Quality-Aware Streaming and Scheduling for Device-to-Device Video Delivery

Joongheon Kim, *Member, IEEE,* Giuseppe Caire, *Fellow, IEEE,* and Andreas F. Molisch, *Fellow, IEEE*

***Abstract*—On-demand video streaming is becoming a killer application for wireless networks. It has been recently shown that a combination of caching on the users' devices and device-to-device (D2D) communications yields throughput scalability for very dense networks, which represent critical bottlenecks for conventional cellular and wireless local area network (WLAN) technologies. In this paper, we consider implementations of such caching D2D systems where each device pre-caches a subset of video files from a library, and users requesting a file that is not in their own library get it delivered through D2D communication. We develop centralized and distributed algorithms for the delivery phase, encompassing a link scheduling and a streaming component. The centralized scheduling is based on the max-weighted independent set (MWIS) principle, and uses message-passing to determine max-weight independent sets. The distributed scheduling is based on a variant of the FlashLinQ link scheduling algorithm, enhanced by introducing video-streaming specific weights. In both cases, the streaming component is based on a quality-aware stochastic optimization approach, reminiscent of current DASH technology (Dynamic Adaptive Streaming over HTTP), for which users sequentially request video "chunks" by choosing adaptively their quality level. The streaming and the scheduling components are coupled by the length of the users' request queues. Through extensive system simulation, the proposed approaches are shown to provide sizeable gains with respect to baseline schemes formed by the concatenation of off-the-shelf FlashLinQ with proportional fair link scheduling and DASH at the application layer.**

***Index Terms*—Adaptive Streaming, Device-to-Device, Video Delivery, Quality-Awareness, Scheduling**

I. Introduction

According to recent predictions of the Cisco Visual Networking Index (VNI) [11], the sum of all forms of video will constitute 80% to 90% of global consumer data traffic by 2017, and the traffic from wireless and mobile devices will exceed the traffic from wired devices by 2016. Therefore, efficient video-aware network algorithms for wireless networks are of highest importance [1][21]-[42]. It has been shown recently [1], [20], [36], [38], [39], [40], [41], [42] that the throughput for delivery of wireless video files can be greatly enhanced by device-to-device (D2D) communications, where direct links between pairs of user devices can be set up without requiring to go through a central base station. In particular, these works propose systems where each device caches independently, according to a certain optimal distribution, a subset of popular video files. When a user needs a file not already present in its own cache, it obtains it from one of its neighbors through a spectrally efficient, short-range D2D link. As user density increases, the aggregate storage capacity of the D2D network increases linearly with the number of users, while the average communication distance decreases (and the spatial reuse increases). For these reasons, D2D networks for video delivery are scalable, such that both demand and throughput increase linearly with user density.

The most common way users consume video is by streaming, i.e., after a pre-buffering time typically much shorter than the duration of the video file, playback is started while, at the same time, the rest of the file is progressively downloaded. More specifically, the video file is divided into "chunks" such that while the already received chunks are played back in sequence, the later chunks are transmitted. A transmission algorithm for such a system consists of two components: (i) a scheduling algorithm that determines which D2D pairs are allowed to transmit at a given time, and (ii) a streaming algorithm that determines adaptively from which device each chunk should be requested and at which quality level. These two components are obviously coupled.

Currently, the most well-known D2D scheduling protocol in both industry and academia is *FlashLinQ* [7], [8]. It is a distributed algorithm that schedules D2D links according to their priorities, such that the higher-priority links do not suffer from significant interference of possibly scheduled lower-priority links. Theoretically, it can guarantee the maximum number of activated D2D links as analyzed by the theory of stochastic geometry [10]. However, FlashLinQ does not incorporate naturally a video quality-aware mechanism, and therefore its suitability for D2D on-demand video streaming remains open.

As far as video streaming quality adaptation is concerned, a number of protocols have been suggested in the past. Such protocols consider various characteristics of video streaming, e.g., cloud-based video streaming protocols are proposed in [45], [46], channel-aware streaming algorithms are discussed in [44], new architectural concepts are presented in [47], rate-distortion theory based (or quality-aware) streaming is addressed in [48], [49], and resource-aware streaming algorithms are mentioned in [50].

The key postulate of this paper is that D2D scheduling and streaming are *coupled*, and that therefore a *joint* algorithm has to be developed for this task. In this paper we propose and analyze both centralized and distributed algorithms that combine aspects of the above-mentioned methods.

For the proposed *centralized* scheduling and streaming algorithm, we utilize the benefits of cellular centralized resources,

J. Kim is with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA e-mail: joonghek@usc.edu.

G. Caire and A.F. Molisch are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA e-mails: {caire, molisch}@usc.edu.

i.e., while the devices communicate directly with each other, they are under control of an existing base station (BS); see [12] for a discussion and further references of such systems. Thus, the scheduling decision can be made by the cellular BS and we assume that the BS knows the channel state information (CSI) for all the possible pairs of D2D links. The scheduling component of the proposed scheme is based on a link conflict graph (e.g., formed centrally by the BS) such that the links scheduled to be active simultaneously at any time slot must form an independent set of such conflict graph. This conflict graph based scheduling can be formulated as a max-weight independent set (MWIS) problem. The MWIS problem is known to be NP-hard; in this work we make use of a message-passing algorithm proposed in [15], [16], to (approximately) solve the MWIS problem. Therefore, our *scheduling* component is designed based on this message-passing concept with D2D-related modifications, i.e., the weight is not only based on the queue backlog size but also on the CSI. Our distributed scheduling scheme is based on FlashLinQ, improved by incorporating weights (similar to MWIS) and thus providing a connection to the streaming decisions for video quality.

Our *streaming* component is based on stochastic network optimization with the consideration of the quality of video streaming. Each video consists of a number of chunks. Each chunk can be requested at different quality levels,[1] such that higher quality corresponds to more bits per chunk to be delivered. Therefore, our algorithm dynamically controls the quality mode of each chunk to maximize total quality subject to all data being supportable over the network. Note that the streaming decisions impact the weights (for MWIS or modified FlashLinQ) of the scheduling.

This work extends previous investigations of adaptive video streaming algorithms [17], [18]; and we retain the notation of those papers for streaming-related aspects. The algorithms in [17], [18] are for adaptive stochastic video streaming for cache/helper (equivalent to access points plus video database in WiFi-based networks) and apply to a bipartite network topology where a set of infrastructure nodes (small cell base stations with cached video files) serve a set of wireless users.

Furthermore, our work differs from these papers as follows:

- In [17], [18] it is assumed that each user can be served simultaneously by multiple infrastructure nodes over each scheduling slot. Instead, here we explicitly consider the constraint of the D2D link conflict graph, such that at each scheduling slot a user can only be served by another (peered) user device, if the corresponding link belongs to the scheduled independent set.
- The algorithms in [17], [18] dynamically match source and destination pairs in every single unit time operation. However, the algorithms in this paper work on fixed source-destination pairs, as this is the relevant case for D2D communications. Let us note that FlashLinQ also considers this case [7], [8].

[1]For example, this can be obtained by storing multiple copies of the same video encoded at different rates, as in current video on-demand delivery systems such as Netflix or Amazon Prime, or by using scalable video coding and requesting more or fewer refinement layers [4], [5], [6].

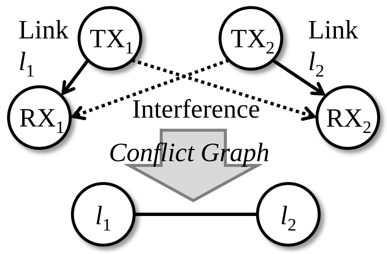


Fig. 1. An Example of a Conflict Graph

- Since user devices are paired permanently over a whole streaming session in the present work, there is no hand-over delay, while per-slot dynamic association used in [17], [18] gives rise to such delay. Such delay must be taken into account or made small through a special network architecture, e.g., single-channel single-IP implementation [2], [3].

We evaluate the performance of our proposed algorithms by extensive simulations and compare them with a baseline scheme formed by FlashLinQ at the MAC layer and DASH at the application layer. In particular, we study the system performance in terms of total throughput, video quality (PSNR), and number of video streaming stall events[2] at the receivers. According to our simulation results, the proposed algorithms provide sizeable performance gains for these quality measures.

The remainder of this paper is organized as follows: Section II gives preliminaries and background information. Section III explains the details of our proposed quality-aware streaming and scheduling algorithms both for the centralized and the distributed cases. Section IV shows the simulation results compared to FlashLinQ variants. Section V concludes this paper.

II. Preliminaries

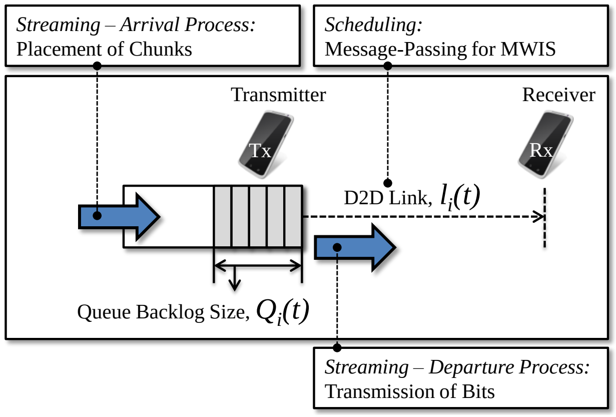
This section defines our reference model including (i) network model (see Section II-A), (ii) link model (see Section II-B), and (iii) video streaming model (see Section II-C).

A. A Reference Network Model: Macro View

Consider a network formed by a set $\mathcal{L}$ of one-hop D2D links, indicated by $l_i \in \mathcal{L}$ [12], [13]. To schedule the D2D links, a conflict graph is constructed such that the set of vertices is $\mathcal{L}$ (the links) and two vertices are connected by an edge if the corresponding links suffer from mutual interference above a certain desired threshold (refer to Fig. 1). The choice of the threshold will be discussed in Section IV-B3. However, for the computation of the achievable rates of each link, we still need to take into account the residual interference, caused by the transmission from simultaneously scheduled links.

The conflict graph is described through its adjacency matrix, whose elements $\mathcal{E}_{(j,k)}$ between $l_j \in \mathcal{L}$ and $l_k \in \mathcal{L}$ are defined as follows:

[2]When the playback buffer does not contain the required video chunk at its due playback time such that playback has to stall and wait until such chunk is delivered.



Streaming – Arrival Process: Placement of Chunks
Scheduling: Message-Passing for MWIS
Transmitter
Receiver
Tx
Rx
D2D Link, $l_i(t)$
Queue Backlog Size, $Q_i(t)$
Streaming – Departure Process: Transmission of Bits

Fig. 2. A D2D Link Model

$$\mathcal{E}_{(j,k)} = \begin{cases} 1, & \text{if } l_j \text{ interferes with } l_k \text{ where} \\ & l_j \in \mathcal{L},\, l_k \in \mathcal{L}, \text{ and } j \neq k, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In addition, the set of neighbor nodes of each node is defined as follows:

$$\mathcal{N}(i) \triangleq \left\{ l_a \middle| \mathcal{E}_{(i,a)} = 1 \text{ where } l_a \in \mathcal{L} \right\}, \forall l_i \in \mathcal{L}. \quad (2)$$

B. A Reference Link Model: Micro View

As can be seen in Fig. 2, each D2D wireless link consists of one transmitter and its associated receiver. Each transmitter has a queue whose length evolves according to

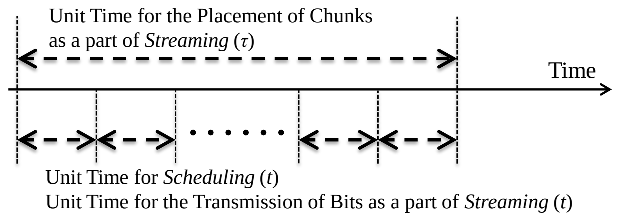
$$Q_i(t+1) = \max\left[0, Q_i(t) - \mu_i(t)\right] + \lambda_i(t) \quad (3)$$

where $Q_i(t)$, $\mu_i(t)$, and $\lambda_i(t)$ stand for the queue backlog size at the transmitter of l_i, the number of bits leaving the queue of the transmitter of l_i, and the number of bits added to the queue of the transmitter of l_i, respectively. As shown in Fig. 2, the arrival process (bits added to the queue) is associated with the placement of chunks of the currently served video in each unit time $t \in \{0, 1, \cdots\}$. When the D2D link is scheduled for transmission by the centralized or distributed controller, the queue has a departure process which depends on the channel state and on the link scheduling decisions. More details are provided in Section III.

C. Video Streaming System Model

The receiver of link l_i requests a video file f_i that is located in the cache of its associated transmitter. A video file is formed by a sequence of *chunks*, i.e., group of pictures (GOPs), which are encoded and decoded as stand-alone units. Chunks must be reproduced in sequence at the D2D receivers. The streaming thus consists of the transmission of sequential chunks from the transmitter to its associated receiver such that the playback buffer at each transmitter contains the required chunks at the beginning of each chunk playback time.

The time scale for the scheduling decision and departure process, i.e., t, is not equivalent to the chunk placement unit time τ, as can be seen in Fig. 3.



Unit Time for the Placement of Chunks as a part of *Streaming* (τ)
Time
Unit Time for *Scheduling* (t)
Unit Time for the Transmission of Bits as a part of *Streaming* (t)

Fig. 3. Two Differentiated Unit Time Scales

A chunk contains $\mathcal{N} = N_{\text{fpc}} \cdot N_{\text{ppf}}$ pixels where N_{ppf} denotes the number of pixels per frame and N_{fpc} stands for the number of frames per chunk. Suppose that each chunk of each file f is encoded at a number of different quality modes $q \in M$ where $M = \{q_1 \cdots q_M\}$. According to the variable bit-rate nature of video coding, the quality-rate profile may vary from chunk to chunk. We let $\mathbb{P}_f(q,\tau)$ and $\mathcal{N}\mathbb{B}_f(q,\tau)$ denote the video quality measure (e.g., peak-signal-to-noise-ratio (PSNR))[3] and the number of bits for file f at chunk time τ with quality mode q, respectively.

The chunk placement procedure consists of choosing the quality mode $q_i(\tau)$ of the chunks requested at chunk unit time τ by the scheduled D2D transmitters i. The choice $q_i(\tau)$ renders the point $(\mathbb{P}_{f_i}(q_i(\tau),\tau); \mathcal{N}\mathbb{B}_{f_i}(q_i(\tau),\tau))$ from the finite set of quality-rate tradeoff points $\{(\mathbb{P}_{f_i}(q,\tau), \mathcal{N}\mathbb{B}_{f_i}(q,\tau))\}_{q=q_1}^{q=q_M}$. The network controller (i) chooses the quality mode $q_i(\tau)$ for chunk time τ for all requesting receivers i, and (ii) allocates the source coding rate (bit per pixel). The transmitter of link i places the corresponding $\mathcal{N}\mathbb{B}_{f_i}(q_i(\tau),\tau)$ bits in its transmission queue $Q_i(\tau)$, to be sent to the receiver whose length evolves according to.

Summarizing, as can be seen in Fig. 4, at each chunk time τ, the transmitter fetches chunks from its local cache in the order in which they are to be played back. The chunks are fetched at the quality mode $q_i(\tau)$ that is computed based on stochastic network optimization algorithms (details are in Section III-C1). Then, the coded bits are packetized to be transmitted over the air-interfaces. The enqueued packets will be transmitted depending on the departure process $\mu_i(t)$ and are dependent on channel states as well as interference from activated neighbor D2D links, i.e., signal-to-interference-plus-noise (SINR) ratio.

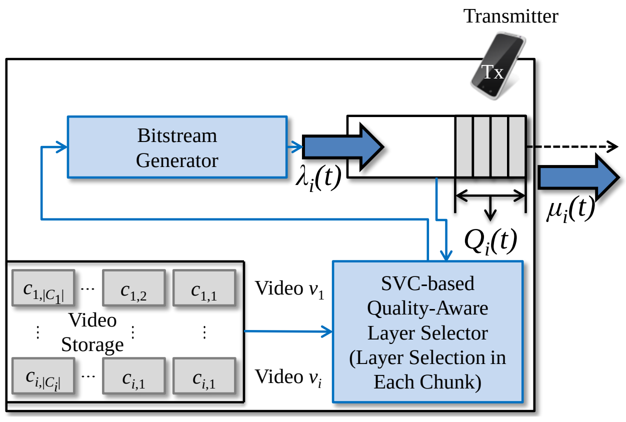
III. Quality-Aware Scheduling and Streaming for Device-to-Device Video Delivery

This section presents the basic design rationale of our proposed two quality-aware streaming and scheduling algorithms.

A. Design Rationale

The proposed centralized algorithm consists of two separable but interconnected parts, i.e., *centralized scheduling based*

[3]There is a rich literature on video quality metrics, e.g., [49], [51], [52]. Our framework works with any video quality measure, but for the sake of simplicity (and because details of quality measures are outside the scope of this paper), we use PSNR henceforth.



Transmitter
Tx
Bitstream Generator
$\lambda_i(t)$
$\mu_i(t)$
$Q_i(t)$
$c_{1,|C_1|}$ ··· $c_{1,2}$ $c_{1,1}$
Video Storage
$c_{i,|C_i|}$ ··· $c_{i,1}$ $c_{i,1}$
Video v_1
Video v_i
SVC-based Quality-Aware Layer Selector (Layer Selection in Each Chunk)

Fig. 4. A Device Model: The *Scalable Video Coding (SVC)-based Quality-Aware Layer Selector* determines the number of layers for each chunk via the proposed algorithm in Section III-C based on the transmitter queue backlog size. Note that a detailed SVC architecture for adaptive layer selection is illustrated in [53].

on MWIS (refer to Section III-B1) and *quality-aware streaming* (refer to Section III-C). In order to (approximately) solve the MWIS problem in a computationally efficient manner, we resort to a message-passing approach. For the streaming decision, the operations to control the arrival and departure processes in the queue of each D2D transmitter are defined. The entire link model is illustrated in Fig. 2.

The proposed distributed algorithm improves FlashLinQ with the concepts of *distributed max-weight scheduling* before transmission (refer to Section III-B2) and quality-aware streaming (refer to Section III-C).

B. Device-to-Device Scheduling

1) Centralized Scheduling with MWIS Formulation: For centralized D2D scheduling, the objective is to find the set of links (i.e., nodes of the conflict graph defined before), that maximize the sum of weights over all possible independent sets. This yields the MWIS problem:

$$\max: \quad \mathcal{F}(\mathcal{I}) \triangleq \sum\nolimits_{\forall l_i \in \mathcal{L}} w_i \mathcal{I}_i, \tag{4}$$

$$\text{s.t.} \quad \mathcal{I}_j + \mathcal{I}_k + \mathcal{E}_{j,k} \leq 2, \forall l_j \in \mathcal{L}, \forall l_k \in \mathcal{L}, \tag{5}$$

$$\mathcal{I}_i \in \{0,1\}, \forall l_i \in \mathcal{L}, \tag{6}$$

where $\mathcal{I}_i$ is defined as

$$\mathcal{I}_i = \begin{cases} 1, & \text{if } l_i \text{ is scheduled where } l_i \in \mathcal{L}, \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The above formulation ensures that conflicting links are not scheduled simultaneously: if $\mathcal{E}_{j,k} = 0$ (no edge between l_j and l_k) then $\mathcal{I}_j + \mathcal{I}_k \leq 2$, i.e., both indicator functions can be equal to 1. In contrast, if $\mathcal{E}_{j,k} = 1$, $\mathcal{I}_j + \mathcal{I}_k \leq 1$, i.e., at most one of the two indicator functions can be equal to 1.

In (4), $w_i, \forall i \in \{1, \cdots, |\mathcal{L}|\}$ is given by [19]:

$$w_i \triangleq r_i(t) \cdot Q_i(t) \tag{8}$$

where $Q_i(t)$ is the queue backlog size at the transmitter of D2D link l_i, defined in (3), and $r_i(t)$ stands for the achievable rates of D2D link l_i.

The exact value of $r_i(t)$ of D2D link l_i cannot be obtained before a scheduling decision is made because it depends on the actual interference that all active transmitters (of the scheduled links) cause on the receiver of link l_i, which is known only when the scheduling decision is actually made.

To circumvent this problem, the decision in (4) is made on the basis of an estimated value of the link achievable rate, given by:

$$r_i(t) = \log_2\left(1 + \frac{\mathcal{P}_{s_i \to d_i}(t) \left\| h_{i\to i} \right\|^2}{\sigma^2 + \gamma}\right) \tag{9}$$

where $\mathcal{P}_{s_i \to d_i}(t)$ stands for the transmit power from s_i at unit time t, $h_{i\to i}$ stands for the (complex amplitude) channel gain from s_i to d_i, σ is the standard deviation of the (Gaussian) background noise, γ stands for the interference thresholds, i.e., the maximum admissible interference level γ from a single interferer scheduled at the same time as the considered link. In FlashLinQ, γ is set to 9 dB [7], [8]. (9) implies the assumption that the *aggregate* interference from other links is equal to the interference from a maximally strong single interferer. Since the overall interference levels tend to be dominated by the strongest interferer [43], this is a reasonable approximation.

After solving this MWIS problem, a set of active links is obtained and the actual rates (including all the interference caused by the active transmitters on the link receivers) are used to update the transmission queues (see later). For solving the MWIS problem, various heuristic and approximation algorithms have been proposed due to the fact that MWIS is a well-known NP-hard problem. One of these methods is the computation with *message-passing* [15], [16] which we will use henceforth. The corresponding pseudo-code is presented in Algorithm 1.

2) Distributed Max-Weight Scheduling: For distributed scheduling, we improve FlashLinQ with the concept of max-weight scheduling.

FlashLinQ sorts the links according to a priority order externally determined (e.g., at random, or according to round robin), and considers the links in sequence, according to the priority order. A link is declared active if it does not create significant interference to the already active links with higher priority and if its own achieved rate, considering the interference from the already active links with higher priority, is large enough. The decision is based on measurements of channel strengths in both directions [7], [8]. In the original FlashLinQ, priorities are randomized over time, to provide a basic level of fairness. With the concept of max-weight scheduling, we set the priorities of D2D links instead as follows:

$$U_i \triangleq \frac{1}{r_i(t) \cdot Q_i(t)}. \tag{10}$$

C. Streaming with Quality-Aware Stochastic Control

The streaming consists of two parts, i.e., (i) placement of chunks (i.e., arrival process of the queue) and (ii) transmission of bits (i.e., departure process of the queue). Notice that the streaming method investigated in this section is used for both centralized and distributed algorithms.

```
• Input
    – $w_i$ in (8) where $\forall l_i \in \mathcal{L}$
    – $\mathcal{E}_{(j,k)}$ in (1) where $\forall l_j \in \mathcal{L}, \forall l_k \in \mathcal{L}$
• Output
    – $\mathcal{F}(\mathcal{L}, \mathcal{E})$
    – $\mathcal{L}^*$ // set of scheduled D2D links
Update Phase;
$n = 1$;
while $n \le K$ do
    // $K$: the number of message-passing iteration;
    $m^n_{i\to j} = \max\left[0, w_i - \sum_{k\in\{N(i)-j\}} m^{n-1}_{k\to i}\right], \forall j \in \mathcal{N}(i)$;
    $i$ sends $m^n_{i\to j}$ to all $j \in \mathcal{N}(i)$;
    $n++$;
end
Estimation Phase;
$\mathcal{I}_i = \begin{cases} 1 & \textbf{if } \sum_{k\in\mathcal{N}(i)} m^K_{k\to i} < w_i \\ 0 & \textbf{otherwise} \end{cases}$;
If $\mathcal{I}_i = 1$ then $l_i \in \mathcal{L}^*$;
MWIS Computation Phase;
$\mathcal{F}(\mathcal{L}, \mathcal{E}) = \sum_{\forall l_i \in \mathcal{L}} w_i \mathcal{I}_i$;
```

Algorithm 1: MWIS-based scheduling with message-passing in each $l_i \in \mathcal{L}, \forall i \in \{1, \cdots, |\mathcal{L}|\}$

1) Arrival Process (Placement of Chunks): In each chunk time slot $\tau \in \{0, 1, \cdots\}$, the transmitter of each link places a chunk into its transmission queue.

In order to dynamically and adaptively select the quality level of the chunks, we consider the following stochastic optimization approach that aims at maximizing the total average video quality of the users. Let $\mathbb{P}(t) = \sum_{l_i\in\mathcal{L}} \mathbb{P}_{f_i}(q_i(t), t)$; and

$$\mathbb{P}_{f_i}(q_i(t), t) = \begin{cases} \mathbb{P}_{f_i}(q_i(t), t), & \tau \bmod t = 0, \\ 0, & \tau \bmod t \neq 0. \end{cases} \quad (11)$$

Then, the proposed stochastic optimization problem is given by:

$$\max \quad \lim_{t\to\infty} \frac{1}{t} \sum_{t^*=0}^{t-1} \mathbb{E}\left[\mathbb{P}(t^*)\right] \quad (12)$$

$$\text{subject to} \quad \lim_{t\to\infty} \frac{1}{t} \sum_{t^*=0}^{t-1} \mathbb{E}\left[Q_i(t^*)\right] < \infty, \forall l_i \in \mathcal{L} \quad (13)$$

where (13) means all given queues should fulfill mean rate stability. Let $\Theta(t)$ denote the column vector of all scheduled queues at time t, and define the quadratic Lyapunov function

$$L(t) = \frac{1}{2}\Theta^{\mathsf{T}}(t)\Theta(t) = \frac{1}{2}\sum\nolimits_{\forall i\in\mathcal{L}} |Q_i(t)|^2 \quad (14)$$

where $\Theta^{\mathsf{T}}(t)$ stands for the transpose of $\Theta(t)$. Then, let $\Delta(t)$ be a conditional quadratic Lyapunov function that can be formulated as $\mathbb{E}\left[L(t+1)|\Theta(t)\right] - L(t)$, i.e., the drift on slot t. The *drift-plus-penalty (DPP)* policy is designed to solve the given optimization formulation by observing only the current queue backlog sizes $\Theta(t)$ and choose q (i.e., quality mode) to maximize a bound on

$$\mathbb{P}(t) - \alpha\Delta(t) \quad (15)$$

where α is a positive constant control parameter of the DPP policy that affects the quality-delay tradeoffs [19].

Now, the quality control decision involves choosing $q_i(t)$, the quality mode for all scheduled receivers at chunk time t[4]. This choice is made as

$$\boxed{\arg\max_{q_i(t)\in M} \left[\mathbb{P}_{f_i}(q_i(t), t) - \alpha\left\{\mathcal{N}\mathbb{B}_{f_i}(q_i(t), t)\right\} Q_i(t)\right].} \quad (16)$$

Since the placement of chunks constitutes the arrival process of the queue, $\lambda_i(t)$ can be denoted as follows when the optimal q is determined in each $l_i \in \mathcal{L}$:

$$\lambda_i(t) = \begin{cases} \mathcal{N}\mathbb{B}_{f_i}(q_i(t), t), & \tau \bmod t = 0, \\ 0, & \tau \bmod t \neq 0. \end{cases} \quad (17)$$

2) Departure Process (Transmission of Bits): Once a set of active links $\mathcal{L}^*$ is determined as described in Section III-B, the transmitters of the scheduled links can transmit bits up to the amount of the achievable rates actually supported by the link at time t, i.e., $\mu_i(t) = r_i(t), \forall l_i \in \mathcal{L}^*$.

According to Shannon's capacity equation, i.e., $\mu_i(t)$ in (8) can be computed as follows [14]:

$$\mu_i(t) = \mathcal{B} \cdot \log_2\left[1 + \frac{\mathcal{P}_{s_i\to d_i}(t)\,\|h_{i\to i}\|^2}{\sigma^2 + \sum_{j\neq i}\mathcal{P}_{s_j\to d_i}(t)\,\|h_{j\to i}\|^2}\right] \quad (18)$$

where $\forall l_i \in \mathcal{L}^*, \forall l_j \in \mathcal{L}^*, i \neq j$, $\mathcal{P}_{s_a\to d_b}(t)$ stands for the power transmitted by s_a intended for d_b, and $h_{j\to i}$ stands for the channel gain from the transmitter of link j to the receiver of link i where $\forall a, \forall b \in \{1, \cdots, |\mathcal{L}|\}$ at time t, $\mathcal{B}$ stands for the channel bandwidth of the system.

While here we have used the SINR-based capacity equation in (18) to evaluate achievable rates, any suitable function of SINR can be included in our algorithms, for example, if the physical layer (PHY) of the D2D system includes a family of modulation and coding scheme (MCS) each of which has a certain operational range of SINR and a given rate, we can substitute such a piece-wise constant function into our scheme and get meaningful results that explicitly include the properties of the MCS set (e.g., the MCS modes of 802.11-based standards, or 3GPP LTE).

IV. Simulation Study

The performance of our proposed joint scheduling and streaming algorithms is simulated and evaluated in this section. The basic simulation settings are presented in Section IV-A; and the simulation results are presented and analyzed in Section IV-B.

[4]This quality mode $q_i(t)$ selection decision also determines $\Phi(t)$ the column vector of the number of source-coded bits $\mathcal{N}\mathbb{B}_{f_i}(q_i(t), t)$ with selected quality mode $q_i(t)$ that each receiver i must download from its transmitter for the chunk requested at time t.

TABLE I
VIDEO TRACE INFORMATION

	Basic Information (Names of Test Sequences)	Resolution	Average Bitrates (Full Video Stream with All Layers)
Video Trace 1	`highway`	352×288 Pixels	631 Kbps (8 Layers Encoded)
Video Trace 2	`city`, `crew`, `harbour`, `train`	704×576 Pixels	3908 Kbps (4 Layers Encoded)
Video Trace 3	`parkrun`, `stockholm`	1024×576 Pixels	6679 Kbps (4 Layers Encoded)
Video Trace 4	`bridge_close`, `bridge_far`	352×288 Pixels	556 Kbps (8 Layers Encoded)

A. Simulation Settings

1) Video Traces: For the simulation study, we use four different types of video traces. Each video consists of 14400 chunks where the playback time of each chunk is 0.5 seconds. Thus, the overall playback time of each video trace is 2 hours, corresponding to a typical movie playback time. The video sequences are standard moving picture experts group (MPEG) test sequences, commonly used in the literature. The original video sequences consist of 200 chunks; To create one 2-hour video, we concatenated the same sequence 72 times. Details of the traces are summarized in Table I.

The quality of each chunk can be numerically represented by the PSNR, i.e., $\mathbb{P}_{f_i}(q_i(t), t)$.

We note that the video streams are not synchronized between the D2D links, i.e., starting times for the different links (streams) are independent of each other.

2) Baseline Terms of Comparison: To show the effectiveness of our proposed centralized or distributed quality-aware streaming and scheduling algorithms, their performances are evaluated and compared with the performances of FlashLinQ variants.

- *FlashLinQ:* This is the standard FlashLinQ algorithm with random U_i priority selection in each D2D link, such that each link has at least $\frac{1}{N}$ probability of being scheduled where N stands for the number of D2D links. Since FlashLinQ does not consider video quality at all, we fix the quality level as 2 in video trace 1 (among given 4 levels), 4 in video trace 2 (among given 8 levels), 4 in video trace 3 (among given 8 levels), and 2 in video trace 4 (among given 4 levels). We choose a "medium" quality level since a selected high quality level leads to high PSNR but might negatively impact queue stability.
- *FlashLinQ-P:* This variant of FlashLinQ uses prioritized U_i selection. The U_i are computed as

$$U_i = \frac{1}{r_i(t) \cdot Q_i(t)} \tag{19}$$

  corresponding to the max-weight scheduling concept. Also *FlashLinQ-P* does not consider video-quality related aspects, and we again fix the qualities of the streams to the same values as above.
- *FlashLinQ-Q:* This variant of FlashLinQ uses the video streaming quality decisions as in Section III-C, but the link scheduling is standard FlashLinQ, i.e., the priorities U_i are chosen at random.

3) Simulation Topology Construction: The considering simulation topology consists of uniformly random deployed ten D2D transmitter and receiver pairs in a 600m $\times$ 600m

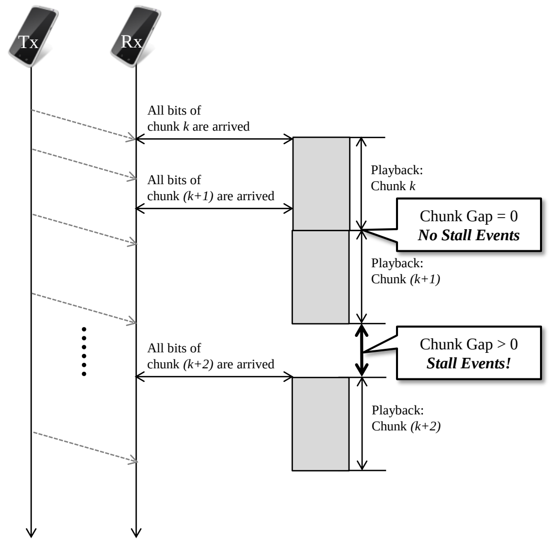
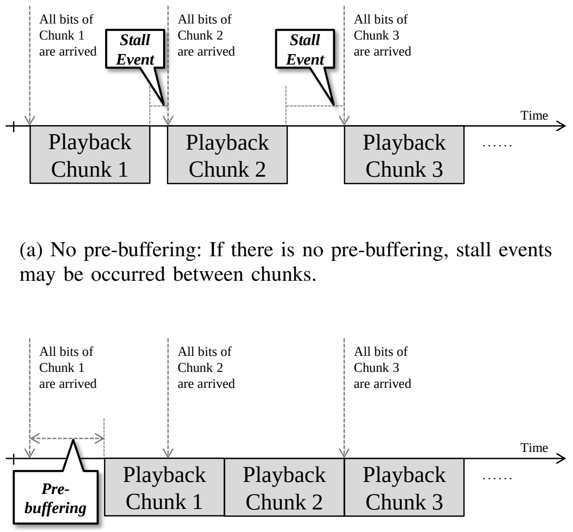

Tx
Rx
All bits of chunk k are arrived
All bits of chunk (k+1) are arrived
All bits of chunk (k+2) are arrived
Playback: Chunk k
Chunk Gap = 0 No Stall Events
Playback: Chunk (k+1)
Chunk Gap > 0 Stall Events!
Playback: Chunk (k+2)

Fig. 5. An example of stall events: If all bits of next chunk are arrived at the playback buffer of D2D receiver, there is no stall event since the receiver can immediately play the next chunk when the playing of current chunk is completed. Otherwise, the stall event will be occurred when the playing of current chunk is completed.


All bits of Chunk 1 are arrived
Stall Event
All bits of Chunk 2 are arrived
Stall Event
All bits of Chunk 3 are arrived
Time
Playback Chunk 1
Playback Chunk 2
Playback Chunk 3
……
Pre-buffering

(a) No pre-buffering: If there is no pre-buffering, stall events may be occurred between chunks.

(b) Pre-buffering: By setting certain amounts of pre-buffering time, we can reduce the number of stall events.

Fig. 6. Pre-buffering: With the definition of pre-buffering, the number of stall events can be reduced, i.e., user satisfaction can be increased.

square layout. The path-loss is computed according to the Winner II model (indoor in 2.4 GHz D2D communications) [20]:

$$PL(d) = a_1 \log_{10}(d) + a_2 + a_3 \log_{10}\left(\frac{f_c^{\mathrm{GHz}}}{5}\right) + X_\sigma \quad (20)$$

where f_c^{GHz} is the carrier frequency in a GHz scale. a_1 includes the path loss exponent and its value is 18.7 dBm in LOS and 36.8 dBm in NLOS. a_2 is the intercept, which is 46.8 dBm in LOS and 43.8 dBm in NLOS. a_3 describes the path loss frequency dependence and it is set to 20 in both LOS and NLOS. X_σ is the shadowing assumed to be a normal distribution (in dB) with mean 0 and standard deviation σ, where $\sigma = 3$ dB in LOS and $\sigma = 6$ dB in NLOS. Notice that we assume that no communication is possible for a distance longer than 100 m.

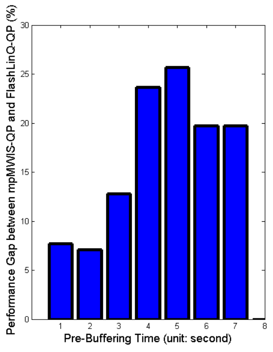
With the given topology, we simulated transmission for 24 hours, assuming each D2D link streams 10 video traces. Then, ten simulations will be operated with different random geometries.

The connectivity of conflict graphs changes according to the setting of the interference threshold. If the received powers from nearby D2D transmitters j to current D2D receiver i are less than the interference threshold, they will be considered as noise. Otherwise, they will be considered as interference. Thus, low γ increases the number of edges in the corresponding conflict graph. Consequently, a relatively small number of D2D links will be scheduled; that may also reduce the sum rate. On the other hand, high γ decreases the number of edges in the given corresponding conflict graph. Thus, a relatively large number of D2D links can be scheduled; however it will reduce the signal-to-interference-plus-noise ratio and thus rate, for each link. Therefore, we need to consider various interference threshold settings in the simulation studies.

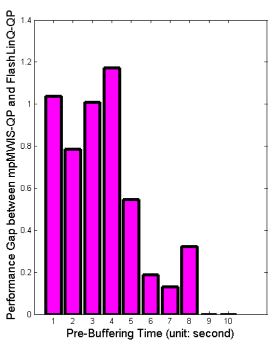
4) Performance Analysis Metrics: Whenever a receiver finishes playing back chunk i, all bits of chunk $i+1$ should have arrived. Otherwise, a pause (*stall*) occurs in the playback, see Fig. 5. Obviously, the stall events introduce user dissatisfaction. Typically, 3 or more stalls during one movie would be judged to be unacceptable quality. Thus the number of stall events can be an important index for measuring user satisfaction of video streaming.

To avoid stall events, *pre-buffering* is frequently used. As shown in Fig. 6, a number of chunks are transmitted before playback at the receiver starts; this introduces a viewing delay for the user but reduces the number of stall events. Obviously a very long pre-buffering time leads to user dissatisfaction as well, and if we set extremely large pre-buffering time, there is no difference between streaming and downloading/transmission. Therefore, there exists a tradeoff, and defining an appropriate pre-buffering time is required. In addition, once a stall occurs, the receiver buffers again for the same amount of pre-buffering time as for startup[5].

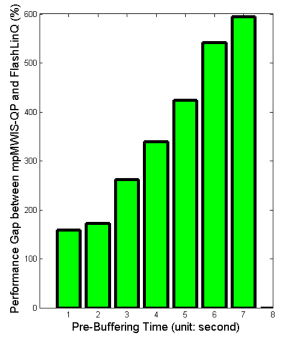
[5]Notice that the re-buffering time could also be shorter than the initial buffering time, i.e., they do not need to be the same, but we set it as equal in order to reduce the number of variables in the simulation.



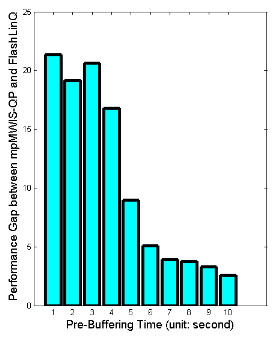
(a) Effectiveness of mpMWIS-QP compared to FlashLinQ-QP in terms of $\mathcal{T}_s$ defined in (21)



(b) Effectiveness of mpMWIS-QP compared to FlashLinQ-QP in terms of $\mathcal{M}_s$ defined in (22)



(c) Effectiveness of mpMWIS-QP compared to FlashLinQ in terms of $\mathcal{T}_s$ defined in (21)



(d) Effectiveness of mpMWIS-QP compared to FlashLinQ in terms of $\mathcal{M}_s$ defined in (22)

Fig. 7. Performance Comparison between mpMWIS-QP and FlashLinQ-QP/FlashLinQ in terms of the Expected Number of Stall Events

B. Simulation Results

With the given two performance metrics, we evaluates the performance of our proposed algorithms and three various FlashLinQ variants as a function of the following parameters:

- various pre-buffering time settings (see Section IV-B1),
- various α, i.e., quality-delay tradeoffs (see Section IV-B2),
- various interference thresholds γ (see Section IV-B3), and
- average quality vs. the expected number of stall events (see Section IV-B4).

Notice that our proposed centralized algorithm is denoted as mpMWIS-QP (i.e., *m*essage-*p*assing for *MWIS* formulation with *Q*uality-awareness and max-weight *P*rioritization); and also our proposed distributed algorithm is denoted as FlashLinQ-QP (i.e., improved *FlashLinQ* with *Q*uality-awareness and max-weight *P*rioritization).

1) Effects of the Pre-Buffering Time: We vary the pre-buffering time from 1 second to 10 seconds with a step size of 1 second. In addition, α and the interference threshold γ are set to 2 and 5 dB, respectively. The resulting expected stall

TABLE II
THE EXPECTED NUMBER OF STALL EVENTS, I.E., $\mathbb{E}[N_s]$, IN EACH VIDEO STREAMING IN EACH D2D LINK WITH VARIOUS PRE-BUFFERING TIMES

Pre-buffering Time	mpMWIS-QP: $\mathbb{E}[N_s]$	FlashLinQ: $\mathbb{E}[N_s]$	FlashLinQ-P: $\mathbb{E}[N_s]$	FlashLinQ-Q: $\mathbb{E}[N_s]$	FlashLinQ-QP: $\mathbb{E}[N_s]$
1 second	13.4	34.8	26.0	26.6	14.5
2 second	11.1	30.2	20.9	22.9	11.9
3 second	7.9	28.5	15.5	18.3	8.9
4 second	5.0	21.8	10.2	11.3	6.1
5 second	2.1	11.1	4.4	5.7	2.7
6 second	0.9	6.0	2.2	3.0	1.1
7 second	0.7	4.6	1.6	2.4	0.8
8 second	**0** [No Stalls]	3.7	1.2	1.9	0.3
9 second	**0** [No Stalls]	3.3	0.6	1.1	**0** [No Stalls]
10 second	**0** [No Stalls]	2.6	**0** [No Stalls]	0.3	**0** [No Stalls]

TABLE III
THE EXPECTED NUMBER OF STALL EVENTS, I.E., $\mathbb{E}[N_s]$, IN EACH VIDEO STREAMING IN EACH D2D LINK WITH VARIOUS α

α	mpMWIS-QP: $\mathbb{E}[N_s]$	FlashLinQ: $\mathbb{E}[N_s]$	FlashLinQ-P: $\mathbb{E}[N_s]$	FlashLinQ-Q: $\mathbb{E}[N_s]$	FlashLinQ-QP: $\mathbb{E}[N_s]$
8	0 [No Stalls]	0.8	0.3	0.5	0 [No Stalls]
4	0 [No Stalls]	1.6	1.1	0.9	0.1
2	0 [No Stalls]	3.7	1.2	1.9	0.3
0.1	1.2	7.0	3.7	4.2	1.6
0.05	3.0	15.3	8.4	9.0	4.7

probability is presented in Table II and Fig. 7. For quantitative comparison, two indices, i.e., $\mathcal{T}_s$ and $\mathcal{M}_s$, are defined as formulated in (21) and (22), respectively.

$$\mathcal{T}_s = \frac{\mathbb{E}[N_s] \text{ of FlashLinQ Variants} - \langle\mathbb{E}[N_s]\rangle}{\langle\mathbb{E}[N_s]\rangle} \quad (21)$$

$$\mathcal{M}_s = \mathbb{E}[N_s] \text{ of FlashLinQ Variants} - \langle\mathbb{E}[N_s]\rangle \quad (22)$$

where $\mathbb{E}[N_s]$ stands for the expected number of stall events and $\langle\mathbb{E}[N_s]\rangle$ denotes the $\mathbb{E}[N_s]$ of mpMWIS-QP.

As anticipated, the expected number of stall events reduces as the pre-buffering time increases. If there is no pre-buffering time, mpMWIS-QP has 13.4 stall events on average whereas FlashLinQ has 34.8. For 8 second pre-buffering time, mpMWIS-QP has no stall events; FlashLinQ-QP has 0.3; and this performance is the best among the given three FlashLinQ variants. Pure FlashLinQ shows the lowest performance. As shown in Fig. 7(c), it has 6 times more stall events when the pre-buffering time is 7 second.

We furthermore see that the performance advantage of mpMWIS-QP vs. FlashLinQ stems from a variety of factors: max-weight scheduling, incorporation of the interconnection between scheduling and streaming, and centralized control. We see that only incorporating max-weight scheduling (i.e., going from FlashLinQ to FlashLinQ-P) provides a significant advantage, while only incorporating video quality without max-weight scheduling (i.e., going from FlashLinQ to FlashlinQ-P gives slightly lower improvement. The advantage of centralized scheduling over distributed scheduling (mpMWIS-QP) is very small, which is an important insight for actual deployment.

2) Effects of the Parameter α: α stands for a quality-delay tradeoff constant as formulated in (15). If α is small, quality-awareness takes higher priority. On the other hand, larger α considers queue stability with higher priority, so that lower probability of stalls can be anticipated. The simulation results in this section investigate results when α takes on the values 0.05, 0.1, 2, 4, and 8. Notice that our considered pre-buffering time is 8 seconds, which is an optimum value for mpMWIS-QP; and the interference threshold γ is set to 5 dB. A further discussion of video quality versus stall events will be given in Section IV-B4.

The simulation results are summarized in Table III. If α is 2, 4, or 8, there are no stall events in mpMWIS-QP. Pure FlashLinQ has between 3.7 and 0.8 stall events for those values. FlashLinQ-QP will have no stalls when $\alpha = 8$. This performance is lower than the performance of mpMWIS-QP; however the performance of FlashLinQ-QP is the best among the given FlashLinQ variants.

3) Choice of the Interference Threshold γ: As discussed in Section IV-A3, the interference threshold trades off the number of active links with the rate per link that can be obtained. Thus, finding an appropriate interference threshold is important for optimizing performance. In our given network geometry, our minimum and maximum interference thresholds are 0 dB and 13 dB, respectively. With $\gamma = 0$ dB, our geometry is extremely densely connected in its corresponding conflict graph, i.e., only one D2D link will be scheduled in each unit time. On the other hand, our geometry has no edges in its corresponding conflict graph when $\gamma = 13$ dB, i.e., all D2D links will be scheduled and will generate interference all together in each unit time. We additionally performed the simulation with $\gamma = 5$ dB.

We again set the pre-buffering time to 8 second; and set $\alpha = 2$. Results with the mentioned three interference thresholds are listed in Table IV. For all algorithms, performance with $\gamma = 5$ dB is the best; and the performance with $\gamma = 0$ dB is lower than the performance with $\gamma = 13$ dB.

4) Average Quality vs. Expected Number of Stall Events: This section presents average quality values depending on the expected number of stall events for mpMWIS-QP, FlashLinQ-Q, and FlashLinQ-QP. The other two FlashLinQ variants, i.e., FlashLinQ and FlashLinQ-P, are not considered in this simulation since they statically select their quality mode. Pre-buffering time is 8 second, and the interference threshold is

TABLE IV
THE EXPECTED NUMBER OF STALL EVENTS, I.E., $\mathbb{E}[N_s]$, IN EACH VIDEO STREAMING IN EACH D2D LINK WITH VARIOUS INTERFERENCE THRESHOLDS γ

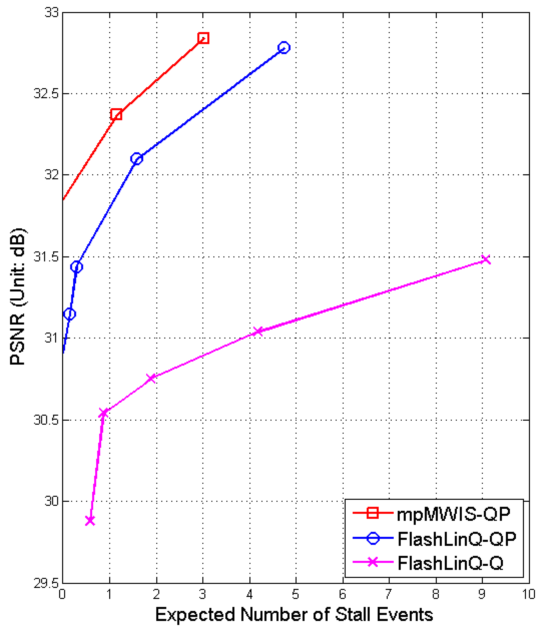
γ	mpMWIS-QP: $\mathbb{E}[N_s]$	FlashLinQ: $\mathbb{E}[N_s]$	FlashLinQ-P: $\mathbb{E}[N_s]$	FlashLinQ-Q: $\mathbb{E}[N_s]$	FlashLinQ-QP: $\mathbb{E}[N_s]$
0 dB	5.4	26.9	20.8	22.0	17.9
5 dB	0 [No Stalls]	3.7	1.2	1.9	0.3
13 dB	3.1	23.8	13.0	16.2	10.2

TABLE VI
THE EXPECTED NUMBER OF STALL EVENTS $\mathbb{E}[N_s]$ VS. AVERAGE QUALITY (AVERAGE PSNR) WHEN THE SYSTEM BANDWIDTH IS 2 MHZ

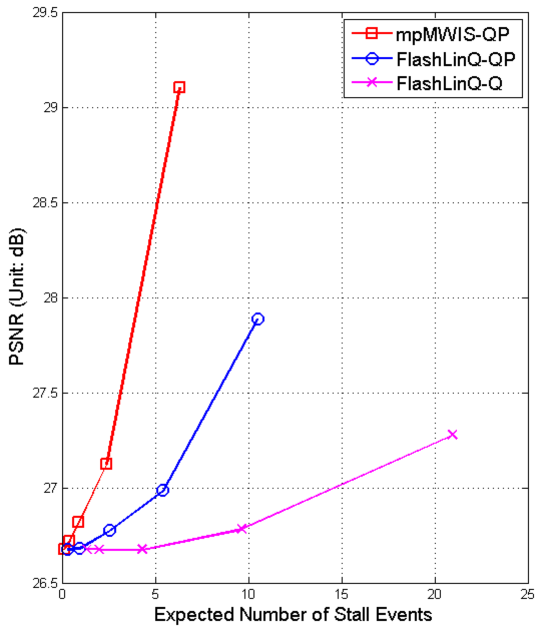
α	mpMWIS-QP	FlashLinQ-QP	FlashLinQ-Q
0.05	$\mathbb{E}[N_s]$: 3.02, Average PSNR: 32.84	$\mathbb{E}[N_s]$: 4.75, Average PSNR: 32.78	$\mathbb{E}[N_s]$: 9.07, Average PSNR: 31.48
0.1	$\mathbb{E}[N_s]$: 1.15, Average PSNR: 32.37	$\mathbb{E}[N_s]$: 1.58, Average PSNR: 32.10	$\mathbb{E}[N_s]$: 4.18, Average PSNR: 31.04
2	$\mathbb{E}[N_s]$: 0.00 [No Stalls], Average PSNR: 31.85	$\mathbb{E}[N_s]$: 0.29, Average PSNR: 31.44	$\mathbb{E}[N_s]$: 1.87, Average PSNR: 30.75
4	-	$\mathbb{E}[N_s]$: 0.14, Average PSNR: 31.15	$\mathbb{E}[N_s]$: 0.86, Average PSNR: 30.54
8	-	$\mathbb{E}[N_s]$: 0.00 [No Stalls], Average PSNR: 30.91	$\mathbb{E}[N_s]$: 0.58, Average PSNR: 29.88

TABLE VII
THE EXPECTED NUMBER OF STALL EVENTS $\mathbb{E}[N_s]$ VS. AVERAGE QUALITY (AVERAGE PSNR) WHEN THE SYSTEM BANDWIDTH IS 1 MHZ

α	mpMWIS-QP	FlashLinQ-QP	FlashLinQ-Q
0.05	$\mathbb{E}[N_s]$: 6.29, Average PSNR: 29.10	$\mathbb{E}[N_s]$: 10.50, Average PSNR: 27.89	$\mathbb{E}[N_s]$: 20.95, Average PSNR: 27.28
0.1	$\mathbb{E}[N_s]$: 2.39, Average PSNR: 27.12	$\mathbb{E}[N_s]$: 5.41, Average PSNR: 26.99	$\mathbb{E}[N_s]$: 9.65, Average PSNR: 26.78
2	$\mathbb{E}[N_s]$: 0.89, Average PSNR: 26.82	$\mathbb{E}[N_s]$: 2.55, Average PSNR: 26.78	$\mathbb{E}[N_s]$: 4.32, Average PSNR: 26.68
4	$\mathbb{E}[N_s]$: 0.39, Average PSNR: 26.72	$\mathbb{E}[N_s]$: 0.95, Average PSNR: 26.68	$\mathbb{E}[N_s]$: 2.00, Average PSNR: 26.67
8	$\mathbb{E}[N_s]$: 0.16, Average PSNR: 26.68	$\mathbb{E}[N_s]$: 0.32, Average PSNR: 26.68	$\mathbb{E}[N_s]$: 1.32, Average PSNR: 26.67



(a) System Bandwidth: 2 MHz (The data in this figure is listed in Table VI)



(b) System Bandwidth: 1 MHz (The data in this figure is listed in Table VII)

Fig. 8. Average Quality vs. Expected Number of Stall Events

5 dB. To numerically represent video quality, PSNR is used; the minimum and maximum PSNR values in each video trace are listed in Table V.

We simulate the algorithms with $\alpha = \{0.05, 0.1, 2, 4, 8\}$, and compute the expected numbers of stall events and the average PSNR values. Results are shown for a system bandwidth of 2 MHz in Table VI. In mpMWIS-QP, there are no stall events if $\alpha \geq 2$. However, higher α leads to the degradation

TABLE V
PSNR TABLE OF GIVEN FOUR VIDEO TRACES

	minimum PSNR	Maximum PSNR
Video Trace 1	29.4835 dB	37.8063 dB
Video Trace 2	25.7136 dB	36.2584 dB
Video Trace 3	24.3273 dB	35.0470 dB
Video Trace 4	28.8283 dB	37.1691 dB

of average PSNR to guarantee more stability on the D2D transmitter queue. If there are no stall events, there is no need to improve the stability of the D2D transmitter queue. Thus, simulation results of mpMWIS-QP where $\alpha = 4$ and $\alpha = 8$ are not shown. In addition, FlashLinQ-QP has no stall events when $\alpha = 8$.

Fig. 8(a) shows that our mpMWIS-QP provides the highest PSNR for a given stall probability; and the performance of FlashLinQ-QP is approximately 0.4 dB lower than the performance of mpMWIS-QP. However, FlashLinQ-Q shows around 1.6 dB lower PSNR compared to the PSNR of mpMWIS-QP.

Results for a system bandwidth of 1 MHz are in Table VII. Due to the lower bandwidth, all the three evaluated algorithms have stall events. Similar to the cases of 2 MHz system bandwidth, higher α leads to the degradation of average PSNR to guarantee more stability on the D2D transmitter queue. Fig. 8(b) shows that mpMWIS-QP again provides the highest PSNR; and the PSNR of FlashLinQ-QP is approximately 1.5 dB lower than the performance of mpMWIS-QP when the expected number of stall events is near 5. However, FlashLinQ-Q shows around 2.3 dB lower PSNR compared to the PSNR of mpMWIS-QP. According to Table VII, the lowest PSNR is near 26.7, which is close to the PSNR of the lowest-quality mode available.

As observed in Fig. 8, a higher expected number of stall events is associated with a higher PSNR (i.e., video quality). For guaranteeing more video quality, lower α is used for putting more weights on PSNR and lower weights on queue stability. Therefore, there are more possibilities to increase the queue backlog sizes at D2D transmitters, i.e., this leads to higher expected number of stall events at D2D receivers.

V. CONCLUDING REMARKS

This paper proposed centralized or distributed quality-aware streaming and scheduling algorithms which can be used for device-to-device video delivery applications. In terms of scheduling, we have considered both a centralized and a distributed approach. For centralized scheduling, a message-passing based algorithm is used to obtain the solutions from a maximum independent set problem formulation. For distributed scheduling, we improved the FlashLinQ D2D scheduler by introducing a max-weight priority across the links. In terms of streaming, a quality-aware stochastic chunk selection algorithm is introduced that works based on the queue backlog sizes in each D2D transmitter queue. The stochastic algorithm in the streaming part controls the quality of each video chunk to maximize the qualities of streamed video subject to queue rate stability. We can draw several important conclusions from the simulations: (i) it is essential to use a transmission scheme that accounts for the inter-relationship between scheduling and quality selection, (ii) a good distributed scheme performs only marginally worse than our centralized scheme, and (iii) we can trade off average video quality with probability of stalls. These results give important insight in the deployment of D2D-based video streaming.

REFERENCES

[1] N. Golrezaei, A.F. Molisch, A.G. Dimakis, and G. Caire, "Femtocaching and Device-to-Device Collaboration: A New Architecture for Wireless Video Distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142 - 149, April 2013.

[2] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM SIGCOMM Computer Communications Review*, vol. 33, no. 2, pp. 93 - 102, April 2003.

[3] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. IEEE INFOCOM*, 2005.

[4] Y. S. de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louedec, "Efficient HTTP-based streaming using Scalable Video Coding," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 329–342, April 2012.

[5] T. Schierl, Y. S. de la Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based Media Delivery using SVC with RTP and HTTP streaming," *Multimedia Tools and Applications*, vol. 55, no. 2, pp. 227-246, 2011.

[6] Y. S. de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Loudec, "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding," in *Proc. ACM Multimedia Systems Conference (MMSys)*, 2011.

[7] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "FlashLinQ: A Synchronous Distributed Scheduler for Peer-to-Peer Ad Hoc Networks," in *Proceedings of Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, October 2010.

[8] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "FlashLinQ: A Synchronous Distributed Scheduler for Peer-to-Peer Ad Hoc Networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 4, pp. 1215 - 1228, August 2013.

[9] F. Baccelli, J. Li, T. Richardson, S. Shakkottai, S. Subramanian, and X. Wu, "On Optimizing CSMA for Wide Area Ad-hoc Networks," in *Proceedings of IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Princeton, NJ, USA, May 2011.

[10] F. Baccelli, J. Li, T. Richardson, S. Shakkottai, S. Subramanian, and X. Wu, "On Optimizing CSMA for Wide Area Ad-hoc Networks," *Queueing Systems*, vol. 72, pp. 31 - 68, 2012.

[11] Cisco, "Cisco Visual Networking Index: Forecast and Methodology 2012–2017," *Cisco White Paper*, 2013.

[12] K. Doppler, M. Rinne, C. Wijting, C.B. Ribeiro, and K. Hugl, "Device-to-Device Communication as an Underlay to LTE-Advanced Networks," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42 - 49, December 2009.

[13] L. Lei, Z. Zhong, C. Lin, and X. Shen, "Operator Controlled Device-to-Device Communications in LTE-Advanced Networks," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 96 - 104, June 2012.

[14] A.F. Molisch, *Wireless Communications*, 2nd Edition, IEEE-Wiley, February 2011.

[15] S. Sanghavi, D. Shah, and A.S. Willsky, "Message Passing for Max-weight Independent Set," in *Proceedings of Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2007.

[16] S. Sanghavi, D. Shah, and A.S. Willsky, "Message Passing for Maximum Weight Independent Set," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 4822 - 4834, October 2009.

[17] D. Bethanabhotla, G. Caire, and M.J. Neely, "Joint Transmission Scheduling and Congestion Control for Adaptive Streaming in Wireless Device-to-Device Networks" in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, November 2012.

[18] D. Bethanabhotla, G. Caire, and M.J. Neely, "Joint Transmission Scheduling and Congestion Control for Adaptive Video Streaming in Small-Cell Networks," *IEEE Transactions on Communications* (Under Revision). Available on ArXiv: http://arxiv.org/abs/1304.8083

[19] M.J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, Morgan & Claypool, 2010.

[20] M. Ji, G. Caire, and A.F. Molisch, "Wireless Device-to-Device Caching Networks: Basic Principles and System Performance," *IEEE Journal on Selected Areas in Communications* (Under Revision). Available on ArXiv: http://arxiv.org/abs/1305.5216

[21] A. Abdel Khalek, C. Caramanis, and R. W. Heath, Jr., "A Cross-layer Design for Perceptual Optimization of H.264/SVC with Unequal Error Protection," *IEEE Journal On Selected Areas in Communications*, vol. 30, no. 7, pp. 1157–1171, August 2012.

[22] S. Singh, J. G. Andrews and G. de Veciana, "Interference Shaping for Improved Quality of Experience for Real-Time Video Streaming," *IEEE Journal On Selected Areas in Communications*, vol 30., no. 7, pp. 1259–1269, August 2012.

[23] A. Abdel Khalek, C. Caramanis, and R. W. Heath, Jr., "Video-aware MIMO Precoding with Packet Prioritization and Unequal Modulation," in *Proc. European Signal Processing Conference (EUSIPCO'12)*, Bucharest, Romania, August 2012.

[24] S. Singh, O. Oyman, A. Papathanassiou, D. Chatterjee, and J. G. Andrews, "Video Capacity and QoE Enhancements over LTE," in *Proc. IEEE International Conference on Communications Workshop on Realizing Advanced Video Optimized Wireless Networks (ICC ViOpt'12)*, Ottawa, Canada, June 2012.

[25] V. Joseph and G. de Veciana, "Jointly Optimizing Multi-user Rate Adaptation for Video Transport over Wireless Systems: Mean-Fairness-Variability Tradeoffs," in *Proc. IEEE INFOCOM*, pp. 567–575, Orlando, FL, USA, March 2012.

[26] A. A. Khalek, C. Caramanis, and R. W. Heath Jr., "Joint Source-Channel Adaptation for Perceptually Optimized Scalable Video Transmission," in *Proc. IEEE GLOBECOM*, Houston, TX, USA, December 2011.

[27] Z. Lu and G. de Veciana, "Opportunistic Transport for Stored Video Delivery over Wireless Networks: Optimal Anticipative and Causal Approximations," in *Proc. 49th Allerton Conference on Communication, Control and Computing, Chicago*, IL, October 2011.

[28] C. Chen, R. W. Heath Jr., A. C. Bovik and G. de Veciana, "Adaptive Policies for Real-Time Video Transmission: a Markov Decision Process Framework," in *Proc. IEEE International Conference on Image Processing (ICIP'11)*, September 11-14, 2011, Brussels, Belgium.

[29] L. Toni, P. C. Cosman, and L. B. Milstein, "Channel Coding Optimization Based on Slice Visibility for Transmission of Compressed Video over OFDM Channels," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 7, pp. 1172–1183, August 2012.

[30] L. Toni, P. C. Cosman, and L. B. Milstein, "Subcarrier mapping based on slice visibility for video transmission over OFDM channels," in *Proc. Asilomar Conference on Signals, Systems and Computers (Asilomar'12)*, Pacific Grove, CA, USA, November 2012.

[31] H. Ahlehagh and S. Dey, "Hierarchical video caching in wireless cloud: approaches and algorithms," in *Proc. IEEE International Conference on Communications Workshop on Realizing Advanced Video Optimized Wireless Networks (ICC ViOpt'12)*, Ottawa, Canada, June 2012.

[32] H. Ahlehagh and S. Dey, "Video Caching in Radio Access Network: Impact on Delay and Capacity," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC'12)*, Paris, France, April 2012.

[33] S. Dey, "Cloud Mobile Media: Opportunities, Challenges, and Directions," in *Proc. IEEE International Conference on Computing, Networking and Communications (ICNC'12)*, Maui, HW, USA, Januaray 2012.

[34] D. Wang, P. C. Cosman, and L. B. Milstein, "Cross layer resource allocation design for uplink video OFDMA wireless systems," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'11)*, Houston, TX, USA, December 2011.

[35] L. Toni, P. C. Cosman, and L. Milstein, "Unequal error protection based on slice visibility for transmission of compressed video over OFDM channels," in *Proc. IEEE International Conference on Multimedia and Expo (ICME'12), Workshop on Acoustics and Video Coding and Communication (AVCC)*, Barcelona, Spain, July 2011.

[36] N. Golrezai, A. F. Dimakis, and A. F. Molisch, "Device-to-Device Collaboration through Distributed Storage," in *Proc. IEEE Globecom*, 2012.

[37] K. Shanmugam and G. Caire, "Wireless Downloading Delay under Proportional Fair Scheduling with Coupled Service and Requests: An Approximated Analysis," in *Proc. IEEE International Symposium on International Theory (ISIT'12)*, Boston, MA, USA, July 2012.

[38] N. Golrezaei, A.G. Dimakis, and A.F. Molisch, "Wireless Device-to-Device Communications with Distributed Caching," in *Proc. IEEE International Symposium on International Theory (ISIT'12)*. Boston, MA, USA, July 2012.

[39] N. Golrezai, A. F. Molisch, and A. Dimakis, "Base-Station Assisted Device-to-Device Communications for High-Throughput Wireless Video Networks," in *Proc. IEEE ICC Workshop on Video-Aware Wireless Networks 2012 (ICC ViOpt'12)*. Ottawa, Canada, June 2012.

[40] N. Golrezai, K. Shanmugam, A. Dimakis, A. F. Molisch, G. Caire, "Wireless Video Content Delivery through Coded Distributed Caching," in *Proc. IEEE International Conference on Communications (ICC'12)*, Ottawa, Canada, June 2012.

[41] N. Golrezai, K. Shanmugam, A. Dimakis, A. F. Molisch, G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers," in *Proc. IEEE International Conference on Computer Communications (INFOCOM'12)*, Orlando, FL, USA, March 2012.

[42] N. Golrezaei, A.G. Dimakis, A.F. Molisch, and G. Caire, "Wireless video content delivery through distributed caching and peer-to-peer gossiping," in *Proc. IEEE 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'11)*, pp.1177-1180, Pacific Grove, CA, USA, November 2011.

[43] M.Z. Win, P.C. Pinto, and L.A. Shepp, "A Mathematical Theory of Network Interference and Its Applications," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 205 - 230, February 2009.

[44] H. Mansour, V. Krishnamurthy, and P. Nasiopoulos, "Channel Aware Multiuser Scalable Video Streaming Over Lossy Under-Provisioned Channels: Modeling and Analysis," *IEEE Trasactions on Multimedia*, vol. 10, no. 7, November 2008.

[45] X. Wang, M. Chen, T.T. Kwon, L.T. Yang, and V.C.M. Leung, "AMES-Cloud: A Framework of Adaptive Mobile Video Streaming and Efficient Social Video Sharing in the Clouds," *IEEE Transactions on Multimedia*, Vol. 15, no. 14, pp. 811 - 820, 2013.

[46] X. Wang, T.T. Kwon, Y. Choi, H. Wang, and J. Liu, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 72 - 79, 2013.

[47] L. de Cicco and S. Mascolo, "An Adaptive Video Streaming Control System: Modeling, Validation, and Performance Evaluation," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 526 - 539, 2014.

[48] J. Chakareski, J.G. Apostolopoulos, S. Wee, W.-T. Tan, and B. Girod, "Rate-distortion hint tracks for adaptive video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1257 - 1269, 2005.

[49] S. Tavakoli, J. Gutierrez, and N. Garcia, "Subjective Quality Study of Adaptive Streaming of Monoscopic and Stereoscopic Video," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 684 - 692, 2014.

[50] T.C. Thang, H.T. Le, H.X. Nguyen, A.T. Pham, J.W. Kang, and Y.M. Ro, "Adaptive video streaming over HTTP with dynamic resource estimation," *Journal of Communications and Networks*, vol. 15, no. 6, pp. 635 - 644, 2013.

[51] S. Winkler and P. Mohandas, "The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics," *IEEE Transactions on Broadcasting*, , vol. 54, no. 3, 2008.

[52] N. Staelens, D. Deschrijver, E. Vladislavleva, B. Vermeulen, T. Dhaene, and P. Demeester, "Constructing a No-Reference H.264/AVC Bitstream-Based Video Quality Metric Using Genetic Programming-Based Symbolic Regression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 8, pp. 1322 - 1333, 2013.

[53] J. Kim, Y. Tian, S. Mangold, and A.F. Molisch, "Joint Scalable Coding and Routing for 60 GHz Real-Time Live HD Video Streaming Applications," *IEEE Transactions on Broadcasting*, vol. 59, no. 3, pp. 500 - 512, September 2013.