Implementation of an Efficient Transport for Real-Time Game Applications on HFC Cable Networks

Jian Liu, Raheem Beyah, and John Copeland
Communication System Center
Georgia Institute of Technology
Atlanta, GA 30332
{liu, raheem, copeland}@csc.gatech.edu

***ABSTRACT*—On HFC cable networks, in order to develop a real-time game application, we face a design problem to efficiently transmit packets with reliability. This paper introduces the SCRA (server-based collision resolution and avoidance) scheme. SCRA is a reliable transport method using UDP for real-time applications on HFC networks. In this design, we first analyze the collision probability, and estimate the required throughput on an upstream contention channel. Then, we use SCRA to spread the packets from a large number of users to reduce the traffic burst and to maintain a stable throughput for the application. The simulation shows that SCRA achieves ideal transmission time and bandwidth utilization. We also give the formulae to derive the parameters for a specific real-time application using SCRA.**

Keywords—HFC cable networks, upstream channel, reliability, transport protocol, collision

1. INTRODUCTION

In recent years, the hybrid fiber coax (HFC) cable system for broadband access from household users has gained its popularity dramatically. Besides typical data services, such as email, and web browsing, many other types of applications have been developed on HFC networks. Among them, the game application is the most popular one attracting many customers. To operate such an application, a game server resides at the digital head-end, and broadcasts messages to a group of users. The cable modem clients will send the user response messages to the server.

On HFC cable networks, there are some challenges to this interactive application. First, a data session for a single transaction is usually very short but very bursty, i.e. sometimes it only needs one packet from each client to the server, but a large number of clients exist at the same network. Those packets from different clients will interfere each other because of the contention on upstream channel. Second, the response packet intervals are spontaneous and have comparatively large time ranges. Bandwidth reservation is not efficient for network operators. Third, each packet has a real-time requirement with a hard deadline. A packet received after a deadline is useless.

In practice, HFC networks are heavily contention-based. A large number of users could coexist in a single collision domain, such as 500 – 2000 users sharing one upstream channel. For such an application MAC protocol is operated at the *immediate mode*, which resembles a Slotted-Aloha scenario, to reduce the inefficiency of reservation. Without reservation of time slots, packet collisions at upstream could be a very serious problem. Furthermore, on HFC MAC layer, senders cannot directly sense a packet collision, unlike the case on Ethernet. Loss discovery requires other mechanisms. On transport layer, though TCP is capable to recover lost packets, it is very inefficient for the real-time requirement due to its extremely slow response time. To use UDP, we will have to implement an efficient transport scheme with reliability.

Nowadays, despite of the popularity in industry, few literatures have directly addressed the issues of interactive real-time applications on HFC networks. Related work can be found in [1] and [2], which focus more on signaling design to provide reliability. The transport layer performance of TCP on HFC network is studied in [3]. The related MAC layer issue is addressed in [4].

The paper is organized as the following. In section 2, we introduce the cable network architecture and the example game application. The proposed SCRA algorithm is presented in section 3. In section 4, we show the simulation results. Finally, we conduct a system analysis to estimate SCRA parameters for the design in section 5.

2. ACHITECTURE AND APPLICATION

2.1 The simulation model

The abstract network topology and parameters for the simulated HFC system are shown in Figure 1. (NS2 is used for this simulation.) There are separate nodes representing the game server, the head-end (HE), one QPSK demodulator for an upstream channel, and 500 cable modem users (CM $_{1\text{-}500}$). The downstream is a broadcast link from the head-end to cable modems. The upstream channel is competed by the traffics from a number of simultaneous game users and the CMs using other applications, such as web, and email. The upstream channel is assumed in an immediate mode, which means the protocol at the MAC layer is *Slotted-Aloha*. Therefore, in the simulation, we will study the packet delay mainly affected by the upstream contentions.

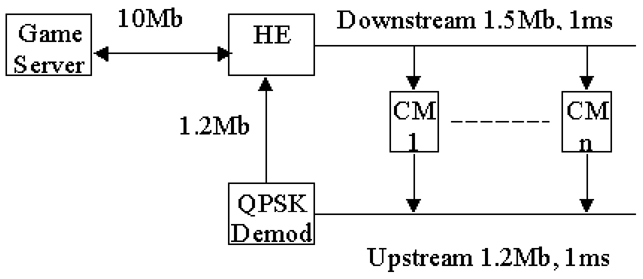

Figure 1. The Abstract Network Topology

Figure 1 also shows the basic system parameters. These parameters are estimated based on following situations. The QPSK channel is 1.5Mb/s, but it will be reduced by an overhead of the packet conversion between Ethernet frames and cable system frames on upstream. So, the available upstream bandwidth is about 1.2Mb/s. The one-way propagation delay (1ms) is an approximation of a typical cable network.

2.2 The game application

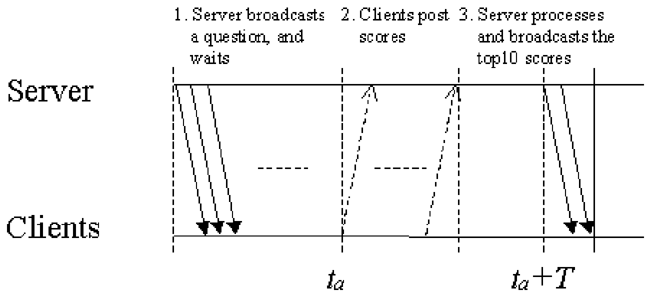
1. Server broadcasts a question, and waits
2. Clients post scores
3. Server processes and broadcasts the top10 scores
Server
Clients
t_a
t_a+T

Figure 2. The Game Procedure

In our simulation, the transaction of the game application is relatively simple. It exemplifies an application such as a TV game. During the game, the game server first broadcasts a question through the downstream channels to the client cable modems. Game players have t_a seconds to answer the question. Nearly at the end of t_a seconds, the client program will score the answer, and post the score to the game server. Only one packet will hold the score from each client, and the packet size is 80 bytes. At the server side, after t_a seconds, it starts receiving the client packets. The server waits until another T seconds when all packets arrive. Then, the server will sort all scores and broadcast the best 10 scores to clients. To avoid a long latency, the server must set a deadline T for all client packets to come. The procedure is shown in Figure 2.

During the response period of T, packets could be lost due to the collision at the upstream channel. If a packet loss is detected, the client needs to retransmit the packet. For every retransmission, the packet latency increases. Therefore, the response period T is a critical time. In this design, to meet the customer satisfaction, we must be able to keep the overall packet loss rate negligible in a given response time T. On another hand, the period T also limits the number of retransmissions. Furthermore, how does the application adapt to varied traffic loads? Those problems have to be considered for the implementation of the transport scheme.

3. SCRA ALGORITHM

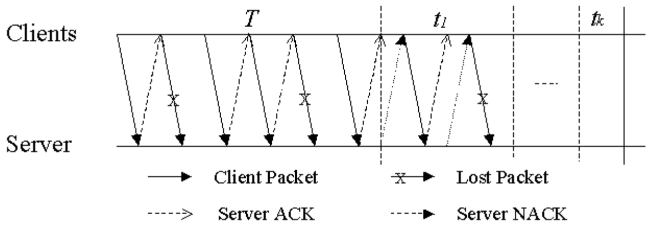

Figure 3. SCRA packet transactions

SCRA is a server-initiated collision resolution and avoidance scheme for real-time packet transport using UDP. Because of using UDP, SCRA must be able to recover the lost packets. Figure 3 illustrates the SCRA transactions. SCRA uses a server-initiated retransmission. The server maintains a timer to track client's packets. For each received packet from a client, the server replies an ACK. At a timeout, the server will send a request (NACK) packet to those "missing" clients. On the client side, after sending a packet, it waits for an acknowledgement. If ACK is received, the client does nothing. If NACK arrives, it will resend the packet.

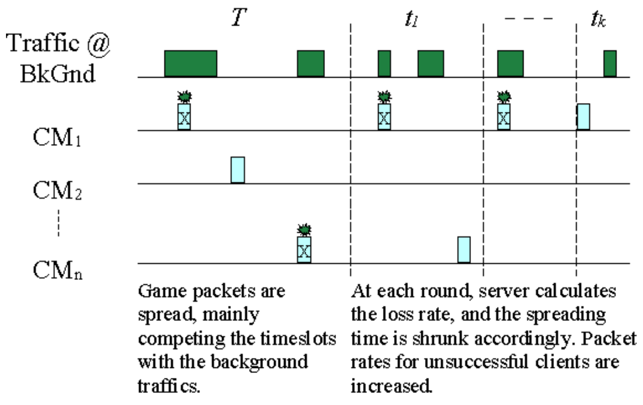
Traffic @ BkGnd
T
t_l
t_k
CM1
CM2
CMn
Game packets are spread, mainly competing the timeslots with the background traffics.
At each round, server calculates the loss rate, and the spreading time is shrunk accordingly. Packet rates for unsuccessful clients are increased.

Figure 4. SCRA Algorithm

SCRA is not only a scheme for packet loss recovery, but also an algorithm that can effectively avoid and resolve collisions for the game clients. Figure 4 shows the mechanism. At the beginning, we estimate the aggregated traffic G from all clients. Then, spread the traffic out along a time period of T. For a large T, it contains more timeslots than the number of game users so the total game traffic has approximately a uniform distribution with an average rate less than one packet per timeslot. So, the contentions among game clients are greatly reduced. Because all stations on HFC are synchronized, this could be implemented as each user independently running a random generator. However, it is still possible that the packets from game clients will collide with the packets from other applications, as well a small chance of collisions from game clients themselves. Therefore, after the first transmission period *T*, parts of the packets will success to arrive at the server. For those missing clients, the server will send the request packets (NACK) during a time t_1. t_1 is proportional to the number of missing users, $t_1 = T * L$, (*L* is the loss rate, and $L < 0.63$ for slotted-Aloha), so the retransmission time window is shrunk at each time, such as $T > t_1 > \ldots > t_k$. 0. By this way, the total traffic *G* generated by all clients is unchanged. For an unlucky client, however, if the collisions occur repeatedly, the individual packet rate *g* from this client will increase, and converge to *G* exponentially, *g* .

$t_k = T \, . \overset{k}{\underset{i}{}} L(t_i)$, where $L(t_i)$ is the average loss rate within the *ith* retransmission. (If in this process, $L(t_i)$ is assumed to be constant, then, $L(t_i) \sim L$, and $t_k \sim TL^k$.) Therefore, at a heavy load condition, *L* will be greater, and the spreading time window reduces less fast. Oppositely, at a low traffic load, the window could shrink rapidly. In all, the overall traffics from the game clients remains in constant in the transmission period *T*, such as *G* not growing with time. This assures the network less affected by the burstness of the game traffics, and prevents the network from an unstable state.

SCRA is designed based on the analysis of Slotted-Aloha MAC protocol, which is used in the HFC immediate mode. Under a certain traffic load, Slotted-Aloha produces a steady throughput. By spreading the client game packets, it de-correlates the traffic, minimizes the interference with the background traffic, and achieves the optimum transmission.

4. SIMULATION RESULTS

In the simulation, the applied traffic *G* is designed at 160kb/s from 500 game clients. This traffic is less than 15% of the upstream link capacity at 1.2Mb/s, and the packets are spread out in a 2s window at the first transmission. SCRA is tested under different background traffics at 300, 600, and 900kb/s respectively.

Figure 5 shows the percentage of finished users versus time. We find under traffic loads of 25% (300kb/s), 50% (600kb/s), and 75% (900kb/s), the finish times are nearly linear. That indicates the success rate of packet transmission is unchanged. This is an optimum result, because it indicates the interference between the game traffic and the background traffic is minimal. The three lines are distinguished by the different skew rates, which are determined by the background traffic loads. Figure 6 shows the game traffic bandwidth under the three traffic loads. The labeled '**US**' is the total traffics that the clients try to send on upstream, and '**Rcv**' is the traffic received by the game server. From the plots, we see the traffics are less bursty. Especially at the loads of 25% and 50%, the upstream traffics are almost even everywhere. This proves that the network is steady during the packet transaction, and SCRA does not reduce time to meet the deadline by overloading the network.

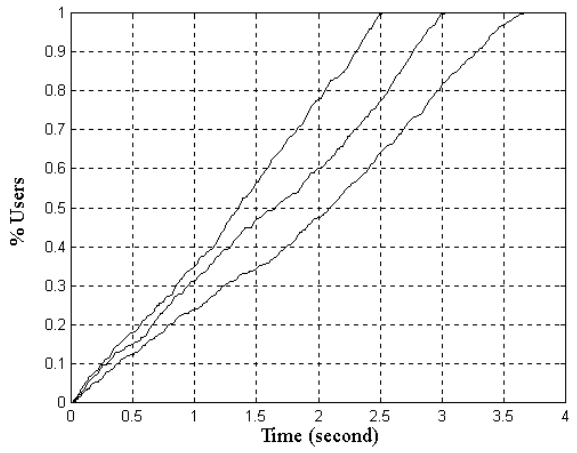

Figure 5. The percentage of finished users vs. time with the different background traffics (The three plots, from left to right, corresponds 300, 600, 900 kb/s)

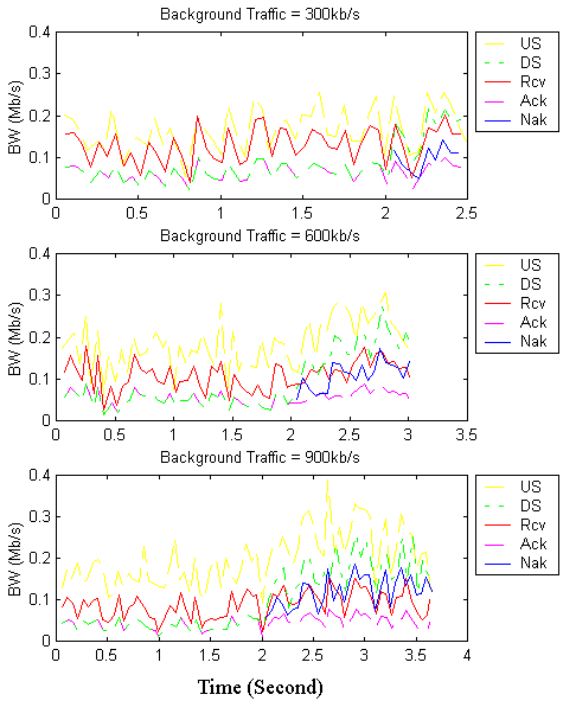

Figure 6. Traffics from the game application
US: total applied game traffics at upstream; DS: total game traffics at downstream; Rcv: the actually received traffics at upstream; Ack: the downstream traffic by the server sending ACK packets. Nak: the downstream traffic by the server sending NACK packets.

5. SYSTEM ANALYSES

Referred to the second paragraph of section 2.2, in the implementation of this SCRA algorithm, we have the following parameters: the response time T, the number of clients on the network N, the typical traffic load X, and the expected packet missing rate P_{miss}. So, the purpose of this system analysis is to derive a set of design formulae using those parameters.

5.1 Throughput and packet loss rate

For a channel using Slotted-Aloha MAC protocol, the throughput is given by $S = X/e^X$ [7], where X is the traffic load. To simplify our analysis, we made following assumptions:

i. The traffics from the game clients $G = \sum_{n}^{N} g_n$, g_n is the traffic from the nth client. To avoid collision from different game users, the packets are spread uniformly in a time window T, so $g_n \sim g$, $G = gN$. G is designed to be only a portion of link bandwidth, and constant.

ii. The throughput of game application is proportional to its applied traffic. It can be shown as,

$$S_g = Ge^{-(X+G)} \qquad (1)$$

iii. The background traffic X is steady during the game traffic transaction period. So, the packet loss rate due to collision is approximately invariant, $L(t) \sim L$.

iv. G is chosen to be comparably small and uniform, the negative impact of the game traffic to the network will be negligible. For example, when G is 0.1 and traffic load X is 0.2, the throughput for X is about 0.15, only reduced by 1.5%. So, for the game clients, the packet loss rate from collision would be

$$L \quad 1-e^{-X} \qquad (2).$$

If T is small enough, it is not unreasonable to assume the background traffic X will be constant[1]. This validates i - iii. Furthermore, we have intentionally spread the packet delivery in uniform. The throughput from Slotted-Aloha indicates there are at least a percentage of $1/e^X$ timeslots available, given the background traffic sources are independent to each other. Here, the game traffics are independent from the background traffic, and are distributed uniformly, so we expect the throughput for the game traffics is proportional to the available timeslots under the traffic load of X. Therefore, this argument justifies Eq.(2) from the assumption iv.

In SCRA, G is a design parameter. The larger G the higher throughput for game traffics, but Eq.(2) may no longer be held. When a larger G is used, L also increases. This requires a balance to determine a proper G. Figure 7 plots the goodput vs. traffic load with G = 0.1. The lower bound of goodput corresponds (1), *as* $e^{-(X+G)}$, and the upper bound of goodput is e^{-X} , a case of not considering the impact of G. The plots will be useful to decide an operation range for the HFC network under a certain traffic condition. For example, if the typical background traffic load is around 0.4, and we need a goodput of no less than 0.6 to meet the time requirement, the proper G is 0.1. We should only choose G in a small range that the transmissions meet the deadline.

[1] However, we realize that it might not be the case for the fractal traffic[8][9]. So, here is a simplified case.

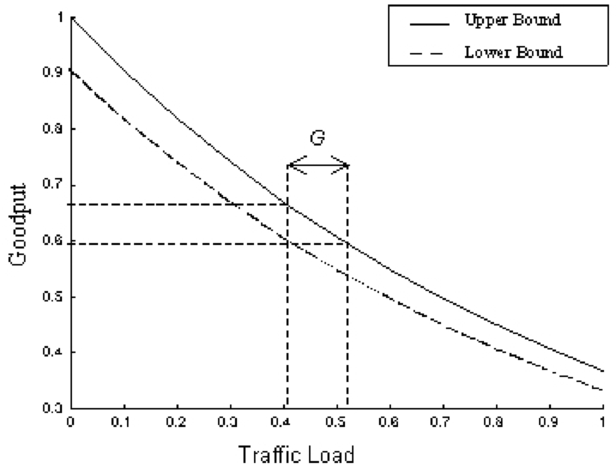

Figure 7. Goodput vs. Traffic Load

5.2 Packet latency and missing deadline probability

Whether a sender can meet the deadline depends on the number of retransmissions due to the collisions. Therefore, the packet latency can be found in (3):

$$D(K) = \sum_{i=0}^{K} .\ L^i = .\ \frac{1-L^{K+1}}{1-L} \qquad (3).$$

K is the number of transmissions. . represents the time interval at the first transmission, and can be calculated by . = . NB/GC . N is the number of clients; B is the packet size; . > 1, a constant; and C is the link capacity.

Given a time deadline T, we can solve K as:

$$K = \left\lfloor \log\left\{1-\frac{(1-L)T}{.}\right\} \Big/ \log L - 1 \right\rfloor \qquad (4).$$

Therefore, K is the maximum number of transmissions before missing the deadline.

The *probability of missing deadline* can be expressed as:

$$P_{miss}(k) = L(1-L)^{k-1} \qquad (5).$$

So, using (4) and (5), we can estimate the probability P_{miss}, and derive . and G. That is, if the application requires no more than α% of packets missing their deadline, under the traffic condition, from (5), we can find *k*. Compared with the *K* from (4), if *K* is larger, we have to reduce . , so *G* may increase. Note that, to make the system stable, and to have less interference to the background traffic, we should keep *G* as small as possible. Because increasing *G* only increases traffic burst and makes the system fluctuated.

6. CONCLUSIONS

Usually, a collision resolution and avoidance scheme is in MAC protocol. However, for the real-time game applications on HFC networks, it would be more flexible to design such a scheme on the application layer using UDP.

On design and implementation of such a scheme, there are several parameters and constraints: deadline, packet missing rate, the customer number, and network traffic load. We have considered all those parameters in SCRA algorithm, and derived the formulae. In analysis, SCRA is based on the most common MAC protocol Slotted-Aloha. Under a certain traffic load, Slotted-Aloha produces a steady throughput. By spreading the client game packets, it de-correlates the traffic, minimizes the interference with the background traffic, and achieves the optimum transmission.

In conclusion, we find that there are several positive aspects of SCRA: (1) with proper parameters, SCRA can meet a hard deadline requirement. (2) SCRA is very flexible to adapt to various traffic conditions. (3) SCRA does not generate the bursty traffic on the upstream channel. In fact, the negative impact by SCRA is limited, and can be estimated and adjusted in design. (4) Obviously, SCRA can be extended with some features in a specific application to achieve better performance.

ACKNOWLEDGMENT

The authors would like to thank Bill Wall, Mark Schutte, Tom Heagy from Scientific-Atlanta for their helpful support and useful discussion in this study.

7. REFERENCES

[1] T. Bova, T. Krivoruchka, "Reliable UDP Protocol," IETF Draft, Feb. 1999

[2] G. Bai; Z. Shen; W. Wang; S. Cheng, "RSTP: A New Lightweight Transport Protocol for VoIP," International Conference on Communication Technology Proceedings, 2000. Vol. 1

[3] R. Cohen, S. Ramanathan, "TCP for High Performance in Hybrid Fiber Coaxial Broad-Band Access Networks," IEEE/ACM Transactions on Networking, Vol.6 Feb. 1998

[4] J.Limb, D. Sala, "A Protocol for Efficient Transfer of Data Over Hybrid Fiber/Coax Systems," IEEE/ACM Transactions on Networking, Vol.5 Dec. 1997

[5] Cidon, I.; Rom, R.; Gupta, A.; Schuba, C. "Hybrid TCP-UDP Transport for Web Traffic" IEEE International Performance, Computing and Communications Conference, 1999

[6] D. Velten, R.Hinden, J.Sax, IETF RFC-908 Reliable Data Protocol, Jul. 1984

[7] A. S. Tanenbaum, Computer Networks, 3rd edition, Prentice Hall, 1996

[8] W. E. Leland, M. S. Taqqu, W. Willinger, D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," IEEE/ACM Trans. Networking Vol. 2, pp.1-15. 1994.

[9] V.Paxson and S.Floyd, "Wide-area traffic: The failure of Poisson modeling," IEEE/ACM Trans. Networking, vol.3, pp.226-244, 1995