Improve TCP Performance in a wireless networks Using TCP Silence Periods

Elsadig Gamaleldeen Elsadig

Abstract: The large demand for the wireless links and the mobility in the devices increase the need to utilize the bandwidth in the low bandwidth networks like satellite. The gaping method which propose in this paper is an effective approach to improving the performance of the TCP protocol in the wireless networks bandwidth at high delay networks in which the link is highly expensive. The new gaping technique had proposed sending data in TCP silence time by sending packets in the silence time, the new designed algorithm use leaky-bucket scheme for injecting packets in the time gaps through the stream of data in the network. The result had explaining and clarifying that, more data had went pass through the highly delay satellite link. The TCP gaping method could achieve the maximum throughput in interval of RTT and the TCP gaping will spreads data out successfully compared with standard TCP.

Keywords: TCP protocol, TCP silence time, performance evaluation, simulation,

1. Introduction

The transport control protocol is the core of the network traffic in the internet and intranet .Traditional networks made up of links with low bit-error rates. Networks with higher bit-error rates, such as those with wireless links, mobile hosts and noisy environment, violate many of the assumptions made by TCP, causing degraded end-to-end performance and produces higher queuing delays, more packet losses and lower throughput in the noisy environment. One problem common to both satellite and wireless networks is that the capacity of these networks, determined by the product of the bandwidth and the network delay that can be more than 10 times greater than in conventional networks. The mismatch between the high capacity of these networks and available storage at the intermediate routers in the network poses unique problems for TCP [1]. In a typical network, TCP optimizes its send rate by releasing increasingly large bursts (or windows) of packets (one burst per Round-Trip Time) to the receiver until it reaches its maximum window size. In a network with a high delay-bandwidth product, however, TCP's maximum window size may be larger than the queue capacity of some of the network's intermediate routers [2]. The paper objective is to transmit out the window of packets over a Round-Trip-Time (RTT), so that packets are injectedinto the network at the desired rate of congestion window size/RTT. The solution will make better use of the available space in TCP-RTT time to improve data transfer and throughput in the connection without affecting forward throughput. Implementation the new TCP technique to obtain better throughput and performance under different low bandwidth networks and wireless networks.

2. Overview of TCP Gaps

The Gaps is a packet space, due to the delay introduced between two consecutive packets in the Round Trip Time. This time is represented by T_g in Figure (1) which is the amount of gaps between packets, and T_p is the transmission time for each packet.

This type of lost time can be define as a delay so bursts of packets can be send in this time to increase out the traffic and utilize the link bandwidth , so there will be fewer packet drops at the intermediate routers and there will be a potentially higher throughput at the end host. Thus, this packet gaps can also potentially be used as a mechanism to assist in congestion avoidance and control.

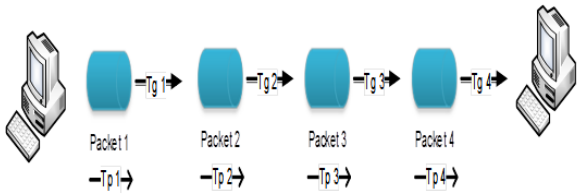

Figure (1) *Packet gaps time between two nodes*

Based on Figure (1) the gaps that get from the Round Trip Time or the ideal operating region that will be used for the new designed protocol mechanism is ranged from $50\,ms$ to $500\,ms$.The (RFC 47) state that the delay must be less than $500\,ms$. There is no packet gap or packet space time less than $50\,ms$ even in the very high packet loss wireless network with less delivered bandwidth. This gap is depending on the application, type of the traffic and the bandwidth that used. In the simulation part and the measurement that had been done the ideal packet gap range may be as small as 100 ms to 200 ms with bandwidth $2Mbps$ while packet loss is only 10% when there is a high TCP traffic in the wire link but this will be more in the wireless link. The Round Trip maybe more than standard state and worse in the low bandwidth link like microwave or satellite networks. The fact that several flows share the network making the bandwidth delay product large, also affects the performance of TCP. The burst nature of TCP can be reduced significantly by using the gaps and send data in the spaces in TCP packet transmissions at the source. These gaping mechanisms are to be deployed on transmitting end-systems and to ensure quality-of-service in traditional networks.

3. Related work

There are many researches done on this area to use this gap and the most one is TCP pacing which is a hybrid between pure rate control and TCP's use of acknowledgments to trigger

- *Elsadig Gamaleldeen is currently pursuing PhD degree program in communication engineering in KarariUniversity, Sudan, PH 0912345838.*
- *E-mail elsadigg@gmail.com*

new data to be sent into the network. Zhang et al. initially suggested using pacing in the TCP context to correct for the compression of acknowledgments due to cross traffic [4]. Other researchers have suggested using pacing when acknowledgments are not available to use for clocking, for example, to avoid TCP slow start at the beginning of a connection, after a packet loss, or when an idle connection resumes. Similarly, pacing can be used to avoid burstiness in asymmetric networks caused by batching acknowledgments [5]. More recently, researchers have suggested using pacing across the entire lifetime of the connection. Partridge argues that pacing can address problems in TCP performance on long-latency, high bandwidth satellite links [6] while the Berkeley WebTP group has combined pacing, receiver-driven congestion control, and application-level framing into a transport protocol specialized for web traffic [7]. I propose use the TCP gaps at the TCP sender to improve TCP performance in wireless networks and find a methodology to use the fragments and gaps (silence time) in TCP transmissions in networks and transmit out the window of packets over a Round-Trip-Time (RTT), so that packets are injected into the network at the desired rate of congestion window size/RTT.

4. The wireless silence time protocol design

In the wireless the gap protocol sender start transmits packets at the highest possible rate after the slow start mechanism, i.e., gaps or silence time, and the receiver sends acknowledgments every Round-Trip Time (RTT) for the packets it receives. The RTT is the base of the propagation-delay ime in the network between the sender and the receiver so in the design the protocol will work in static RTT time, not the dynamic to keep the protocol simple. During the experiments that done the calculation for the RTT was done by performing and using *ping* command during connection starting (connection establishment). After the connection is established, the sender transmit the calculated RTT to the receiver by injected it within the header of each packet so the receiver will be informed by the RTT time from the other side Figure (2). Each acknowledgment contains the number of packets that were received in the previous RTT. When the sender receives the acknowledgments, it compares the number of packets sent, in the previous RTT to the number of packets received just now. After that based on the values of sent and received packet information, the sender adapts its packet time gap.

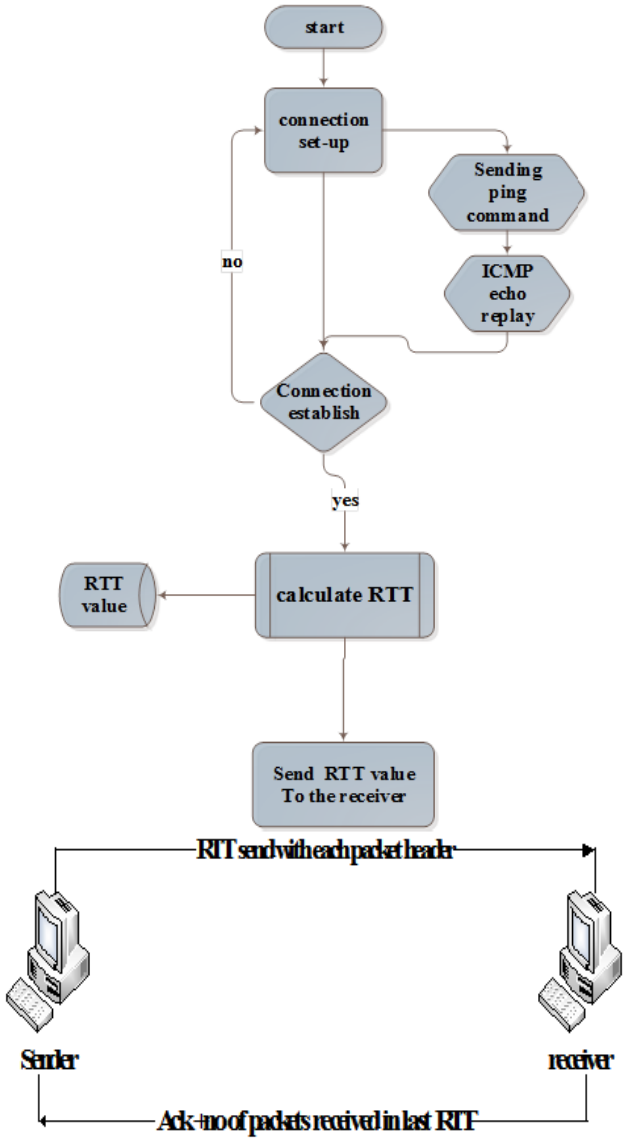

Figure (3) The sender informed by number of packet last received

Figure (2) *RTT Calculation process*

The sender used the Leaky-Bucket Scheme to inject packets in the calculated RTT and every time the receiver acknowledges the sender with received number of packet in the RTT beside the received packets during the normal transmission in the same ACK packet Figure (3). Sender begins in every time to compare number of packet received to the number of packet sent in RTT. If packet sent is more than the packet received then there will be a lost in the network and it begin to reduce number of packets sent but if it's equal zero sender start initial value for the RTT packet gap which is 100 ms (less than that will generate burst traffic and congestion and will not be useful to use in the algorithm) this value will be reached after some RTT rounds so the algorism go direct to the suitable value to start with. If the number of packets sent is less than received it start to increase the number of packets gradually Figure (4).

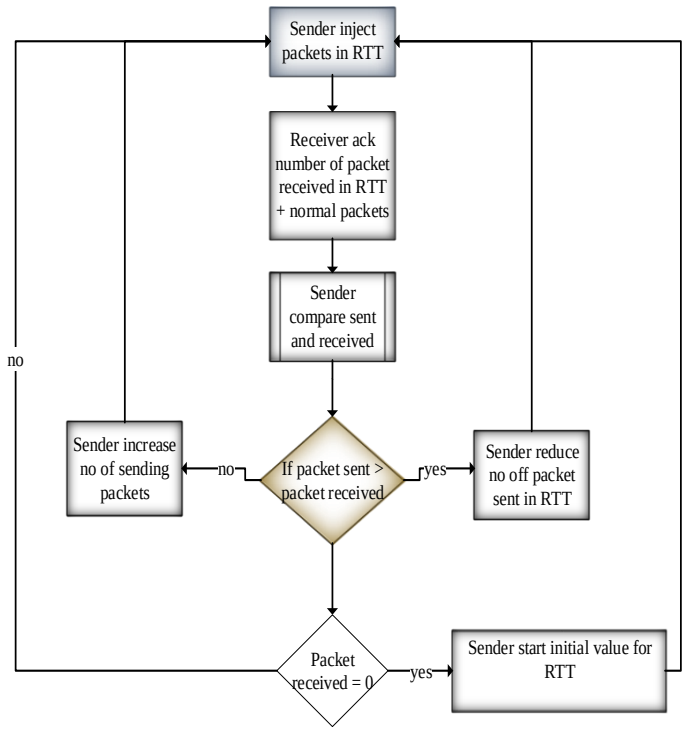

Figure (4) *Algorithm for sending packets in RTT gaps*

4.1 A Leaky-Bucket Scheme modification

To implement sending packets in the silence time, the new designed algorithm use leaky-bucket scheme for injecting packets in the gaps time (Tg) through the stream of data in the network. In a typical leaky-bucket scheme, packets can only enter the network if they obtains "token" from a bucket of tokens. If a packet wishes to enter the network and the bucket is empty, it must wait until a token becomes available in order to enter the network. There are two parameters that govern the behavior of a leaky-bucket flow:

1. The maximum number of packets that the bucket can hold
2. The rate at which the bucket is replenished with tokens.

By altering the size of the bucket it can prevent large bursts from entering the network. By limiting the rate the bucket with tokens can limit the frequency by which these bursts can enter the network.

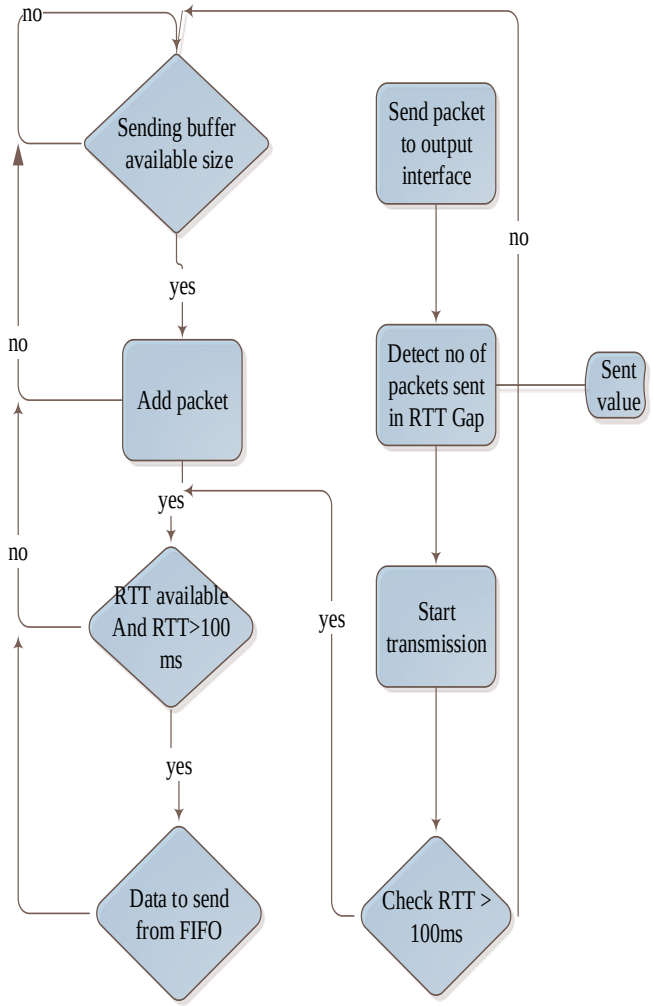

Figure (5) *The modified Leaky-Bucket Scheme for sending data*

A packet inside the congestion window is sent to the output link, if there is a token inside the token buffer. After the packet is sent, a token is removed from the token buffer. If there is no token inside the token buffer, packet must wait until a token arrives. Token buffer is filled with a rate of $\frac{RTT}{CWND}$. Changing the token buffer size affects the burstiness of the traffic flow.

4.2 Simulation

To simulate the silence time or gaps, the TCP must be modified but keeping the same or the original performance for persistent application sessions by modifying the congestion window after idle periods. That can be done by implement the “Restart Window” algorithm as defined in RFC 5681. The new method is used and investigated the effect of gaping technique for TCP networks on high delay network. For simulation experiments that explain the new technique utilization for the network. The design in Figure (6) with satellite gateway link (which is propose as the suitable media with high delay network) was used for simulating the test experiment. The network model used for the simulations is consists of two different networks, two intermediate gateways, and links interconnecting terminals with switches.

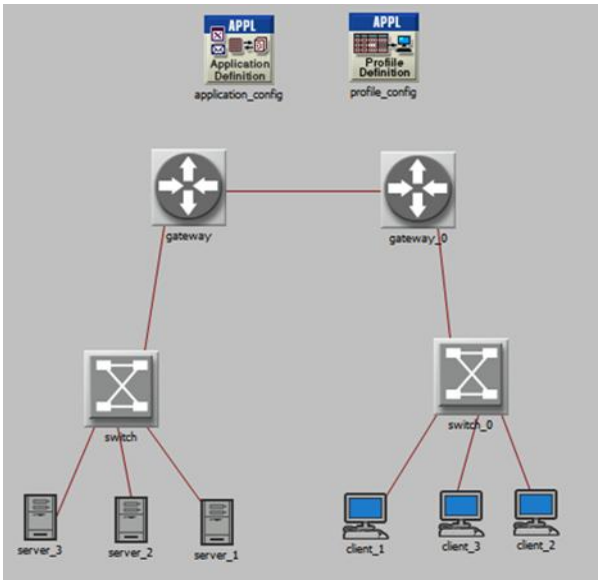

Figure (6) *Network design for testing the TCP Gaping method*

The link between the two gateways is a bottleneck link with a $D\ ms$ propagation delay and C Mbps bandwidth. The links between the terminals and the gateways have a $5ms$ propagation delay and bandwidth equal to the bottleneck link. When running the simulation with variable values in nodes and gateways, the access link bandwidth in the simulations and investigate the performance of the two buffer sizing disciplines.

4.3 Single-Stream using TCP gaping Performance

The study of the TCP gaps started by looking at the performance of classic TCP over a single bottleneck link. The network topology that had been used for the simulations is given in Figure (6) three networks servers with idea link to a client's side through one satellite communication network and same satellite area using two gateways. The results of this simulation is that, the gaping TCP sender is able to smoothly increase its congestion window to the ideal, maximum window of 88 packets during slow-start, without encountering a false-bottleneck. The absence of the false-bottleneck can be explained by the fact that the bottleneck queue never grows beyond zero packets. However, a TCP sender with a $1ms$ timer can only support window sizes of 78 or 100 packets. This explains why the number of packets in the pipe never reaches beyond 78 packets per RTT.

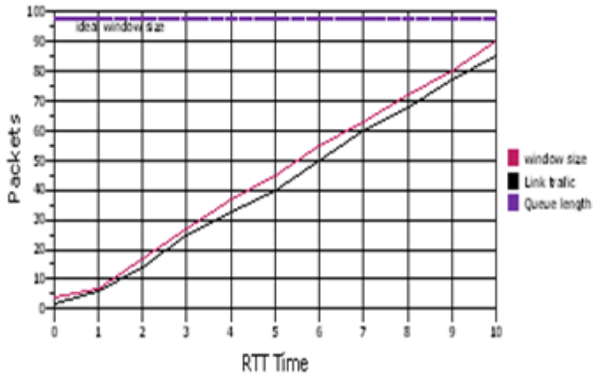

Figure (7) *Queue size and window evolution for a single TCP link*

Figure (7) shows the same simulation with the TCP sender maximum window set to 100 packets. As this figure illustrates, this TCP sender is again able to smoothly able to ramp up to the maximum congestion window in a little over 7 round-trips. In Figure (7), the line representing the packets in the traffic does eventually rise to meet the line representing the congestion window. The effect for increasing the maximum window to 100, beyond the ideal window of 91, prove itself in the evolution of the bottleneck queue length. At 100 packets / RTT, the TCP sender sends packets to the receiver at a rate faster than the bottleneck server can afford. Though it never actually exceeds the capacities of the queues, the queues stay almost constantly full at a level of 8 packets every RTT. By increasing the size of the bursts packets in gaps time out by TCP, it can exactly match the capacity of the network, rather than over utilizing or underutilizing it. The time with the TCP burst size set to 2 packets. A TCP sender with a $1\,ms10ms$ timer and a burst size of 2 packets can afford a windows of size 90 packets, close to the ideal window size of 91 packets. As Figure (7) illustrates, the line representing the packets in the pipe again rises to meet the line representing the congestion window. This time, however, the queues in the network never build up to more than 1 packet. Instead of using the gap technique to send out single packets, this connection sends out bursts of 2 packets. Every time one of these 2-packet burstsenters the network, the first packet enters service, and the second packet waits in the queue for service. This behavior is reflected in the fact that the line representing queue length goes up to 1 every time the TCP sender sends out a burst of packets and Table x summarizes these results.

Table 1 *Summary of the results for a single-link simulation*

Maximum window	TCP Gaping	Queue length	Burst-size	Throughput
91	N	100	NA	1734
91	N	10	NA	1450
91	Y	10	1	1614
100	Y	10	1	1771
91	Y	10	2	1728
91 (with probes)	y	10	2	1750

These results show that the new modification in the TCP protocol affects the connection with burst-size 2 and can achieve close to the same throughput as the classic TCP connection running over a bottleneck with large queues. When using the TCP gap protocol with buffer length 10 packets and burst size 1 packet the new designed protocol capable to send 164 packets more than the classic which is mean 11% enhance in the link utilization. With the same conditions when increasing the maximum window side to 100 packets the new designed protocol capable to send 321 packets more than the classic which is mean 22% enhance in the link utilization.

4.4 Multiple Stream TCP Performance

The TCP is a stream-oriented protocol that means if a sender process sends a stream of bytes, the receiver process will be getting exactly the same stream of bytes. Each connection between a TCP client and a TCP server involves one single stream. The problem with this approach is that a loss at any point in the stream blocks the delivery of the rest of the data. This can be acceptable when transferring text; it is not

acceptable when sending real-time data such as audio or video. TCP allows multi-stream service in each connection. If one of the streams is blocked, the other streams can still deliver their data. The idea is similar to multiple lanes on a highway. When one stream of data blocked with multiple traffic or data deadlock the other streams can send the remaining data.

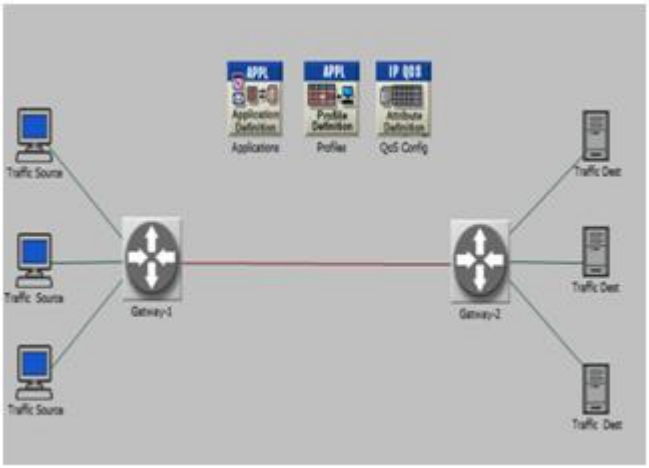

Figure (8) *Network topology for multiple stream simulations*

Performance of gaping protocol modification in several streams compete for access over the same bottleneck link. The topology used for these simulations is shown in Figure (8) and the same set of parameters used in Figure (6), setting the maximum window size for each TCP sender to auto assigned and the gaping burst-size to 2. The start times for each TCP connection, starting each connection 60 seconds after the last connection, and then letting each stream run for 180 seconds. The traffic is sent in the streams in order to force the TCP connections to adjust to both the arrival and relief of congestion", each simulation had run for 20 times and averaged the results together to obtain the results.

Table 2 *Simulation results for different algorithms running over a shared bottleneck link.*

Algorithm	TCP Gaping	Packets sent	Packets successfully added
Classic TCP	N	33478	
Classic TCP	Y	37719	4241
TCP-Reno	N	35813	
TCP-Reno	Y	39420	3607
TCP-FACK	N	42833	
TCP-FACK	Y	45320	2487

Table 2 shows the simulation results for 3 different TCP algorithms, with or without the gaping technique. The results show that new the gaping technique improves all three TCP algorithms, with classic TCP and TCP-Reno achieving the most improvements from others. TCP-Fack with gaping technique achieves the highest throughput of all the algorithms, it improvement over normal, classic TCP.

5. Conclusion

Using TCP gaps to improve TCP performance in low bandwidth networks is a new designed and modified TCP protocol that can be added to the mechanisms that had been proposed to enhance the performance of TCP over wireless links in end to end method which use rate control to manage the traffic and follow of data. The TCP gaps can be used for improving the TCP performance in wireless media in the low bandwidth network or networks with high delay, where the bandwidth is highly cost. Use of TCP gap method is not expected to introduce new problems in the network but it depends on TCP hosts that may control the transmission traffic. The new unexpected result that there is no any packet drops in the networks that used the new gaping technique when buffer size is increased. This phenomena is happened due to the using the small gaps, the fully use of the bandwidth and the arrangement in SYN and ACK.

References

[1] AshuRazdan, AlokNandan, Ren Wang, M. Y. Sanadidi, Mario Gerla:Enhancing TCP performance in networks with small buffers. ICCCN 2002: 39-44.

[2] William D. Ivancic, TCP Pacing Developed, Reprint Edition, Repressed Publishing LLC,2012

[3] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In IEEE INFOCOM, Israel, 2000.

[4] L. Zhang, S. Shenker, and David D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two Way Traffic. In Proceedings of the ACM SIGCOMM ’91 Conference on Communications Architectures and Protocols, pages 133–147, September 1991.

[5] 2010. Advances in Multimedia Information Processing - PCM 2009: 10th Pacific Rim Conference on Multimedia, Bangkok, Thailand, December 15-18, 2009. ... Applications, incl. Internet/Web, and HCI). 2009 Edition. Springer.

[6] C. Partridge. ACK Spacing for High Bandwidth-Delay Paths with Insufficient Buffering. Work in Progress, Internet Draft draft-rfced-infopartridge-01.txt, September 1998.

[7] Rajarshi Gupta, Mike Chen, Steven McCanne, and Jean Walrand. WebTP: A receiver-driven web transport protocol. Submitted to INFOCOM 99,July 1998.