Scientia Iranica, Vol. 8, No. 4, pp 297–302
© Sharif University of Technology, October 2001

SCIENTIA
IRANICA

A New Protocol for Asymmetric Communication Channels: Reaching the Lower Bounds

S. Ghazizadeh[1], **M. Ghodsi**[*] **and A. Saberi**[1]

In this paper, the problem of sending n-bit of data from a client to a server using an asymmetric communication channel is discussed where the line bandwidth from server to client is much higher than that of client to server. The goal is to provide a protocol for this problem so that the average number of actual bits sent by client to server becomes as low as possible. Assuming that data is drawn from an arbitrary distribution D, the average number of bits sent by client to server cannot be less than $H(D)$, where $H(D)$ is the binary entropy of D. This problem was first considered by Adler and Maggs [1]. They assumed that the distribution of D is known only by the server and introduced a protocol in which the number of bits sent from client to server is $1.71H(D)+1$. In this paper, this work has been extended and a protocol is proposed which reduces the constant factor 1.71 to $1+\epsilon$ ($\epsilon > 0$) in a way that the computation performed by the server is still reasonable.

INTRODUCTION

In recent years, many commercial networks have been introduced with asymmetric bandwidth capabilities where the speed of one direction, from server to client, is much more than that of client to server. Telephone companies have started using these capabilities for their subscriber lines. Asymmetrical Digital Subscriber Lines (ADSLs) utilize current phone lines to deliver up to eight megabytes per second. This technology has also been used in satellite communications and wireless networks and can, for example, provide a download speed of 1.5 mbps and an upload speed of 64 kbps.

There are many issues involved in asymmetric communication, but the main concern, here, is whether it is possible to utilize the high speed link to improve the performance of the low speed link and, thus, efficiently transfer a string of information from a client to a server on the low speed link.

The asymmetric communication can also be used in situations where the client processing power and/or the amount of memory is very limited compared to those of the server, although the channel may be symmetric. For example, the client can be a mobile node which uses a wireless channel for communication with the base. The mobile node may need to limit the transmission in order to conserve power, while in the base this is not the case. Such problems can also be solved with the protocol introduced here.

In this paper a protocol is presented that allows the client to send x, an n-bit string, to the server on an asymmetric channel, through forwarding a nearly optimal number of actual bits across the low-speed channel. The protocol is an extension of the one presented in [1] for which the average number of bits sent by the client is $1.71H(D)+1$, where $H(D)$ is the binary entropy of D, the distribution of x. The first protocol we propose solves this problem in $1.47H(D)+4$ on the average. Then, the protocol is improved and the factor is reduced from 1.47 to $1+\epsilon$ for an arbitrary small value of ϵ. This is very close to the theoretically optimal $H(D)+1$ bits required to be sent by the client.

Although this paper is more concerned with the theoretical aspects of the performance of such protocols, there is no need to further emphasize the practical benefits of such efficient protocols.

The organization of this paper is as follows. First, some classical results and the lower bound on such communications are presented. Then, the model used in the protocol is discussed. Next, the details of the protocol considered for extension are provided. This is

1. *Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*
*. *Corresponding Author, Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

followed by the new protocol. At the end, two open problems are presented for further work.

CLASSICAL ALGORITHMS AND RELATED WORK

To reduce the expected number of bits sent over a communication channel, classical coding schemes such as Shannon codes [2] and static Huffman codes [3] have been proposed. In these techniques, the string $x \in \{0,1\}^n$ of length n to be sent is assumed to be drawn randomly from a probability distribution D, which is known to both server and client. In 1948, Shannon proved that the expected number of bits sent from the client to the server cannot be less than $H(D)$. $H(D)$ is the entropy of the distribution D, a quantity between 0 and n, defined as:

$$H(D) = \sum_{x \in \{0,1\}^n} D(x) \log_2 \frac{1}{D(x)}.$$

Using Shannon and Huffman codings, the expected number of bits sent from the client to the server is, at most, $H(D)+1$. This can be achieved by having the server sort the distribution D in an ascending order of probabilities and send all of them to the client. The client can now just send the index of the desired string, x, to the server. It is clear that the number of bits sent by the client depends on the index but, on average, it is, at most, $H(D)+1$. However, this protocol is impractical because the number of bits sent by the server is exponential in n. It is also very inefficient for asymmetric channels where the client, which can be a small mobile, with its often limited bandwidth, lacks enough resources to locally compute or store the distribution D.

The question dealt with in this paper is whether there is a protocol with a reasonable complexity in which the number of bits sent from server to client is polynomial in n and the number of bits sent from client to server is very close to $H(D)+1$.

Several researches have already considered this problem. In the area known as interactive communication, there are a good number of researches on efficiently sending a string x from a client to a server, where the server has some knowledge about x that is unknown to the client [4-7]. In all of these works, however, the channel is assumed to be symmetric and the goal is to reduce the total number of bits sent on the channel.

For the asymmetric channels, the problem was first considered by Adler and Maggs [1]. In their model, only the server knows the distribution D on string x. They introduced several protocols including the one in which the average number of bits sent from client to server is almost $1.71H(D)+1$.

THE MODEL

The same model as in [1] has been used here which is an asynchronous model based on Yao's two-party communication complexity model [8]. In this model, client and server exchange some bits as stated in a specific protocol which enables the client to send its n-bit string to the server. The protocol determines the steps to be followed exactly by the two parties, that is, which one of the parties should send some bits, as well as the values of the bits sent. The bits sent by the server are based on the information in D, which is available to the server, and on the information thus far received from the client. The client does not have any information on D and the bits it sends are only dependent on the string and the bits thus far received from the server. At the end of the protocol, the server knows the n-bit string of the client.

The model considers the local computation performed by the server. It is assumed that the local computation in the client side is very little and is not considerable. Instead, the computations performed by the server is very important and can be extensive. This computation is mainly determined by the complexity of accessing the distribution D which is modeled by queries to a black box. It is assumed that D is represented as a black box and the server queries the box for a k-bit string r, where $0 \leq k \leq n$. The box returns the cumulative probability of n-bit strings that start with r. The complexity of the server computation is, thus, the number of black box queries plus other local computation.

In the protocol presented here, a stronger model is used for the black box query. A black box query in this model involves two n-bit numbers a and b, the box returns the cumulative probability of strings between a and b. It can easily be shown that a query of this kind can be performed in the original model with, at most, $O(n)$ additional cost.

The actual computation performed by the server depends on the representation of D in it and, thus, the black box model does not accurately estimate its complexity, nevertheless, this model is very simple and provides a good estimate. The important point is that the accuracy of complexity of computation does not affect other properties of the protocol.

In general, a protocol is characterized in terms of three parameters, $[\sigma, \phi, \lambda]$, where σ is the expected number of bits sent by the server, ϕ is the expected number of bits sent by the client and λ is the number of black box queries done by the server. For example, the classic protocol is a $[2^n, 1, 0]$-protocol. Adler and Maggs have defined a fourth parameter which is the number of *rounds* in the protocol, expressed as the maximal sequence of consecutive bits sent by the server (without any bits sent in between by the client), followed by a maximal sequence of consecutive bits sent by the client. Minimizing the number of rounds is important. However, since in the referenced protocol and the protocol presented here, the client always sends one bit to the server, the number of rounds equals the number of bits sent by the client. Therefore, this parameter is ignored in this paper.

ADLER AND MAGGS PROTOCOLS

Addler and Maggs have presented three protocols in their paper. The first protocol is a $[3n, 1.71H(D) + 1, 3n]$-protocol and is considered a computation-efficient protocol. Here, this protocol is extended to its lower bound limit in terms of the computation performed by the client at the expense of more computation by the server. Because of the need to refer to the details, this protocol with its proof are presented here. Their second protocol is a $[O(n), O(H(D) + 1), 2^n]$-protocol with O(1) rounds designed to provide the efficient number of rounds. The third is a $[O(n), O(H(D) + 1), O(\frac{n2^c}{c})]$-protocol with $O(\min(\frac{n}{c}, H(D)+1))$ number of rounds which is a trade-off between the first two protocols (c is the trade-off parameter).

Now, the first protocol is presented in detail. In this protocol, the server, in each run, sends the client a query consisting of a candidate prefix for the client string, x and the client responds whether the prefix is indeed correct or not. The server keeps track of the prefix r of x thus far correctly determined and as the protocol proceeds, the server expands r to x. If the client response to the query is positive, the server expands r and if it is negative, the server eliminates that string from the candidate set. Further queries to the client depend on its responses to the previous queries. To do this effectively, the original probabilities are adjusted based on the information received from the client so far. The authors define *adjusted probability* of a black box query, say Q, as follows. Let X be the set of rejected prefixes thus far and P_X be their cumulative probability. Also let P_Q be the result of the black box query. The adjusted probability of the string Q is defined as $\frac{P_Q - P_X}{1 - P_X}$.

The mentioned protocol is as follows:

- Let r be the empty string,
- Repeat until $r = x$,
  - Conditioning on all information learned from the client thus far, the server finds a prefix of the unknown bits as follows:
    - ∗ Let s be an empty string. The server repeats the following until it has a prefix rs that either occurs with probability between $\frac{1}{3}$ and $\frac{2}{3}$, inclusive, or extends to the end of the string.
      - · Query the black box for rs0.
      - · If the exclusion adjusted probability of the value returned by the black box is $> \frac{2}{3}$, then 0 is appended to the end of s.
      - · If the exclusion adjusted probability of the value returned by the black box is $< \frac{1}{3}$, then 1 is appended to the end of s.
      - · If the exclusion adjusted probability of the value returned by the black box is between $\frac{1}{3}$ and $\frac{2}{3}$, then 0 is appended to the end of s.
  - The server sends s to the client.
  - If rs is a prefix of x, the client responds with a "y", after which the server sets $r = rs$.
  - If rs is not a prefix of x, the client responds with a "n", after which the server updates the exclusion adjusted probabilities accordingly.

In this protocol, the behavior of the server and the client is deterministic and the only source of randomness is the value of x. Also, the prefix that is sent either extends to the end of string x, or occurs with the probability between $\frac{1}{3}$ and $\frac{2}{3}$, since when a prefix that occurs with probability $p > \frac{2}{3}$ is extended by one bit, the prefix with the more likely of the two settings for that bit occurs with probability of at least $\frac{p}{2}$.

A new version of the analysis of this protocol which gives hint to the protocol presented here is now provided.

Theorem 1

For any distribution D, the above protocol is a $[3n, 1.71H(D) + 1, 3n]$-protocol.

Proof

Considering the way prefix s, to be sent, is selected, it is obvious that the probability of receiving a positive response from client is at least $\frac{1}{3}$. Thus, for every bit, the server needs to query the client three times, on average, to get a positive response. Therefore, on average, the server needs to send $3n$ bits to the client. Using the same proof, the average number of black box queries is also $3n$. The probability space from where x is selected in each phase is reduced at least by a factor of $\frac{1}{3}$ by each query so, for every x, the number of queries to the client is $1 + \log_{\frac{2}{3}} D(x)$ and, thus, the average number of bits sent by the client to the server is:

$$\sum_{x_i} D(x_i)(1 + \log_{\frac{2}{3}} D(x_i)) = 1 + \frac{H(D)}{\log_2 \frac{3}{2}},$$

which is approximately $1.71H(D) + 1$.

THE PROPOSED PROTOCOL

The main idea in the above protocol is to divide the probability space into two portions of $\frac{1}{2} \pm \frac{1}{6}$ size. In other words, the server in each phase reduces the probability space by at least $\frac{1}{3}$ through querying the client.

Limiting probability space in each step allows to find x in distribution D in, at most, $\log_{\frac{3}{2}} D(x)$ queries. Therefore, the average number of queries is $1.71H(D) + 1$. As can be seen, the constant factor 1.71 comes from the approximation $\frac{1}{6}$ in dividing the probability space. If the space can be divided with a better approximation, then, 1.71 constant is decreased; this is the main idea behind the proposed protocol.

As the protocol proceeds, a distribution I is dealt with that holds all possible values of x. Clearly, I is D at the beginning. $I = [s_I..e_I]$ is denoted where s_I and e_I are the smallest and largest points of I in their binary representations, respectively. Originally, $I = [0..2^n]$. All points of I are sorted according to their binary representations on a circle in a way that s_I and e_I are adjacent. This is denoted as the distribution circle. Let $I(x,y)$ be the cumulative probability of numbers in I between x and y, i.e.,

$$I(x,y) = \begin{cases} \displaystyle\sum_{x<z\leq y} I(z) & x \leq y \\ \displaystyle\sum_{\substack{s_I\leq z\leq y \\ x<z\leq e_I}} I(z) & x > y \end{cases}$$

The protocol involves the following two kinds of queries:

1. The server sends two numbers, a and b, on the distribution circle, to the client (instead of just a prefix) and asks whether the desired number x lies between those two on the circle.
2. The server sends a number to the client. The client responds positively if the number is x, or negatively otherwise.

It is shown that for any distribution, either there is at least one point with the probability not less than $\frac{1}{4}$ or two points can be found in the distribution circle such that the cumulative probability of numbers between them are in the range $\frac{1}{2} \pm \frac{1}{8}$. This will lead to a better reduction of the distribution space.

Theorem 2

For any distribution I, there exist either two points x and y such that $\frac{1}{2} - \frac{1}{8} \leq I(x,y) \leq \frac{1}{2} + \frac{1}{8}$, or x such that $I(x) \geq \frac{1}{4}$.

Proof

Let $x_0 < x_1 \ldots < x_k$ be the numbers in I. Suppose $I(x_i) < \frac{1}{4}$ for all $0 \leq i \leq k$. Let j be the smallest number such that $I(x_0, x_j) > \frac{1}{2} + \frac{1}{8}$. Obviously, $\frac{1}{2} - \frac{1}{8} \leq I(x_0, x_{j-1}) \leq \frac{1}{2} + \frac{1}{8}$.

The Protocol

- Set I to $[0..2^n - 1]$,
- Repeat the following steps until I has only one member:
  - Set probability space to I,
  - Set a to the first element of I and let b be the largest number such that $I(a,b) < \frac{1}{2}$.
  - Let $I' = [a..b]$. If $I(a,b) < \frac{1}{2} - \frac{1}{8}$, then:
    * Consider c, the number next to b on the distribution circle of I, and let $I' = [a..c]$.
    * If $I(a,c) > \frac{1}{2} + \frac{1}{8}$, then:
      · Server sends c to the client.
      · If the client responds negatively, delete c from I, else x is found and halt.
  - Otherwise, the server sends the start and the end points of I' to the client. If the client responds positively, set I to I', else $I = I - I'$.

The following procedure is used to find b in the above protocol:

- Let b be the first element of I.
- Set $x = \frac{1}{2}$.
- Delete right-hand bits of b until the adjusted probability of black box using the new probability space (I) in answer to b is more than x. In each step, if the deleted bit is 1, add the adjusted probability of $b0$ to x.
- Repeat the following until length of b becomes n:
  - If probability of $b0$ in I is more than x, $b = b0$. Otherwise, subtract the adjusted probability of $b0$ from x and let $b = b1$.

Note that for I only the first, last and exceptional points are saved. It can be seen that the number of exceptional points (deleted ones) will not exceed $O(H(D))$. Therefore, the complexity of the last procedure will not be more than $O(n)$.

Theorem 3

The above protocol is a $[O(nH(D)), 1.47H(D) + 4, O(nH(D))]$-protocol.

Proof

The main loop of the algorithm will be repeated at most $H(D)$ times. In the above procedure, $O(n)$ queries are used, each time it is called to find b. Therefore, the complexity in the server is $O(nH(D))$. For the same reason, the number of bits sent from server to client will be $O(nH(D))$.

Now, for proof of the complexity of the client, assume that the expected number of bits sent by the client is $f(D)$. Induction on data items in D is used. The protocol tries to find an item with probability of α (greater than $\frac{1}{4}$). In the case that this item is found, one query is sent to the client. With probability of α, the client sends a positive response and the protocol halts, otherwise this item is deleted from the distribution and the protocol is continued with a new distribution D'. Therefore, for this case,

$$f(D) = \alpha \times 1 + (1-\alpha)(f(D')+1).$$

This leads to $f(D) \leq \frac{1}{4} + \frac{3}{4}f(D') + \frac{3}{4}$. Using induction (assuming $f(D') \leq 1.47H(D') + 4$),

$$f(D) \leq 1 + \frac{3}{4}(1.47H(D') + 4).$$

It is known that $H(D') = \frac{4}{3}(H(D) - \frac{1}{2})$, because an item with a probability greater than $\frac{1}{4}$ has been eliminated and the probabilities are adjusted. Thus,

$$\begin{aligned} f(D) &\leq 1 + \frac{3}{4}(1.47 \times \frac{4}{3}(H(D) - \frac{1}{2}) + 4), \\ f(D) &\leq 1.47H(D) + 4. \end{aligned}$$

For a case in which such an item is not found, the server divides the space with an accuracy equal to $\frac{1}{2} \pm \frac{1}{8}$ and continues on with one of the portions after one query to the client and the client response. Therefore, the probability space is reduced by at least $\frac{5}{8}$, with the client sending just one bit. Using the same approach as presented for the proof of the first protocol, it can be easily shown that:

$$f(D) \leq \frac{1}{\log \frac{8}{5}} H(D) + 4 \approx 1.47H(D) + 4.$$

Notice that since the results are the same for both of these cases, the induction is correct in the general case.

REACHING THE LOWER BOUND

The protocol can be improved by dividing the space into two portions of $\frac{1}{2} \mp \frac{1}{2k}$ or finding an item with probability of $\frac{1}{k}$, for any fixed k. The overall protocol remains the same and the proved theorem holds for the general case as well. The analysis of this protocol is also similar. For the first case of the algorithm,

$$\begin{aligned} f(D) &\leq \frac{1}{k} + \frac{k-1}{k}(f(D')+1), \\ &\leq 1 + \frac{k-1}{k}(\beta H(D') + k), \\ &\leq 1 + \frac{k-1}{k}(\beta \times \frac{k}{k-1}(H(D) - \frac{\log k}{k}) + k), \\ &\leq \beta H(D) + k, \end{aligned}$$

where $\beta = \frac{1}{\log \frac{2k}{k+1}}$. For the second case, the same complexity is obtained as well. Thus, this protocol has the complexity of $[O(nH(D)), \beta H(D) + k, O(nH(D))]$. Increasing k (for large n), β will quickly reach 1, which yields to reaching the lower bound. This is achieved at the expense of additional $O(H(D))$ cost for the server. The number of bits sent by the server, however, can be reduced if the common prefixes of strings a and b sent to the client in some phases are not forwarded.

OPEN PROBLEMS

According to Orlitsky [7], in every protocol, the average number of bits transmitted between server and client cannot be less than n. In other words, the number of bits sent from server to client can only reach $O(n)$ from $O(nH(D))$. The following open problems are very attractive:

1. Is there any protocol in which exactly $H(D)+1$ bits are sent from client to server and the number of bits sent from server to client remains less that $O(n)$?
2. If the cost of sending one bit from server to client is α and that from client to server is β, provide a protocol that minimizes the total cost of communication.

CONCLUSIONS

In this paper, a new protocol is presented for sending n-bit of data from a client with limited resources to a server using an asymmetric communication channel. It is assumed that the line bandwidth from server to client is much more than that of client to server. The goal is to use the high-speed link to reduce the number of bits needed to be sent on the low-speed link to its lower bound limit. The protocol presented here is an extension of the protocol by Adler and Maggs [1]. They have presented a protocol in which the number of bits sent from client to server is $1.71H(D) + 1$, where the string to be sent by the client is assumed to be drawn from an arbitrary distribution D and $H(D)$ is the binary entropy of D. The protocol assumes that D is only known by the server. This work is extended and a new protocol is proposed which reduces the constant factor of 1.71 to 1.47 at the expense of a reasonable amount of extra work by the server. The protocol is further extended and 1.47 is reduced to $1+\epsilon$ for any small value of $\epsilon > 0$, which is very close to the lower bound of $H(D)+1$. Two open problems are also presented. Further work on this subject is underway.

REFERENCES

1. Adler, M. and Maggs, B.M. "Protocols for asymmetric communication channels", *Proceedings of FOCS'98* (1998).
2. Shannon, C.E. "A mathematical theory of communication", *Bell System Technical Journal,* **27**, pp 379-423 and 623-656 (1948).
3. Huffman, D.A. "A method for the construction of minimum redundancy codes", *Proceedings of IRE,* **40**(10), pp 1099-1101 (1952).
4. Orlitsky, A., Naor, M. and Shor, P. "Three results on interactive communication", *IEEE Trans. on Information Theory,* **39**(5), pp 1608-1615 (1993).
5. Orlitsky, A. "Worst-case interactive communication I: Two messages are almost optimal", *IEEE Trans. on Information Theory,* **36**(5), pp 1111-1126 (1990).
6. Orlitsky, A. "Worst-case interactive communication II: Two messages are not optimal", *IEEE Trans. on Information Theory,* **37**(4), pp 995-1005 (1991).
7. Orlistky, A. "Average-case interactive communication", *IEEE Trans. on Information Theory,* **38**(4), pp 1534-1547 (1992).
8. Yao, A.C. "Some complexity questions related to distributive computing", *11th ACM Symposium on Theory of Computing,* pp 209-213 (1979).