

Content Distribution for Seamless Transmission

E. G. Coffman, Jr.*

Andreas Constantinides*

Dan Rubenstein*

Bruce Shepherd†

Angelos Stavrou*

1. INTRODUCTION

We introduce a new paradigm in information transmission, the concept of SEAMLESS TRANSMISSION, whereby any client in a network requesting a file starts receiving it immediately, and experiences no delays throughout the remainder of the downloading time. This notion is based on the partial caching concept [2] which was introduced to overcome some of the disadvantages of traditional cache replacement algorithms such as LRU and LRU-threshold [1]. The main idea of partial caching is to store an initial part of the file in the cache and to obtain the rest of the file from the origin server. To achieve the maximal retrieval performance of seamless transmission, clients must be prepared to re-sequence segments of the files received out of order. With this caveat, seamless transmission can be viewed as a way to implement strict quality of service (QoS) guarantees to all clients of a network. This paper gives a provably correct technique for achieving seamlessness for a given file located at the root node in a tree structured network.

The SEAMLESS TRANSMISSION PROBLEM has interdependent, combinatorial *allocation* and *scheduling* subproblems. The solution to the allocation problem prescribes the subsets of file segments to be stored in different nodes in the network, and the solution to the deterministic scheduling problem specifies transmission schedules for each node based on the given allocation of file segments. In the systems being modeled, cache capacity is at a premium, so our goal is to find solutions that minimize required cache capacity without sacrificing seamless transmission.

2. THE SINGLE FILE PROBLEM ON TREE NETWORKS

A tree network $T = (V, E)$ has a root node r that stores the file F under consideration, and functions as an origin server for the file. The file $F := \{1, 2, \dots, M\}$ is a set of M distinct *units*, each having an amount of data that a node $u \in V$ can send out on any link incident to it in one time unit. Node $v \neq r$ has a cache with capacity $C_v \geq 0$ units and contents denoted by S_v . Our focus is on computing *static* cache allocations: S_v does not change over time. Units from F will be transmitted through the tree network to a subset of the nodes (*clients*) requesting the file. In what follows, $p(v)$ is the *parent*, or upstream neighbor, of $v \neq r$; v 's children are v 's *downstream* neighbors; T_v is the subtree of T rooted at v . REQUESTS: A *request* specifies a client node v , and the time $t_v < \infty$ when the request is made. A node v can make at most one request, and we set $t_v = \infty$ if it makes no request. Seamless transmission requires that, for each request, we ensure that its client

node v has received all units of F by time $t_v + |F|$. These units may be stored locally in v 's cache, or they may need to be retrieved from upstream. Without loss of generality, we assume that there are no 'useless' subtrees, i.e., a subtree rooted at a node v for which $t_u = \infty$ for all $u \in T_v$.

TRANSMISSION: Nodes in the network start transmitting units of data at integer times t . The transmission delay along edge (u, v) is an integer δ_{uv} . For convenience, $\delta_v \equiv \delta_{vp(v)}$. If v starts sending a unit to u at time t , then u starts receiving it at time $t + \delta_{vu}$. It is convenient to append to each node v for which $t_v < \infty$ a downstream neighbor (leaf) w with $\delta_w = 0$. This distinguishes the set of clients as precisely the leaves of the tree without changing the problem under consideration.

Only file requests are transmitted upstream, and only file units are transmitted downstream. A *transmission step* for node v consists of an *upstream* and *downstream* part. For $v \neq r$, $\Phi_v(t)$ denotes v 's *upstream transmission* at time t . Such a transmission corresponds to sending a request for the file to v 's parent $p(v)$. As such, $\Phi_v(t)$ can be viewed as containing a set of requests. The system operates according to the following model.

1. Each leaf v sends its request to $p(v)$ immediately, and when a non-leaf node v receives a request from a child, it immediately passes it upward.
2. A *downstream transmission* step at v at time t specifies what units to send to v 's downstream neighbors. Let $U_v(t)$ be the unit that v received from upstream at time t . Also, for a downstream neighbor, say w , of v , let $D_v(w, t)$ denote the unit transmitted to w at time t . "At time t " here means that the transmission or reception is complete at time t , and took place in the interval $(t - 1, t]$. Thus, for each v , $U_v(t) = D_{p(v)}(v, t - \delta_v)$. If $v_1, \dots, v_c$ are the downstream neighbors of v , downstream transmission at time t is a c -tuple $(u_1, \dots, u_c)$, where $u_i := D_v(v_i, t)$ is either an element of the cache or has just been received from upstream. Formally, for each node v and each child v_i of v , $D_v(v_i, t) \in S_v \cup U_v(t) \cup \{\epsilon\}$ for all t . The null value $D_v(v_i, t) = \epsilon$ means that no unit is transmitted to neighbor v_i at time t .
3. Each node can transmit to any number of downstream neighbors simultaneously, but it can not transmit to any specific downstream neighbor until it receives a request from it. Denote by t_v^* the time at which an internal node v receives its first request from downstream. (For leaf nodes v , set $t_v^* = t_v$). Then for each node v and each child w of v , we have $D_v(w, t) = \epsilon$ unless $t > t_w^* + \delta_w$.

We note that, for a given set of file requests, the times t_v^* are all easily computable recursively starting at the bottom of the tree and

*Department of Electrical Engineering, Columbia University, New York 10027

†Bell Labs, Lucent Technologies, Murray Hill, NJ 07974

using $t_v^* = \min_{1 \leq i \leq c} \{t_{v_i}^* + \delta_{v_i}\}$.

In an instance of our seamless transmission problem, we are given a transmission tree $T_r = (T, \delta, C)$, where δ is a vector giving the link delays, and C is a vector giving the cache capacities, and a file F to be transmitted. We are also given a collection of requests (v, t_v) . The output for a given instance consists of an *allocation-schedule* pair that achieves seamless transmission: For each client v making a request, v receives at every time $t = t_v + 1, \dots, t_v + |F|$ a unit of the requested file F which is different from those already received.

The solution specifies schedules for the upstream and downstream transmissions at each node v at each time t which satisfy (i) for every v , $|S_v| \leq C_v$ and (ii) the operational scheme enumerated in the last section.

3. THE PROBLEM WITH SIMULTANEOUS REQUESTS

We now specialize to instances of the problem where every request occurs at time 0 (i.e., $t_v = 0$ or ∞ for every v). Thus our focus is on the time period 0 through $|F|$. For each node v , let A_v denote the set of clients in T_v . In the simultaneous-request case, we can significantly narrow the time interval of interest: For each node v and each child w of v , we have $D_v(w, t) = \epsilon$ unless $t_w^* + \delta_w + 1 \leq t \leq M - (t_w^* + \delta_w)$.

Since each client w has 0 delay from its parent $p(w)$ and can thus receive units from upstream immediately, we choose $S_w = \emptyset$. Using now the above constraint and the fact that for each client w , $t_w^* = \delta_w = 0$, the transmission is seamless if for every client w of the tree

$$\bigcup_{1 \leq t \leq M} \{U_w(t)\} = F \quad (1)$$

We now present a solution for this version of the seamless transmission problem. Let P be the path between v and u , and denote the one-way communication delay between nodes v and u as $d(v, u) = \sum_{x \in P-v} \delta_x$. It follows that in the simultaneous-request problem, $t_v^* = \min_{w \in A_v} \{d(v, w)\}$. In addition, to avoid trivialities, we make the ‘large file’ assumption: $|F| > 2 \cdot \max_{w \in A_r} \{d(r, w)\}$.

A SOLUTION. In the scheme of section 2, for each $w \in A_v$, $\Phi_v(d(v, w))$ contains the file request from client w . Assume that $r_1, \dots, r_b$ are the downstream neighbors of r . Since r stores the whole file and can send any unit to any of its downstream neighbors, let us concentrate on any r_j , $1 \leq j \leq b$. Note here that r receives the first request from a client $w \in A_{r_j}$ at time $t = t_{r_j}^* + \delta_{r_j}$, and from each $w \in A_{r_j}$ at time $t = d(r, w) = d(r_j, w) + \delta_{r_j}$. Consider what the contents of the cache of each node v should be. The seamless transmission constraint requires each node $v \in T_{r_j}$ to store enough units in its cache to cover at least the two-way communication delay with its parent $p(v)$. In other words, each node v needs to have $|S_v| \geq 2\delta_v$. For ease of exposition, think of v ’s cache as being composed of two disjoint sub-caches: S_v' which stores exactly enough units to cover the two-way communication delay with $p(v)$, and S_v'' which stores all other necessary units required by clients in T_v to achieve seamless transmission. We are now in position to describe an allocation-schedule pair for which we can prove seamless transmission.

THEOREM 3.1. *Seamless transmission to all clients $w \in A_{r_j}$ is achieved by the following scheme. The allocation policy is defined*

by $C_v \geq |S_v| = |S_v'| + |S_v''|$, where $|S_v'| = 2\delta_v$, and

$$|S_v''| = \max \left\{ 0, \min \left\{ 2d(r, p(v)), \max_{1 \leq i \leq c} \{t_{v_i}^* + \delta_{v_i}\} - (t_v^* + 2\delta_v) \right\} \right\}, \quad (2)$$

$$S_v' = \{M - 2d(r, v) + 1, \dots, M - 2d(r, v) + 2\delta_v\}, \quad (3)$$

$$S_v'' = \{U_v(t_v^* + 2\delta_v + 1), \dots, U_v(t_v^* + 2\delta_v + |S_v''|)\}. \quad (4)$$

Under the above allocation policy, the downstream transmission schedule for r is defined by

$$D_r(r_j, t) = \begin{cases} \epsilon & t \leq t_{r_j}^* + \delta_{r_j}, \\ M - 2 \cdot (t - 1) & t_{r_j}^* + \delta_{r_j} + 1 \leq t \leq \lceil (M/2) \rceil, \\ 2 \cdot t - M - 1 & \lceil (M/2) \rceil + 1 \leq t \leq M - (t_{r_j}^* + \delta_{r_j}), \\ \epsilon & M - (t_{r_j}^* + \delta_{r_j}) + 1 \leq t \leq M. \end{cases} \quad (5)$$

For a node $v \in T_{r_j}$ define the following two index sets

$$\begin{aligned} \Lambda_v' &= \{t_v^* + \delta_v + 1, \dots, t_v^* + \delta_v + 2 \cdot d(r, p(v))\} \\ \Lambda_v'' &= \{t_v^* + \delta_v + 2 \cdot d(r, p(v)) + 1, \dots, M - (t_v^* + \delta_v)\} \end{aligned} \quad (6)$$

Then, the downstream transmission schedule for node v is

$$\begin{aligned} D_v(v_i, t) &= \epsilon, & t \leq t_{v_i}^* + \delta_{v_i}, \\ \bigcup_{t \in \Lambda_{v_i}'} \{D_v(v_i, t)\} &= \{M - 2 \cdot d(r, v) + 1, \dots, M\}, \\ D_v(v_i, t) &= D_r(r_j, t - d(r, v)), & t \in \Lambda_{v_i}'', \\ \bigcup_{t \in \Lambda_{v_i}''} \{D_v(v_i, t)\} &= \{1, \dots, M - 2 \cdot (t_{v_i}^* + \delta_{v_i} + d(r, v))\}, \\ D_v(v_i, t) &= \epsilon, & M - (t_{v_i}^* + \delta_{v_i}) + 1 \leq t \leq M, \quad 1 \leq i \leq c. \end{aligned} \quad (7)$$

A proof that the downstream transmission schedule for node v is as described by (7) can exploit the inductive structure of trees. Then, using (7) we can show that seamless transmission is achieved by each client node $w \in A_{r_j}$.

4. CONCLUDING REMARKS

Multiple-file problems, structures more general than trees, and non-simultaneous requests are among the many directions for future research that we are pursuing. A more general problem for which we have results leaves the requesting nodes unspecified; the solution must provide seamlessness for every subset of clients. Note that the number of units a node v must store in its cache is $2\delta_v \leq |S_v| \leq 2 \cdot d(r, v)$. An important future question would be how much $|S_v|$ can be decreased and still achieve seamless transmission if there are $k > 1$ download channels, instead of the single channel assumed here.

5. REFERENCES

- [1] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox. Caching proxies: Limitations and potentials. In *Proceedings of 4th International World Wide Web Conference*, pages 119–133, Boston, USA, December 1995.
- [2] S. Sen, J. Rexford, and D. F. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of IEEE INFOCOM*, pages 1310–1319, New York, March 1999.