Multi-Client File Download Time Reduction from Cloud/Fog Storage Servers

Ahmed A. Al-Habob, *Student Member, IEEE,* Yousef N. Shnaiwer, *Student Member, IEEE,* Sameh Sorour, *Senior Member, IEEE,* Neda Aboutorab, *Member, IEEE,* and Parastoo Sadeghi, *Senior Member, IEEE*

Abstract—In this paper, we study the problem of reducing the download time of multiple files requested by multiple clients from multiple servers of cloud/fog storage servers. Given possible previous file downloads by the clients from the cloud/fog servers, network coding can be efficiently exploited to expedite the download process by taking advantage of this side information. Since each client can tune to only one server at a time, the sets of clients served by the different servers must be disjoint in order to guarantee a maximum reduction in download time. To accomplish disjoint download mechanisms, a dual conflict network coding graph is proposed. Given the intractability of the optimal solution, we propose an online algorithm to reduce the multi-client file download time using the designed dual conflict graph. For the special yet typical case of one file request per client per time epoch, both asymptotic lower and upper bounds of the performance of the proposed conflict-free algorithm are derived. Simulation results show that this proposed algorithm exhibits near optimum performance compared to the optimum solution, and a significant reduction in download time as compared to the per-server network coding scheme. Furthermore, imperfect feedback environment scenarios are investigated (i.e., when feedback loss event occurs). A maximum likelihood approach is employed at the server to estimate the network state, which can be then used by our proposed algorithm to reduce the download time in such scenarios.

Index Terms—File Download Time Reduction, Could Storage, Fog Radio Access Networks, Network Coding.

1 Introduction

In the last decade, an exponential growth of storage systems capacity has been witnessed as the need for storing e-mails, photos and videos increased. This growth encouraged the development of new data storage techniques in order to enhance the two major performance indicators of data storage systems, namely reliability and accessibility [2], [3]. To this end, cloud storage systems, in which files are stored in multiple server nodes with appropriate repetition (and possibly coding), emerged as a viable solution to improve the availability and reliability of storage systems [2], [4]. Different distributed storage strategies were proposed in the literature for this paradigm to improve the system reliability. One strategy is full replication of the same data in multiple storage units, which imposes extremely high storage overhead. Another strategy that can provide better redundancy-reliability trade-off is erasure/network coding designed to protect data against partial data loss events [4], [5], [6], [2], [3].

- *A.A. Al-Habob and Y. Shnaiwer are with the Department of Electrical Engineering, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia (e-mail: {g201301090, g201204420}@kfupm.edu.sa)*
- *S. Sorour is with the Department of Electrical and Computer Engineering, University of Idaho, Moscow, ID, USA (e-mail: samehsorour@uidaho.edu)*
- *N. Aboutorab is with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia (e-mail: n.Aboutorab@adfa.edu.au)*
- *P. Sadeghi is with the Research School of Engineering, The Australian National University, Canberra, ACT, Australia (e-mail: parastoo.sadeghi@anu.edu.au)*
- *This work is an extension to our paper [1] in ICC 2015.*

Unlike system reliability, the system accessibility problems (i.e. problems regarding clients of a network accessing and downloading their requested files in an efficient manner) were less studied in the literature. A connectivity model with tree structure, consisting of a queue connected to a set of storage servers that receives client requests and schedule their download, was proposed in [7]. Despite the merits of this approach, it did not consider two practical aspects of cloud storage systems. First, the point-to-multipoint model considered in this work limits the potentials of cloud storage servers to operate in a multipoint-to-multipoint fashion. Indeed, the multiple requests of different clients can be served simultaneously from the different cloud servers. Second, it ignores the possibility of prior file downloads by the clients from the cloud servers. These prior downloaded files, if exist, can be used as side information to significantly reduce the download time of the current client requested files using network coding.

On another note, the wide spread use of the smart phones has resulted in a gigantic burst in heavy-traffic demand (mainly video content) from 4G cellular base-stations. The expected massive growth in such demands cannot be tolerated with the current cellular network architectures. This challenge led to the emergence of fog radio access networks (F-RANs), as a very potential candidate architecture for 5G cellular networks [8], [9], [10]. F-RANs were inspired by recent studies showing both a very high and very slow varying temporal correlation among the video traffic demanded by end-clients. This interesting finding motivated the idea of proactively (i.e. without user request) caching of such "popular" files in a "fog" of storage units close to the end-clients, which could be done with very low rates or in off-

peak times of the macrocell base-stations. The clients can thus access these files from these fog storage network thus offloading this kind of heavy-load traffic from the macrocell base-stations.

A large body of literature (e.g., [11], [12], [13], [14], [15] and references therein) has studied different caching policies of these popular files in the fog storage units in order to maximize the client hit probability (i.e. the probability of a client finding the desired files in an accessible storage unit). However, very limited works have addressed the scheduling problems of such file downloads when multiple clients request a set of different files from the same set of fog storage units. In [16], an attempt was made to optimize the file download schedule in order to offload the traffic drawn from the macro base-station. It also exploited the fact that "popular" must have been downloaded extensively by the different clients of the network to be deemed popular. These prior file download were thus used a side information to maximize the macrocell offloading using network coding. However, the problem of scheduling the client requests with the aim to minimize their download time only from the fog storage units (without the intervention of the macrocell base-station) was not considered. Moreover, possibilities of file corruption or acknowledgment losses due to the wireless channel impairments, and their effect on the scheduling process and download time were not investigated.

In this paper, we consider a cloud or fog storage system, in which files are be stored (or cached) in any arbitrary manner, whether fixed or randomized. Multiple clients of a network, with possible side information from prior file downloads, request new file downloads from these servers over possibly lossy wired or wireless channels. Each client can request one or more files to download, and can tune to only one server at time to download one full file from it. Scenarios where files are split into chunks, each of which stored (with or without coding) in different servers, can be captured into the above considered model, by assuming that each chuck in these scenarios corresponds to a full file in our model. Thus, a client requesting all the chucks of the same file from their storing servers corresponds to multiple full file requests by this client in the above model. Clients receiving corrupted files due to channel impairments will request their retransmission from the cloud/fog storage system. *The main aim of this study is to design efficient algorithms to minimize the download time of these multiple file requested by the multiple network clients from the cloud or fog storage servers, using network coding*.

Network coding (NC) emerged more than a decade ago as the concept of blending data flows at the transmitter or intermediate nodes in order to attain uttermost information usefulness [17]. One well-known subclass of NC, namely opportunistic NC (ONC) [18], [19] and its further well-investigated subclass, namely Instantly decodable NC (IDNC) [20], [21], [22], [23], [24], exploit the diversity of side information at the network's clients to generate optimized combinations of messages/packets/files that ensures their decodability at a subset or all the clients. IDNC differs from general ONC in that it does not buffer un-decodable packets to both simplify the encoding/ decoding processes and avoid adding extra buffers for the NC process [25], [22]. Furthermore, this simplification enabled the design of graph models for IDNC, which were widely used to optimize the coded transmissions in order to achieve different targets, such as minimizing the completion time of delivering a frame of packets [23], [24], minimizing the decoding delays [26], [27], minimizing received video distortion [28], [29], and speeding up cooperative data exchange in device-to-device (D2D) networks [30], [31].

However, most of the studies on IDNC considered a point-to-multipoint (PMP) model with one transmitter sending to multiple nodes. Even in the D2D scenarios, one device was elected as transmitter in each transmission. Applying the developed PMP IDNC techniques to our cloud/fog storage model may be inefficient. Indeed, each server can implement its own PMP IDNC algorithm, but this may result in situations where multiple servers are simultaneously serving the same client. This is considered not only a waste of resources as each client can listen to only one server at a time, especially in wireless scenarios, but a limitation on the overall set of servers in serving more clients.

To address this problem and achieve the aforementioned paper target (i.e. minimizing the total download time of files requested by multiple clients from cloud/fog storage servers), we first develop a new dual conflict IDNC graph representation that, not only represents coding conflicts due to side information, but also represents simultaneous transmission conflicts from multiple servers to one client. With this new graph representation, we formulate the file download time minimization problem as a stochastic shortest path (SSP) problem, which is shown to be intractable due to its exploding state and action spaces. We thus design a heuristic algorithm that employs the newly designed graph and client demands in order to reduce the file download time compared to the PMP version of IDNC. For the special yet typical case of one file request per client per time epoch, both asymptotic lower and upper bounds of the performance of the proposed conflict-free algorithm are derived. Simulation results show that this proposed algorithm exhibits near optimum performance compared to the optimum solution, and a significant reduction in download time as compared to the per-server network coding scheme. Furthermore, imperfect feedback environment scenarios (i.e., when file delivery acknowledgements are subject to loss) are investigated. A maximum likelihood approach is employed at the server to estimate the network state, which can be then used by our proposed algorithm to reduce the download time in such scenarios.

The remaining of this paper is organized as follows. Section 2 illustrates the system model. Section 3illustrates an example to motivate the need for conflict free network coded file downloads from cloud/fog storage systems. The novel dual conflict IDNC graph design is introduced in Section 4. Section 5 presents the SSP formulation of the problem and details the proposed heuristic to solve it. The asymptotic upper and lower bounds on the performance of our proposed algorithm are derived in section 6. The algorithm's extensions to imperfect feedback scenarios is illustrated in 7. Section 8 illustrate the simulation results, and Section 9 concludes the paper.

2 System Model and Parameters

2.1 Network and Data Model

The network model of interest consists of a set $\mathcal{S} = \{s_i\}_{i=1}^{N_s}$ of N_s cloud or fog storage servers, a set of $\mathcal{F} = \{f_k\}_{k=1}^{N_f}$ N_f files, and a set $\mathcal{C} = \{c_j\}_{j=1}^{N_c}$ of N_c clients. Each server s_i stores a subset $\mathcal{H}_{s_i}$ of $\mathcal{F}$ that we will refer to as the server Has set. Any file in $\mathcal{F}$ can be duplicated among the Has sets of the different servers. It is assumed that all the files in the library are stored in the servers collectively (i.e., $\bigcup_{i=1}^{N_s} \mathcal{H}_{s_i} = \mathcal{F}$). Each client c_j possibly stores some files from prior downloads in its Has set $\mathcal{H}_{c_j}$, and requests to download another set of files $\mathcal{W}_{c_j}$ termed as the Wants set of c_j.

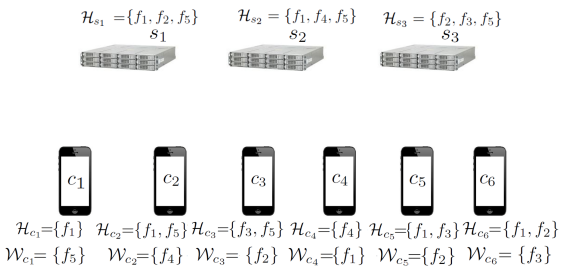
For simplicity of analysis, it is assumed that files are equally sized, and that all servers have the same storage capacity $S = |\mathcal{H}_{s_i}|$. We consider two storage models, namely the fixed and random storage models. In the former model, S non-duplicated files are stored in fixed determinstic (possibly optimized) manner. In the latter, each server stores S non-duplicated files uniformly from $\mathcal{F}$.

The servers are assumed to transmit files with a fixed bit rate, and thus the transmission of any one file takes a fixed amount of time. Once a requested file is received by a client, it send a feedback packet to the servers, which is naturally much smaller in size compared to the file size. The storage servers are assumed to operate on orthogonal channels. Before the transmission of each file by the servers, the client tunes its reception frequency to only one server.

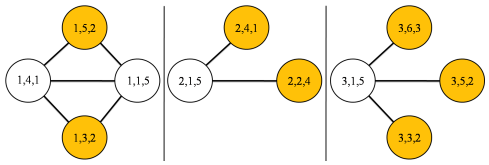
Finally, we assume that time is slotted into download time units (DTUs), each of which fitting the sum of the times required for (1) the client to tune its reception frequency to the one used by the desired server, (2) the transmission of one file by the server, (3) the server reception of the feedback packets sent by the clients. The overall time to download all the files requested by all the clients will be thus measured in DTUs.

2.2 Parameters and Definitions

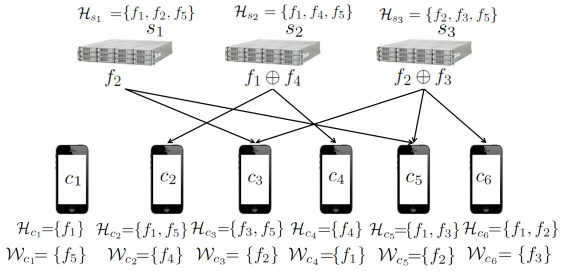
Let p_{ij} be the probability of file corruption observed by client c_j on the downlink channel of server s_i. Also, let q_{ij} be the probability of feedback corruption observed by server s_i on the uplink channel with client c_j. It is intuitive to assume that $q_{ij} < p_{ij}$ due to the larger size of the files on the downlink as compared to uplink, the lower data rates on the uplink, and the lower levels of interference affecting the servers as compared to the clients. Note that the lossless channel case can be represented as a special case in which $p_{ij} = q_{ij} = 0 \ \forall \ s_i \in \mathcal{S}$ and $c_j \in \mathcal{C}$. Let $H_j = |\mathcal{H}_{c_j}|$ be the number of files previously downloaded by client c_j, the average side information at a client H is given by $H = (\frac{1}{N_c}) \sum_{j=1}^{N_c} H_j$. In addition, let $D_j = |\mathcal{W}_{c_j}|$ be the number of the demanded (wanted) files by client c_j, the average demand of any client D is given by $D = (\frac{1}{N_c}) \sum_{j=1}^{N_c} D_j$. Finally, for the random storage model, the repetition index $1 \leq R \leq N_s$ is defined as the average number of file copies stored in all servers $R = \frac{N_s S}{N_f}$. For the fixed storage mode, R is a fixed deterministic quantity.



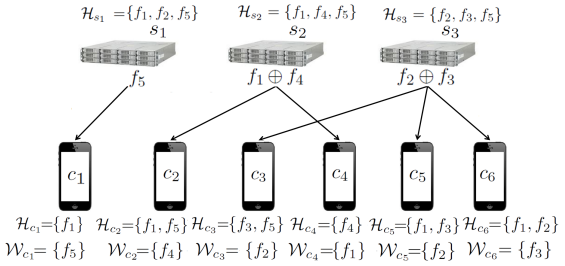
(a) Contents and requests of the servers and clients.



(b) Separated IDNC conflict graphs of s_1, s_2 and s_3, respectively.



(c) Download pattern corresponding to separated IDNC solutions.



(d) Conflict-free download pattern.

Fig. 1: Motivating Example

3 Motivating Example

Let us investigate the simple example shown in Fig. 1, which includes 3 storage servers having $\mathcal{H}_{s_1} = \{f_1, f_2, f_5\}$, $\mathcal{H}_{s_2} = \{f_1, f_4, f_5\}$, $\mathcal{H}_{s_3} = \{f_2, f_3, f_5\}$, and 6 clients having the following contents and requests:
$\mathcal{H}_{c_1} = \{f_1\}$, and $\mathcal{W}_{c_1} = \{f_5\}$.
$\mathcal{H}_{c_2} = \{f_1, f_5\}$, and $\mathcal{W}_{c_2} = \{f_4\}$.
$\mathcal{H}_{c_3} = \{f_3, f_5\}$, and $\mathcal{W}_{c_3} = \{f_2\}$.
$\mathcal{H}_{c_4} = \{f_4\}$, and $\mathcal{W}_{c_4} = \{f_1\}$.
$\mathcal{H}_{c_5} = \{f_1, f_3\}$, and $\mathcal{W}_{c_5} = \{f_2\}$.
$\mathcal{H}_{c_6} = \{f_1, f_2\}$, and $\mathcal{W}_{c_6} = \{f_3\}$.
Assuming no corruption occurs, the uncoded download process requires 2 DTUs and can be performed as follows:

- DTU 1:
  - s_1 sends f_2 to c_3 AND c_5.
  - s_2 sends f_1 to c_4.
  - s_3 sends f_3 to c_6.

- DTU 2:
  - s_1 sends f_5 to c_1.
  - s_2 sends f_4 to c_2.

In the conventional PMP IDNC case (as in [23]), each server s_i can utilize its knowledge of the pervious download by the clients, and their current requests, to generate its own IDNC conflicts graph. The IDNC graph is a graphical model that represents all coding conflicts (i.e. all cases of file requests that, when XORed together, won't be decodable at least one of its targeted clients).

The graph is first build by adding a vertex $v_{j,k}$ for each requested file $k \in \mathcal{W}_{c_j}$ by client $c_j \in \mathcal{C}$. Any two vertices v_{j_1,k_1} and v_{j_2,k_2} are set adjacent by a coding conflict edge if both following conditions are satisfied:

- $f_{k_1} \neq f_{k_2}$
- $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$

Clearly, if the above two conditions apply, either client c_{j_1} or c_{j_2} will not have the necessary side information to decode the combination $f_{k_1} \oplus f_{k_2}$. On the other hand, any two non-adjacent vertices v_{j_3,k_3} and v_{j_4,k_4}, which do not satisfy both above conditions, will have one of two cases: (1) Either $f_{k_3} = f_{k_4}$, which is one uncoded file that can be always received by both clients (2) Client c_{j_3} has f_{k_4} and c_{j_4} has f_{k_3}, which they can re-XOR with the combination of $f_{k_3} \oplus f_{k_4}$ to decode their desired file.

Given this configuration, each independent set (i.e. each set of pairwise non-adjacent vertices) represent a coding opportunity (i.e. the XOR of the files of such vertices will be decodable at all the clients of these vertices). Each server finds its maximum independent set (or maximum weighted independent set given the guidelines in [23]) and transmits the corresponding coded files to the targeted clients. Fig. 1b illustrates the separate IDNC conflict graphs built by the 3 servers and the resultant maximum independent sets (shown in darker colour in the figures). In this case, the servers transmission pattern in the first DTU will be:

- s_1 sends f_2 to c_3 AND c_5.
- s_2 sends $f_1 \oplus f_4$: c_2 AND c_4 can both decode it.
- s_3 sends $f_2 \oplus f_3$: c_3, c_5 AND c_6 can all decode it.

This download pattern is shown in Fig. 1c. The figure depicts several cases where the same client are served by two different servers. For instance, servers s_1 and s_3 both delivers f_2 to client c_3 which can only receive from one of them. This event will be called a *transmission conflict*, and is obviously undesirable as it is a clear waste of resources. At the same time, c_1 is not served by any server in the first DTU, will consequently be served in the second DTU, thus making the overall download time equals to 2 DTUs. As a result, the no coding scenario and the conventional IDNC have the same performance.

This example raises the question of whether a better download pattern can be found to reduce the download time. By careful examination of all options, the files download process can be achieved in only 1 DTU if the servers adopt the following download pattern and coded transmissions:

- s_1 sends f_5 to c_1.
- s_2 sends $f_1 \oplus f_4$: c_2 AND c_4 can decode it.
- s_3 sends $f_2 \oplus f_3$: c_3, c_5 AND c_6 can decode it.

This preferable transmission pattern is shown in Fig. 1d, which clearly shows that all client requests are served in 1 DTU. Thus, a better solution can be obtained when transmission conflicts are avoided, although that may result in sub-optimal coding from each server's perspective. The question is how to find such a solution in a systematic manner for an arbitrary number of clients, files and servers. This motivate the introduction of the novel dual-conflict graph model in the next section, which will be shown to systematically identify the above conflict-free download pattern and coded transmissions.

4 Dual-Conflict IDNC Graph

To avoid transmission conflicts, the *dual conflict IDNC graph* model is proposed to represent, in one augmented graph, both transmission and coding conflicts. The proposed dual conflict IDNC graph $\mathcal{G}$is constructed as follows as follows:

Vertex Set: The vertex set is constructed by adding a vertex $v_{i,j,k}$ for each fike $f_k \in \mathcal{H}_{s_i} \cap \mathcal{W}_{c_j}$ $\forall$ $c_j \in \mathcal{C}$ and $\forall$ $s_i \in \mathcal{S}$. In other words, we add a vertex for each file f_k that is required by each client c_j and stored in each server s_i.

Coding Conflict Edges: Any two vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} in $\mathcal{G}$ are set adjacent by a coding conflict edge if all three conditions below are satisfied:

1) $i_1 = i_2$
2) $f_{k_1} \neq f_{k_2}$
3) $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$.

Clearly, Conditions 2 and 3 are the exact same coding conflict conditions in the conventional IDNC conflict graph, explained in Section 3. The only difference lies in Condition 1, where it is imposed that the two vertices must represent a transmission from the same server. This is trivial conditions as coding is done at each server separately, and no coding conflict can occur between two different combinations sent by two different servers.

Transmission Conflict Edges: Any two of vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} in $\mathcal{G}$ are set adjacent by a transmission conflict edge if both below conditions are satisfied:

1) $j_1 = j_2$
2) $i_1 \neq i_2$

In other words, any pair of vertices symbolizing the simultaneous transmission to the same client (Condition 1) by two different servers (Condition 2) will be set adjacent. Indeed, the simultaneous service of these two vertices represents a clear transmission conflict case as the ones illustrated in Section 3, and must thus be avoided.

Given this new configuration of the conflicts, any maximal independent set in this graph will represent a set of coded transmissions for all the servers without any transmission nor coding conflicts. Fig. 2 shows the dual-conflict graph of the motivating example in Section 3, and the maximum independent set (represented by darker colour in the figures). Obviously, the resulting download pattern from this maximum independent set is identical to that in Fig. 1d.

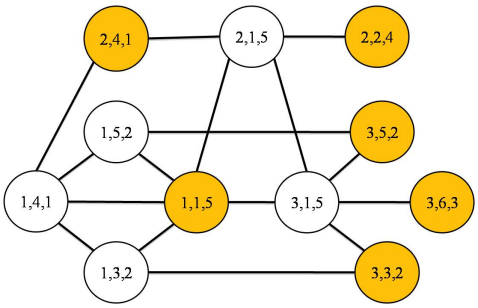


Fig. 2: Dual-conflict IDNC graph of the example in Section 3. Dark colored vertices represents the maximum independent set that result in the download pattern in Fig. 1d.

5 Minimum Download Time Problem

5.1 SSP Formulation for Perfect Feedback Environment

The download time minimization problem has the same problem structure as an SSP problem, where the possible state space, action space, transition probabilities and the state-action costs can be defined as follows.

- *State Space*:
  Due to the fact that the client can tune to a single server at each time epoch, the states space can be considered as possible subgraphs of the dual conflict graph, and their corresponding edge evaluations (i.e., possible removal of coding conflicts due to progressive client file receptions) until all its vertices vanish (i.e., until the download of all requested files is completed).
- *Action Space*:
  The action space for each state can be defined as the set of all independent sets of the state's corresponding dual-conflict IDNC graph.
- *State-Action Transition Probabilities*:
  The system will stay at the same state or transit to another state based on the action (i.e., transmission pattern for a given DTU) taken at each state. For each action in a given state, the transition probability from this state to each potential next state will correspond to the product of the file reception and corruption probabilities of the targeted clients by this action (i.e., transmissions) from their targeting servers, which results in ending up at this specific state.
- *State-Action Cost*:
  One DTU is the cost of each action taken by the servers towards download completion.

It is clear that this SSP formulation suffers from the well-known curse of dimensionality, due to the exponential size of both its state space $O\left(2^{N_c N_f}\right)$ and action space $O\left(3^{\frac{N_s N_c N_f}{3}}\right)$. Accordingly, solving this problem optimally is intractable even for relatively small number of servers (N_s), clients (N_c) and files (N_f). Consequently, an efficient online heuristic algorithm is required to solve it, specially in real time.

5.2 Maximum weighted Vertex Search Algorithm

The online approach aims to select one download pattern for only one DTU, collect the feedback, and then select the download pattern for the next DTU on so on. To select a download pattern for each DTU, a maximum (or maximum weight) independent set must be selected. Though finding the maximum weight independent set in a graph is an NP-hard problem, many well-known solvers can be used to find it with high level of accuracy, such as the Bron-Kerbosh algorithm [32]. However, we will design an even simpler online heuristic algorithm that can find a maximal independent set in real time through a maximum weight vertex search approach. From IDNC's properties shown in [23], the overall download time is lower bounded by the want set size of the client that needs the greatest number of files. Consequently, we need to address the maximum number of clients with large want sets in each transmission. Furthermore, in imperfect environment, the client needs to tune to the server with the best channel from among the ones that store its requested files. To achieve these two targets, we first assign a raw weight w_{ijk} to each vertex $v_{i,j,k}$ in $\mathcal{G}$. This weight reflects the file reception probability at client c_j from server s_i, and also the size of the wants set of c_j. This raw weight is defined as follows:

$$w_{ijk} = (1 - p_{ij})|\mathcal{W}_j|. \tag{1}$$

These raw weights in (1) can be used for one or multiple file request (in the former case, $|\mathcal{W}_j| = 1$), and for corruption-free and corrupted environments (in the former environment, $p_{ij} = 0$). To refine the greedy vertex selection process, we define the non-adjacency indicator of vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} ($a_{i_1 j_1 k_1, i_2 j_2 k_2}$) as follows

$$a_{i_1 j_1 k_1, i_2 j_2 k_2} = \begin{cases} 1, & v_{i_1,j_1,k_1} \text{ is not adjacent to } v_{i_2,j_2,k_2} \\ 0, & v_{i_1,j_1,k_1} \text{ is adjacent to } v_{i_2,j_2,k_2}. \end{cases} \tag{2}$$

Finally, the vertex modified weight w'_{ijk} is defined as

$$w'_{ijk}(\mathcal{G}) = w_{ijk} \sum_{\forall v_{xyz} \in \mathcal{G}} w_{xyz} \cdot a_{ijk,xyz}. \tag{3}$$

Hence, a vertex with the highest modified weight has two attractive aspects. The first aspect is that it has a large raw weight. The second aspect is that it is non-adjacent to a large number of vertices induced by clients with high raw weight w'_{ijk}. Using this definition of the modified weights, Algorithm 1 executes iteratively a greedy weighted vertex search procedure to build a maximal independent set, with a possibly high weight. Each iteration will be implemented as

Algorithm 1 Maximum Weighted Vertex Search Algorithm

1: Initialization:
- Set Graph $\mathcal{G}_s \leftarrow \mathcal{G}$.
- $\forall$ vertices in $\mathcal{G}$, compute the raw weights as in (1).
- Set the Selected Independent set $\Gamma = \phi$.

2: **repeat**
3: $\quad \forall$ vertex $v_{ijk} \in \mathcal{G}_s$: Calculate $w'_{ijk}(\mathcal{G}_s)$ as in (3)
4: $\quad v^*_{ijk} \leftarrow \max_{v_{ijk} \in \mathcal{G}_s} \{w'_{ijk}(\mathcal{G}_s)\}$
5: $\quad \Gamma = \Gamma \cup v^*_{ijk}$
6: $\quad \mathcal{G}_s \leftarrow \mathcal{G}_s \setminus \left(v^*_{ijk} \cup \mathcal{N}_{\mathcal{G}_s}(v^*_{ijk})\right)$
7: **until** $\mathcal{G}_s = \phi$, *end repeat.*
8: *Return the Selected independent set* Γ

follows: First, the algorithm computes the modified weights of all the vertices in the graph $\mathcal{G}_s$, initially set to $\mathcal{G}$ before the first iteration. Second, the maximum weighted vertex v^*_{ijk} will be picked out and added to Γ. Finally, the graph $\mathcal{G}_s$ is updated by eliminating the chosen vertex v^*_{ijk} and all the vertices that are adjacent it (symbolized in the algorithm by $\mathcal{N}_{\mathcal{G}}(v_{ijk})$) from the previous iteration graph. This elimination is done to guarantee that the next picked vertex is not in coding nor transmission conflict with the already selected ones in Γ. The algorithm continues until no more vertices exist in $\mathcal{G}_s$.

5.3 Implementation Issues

We can implement the proposed conflict-free IDNC algorithm in either centralized or decentralized (distributed) approach. In the former approach, a cloud/fog storage controller, which knows the side information of all clients (from the prior download log files) and the stored content at the servers, can build the dual-conflict graph and use Algorithm 1 to schedule the conflict-free IDNC download pattern for each DTU. In the latter scenario, each server can build the dual-conflict graph separately, by exchanging their knowledge of content and the clients sub-channels parameters with each other using their backbone connections. Each server runs Algorithm 1 separately and finds the conflict-free IDNC combination that it is required to transmit in each DTU. Since the content update rate of cloud/fog storage systems is orders of magnitude lower compared to DTUs, the decentralized approach can be maintained and work appropriately at each server without frequently exchanging information regarding stored files. The only parameters that will require frequent exchange are the corruption probabilities, which can be easily managed through the very high speed (usually fibre-optical) backbone.

6 Asymptotic Bounds for the Fixed Storage Model

In this section, we aim to derive asymptotic bounds on the download time performance of our proposed online solution for the following scenario:

- Lossless channels
- Fixed storage model
- One file request per client

Asymptotically speaking, the sever and clients will go through all possibilities of file storage and requests. Thus, the file identities can be ignored in this analysis. Only the number of files stored at the the servers and clients will be used to transform the dual-conflict graph into an asymptotically-equivalent random graph $\mathcal{G}_{\nu,\pi}$ with the same number of vertices ν and vertex adjacency probability π [33]. Once these random graph parameters are computed, we can use the result derived by Bollobas in [33], stating that every random graph $\mathcal{G}_{\nu,\pi}$ has a chromatic number $\chi(\mathcal{G}_{\nu,\pi})$ that can be approximated as:

$$\chi(\mathcal{G}_{\nu,\pi}) = \left(\frac{1}{2} + o(1)\right) \log\left(\frac{1}{1-\pi}\right) \frac{\nu}{\log \nu}. \quad (4)$$

We will thus start by deriving the parameter ν and π of the asymptotically equivalent random graph corresponding to the proposed dual-conflict IDNC graph, then use it to introduce the upper and lower bounds on the performance of our proposed online solution.

Theorem 1. For the considered scenario, the dual-conflict IDNC graph $\mathcal{G}$ can be asymptotically modelled by a random graph $\mathcal{G}_{\nu,\pi}$, such that:

$$\nu = N_c R \quad (5)$$

and

$$\pi = \left(\frac{N_c - 1}{N_c N_s - 1}\right)\left(1 - \frac{N_c - 1}{N_c S - 1}\right)(1-\psi) + \frac{R(R-1)}{\nu(\nu-1)}. \quad (6)$$

where ψ is expressed as shown in (7).

Proof: The proof is in Appendix A. □

Theorem 2. For the considered scenario, the performance of the dual-conflict IDNC algorithm is asymptotically upper bounded by $\chi(\mathcal{G}_{\nu,\pi})$, where ν and π are as defined in Theorem 1.

Proof: For graph $\mathcal{G}$, the chromatic number is the smallest required number of colors to color the vertices of $\mathcal{G}$ such that any two adjacent vertices do not share the same color [34]. Thus, the chromatic number represents the number of independent sets in that graph. In the case of one requested file per client, the transmission of an IDNC file will not generate any new coding opportunities since each client wants only one file and it will leave the competition once it receives the file. However, the dual conflict IDNC graph consists of $N_c R$ vertices. N_c of these vertices represent the actual number of the wanted files. The remaining $N_c(R-1)$ vertices represent repeated copies of the wanted files due to the file repetition among the servers. Since, after a request is being served, all the vertices representing this request will be removed from the graph after applying the conflict free IDNC algorithm. On the other hand, the chromatic number of the graph is calculated with presence of these repeated vertices and also the conflict edges connect them. Consequently, the actual number of transmissions should be smaller than the dual-conflict graph chromatic number. Hence, the performance of the conflict free IDNC algorithm is upper bounded by the chromatic number of the dual-conflict graph. □

Corollary 1. For the considered scenario, the performance of the dual-conflict IDNC algorithm is asymptotically lower bounded by $\frac{\chi(\mathcal{G}_{\nu,\pi})}{R}$, where ν and π are as defined in Theorem 1.

Proof: Let N^{IDNC} be the actual minimum number of the dual-conflict graph maximal independent sets after removing the copies. We need to prove the following

$$N^{IDNC} R \geq \chi(\mathcal{G}_{\nu,\pi}). \quad (8)$$

To do so, we need to use the original graph $\mathcal{G}_{\nu,\pi}$ to generate a new graph with a chromatic number $N^{IDNC}R$. This can be easily done by taking the graph containing the actual solution (the one with the minimum number of maximal independent sets) after removing all the copies, and repeating this graph R times. The last step is to connect all the R copies of that graph (i.e., every vertex in a copy of the graph should be connected to every vertex in another copy) to

$$\psi = \sum_{y=0}^{H} \frac{\binom{H}{y}\binom{N_f-H}{H-y}}{\binom{N_f}{H}} \sum_{e_1=max\left(0,H-\left(N_f-S\right)\right)}^{min(H-\mathbf{y},S)} \binom{S}{e_1}\binom{N_f-S}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_1}{S} \times \sum_{e_2=max\left(0,H-\left(N_f-(S-e_1)\right)\right)}^{min(H-\mathbf{y},(S-e_1))} \binom{S-e_1}{e_2}\binom{N_f-(S-e_1)}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_2}{S-1}. \quad (7)$$

make sure that the chromatic number of the whole resultant graph is equal to the sum of the chromatic numbers of all copies.

The resultant graph will have a greater number of edges than the original one, because we do not connect a copy of a request to all the other copies of another request in $\mathcal{G}_{\nu,\pi}$. Since increasing the number of edges in a graph means it has a greater or equal chromatic number, the new graph chromatic number is greater than or equal to that of $\mathcal{G}_{\nu,\pi}$, and hence inequality in (13) is hold [33]. The theorem follows from dividing both sides in (13) by R.

□

7 Imperfect Feedback Environment

In perfect feedback environment, the servers get perfectly updated on the reception status of their transmitted files from each of the receivers. This allows the proposed online algorithm to determine the next download pattern given perfect knowledge of the side information at the clients. On the other hand, the reception feedback sent by the clients are subject to loss/corruption. Consequently, the online algorithm has to perform the subsequent download pattern decision based on uncertainties resulting from such missing information. In this section, we will investigate the imperfect feedback consequences in this multipoint-to-multipoint system and its effect on the overall download time problem.

7.1 Feedback Model

At the first time epoch, each server has full knowledge about the clients side information and the Wants set of each client. The feedback from clients assist the servers to update the graph based on the updated clients' Want and Has sets after each DTU. Each targeted client c_j broadcasts a feedback packet to all servers after each download of one of its requested files, using a common control channel. If c_j is not targeted in any download pattern, it does not issue any feedback packets at all. Thus, the event of an unheard feedback in the perfect feedback environment makes the server certain that the transmitted file was lost and the targeted client has not received it.

However, these packets are in general subject to loss on the uplink channels. The file with lost feedback from a certain client will be called an uncertain file for that client. In this situation, an unheard feedback from a targeted client can be due to either the corruption of its received file on the downlink or the loss of its issued feedback when it successfully received it. In this case, the server will not be able to tell which case occurred and must thus select the subsequent download patterns given these uncertainties.

In the centralized implementation approach, the cloud/fog controller of all servers (i.e., the component that execute the proposed algorithm) will be uncertain about file reception status at its targeted client if all the servers in the system lose the feedback. Indeed, the cloud/fog controller needs only at least one server to receive each client feedback, in order to have the exact status of the network. Moreover, the uncertainties are identified by a unified entity (i.e. the cloud/fog controller), thus guaranteeing the conflict-free property of subsequent download patten from all servers despite uncertainties.

In the distributed approach, each server s_i will be uncertain about a file reception status at a targeted client c_j if server s_i loses the feedback sent by c_j. Due to the different channel qualities from each targeted client to each of the servers, the feedback issued by this client may be heard by some servers and unheard by others. Clearly, if such discrepancies in the set of uncertain files perceived by each server are unresolved, this may result in different ML decisions and thus different subsequent dual-conflict graphs at the different servers. Since the distributed implementation requires each server to perform its own decision based on its own built dual-conflict graph, such discrepancies may result in both transmission conflicts and wrong assumptions at each server about the transmissions of other server. This may clearly lead to a great increase in the download time.

To avoid this perturbed situation, we will assume that the distributed approach will employ the backbone high speed connectivity to exchange the collected feedback by all servers. This ends up with the server having the same knowledge about uncertain files as in the centralized approach and thus can achieve the exact same performance.

7.2 Imperfect Feedback Implications on the Problem

Since unheard feedback events hide the exact state of the system from the server, they thus convert the SSP problem formulation to a partially observable SSP (POSSP) problem [35]. Clearly, this POSSP problem will be more complicated and thus intractable to solve optimally. Nonetheless, the solution that was proposed in section 5 can still be used to sub-optimally solve this problem using an estimation of the actual state of uncertain file. In such partially observable scenario, the best estimation of an uncertain file state is the one representing the maximum likelihood (ML), and will thus be investigated in the next section.

The maximum likelihood state estimation step can be implemented after each DTU to assist the cloud/fog controller (in the centralized approach) or each server (in the distributed approach) to estimate the most likely reception state of all uncertain files. The resulting decision on each file can be thus used to efficiently create a most likely version

of the actual dual-conflict graph before the start of the new DTU. Errors in estimating the state of any file can be easily resolved as transmissions evolve. In the next two section, the ML rule in the considered network and feedback model will be derived, and the corresponding graph update procedures and recovery from estimation errors will be then detailed.

7.3 ML Estimation Rule

for each instance of unheard feedback at the cloud/fog storage system when a file f_k is sent to client c_j from server s_i, one of two cases may have occurred:

- c_j did not receive the file and thus did not issue a feedback packet. This case occurs with a probability p_{ij}.
- c_j received the file, and sent a feedback with was lost on the uplink channel by all servers. This event occurs with probability $(1-p_{ij})\prod_{i=1}^{N_s} q_{ij}$.

Clearly, these two cases are mutually exclusive.

Now at any DTU, the storage servers may have attempted the transmission of file f_k to client c_j n times without hearing feedback for any of them. Note that each of these attempts of f_k to c_j may have been performed by different servers. Consequently, define n_{ijk} as the number of attempts (out of the n attempts) in which client c_j is targeted by server s_i with file f_k without hearing a feedback from any of them. Clearly, $n=\sum_{i=1}^{N_s} n_{ijk}$. We can use these facts and definitions to derive the ML rule as follows.

Theorem 3. For any client c_j, if any file f_k is attempted n_{ijk} times by server $s_i\ \forall\ s_i \in \mathcal{S}$, this file is most likely to be not received at that client (and thus needs to be further re-attempted) if:

$$\frac{\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}}}{\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}} + P\left(\bigcup_{i=1}^{N_s} E_i\right)} \geq \frac{1}{2} \tag{9}$$

where E_i is the event of file f_k being received by client c_j from any of the n_{ijk} attempts of server s_i while still deemed uncertain by all servers, whose probability of occurrence is equal to:

$$P(E_i) = \sum_{m=1}^{n_{ijk}} \binom{n_{ijk}}{m} (1-p_{ij})^m p_{ij}^{(n_{ijk}-m)} \prod_{i=1}^{N_s} q_{ij}^m\ . \tag{10}$$

Note that events E_i are mutually non-exclusive yet independent events, and thus the knowledge of $P(E_i)$ $\forall\ s_i \in \mathcal{S}$ is all what is required to compute $P\left(\bigcup_{i=1}^{N_s} E_i\right)$.

Proof: To prove this theorem, we define $\mathcal{P}_{jk}^R$ as the probability that an uncertain file f_k of client c_j has been received, and $\mathcal{P}_{jk}^L$ as the probability that an uncertain file f_k of client c_j has not been received. To derive the ML rule, we first need to derive expressions for $\mathcal{P}_{jk}^L$ and $\mathcal{P}_{jk}^R$. Given the definitions of n_{ijk}, the probability that file f_k is actually not received at client c_j and deemed uncertain at the servers is equal to the probability that this file was corrupted on the downlink channels for each of the n attempts. This occurs with probability $\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}}$.

On the other hand, a file f_k is actually received client c_j and deemed uncertain at the servers if this client has successfully decoded it in any of the n_{ijk} attempts of each server s_i but the feedback(s) of all such instances were lost by all servers. Denoting the occurrence of such event for server s_i by E_i, the probability of such event is equal to:

$$P(E_i) = \sum_{m=1}^{n_{ijk}} \binom{n_{ijk}}{m} (1-p_{ij})^m p_{ij}^{(n_{ijk}-m)} \prod_{i=1}^{N_s} q_{ij}^m. \tag{11}$$

The overall probability of occurrence of this event from all servers combined is thus $P\left(\bigcup_{i=1}^{N_s} E_i\right)$. Note that events E_i are both independent and mutually non-exclusive. Thus, the knowledge of $P(E_i)\ \forall\ s_i \in \mathcal{S}$ is all what is required to compute $P\left(\bigcup_{i=1}^{N_s} E_i\right)$ using the conventional union probability expression for N_s independent, mutually non-exclusive events.

Therefore, given n attempts to transmit file f_k to client c_j with no feedback on any of them, $\mathcal{P}_{jk}^L$ and $\mathcal{P}_{jk}^R$ can be expressed as shown in (12).

$$\begin{aligned}
\mathcal{P}_{jk}^L &= \frac{\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}}}{\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}} + P\left(\bigcup_{i=1}^{N_s} E_i\right)}, \\
\mathcal{P}_{jk}^R &= \frac{P\left(\bigcup_{i=1}^{N_s} E_i\right)}{\prod_{i=1}^{N_s} p_{ij}^{n_{ijk}} + P\left(\bigcup_{i=1}^{N_s} E_i\right)} = 1 - \mathcal{P}_{jk}^L.
\end{aligned} \tag{12}$$

Consequently, file f_k is most likely not received at c_j after all n_{ijk} attempts from each server $s_i, \forall\ s_i \in \mathcal{S}$, and thus needs to be re-attempted, if

$$\mathcal{P}_{jk}^L \geq \mathcal{P}_{jk}^R \Rightarrow \mathcal{P}_{jk}^L \geq 1 - \mathcal{P}_{jk}^L \Rightarrow \mathcal{P}_{jk}^L \geq \frac{1}{2} \tag{13}$$

The theorem follows from substituting the first term of (12) in (13). □

7.4 Partially Blind Graph Update Algorithm Based on ML

Based on the ML decision rule, we adopt the partially blind graph update algorithm proposed in [36] to solve the download time minimization problem with imperfect feedback environment. When the cloud/fog controller or independent servers have uncertain files, the ML state estimation rule, described in the previous section, can be employed to update the conflict-free IDNC graph as follows. When the file f_k is most likely received at client c_j (according to the ML rule), the set of vertices $v_{i,j,k}\ \forall\ s_i$ storing f_k will be considered as a hidden vertices within the dual-conflict IDNC graph. This means that the client c_j will not be targeted by the file f_k temporarily. Otherwise, these vertices will be kept in the graph as ordinary vertices induced by a wanted file. Consequently, it could be considered for the possible subsequent transmission.

The hidden vertices of any given client will be treated according to what happens later as follows:

- The server will know the actual state of the transmitted file once it receives a feedback from the targeted

client. Accordingly, it will update the status of the hidden vertex.

- If the client has only hidden vertices in the dual-conflict IDNC graph, all these vertices induced by the client will be brought back as ordinary vertices and will be considered in the subsequent conflict-free IDNC files transmission until a corruption-free feedback is received from this client.

8 Simulation Results

In this section, we compare, through extensive simulations, the performance of our proposed dual conflict IDNC algorithm to that of the conventional IDNC solution designed to minimize the completion time in PMP systems in which each server utilizes separately the algorithm in [23] to minimize the download time without considering the requests served by the other servers. We consider two cases for the feedback channels (the channels between the servers and the clients) namely, perfect feedback and imperfect feedback channels.

8.1 Comparison with Optimal Online Performance and Bounds

This section provides a comparison between the performances of the heuristic algorithm described in Section 5.2 with both the optimal online algorithm based on Bron-Kerbosh approach [32], and the upper and the lower bounds derived in Section 6. The two file storage models (i.e., fixed and random) are considered in the simulation of the optimal and heuristic online scenarios. Fig. 3 depicts the average download completion time performances of the algorithms and the bounds versus the number of clients (with fixed number of files $N_f = 100$) and the numbers of files (for a fixed number of clients $N_c = 20$). The number of servers is $N_s = 4$, the size of each server is $\frac{N_f}{2}$, and the size of the client side information is $\lceil \frac{N_f}{3} \rceil$ files. For fair comparison with the bounds, we assume corruption-free channels and one file request per client (i.e., $|\mathcal{W}_{c_j}| = 1 \ \forall \ c_j \in \mathcal{C}$). We note that the performance of our heuristic algorithm is very close to the optimal online solution. The effect of the file placement at the servers is notable as the performance of the random file storage model achieves about 0.2 less DTUs compared to the fixed placement model. Despite the derivation of the bounds for the fixed storage model only, the figure clearly shows that they indeed bound the performance of both storage models.

8.2 Perfect Feedback (PF) Scenarios

In this section, we introduce the simulation results of the dual-conflict IDNC algorithm and compare it to the conventional separate IDNC solution, both applied to a cloud/fog storage system with $N_s = 10$ servers. In each simulated scenario, each client has previously downloaded and wants to download a random number of files, both with an average of $\lceil \frac{N_f}{3} \rceil$ files. We assume that the probability of file corruption at the downlink channel between any server-client pair is uniformly distributed in the interval $[0.01, 0.3]$. The corruption probability of each server-client pair does not change during the entire process of requested

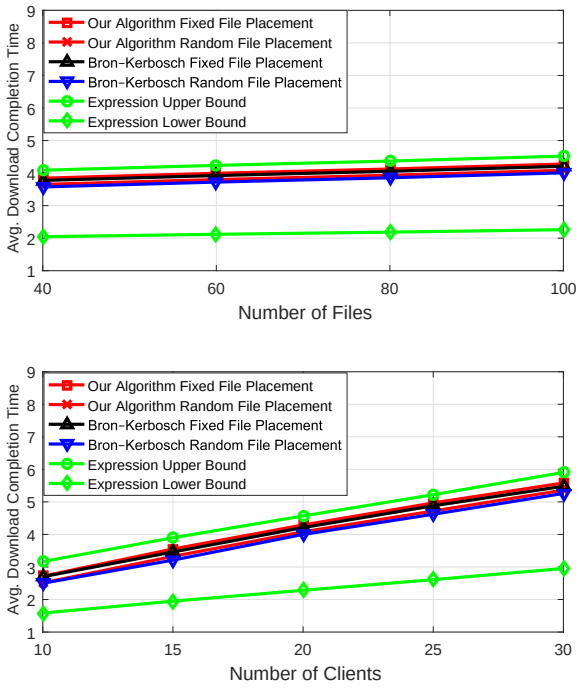


Fig. 3: Performance comparison versus the number of the clients N_c and the number of files.

file download. The uplink channel between any arbitrary client and server is assumed to be corruption-free.

Fig. 4 shows the average download completion time versus the number of clients N_c, with $N_f = 100$ files and each server stores 50 files, which are taken randomly from the files library that consists of 100 files. The figure depicts a reduction of 7 to 10 DTUs in the average download completion time when the conflict-free algorithm is compared with the conventional PMP scheme. We can also observe that the gap in performance between the conventional and conflic-free algorithms slightly decreases as the total number of clients increases. This can be explained by the fact that an increasing number of clients results in fewer transmission conflicts the conventional IDNC approach is utilized, given fixed server sizes. This slightly enhances the performance of the conventional IDCN approach.

Fig. 5 illustrates a similar comparison versus the total number of files N_f, with $N_c = 60$ clients and each server stores $\frac{N_f}{2}$ files. We can notice the increasing gain in performance when utilizing the conflict-free IDNC approach as the tested values of N_f increases.

Fig. 6 illustrates the effect of the server storage capacity on the performance of both conflict-free IDNC and conventional approaches with fixed library size of $N_f = 100$ files and $N_c = 60$ clients. It can be observed from the figure that the performance of the conventional IDNC scheme degrades as the server storage capacity increases, while the conflict-free IDNC scheme performance is enhanced. This can be interpreted as follows. For the conventional IDNC scheme, more files per server means more clients served per server per DTU, thus increasing the chances of transmission conflicts (as no pre-cautions are taken to prevent them) and

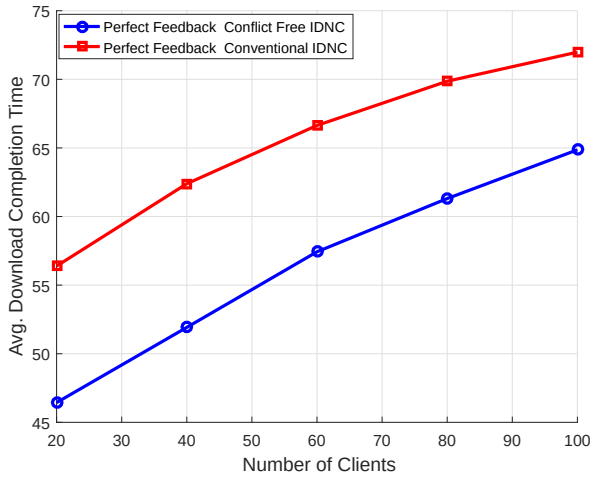


Fig. 4: The average download completion time versus the number of clients N_c.

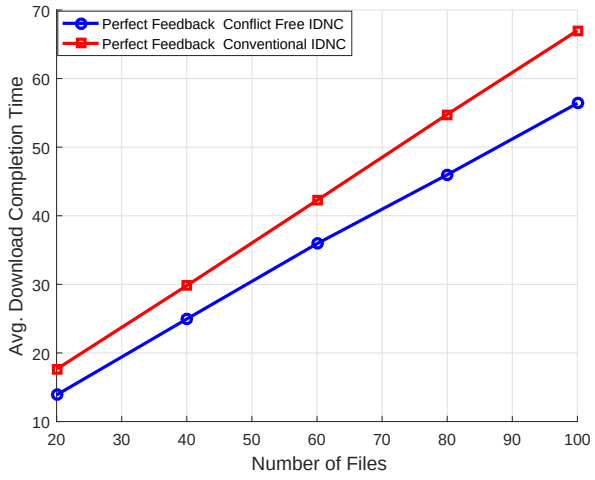


Fig. 5: The average download completion time versus the number of files N_f.

degrading the performance. On the other hand, the serves in the conflict-free IDNC scheme still serve more clients as their size increases but with strictly no transmission conflicts, which results in a lower download completion time.

8.3 Imperfect Feedback Scenarios

In this section, we consider the system described in the previous section with imperfect feedback channels. Feedback loss would result in different download patterns for the centralized and the distributed scenarios described in Section 5.3.

In each simulation, we consider the conventional PMP IDNC and the dual-conflict IDNC algorithm implemented in the centralized and the distributed scenarios with imperfect feedback environment. We assume that $q_{ij} = \frac{1}{2}p_{ij}$ and p_{ij} is uniformly distributed over the interval $[0.01, 0.3]$ uniformly. The ML rule and graph update policy in (**??**) are applied in the conventional PMP IDNC. In addition to the partially blind algorithm based on ML described in section

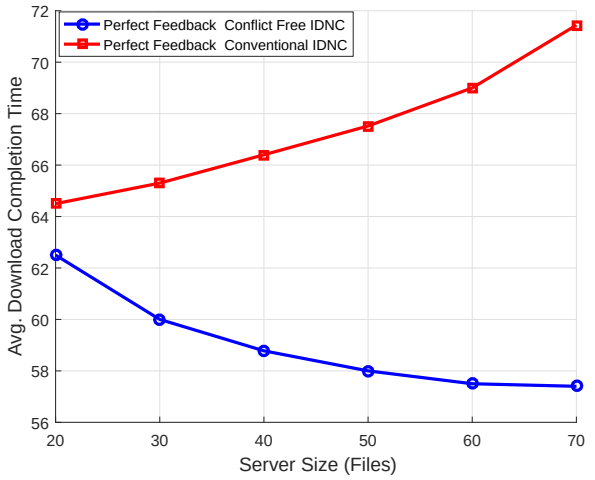


Fig. 6: The average completion time versus the server size.

7.3, two other partially blind graph update approaches are simulated for comparison:

- Full Vertex Elimination (FVE): The vertices induced by all uncertain files at their corresponding clients will be all made temporary from hidden in the graph, without any decision criteria. They will be brought back to the graph when the same conditions described in Section 7.4 occur.
- Client Block (CB): Each attempted file delivery to a given client with unheard feedback causes all the vertices representing the requested files by that client client will be hidden in the graph and are temporarily not considered for transmission.
  Again, the hidden vertices in this CB approach are treated as described in in Section 7.4.

Figs. 7-9 depict the performance of the different graph update approaches implemented for the conventional and conflict-free IDNC schemes verses N_c (with $N_f = 100$, server size of 50 files), N_f ($N_c = 60$, server size of $\frac{1}{2}N_f$) and the server size S ($N_s = 10$, $N_f = 100$), respectively. The same performance versus the ratio $\frac{q_{ij}}{p_{ij}}$ (with $N_c = 60$, $N_f = 100$ and server size of 50)is illustrated in Fig. 10. For this latter figure, p_{ij} is constant and q_{ij} is varying from $0.2p_{ij}$ up to the worst case $q_{ij} = p_{ij}$.

From these figures, we can perceive the superiority in performance of the ML approach over the CB and the FVE approaches in minimizing the average download completion time. The performance of the ML is slightly degraded to that of the perfect feedback. Indeed, the chance of lost feedback from all servers is quite low. This makes the ML rule decision more accurate. We can also notice that FVE outperforms CB. This can be deduced from the characteristics of the two approaches. FVE hides the vertices that represent unheard feedback only. However, the server s_i is still able to target the client c_j with its other requested files. On the other hand, CB hides the vertex v_{ijk} and all the vertices induced by client c_j. Consequently, the server s_i will temporarily block client c_j completely. Finally, we can also observe that the difference in performance between the dual conflict IDNC and the conventional IDNC are still noticeable in imperfect feedback environment.

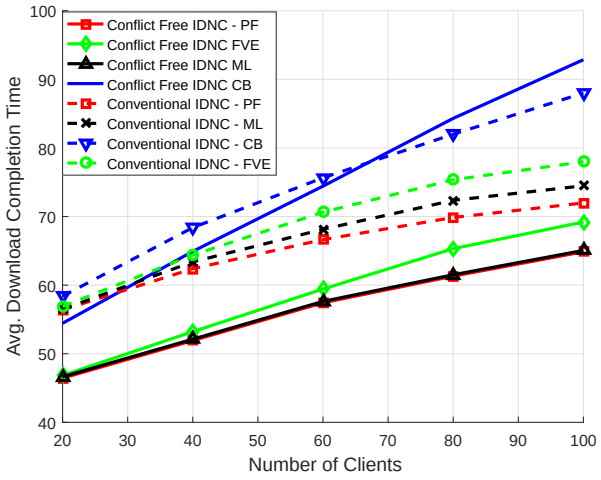


Fig. 7: The average completion time versus the number of the clients N_c.

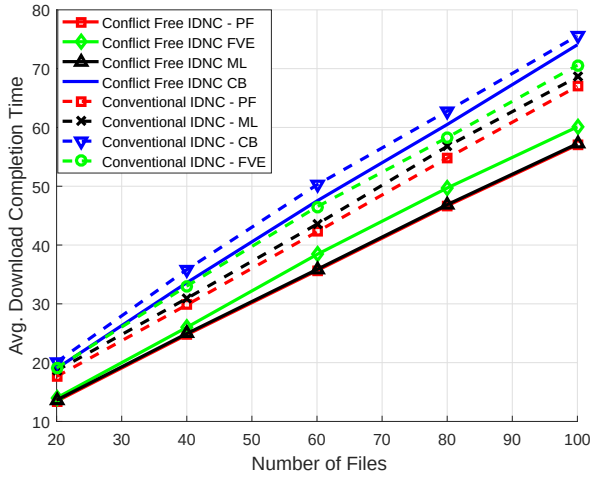


Fig. 8: The average completion time versus the number of the files N_f.

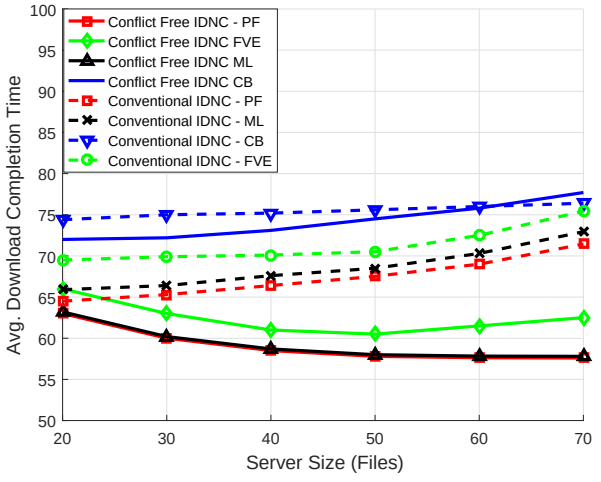


Fig. 9: Perfect and imperfect feedback performance comparison over a range of the server size S.

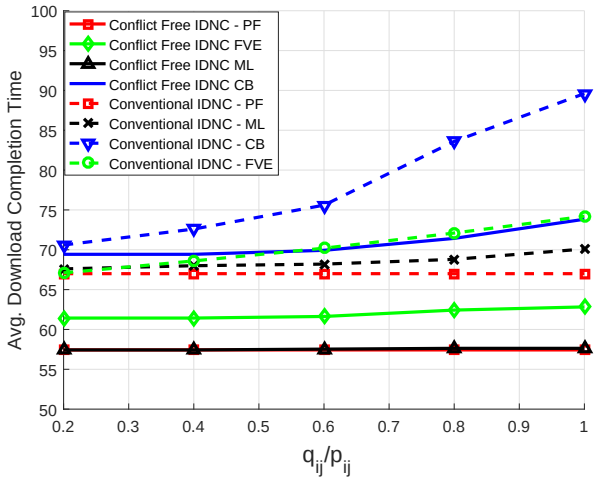


Fig. 10: Perfect and imperfect feedback performance comparison over a range of the ratio $\frac{q_{ij}}{p_{ij}}$.

9 Conclusions

In this paper, the multi-client download time reduction problem from cloud/fog storage servers was investigated in perfect and imperfect feedback environments. Applying the conventional PMP IDNC algorithm at each server separately was shown to result in transmissions conflicts, which reduce the download efficiency. Consequently, we proposed a novel dual-conflict graph model that avoids such conflicts and guaranteed conflict-free transmissions. The download time reduction problem was first formulated as an SSP problem and shown to be intractable. We thus designed an online heuristic algorithm that applies maximum weight vertex search over the dual-conflict graph to find the most suitable file download pattern at DTU. For the special case of lossless-channels, fixed storage model, and one file request per client, an upper and lower bounds of the conflict-free IDNC algorithm performance were derived. The simulation results show that this bounds are valid for both fixed and random storage models. The proposed heuristic algorithm was also shown to achieve near-optimal online performance, and a significant improvement compared to the conventional PMP IDNC scheme.

We extended the formulation of the download time minimization problem to a POSSP problem in the imperfect feedback environment. Being also intractable, we derived an ML state estimation rule for all uncertain files at their corresponding clients, and employed it to update the dual-conflict graph after each DTU to compute the subsequent download patterns. Simulation results show that the proposed ML based conflict-free IDNC scheme achieves a very slight degradation compared to the perfect feedback scenarios.

Appendix A
Proof of Theorem 1

Based on the log matrix $L(s_i)$, we create a vertex $v_{i,j,k}\ \forall\, c_j \in \mathcal{C}$ and $f_k \in \mathcal{H}_{s_i} \cap \mathcal{W}_{c_j}$. Thus, the total number of vertices

ν in the dual conflict graph in the case of one requested file per client is written as

$$\nu = N_c R. \tag{14}$$

To derive π in (4), we need to define the conditions needed for two vertices to be connected in the dual conflict graph as events:

- E1: Any two vertices v_{i,j_1,k_1} and v_{i,j_2,k_2}, both of which are induced by the same server, will be set adjacent by an undirected edge represents a coding conflict if
  - C1: $f_{k_1} \neq f_{k_2}$.
  - C2: $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$.
- E2: Any two vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} will be set adjacent by an undirected edge represents transmission conflict if $c_{j_1} = c_{j_2}$ AND $s_{i_1} \neq s_{i_2}$.

Any two vertices will be set adjacent if E1 is satisfied or if E2 is satisfied. It is clear that E1 and E2 are mutually exclusive events (E1 implies $s_{i_1} = s_{i_2}$ while E2 implies $s_{i_1} \neq s_{i_2}$), so the connectivity probability in the dual conflict graph is expressed as

$$\pi = \mathbb{P}(\mathrm{E1}) + \mathbb{P}(\mathrm{E2}) \tag{15}$$

The probability that E1 occurs can be written as

$$\mathbb{P}(\mathrm{E1}) = \mathbb{P}(1s \cap \mathrm{C1} \cap \mathrm{C2}) = \mathbb{P}(1s)\,\mathbb{P}(\mathrm{C1}|1s)\,\mathbb{P}(\mathrm{C2}|1s, \mathrm{C1}). \tag{16}$$

where $1s$ is the event that the two vertices are induced by the same server. To find $\mathbb{P}(1s)$, let X_i be the number of vertices induced by server s_i. Since the files are distributed among the servers with fixed placement R will be integer and the total files will be stored in a block of servers, the number of blocks in the system is $B = \frac{N_s}{R}$. Consequently, X_i can be modeled as a binomial random variable $Bin\left(N_c, \frac{1}{B}\right)$. The total number of the vertices in $\mathcal{G}$ can be expressed as $\nu = \sum_{i=1}^{N_s} X_i$, and without loss of generality, we can consider that $X_{N_s} = \nu - \sum_{i=1}^{N_s-1} X_i$. Based on these facts, the probability $\mathbb{P}\left(\mathbf{x} = \mathbf{x}' | \sum_{i=1}^{N_s} X_i = \nu\right)$ can be found as follows

$$\begin{aligned}
\mathbb{P}\left(\mathbf{x} = \mathbf{x}' | \sum_{i=1}^{N_s} X_i = \nu\right) &= \frac{\mathbb{P}\left(\mathbf{x} = \mathbf{x}', \sum_{i=1}^{N_s} X_i = \nu\right)}{\mathbb{P}\left(\sum_{i=1}^{N_s} X_i = \nu\right)} \\
&= \frac{\mathbb{P}\left(X_1 = x_1, \ldots X_{N_s-1} = x_{N_s-1}, X_{N_s} = \nu - \sum_{i=1}^{N_s-1} X_i\right)}{\mathbb{P}\left(\sum_{i=1}^{N_s} X_i = \nu\right)} \\
&= \prod_{u=1}^{N_s-1} \binom{N_c}{x_u} \left(\frac{1}{B}\right)^{x_u} \left(1 - \frac{1}{B}\right)^{N_c - x_u} \binom{N_c}{\nu - \sum_{i=1}^{N_s-1} x_i} \\
&\times \left(\frac{1}{B}\right)^{\nu - \sum_{i=1}^{N_s-1} x_i} \left(1 - \frac{1}{B}\right)^{N_c - \nu + \sum_{i=1}^{N_s-1} x_i} \\
&\times \left(\binom{N_c N_s}{\nu} \left(\frac{1}{B}\right)^{\nu} \left(1 - \frac{1}{B}\right)^{N_c N_s - \nu}\right)^{-1} \\
&= \prod_{u=1}^{N_s-1} \binom{N_c}{x_u} \binom{N_c}{\nu - \sum_{i=1}^{N_s-1} x_i} \binom{N_c N_s}{\nu}^{-1}.
\end{aligned} \tag{17}$$

So given ν, X is following multivariate hypergeometric distribution. Now, the probability $\mathbb{P}(1s|\nu, \mathbf{x} = \mathbf{x}')$ can be written as

$$\mathbb{P}(1s|\nu, \mathbf{x} = \mathbf{x}') = \sum_{m=1}^{N_s} \frac{x_m (x_m - 1)}{\nu (\nu - 1)}, \tag{18}$$

from which we can find $\mathbb{P}(1s|\nu)$ as follows

$$\begin{aligned}
\mathbb{P}(1s|\nu) &= \mathbb{E}_{\mathbf{x}|\nu}\left(\sum_{m=1}^{N_s} \frac{x_m (x_m - 1)}{\nu (\nu - 1)}\right) \\
&= \sum_{m=1}^{N_s} \mathbb{E}_{\mathbf{x}|\nu}\left(\frac{x_m (x_m - 1)}{\nu (\nu - 1)}\right) \\
&= \sum_{m=1}^{N_s} \sum_{x_u=0}^{N_c} \frac{x_m (x_m - 1)}{\nu (\nu - 1)} \times \\
&\quad \prod_{u=1}^{N_s-1} \binom{N_c}{x_u} \binom{N_c}{\nu - \sum_{i=1}^{N_s-1} x_i} \binom{N_c N_s}{\nu}^{-1} \\
&= \sum_{m=1}^{N_s} \frac{N_c - 1}{N_s (N_c N_s - 1)} = \frac{N_c - 1}{(N_c N_s - 1)}.
\end{aligned} \tag{19}$$

Next, we need to find $\mathbb{P}\left(\overline{C1}|1s\right) = 1 - \mathbb{P}\left(C1|1s\right)$ where, $\overline{C1} = \{f_{k_1} = f_{k_2}\}$ given the two vertices are induced by the same server.

Let $\mathbf{Z}_k$ be a random vector representing the number of clients requesting file f_k from the same server; $\mathbf{Z}_k \sim Bin\left(N_c, P_s P_w\right)$, where $P_s = \frac{S}{N_f}$ is the probability that a given file is stored at a given server, and $P_w = \frac{1}{N_f}$ is the probability that a given file is wanted by a given client (each client wants only one file). The total number of vertices induced by server s_i can be expressed as $X_i = \sum_{k=1}^{S} z_k$. By following the same steps in (17) we get

$$\mathbb{P}\left(\mathbf{z} = \mathbf{z}' | \sum_{k=1}^{S} Z_k = X_i\right) = \frac{\prod_{u=1}^{S-1} \binom{N_c}{z_u} \binom{N_c}{X_i - \sum_{v=1}^{S-1} z_v}}{\binom{N_c S}{X_i}}, \tag{20}$$

$$\mathbb{P}\left(\overline{C1}|\mathbf{z} = \mathbf{z}', \mathbf{x} = \mathbf{x}', 1s\right) = \sum_{i=1}^{N_s} \sum_{k=1}^{S} \frac{z_k (z_k - 1)}{x_i (x_i - 1)},$$

$$\Rightarrow \mathbb{P}\left(\overline{C1}|1s\right) = \sum_{i=1}^{N_s} \sum_{k=1}^{S} \mathbb{E}_{\mathbf{x}|\nu}\left(\mathbb{E}_{\mathbf{z}|x_i}\left(\frac{z_k (z_k - 1)}{x_i (x_i - 1)}\right)\right),$$

$$\begin{aligned}
\mathbb{E}_{\mathbf{z}|x_i}\left(\frac{z_k (z_k - 1)}{x_i (x_i - 1)}\right) &= \sum_{z_u=0}^{N_c} \frac{z_k (z_k - 1)}{x_i (x_i - 1)} \times \\
&\quad \frac{\prod_{u=1}^{S-1} \binom{N_c}{z_u} \binom{N_c}{X_i - \sum_{v=1}^{S-1} z_v}}{\binom{N_c S}{X_i}} \\
&= \frac{N_c - 1}{S (N_c S - 1)}.
\end{aligned}$$

Since the resulting term does not depend on x, averaging over its distribution would result in the same term. There-

fore

$$\mathbb{P}\left(\overline{C1}|1s\right) = \sum_{k=1}^{S} \frac{N_c - 1}{S\left(N_c S - 1\right)} = \frac{N_c - 1}{N_c S - 1},$$

$$\Rightarrow \mathbb{P}\left(C1|1s\right) = 1 - \frac{N_c - 1}{N_c S - 1}. \tag{21}$$

The next step is to find $\mathbb{P}\left(C2|1s, C1\right) = 1 - \mathbb{P}\left(\overline{C2}|1s, C1\right)$, where $\overline{C2} = \{f_{k_1} \in \mathcal{H}_{c_{j_2}} \text{ AND } f_{k_2} \in \mathcal{H}_{c_{j_1}}\}$. Based on the definition of the conflict-free IDNC graph, it is intuitive to say that $f_{k_1} \notin \mathcal{H}_{c_{j_1}}$ AND $f_{k_2} \notin \mathcal{H}_{c_{j_2}}$. Thus,

$$\begin{aligned}\mathbb{P}\left(\overline{C2}|1s, C1\right) &= \mathbb{P}\left(f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right) \cap f_{k_2} \in \left(\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{s_i}\right)\right) \\ &= \mathbb{P}\left(f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right)\right) \times \\ &\quad \mathbb{P}\left(f_{k_2} \in \left(\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{s_i}\right) | f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right)\right).\end{aligned} \tag{22}$$

To find $\mathbb{P}\left(\overline{C2}|1s, C1\right)$, we need to find the intersection between the has set of s_i and the has set of each client after removing the intersection between them. Let Y be a random variable denoting the number of files in the set $\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{c_{j_2}}$, which can be modelled as follows

$$\mathbb{P}\left(Y = y\right) = \binom{H}{y}\binom{N_f - H}{H - y}\binom{N_f}{H}^{-1}. \tag{23}$$

Let e be the number of the files in the set $\{\left(\mathcal{H}_{c_{j_1}} - \mathcal{H}_{c_{j_2}}\right) \cap \mathcal{H}_{s_i}\}$, which is modelled as

$$\mathbb{P}\left(e = e_1\right) = \binom{S}{e_1}\binom{N_f - S}{H - y - e_1}\binom{N_f}{H - y}^{-1}. \tag{24}$$

Let e' be the number of files in the set $\{\mathcal{H}_{s_i} - \{\left(\mathcal{H}_{c_{j_1}} - \mathcal{H}_{c_{j_2}}\right) \cap \mathcal{H}_{s_i}\}\} \cap \left(\mathcal{H}_{c_{j_2}} - \mathcal{H}_{c_{j_1}}\right)$, which is modelled as

$$\mathbb{P}\left(e' = e_2\right) = \binom{S - e_1}{e_2}\binom{N_f - (S - e_1)}{H - y - e_2}\binom{N_f}{H - y}^{-1}. \tag{25}$$

Using the three variables defined above, $\psi = \mathbb{P}\left(\overline{C2}|1s, C1\right)$ can be expressed as

$$\begin{aligned}\psi = \mathbb{P}\left(\overline{C2}|1s, C1\right) = \sum_{y=0}^{H} \frac{\binom{H}{y}\binom{N_f - H}{H - y}}{\binom{N_f}{H}} \times \\ \sum_{e_1 = max(0, H - (N_f - S))}^{min(H - \mathbf{y}, S)} \frac{\binom{S}{e_1}\binom{N_f - S}{H - y - e_1}}{\binom{N_f}{H - y}} \frac{e_1}{S} \times \\ \sum_{e_2 = max(0, H - (N_f - (S - e_1)))}^{min(H - \mathbf{y}, (S - e_1))} \frac{\binom{S - e_1}{e_2}\binom{N_f - (S - e_1)}{H - y - e_1}}{\binom{N_f}{H - y}} \frac{e_2}{S - 1}.\end{aligned} \tag{26}$$

Hence, $\mathbb{P}\left(C2|1s, C1\right)$ is written as

$$\mathbb{P}\left(C2|1s, C1\right) = 1 - \psi. \tag{27}$$

By substituting (19),(21) and (27) in (16) we get

$$\mathbb{P}\left(E1\right) = \left(\frac{N_c - 1}{\left(N_c N_s - 1\right)}\right)\left(1 - \frac{\left(N_c - 1\right)}{N_c S - 1}\right)\left(1 - \psi\right) \tag{28}$$

To find $\mathbb{P}\left(E2\right) = \{c_{j_1} = c_{j_2} \text{ AND } s_{i_1} \neq s_{i_2}\}$, we need to recall that this event happens if a given client wants a file stored in two different servers. Therefore

$$\mathbb{P}\left(E2\right) = \mathbb{P}\left(f_k \in \mathcal{H}_{s_{i_1}} \; AND \; f_k \in \mathcal{H}_{s_{i_2}}\right) = \frac{R}{\nu}\frac{R - 1}{\nu - 1}. \tag{29}$$

From (28), (29) and (15) we get

$$\pi = \left(\frac{N_c - 1}{\left(N_c N_s - 1\right)}\right)\left(1 - \frac{\left(N_c - 1\right)}{N_c S - 1}\right)\left(1 - \psi\right) + \frac{R\left(R - 1\right)}{\nu\left(\nu - 1\right)}$$

REFERENCES

[1] A. A. Al-Habob, S. Sorour, N. Aboutorab, and P. Sadeghi, "Conflict free network coding for distributed storage networks," in *2015 IEEE Int. Conf. on Commun. (ICC)*, June 2015, pp. 5517–5522.

[2] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A Survey on Network Codes for Distributed Storage," *Proc. of the IEEE*, vol. 99, no. 3, pp. 476–489, March 2011.

[3] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *2012 Proc. IEEE INFOCOM*, March 2012, pp. 2801–2805.

[4] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Peer-to-Peer Systems*. Springer, 2002, pp. 328–337.

[5] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. on Inf. Theory*, vol. 52, no. 6, pp. 2809–2816, June 2006.

[6] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," in *ACM SIGCOMM Comp. Commun. Review*, vol. 36, no. 4. ACM, 2006, pp. 255–266.

[7] U. J. Ferner, P. Sadeghi, N. Aboutorab, and M. Mdard, "Scheduling advantages of network coded storage in point-to-multipoint networks," in *2014 Int. Symp. on Net. Coding (NetCod)*, June 2014, pp. 1–6.

[8] R. Tandon and O. Simeone, "Harnessing cloud and edge synergies: toward an information theory of fog radio access networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 44–50, August 2016.

[9] S. H. Park, O. Simeone, and S. Shamai, "Joint optimization of cloud and edge processing for fog radio access networks," in *2016 IEEE Int. Symp. on Inf. Theory (ISIT)*, July 2016, pp. 315–319.

[10] R. Tandon and O. Simeone, "Cloud-aided wireless networks with edge caching: Fundamental latency trade-offs in fog Radio Access Networks," in *2016 IEEE Int. Symp. on Inf. Theory (ISIT)*, July 2016, pp. 2029–2033.

[11] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *2012 Proc. IEEE INFOCOM*, March 2012, pp. 1107–1115.

[12] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "Cache-aware user association in backhaul-constrained small cell networks," in *2014 12th Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 37–42.

[13] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *2015 IEEE Int. Conf. on Commun. (ICC)*, June 2015, pp. 3358–3363.

[14] S. Krishnan and H. S. Dhillon, "Distributed caching in device-to-device networks: A stochastic geometry perspective," in *2015 49th Asilomar Conf. on Signals, Systems and Computers*, Nov. 2015, pp. 1280–1284.

[15] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. on Select. Areas in Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.

[16] Y. N. Shnaiwer, S. Sorour, N. Aboutorab, P. Sadeghi, and T. Y. Al-Naffouri, "Network-coded content delivery in femtocaching-assisted cellular networks," in *2015 IEEE Global Commun. Conf. (GLOBECOM)*, Dec 2015, pp. 1–6.

[17] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. on Info. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[18] S. K. D. K. W. Hu and H. R. M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," *Newsletter ACM SIGCOMM Comp. Commun. Review*, vol. 36, no. 4, 2006.

[19] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. on Net.*, vol. 16, no. 3, pp. 497–510, June 2008.
[20] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *2009 Workshop on Network Coding, Theory, and Applications*, June 2009, pp. 80–85.
[21] P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP J. on Wireless Commun. and Networking*, vol. 2010, p. 4, 2010.
[22] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. on Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb 2009.
[23] S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network codes," *IEEE/ACM Trans. on Net.*, vol. 23, no. 5, pp. 1553–1567, Oct 2015.
[24] ——, "On minimizing broadcast completion delay for instantly decodable network coding," in *2010 IEEE Int. Conf. on Commun.*, May 2010, pp. 1–5.
[25] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *2013 Int. Symp. on Net. Coding (NetCod)*, June 2013, pp. 1–6.
[26] S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *2010 IEEE Global Telecommunications Conf. GLOBECOM 2010*, Dec. 2010, pp. 1–5.
[27] A. Douik, S. Sorour, T. Y. Al-Naffouri, and M. S. Alouini, "Delay Reduction for Instantly Decodable Network Coding in Persistent Channels With Feedback Imperfections," *IEEE Trans. on Wireless Commun.*, vol. 14, no. 11, pp. 5956–5970, Nov. 2015.
[28] M. S. Karim, P. Sadeghi, S. Sorour, and N. Aboutorab, "Instantly decodable network coding for real-time scalable video broadcast over wireless networks," *EURASIP J. on Advances in Signal Processing*, vol. 2016, no. 1, p. 1, 2016.
[29] M. Karim, S. Sorour, and P. Sadeghi, "Network Coding for Video Distortion Reduction in Device-to-Device Communications," *IEEE Trans, on Veh. Technol.*, vol. PP, no. 99, pp. 1–1, Oct. 2016.
[30] A. Douik, S. Sorour, H. Tembine, T. Y. Al-Naffouri, and M. S. Alouini, "A Game-theoretic Framework for Network Coding Based Device-to-Device Communications," *IEEE Trans. on Mobile Computing*, vol. PP, no. 99, pp. 1–1, June 2016.
[31] A. Douik, S. Sorour, T. Y. Al-Naffouri, H. C. Yang, and M. S. Alouini, "Delay reduction in multi-hop device-to-device communication using network coding," in *2015 Int. Symp. on Network Coding (NetCod)*, June 2015.
[32] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.
[33] B. Bollobás, "The chromatic number of random graphs," *Combinatorica*, vol. 8, no. 1, pp. 49–55, 1988.
[34] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.
[35] S. D. Patek, "On partially observed stochastic shortest path problems," in *Proc. of the 40th IEEE Conf. on Decision and Control (Cat. No.01CH37228)*, vol. 5, Dec 2001, pp. 5050–5055 vol.5.
[36] S. Sorour, A. Douik, S. Valaee, T. Y. Al-Naffouri, and M. S. Alouini, "Partially blind instantly decodable network codes for lossy feedback environment," *IEEE Trans. on Wireless Commun.*, vol. 13, no. 9, pp. 4871–4883, Sept 2014.