On File-based Content Distribution over Wireless Networks via Multiple Paths: Coding and Delay Trade-off

Jun Sun
MIT

Yonggang Wen
Nanyang Technological University

Lizhong Zheng
MIT

***Abstract*— With the emergence of the adaptive bit rate (ABR) streaming technology, the video/content streaming technology is shifting toward a file-based content distribution. That is, video content is encoded into a set of smaller media files containing video of 2-10 seconds before transmission. This file-based content distribution, coupled with increasingly rapid adoption of smartphones, requires an efficient file-based distribution algorithm to satisfy the QoS demand in wireless networks. In this paper, we study the transmission of a finite-sized file over wireless networks using multipath routing, with the objective to minimize file transmission delay instead of average packet delay. The file transmission delay is defined as the time interval from the instant that a file is first transmitted to the time at which the file can be reconstructed in the destination node. We observe that file transmission delay depends not only on the mean of the packet delay but also on its distribution, especially the tail. This observation leads to a better understanding of the file transfer delay in wireless networks and a minimum delay file transmission strategy.**

In a wireless multipath communication scenario, we propose to use packet level erasure code (e.g., digital fountain code) to transmit data file with redundancy. Given that a file with k packets is encoded into n packets for transmission, the use of digital fountain code allows the file to be received when only k out of n packets are received. By adding redundant packets, the destination node does not have to wait for the packet to arrive late, hence reducing the delay of the file transmission. We characterize the tradeoff between the code rate (i.e., the ratio of the number of transmitted packets to the number of the original packets) and the file delay reduction. As a rule of thumb, we provide practical guidelines in determining an appropriate code rate for a fixed file to achieve a reasonable transmission delay. We show that only a few redundant packets are needed to achieve a significant reduction in file transmission delay.

I. Introduction

With the growing popularity of smartphones such as the iPhone and the Android phone, mobile data traffic, driven mostly by video content, will increase at an estimated compound annual growth rate of 108 % in next few years [21]. Therefore, a key challenge in wireless network is to maintain a high level of QoS for the emerging mobile video data.

There is a rich set of research in the area of video streaming communication which use flow based packet transmission model [13] [14] [15]. However, with the emergence of adaptive bit rate (ABR) streaming technology [16][17][18][19], the video/content streaming technology is shifting toward a file-based content distribution. Specifically, instead of sending a huge media file, which validates the flow-based model, the video content is normally encoded into a set of smaller media files, each of which contains video content of 2-10 seconds. All these smaller media files are scheduled to transfer to the client, according to the organization specified in a manifest file. In light of this new development in video streaming technique, designing efficient filed-based content distribution algorithm becomes crucial in providing the end-user an enjoyable experience. Ideally, each of the media files should arrive as early as possible to meet its playback deadline. Hence in this paper, we focus on the problem of minimizing the file transmission delay in a wireless network.

To improve the delay performance in a wireless network, researchers have been exploring multiple paths that may exist between a source and destination pair by using parallel transmission. That is, for a file with a fixed number of packets, one can assign a certain fraction of these packets on each path and transmit them simultaneously. This strategy has been used in various communication scenarios such as the sparse MANET(Mobile Ad-hoc Network) with random mobility and the multi-beam satellite network with multiple ground stations. The sparse MANET communication scenario was studied in [10] and [11], and it is becoming technically feasible with the emerging smart phones. For example, Android phones [20] can work both as a video consumer and as an access point to relay traffic. In a sparse MANET, nodes independently roam within a specified area, and nodes can communicate with each other only when they are close to each other. When two nodes are within the communication range of each other, one can transfer a single packet (packet size scalable) to the other before the link severed. Since nodes are roaming randomly, a natural packet transmission strategy is to have the source node sending packets to some relay nodes, and let relay nodes deliver the packet to its destination when relay nodes and the destination node are close to each other. Each different relay node can be thought of as a disjoint path between the source and destination nodes. Similarly, in a multi-beam satellite system with multiple ground stations where the ground stations are connected through a terrestrial network, each channel from the satellite to a ground station can be thought of as a disjoint path.

In the aforementioned communication scenarios, the file transmission delay then can be very different from the packet

transmission delay especially when the distribution of the packet transmission delay has a heavy tail. After a source distributed packets of a file among the available paths, the destination can reconstruct the file when *all* the packets of that file have arrived. This may take a long time due to the heavy tail of the packet delay distribution. The potential long file transmission delay prompts us to code the original file at the packet level. Specifically, for a file with k packets originally, the source transmits $n > k$ packets by adding some redundant packets to the original file. At the destination node, upon receiving the first k packets out of the n transmitted packets, the destination node can reconstruct the original file. This type of coding scheme at the packet level exists such as the digital fountain code or tornado code [8].

Our objective in this paper is to obtain an intuitive understanding of the tradeoff between the code rate and delay reduction in a communication setting with a single or multiple source-destination pairs that sharing a set of parallel paths. In the single source-destination case, given a file size, we provide a practical guideline in determining a code rate that achieves a good balance between the packet redundancy and the file transmission delay. We show that only a few redundant packets are required for achieving a significant reduction in file transmission delay. The trade-off between the file transmission delay and code rate in a multiple users environment is presented in [3].

The problem of two nodes communicating using multiple paths has received considerable attention in various contexts. [1] [2] [4] [9] study the multipath communication problem in wireline scenarios. [12] [5] [6] focus on wireless scenarios. Due to space limitation, we will not discuss them in detail.

The rest of this paper is organized as follows: Section II describes the detailed formulation of this file transmission delay minimization problem. In section III, the coding and delay tradeoff in the case of a single source and destination pair is presented. Finally, section IV concludes this paper.

II. Problem Formulation

In this paper, we consider a communication network with a rich set of disjoint paths between a source and destination pair of interest. Given this set of disjoint paths, we focus on the problem of how to best utilize these paths to minimize the file transmission delay from the source node to the destination node. Here, the transmission delay of an object is defined as the time interval from the instant that the object is first transmitted to the time at which the object can be reconstructed at the destination node. This object can be a file or a packet. To characterize the file transmission delay and the packet transmission delay, we make the following assumptions:

- (A1) On each path, the packet transmission delay are independent and identically distributed random variables.
- (A2) The packet transmission delay are independent across all paths.
- (A3) On each path, the transmission delay of a group of consecutive packets is the sum of the transmission delay of each individual packet in that group. For example, given the transmission delay for packet 1 and packet 2 are T_1 and T_2 respectively, the transmission delay for sending both packet 1 and packet 2 is $T_1 + T_2$.

In A3, note that the transmission delay for sending both packet 1 and packet 2 is not $T_1 + T_2$ in most wireline networks. In those networks, a significant portion of T_1 and T_2 will overlap each other. The transmission delay for sending both packet 1 and packet 2 will be smaller than $T_1 + T_2$.

To justify the above assumptions, we consider the two communication scenarios mentioned in the previous section: a sparse MANET with random mobility [10] and a multi-beam satellite network with multiple ground stations. Relating the sparse MANET communication scenario to the multiple paths delay problem here, we see that each relay node can be thought as a disjoint path to the destination. Assumption A1 makes sense due to the random roaming nature *in time* of a relay node. For a particular relay node, the time between each encounter with the destination node is independent and identically distributed random variable. Since all nodes roam independently, the packet transmission delay of one relay node is independent from the other relay nodes; hence, A2 holds. A3 is valid because relay node can transfer only one packet to the destination during their encounter. For a relay node to deliver two packets to the destination node, two encounters with destination node are required.

In a multi-beam satellite system with multiple ground stations where the ground stations are connected through a terrestrial network, each channel can be modeled as an ON/OFF channel due to blockage. Assume that one packet can be transmitted in an ON slot, and the channel state (ON/OFF) in different slots are independent and identically distributed Bernoulli random variables. Also assume each channel is independent both in time and to each other (a reasonable assumption given the ground station are located far apart). We see that A1 and A2 immediately follows. Since a packet can only be transmitted during an ON slot, sending two packets on a path would require two ON slots. Hence, A3 holds as well.

In above communication scenarios, one can verify that the single packet transmission delay distribution has long tail (i.e., the probability of having a very large delay is nonzero). In fact, the packet transmission delay distribution approaches exponential distribution in both communication scenarios. It is straightforward to see this in the multibeam satellite network. To see this in the sparse MANET communication scenario, note that the packet transmission delay consists of two parts: the time needed for the source node to find a relay node and the time needed for the relay node to find the destination node. Due to random mobility, packet transmission delay will be dominated by the time needed for the relay node to find the destination node (i.e., it is easy for the source node to find a relay node). Hence, we ignore the time needed for the source node to find a relay node in the transmission delay. It is easy to see that the distribution of the time needed for the relay node to find the destination node appraoches exponential distribution. For simplicity, here we use exponential distribution with rate

μ to model the delay distribution of a single packet.

At the source node, a file consists of k packets. We assume there are n_p paths between the source and destination pair. The source can then encode this file into n packets such that the destination node can decode the whole file as soon as it received k packets (i.e., Digital Fountain code). Note that Digital Fountain code actually require the destination node to receive $k(1+\epsilon)$ packets in order to decode the original file, where ϵ is small. For the first part of the paper, we consider the case that the source is the only node that has packets to send to the destination node. Assume that a file is generated at the source node at time zero. The file transmission delay, denoted here as $\mathcal{D}$, is the time at which the destination node receive k packets. The code rate is define to be n/k.

III. Delay-coding tradeoff for a single source destination pair

We start this section by giving the following motivating example. Consider sending a file with k packets, numbered $(P_1, \cdots, P_k)$, from the source to the destination. Let the number of paths $n_p = 2$, and these two paths be identical. The time required for sending a packet P_i, denoted here as τ_i, is an i.i.d. exponential random variable with mean $\frac{1}{\mu}$. To transmit the file using both paths, a simple way would be to allocate packets $(P_1, \cdots, P_{k/2})$ on the first path and packets $(P_{k/2+1}, \cdots, P_k)$ on the second path, assuming that k is even. Let T_j, $j \in \{1, \cdots, n_p\}$, represents the total time needed for a path to clear all packets assigned to it. Here, we have the average packet delay

$$T_1 = \tau_1 + \cdots + \tau_{k/2}, \;\; T_2 = \tau_{k/2+1} + \cdots + \tau_k. \qquad (1)$$

The file transmission delay $\mathcal{D} = \max\{T_1, T_2\}$. For the case where k is much larger than n_p, we have

$$\mathcal{D}/(k/2) \approx T_1/(k/2) \approx T_2/(k/2).$$

Since k is large, both paths will be busy in serving the packets. In this case, the average packet delay will be a good indicator of the average file transmission delay. There is not much we can do to further reduce the file transmission delay. However, in this paper, we focus on the case where k is *not* much larger than n_p.

Now, consider the case where a file consists of only six packets and $n_p = 2$. We can assign packets (P_1, P_2, P_3) to the first path and (P_4, P_5, P_6) to the second path. If any one of the two paths clear its packets first, it will remain idle while the other path is transmitting its packets. We see that there is an non-negligible fraction of system time wasted in idle instead of serving packets. A natural way to resolve the above problem and reduce the file transmission delay is to do the following: assigning packets $(P_1, P_2, \cdots, P_6)$ on path one and transmitting these packets in this order; similarly, assigning packets $(P_6, P_5, \cdots, P_1)$ on path two and sending them in this order. This way, whenever the destination received a total of six packets, the original file can be reconstructed. Since both paths are transmitting packets (i.e., a faster path can serve more packets than the slower path instead of waiting idly), the file transmission delay will be reduced. In fact, the arrangement will minimize the file transmission delay. Therefore, as we added redundant packets on each path, the file transmission delay can be reduced. We now come to the first points that we want to illustrate in this paper: there is a relationship between the redundancy and the file transmission delay.

With only two paths, the above transmission strategy achieves the minimum file transmission delay. With $n_p \geq 3$, it is not clear how to allocate packet on each path so as to reduce the file transmission delay. In that case, we use digital fountain code for transmission so that we do not have to worry about the assignment of each specific packet on a particular path. All we need to concern is the total number of packets assigned to a particular path. In the following section, we are going to present the trade-off between coding and file transmission delay for two different cases: $n \leq n_p$ and $n > n_p$.

A. Case I: $n > n_p$

First, we consider the case where the number of packets in the file is greater than the number of parallel paths. For convenience, we let $k = l \cdot n_p$ and $n = m \cdot n_p$. In this section, we study the tradeoff between the file transmission delay and the amount of redundant packets added to the file. Specifically, we consider the file delay of a single source and destination pair with n_p identical and disjoint paths between them. We also define a transmission strategy to be a packet allocation vector $\vec{a} = \{a_1, a_2, \cdots a_{n_p}\}$, where a_i denotes the number of packets that needs to be transmitted on path i. The following lemma provides the optimal transmission strategy for using the multiple paths.

Lemma 1: Given a set of identical paths between a source and destination pair, the expected file delay is minimized when allocating packets evenly on each path.

The proof of the above lemma is omitted for brevity. For a file with $l \cdot n_p$ packets, the source can encoded the file to $m \cdot n_p$ packets. From the above lemma, we know that allocating packets evenly on each path will result in the minimum expected file transmission delay. Now, consider all n_p paths. With each path assigned m packets, we define $N_i(t)$ to be the number of packets that had arrived the destination node by time t on path i. We also define $N(t)$ similarly to $N_i(t)$ except that $N(t)$ does not associate with any particular path. To reconstruct the original file at the destination node at time t, the following condition has to be satisfied:

$$\sum_{i=1}^{n_p} N_i(t) \geq l \cdot n_p \quad \text{or} \quad \frac{1}{n_p} \sum_{i=1}^{n_p} N_i(t) \geq l \qquad (2)$$

The file transmission delay is given by:

$$\mathcal{D} = \inf\{t : \sum_{i=1}^{n_p} N_i(t) \geq l \cdot n_p\} \qquad (3)$$

As the number of path gets large, from the law of large number, we have

$$\lim_{n_p \to \infty} \frac{1}{n_p} \sum_{i=1}^{n_p} N_i(t) \to E[N(t)], \qquad (4)$$

and the file transmission delay can be written as:

$$\mathcal{D} = \inf\{t : E[N(t)] \geq l\} \tag{5}$$

Let $\hat{N}(t)$ denote the number of arrivals by the time t for a poisson process with rate μ. In the case where the packet transmission is exponentially distributed with rate μ, note that the first m arrivals of the process $N(t)$ and $\hat{N}(t)$ are statistically identical. Hence, we can write the expected number of packet arrived by time t as the following:

$$E[N(t)] = \sum_{i=1}^{m} i \cdot P(\hat{N}(t) = i) + m \cdot \sum_{i=m+1}^{\infty} P(\hat{N}(t) = i) \tag{6}$$

To get the file transmission delay, we need to first evaluate $E[N(t)]$. Expand Eq. (6), we have:

$$\begin{aligned} \sum_{i=1}^{m} iP(\hat{N}(t) = i) &= \frac{(\mu t)\Gamma(m, \mu t)}{\Gamma(m)} \\ m \sum_{i=m+1}^{\infty} P(\hat{N}(t) = i) &= \frac{\Gamma(m+1) - \Gamma(m+1, \mu t)}{\Gamma(m)} \end{aligned} \tag{7}$$

where

$$\Gamma(m, \mu t) = \Gamma(m) e^{-\mu t} \sum_{i=0}^{m-1} \frac{(\mu t)^i}{i!}$$

and $\Gamma(m) = 1/(m-1)!$. Thus, we have

$$E[N(t)] = \frac{(\mu t)\Gamma(m, \mu t) + \Gamma(m+1) - \Gamma(m+1, \mu t)}{\Gamma(m)} \tag{8}$$

Since $E[N(t)]$ is a continuous function in t, the file transmission delay $\mathcal{D}$ satisfies $E[N(\mathcal{D})] = l$. The above equation for $E[N(t)]$ is hard to solve in general. However, when $\mu \cdot t = m$, Eq. (8) has a simpler form. In this case, we have

$$(\mu t)\Gamma(m, \mu t) - \Gamma(m+1, \mu t) = -(\mu t)^m e^{-\mu t} = -m^m e^{-m}$$

and

$$E[N(t)] = m - \frac{m^m e^{-m}}{(m-1)!} \tag{9}$$

By using the Stirling approximation, we can further simplify the above equation as follows:

$$\begin{aligned} E[N(t)] &= m - \frac{m^m e^{-m}}{(m-1)!} = m - m \cdot \frac{m^m e^{-m}}{m!} \\ &\approx m - m \cdot \frac{m^m e^{-m}}{\sqrt{2\pi m} m^m e^{-m}} = m - \sqrt{\frac{m}{2\pi}} \end{aligned} \tag{10}$$

At this point, one may think that $\mu t = m$ is merely an equation that simplifies $E[N(t)]$. However, as we will show later, the equation $\mu t = m$ provides important insight in obtaining an asymptotically optimal coding and file delay tradeoff. First, let's examine the implication of $\mu t = m$. This equation tells us the following: Given a file that contains $(m - \sqrt{m/2\pi})n_p$ packets, the source first encodes the file into $m \cdot n_p$ packets and transmits m packets on each path. To reconstruct the file at the destination node, the destination node has to wait for $(m - \sqrt{m/2\pi})n_p$ packets to arrive. The time takes for $(m - \sqrt{m/2\pi})n_p$ packets to arrive the destination node, or the file delay, is simply $t = m/\mu$.

For a file with a fixed size, it is intuitive to see that the file delay will decrease as more redundant packets are added during the actual transmission of the file. However, the above communication scenario only provides the delay for one specific code rate (i.e., $m/(m - \sqrt{m/2\pi})$). It does not give us the file delay for other code rates. As the source adds redundant packets for transmission, at the beginning a few redundant packets may reduce the delay significantly while each additional redundant packet may not reduce the file delay much. Without a complete code rate and delay curve, it seems that we do not know how to achieve an appropriate balance between code rate and delay. Nevertheless, we are going to show next a coding strategy that achieve a good balance between the code rate and the file delay by using the equation $\mu t = m$. First, we derive the minimum file transmission delay, denoted here as $\mathcal{D}_{min}$, for a file with fixed size $l \cdot n_p$ in the following lemma.

Lemma 2: Given a file with $l \cdot n_p$ packets, the minimum achievable file transmission delay $\mathcal{D}_{min} = \frac{l}{\mu}$.

The following theorem presents a coding scheme that provides a good tradeoff between the code rate and the file transmission delay.

Theorem 1: For a file with $k = l \cdot n_p$ packets, coding the file with $n = m \cdot n_p$ packets, where

$$m = \left(\frac{1/\sqrt{2\pi} + \sqrt{1/(2\pi) + 4l}}{2} \right)^2,$$

will result in a file transmission delay that is $O(\sqrt{l})/\mu$ away from $\mathcal{D}_{min}$.

We omit the proof of the theorem due to limited space. The above theorem states that using coding rate

$$\left(\frac{1/\sqrt{2\pi} + \sqrt{1/(2\pi) + 4l}}{2}\right)^2 \frac{1}{l}$$

we can achieve a file transfer delay that is asymptotically optimal. That is, let

$$\epsilon = \frac{\mathcal{D} - \mathcal{D}_{min}}{\mathcal{D}_{min}},$$

we have the difference between the delay obtained using our simple coding scheme and the minimum delay goes to zero as l gets large (i.e., $\epsilon = O(1/\sqrt{l})$). This idea is illustrated in Fig. 1. For simplicity, we let $\mu = 1$ when we generate Fig. 1. The x-axis of the plot indicates the file transmission delay. The y-axis denotes the number of packet (per path) in the original file. Each curve in the plot represents a coding and file delay tradeoff for a fixed number of packets assigned to each path. For example, the top curve represents the coding and file delay tradeoff if we assign six packets to each path. Let the x-axis and y-axis of the points A and B be represented by (A_x, A_y) and (B_x, B_y) respectively. Thus, if the original file contains B_y packets per path, the file transmission delay will be B_x if we encoded the original file to six packets per path. As for point A, if the original file size is still $B_y = A_y$, the

delay will be A_x if we encoded the original file with infinity number of packets on each path. The benefit of using a code rate of infinity rather than $6/B_y$ is the reduction of the file transmission delay by $B_x - A_x$. The reduction of a file delay for using a code rate of $5/B_y$ instead of infinity is also shown in the plot. For the same file, the code rate of $6/B_y$ and $5/B_y$ will result in a different file transmission delay. Obviously, depending on the source's preference, one rate may be more suitable than the other. Using the transmission strategy stated in Theorem 1, the resulting coding and file delay tradeoff is plotted with a dashed line. As l gets large, this transmission strategy achieves a code rate of almost one (i.e., the redundant packets is negligible comparing with the original file size), and a file delay that is within $O(\sqrt{l})/\mu$ of the minimum file delay. Hence, Theorem 1 can serve as a practical guideline for adding

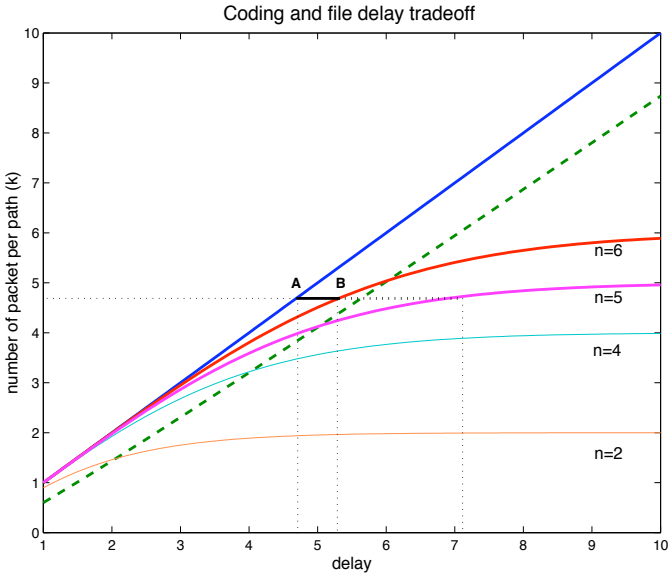

Fig. 1. Coding and file delay tradeoff.

redundant packets to the original file in order to reduce the file transmission delay.

B. Case II: $n \leq n_p$

In the case where the number of parallel paths, n_p, is larger than the number of encoded packets in a file, n, the expected file transmission delay can be obtained without the assumption that n_p is large. Assigning at most one packet to each path, the expected file transmission delay will be the expected value of the kth largest random variable out of the n random variables. Let X_i for $i = 1, \cdots, n$ denote the arrival time of the ith packet. Then, let S_i for $i = 1, \cdots, n$ denote the order statistics of X_i. The expected file transmission delay (i.e., the expected value of S_k) is given by the following:

$$E[\mathcal{D}] = E[S_k] = \sum_{i=0}^{k-1} \frac{1}{(n-i)\mu}. \quad (11)$$

In the case where the number of packets in a file is large while still satisfying $k < n < n_p$, we have the file transmission delay given by the following:

$$\mathcal{D} = -\frac{1}{\mu} \ln(1 - \frac{k}{n}). \quad (12)$$

See [3] for detailed analysis.

IV. Conclusion

In this paper, we explore the use of multipath routing to reduce the file transmission delay in a wireless network. By avoiding the long tail in the distribution of a packet's transmission time, we show that the file transmission delay can be significantly reduced with only a few redundant packets in the single source destination case. For a given file, an encoding strategy is provided to obtain a good code and file transmission delay tradeoff.

Acknowledgment

We acknowledge gratefully the helpful discussions with Professor Vincent W. S. Chan. The first author also like to thank the support of Professor Eytan Modiano.

References

[1] V. W. S. Chan and A. H. Chan, "Reliable Message Delivery Via Unreliable Networks," *IEEE International Symposium on Information Theory*, Ulm, Germany, 29 June - 4 July 1997.

[2] N.F. Maxemchuck, "Dispersity routing," *IEEE ICC'75*, San Francisco, CA, June 1975, IEEE, vol. 41, pp. 10-13.

[3] J. Sun, *Dynamic Channel Allocation in Satellite and Wireless Networks*, PhD thesis, MIT, May 2007.

[4] S. Mao, S. S. Panwar and Y. T Hou, "On Optimal Partitioning of Realtime Traffic over Multiple Paths," *INFOCOM 2005*, Miami, March 2005.

[5] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-Coding Based Routing for Opportunistic Networks," *SIGCOMM'05 Workshops*, Philadelphia, August 2005.

[6] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using Redundancy to Cope with Failures in a Delay Tolerant Network," *SIGCOMM'05*, Philadelphia, August 2005.

[7] M. Neely, E. Modiano, and C. Rohrs, "Power Allocation and Routing in Multi-Beam Satellites with Time Varying Channels," *IEEE Transactions on Networking*, February 2003.

[8] M. Mitzenmacher, "Digital Fountains: A Survey and Look Forward," *Information Theory Workshop*, 2004.

[9] Y. Wen and J. Sun, "On Minimum-Delay File Transport over Two-Connected Heterogenous Networks," *2007 IEEE Wireless Communication and Network Conference*, Hong Kong, P. R. China, March 2007.

[10] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Adhoc Wireless Networks," *IEEE/ACM Transactions on Networking*, August 2002.

[11] M. Neely and E. Modiano, "Capacity and Delay Tradeoffs for Ad Hoc Mobile Networks," *IEEE Transactions on Information Theory*, May 2005.

[12] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On Delay Performance Gains from Network Coding," *Proc. of Conference on Information Sciences and Systems (CISS)*, 2006.

[13] S. Mueller, R. P. Tsang and D. Ghosa, "Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges," *Lecture Notes in Computer Science*, Volume 2965, April 2004.

[14] J. He and J. Rexford, "Towards Internet-wise Multipath Routing," *Technical Report*, Princeton University, 2008.

[15] C. Cetinkaya and E. K. Knightly, "Opportunistic Traffic Scheduling over Multipath Network Paths," *INFOCOM*, 2004.

[16] *http://www.smoothhd.com/*

[17] *http://movenetworks.com/*

[18] *http://www.adobe.com/devnet/video/articles/osmf_overview_03.html*

[19] *http://en.wikipedia.org/wiki/QuickTime*

[20] *http://code.google.com/events/io/2010/*

[21] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2014.