Efficient Peer-to-Peer Data Dissemination in Mobile Ad-Hoc Networks

Siddhartha K. Goel, Manish Singh, Dongyan Xu*
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada

Abstract

Without infrastructural support from base stations, mobile nodes in an ad hoc network communicate with each other in a peer-to-peer fashion. This poses a challenge in data dissemination among the mobile peers, each having limited transmission range and unpredictable mobility. In this paper, we first propose a novel mobility model to characterize user mobility in a civilian environment, such as a college campus. We then propose the application of Tornado coding as an efficient solution to the challenge of ad hoc peer-to-peer data dissemination. A mobile node is able to download coded file segments from different peers - at different times and in different locations. When it receives sufficient segments, it will be able to re-construct the original file. Although Tornado coding itself is not new, we for the first time propose its application to mobile ad hoc data dissemination. Furthermore, we show how Tornado coding parameters affect the performance of peer-to-peer data dissemination.

1. Introduction

A mobile ad-hoc network is formed dynamically without requiring any wireline infrastructure. Therefore, it has found wide applications in military operations, disaster relief efforts, and more recently, in civilian and ubiquitous environments. In this paper, we study a specific type of application: the dissemination of popular data files among mobile ad hoc users. Such dissemination is peer-to-peer in nature: a mobile node may be a requester of a popular data file, such as an image or an audio clip; and it becomes a supplier of this data file after it retrieves the file from other supplier(s). Meanwhile, unlike wired or wireless cellular networks, every mobile node has a limited transmission range. Therefore, two mobile nodes can communicate with each other directly, only if they are in the transmission range of each other. To communicate with a peer outside its transmission range, a mobile node has to rely on one or more intermediate peer(s) as relay(s). Due to the free movement of mobile nodes, both direct and indirect connections between peers can be disconnected very frequently.

Recently, there have been research efforts in data sharing and dissemination in mobile (not necessarily ad hoc) environments. One approach is for each mobile node to *prefetch* all possible data files from Info-stations, i.e. small scattered areas each covered by a high speed wireless LAN [6], before roaming; and the mobile nodes will *not* share data with each other while they are on the move. Another approach is for a requesting peer to query geographically nearby peers; and the requester will get the requested data file from *one* of the nearby supplying peers [8]. We argue that both approaches have limitations. In the first approach, each mobile node needs to predict all data files it will access in future. However, such prediction is very unlikely to be accurate. Furthermore, the 'prefetching without sharing' approach does not exploit the ad hoc communication capability between mobile nodes. In the second approach, the 'download from one peer' design may not be effective, if the file size is relatively large and the peers move frequently and get out of the transmission range of each other before the file transmission is completed. Even if this does not happen, a supplier peer tends to be overloaded with multiple file downloading connections to multiple requesters.

In this paper, we propose an efficient scheme for the

*Corresponding author (dxu@cs.purdue.edu).

sharing and dissemination of popular data files in mobile ad-hoc networks. Our proposed scheme does *not* assume instantaneous file transfer between mobile nodes, yet it brings both significant speed-up and reliability to file sharing and dissemination. Our major contributions in this paper are the following. First, we propose a novel mobility model called 'Street-and-Building Model', which is suitable for the modeling of peer mobility in a civilian environment, such as a college campus or a downtown business area. Second, we present our protocol which enables efficient and reliable data dissemination in mobile ad-hoc networks by applying the technique of Tornado coding [2]. In our protocol, supplying peers of a data file broadcast the Tornado-encoded file segments to requesting peers. On the other hand, a requesting peer downloads the encoded segments from different supplying peers - at different times and in different locations. When it receives sufficient segments, it will be able to re-construct the original file and it will also become a supplier. By using Tornado coding, our protocol is highly robust against packet losses common in ad hoc networks. Furthermore, it is highly efficient in data transmission, eliminating the need for multiple unicast downloading connections from multiple suppliers. Finally, we show how to determine the Tornado coding parameters under our Street-and-Building mobility model.

The rest of the paper is organized as follows. In Section 2, we present the Street-and-Building mobility model. In Section 3, we describe details of our data dissemination protocol. In Section 4, we present our simulation results. We discuss and compare related work in Section 5. Finally, we conclude the paper in section 6.

2. Street-and-Building Mobility Model

To design protocols for data dissemination in mobile ad-hoc networks, we first need a mobility model to capture the mobility pattern of mobile nodes with reasonable accuracy. Some recent works use either a random walk model [8] or a group clustering model [9]. The former model is not able to characterize the regularity of user mobility, commonly found in a civilian environment. On the other hand, the latter model makes too strong an assumption about the uniformity of mobility among mobile nodes. To overcome these problems, we propose a new *Street-and-Building* model for mobility modeling, which is especially suited for civilian environments.

Our model is based on the fact that all walking users on a given street have quite similar velocity at a given time, with a certain adjustable variation. Therefore, it is quite reasonable to assume that on a street for pedestrians, the maximum walking speed of mobile nodes is a constant v_0 (v_0 can be, say, 3 miles/hour). Furthermore, as the population density increases on the street, the walking speed of people decreases. The walking speed $v(t)$ at time t on a street s can be given by the formula:

$$v_s(t) = v_0 * (1 - x_s(t)) \quad (1)$$

$x_s(t)$ is the population density factor, and its value is in the range of $[0, 1]$. The higher the $x_s(t)$, the more congested the street is at time t.

For a given area such as a campus or a business area, the mobility modeling using the Street-and-Building model takes the following steps: (1) identify the streets and buildings in the area; (2) define the speed function $v_s(t)$, and direction θ_s for each street s in the area; (3) at any time instance, a mobile node can be either inside a building or on a street. For each node i on a street s, its direction is either θ_s or $-\theta_s$ (assuming there is bi-directional traffic on each street, with equal probability of motion in both directions and no component of the velocity being perpendicular to the street). Therefore the velocity of node i can be represented by $v_s(t) + \sigma_i$, σ_i is the velocity variation specific to node i.

3. Peer-to-Peer Data Dissemination Protocol

We now present the peer-to-peer data dissemination protocol. Consider an area with $x\%$ of the mobile nodes having tornado encoded segments of data file D in their mobile devices (for the rest of the paper, we will refer the file segments as application-level packets, or simply *packets*). We assume that they obtain the encoded packets while they are connected to the Internet before roaming. These mobile nodes are the initial supplying peers of D, and they periodically broadcast the packets of D to all interested peers in their transmission range [1]. On the other hand, a requesting peer listens to the packet broadcast from different supplying peers - at different times and in difference locations - while it moves along. Requesting peers do not send explicit requests for particular packets. They listen to broadcasts and store the packets received. Note that both requesting peers (before they are able to re-construct D) and supplying peers cooperate in the protocol by relaying (broadcasting) packets they receive or possess. Thus packets are broadcasted multiple hops away from the supplying peers, until they reach such nodes that have broadcasted the same packets in the *recent* past. As soon as a requesting peer receives sufficient number of distinct packets needed to re-construct file D, it will perform the Tornado decoding and restore D. At this time, it becomes a supplying peer, or as we call it, 'infected'. This protocol does not involve the sharing of transmission states among peers (via beacon messages, for example). Each supplying

[1] A supplying peer may choose not to broadcast, if it realizes that another supplying peer in its transmission range is already broadcasting the packets.

peer is not aware of the requesting peers around it, and the order of packet broadcast is determined *randomly*. Such a design significantly improves the system scalability and reduces the overhead and complexity of both supplying and requesting peers. The only non-trivial overhead introduced is the Tornado decoding performed by the requesting peers.

In the aspect of Tornado coding, let c denote the coding stretch factor [2]. File D is divided into k packets of equal size, and Tornado encoding will generate $n = c * k$ packets for each supplying peer to broadcast. For each requesting peer, it needs to receive $m = \epsilon * k$ packets to re-construct file D, where ϵ determines the extra number of packets needed. ϵ is also known as decoding inefficiency [2]. To apply Tornado coding to our peer-to-peer data dissemination protocol, we need to determine two key coding parameters: the packet size (or k - because $k*$ packet size = file size) and the stretch factor c, which are discussed in the following subsections, respectively.

3.1 Determining Packet Size

Constraints on the packet size can be set such that the wastage of bandwidth is minimized. The wireless bandwidth is wasted if a requesting peer receives an incomplete packet. More specifically, if the packet size is too large, then before the requesting peer is able to receive a complete packet, it may be out of the transmission range of the supplying peer. The incomplete packet is of no use to the requesting peer and furthermore, the bandwidth consumed by the supplying peer to transmit the incomplete packet is wasted.

Consider Figure 1, if a requesting peer is in the transmission range of supplying peer i for an overlapping period of t_i, and the data transmission rate is R, then the amount of data the requesting peer receives during the overlapping period is: $R * t_i$. Let p denote the size of a complete packet. Then the size of the incomplete packet received last during the overlapping period is $(R * t_i)\%p$. Suppose the requesting peer has received packets from n supplying peers before it is able to re-construct the file, the total volume of incomplete packets it has received can be expressed as $\sum_{i=1}^{n}(R * t_i)\%p$. This volume should be minimized for each requesting peer, which calls for a packet size as small as possible. However, p cannot be infinitely small, because that will increase the Tornado decoding overhead at the requesting peers. Based on the above observation, we propose the following heuristics to determine p: let t_{min} be the shortest overlapping period experienced by any pair of requesting/supplying peers in the network, then we can have $p = \lfloor R * t_{min}/m \rfloor$. In other words, this value of p will ensure that at least m (m is a configurable system parameter) complete packets will be received, even during the shortest overlapping period.

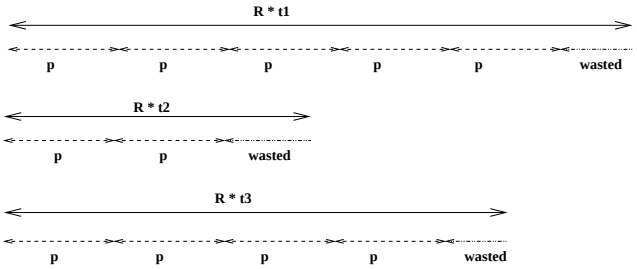

Figure 1. Determining packet size p: during each overlapping period, the requesting peer receives a sequence of complete packets followed by an incomplete packet

3.2 Determining Stretch Factor

For a requesting peer to re-construct the original file, it must receive $\epsilon * k$ distinct packets. These packets are transmitted from different supplying peers. The supplying peers randomly determine their packet broadcast sequences, in order to minimize the number of duplicate packets received by each requesting peer. However, this will not prevent a requesting peer from receiving duplicates. Following the analysis in [2], we can estimate the number of distinct packets received from each supplying peer. Suppose $\lfloor (R * t_1)/p \rfloor$ packets are received from the first supplying peer. Then if $\lfloor (R * t_2)/p \rfloor$ packets come from the second supplying peer, each of these packets may be a duplicate with a probability of $(\lfloor (R * t_1)/p \rfloor)/(c * k)$. Therefore, $\lfloor (R * t_2)/p \rfloor * (1 - ((\lfloor (R * t_1)/p \rfloor)/(c * k)))$ packets from the second supplying peer are expected to be distinct. Similarly, we can determine the expected number of distinct packets from the ith supplying peer... This way, we can sum up the total number of distinct packets from different supplying peers, until the number exceeds $\epsilon * k$. At this time, we derive the expected number of supplying peers a requesting peer is supposed to listened to before the file re-construction.

With the expected number of supplying peers for each requesting peer, we then determine the stretch factor. The stretch factor must be chosen such that a requesting peer can re-construct the file, without any supplying peer having to cycle through the $c * k$ packets again. Let δ be the probability that a packet is dropped or corrupted during transmission, then the probability that the packet is received without loss/corruption is $(1 - \delta)$. Therefore, during one transmission cycle of a supplying peer, the requesting peer can receive a maximum of $\eta = c*k*(1-\delta)$ packets from the supplying peer. As stated earlier, $\lfloor (R*t_i)/p \rfloor$ is the number of packets received by the requesting peer from supplying peer i. The value of c should then be determined such that $\eta \geq max_{\forall i}(\lfloor (R * t_i)/p \rfloor)$. In other words, the number of

packets received from any supplying peer should not exceed η, or the expected number of packets that can be completely received by the requesting peer in one transmission cycle of each supplying peer. We have thus determined a lower bound on c. By increasing the stretch factor, we can further reduce the probability that a requesting peer receives duplicate packets. However, c should not be infinitely large, because the larger the c, the greater the Tornado encoding and decoding overhead introduced.

4 Performance Evaluation

4.1 Simulation Setup

To evaluate the performance of our protocol, we perform extensive simulations using a simulator we have developed in C. Simulation experiments are performed with 1000 nodes randomly placed across a campus with a size of one square kilometer. A snapshot of the simulated environment is shown in Figure 2. A data file of size 1MB is used for dissemination. Each supplying peer has a transmission rate of $R = 128$K bps. 10% of the peer population are randomly assigned as initial supplying peers, while 70% of the peer population are randomly assigned as requesting peers. All the peers are assumed to be cooperative. It is also assumed that to re-construct a data file of size 1MB, a requesting peer will need 1.055MB data due to the decoding inefficiency factor ϵ. The transmission range of each peer is 20 meters. One run of the simulation lasts for 500 seconds. Initially each peer is positioned in a building or on a street. Initial coordinates within building (corresponding to a room) or on street are randomly determined. If a peer is placed in a building, it stays static for a certain wait period. If a peer is placed on a street, it is assigned a velocity according to the mobility model explained earlier.

Our simulation progresses in a second-by-second fashion. During each second, the following steps are performed by every supplying peer or a non-supplying but collaborative peer willing to re-broadcast the encoded packets:

- Determine the set of non-supplying peers (neighbors) within its transmission range. These are the peers that will receive the packets it broadcasts.
- For a supplying peer, randomly select a packet - previously unsent, from the set of encoded packets to broadcast to all its neighbors. It maintains an array of the encoded data packets and cycles through them.
- For a non-supplying peer which is relaying packets, randomly select a packet from the packets received previously but not yet relayed, and broadcast it to all its neighbors. Note that such a peer has fewer number of encoded packets than a supplying peer.

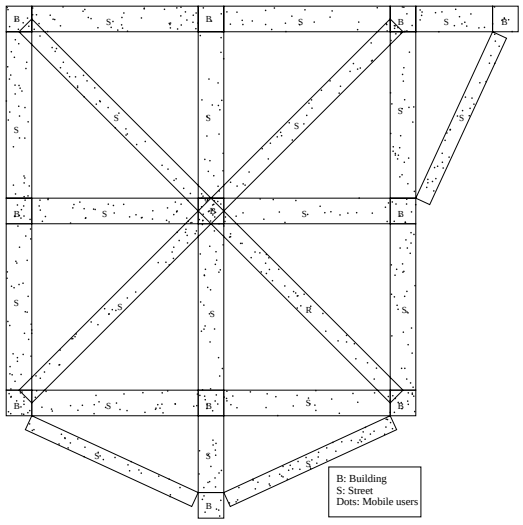

Figure 2. A snapshot of the simulated environment

- If time remains in the second (note the time to broadcast a packet is p/R), then the above process is repeated. It may be that a packet is partially received by a neighbor in the remaining time, because it moves out of the transmission range. In this case, the incomplete packet will be discarded.

For simplicity, our simulation assumes a reliable medium access layer. Therefore, packet losses are only due to loss of connectivity (moving beyond transmission range). For each requesting peer, the downloading time is the time required to obtain the $\epsilon * k$ packets. In addition, there will be a non-trivial latency for the file re-construction from these encoded packets.

4.2 Simulation Results

Figure 4 compares the progress of data dissemination with and without Tornado encoding. For the simulation without Tornado encoding, the 1MB data file is divided into 400 packets, each with a size of 2.5KB. All these 400 packets have to be received by each requesting peer in order to restore the file. For the simulation with Tornado encoding, a stretch factor of 2 is used. Therefore, 800 packets of size 2.5KB are generated, out of which 422 ($\epsilon * 400$) packets are needed by each requesting peer to correctly re-construct the file. We observe that the first requesting peer becomes 'infected' after 25 seconds in the Tornado encoding simulation, while it takes more than 100 seconds to infect the first requesting peer in the simulation without Tornado encoding. Almost all the requesting peers become 'infected' after 150 seconds in the Tornado encoding simu-

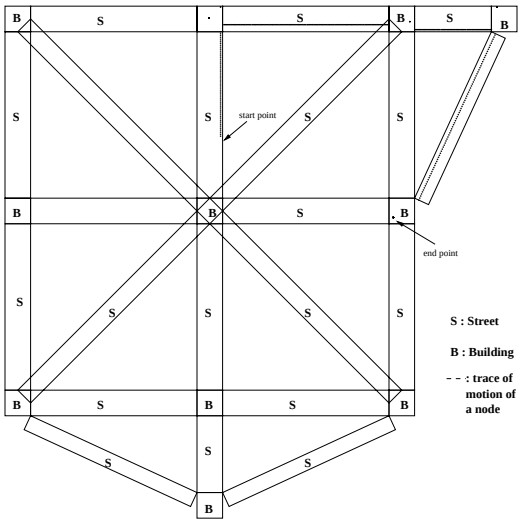

Figure 3. A trace of the motion of a particular peer during the simulation.

lation, while it takes around 450 seconds in the simulation without Tornado encoding to infect all the requesting peers. This experiment demonstrates the efficiency of Tornado-coding-based dissemination of popular content in a mobile ad-hoc environment.

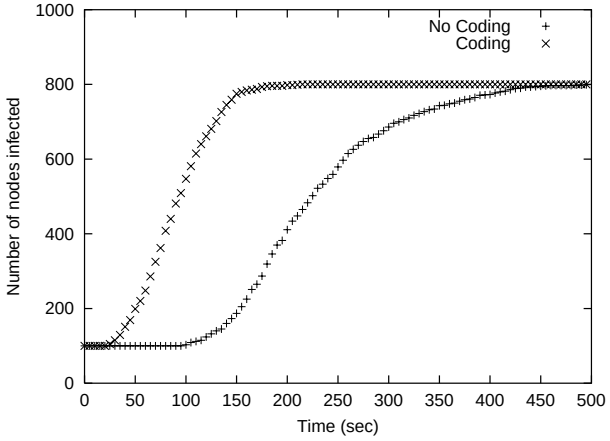

Figure 4. Progress of data dissemination: with and without Tornado coding

Figure 5 shows the progress of Tornado-coding-based data dissemination, under various packet sizes. As in the previous experiment, the 1MB data file is stretched to 2MB of encoded data under a stretch factor of 2. However, this may be done by generating 800 packets of size 2.5KB, 200 packets of size 10KB, 40 packets of size 50KB, or 20 packets of size 100KB. The transmission rate is 128Kbps. For packets of size 2.5KB or 10KB, they can be transmitted in an overlapping period as short as 1 second. However, packets of size 50KB can be fully transmitted only when the overlapping period is at least 4 seconds; and packets of size 100KB require an overlapping period of at least 7 seconds. Recall that a packet is deemed useless if it cannot be received in full by a requesting peer. This explains why the curves that correspond to packet sizes 2.5KB and 10KB grow faster than that for the 50KB packet size, which in turn grows faster than the curve for the 100KB packet size. We also notice that the curve for 2.5KB packet size is slightly steeper than the curve for 10KB packet size. This can be explained as follows: suppose the average overlapping period between a supplying peer and a requesting peer is 1 second, then six 2.5KB packets, or 15KB of data can be transmitted, while only one 10KB packet can be fully transmitted during the same period. Similarly, if the overlapping period is 3 seconds, then nineteen 2.5KB packets, or 47.5KB of data can be transmitted fully, versus only four 10KB packets or 40KB of data fully transmitted. This experiment shows that bandwidth wastage is reduced with decrease in packet size. However, excessively small packet size will lead to a large number of packets and thus increase the Tornado decoding overhead.

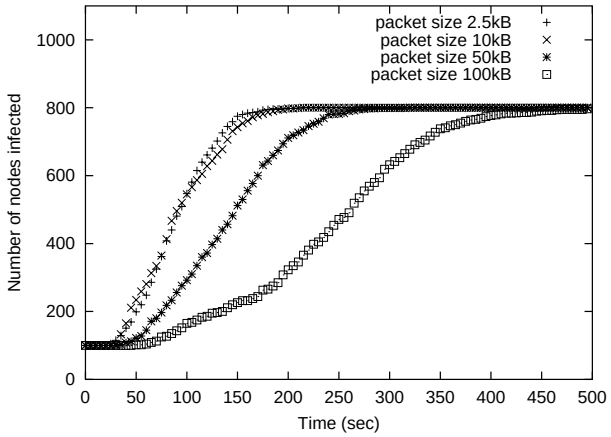

Figure 5. Progress of data dissemination: under different packet sizes

Figure 6 shows the Tornado-coding-based data dissemination overhead, under various stretch factors. The overhead is measured by the number of duplicate packets received by each requesting peer. Each point on the x-axis corresponds to one of the 700 (i.e. 70% of the 1000 nodes) requesting peers. In this figure, all three curves are monotonically increasing because the requesting peers are sorted by the number of duplicate packets they receive. In this experiment, the packet size is 5KB, while the stretch factor is 1.2, 2, or 4. In all cases, 211 ($\epsilon * k$) packets

are needed to re-construct the data file. We observe that the higher the stretch factor, the lower the number of duplicate packets received by a requesting peer. Since Tornado encoded packets are sent in a random order by each supplying peer, a higher stretch factor leads to lower probability of receiving duplicates. However, we also notice that when the stretch factor increases, its effect in duplicate packet reduction becomes less significant.

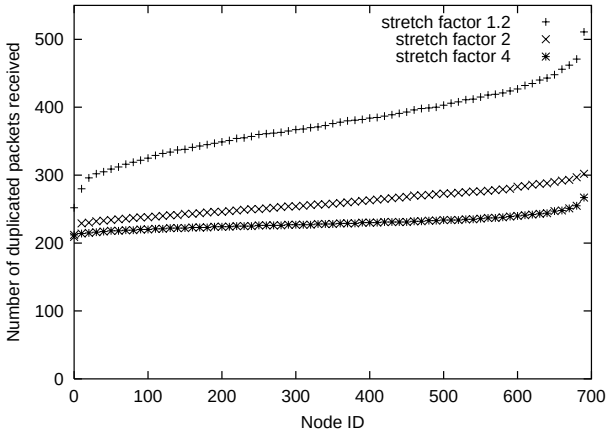

Figure 6. Number of duplicate packets received by each requesting peer: under different stretch factors

Figure 7 compares the performance of data dissemination under various stretch factors. The performance is measured by the total file downloading time experienced by each requesting peer. Similar to Figure 6, the curves are monotonically increasing as the requesting peers are sorted by the total file downloading time they experience. For a stretch factor of 2, the first peer fully receives and re-constructs the data file after 25 seconds, and all 700 requesting peers are able to re-construct the file within 155 seconds. We notice that this is very close to the downloading time under the stretch factor of 4. In the latter case, the first peer re-constructs the file after 20 seconds, and all the 700 requesting peers re-construct the file within 150 seconds. However, under a stretch factor of 1.2, the downloading time increases considerably: the first peer re-constructs the file in 45 seconds, and all the 700 requesting peers re-construct the file within 215 seconds. This is because with a low stretch factor, there are more duplicate packets received by the requesting peers, thus requiring them to wait longer to collect the sufficient distinct packets to re-construct the file.

Finally, Figure 8 compares the file downloading time under different packet sizes. In this experiment, the stretch factor is fixed at 2. When the packet size is 5KB or 10KB, we observe that all requesting peers experience very

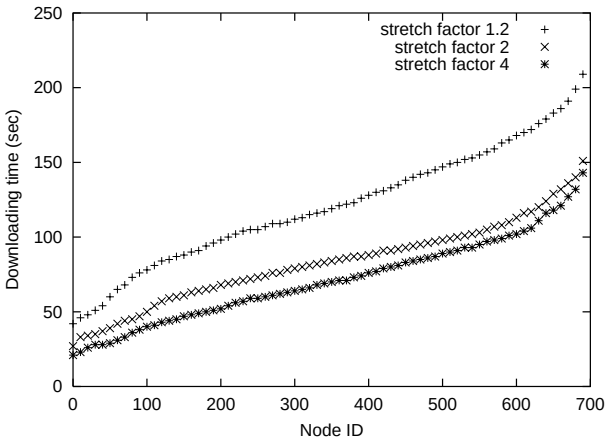

Figure 7. File downloading time of each requesting peer: under different stretch factors

similar downloading time, with minimum and maximum downloading time of 30 seconds and 150 seconds, respectively. When the packet size is 20KB, the downloading time increases - but not significantly, with minimum and maximum downloading time of 30 seconds and 180 seconds, respectively. However, when the packet size becomes 50KB, the downloading time increases significantly. The explanation is that as packet size increases, the reception of incomplete packets increases, resulting in bandwidth wastage and longer file downloading time.

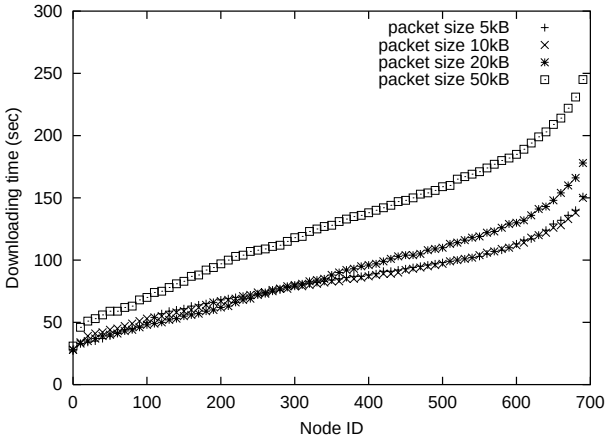

Figure 8. File downloading time of each requesting peer: under different packet sizes

5 Related Work

In this section, we discuss the related work previously proposed to improve data accessibility in mobile (not necessarily ad hoc) networks. Kubach *et al* propose a prefetching approach to accessing data in wide-area wireless networks [6]. Their approach predicts the information needed in the future, and hoards this information on the mobile device from Info-stations before the user begins roaming. The limitation of this approach is that it assumes the information is location-dependent, and that the information will be requested only when the user is close to the corresponding location. Future requests are predicted by learning from the history of mobility/request patterns of the same or other users, assuming a recurring pattern in user preference and motion. The amount of information hoarded is limited by the memory size of mobile devices. Meanwhile, their approach assumes a wide deployment of Info-stations, which may not be true in practice.

Various mobility models have been proposed for mobile ad hoc networks [9, 8, 5]. Hong *et al* have proposed a Reference Point Group Mobility Model in which groups of mobile nodes are formed based on their geographical proximity [5]. Wang *et al* have proposed a Reference Velocity Group Mobility Model in which groups of mobile nodes are formed based on the relative velocities of the nodes [9]. In both [9] and [5], each peer must know the identities of all group members. In terms of information exchange, this may prove quite expensive. The frequency at which group membership information is exchanged, as well as the paths along which group membership information is propagated require careful consideration. Papadopouli *et al* have adopted a Random Walk Model for mobility modeling in their simulation experiments [8]. The Random Walk Model may not be suitable for highly regulated civilian environment, because it assumes that a mobile node has equal probability of moving in *any* direction.

In [4], Hara proposes a placement and partition scheme for replicated data access in wireless ad hoc networks. It focuses on the provision of high data availability in a partitionable ad hoc network, so that each mobile node can access a copy of the data it needs with high probability. However, it does not address the issue of dynamic data dissemination, especially in a peer-to-peer fashion.

Byers *et al* first propose Tornado coding [1, 2, 3, 7]. Their scheme involves the encoding of a large file (of k packets) to n encoded packets (where n = stretch factor * k). The original file can then be recovered by decoding $\epsilon * k$ arbitrary but distinct encoded packets. They apply Tornado coding to parallel file downloading from multiple *static* hosts, while we propose to apply Tornado coding to data dissemination in mobile ad hoc environments. We argue that the selection of Tornado coding parameters has to reflect the mobility model of the targeted ad hoc environment.

6 Conclusions

In this paper, we propose the application of Tornado coding to data dissemination in mobile ad-hoc networks, with the objective of enabling efficient and reliable peer-to-peer data sharing among mobile users. Our solution consists of (1) a Street-and-Building mobility model suitable for modeling mobile users in a regulated civilian environment and (2) a peer-to-peer data dissemination protocol to dissemination Tornado encoded file segments (packets). We discuss the impact of Tornado coding parameters on the performance of our peer-to-peer data dissemination protocol. Our simulation results show satisfactory performance of the protocol: it reduces the file downloading time of a requesting peer by as much as 75%. The simulation results also demonstrate the importance of mobility-aware Tornado coding parameter selection.

References

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. *In Proceedings of ACM SIGCOMM 2002*, August 2002.

[2] J. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. *In Proceedings of IEEE INFOCOM'99*, pages 275–83, March 1999.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. *In Proceedings of ACM SIGCOMM'98*, September 1998.

[4] T. Hara. Effective replica allocation in ad-hoc networks for improving data accessibility. *In Proceedings of IEEE INFOCOM 2001*, April 2001.

[5] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad-hoc wireless networks. *In Proceedings of the 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems*, 1999.

[6] U. Kubach and K. Rothermel. Exploiting location information for infostation-based hoarding. *In Proceedings of ACM/IEEE MOBICOM 2001*, 2001.

[7] M. Luby, M. Mitzenmacher, and A. Shokrollahi. Analysis of random processes via and-or tree evaluation. *ACM/SIAM Symposium on Discrete Algorithms (SODA'98)*, 1998.

[8] M. Papadopouli and H. Schulzrinne. Design and implementation of a peer-to-peer data dissemination and prefetching tool for mobile users. *In Proceedings of the 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems*, 1999.

[9] K. Wang and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. *In Proceedings of IEEE INFOCOM 2002*, June 2002.