Cedar: An Optimized Solution for P2P Video Multicast

Elie Gabriel Mora, Claudio Greco, Béatrice Pesquet-Popescu, Marco Cagnazzo, Joumana Farah

TELECOM-ParisTech, TSI department
46 rue Barrault, F-75634 Paris Cedex 13
Paris, FRANCE

{mora,greco,cagnazzo,pesquet}@telecom-paristech.fr
joumanafarah@usek.edu.lb

***Abstract*—Video multicast is an important application for P2P networks, which can benefit from the load repartitioning and large scale distribution properties they have to offer. In this context, the Orchard algorithm was developed in 2007, with an aim to build a P2P system meeting video multicast requirements. However, Orchard suffers from a slow overlay construction and a potentially high end-to-end delay. In this work, we propose a new algorithm that we call Cedar. Based on Orchard, it integrates original functionalities to speed-up the tree construction and minimize end-to-end delay. Results showed that Cedar fulfilled its requirements as the tree construction speed was increased by at least 64%, and the end-to-end delays reduced to a maximum of 5 seconds.**

***Index Terms*—Distributed algorithms, Multiple description coding (MDC), Peer-to-peer (P2P), Application Lever Multicasting (ALM), Resource reciprocation.**

I. Introduction

Multimedia content distribution has been a widely investigated topic over the last few years. Traditionally, multimedia content has been delivered using the client/server paradigm, which is being abandoned as a server alone could not be able to handle a large number of clients. Nevertheless, the client/server paradigm is still employed; *e.g.*, YouTube uses it for low popularity videos [1].

Content Distribution Networks (CDNs), are also used for distributing multimedia content [2]. They inherit the client/server model, but they employ multiple servers interconnected through a dedicated infrastructure, with the aim of partitioning the large load [3]. YouTube[1], for instance, employs the content distribution networks of Akamai to deliver popular videos to users.

Nowadays, a new paradigm for multimedia content distribution is emerging: video multicast. In this context, two types of multicast are considered: IP multicast and Application Level Multicast (ALM). IP multicast takes advantage of the implementation of multicast functionalities in the network layer, as at these levels the network topology is best known.

Unfortunately, IP multicast suffers from scaling issues and a lack of support for higher layers functionalities. It also requires costly infrastructural changes, since the routers are to integrate IP multicast [4]. ALM is thus favored for its low cost and its simplicity of implementation, which does not require any infrastructural changes, as the routers need only to support IP unicast.

Peer-to-peer (P2P) networks implement application level multicast and can therefore serve as a distribution infrastructure for multimedia content. Using P2P to distribute multimedia content has been investigated, for instance, within the popular P2P community BitTorrent[2] leading to the apparition of BitTorrent Live Streaming, still under development.

Several issues however limit the deployment of multicast video streaming on P2P networks. Common problems include *freeriding*, in which peers do not relay the streams they receive thus limiting the multicast distribution, and *churn*, in which high peer arrival and departure rates destabilize the system. In addition, video streaming can be quite resource consuming for users with limited bandwidth.

Within this context, Mol *et al.* developped in 2007 the Orchard algorithm to deal with these issues [5]. Orchard used a form of tit-for-tat, popular in BitTorrent community [6], in order to minimize free-riding. It also assumed that a Multiple Description Coding technique (MDC) was used to encode the stream, in order to guarantee a robust and flexible system [7].

Orchard builds multicast trees to ensure the video distribution, placing each peer as a relaying node in different trees, each corresponding to a description. And although Orchard ensures that most peers will receive all the descriptions, it does not account for connection time (time taken by a peer to be part of all the trees) and end-to-end delays (time taken by a video stream to travel from the source, down the ALM tree, and up to a particular peer). In addition, Orchard assumes peers to be naturally cooperative, which is not always the case in a real life scenario. Cooperation can indeed be achieved, providing that a proper incentive scheme is producted [8]. Finally, Orchard fails to handle pathological topologies in which a node fails to join the network in a reasonable time. Even though in a theoretical analysis the incidence of such topologies may be negligible, they must be dealt with when providing a commercial service,

In this work, we propose a new protocol based on Orchard: the Cedar protocol. Cedar implements primitives to achieve multi-tree construction abiding by low-latency constraints, while handling pathological topologies that may lead to a denial of service. In addition, even though the actual implementation of an incentivation scheme is out of the scope of

[1]Website: http://www.youtube.com/

[2]Website: http://www.bittorrent.com/

this paper, it relies on such an incentivation scheme, allowing us to introduce new types of deals which improve availability and average latency.

The rest of this paper is organised as follows: In section II, we present the Orchard algorithm and its different functionnalities, while underlining its inherent weaknesses. In section III, we describe the proposed functionnalities added to Orchard, yielding the Cedar algorithm. Then, in section IV, we present the experiments and results that show the multi-tree construction speed increase and the reduction in end-to-end delays offered by Cedar in comparison with Orchard. We will conclude this paper with section V while underlining the possibilities for future work.

II. The Orchard Algorithm

This section aims at presenting the basics of the Orchard algorithm. First, we will talk about general primitives, then we will see how the different peers collaborate with each other to obtain streams. Finally we will present an evaluation of Orchard where we expose its principal weaknesses.

A. Orchard Basics

Orchard assumes that Multiple Description Coding [7] is used at the encoder side. The technique aims to divide a stream into multiple sub-streams, also called descriptions. The more descriptions a peer downloads, the better the resulting video quality will be, though only one received description is enough to provide a minimal quality. Note that each description can take a completely different route from the others to arrive to a given peer, hence creating multiple distribution trees (each tree corresponding to one description) or multi-tree.

In Orchard, descriptions are designated by colors. The Source for instance has all the descriptions, and therefore, it has the color white. In contrast, a peer that joins the system has initially no descriptions and is therefore blank. Note that the first color obtained by a peer will become its own. Even if a peer has more descriptions, it can only distribute the description corresponding to its own color.

A peer will always try to maintain a neighbor set of m neighbors, selected randomly by a server. In addition, it stores a list of its neighbors' properties, which include their respective colors. Hence, a peer knows exactly which neighbor to ask to obtain a description which it does not already have.

B. Strikable Deals

The following are the 3 types of deals that a peer can strike in order to obtain the different descriptions:

1) The *Join* deal: If a peer has the source as its neighbor, it could try to strike a *Join* deal, where it simply asks the source for a particular description. If the request meets several conditions, the source will accept and reply with the requested description, without expecting anything in return. This is the only form of free-riding tolerated in the system.
2) The *Exchange* deal: If two peers have different colors and require each other's color, they can simply exchange their descriptions.
3) The *Redirection* deal: Redirections are always based on an *Exchange*. A peer exchanging descriptions may receive a redirection request, upon which – if it accepts – it redirects its outgoing substream through the requesting peer.

A more detailed explanation and a number of illustrative figures can be found in [5].

C. Evaluation of Orchard

Even with the redirection deals and the relaxation offered by the redirections through colored peers, deadlock situations are still possible, forcing a peer to spend precious time searching for collaborators. This results in a slow multi-tree construction. In addition, intricate exchange and redirection deals resulting from this lack of flexibility, push peers farther and farther away from the Source. A video stream would thus be relayed through many intermediate peers before arriving to its destination, hence the high end-to-end delays experienced in Orchard. Finally, even if the free-riding problem is minimized in Orchard, it still remains an issue to be adressed in *Join* deals. A peer should always repay the Source with something, even if it is not a description.

III. Proposed Functionnalities

We have implemented four new functionnalities to accelerate the multi-tree construction and reduce end-to-end delays: *Transferable Joins*, *Super Peers*, *Redirection-to-Exchange Transformation*, and the *Special Redirection*. The aim was to inject more flexibility in the system to allow a peer to quickly find peers to collaborate with. Further more, if exchange-redirections loop formations are avoided, and a more horizontal multi-tree configuration is obtained, end-to-end delays will be reduced.

A. Transferable Joins

To reduce the free-riding issue in the *Join* deal, the *Transferable Joins* functionnality forces a peer getting a description from the source to commit itself to provide this description to another peer upon request. While in Orchard, the number of available slots that permitted direct connections to the server was limited, in Cedar, using the *Transferable Joins* functionnality, we always have the same number of free slots at any time in the system, since a peer that reserves a slot offers his neighbors a new one to connect to. This effectively avoids system saturation.

B. Superpeers

Superpeers can be considered as bandwith resourceful proxy servers, delivering connections to other peers. Hence, once they have all the descriptions, they can accept *Join* deals (which are usually addressed to the source) and thus deliver free descriptions to the requesting peers. This functionnality allows to have hierarchical levels of distribution, where proxy servers negotiate with the main server or source (top level) and clients collaborate with those superpeers (bottom level). The resulting model is more representative of the reality of

the topology and can be easily integrated in the framework of P4P [9]. To avoid free-riding, a peer can give the superpeer a high trust level, increasing the popularity of that superpeer in a reputation management system like the one in [10].

C. The Redirection-to-Exchange Transformation

In Orchard, an *Exchange* could not be redirected more than once, to avoid redirection chains that implied latency increases. However, if a situation occurs where an exchange is redirected in both ways through the same peer, then we can consider that the peer getting the redirections is in fact *exchanging* descriptions with the peers it is getting the descriptions from. The advantage is that those exchanges, by definition, can be redirected through other newly joined peers.

D. The Special Redirection

The *Special Redirection* functionnality forces a peer getting a certain color through a redirection to ask for the other color being exchanged to the peer delivering it in that *Exchange* deal. This reduces considerably the search time for the wanted color. Plus, it will lead to a situation where an exchange is redirected, in both ways, through the same peer. Consequently, the combination of this functionnality with the *Redirection-to-Exchange Transformation* is extremely efficient in reducing the multi-tree construction delay as well as the end-to-end delays.

IV. Experimental Results

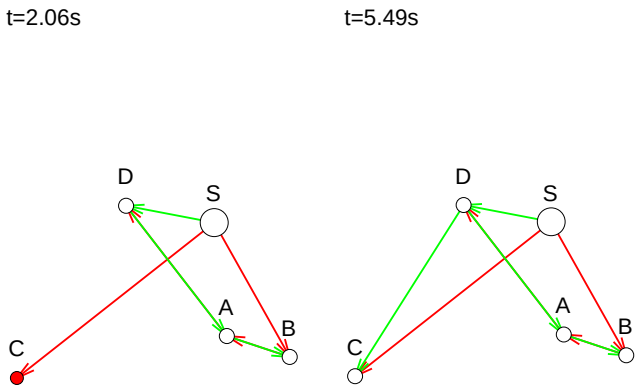
This section aims at evaluating each proposed functionnality compared to the original Orchard algorithm. We will show how the *Transferable Joins* and the *Superpeers* functionnalities help avoid system deadlocks, and how the *Redirection-to-Exchange Transformation* and the *Special Redirection* functionnalities reduce the multi-tree construction and the end-to-end delays.

A. Results of the Transferable Joins and Superpeers functionnality

Figure 1 shows the flexibility offered by the *Transferable Joins* functionnality. In this scenario, the server is saturated and peer C has still not got the green color. No available options exist in Orchard, since D and B have already redirected their exchanges through A. This deadlock situation is resolved using the *Transferable Joins* functionnality, since in that case, C can simply ask D for the transfer of its *Join* deal for the green color. Figure 2 shows the flexibility offered by the *Superpeers* functionnality. The scenario is the same as in Figure 1. The deadlock situation in Orchard is portrayed in Figure 2(a) where peer C does not succeed in getting the green color. This is solved with the *Superpeers* functionnality implemented, as shown in Figure 2(b) where peer C struck a *Join* deal with superpeer A (as if it was the Source) to get the green color.

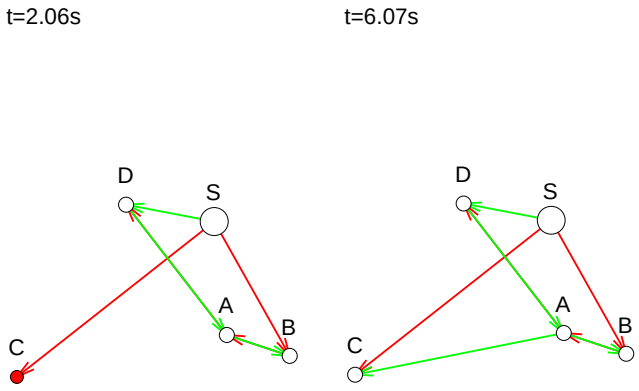
B. Results of the Redirection-To-Exchange and Special Redirection functionnalities

As shown in Figure 3(b), the combination of those functionnalities allows the protocol to maintain a high percentage



(a) Orchard (b) Orchard + *Transferable Joins*

Figure 1. Final network state. No changes occur after $t = 2.06$ [s] in Orchard and after $t = 5.49$ [s] in Orchard + *Transferable Joins*

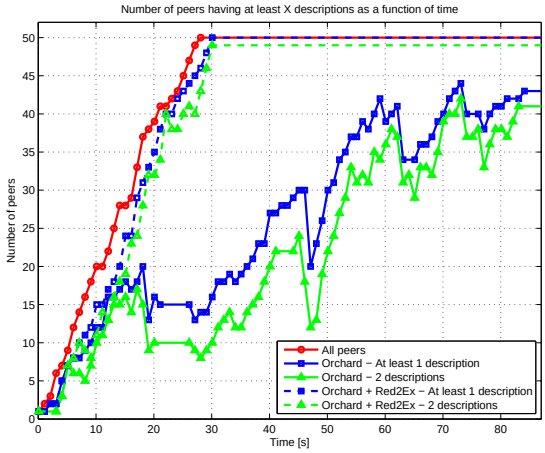


(a) Orchard (b) Orchard + *Superpeers*

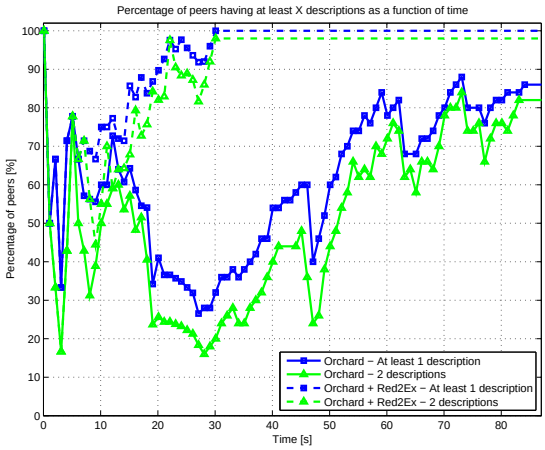
Figure 2. Final network state. No changes occur after $t = 2.06$ [s] in Orchard and after $t = 6.07$ [s] in Orchard + *Superpeers*

of peers having at least one or exactly two descriptions over time. In the first 12 seconds, the performances are comparable since no situation of an exchange being redirected in both ways through the same peer arises (the peers have not had the time yet to establish the required *Exchange* and *Redirection* deals). Afterwards however, with the help of the *Special Redirection* functionnality, the *Redirection-to-Exchange Transformation* created more opportunities for other peers to get their colors, hence the performance gap with the original Orchard protocol.

Furthermore, the flexibility offered by the *Redirection-to-Exchange Transformation* and the *Special Redirection* functionnalities reduces the time it takes a peer to get its first and second description, as shown in Figure 4 and Figure 4(b) respectively. While in Orchard, the distribution of the corresponding "0 to 1" and the "1 to 2" delays is widespread over the range of 42 and 58 seconds respectively, with the two aforementionned functionnalities, the delays are constrained to the 0–5 [s] interval. Note that the stem at $t = 75$ [s] represents the peers that never got their first or second description due to a deadlock situation.

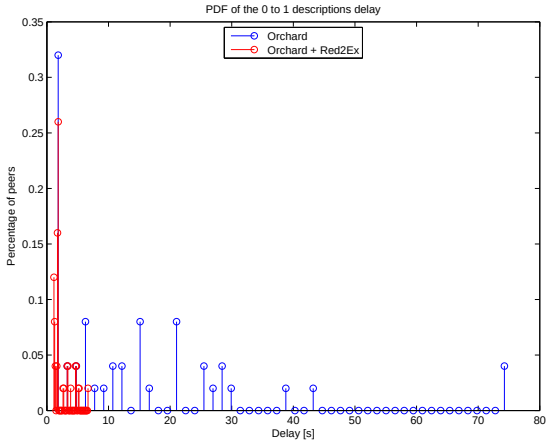


(a)

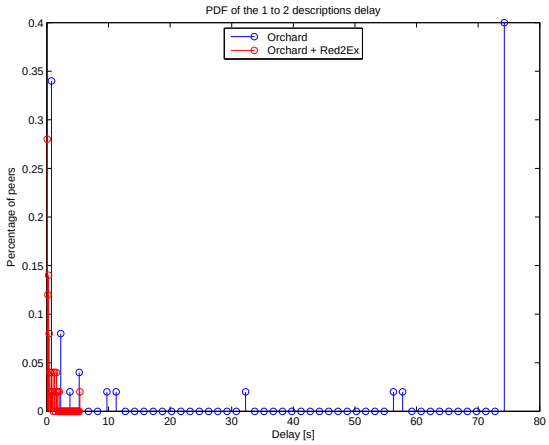


(b)

Figure 3. Number and percentage of peers having at least one or two descriptions over time, with and without the *Redirection-to-Exchange Transformation* and the *Special Redirection* functionnalities.



(a)



(b)

Figure 4. Probability Distribution Functions of the “0 to 1” and “1 to 2” delays in Orchard, with and without the *Redirection-to-Exchange Transformation* and the *Special Redirection* functionnalities.

V. Conclusion and Future Work

In this paper, a new algorithm for P2P video multicast was introduced. It enhances the Orchard algorithm with four new functionnalities that help to avoid deadlock situations, reduce free-riding and most importantly reduce multi-tree construction and end-to-end delays as shown by our experiments.

In the future, we could address another problem in Orchard: the random selection of neighbors for a peer, which may force it to collaborate with geographically distant peers hence increasing end-to-end delays. This could be countered by implementing some form of Network Awareness, to build an overlay network which takes into account the geographical network topology.

References

[1] R. Santos, B. Rocha, C. Rezende, and A. Loureiro, “Characterizing the YouTube video-sharing community,” White Paper, 2008. [Online]. Available: http://security1.win.tue.nl/~bpontes/pdf/yt.pdf

[2] B. Krishnamurthy, C. Wills, and Y. Zhang, “On the use and performance of Content Distribution Networks.” ACM SIGCOMM, 2001, pp. 169–182.

[3] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, “Distributing streaming media content using cooperative networking,” in *NOSSDAV ’02: International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM Sigcomm, April 2002.

[4] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, “Deployment issues for the IP multicast service and architecture,” *IEEE Network*, vol. 14, pp. 78–88, 2000.

[5] J. D. Mol, D. H. P. Epema, and H. J. Sips, “The Orchard algorithm: Building multicast trees for P2P video multicasting without free-riding,” *IEEE T Multimedia*, vol. 9, no. 8, pp. 1593–1604, December 2007.

[6] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, “Influences on cooperation in BitTorrent communities.” ACM SIGCOMM, 2005, pp. 111–115.

[7] V. K. Goyal, “Multiple description coding: Compression meets the network,” *IEEE Signal Proc Mag*, vol. 18, no. 5, pp. 74 – 93, September 2001.

[8] J. Park and M. van der Schaar, “A game theoretic analysis of incentives in content production and sharing over peer-to-peer networks,” *IEEE J Sel Topics Signal Process*, vol. 4, no. 4, pp. 704–717, 2010.

[9] H. Xie, A. Krishnamurthy, A. Silberschatz, and R. Y. Yang, “P4P: Explicit communications for cooperative control between P2P and network providers,” White Paper, 2008. [Online]. Available: http://www.dcia.info/documents/P4P_Overview.pdf

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The Eigentrust algorithm for reputation management in P2P networks,” in *WWWC ’03: Twelfth International World Wide Web Conference*. IW3C2, 2003.