Network Coding Based Caching for Near Real-Time Streaming Media

Csaba Simon, Markosz Maliosz

Abstract— **During crowded events streaming services generate high demands in the wireless access networks. In this paper we present a solution to offload the access network in case of such a streaming service. We detail the streaming service itself, and our offload solution based on local caching and network coding. We introduce a model that allows us to analyze our proposal, we implement it in a simulation environment and assess it. Finally we discuss the consequences of several design decisions we made during our work.**

***Index Terms*—multimedia applications, network communication, network coding, caching**

I. INTRODUCTION

The Internet traffic is dominated by streaming multimedia content as users demand higher quality video and ubiquitously available services. With the advent of high performance smart handheld devices the users expect that their usual services received on their desktops are available on these smart devices, too. Thus users can access advanced services from new places where they start to use their devices on regular basis. On turn, these new situations generate new demands: once the users get used to the new scenario, they start to require new, adapted services.

A typical scenario is a crowded event, where even a few years ago users could not use their mobile devices due to network congestion. E.g., it was common that during New Year's Eve calls were blocked and only SMS-es went through the overloaded networks. Similarly, sporting events at remote areas required a careful design and temporary increase in mobile access capacity to serve the increased demand. This motivated us to offload access networks during crowded events for a new streaming service, specific to this environment.

Users attend crowded events for the live experience, which combines the feeling of "being there" with the potential of rich social interactions among fellow users with similar interests. Nevertheless, until recently the participation at such events forced the attendants to stop following the online (e.g., live commentaries, additional info) and broadcasted (e.g., TV) content. The solution that offers both experiences, live attendance and online information stream, comes with the introduction of the so called "second screen".

Manuscript submitted September 29, 2014, revised February 22, 2015.

The authors are with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary, 1117 Budapest, Magyar Tudosok krt. 2. (e-mail: {simon|maliosz}@tmit.bme.hu)

Second screen originally refers to the use of an online device (e.g., smartphone, iPad) that doubles the screen of a device offering "linear" program (e.g., TV, projector). We extend the meaning of this term, calling second screen any online device that offers additional content associated with a live event, attended by the user of the device. Current access networks are hard pressed to provide the required QoS, because attendees of live events continue using their smart devices as second screens (to consume more and more multimedia content). In this environment, shortage of available capacity seems to perpetuate at least until the mid-2020s, when 5G technologies will mature. The focus of the operators is on assuring the basic service, not to mention any new service with additional bandwidth demand. Therefore offloading the wireless access currently is very important for the operators, and it will be so for the coming decade.

The data to be distributed in such an environment is not only the real-time, live multimedia stream, but also extra, add-on content, which has less strict delay constraints, and is related to recent events (e.g., replays, statistical analysis of the game, etc.). Still, their importance is higher soon after the original event happened (e.g., a goal right after it was scored), that is why we call them near real-time events. We propose a novel streaming service specific to this environment that can be offered on top of classical streaming media services, consisting of replayed live scenes. At the core of our solution to offload the access network delivering this service is the distributed local caching of the data, made reliable and versatile by the introduction of network coding techniques. To best of our knowledge, network coding was not proposed before to support such caching solutions (also see section III-C). The motivation behind such novel add-on services are not only recognized by recent research projects [1], but also attract major players from the streaming live event distribution industry [2].

In the next section we present related work that we relied on in our research. Then we present several scenario variants for our proposal and introduce the novel near real-time data delivering service that can be offered on top of classical streaming media services. In section IV we present a model that will allow us to analyze its behavior, and we evaluate it in section V. Finally we conclude our paper.

II. Communication in the local wireless domain

The support of new service types in such a challenging environment as crowded spaces needs a complex approach, which relies on results from several research areas of communications. In this section we briefly introduce the main aspects that influence the most our proposal and are referenced in later Sections during the definition of the model of our proposal. Specifically, the wireless technologies define the limitations of direct inter-node communication, while network coding ensures the flexibility and robustness of local data distribution.

A. Offloading the Local Wireless Access

Smartphones have several wireless interfaces that can be used to achieve direct communication. The natural choice is WiFi, with its adhoc variant. WiFi adhoc was very popular among researchers in the laptop era. Lots of mobile ad hoc protocols (MANET) were prototyped and investigated using such connections. Unfortunately this technology is not supported anymore by the vendors, although some Android smartphone models still can be tweaked to work in adhoc mode. The main advantage of the ad hoc mode is that it is very flexible.

Officially the replacement technology of the WiFi ad hoc mode is the WiFi Direct [3]. Nevertheless, the latter comes with some limitations, but these do not affect our scenarios. Both technologies have a different problem, too: the interfaces can work only in one WiFi mode only. Nevertheless, for modelling purposes we can use the WiFi Direct interface, as most of the community is familiar with this technology. Note that in real life deployments the local networking connection might be one of the UMTS/LTE technologies (e.g., using femtocells). Then the WiFi interface of the smartphone is available to support our service.

A different option might be the new variants of Bluetooth. The advantage is that typically this interface is not used, but it has lower capacity and it is harder to set up a link.

Finally we have to mention the promising new LTE variant, the LTE Direct [4] (or LTE D2D), which offers direct connectivity in a non-WiFi band, but it will cost more, as operates in a licensed band.

B. Network Coding

Network coding is a technique that, in contrast to channel coding, “allows and encourages the mixing of data at intermediate network nodes”, instead of just encoding messages in a redundant way, allowing the network to have a maximum flow of information achieving a larger throughput [5]. With network coding, information transmitted from a source can be received by the receivers, but it can also be inferred or decoded. Intermediate nodes are still able to forward information but if it is the case, the node can combine different received streams of information into just one and transmit it to its outgoing nodes.

The fixed version of network coding uses simpler coding techniques (e.g., bitwise XOR-ing the packets of the involved stream(s)), making the flow always encodable and decodable, however this advantage comes with the downside of having to define the structure and the number of participants of the entire network previously. There is an alternative that follows a random behavior [6], where nodes assign coefficients to each packet randomly and according to the finite field used, there is a probability of these coefficients being decodable. Using random network coding all nodes are independent and randomized, without the need of any knowledge of the rest of the network. Intermediate nodes build a linear combination of incoming messages that then transmit on each on their outgoing links. Differently than the fixed method (e.g., bitwise XOR-ing), this combination uses independently and randomly chosen coefficients over a finite field. By allowing this kind of local encoding under a sufficiently large Galois field (i.e. finite field), the received coded blocks are decodable with a very high probability at the sink peers, on the order of the inverse of the size of the finite field [7].

The first practical wireless network coding scheme designed to deal with inter-flow traffic is also based on Random Linear Network Coding (RLNC) [8]. It exploits the shared nature of wireless medium and combines available data chunks with ones overheard from neighbors to restore the original information. Although it significantly improves network throughput, it is limited to situations where multiple streams cross the same network segment. A different approach uses RLNC to encode data within the same flow [9]. This intra-flow network coding improves the performance of the network over wireless links.

We divide the stream in generations. Only packets from a generation are linearly combined (encoded), thus at the receiving end enough linearly independent packets from a generation should be collected to be able to decode the content. For the streaming service it acts as a time window, because all packets must arrive before the playout of that particular encoded sequence can start. Any encoded packet belonging to a generation is useful (until we receive enough of them), the packets are not ordered, which simplifies the timing at the receiver side.

Network Coding was mostly proposed to be used in information distribution [10], multicast [11] and data averaging [12], which assures the usage in distributed sensor network data collection as well.

III. Caching of streaming media at the end nodes

A. Crowded Event Scenarios

We have identified three different scenarios for crowded events. All three scenarios offer a solid business model to build on and attract dedicated users who have the motivation to be actively involved in the content consumption process. E.g., they are interested in the details of the performers, want to know previous stories about the protagonists, etc. This offers a good audience for our proposed service.

The first scenario is the open air city festival, where attendees have access to multiple scenes and several selling and catering locations within a geographically limited area. Note that such events might become very congested, especially around sites of interest. We will refer to this scenario as the “festival” one.

A similarly crowded scenario is offered by the stadiums (“stadium” scenario). The main difference is that in the stadiums the participants are bound to their seats and typically these events last for only a few hours. Therefore the people are not moving as much as in the previous scenario.

Combination of the previous two scenarios are the open air sporting events (bicycle tour, triathlon, etc). In such scenarios the attendance is scattered along the track, but usually they form small groups of people at interesting or spectacular portions of the track. Due to the largest such cycling event, (Tour de France), we will refer to this scenario as the "Tour" scenario.

Note that we expect different user behaviors and topologies in each of these scenarios, as detailed later in sub-section V-C.

B. Streaming Media Services

The traffic volume of streaming media had exceeded that of any other traffic type, including peer-to-peer or web access and researchers tried to reduce its bandwidth demand by various methods, including caching. There is a vast available literature in this field. In [17] we highlighted the most important ones. The reader interested in further details of streaming video caching is directed to a thorough overview of this field [14].

Our proposal is an additional service to extend the original streaming service. Currently replays are broadcasted within the original content, there is no possibility to watch them on-demand. In this paper we focus only on the service offering replays close to the moment in time when it originally happened (e.g., 100 minutes), which happens in a near real time fashion (in the worst case). In lots of cases the users want to re-watch the missed content, too. But in such cases only some special moments are of high interest, as the user then wants to resume and follow the original live content. Therefore recording the whole stream and playing it with a constant lag is not an option. In this case it is better to cache such short sequences from the live stream and make it available for instant replay. This caching service is detailed in the next sub-section.

C. Caching of the Replayed Content

In this section we introduce our solution for near real-time media streaming that also offloads the wireless access.

The load in the wireless access is decreased by the use of network coding and stretching the lifetime of the network encoded packets somewhere in the network distributed in end devices or well-placed points in the distribution/access domain. The cache distribution is implemented primarily on the user's devices.

This scenario is depicted in Fig. 1. The original live streaming media is distributed by the AP_i Access Points. The nodes receiving the data encode it and cache it locally (dark gray stars –S_1, …). Any time a node (light gray star) wants a replay, they will have the data chunks readily available in their local mesh network.

As already mentioned, the nodes should organize themselves to find the caches, this can be done using techniques used by peer-to-peer applications [13]. The advantage is that the neighbor list maintenance can be supported by the APs with limited extra costs in terms of bandwidth usage (e.g., neighbouring APs can exchange the list of connected devices that act as caches and broadcast that list periodically for all).

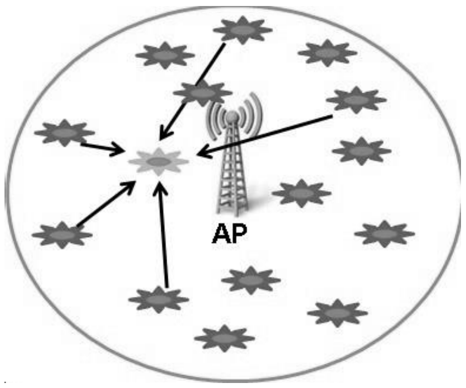

Fig. 1. Media streaming with caching

Also note that the direct node-to-node control traffic is not affecting directly the AP (for details also see sub-section V-A). In our scenario each node is attached to one AP and additionally it can communicate with several fellow nodes from its. We do not analyze the technological details, but two possible alternatives for such local communications to be largely adopted in the coming years are the WiFi Direct [3] and LTE Direct [4].

In our solution we use additional wireless sessions to receive the data from the caches, which run in parallel with the AP-to-node sessions. This might increase the number of collisions; some of these collisions are hidden by lower layers, others will result in packet losses. At higher layers (networking and above) these packet losses are perceived as a lower bandwidth. We measured the effect of this parallel communication, as described in section 5.1 and we used the results in our simulations. Thus during the evaluation of our model we take into account the impact of the addition of cache-to-node sessions through the modified data transfer rates.

IV. MODELING THE CACHE BASED STREAMING SERVICE

A. Network Coding Based Caching

From our model's perspective the important thing is to have a direct node-to-node communication in parallel with the AP-to-node connection. However we also have to take into account that a given node cannot communicate directly with every neighbor, the degree of connectivity is upper bounded. Currently the most used file distribution is BitTorrent, and the most used p2p streaming application (SopCast) is based on the same principles [21]. In BitTorrent only 5 slots are available for uploading data. The goal of this limitation is to protect the uploaders (i.e., the cache) from overload.

We consider that the potential coverage areas of the APs will be much larger than the area they accept connection requests from. Therefore, if there is a node somewhere at the border of two (or more) neighbouring AP coverage areas, then it can be identified by the respective APs. Based on that they can provide this node with the necessary information about cached content within its reach (but it will not "spam" this node with the list of far distant caches). The request statistics for a given replay or generally, for this type of service and similar statistics specific to streaming services can be computed in the background

[15][16]. Note that caching nodes can also report the read statistics to their APs. The details of the practical implementation of such services (e.g., the description of a protocol implementing it) are out of the scope of this paper. In our model we consider that these data are available, and it can be obtained from the AP a given node is connected to.

In our model the nodes that actually execute the network coding task are the same ones that work as caches. Due to the nature of RLNC we do not have to previously configure them. Therefore in our model we do not have to dedicate special attention to network coding aspects, nor to the selection process of the coding nodes. During the construction of our model we just had to focus on the caching aspects, the selection of the network coding nodes implicitly resulted from it.

Time dependencies are also not addressed directly, because the focus is on collecting the packets of a generation. Once collected enough packets, the content is readily available, implicitly meaning that the timing of content distribution conforms to the requirement of the service. If this requires slightly more time, then it can be considered as a slightly longer buffering time, which will not affect the (near real-time) service, because it has less strict delay requirements then a real-time one.

In the simplest model we search for the number of caching nodes to serve those nodes that are attached to the same AP, but we do not limit the cache size. Note that this solution does not allow that a node attached to AP_i to request data from a caching node attached to AP_j. In order to deduce the minimum number of overall caching points in the network, we can formulate an Integer Linear Program (ILP), as presented in the following sub-section. Then, in section IV-C we refine our model, aiming to minimize the size of the cache, at the same time letting more nodes to step in as caches, and we formulate an ILP for this case, too.

B. *Minimizing the Number of Caching Nodes*

We have a set of $\{AP_i\}$ Access Points, but for our model we should rather focus on the nodes. The nodes $\{v_1, ..., v_N\}$ and the direct links between them $\{ e_{ij} \}$ can be considered the vertices and the edges of a graph G. We define the connectivity matrix $\{a_{ij}\}$, where a matrix element is 1 if there is a direct connection between nodes i and j, 0 otherwise. Note that this can be obtained by recoding the original content locally with pseudo random coefficients.

Our goal is to determine the minimal number of caching nodes, a subset of G. c_i is the total capacity of the direct link from node i (we consider that the total incoming and outgoing capacities are equal). In order to decode the original content, we need at least a full generation of encoded packets (the size of a generation is noted with g_w). We introduce two binary variables; x_{ij} denotes a direct link between nodes v_i and v_j, $x_{ij} = 1$ if v_i sends the cached content to v_j, $x_{ij} = 0$ otherwise. Similarly, $u_i = 1$, if node i is a cache, $u_i = 0$ otherwise.

Our optimization problem is:

$$\text{minimize} \sum u_i \quad (1)$$

subject to

$$\sum_j x_{ij} \le c_i \quad \forall i \quad (2)$$

$$\sum_i x_{ij} \le c_j \quad \forall j \quad (3)$$

$$\sum_i x_{ij} \ge g_w \ \forall i, \forall j \quad (4)$$

$$x_{ij} \le a_{ij} \quad \forall i \, \forall j \quad (5)$$

$$x_{ij} \le u_i \quad \forall j \quad (6)$$

$$u_i, x_{ij} \in \{0,1\} \quad \forall i, \forall j \quad (7)\text{-}(8)$$

Equations (2) and (3) ensure that the caches and the regular nodes cannot exceed their total link capacities. Equation (4) ensures that all demand is served. Equation (5) assures that x_{ij} can be greater than 0 only if there is a direct physical connection between the nodes and eq. (6) that x_{ij} is greater than 0 only if the connection is originating from a cache.

The generic form of this kind of optimization problem is known as the geometric set cover problem, and has been continuously researched in the last decades. Based on the earlier research results it is hard to solve [18][19], and the most versions of the problem are still considered to be NP-hard [20]. Also, [25] states that 0-1 Integer Linear Programming (ILP) is NP-complete.

C. *Minimizing the Cache Size*

In the previous two sub-sections we analyzed the ways to minimize the number of caching nodes, but we did not restrict the size of the cache. In this subsection we try to minimize the size of the cache, but we do not restrict the number of caching nodes. Note that every node is a potential cache node, because every node plays the streaming content (we exclude those nodes that do not follow the video stream). Let us keep the same notations we introduced earlier in this section. We note the number of chunks stored at node i with k_i, and the number of actually downloaded encoded chunks from i to j with h_{ij} .

Now the objective is to

$$\text{minimize} \sum k_i \quad (9)$$

subject to

$$\sum_j x_{ij} \le c_i \quad \forall i \quad (10)$$

$$\sum_i x_{ij} \le c_j \quad \forall j \quad (11)$$

$$x_{ij} \le a_{ij} \quad \forall i \, \forall j \quad (12)$$

$$\sum_i h_{ij} \ge g_w \quad \forall j \quad (13)$$

$$0 =< h_{ij} <= x_{ij} M \quad \forall i, \forall j \quad (14)$$

$$k_i >= h_{ij} >= 0 \quad \forall i, \forall j \quad (15)$$

$$x_{ij} \in \{0,1\} \quad \forall i, \forall j \tag{16}$$

$$k_i, h_{ij} \in \mathrm{N} \quad \forall i, \forall j \tag{17}$$

We have (10), (11), (12) and (13), since conditions (1), (2), (5) and (8) are valid in this scenario, too. Eq. (13) shows the actually downloaded number of chunks for one node, v_j should at least be equal with g_w. If we select M sufficiently large (e.g., $M \geq g_w$), then eq. (14) states that downloads are possible only from selected caches. Eq. (15) says that the number of downloaded packets from a given cache is upper bounded by the content available at that cache.

V. EVALUATION OF THE MODEL

In this section we evaluate the proposed model and discuss the particularities of the proposed scenarios.

A. *Offloading the Wireless Access*

We evaluated the effect of the additional service on the original AP capacity. In this experiment we considered that the AP is using WiFi, while the nodes for their direct communication (i.e., cache access) use such technologies that uses the same frequency band (e.g., WiFi Direct or Bluetooth). We measured the impact of WiFi Direct on WiFi, when 5 to 20 streaming devices are connected to the AP and we had pairs of nodes testing WiFi Direct connections with iPerf. We found that if the two technologies run on different channels, the total capacity is relatively less affected when larger number of direct node-to-node pairs communicate. We used this value in our simulator to represent the effect of direct node-to-node communication on overall network load (e.g., packet losses due to collisions). Note that the combination of LTE Direct with WiFi APs yields better results in the favor of the distributed caching solutions. The co-existence of LTE-based streaming and LTE Direct was not assessed in this paper, as we focus on local wireless technologies. In this sub-section we compare three scenarios. The original one is when the replays are sent by the AP, which increases the load linearly with the number of requests.

The alternative solution is when caching is implemented without network coding. In this case the packets are sent directly from node to node, without directly consuming AP bandwidth. In this case there is a large control traffic overhead required to organize the download of the content. This case resembles the pure peer-to-peer streaming solutions, where control overhead in terms of number of packets is reported to vary between 5% and 20%, with the larger values for the leading streaming peer-to-peer application, SopCast [21][22]. The size of the control packets is one order of magnitude lower than the size of data packets, but p2p streaming applications contact many other peers, not only those they are downloading from. Note that based on our measurements, this 20% packet overhead is a conservative value, because at the beginning of downloads (starting to watch a replay) or when a seeder has to be replaced (churn event), the control traffic exceeds 2/3 of the total packet counts. As a consequence, we used a 15% overhead in terms of bandwidth (on the direct node to node links). In [21] they calculated with minimum 10% signalling overhead, SopCast having larger overheads.

The third case is the proposed network coding based caching. For RLNC based distribution in [23] the authors calculated with 5% overhead, but our scenario is simpler, because the infrastructure takes over some parts of the discovery and maintenance job and the peers are within direct layer 2 contact. Therefore we calculated with a minimal overhead (we used a 3% value in our simulations), as the requester also does not have to deal with the uniqueness of the segments.

The result of the evaluation is shown in Fig. 2, with linear trends fitted on all three data series. We simulated 50 requesting nodes the most, because a WiFi AP will not serve more than 100 streams, and out of this maximum number of connected nodes only a fraction of them will access the cache at the same time. It can be seen that both distributed caching solutions significantly offload the network, and scales well with the growing replay demands. Also we can see that the proposed network coding based caching solution outperforms both alternatives.

B. *Optimization of the Caches*

We have built a simulator to test the scenarios given in section IV in the different network conditions. We applied the graph libraries of the lemon tool [26] and the glpk and gurobi public ILP solver tools [27][28]. We generated the connectivity matrix considering that the network nodes were uniformly distributed. We have generated several different networks and averaged the solutions to get the presented results. We limited the upload capacity to 5 units.

The size of the encoded chunks should be less than the size of an UDP packet, somewhere around 1kB. The number of data

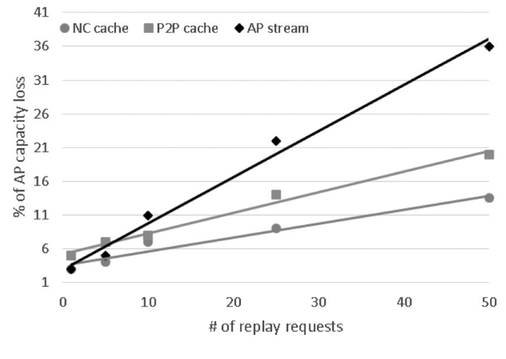

Fig. 2. Evaluation of AP offloading efficiency

chunks encoding the same generation (g_w) should be of orders of tens, eventually few hundreds.

These results give the theoretical bounds as a result. However, in practically feasible implementations, due to the distributed implementation, we can just approximate this result at the best. We have proposed heuristic algorithms for both optimization problems in [17], but in this paper we will analyse the ILP models only.

Because of the complexity of the problem, in order to allow the ILP solver to find the optimum, we used max. 100 nodes (N)

in the system, such as the diameter of the network is 5 hops in a dense scenario. As explained earlier, the number of nodes connected to a given AP should be less than the maximum capacity, in order to maintain the required QoS, so the above parameters would mean the deployment of 10 APs. We also investigated a sparse scenario, where nodes are farther away, and the maximum number of APs is 15. The number of nodes demanding the service was set to 20% of N, a worst case upper bound. The g_w parameter is also downsized to allow the solver to complete, and we set it for $g_w=10$, meaning that a node can not get all its chunks at once from a single cache.

TABLE I
OPTIMAL CACHE PARAMETERS FROM THE OPTIMIZATION RESULTS OF ILP MODELS

	N = 10	*N = 50*	*N = 100*
Total nr. of caches (dense)	*2*	*7*	*13*
Total nr. of caches (sparse)	*3*	*10*	*15*
Average cache sizes (dense)	*1*	*3*	*4*
Average cache sizes (sparse)	*2*	*4*	*5*

Table I presents the average values resulted from 5 successful runs for each scenario. For the first optimization problem we show the total number of caches in the network. For the second optimization problem we show the average cache size / each node in the network.

We can see that when we optimize on the number of caches, we get larger numbers compared to the situation when we would evenly assign the maximum number of requesting nodes to each cache. This occurs due to the randomness of the topology: some caches might not serve nodes at full speed, as the requesting node is out of its contact area. As the network grows in size, the requests are distributed more evenly on average, thus relatively fewer caches suffice. For the second optimization problem we see that the average number of chunks to be stored at nodes remains low.

C. *Particularities of Different Networking Topologies*

The three scenarios introduced in section III-A have direct implications on the underlying networking topologies. Actually, the difference is made by the users (participants at the event), whose actions and movement is constrained in different manners. Obviously, any categorization of such behavior simplifies the scenario and in the case of real life deployment the operator or service provider should conduct its own assessment on the user group it wants to serve. With this remark in mind we still can model with good accuracy the behavior of the users, which has direct effect on the networking topology and mobility of their handheld smart devices. In the following we will focus on the nodes, even if the decision on their position and mobility is taken by their respective owners.

We have analysed the requirements in each scenario and we recommended the appropriate caching method for each, as follows (also see Table II).

TABLE II
CACHING METHODS FOR DIFFERENT SCENARIOS

	Stadium	*Festival*	*Tour*
XOR based coding			**X**
Minimal nr. of caches	**X**		
Minimal cache size		**X**	

In the Tour scenario the nodes are partitioned in separate groups along the track of the competition. Practically this results in smaller, isolated groups of nodes. Additionally, once a viewer joins a group, she/he will stick to that group. We have in this group lower number of nodes and higher group stability, so we considered that in this particular scenario we should rethink the priorities based on which we selected the network coding method implemented in the caches. Note that the RLNC variant of network coding offers us scalability (we can bring in many coding nodes if more caches are required) and flexibility (we do not have to define in advance the roles among caches). Nevertheless, in this tour scenario it is worth considering the fixed network coding, which trades flexibility for simplicity. Practically this requires in addition the definition of the nodes that have to act as caches, encoding nodes and for each cache the nodes that are linked to them.

The lightest technique for fixed network coding is the bitwise XOR [5][7]. In order to confer some flexibility and robustness to the caches using XOR based coding we propose the following solution. First, several nodes should agree on serving as caches. Then they should divide the roles, some of them storing packets without encoding, while at least one of them should store bitwise XOR-ed packets. Once constructed this caching group, any requester should choose any combination of those caches in order to be able to replay the stream.

In the stadium scenario we have static nodes, bound to the seats and we have a high node density. But due to the large number of nodes we might use the advantages of statistical multiplexing of more sources compared to the tour scenario. Under such conditions we should try to use the RLNC based method and minimize the number of caches. In order to avoid the battery drain, periodically we should change the caching roles of the nodes, similarly to the top peer rotation mechanism in SopCast.

Compared to the previous two scenarios, in the festival scenario we have much higher node mobility, because the participants can walk within the festival area. Also, we have large number of nodes. This leads us to the use of RLNC based solution, minimizing the cache size. This also spreads content as much as possible, and the nodes are not forced to rely only on few caches (which would happen if we minimize the number of caches instead).

D. Discussions on the Caching Details

In the case of RLNC, one question is to set the size of the Galois field. In our earlier work we used GF(2^8). This means that for each packet we should attach a 1 byte coefficient, which would result in large overhead. Given that there is no need to encode the same packet multiple times (which would imply the encoding of the coefficients, too) we can apply the workaround proposed in [23]. It suffices to embed only the seed to be used to generate the series of random coefficients. Additionally we have to take care to use the same pseudo random number generator at each node. This trick allows us to reduce the coefficient related overhead to a mere four bytes, and this value remains constant, whatever the number of encoded segments might be.

In our model we did not include the effect of the distance between the wireless source and destination, although in some wireless technologies this might change the coverage area of the source. Also we did not consider the possibility of overhearing [24] (which might be considered as an implicit multicast packet distribution) that would further increase the efficiency of our proposal.

Note that when we minimize the number of caches we might gain a collateral advantage. Because the cache will operate at higher loads, it will be more efficient, since it avoids idle periods (in terms of data transfer), under which it still has to keep its wireless interface active, waiting for newer requests. Therefore from p.o.v. of green networking the first optimization problem corresponds to the maximization of a naïve green networking model. Nevertheless, the details of this relation should be further investigated (e.g., the effect of the receiver's distance from the source).

VI. CONCLUSION

In crowded events several scenarios are possible where streaming media based services are required. Due to their high bandwidth demand, these applications heavily stress the local access networks. In such cases any extra service results in dramatical QoS degradation. One possibility to support such services is to offload the access network by local, distributed caching mechanisms. We have proposed such a solution and built a model to investigate the behavior of our proposal. We found that it is more advantageous than a simple distributed caching solution and discussed the particularities of the proposed scenarios.

Our proposal allows the design and deployment of added value services for future large events at lower infrastructure costs. In our future work we plan to investigate the integration of caching and peer-to-peer mechanisms for the real time streaming media distribution, expecting that the application supporting near real-time services brings further advantages to the service providers.

ACKNOWLEDGMENT

Csaba Simon's research work was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4.A/1-11-1-2012-0001 National Program of Excellence (NKP). The authors thank their colleagues Krisztián Németh and Attila Kőrösi for assistance in ILP modeling and simulation.

REFERENCES

[1] Cs. Simon, "In-network caching of media streams in access networks", (in Hungarian), National Program for Excellence (NKP) Newsletter, pp. 4., June 2014

[2] Cs. Simon, "Real-time Streaming Support in Crowds", EIT ICTLabs Partner Event – Future Networking Solution Workshop, April 2014

[3] Wi-Fi Alliance, "Wi-Fi Direct" - available from: http://www.wi-fi.org/discover-and-learn/wi-fi-direct

[4] Qualcomm White Paper, "LTE Direct Overview", http://www.qualcomm.com/media/documents/lte-direct-overview, July, 2013.

[5] R. Koetter, M. Medard, "An algebraic approach to network coding", IEEE Trans. on Networking, October 2003

[6] Li B, et al., "Random network coding in peer-to-peer networks: From theory to practice", 2011

[7] Gajic B, Riihijrvi J and Mhnen P., "Performance evaluation of network coding: Effects of topology and network traffic for linear and xor coding", Journal of Communication, vol. 4(11), pp. 885-893, 2009

[8] S. Chachulski and S. Katti, "Trading structure for randomness in wireless opportunistic routing", in Proc. of ACM SIGCOMM 2007

[9] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments", in Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing, 2005

[10] Gkantsidis, Ch., and Pablo R., "Network coding for large scale content distribution.", in Proc. of IEEE INFOCOM 2005. Vol. 4. IEEE, 2005

[11] D. Traskov, Lenz, J., Ratnakar, N. and Médard, M., "Asynchronous Network Coded Multicast", in Proc. of ICC Communication Theory Symposium, 2010

[12] X. Zhang, G. Neglia, J. Kurose, "Network Coding in Disruption Tolerant Networks", Network Coding: Fundamentals and Applications Elsevier Science (Ed.) 2011

[13] Zs. Zalatnay, Cs. Simon, M. Maliosz, B. Terza, "Managing streaming services in a distributed testbed", (accepted for publication) MACRO 2015, March 2015

[14] Li, B., Wang, Z., Liu, J., & Zhu, W., "Two decades of internet video streaming: A retrospective view", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 9(1s), 33., 2013

[15] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," Communications of the ACM, vol. 53, no. 8, pp. 80–88, 2010

[16] Jaleel, A., Theobald, K. B., Steely Jr, S. C., & Emer, J., "High performance cache replacement using re-reference interval prediction (RRIP)", In Proc. of ACM SIGARCH Computer Architecture News, Vol. 38, No. 3, pp. 60-71, 2010

[17] Cs. Simon, M. Markosz, B. Baranyai, "Network based caching for near real-time streaming video", (to appear in) Acta Universitatis Sapiantiae – Electrical and Mechanical Engineering, 1/2014.

[18] U. Feige, "A threshold of ln n for approximating set cover", J. Assoc. Comput. Mach., 45:634–652, 1998

[19] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems", J. Assoc. Comput. Mach., 41(5):960–981, 1994

[20] Har-Peled, S., Lee, M.,"Weighted geometric set cover problems revisited", Journal of Computational Geometry, 3(1), 65-85., 2012

[21] Hei, X., Liang, C., Liang, J., Liu, Y., & Ross, K. W., "A measurement study of a large-scale P2P IPTV system", *IEEE Transactions on Multimedia*, *9*(8), pp. 1672-1687., 2007

[22] Silverston, T., Fourmaux, O., " Measuring P2P IPTV Systems", In *Proceedings of NOSSDAV* (Vol. 7)., 2007

[23] Liu, Z., Wu, C., Li, B., Zhao, S., "UUSee: large-scale operational on-demand streaming with random network coding", IEEE *INFOCOM, 2010* (pp. 1-9). 2010

[24] D.E. Lucani, M. Médard, M. Stojanovic, "Systematic network coding for time-division duplexing", in Proc. IEEE Symposium on Information Theory Proceedings – ISIT, 2010

[25] Karp R. M., "Reducibility Among Combinatorial Problems", in Proc. Sympos. Complexity of Computer Computations, IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y. New York: Plenum, p.85-103. 1972

[26] Homepage of Lemon library - http://lemon.cs.elte.hu/trac/lemon

[27] GNU Linear Programming Kit homepage - https://www.gnu.org/software/glpk/

[28] GUROBI Optimization Libraries homepage - http://www.gurobi.com/

Csaba Simon received his PhD degree in computer science in the field of infocommunication systems at Budapest University of Technology and Economics (BME), Hungary.
He is an assistant professor in the High-Speed Networks Laboratory at the Department of Telecommunication and Media Informatics, BME. His research interests include QoS of IP networks, IP network architectures, network and service management. He participated in numerous national and international projects in the fields of network resource management, mobility management and smart content delivery.
He is a member of Scientific Association for Infocommunications, Hungary (HTE).

Markosz Maliosz received his MSc (1998) and PhD (2006) degrees in computer science in the field of infocommunication systems at Budapest University of Technology and Economics (BME), Hungary.
He is an assistant professor in the High-Speed Networks Laboratory at the Department of Telecommunication and Media Informatics, BME. His research interests include virtual, cloud and sensor networking along with optimization techniques. He participated in numerous national and international projects in the fields of network resource management, multimedia and smart content delivery.
He is a member of Scientific Association for Infocommunications, Hungary (HTE).

Call for Papers

Prospective authors are invited to submit original research papers for publication in the upcoming issues of our Infocommunications Journal.

Topics of interests include the following areas:

- Data and network security
- Digital broadcasting
- Infocommunication services
- Internet technologies and applications
- Media informatics
- Multimedia systems
- Optical communications
- Society-related issues
- Space communications
- Telecommunication software
- Telecom. economy and regulation
- Testbeds and research infrastructures
- Wireless and mobile communications

Theoretical and experimentation research results achieved within the framework of European ICT projects are particularly welcome. From time to time we publish special issues and feature topics so please follow the announcements. Propo-sals for new special issues and feature topics are welcome.

Our journal is currently published quarterly and the editors try to keep the review and decision process as short as possible to ensure a timely publication of the paper, if accepted. As for manuscript preparation and submission, please follow the guidelines published on our website

http://www.infocommunications.hu/for_our_authors.

Authors are requested to send their manuscripts via electronic mail (preferably) or on a CD by regular mail to the Editor-in-Chief:

Csaba A. Szabo
Dept. of Networked Systems and Services,
Budapest University of Technology and Economics
2 Magyar Tudosok krt., Budapest 1117 Hungary
e-mail: szabo@hit.bme.hu

Call for Proposals of Special Issues

Infocommunications Journal welcomes proposals for Special Issues – collections of papers dedicated to a particular topic of interest for the readers of the journal.

A Special Issue can be based on a recent high quality workshop or conference or papers can be collected from open call. Invited papers can be part of the special issue as well.

A Special Issue can fill in a whole issue, in which case the number of papers is expected to be 8 to 10, or it can be a Mini – Special Issue. In the latter case, at least 3, preferably 4 papers are required.

Proposals for special issues should include:

– contact information
(name, e-mail, title, affiliation and address);
– resume(s) of the proposer(s), with a representative list of recent publications and related experience (Editorial Board memberships, Guest Editorships, or roles in relevant conferences' program committees);
– the proposed title for the special issue;
– the way the special issue will be compiled (contributions solicited from a technical event, or to be collected from call for this special issue;
– intent to include invited papers should be also indicated, if possible with the names of professionals who are planned to be invited;
– scope and motivation and description of the special issue;
– guest editors (if different from the proposers) with detailed contact information and resumes.

Proposals should be sent to the Editor-in-Chief:

Csaba A. Szabo
Dept. of Networked Systems and Services,
Budapest University of Technology and Economics
2 Magyar Tudosok krt., Budapest 1117 Hungary
e-mail: szabo@hit.bme.hu