aligned}$$

converges to the solution of the system problem $SYSTEM(U, C, \gamma)$

Proof: Two time scale-decomposition:

View the system as a fast time-scale subsystem and a slow time-scale subsystem.

Congestion Controllers at the sources - Fast time-scale subsystem

Adaptation of \tilde{C} at the links - Slow time-scale subsystem

Fast Time-Scale

- The fast time-scale subsystem is given by:

$$\dot{x}_r = 1 - \beta U_r'^{-1}(x_r) \sum_{l:l \in r} p_l \left(\sum_{j:l \in j} x_j, \tilde{C}_l \right) \quad \forall r \in R$$

where $\{\tilde{C}_l\}$ are treated as constants.

- Asymptotically stable from Kelly et al.

Slow Time-Scale

- The slow time-scale system is given by:

$$\begin{aligned} \dot{\tilde{C}}_l &= \alpha \frac{1}{|\partial p_l(\lambda_l^*, \tilde{C}_l) / \partial \tilde{C}_l|} (\gamma_l C_l - \sum_{j:l \in j} x_j^*) \quad \forall l \in L \\ x^* &= \arg \max \left[\sum_r U_r(x_r) - \beta \sum_{l \in L} \int_0^{\sum_{j:l \in j} x_j} p_l(z, \tilde{C}_l) dz \right] \end{aligned}$$

- Assumes that when α is small, fast system converges rapidly
- Can be shown to be stable using the Lyapunov function

$$W(\tilde{C}) = \sum_{l \in L} (\gamma_l C_l - \lambda_l)^2$$

Stability of the overall system

- The fast time-scale subsystem is exponentially stable uniformly in $\{\tilde{C}_l\}$
- The slow time-scale subsystem is also exponentially stable
- From singular perturbations theory, the entire system is semi-globally exponentially stable

AVQ Algorithm

Model

- Assume that the marking probability is given by:

$$p(\lambda, \tilde{C}) = \left(\frac{\lambda - \tilde{C}}{\lambda} \right)^+$$

- Mark packets when the arrival rate is greater than \tilde{C}

The AVQ Algorithm

- Maintain a virtual queue with capacity \tilde{C} and buffer size B
- Upon each arrival, add a fictitious packet to the VQ
- If VQ overflows, discard the fictitious packet and mark/drop a packet in the real queue

AVQ Algorithm ... (contd.)

Model

- Update \tilde{C} as:

$$\dot{\tilde{C}} = \alpha(\gamma C - \lambda).$$

The AVQ Algorithm

- Discretizing the update equation we get:

$$\frac{\tilde{C}[k + \delta] - \tilde{C}[k]}{\delta} = \alpha\gamma C - \alpha\lambda[k]$$

- If δ is the packet inter-arrival time, then $\lambda[k]\delta = b$, where b is the size of the packet measured in packets or bytes or bits.
- Therefore, upon each arrival, update the virtual capacity (\tilde{C}) in the AVQ algorithm as:

$$\tilde{C}[k + 1] = \tilde{C}[k] + \alpha\gamma C\delta - \alpha b$$

- Can be easily implemented using a token-bucket

- b - number of bytes in current packet
- s - arrival time of previous packet
- VQ - Number of bytes currently in the virtual queue
- B - buffer size
- t - Current time

Pseudo-code for the AVQ Algorithm

At each packet arrival epoch do

$VQ \leftarrow \max(VQ - \tilde{C}(t - s), 0)$ /* Update Virtual Queue Size */

If $VQ + b > B$

 Mark or drop packet in the real queue

else

$VQ \leftarrow VQ + b$ /* Update Virtual Queue Size */

endif

$\tilde{C} = \max(\min(\tilde{C} + \alpha * \gamma * C(t - s), C) - \alpha * b, 0)$ /* Update Virtual Capacity */

$s \leftarrow t$ /* Update last packet arrival time */

Features of the AVQ Scheme

- Implementation complexity of the AVQ scheme is comparable to other well-known AQM schemes. No random number generation is required
- AVQ is a primarily rate-based marking scheme
- AVQ regulates utilization instead of queue length. This is more robust in the presence of extremely short flows or variability in the number of long-flows in the network.
- Two parameters that need to be chosen: γ and α

How to choose α to ensure stability

- Consider a single link
- N TCP users with fixed round-trip propagation delay d (Utility function $-\frac{1}{d^2x}$)
- Update equation at the link is:

$$\dot{\tilde{C}} = \alpha(\gamma C - \sum_{j \in R} x_j(t)) \quad \forall l \in L$$

- The delay-differential equations describing the system are given by:

$$\dot{x}_r = \frac{1}{d^2} - \beta x_r(t)x_r(t-d)p\left(\sum_{j \in R} x_j(t-d), \tilde{C}(t-d)\right) \quad \forall r$$

$$\dot{\tilde{C}} = \alpha(\gamma C - \sum_{j \in R} x_j(t))$$

$$\lambda_l(t) = \sum_{j \in R} x_j(t)$$

- For analytical tractability, we assume that

$$p(\lambda, \tilde{C}) = \frac{(\lambda - \tilde{C})^+}{\lambda}.$$

Linearized Model

- Equilibrium point of the system is given by:

$$\sum_i x_i^* = \lambda^* = \gamma C$$

$$x_i^* = \frac{\gamma C}{N}$$

$$p(\gamma C, \tilde{C}^*) = \frac{N^2}{\beta(d\gamma C)^2}$$

- The linearized model can be written as:

$$\dot{\delta\lambda} = -K_{11}\delta\lambda(t) - K_{12}\delta\lambda(t-d) + K_2\delta\tilde{C}(t-d)$$

$$\dot{\delta\tilde{C}} = -\alpha\delta\lambda(t),$$

where

$$K_{11} := \frac{N}{\gamma C d^2} \quad K_{12} = K_2 = \beta \frac{\gamma C}{N}$$

Laplace Transform of the Linearized Model

- $\Lambda(s)$ - Laplace-Transform of $\delta\lambda(t)$
- $\Psi(s)$ - Laplace-transform of $\delta\tilde{C}(t)$
- Laplace-transforms of the linearized model yields:

$$\begin{aligned} s\Lambda(s) &= -K_{11}\Lambda(s) - K_{12}e^{-sd}\Lambda(s) + K_2e^{-sd}\Psi(s) \\ s\Psi(s) &= -\alpha\Lambda(s). \end{aligned}$$

- Can be rewritten as:

$$\Lambda(s) \left[s + K_{11} + e^{-sd} \left(K_{12} + \alpha \frac{K_2}{s} \right) \right] = 0.$$

Condition for Stability

- For stability, all the roots of the equation given by:

$$\left[s + K_{11} + e^{-sd} \left(K_{12} + \alpha \frac{K_2}{s} \right) \right] = 0$$

should have negative real parts.

- The characteristic equation can be rewritten as:

$$1 + \frac{e^{-sd} \left(K_{12} + \alpha \frac{K_2}{s} \right)}{s + K_{11}} = 0$$

- When $d = 0$, roots are:

$$s = \frac{-(K_{11} + K_{12}) \pm \sqrt{(K_{11} + K_{12})^2 - 4\alpha K_2}}{2}$$

- Therefore, with $d = 0$, the system is stable for all $\alpha > 0$.

Key Idea

- System is stable for $d = 0$ for all $\alpha > 0$,
- Fix any $\alpha = \hat{\alpha}$.
- Since system is stable, all roots lie in the left-half plane.
- Roots are continuous functions of d
- Therefore, as d increases, the roots also change continuously
- Find smallest d (say \hat{d}) when one of the roots hit the imaginary axis
- For all $d < \hat{d}$ the roots lie in the left-half plane which implies stability

Necessary and sufficient condition for stability

Theorem 2 Fix $\alpha = \hat{\alpha}$, the number of TCP users, N and the utilization γ . Find the smallest $d = \hat{d}$ such that

$$\omega(\hat{\alpha}, d, N, \gamma) = \frac{1}{\sqrt{2}} \sqrt{(K_{12}^2 - K_{11}^2) + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4K_2^2 \hat{\alpha}^2}}$$

satisfies

$$\omega d + \arctan\left(\frac{\hat{\alpha}}{\omega}\right) + \arctan\left(\frac{\omega}{K_{11}}\right) = (2k + 1)\pi,$$

for some $k = 0, 1, 2, \dots$. Then, the TCP/AQM system is locally asymptotically stable for all values of $d < \hat{d}$.

Note:

- Not easy to solve for smallest d for a given k
- Need to check for $k = 0, 1, 2, \dots$ to find smallest d
- However, $\arctan\left(\frac{\hat{\alpha}}{\omega}\right) \leq \frac{\pi}{2}$
- Leads to easily verifiable sufficient condition

Sufficient condition for stability given α , N and γ

Theorem 3 Fix $\alpha = \hat{\alpha}$, the number of TCP users, N and the utilization γ . Find any $d^* > 0$, such that

$$\omega d^* + \arctan\left(\frac{\omega}{K_{11}}\right) = \frac{\pi}{2},$$

where ω is given by:

$$\omega(\hat{\alpha}, d, N, \gamma) = \frac{1}{\sqrt{2}} \sqrt{(K_{12}^2 - K_{11}^2) + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4K_2^2 \hat{\alpha}^2}}$$

Then for all $d < d^*$ the system is stable. Moreover, d^* is unique.

Remarks:

- Finds \hat{d} , given α , N and γ
- More practical situation is to find α , given d , N , and γ .
- Can use the same condition to find $\hat{\alpha}$ such that for all $\alpha < \hat{\alpha}$ the system is stable.

Sufficient condition for stability given d , N and γ

Theorem 4 Fix $d = \hat{d}$, the number of TCP users, N and the utilization γ . Find any $\hat{\alpha} > 0$, such that

$$\omega \hat{d} + \arctan\left(\frac{\omega}{K_{11}}\right) = \frac{\pi}{2},$$

where ω is given by:

$$\omega(\alpha, \hat{d}, N, \gamma) = \frac{1}{\sqrt{2}} \sqrt{(K_{12}^2 - K_{11}^2) + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4K_2^2 \alpha^2}}$$

Then for all $\alpha < \hat{\alpha}$ the system is stable.

Remarks:

- Condition remains the same
- More generally, one can fix any three of the four parameters (N , γ , d and α) and use the above equation to find a bound on the fourth parameter to ensure stability
- Given α , d and γ one can find \hat{N} such that for all $N > \hat{N}$, the system is stable
- Given α , d and N one can find $\hat{\gamma}$ such that for all $\gamma < \hat{\gamma}$, the system is stable

Simulations

Simulation Setup

- Single link with capacity 10 Mbps
- TCP reno users with average packet length of 1000 bytes
- Propagation delay of each user between 40ms and 130 ms
- Buffer capacity at the link 100 packets

Experiment 1

- Study the convergence properties and buffer sizes at the link for the AVQ scheme
- Number of long FTP connections = 180
- Short flows (20 packets each) arrive at the rate of 30 flows per second

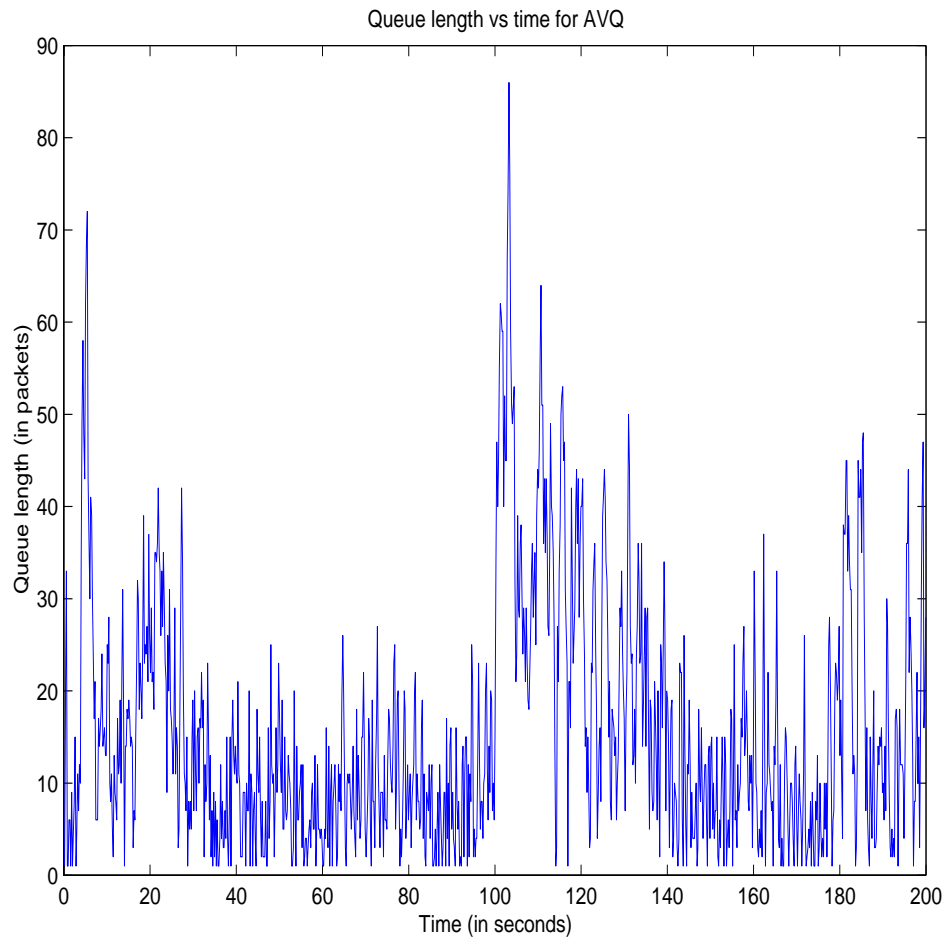


Fig. 1: Queue length vs time for the AVQ scheme

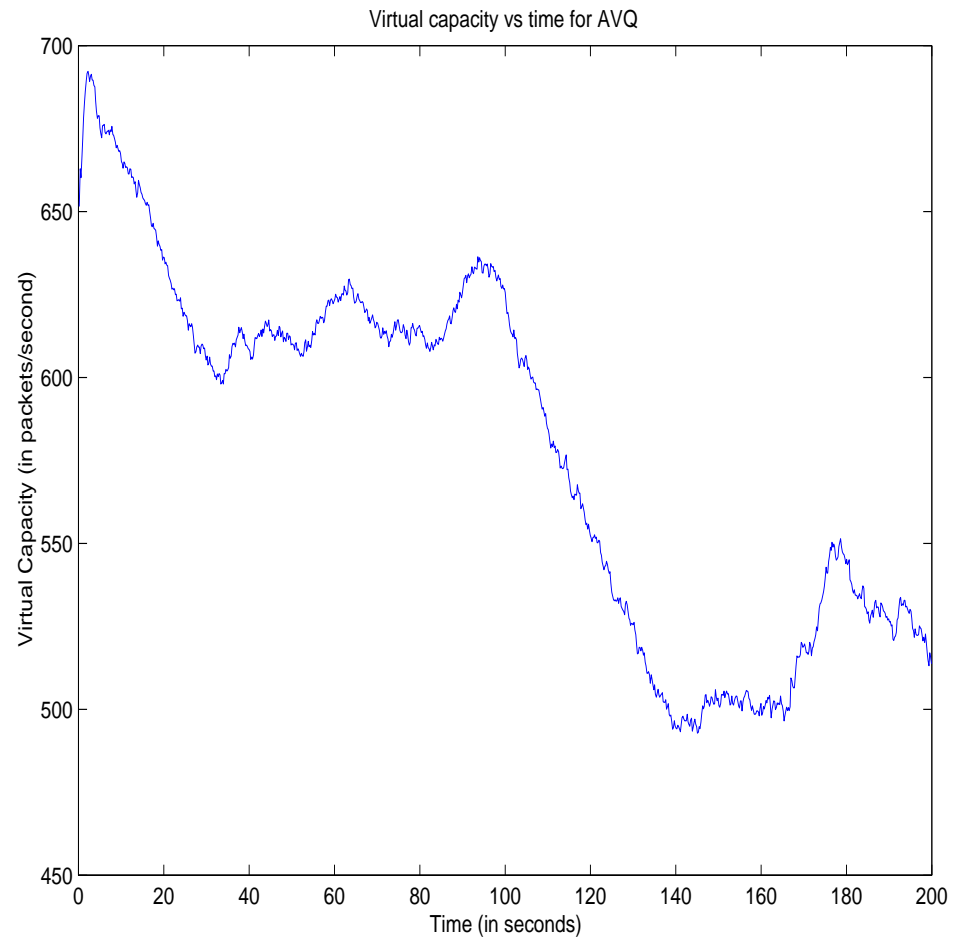
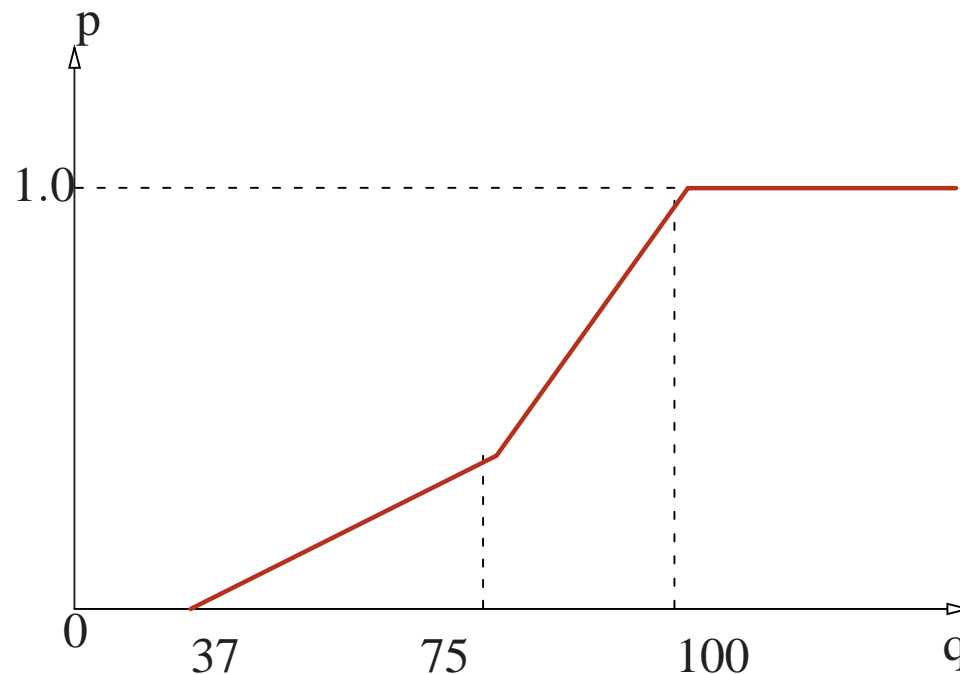


Fig. 2: Virtual capacity vs time for the AVQ scheme

Comparison of the AVQ scheme with other AQM Schemes

- Single link with capacity 10 Mbps
- Buffer capacity at the link is 100 packets
- Require queueing delay to be between 30ms and 60 ms
- For AVQ: $\gamma = 0.98$ and buffer capacity is 50 packets

RED (proposed by V. Jacobson and S. Floyd)



REM (proposed by Low et al.)

- Attempts to regulate the queue length to a desired value (denoted by $qref$) by adapting the marking probability
- Mark each packet with probability $1 - \phi^{-\mu[k]}$ where ϕ is fixed positive constant greater than one and $\mu[k]$ is updated periodically as:

$$\mu[k + 1] = \max(0, \mu[k] + \gamma(q[k + 1] - (1 - \alpha)q[k] - \alpha qref)),$$

where α and γ are positive constants and $qref$ is the desired queue length.

PI (proposed by Hollot, Misra, Towsley and Gong)

- Attempts to regulate the queue length to a desired value (denoted by $qref$) by adapting the marking probability p
- Mark each packet with a probability p
- Update the marking probability periodically as:

$$p[k + 1] = p[k] + a(q[k + 1] - qref) - b(q[k] - qref),$$

where $a > 0$ and $b > 0$ are constants

GKVQ (proposed by Gibbens and Kelly)

- Maintain a virtual queue with a fixed capacity $\tilde{C} = \Theta C$ and fixed buffer size $\tilde{B} = \Theta B$, where $\Theta < 1$.
- On packet arrival, enqueue a fictitious arrival in the VQ
- When VQ overflows, mark all packets in the real queue and also mark all future incoming packets till VQ empties

Experiment 2

- Performance of the AQM schemes in the presence of long-lived FTP connections
- Packet losses, link utilization and responsiveness to the network conditions are compared

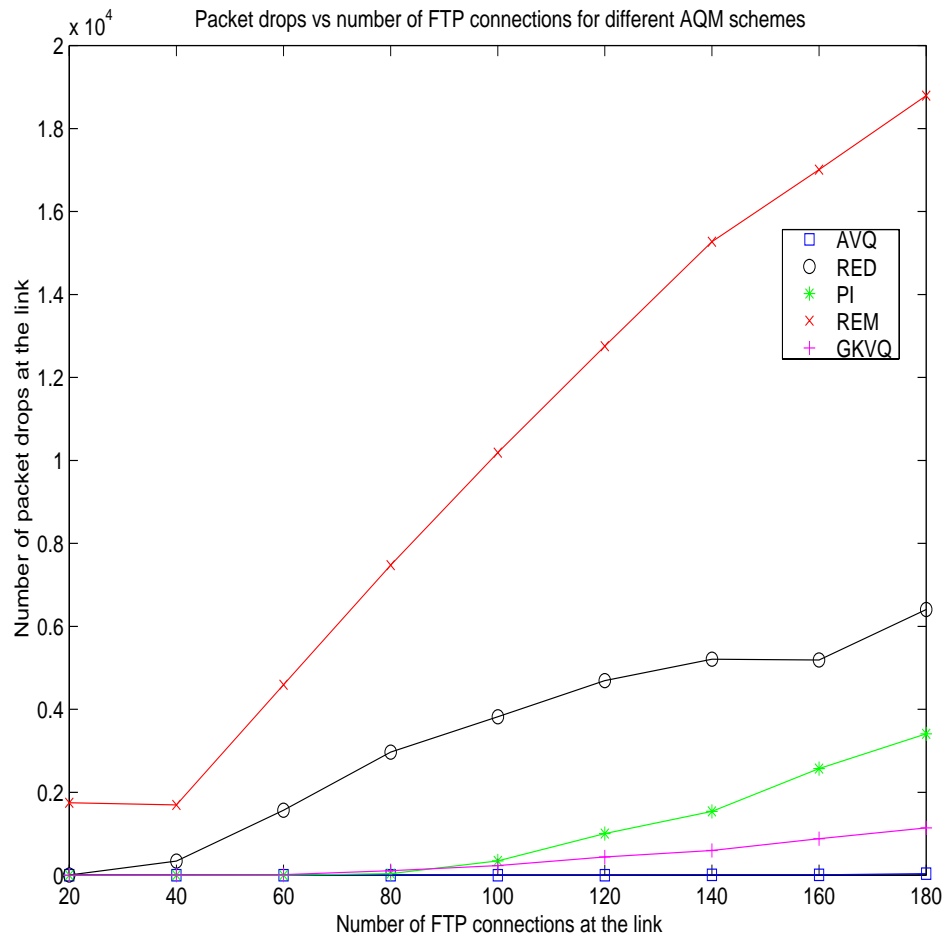


Fig. 3: Losses at the link for varying number of FTP connections for the different AQM schemes

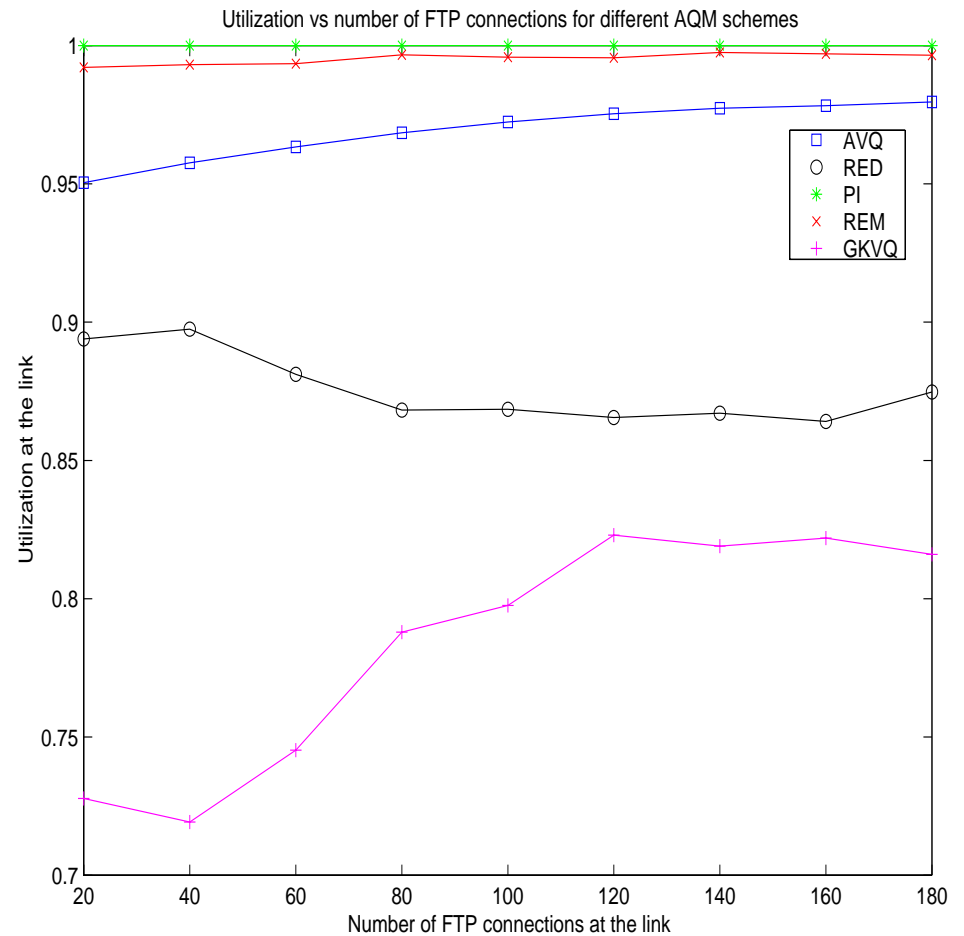


Fig. 4: Achieved Utilization at the link for the different AQM schemes

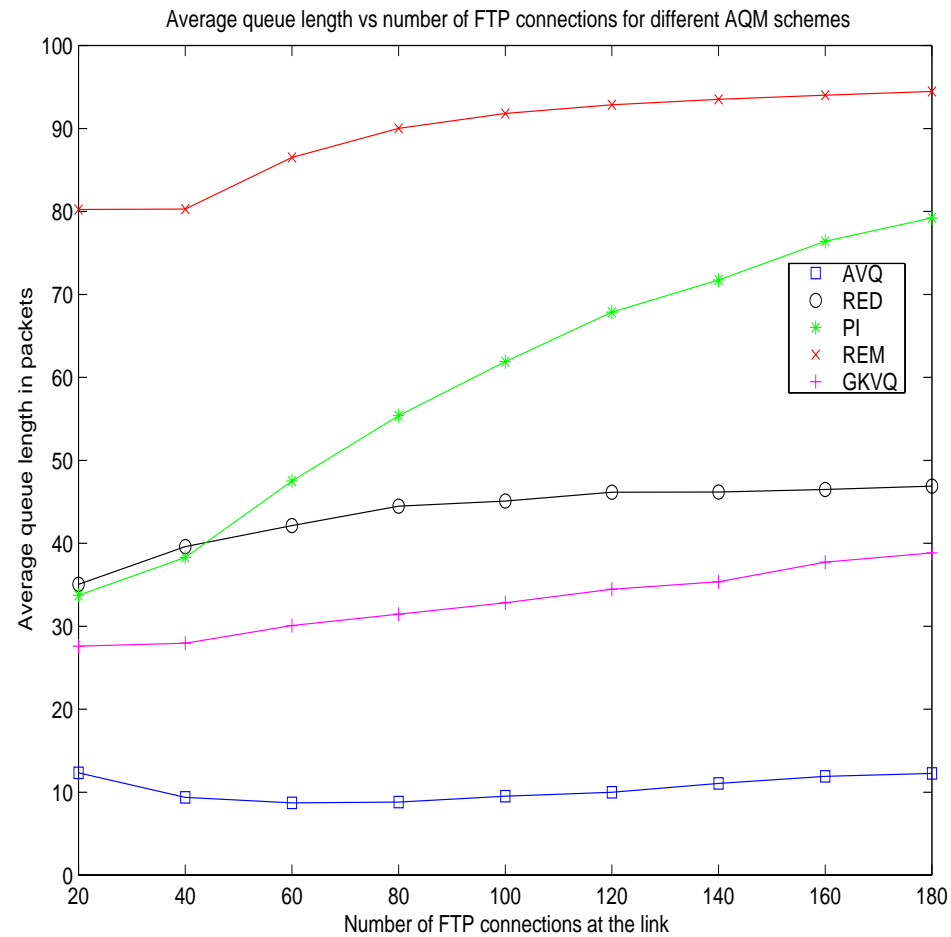


Fig. 5: Average Queue length at the link for varying number of FTP connections for the different AQM schemes

Experiment 3

- Compare the responsiveness of the AQM schemes when flows are dropped and then introduced later on
- Compare only REM and PI with the AVQ scheme
- Number of FTP connections is 140 at time $t = 0$
- At time $t = 100$, 105 FTP connections are dropped
- At time $t = 150$, a new set of 150 FTP connections are added
- Study the evolution of the queue lengths

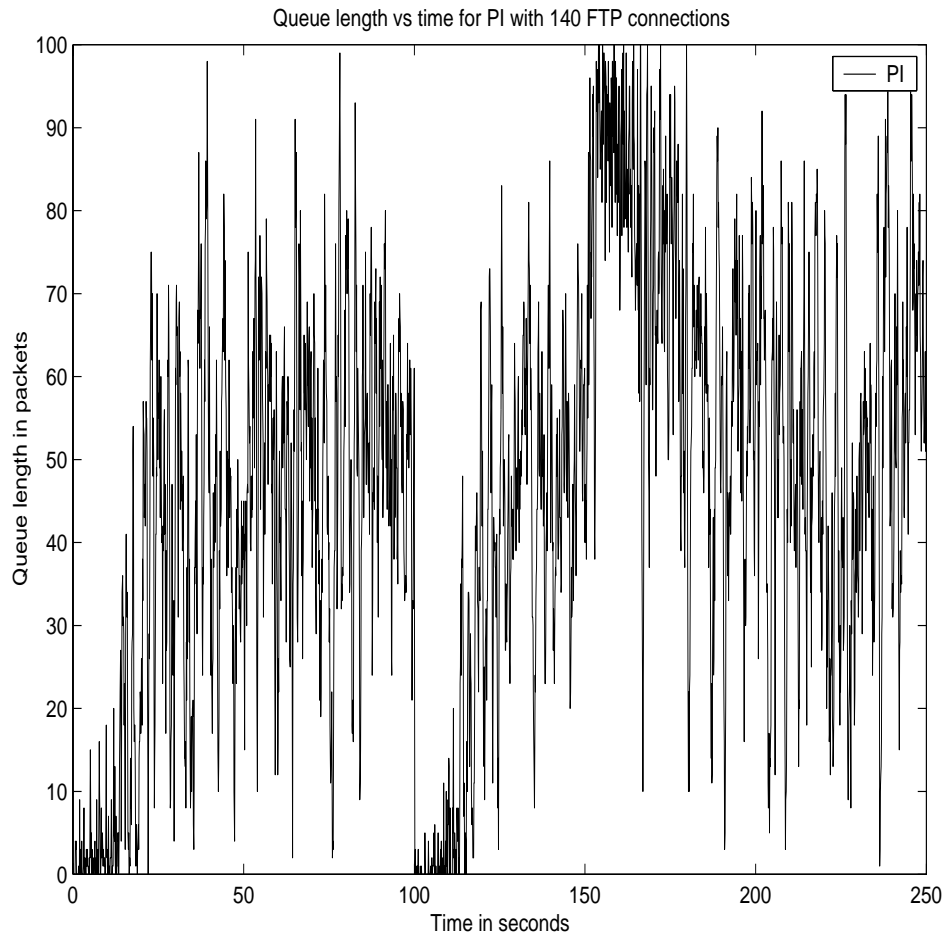


Fig. 6: Evolution of the queue length at varying loads for PI

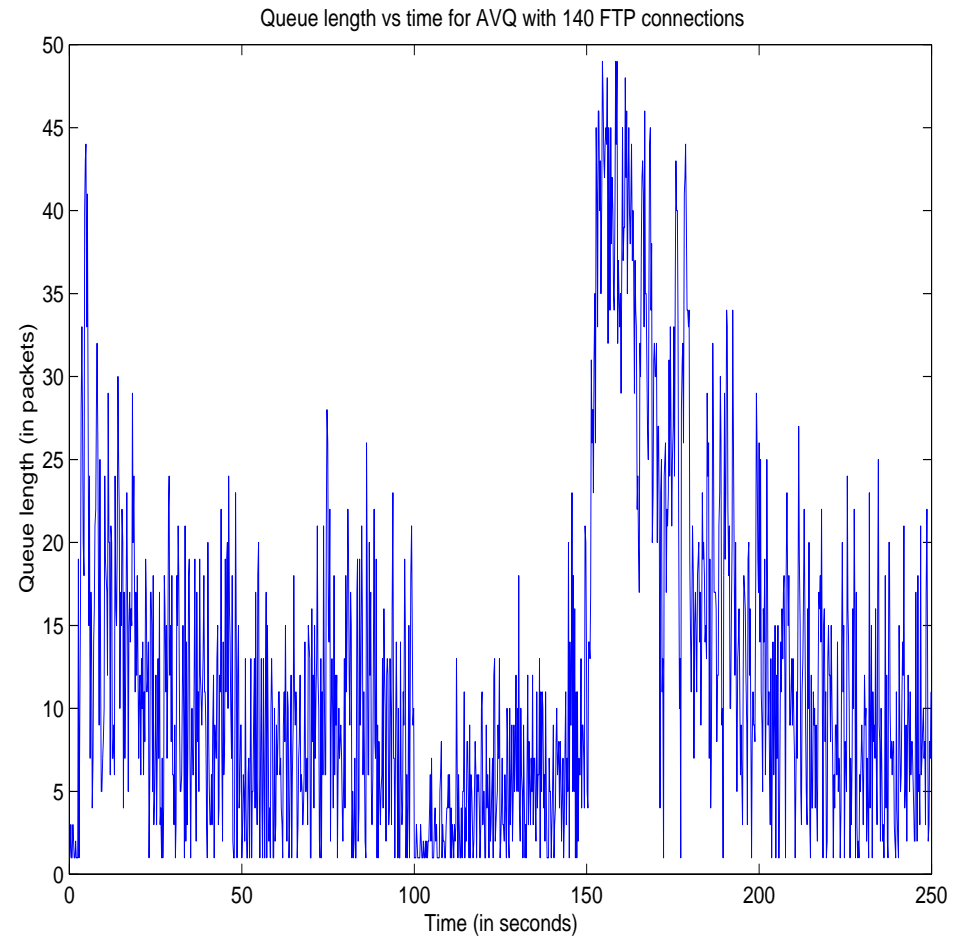


Fig. 7: Evolution of the queue sizes at varying loads for AVQ

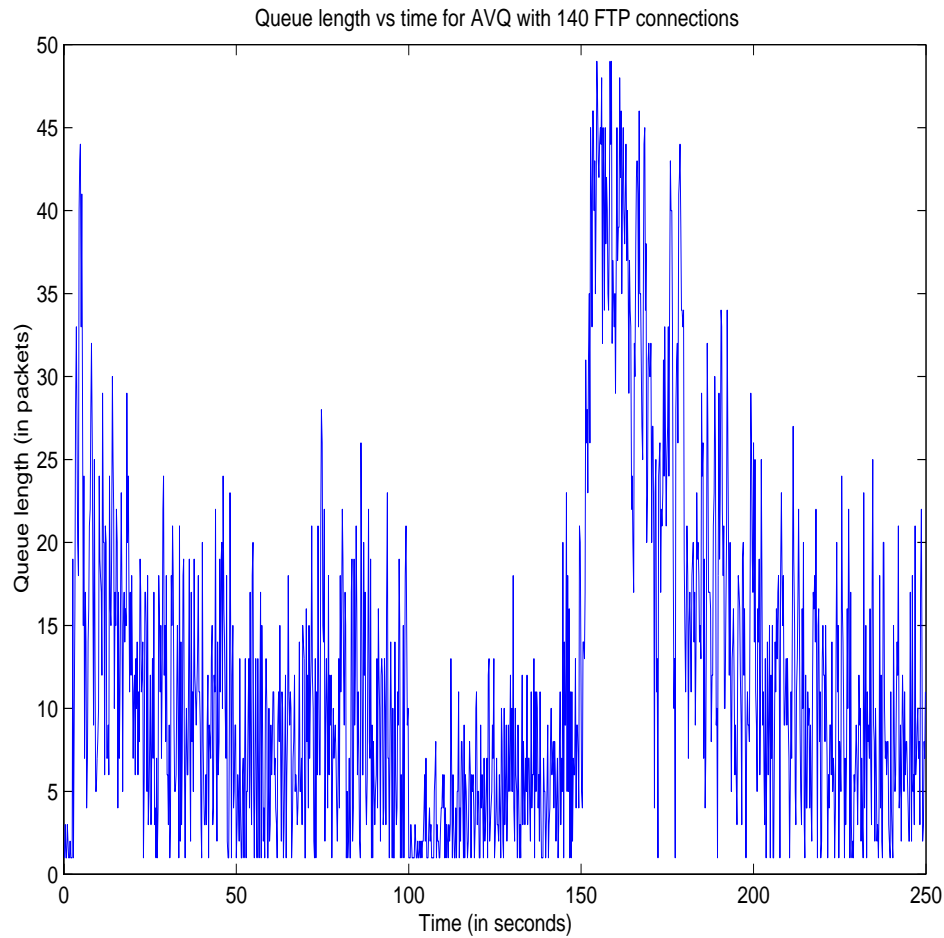


Fig. 8: Evolution of the queue sizes at varying loads for AVQ

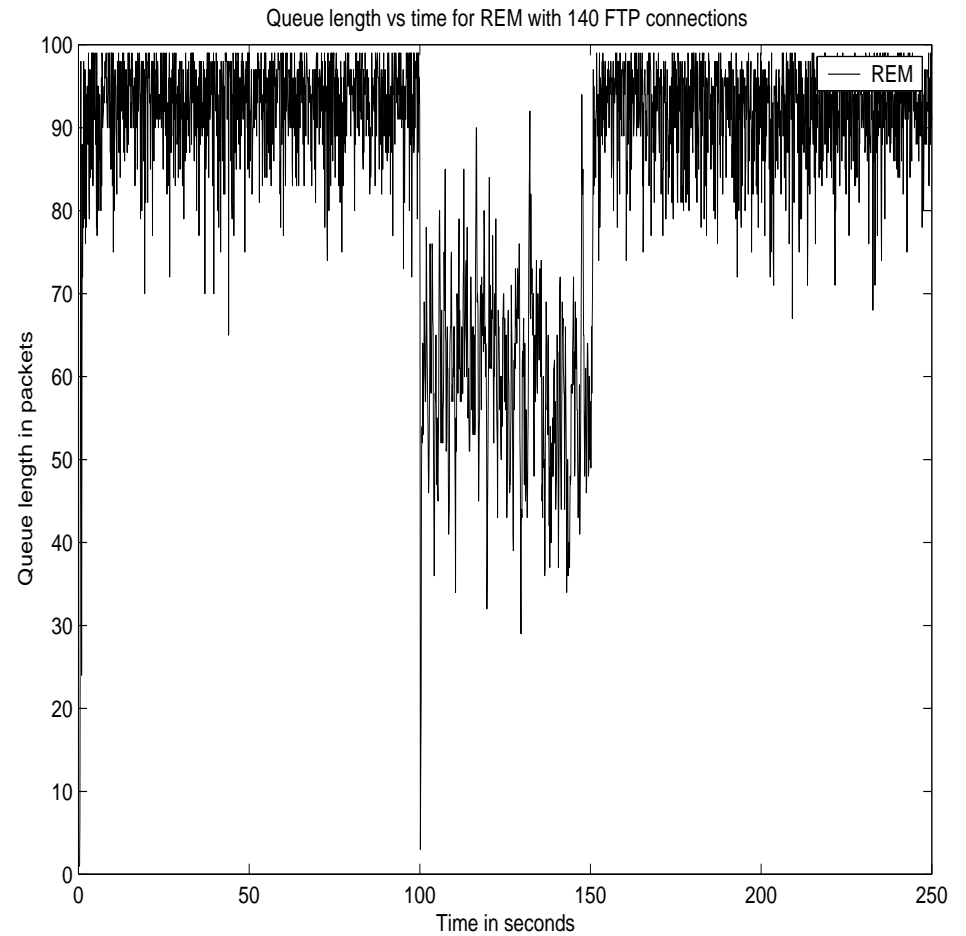


Fig. 9: Evolution of the queue sizes at varying loads for REM

Experiment 4

- Large part of the connections in the Internet comprise of short flows
- Have 40 FTP connections throughout the simulation
- Introduce short flows
- Study the performance of the AQM scheme as the number of short flows increase

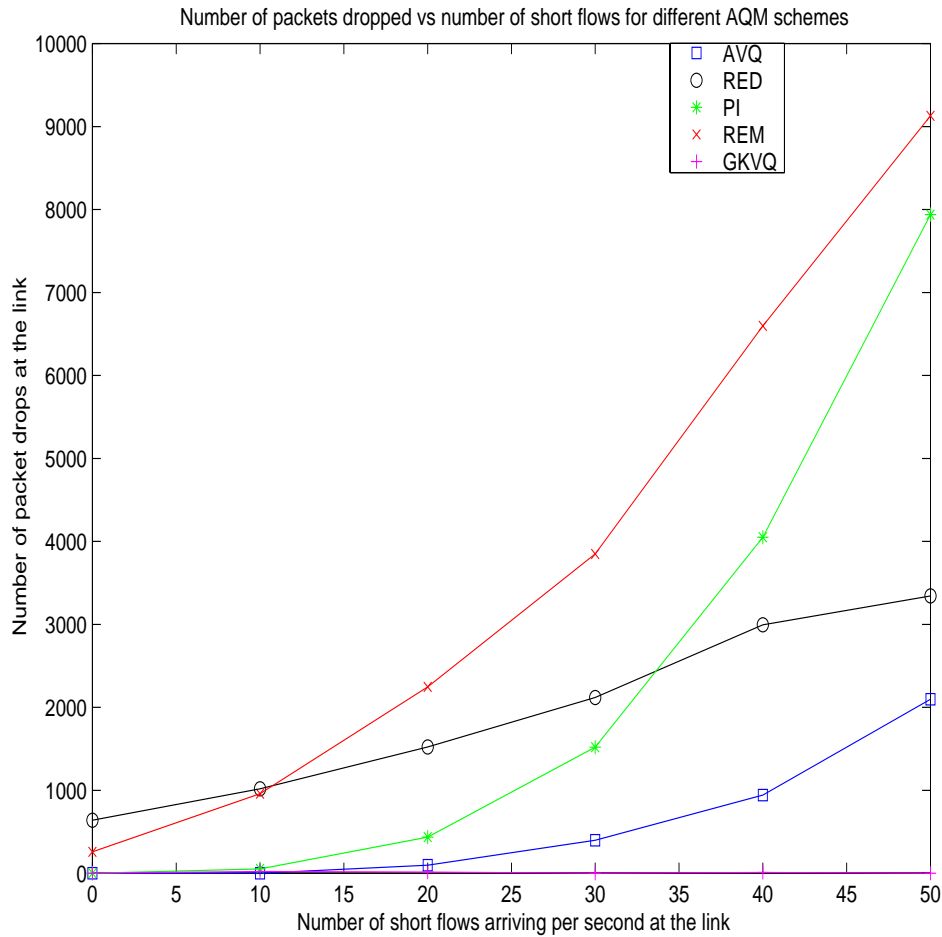


Fig. 10: Packet losses at the link for various AQM schemes

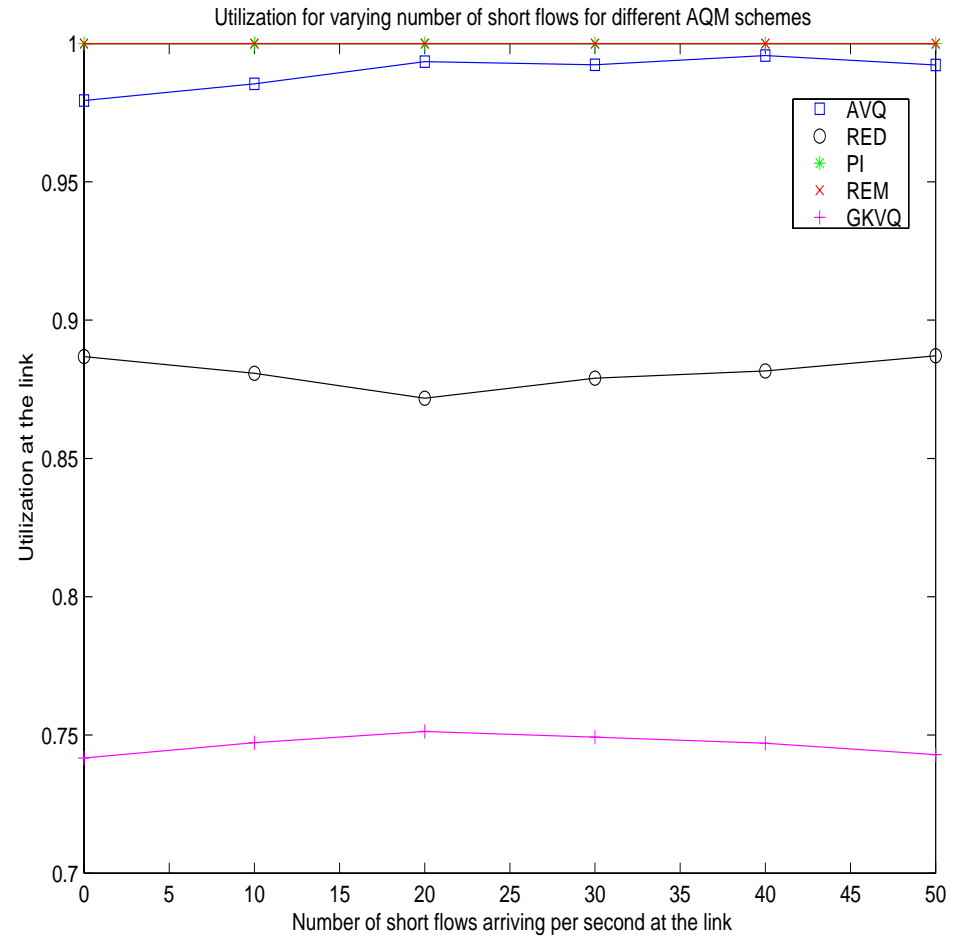


Fig. 11: Utilization of the link for various AQM schemes

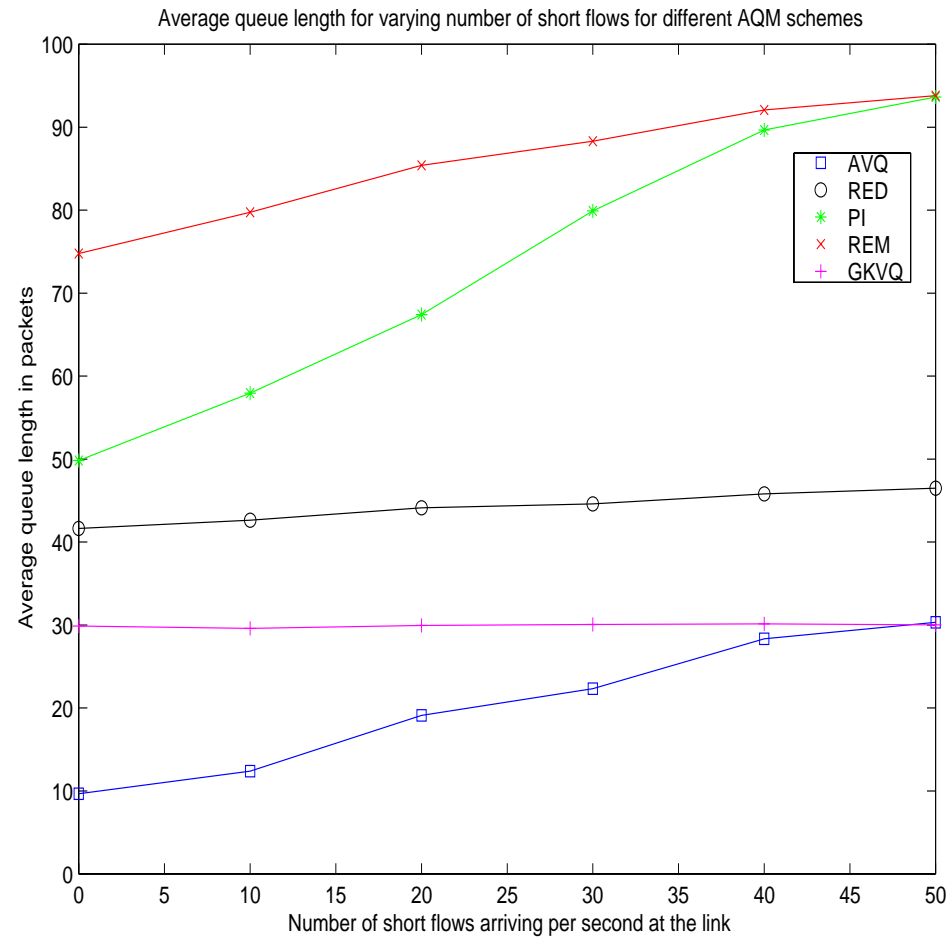


Fig. 12: Average queue length at the link for various AQM schemes

Conclusions

- Studied the impact of random losses on window-flow control schemes derived from various utility functions
- Showed that ECN marking schemes can be designed for each node independently while achieving near loss-free operation for the entire network
- Presented a easily implementable AQM scheme called the Adaptive Virtual Queue Algorithm
- Provided simple rules to choose its parameters
- Compared it with other AQM schemes and showed that it performs better than other well-know AQM schemes