USENIX Association

Proceedings of MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services

San Francisco, CA, USA
May 5-8, 2003

iFlow: Middleware-assisted Rendezvous-based Information Access for Mobile Ad Hoc Applications

Zongpeng Li, Baochun Li, Dongyan Xu, Xin Zhou [*]

Abstract

Due to node mobility and limitations on bandwidth availability in wireless channels, there exist unique challenges towards achieving efficient and effective information access in wireless ad hoc networks with mobile nodes. In this paper, we address two critical questions: (1) How may information be accessed with the highest degree of bandwidth efficiency? and (2) How should algorithms be designed so that node mobility contributes positively towards high performance and efficiency? We present iFlow, a middleware-based framework for bandwidth-efficient and delay-aware information access for mobile ad hoc applications. We present the case of information rendezvous, where the demands for information are satisfied by the supplies in a fully distributed fashion, across third-party nodes beyond information suppliers and consumers. Such rendezvous is achieved via controlled diffusion of information from the suppliers, matched by the gleaning process on the consumers. We validate our claims using simulation and experimental results.

1 Introduction

The driving force and technology push for next-generation wireless networks are, and will always remain to be, the *applications*. A better understanding of the needs of emerging applications and wireless services leads to better designs of network protocols. On the other hand, the behavior of applications are diverse and often unpredictable. We may need to exert influence and control over such behavior, so that their needs are better understood and, in some cases, mathematically tractable. Particularly, *information access* is ubiquitously required in most of such applications. The information flow across wireless networks may exhibit specific patterns. From the point of view of resource utilization, we may prefer the patterns that may achieve the optimal bandwidth efficiency, as long as the requirements of applications are satisfied. This is especially the case in hybrid wireless networks that include ad hoc networks, where bandwidth efficiency is critical to their operations [1].

In this paper, we seek to design a middleware framework and middleware-based algorithms to achieve *bandwidth-efficient* information access, tailored to the needs of distributed applications on mobile ad hoc networks (which we refer to as *mobile ad hoc applications*). Particularly, we consider the case where application components on a subset of the nodes are information *suppliers*, while other components may be the *consumers*. Specific scenarios include: (1) ad hoc sensor networks where a subset of nodes are "sensors" that supply environmental data, and others are "reporters" that deliver sensor data to the users [2, 3]; and (2) hybrid wireless networks with a subset of "gateway" nodes to the Internet that supply information from the web to the regular nodes [4]. Such a case of suppliers and consumers does not limit its generality: with any applications, a node may either be a supplier or a consumer (or both) at any given time, constituting a web of supplier-consumer relationships. Without loss of generality, we use the example of hybrid wireless networks with gateway nodes as an example in this paper.

In such hybrid wireless networks, we present *iFlow*[1], a middleware architecture and a set of distributed algorithms to control the behavior of information access in mobile ad hoc applications, so that the goal of maximizing bandwidth efficiency with the presence of node mobility may be achieved. We identify the advantages of *information rendezvous*, where the demands for information are satisfied by the supplies in a fully distributed fashion, across third-party peer nodes beyond information suppliers and consumers. Such rendezvous is achieved via *controlled diffusion* of information from the suppliers, matched by the *gleaning* process on the consumers. In other words, requests are satisfied by results on third-party nodes in between suppliers and consumers. Beyond information rendezvous, we propose to

[*]Zongpeng Li, Baochun Li and Xin Zhou are affiliated with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are {*arcane,bli,zhoux*}*@iqua.ece.toronto.edu*. Dongyan Xu is affiliated with Department of Computer Sciences, Purdue University. His email address is *dxu@cs.purdue.edu*. This work is sponsored in part by Bell Canada University Laboratories.

[1]*iFlow* stands for *information flow*. Our goal is the efficient flow of information across the network in mobile ad hoc applications.

apply network coding on third-party nodes, so that they may transmit recoded data to achieve even higher bandwidth efficiency.

The original contributions brought forth by the iFlow architecture are the following: (1) We have analyzed the case of activating controlled diffusion compared with separate information access from individual nodes, and show that in most cases iFlow contributes to achieving better bandwidth efficiency. (2) In iFlow, we have identified the relationship between the delay tolerance of applications and achievable bandwidth efficiency, so that for more delay-insensitive applications, bandwidth efficiency may be further improved. (3) Unlike some of the previous work, we explicitly consider node mobility in iFlow, and design adaptive algorithms such that the degree of node mobility contributes positively towards high performance and efficiency. (4) To further exploit available bandwidth and increase the efficiency of information access, we introduce the extensive application of *coding* in iFlow, starting from *erasure codes* (such as Tornado codes) used in information suppliers, complemented by *network coding* used in third-party peer nodes. Since both Tornado codes and network coding use efficient linear codes (e.g., the basic exclusive-or operation), the computational overhead introduced is minimal compared with the bandwidth efficiency gained with such coding processes.

In addition to analytical contributions supported by simulations, we have realized the architecture by implementing iFlow as a layer of middleware components with the Microsoft Component Object Model (COM) technology. The iFlow COM-based middleware exposes interfaces for the applications to invoke, as a wrapper around OS system calls. On the other hand, the application needs to implement event handlers to handle iFlow-specific events delivered by the middleware. Using standard Rapid Application Development tools such as Microsoft Visual Basic, customized event handlers are straightforward to implement and add to existing application functions.

The remainder of this paper is organized as follows. Sec. 2 presents the architecture and algorithms of iFlow, Sec. 3 presents the case of using network coding to further improve bandwidth efficiency. Sec. 4 presents simulation results. Sec. 5 presents a prototype implementation of the iFlow middleware framework to control mobile ad hoc applications. Finally, Sec. 6 and Sec. 7 compare iFlow with related work and conclude the paper.

2 iFlow: Algorithms and Analysis

The iFlow architecture is designed to serve as a middleware framework to support mobile ad hoc applications, whose components are distributed on different nodes in a mobile wireless ad hoc network. The iFlow architecture may be presented and analyzed from two different aspects. From the *horizontal* point of view, iFlow has included a set of fully-distributed algorithms for different application components residing on different nodes to interact with each other. Information flows from the suppliers and satisfies requests from consumers at rendezvous points, which usually are third-party peer nodes beyond the original suppliers. From the *vertical* point of view, iFlow is a middleware architecture designed to control the pattern of requesting and diffusing information in mobile ad hoc applications. In our implementation, such middleware components are implemented with Microsoft COM. Fig. 1 illustrates the iFlow architecture from both the horizontal and the vertical points of view.

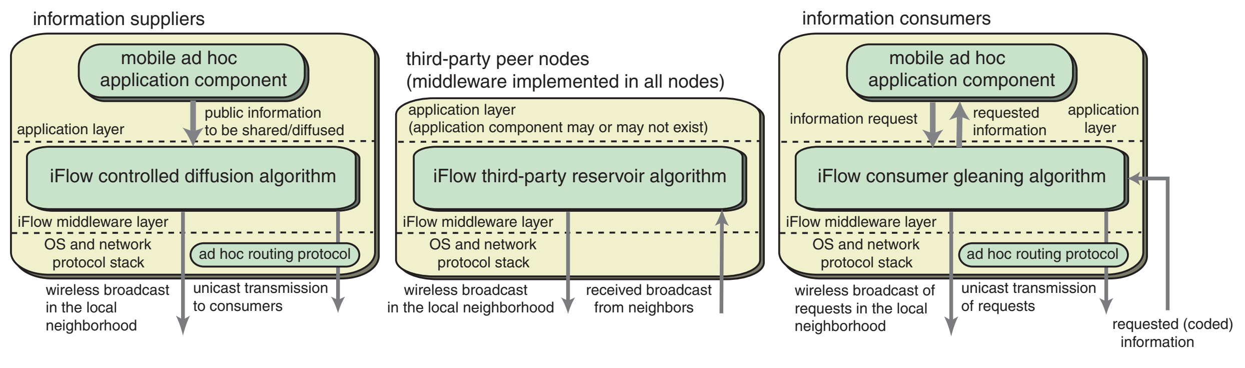
In this section, we present and analyze the design of the distributed algorithms in the iFlow architecture, including the aspects of information rendezvous and source erasure codes such as Tornado codes. For the purpose of simplifying analysis and presentation of algorithms, we first focus on the *single-supplier* case, where there is a unique information supplier in the network. Based on insights and conclusions derived from the single supplier case, we then extend our discussions to include the *multiple-supplier* case, where multiple suppliers exist when the controlled diffusion process is activated.

2.1 iFlow Overview: the Single-Supplier Case

For the remainder of the paper, we consider mobile ad hoc applications deployed in a wireless ad hoc network with m mobile nodes, some of which are suppliers or consumers of a certain piece of information, referred to as a *data item*.

In the single-supplier case, we consider the availability of a unique information supplier, who resides on one of the nodes in the network, and possesses complete information of α, the data item of interest. In the example of hybrid wireless networks, such a node may be the "gateway" node to the Internet via dual network interfaces.

We propose to activate a *controlled diffusion* process so that the data item may be diffused to a subset of third-party nodes referred to as *reservoir* nodes. Each of the reservoir nodes holds a certain subset of the data item, and they collectively achieve a certain degree of *saturation* of the data item in the entire wireless network. Once a certain level of saturation is reached, a consumer of the data item that moves around within the network may use the *gleaning process* to gather the requested segments of data from neighbors who are reservoir nodes of this particular item. We argue that, for data items of moderate and high popularity, it is more bandwidth efficient to use the strategy of controlled diffusion and gleaning, rather than directly sending individual requests to the suppliers. This general idea of taking advantage of third-party peer nodes in the process of information access is referred to



information suppliers
mobile ad hoc application component
public information to be shared/diffused
application layer
iFlow controlled diffusion algorithm
iFlow middleware layer
OS and network protocol stack
ad hoc routing protocol
wireless broadcast in the local neighborhood
unicast transmission to consumers
third-party peer nodes (middleware implemented in all nodes)
application layer (application component may or may not exist)
iFlow third-party reservoir algorithm
iFlow middleware layer
OS and network protocol stack
wireless broadcast in the local neighborhood
received broadcast from neighbors
information consumers
mobile ad hoc application component
information request
requested information
application layer
iFlow consumer gleaning algorithm
iFlow middleware layer
OS and network protocol stack
ad hoc routing protocol
wireless broadcast of requests in the local neighborhood
unicast transmission of requests
requested (coded) information

Figure 1: The iFlow architecture

as *information rendezvous*, since the requests for information are satisfied by peer nodes beyond the suppliers and consumers. The iFlow architecture is illustrated in Fig. 1.

2.2 The Controlled Diffusion Process

Assume that α is the data item of interests, and q_α is its *popularity*, *i.e.*, the percentage of nodes in the network that are consumers of α. When q_α is estimated to be beyond a threshold value, the supplier initiates the controlled diffusion process. The first step is to encode α into *coded symbols* using Tornado coding. The digital fountain approach proposed by Byers *et al.* [5] has included a detailed presentation of Tornado codes and their applications. We include a brief introduction as follows. The Tornado coding scheme generates kn coded symbols out of n uncoded data segments using the bitwise exclusive-or operation ($\oplus$). Coded symbols and uncoded segments are of equal sizes. The number k is referred to as the *stretch factor*. The coding scheme is designed such that a node that collects $n + \epsilon$ symbols is expected to be able to recover the uncoded data segments by applying substitutions and $\oplus$ operations, where $n + \epsilon$ is a number slightly larger than n. The ratio $1 + \epsilon/n$ is referred to as the *decoding inefficiency*. Well designed Tornado coding schemes may achieve decoding inefficiency that is less than 1.05. Compared to other erasure codes such as Reed-Solomon codes, Tornado codes are designed to be computationally efficient for both encoding and decoding processes.

The advantages of diffusing encoded symbols are two fold. First, it provides higher robustness for the system; second, it helps the gleaning process to complete in a timely fashion. We briefly illustrate the second point with an example below. Consider two alternative approaches of diffusing kn segments/symbols of a data item containing n data segments: (1) cyclic repetition, in which the n original data segments are diffused in order, and this procedure is repeated for k times, and (2) the n original data segments are first coded into kn encoded symbols using Tornado coding, which are then diffused into the network. For simplicity, assume that each diffused segment/symbol is held by a distinct node in the network. In the first alternative, to obtain the first segment, a consumer needs to contact one of the kn reservoir nodes; for the second through the last segments, the number of nodes that can provide A with useful segments decreases as follows: $(n-1)k, (n-2)k, \ldots, 2k, k$. Considering that k is usually a small number (e.g., 3), and the total number of nodes present in the network is usually large, the opportunity of encountering one of the k nodes is small, therefore the last few of the segments are exceedingly hard to collect. On the other hand, when erasure codes are used, the number of nodes that can provide A with useful (coded) symbols decreases in a much more graceful manner: $kn, kn-1, kn-2, \ldots, (k-1)n+1$. In this case, $(k-1)n+1$ nodes are able to provide the final symbol to A. In comparison, $(k-1)n+1$ is a much larger number than k, except for extreme cases, e.g., $n = 1$. Such extreme cases correspond to data items of very small sizes, which are not what Tornado coding targets for. The information rendezvous approach can still be applied, but with stricter requirements on data popularity and delay tolerance. In the remainder of this paper, we focus on relatively large data items for which Tornado coding can be applied to facilitate information gleaning; after all, disseminating larger data items consumes more bandwidth.

After coding the n segments in α into kn coded symbols, the supplier subsequently broadcasts these kn symbols in their original order, one after the other. The algorithm for the controlled diffusion process is shown in Table 1.

There exists a random pause between consecutive broadcasts (represented by a random variable T_x) conforming to the uniform distribution, the expected length of which, $E[T_x] = t_x$, is a parameter dependent on the degree of node mobility. Such a random pause is introduced to diversify the set of nodes covered by the diffusion pro-

Table 1: The controlled diffusion process

```
On information supplier: controlled diffusion algorithm

consider a data item $\alpha$ on the supplier:
  if popularity $q_\alpha$ reaches threshold value
    apply Tornado coding on $\alpha$ to generate symbols of $\alpha$
    for each coded symbol $x$
      pause for a random time period $T_x$
        (a random variable), s.t. $E[T_x] = t_s$
      broadcast $x$ to neighbors
    end
  end

On reservoir nodes: third-party reservoir algorithm

Upon receiving a diffused symbol $x$:
  if $x$ is a fresh symbol not previously received
    buffer $x$
    if predefined reach has not been exceeded
      compute relay probability $p$
      if probability test on $p$ succeeds
        broadcast $x$ to neighbors
      end
    end
  end
Upon receiving a probe from a consumer:
  if able to provide requested symbols
    advertise requested symbols in possession
    if confirmation received from consumer
      transfer advertised symbols
    end
  end
```

cess. Ideally, the broadcasting node is located within a relatively different neighborhood during each individual broadcast session. This way, with the same overhead of bandwidth, the diffused symbols are distributed onto a larger number of reservoir nodes within a larger geographical area, which helps the diffused information saturate the network more uniformly. Uniform saturation is desirable in iFlow, since it eliminates the existence of "information void" — a large network area without reservoir nodes holding diffused symbols, which may lead to prolonged gleaning time for consumers residing within the area.

Note that in this paper, we use the term "broadcast" to refer to local broadcasts within the immediate neighborhood of the transmitting node, which can be accomplished using *only a single transmission* due to the local broadcast nature of wireless transmissions using omni-directional antennas. Such an observation is sometimes referred to as the *wireless broadcast advantage* [6].

In the controlled diffusion process, each diffused symbol is accompanied by two control parameters: the *reach* of diffusion, which is the maximum number of wireless hops that a symbol may be relayed during the diffusion process by reservoir nodes, and the *relay probability* p, which is the probability that a reservoir node receiving a diffused symbol will re-broadcast the symbol. A reservoir node always buffers a fresh symbol received in the diffusion process, regardless of its decision on whether to relay that particular symbol. The reach and the relay probability are used to control the bandwidth consumption of diffusion, as well as the expected number of reservoir nodes that receive each symbol being diffused, which we refer to as the *coverage* of diffusion (denoted as c). In comparison, we define the degree of *saturation*, s, of α in a network as the ratio of the average number of symbols received by a node over the number of uncoded segments in α. For example, for a 300-node network and a diffusion process targeting 100 data segments of α, with a stretch factor of 3, 300 symbols are produced by Tornado codes. If we simply assume that, on average, 10 copies of each symbol have been buffered at reservoir nodes, 3000 symbols may then exist in the network. The average number of symbols received by each node is, therefore, 10. In this example, the degree of saturation s is $10/100 = 0.1$. The extreme case is when $s = 1$, where no gleaning process is required — all nodes may reconstruct original copies of α.

The coverage of diffusion is an increasing function of both the reach and the relay probability p. However, for the same coverage, we have the choice of using a smaller reach with larger relay probabilities, and the alternative of using a larger reach with smaller relay probabilities, as illustrated in Fig. 2. The latter approach is more desirable for two reasons. First, it introduces less overlap among different broadcasts, and is therefore more bandwidth efficient (*i.e.,* for the same bandwidth more reservoir nodes are reached). Second, it spreads symbols over a larger range of geographical area in a sparser fashion, which, aided by node mobility, is helpful to achieve uniform saturation more promptly.

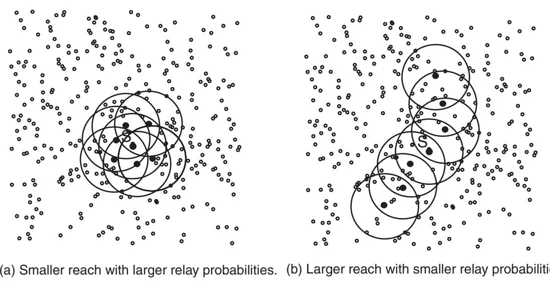

(a) Smaller reach with larger relay probabilities.
(b) Larger reach with smaller relay probabilities.
S

Figure 2: Different choices of reach and relay probability in the controlled information diffusion process

Therefore, in the ideal scenario, we wish to employ a few of the neighboring nodes leading to different directions of the supplier to re-broadcast the diffused symbol, and each re-broadcasting reservoir node employs one successive neighbor to further relay the symbol. From the point of view of the overall diffusion process, we may observe a few non-overlapping routes extending from the supplier towards different directions, while nodes within one hop range of the routes are covered by the diffusion, and subsequently become reservoir nodes that buffer the diffused symbol.

To approximate this ideal scenario in the iFlow algorithms, we need to select nodes from the neighborhood of the supplier that are far apart from one another, so that their coverage areas overlap as slightly as possible. This objective may be achieved, if — rather than allowing each of the neighbors of the supplier to make a random and independent decision on relaying — we allow a neighbor to relay a diffused symbol if and only if it has not heard a neighbor doing exactly the same. The result of such a modified algorithm will be that, *two or three neighbors* (who are beyond the transmission range of each other) are expected to re-broadcast the symbol, and the other neighbors remain "silent".

In addition, we need to guarantee that only one neighbor of each broadcasting node further relays the symbol being diffused, if it is not beyond the predefined reach from the supplier. This may be achieved if the relay probability is set to be inversely proportional to the number of new nodes that receive the symbol during a broadcast. As shown in Fig. 3, this number can be estimated as $\rho S_\delta/(\pi R^2)$, where ρ is the average node degree in the network, and S_δ is the area covered by the downstream broadcaster B, but not by the upstream broadcaster A, which corresponds to the shaded area in the figure. The distance between A and B can be estimated as $\int_0^R r(2\pi r)\mathrm{d}r/\int_0^R 2\pi r \mathrm{d}r = 2R/3$, which is the expected distance between an arbitrary pair of neighbors. It then follows that the estimate on the number of new nodes being covered can be computed as 0.42ρ.

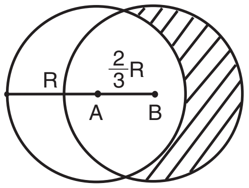


Figure 3: Effective coverage of a successive broadcast

2.3 The Information Gleaning Process

Table 2 presents the algorithm for the information gleaning process on consumers. In the gleaning process, a consumer that generates a request for data item α probes its neighbors for symbols of α as it moves around. Similar to the diffusion process, the requesting node waits for a random time period (represented by a random variable T_c) between two consecutive probes, such that its

Table 2: The information gleaning process

On information consumers: consumer gleaning algorithm

```
while not sufficient symbols to recover $\alpha$ do
    pause for random time period $T_c$
        (a random variable), $E[T_c] = \Delta t$
    broadcast a probing message to neighbors
    pause for time period $t'_c$
    if advertisements received
        confirm with node that can provide max # of symbols
        receive symbols from confirmed neighbor
    end
end
```

set of neighbors may experience some variations during that period. In our analysis, we set the expected length of this waiting period as $E[T_c] = \Delta t$, where Δt is the expected time that a node encounters one new neighbor in its neighborhood. The probe message contains a description of the symbols that the consumer already has for the requested data item α. Upon receiving a request, a neighboring node advertises the symbols it is able to provide for α, if such symbols exist. The consumer confirms with the neighbor that can provide the maximum number of symbols, after which the transfer begins. If the symbols collected after the transfer are still not sufficient to recover α, or if no advertisement is received, the consumer waits to probe again after a subsequent random time period.

To accommodate the interests of applications with stricter deadlines, we include a *panic mode* in the gleaning process. The mobile ad hoc application has the option of specifying a delay requirement for a particular request. In this case, the consumer terminates the gleaning process when the gleaning time reaches the specified delay, and enters the panic mode. In the panic mode, the consumer contacts the supplier immediately with a list of symbols it has collected. The supplier may then deliver complementary symbols to the consumer directly using a separate multi-hop unicast transmission, until the consumer has sufficient symbols to recover α.

However, since multi-hop unicast transfers incur higher bandwidth costs (which is against iFlow's objective of improving bandwidth efficiency), the panic mode should only be considered as a last resort that provides a hard delay guarantee for our rendezvous algorithms, which are inherently probabilistic. Naturally, we would like to control the diffusion process so that the network is saturated to a certain degree, where the expected gleaning time is less than the application-specified delay.

From the perspective of bandwidth efficiency, the degree of saturation is determined by the actual bandwidth consumption of the diffusion process (*i.e.,* the more band-

Table 3: List of mathematical notations

parameter	definition
m	total number of nodes within the network
ρ	average node degree of the network
Δt	expected time it takes for a node to encounter a new neighbor
α	data item of interest
n	number of uncoded data segments in the data item
k	stretch factor of Tornado coding
q_α	popularity of a data item α, *i.e.*, the percentage of nodes that eventually generate a request for α as consumers
c	coverage of diffusion, *i.e.*, expected number of reservoir nodes that hold each symbol after diffusion
s	saturation, *i.e.*, average number of symbols a node receives in diffusion over the number of uncoded data segments in the data item

width used, the higher the saturation). As a minimum requirement, the bandwidth consumption incurred by the information rendezvous process (including both diffusion and gleaning) should be (much) *less* compared with the approach of making separate unicast requests from consumers directly to the supplier. We use such a guideline as one of the design requirements of the iFlow algorithms.

We proceed to analyze critical trade-offs and relationships between two pairs of parameters: (1) the relationship between the degree of saturation and gleaning time; and (2) the relationship between bandwidth consumption and the degree of saturation, especially when compared with the all-unicast approach without using iFlow. For clarity, we list the mathematical notations of several key parameters in Table 3.

2.4 Bandwidth Consumption vs. Saturation

For this part of the analysis, we assume that the sizes of the coded symbols are much larger than the sizes of control messages in iFlow or underlying network protocols (such as routing). Therefore, we focus on the bandwidth consumption incurred when transferring the symbols across the network. More specifically, we calculate the times that the symbols are being transferred. Thanks to the wireless broadcast advantage, each local unicast or broadcast of a particular symbol counts as *a single transmission* (*i.e.,* the bandwidth consumption is 1, with a unit of symbols · hops). Further, it is straightforward to observe that, the total number of symbols replicated in the controlled diffusion process is $c \cdot kn$. Therefore, the degree of saturation, s, may be estimated from the coverage of diffusion: $s = c \cdot kn/(mn) = ck/m$.

In order to estimate the relationship between bandwidth consumption and the degree of saturation s, we first seek to examine the relationship between bandwidth consumption and the coverage of diffusion. Consider a particular symbol being diffused, x. Let b_x be the bandwidth consumption of diffusing x, *i.e.*, the number of times that x is broadcasted in the controlled diffusion process. Let c_x be the coverage of x, *i.e.*, the number of nodes that have x at the end of the diffusion process.

Recall that we have estimated the number of new nodes being covered by a re-broadcast as 0.42ρ. It follows that $c_x = (b_x - 1)0.42\rho + \rho = 0.42b_x\rho + 0.58\rho$. If the total bandwidth consumption of diffusing all symbols of α is $b = b_x \cdot kn$, we have the following relationship between the total bandwidth consumption and the coverage of diffusion: $c = 0.42b\rho/(kn) + 0.58\rho$. Substituting the derived c in $s = ck/m$, we then have

$$s = \frac{\rho}{m}\left(\frac{0.42b}{n} + 0.58k\right).$$

The above estimate suggests that, in order to achieve a certain degree of saturation, the total bandwidth consumption of diffusion should be proportional to the number of symbols to be diffused, as well as to the normalized size of the network, *i.e.*, the area of deployment of the network divided by the disk area within the communication range of a node.

2.5 Saturation vs. Gleaning Time

We now consider the case where the controlled diffusion process of the data item α has completed, and over a certain period of time, the diffused symbols have mingled uniformly in the network, due to the random trajectories of mobile nodes. Under such an assumption, we estimate the expected gleaning time of a consumer, should it now generates a request for α. We first consider a new consumer that has just joined the network, and then modify our estimate for a consumer that was previously in the network.

To facilitate our analysis, we assume that the decoding inefficiency in Tornado coding is 1, *i.e.*, exactly n symbols is required to recover α. We further assume that the requesting consumer can obtain any useful symbols from the surrounding nodes, once they become neighbors to one another. In order to glean n unique symbols from its neighborhood, the number of symbols the requesting consumer is expected to encounter is:

$$\sum_{i=0}^{n-1} \frac{c(kn)}{c(kn) - ci} = \sum_{i=0}^{n-1} \frac{kn}{kn - i} \approx \frac{kn}{k - \frac{1}{2}} \qquad (1)$$

Therefore, the number of nodes the consumer is expected to encounter can be estimated as:

$$\frac{1}{ns}\frac{kn}{k-\frac{1}{2}} = \frac{k}{(k-\frac{1}{2})s}.$$

It follows that the expected gleaning time, t_g, is

$$t_g \approx \max\left\{(\frac{k}{(k-\frac{1}{2})s} - \rho)\Delta t, 0\right\} + t_{tr}.$$

where t_{tr} is the net transfer time of the symbols. Note that the above result is an *overestimate* in that it does not take into account the fact that symbols found at the same reservoir node are distinct; it is an *underestimate* in that the $\approx$ in Eq. (1) is actually $\geq$, and in that it ignores the time overhead it takes for the consumer to set up connections with its neighbors. However, the result should still provide insights on how the gleaning time may be related to the degree of saturation. It shows that, when saturation is beyond a certain level such that a node is able to collect sufficient symbols from one set of neighbors, then the gleaning time is dominated by the transfer time of the symbols. However, if saturation is below such a level, then the consumer needs to spend time in both receiving the symbols and in waiting to meet new neighbors. Since the second term is on the scale of physical movement, it usually dominates the gleaning time. Naturally, the existence of such a dominating factor depends on whether the number of nodes that the consumer needs to encounter is larger than the size of its local neighborhood. On the other hand, the expected number of nodes to be encountered by a consumer is inversely proportional to the degree of saturation.

If the consumer is not new and has been previously present in the network, the expected gleaning time is smaller, since the consumer may very well be a reservoir node itself, and may have accumulated a number of symbols during the diffusion process before its request arrives. We can therefore adjust the above estimate to

$$t_g \approx \max\left\{(\frac{k}{(k-\frac{1}{2})s} - \rho - 1)\Delta t, 0\right\} + t_{tr}.$$

The relationship between t_g and saturation s is similar to the case where the consumer is a new node in the network.

2.6 iFlow vs. Unicast

The total bandwidth consumption of accessing a data item α using iFlow is approximately $b + n(mq_\alpha)$, where the first term represents the bandwidth consumption of diffusion, and the second term is the bandwidth consumption of gleaning. In comparison, the total bandwidth consumption of accessing α using the all-unicast approach may be estimated as $nh(mq_\alpha)$, where h is the expected number of hops between a pair of arbitrary nodes within the network. Therefore, in order to satisfy the minimum design requirement that iFlow should be more bandwidth efficient than the plain unicast approach, we need to satisfy $b < nmq_\alpha(h-1)$. This upper bound may be denoted as b_u.

On the other hand, the delay requirement of the application defines a lower bound on the coverage of diffusion. Since coverage is controlled by bandwidth consumption, we have a corresponding lower bound for b, b_l. For any value of b such that $b_l \leq b \leq b_u$, the delay requirement is expected to be satisfied, while the bandwidth efficiency is expected to be better than the all-unicast approach. The supplier has the choice of trading delay for less bandwidth consumption, by choosing b that is closer to b_l; or, alternatively, trading bandwidth for a smaller delay, by choosing b that is closer to b_u. However, note that the average memory overhead on iFlow nodes is proportional to saturation. Therefore, the former choice is usually more preferable, since it leads to lower saturation and hence lower memory overhead.

Our analysis shows that the lower bound b_l is independent of the popularity q_α, while the upper bound b_u is proportional to q_α. Therefore for the range $[b_l, b_u]$ to be non-empty, we have a lower bound requirement on q_α. This confirms the intuition that iFlow may be more bandwidth efficient when more consumers request a popular data item. However, this requirement is rather loose in many scenarios; that is, contrary to common intuitions, the iFlow algorithms may *both* achieve bandwidth efficiency *and* meet the delay requirement for data items that are not popular, depending on the network characteristics. We proceed to show such an example.

Consider a moderately dense network with total number of nodes $m = 300$, average node degree $\rho \approx 17$, and average number of distance between a pair of nodes $h \approx 4$ hops. Assume that the data item of interest, α, has $n = 100$ symbols and Tornado codes with a stretch factor $k = 2$ is used. We now consider the extreme case where $mq_\alpha = 1$, *i.e.*, only one node in the network has a request for α. We show that if we choose the reach of diffusion to be just 1, then both the lower bound requirement and the upper bound requirement can be satisfied. When the reach is one, $b = kn = 2n < 3n = n(mq_\alpha)(h-1)$, therefore the upper bound requirement due to bandwidth efficiency is satisfied. On the other hand, the saturation $s = ck/m = (\rho+1)k/m = 0.12$; therefore the first term in our estimate of the gleaning time, $\max((\frac{k}{(k-\frac{1}{2})s} - \rho - 1)\Delta t, 0) = 0$ since $\frac{k}{(k-\frac{1}{2})s} \approx 11 < 18 = \rho + 1$. This means that the gleaning time is dominated by the symbol transfer time, and should therefore satisfy any reasonable delay requirement.

Therefore we conclude that there exist abundant opportunities for improving bandwidth efficiency using the information rendezvous algorithms proposed in iFlow, even with non-popular data items and relatively strict application requirements on delay, given that node mobility is present.

2.7 iFlow: The Multiple-Supplier Case

We now briefly consider the case where complete copies of the data item of interest, α, exist on *multiple suppliers* before the diffusion process. Such a scenario may exist if, for example, by the time that the popularity of a data item popularity is estimated to be sufficiently high to initiate diffusion, a few nodes have already acquired α through unicast.

The proposed iFlow algorithms adapt to the multiple-supplier case naturally without modifications. However, to maintain the same level of bandwidth consumption, each supplier may choose to use a small reach as its control parameter. Such a case enjoys two advantages over the single-supplier case. First, since the reach parameters used are small, in many cases just one, there exists less overlap among different broadcasts; the same amount of bandwidth consumption may now lead to a larger coverage. Second, reservoir nodes covered by diffusion has higher diversity in terms of geographical distribution; this observation, again, assists the reservoir nodes to mingle with the regular nodes more promptly and rapidly. We conclude that iFlow performs better in the multiple-supplier case, which conforms to the intuition.

3 Improving iFlow: Network Coding

The ultimate objective in the design of the iFlow architecture is to facilitate bandwidth-efficient information access. In this section, we present the important concepts of network coding [7] in wireline networks, propose to apply network coding on reservoir nodes to further improve bandwidth efficiency in iFlow.

Network coding is a theoretical strategy that has been proposed in the area of information theory [7, 8, 9], the objective of which is to increase end-to-end throughput in multicast sessions in wireline networks, which is subtly different from our goal of improving bandwidth efficiency (*i.e.,* delivering more useful data with limited channel capacity) in wireless networks. With network coding, bits of data to be delivered are not merely treated as "atoms" that may *only* be replicated and forwarded at intermediate nodes; rather, data may be *coded* before being forwarded further.

Similar to Tornado codes on information suppliers, bitwise exclusive-or ($\oplus$) can be employed as the basic coding operation for its computational efficiency. Different from Tornado codes, network coding may be used not only at the source node, but also at intermediate nodes; it may code not only information of the same data item, but independent information from different data items as well. Coded data may be decoded by a downstream or destination node, based on its knowledge of the coding strategy.

3.1 Network Coding: a Review of Concepts

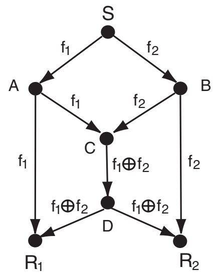


Figure 4: The effectiveness of network coding in wireline networks

We briefly review the concepts of network coding with an example shown in Fig. 4 [7]. The example shows how the session throughput of an 1-to-2 multicast session may be improved in wireline networks. In the figure, f_1 and f_2 represent two independent information flows originating from the source S. Node C transmits the coded flow $f_1 \oplus f_2$ along the "bottleneck" link CD to node D, which then forwards the coded flow to both destinations R_1 and R_2. R_1 and R_2 can recover $\{f_1, f_2\}$ from $\{f_1, f_1 \oplus f_2\}$ and $\{f_2, f_1 \oplus f_2\}$, respectively. The session achieves a throughput of $2C$, assuming each link has capacity C. Without network coding, it can be verified that the achievable throughput is only $3C/2$.

3.2 Network Coding in Wireless Networks

While network coding may increase throughput of multicast sessions in wireline networks, there exist fundamental differences between wireless and wireline networks. With respect to bottleneck formation, wireline networks are *link-centric*, while wireless networks are *node-centric*. In wireline networks, a single link can form a bottleneck due to limited link capacity, while a forwarding node is usually not a concern; in wireless networks, a single node is sufficient to form a bottleneck, since virtual links sharing the same node may not transmit concurrently, [10]. We are not aware of any previous work that has studied the problem of applying network coding in wireless networks, especially when the objective is to improve bandwidth efficiency.

We believe that network coding may still be applied effectively in wireless networks, in order to improve bandwidth efficiency. Fig. 5 shows an example in which network coding helps to reduce the total bandwidth consumption in two 1-to-2 wireless multicast sessions, where S_1 and S_2 are the sources and R_1 and R_2 are the destinations. Without network coding, four transmissions are required to deliver a packet from each source

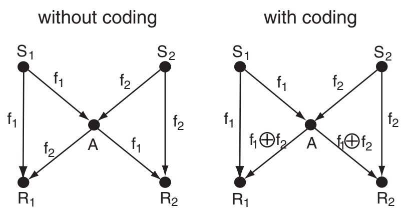


Figure 5: Network coding in wireless networks: an example

to each destination: $S_1 \xrightarrow{f_1} (A, R_1)$, $S_2 \xrightarrow{f_2} (A, R_2)$, $A \xrightarrow{f_2} R_1$ and $A \xrightarrow{f_1} R_2$. With network coding, the "bottleneck" node A transfers the coded flow $f_1 \oplus f_2$ to both receivers at once, only three transmissions are necessary: $S_1 \xrightarrow{f_1} (A, R_1)$, $S_2 \xrightarrow{f_2} (A, R_2)$, and $A \xrightarrow{f_1 \oplus f_2} (R_1, R_2)$. Therefore overall bandwidth efficiency is improved by $1/3$.

3.3 Network Coding: Improving iFlow

We proceed to discuss the details of improving the bandwidth efficiency in the iFlow architecture by applying network coding on reservoir nodes. A major feature of iFlow as opposed to a generic wireless ad hoc network is that, iFlow applies Tornado codes in information suppliers before the diffusion process. We observe that, both network coding and Tornado codes employ the bitwise exclusive-or operation, and therefore can act in concert with each other naturally within the iFlow framework, as we will show shortly. In iFlow, a reservoir node has the opportunity to apply network coding when it is relaying multiple symbols during diffusion, supplying symbols for multiple consumers during gleaning, or a mixture of the two. In these cases, the reservoir node forms a "bottleneck" node that may apply take advantage of the broadcast nature of wireless transmission by broadcasting a recoded symbol that may potentially benefit more neighbors. Due to limit of space, we show an example of applying network coding in information diffusion only.

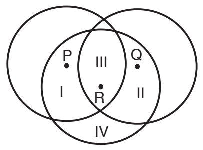


Figure 6: Network coding in information diffusion: an example

Assume that during diffusion, a node R receives a symbol a from node P and a symbol b from node Q, respectively, as shown in Fig. 6. If R plans to further relay a symbol in the diffusion process, it may perform one of the following tasks: (1) re-broadcast a or b directly; or (2) re-broadcast the recoded symbol $a \oplus b$, in accordance with the concept of network coding. We argue that, re-broadcasting $a \oplus b$ is a better choice, since it can potentially benefit more neighboring nodes of R.

Table 4: Network coding in information diffusion: a comparison

	current symbols	broadcast a at R	broadcast b at R	broadcast $a \oplus b$ at R
I	a	a	a, b	a, b
II	b	a, b	b	a, b
III	a, b	a, b	a, b	a, b
IV	ϕ	a	b	$a \oplus b$

Table 4 shows the set of symbols acquired by nodes within different areas, before and after R's re-broadcasting of a, b, or $a \oplus b$. The choice at R does not affect nodes within area III, since they have already received both a and b before R's re-broadcasting. If R re-broadcasts a, then nodes within area II and IV will benefit, since a is new to them; nodes within area I will not benefit since they have received a already. Similar arguments apply if R re-broadcasts b. In comparison, if R re-broadcasts the recoded symbol $a \oplus b$ based on network coding, then nodes within both area I and area II will benefit and obtain complete information on both symbols a and b.

It is not obvious, though, whether nodes in area IV also benefit from the coded transmission from R. Naturally, if nodes within area IV received a or b previously from other nodes, or will receive a or b later on, the value of $a \oplus b$ can then be realized. Furthermore, since a and b are being diffused concurrently within nearby network areas, it is highly probable that they are coded symbols of the same data item α. In that case, the recoded symbol $a \oplus b$ has its own value without being recovered to a and b first. The reason is that, same as a or b, $a \oplus b$ is just another coded symbol obtained by applying the $\oplus$ operations over certain data segments in α, and therefore can be transmitted within the network and be used as input to the Tornado decoding procedure as well.

To further illustrate such harmony between Tornado codes and network coding, consider the following example. For clarity, assume that there are only three data segments in α, 1, 2 and 3. Further, assume that a Tornado coding scheme with a stretch factor of 2 is used, with 1, 2, 3, $1 \oplus 2$, $2 \oplus 3$ and $1 \oplus 2 \oplus 3$ being the coded symbols. We can verify that any combination of three distinct symbols is sufficient to recover the data item with probability 0.8, and any combination of four distinct symbols is sufficient for the recovery with probability 1. Therefore such a coding scheme has the decoding inefficiency of $(3 \times 0.8 + 4 \times 0.2)/3 = 1.07$. If symbol a in the previous example is, say, $1 \oplus 2$, and symbol b is $2 \oplus 3$, then the recoded symbol resulting from network coding, $a \oplus b$, is precisely $1 \oplus 3$, which intuitively also contains useful information for the purpose of recovering α. We then come to the conclusion that re-broadcasting $a \oplus b$ at R is more bandwidth efficient than re-broadcasting a or b with a high probability. This example shows the advantage of network coding in the diffusion process.

To conclude, though iFlow is a complete architecture and set of algorithms to enable bandwidth-efficient information access, further improvements on bandwidth efficiency can be realized if network coding is applied appropriately in the information diffusion process.

4 iFlow: Simulation

For the purpose of evaluating the performance of various aspects of the iFlow architecture and the feasibility of its deployment, we performed simulations of the iFlow algorithms in C++, followed by a prototype implementation of iFlow as a COM-based middleware layer (discussions of which are postponed to Sec. 5).

4.1 Bandwidth Efficiency

The primary design objective of the iFlow architecture is to enable bandwidth-efficient information access within mobile ad hoc applications, given a certain degree of user mobility and information popularity. Our analysis in Sec. 2 has shown that, there exist abundant opportunities that iFlow can achieve this goal. This observation is verified by forthcoming empirical results.

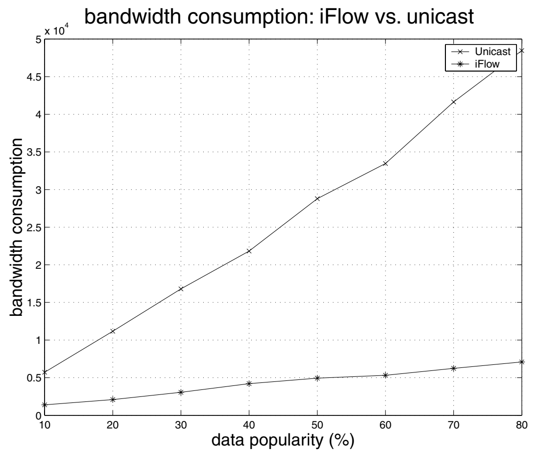


Figure 7: Bandwidth efficiency under different degrees of data popularity: iFlow vs. unicast

Fig. 7 shows a comparative study between the iFlow algorithms and the plain all-unicast approach, with respect to bandwidth consumption of disseminating the same data item. In the comparison, we simulate with the same network environment and the same simulation framework. The simulation parameters are established as follows. (1) The area of deploying the mobile ad hoc network is 500m $\times$ 500m. (2) The total number of users m is 300. (3) The communication range of each node R is 70m. (4) The number of uncoded data segments in the data item $n = 50$. (5) With respect to Tornado codes, the stretch factor of Tornado codes k is 3, and the decoding inefficiency is 1.0. (6) The nodes move around the deployment area using the random way-point mobility model, with the pause time as 0 seconds, and the velocity as 9 m/s. (7) The expected waiting time $E[T_x] = t_s$ between consecutive broadcasts at supplier is 2 seconds; while the expected waiting time $E[T_c]$ between consecutive probes at consumers is 3 seconds.

With respect to the bandwidth consumption in iFlow, we take the bandwidth consumed in both diffusion and gleaning processes into account. For the all-unicast approach, we compute the total length of routes between the supplier and the consumers. In cases that the supplier and the consumer are separated within different partitions of the network, we wait until they move into the same partition.

Simulation results in Fig. 7 show that, the total bandwidth consumption of the all-unicast approach grows linearly with data popularity, and is always larger than the bandwidth consumption of the iFlow system. The difference becomes more significant as popularity grows, up to an order of magnitude. The reason behind this observation is that, for the all-unicast approach, bandwidth consumption is proportional to data popularity; while for iFlow, the total bandwidth consumption is a summation over two terms: diffusion bandwidth consumption and gleaning bandwidth consumption. The first term remains at a constant level regardless of data popularity, only the second term grows linearly with popularity. Since information gleaning consists of one-hop transmissions only, the second term is much smaller than the total bandwidth consumption of multi-hop unicast transmissions. Therefore, the bandwidth consumption of iFlow grows much slower than that of unicast.

4.2 Diffusion Bandwidth Consumption vs Saturation

In Sec. 2, we have analyzed the relationship between bandwidth consumption and expected gleaning time by first deriving the relationship between diffusion bandwidth consumption and saturation, and then deriving the relationship between saturation and gleaning time. Recall that the theoretical result of our analysis on the first relationship is $s = \rho \cdot (0.42b/n + 0.58k)/m$. Below we compare results from our simulations to such a theoretical estimate.

We have performed two sets of simulations, one of which has a deployment size of 500m $\times$ 500m, while

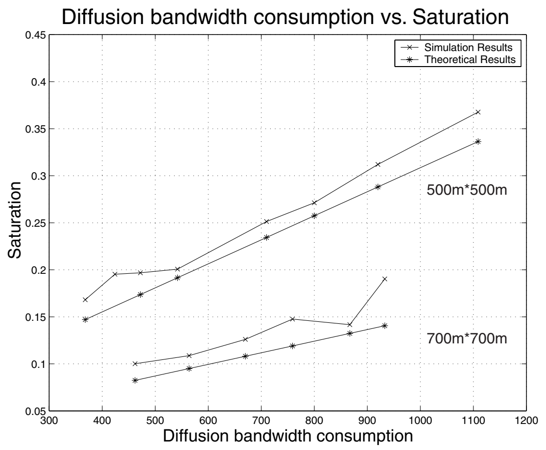


Figure 8: The relationship of diffusion bandwidth consumption and saturation

the other has a deployment size of 700m $\times$ 700m. We vary the reach r from 2 to 4, and vary the relay probability p from 0.1 to 0.2 and from 0.3 to 0.4 for the 500m $\times$ 500m case and the 700m $\times$ 700m case, respectively. Other parameters remain unchanged as in Sec. 4.1. We choose different ranges of relay probabilities for the two cases so that saturation within two networks under the same bandwidth consumption can be compared.

Both simulation and previous analytical results are shown in Fig. 8. It shows that overall, results from our simulations agree with our theoretical analysis. In particular, for the same amount of bandwidth consumption, the degree of saturation in the 500m $\times$ 500m network is approximately twice as high as that in the 700m $\times$ 700m network. This confirms our previous observation that, in order to achieve the same level of saturation, bandwidth consumed in diffusion should be proportional to the normalized network size.

4.3 The Role of Relay Probability

The relay probability p plays an important role in the iFlow algorithms. It is used to arbitrate the trade-off between bandwidth efficiency and the delay of satisfying requests from the application. With the similar network setup[2] carried forward from the previous experiments, Fig. 9 has shown how bandwidth consumption and gleaning time vary as the relay probability varies. As we may observe, as p grows, bandwidth consumption increases and gleaning time decreases. However, when the value of p reaches a certain level (0.5 in this case), further increases of p elevates the amount of bandwidth consumption without significant effects on the gleaning time. This suggests that the combination of a smaller relay probability with a larger reach is a better choice, rather than the combination of a larger relay probability with a smaller reach. This confirms the corresponding statements in our analysis of the algorithms (Sec. 2). As previously explained, this is due to the fact that the latter choice introduces more overlap among the broadcasts during the diffusion, and is therefore less cost-effective.

[2]It is identical except that the reach of diffusion is 5, the user velocity is 2 m/s, and $E(T_x) = 1$ second.

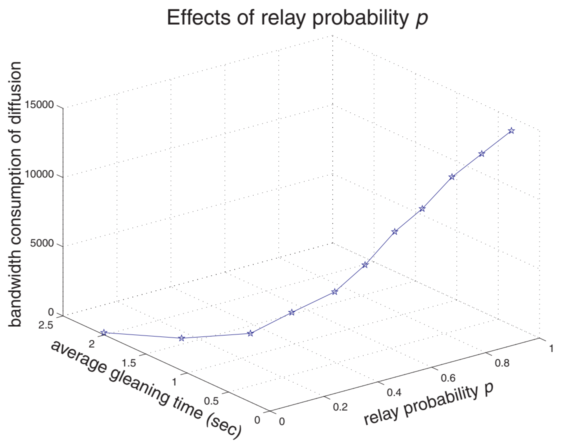


Figure 9: Effects of varying the relay probability p in the controlled diffusion process

4.4 Comparison of Delay Latency

Finally, we perform simulations to compare the delay latency of performing the same data dissemination task using unicast, iFlow, and iFlow with Tornado coding replaced by cyclic repetition. The results are presented in Table 5, using the same network simulation parameters as the first experiment[3]. First, we observe that, by introducing Tornado coding on information suppliers, the latency of satisfying requests experienced by the mobile ad hoc application is dramatically reduced. The justification is that, as we have explained, erasure codes such as Tornado coding gracefully solves the problem of obtaining the last few segments of data, from which the scheme of cyclic repetition suffers. Second, we notice that although delay of unicast is smaller than that of iFlow, they are on the same magnitude. This is due to the poor data availability of the unicast approach, since a consumer node may be partitioned from the only supplier node when its request arrives. In that case, the consumer has to wait until it moves into the same network partition as the supplier. In denser networks where partition rarely occurs, the performance of unicast should be better, in terms of delay latency.

The set of simulation results presented in this section has verified the insights we have obtained from the analysis of iFlow, and has provided solid proof that iFlow

[3]With the exception that the deployment size of the network is 600m $\times$ 600m, the transmission range R = 50 m, the reach is 2, and the relay probability $p = 0.1$.

Table 5: Latency: Tornado coding vs. cyclic repetition

user velocity (m/s)	3	5	7	9
Delay: Unicast(s)	2.3	2.0	1.4	1.5
Delay: iFlow (s)	4.3	4.0	3.4	3.6
Delay: cyclic repetition (s)	11.8	8.8	7.4	6.0

is able to consistently outperform the alternatives without iFlow, with respect to both bandwidth efficiency and latency.

5 iFlow: Implementation

Beyond simulation results previously shown, we have implemented the iFlow architecture as a middleware layer based on the Microsoft Component Object Model (COM), supporting COM-aware applications on the Windows platform. The iFlow middleware layer exposes COM interfaces for the applications to invoke, and delivers COM events to applications, so that the application may implement customized event handlers to handle iFlow events. For basic communication between neighboring nodes, the iFlow middleware utilizes the Windows Sockets library available on Windows. In order to realize a wireless ad hoc network, we have further used the *ad hoc mode* of IEEE 802.11b wireless LAN as the MAC and physical substrate in our testbed, without fixed access points. We employ Windows-based laptop computers for our testbed, mainly with Windows 2000 and Windows XP as operating systems.

The advantage of using COM as our middleware substrate is to support the ubiquitous availability of Windows applications, including those on Pocket PC based platforms. The entire set of iFlow algorithms are built as a Dynamic Link Library (DLL) in Windows, to be readily loaded by any COM-aware applications. The availability of Rapid Application Development tools such as Microsoft Visual Basic greatly facilitates making the necessary modifications to existing applications to take advantage of iFlow, if there are interests for iFlow events to be handled. Using Visual Basic, we have implemented a prototype application to employ the services of the iFlow middleware, with graphical user interfaces for the purpose of illustrating the status of iFlow in action.

For the purpose of showing global properties of the entire wireless ad hoc network, we have resorted to the creation of an *omniscient observer*, which, obviously, may not be available in real-world scenarios. However, the availability of the omniscient observer in our implementations greatly facilitates the monitoring of instantaneous network and node states, such as total number of nodes, node locations, roles of different nodes (consumers, reservoir nodes or information suppliers), as well as the number of consumers that have already reconstructed the requested data item. Due to the unavailability of GPS devices and the fact that most of our tests are conducted indoors, we have simulated the node locations on the omniscient observer, and then delivered the node locations to participating nodes on a periodic basis. The other advantage of such simulated locations is that the degree of node mobility may be easily varied, leading to more deterministic studies of the effects of mobility[4].

The implementation of iFlow middleware layer is designed to be multi-threaded in order to accommodate multiple incoming requests and ongoing connections. There exists three types of threads: (1) the main thread to handle COM-based method invocations; (2) the server thread to listen on the well-known port and create TCP-based stream sockets; and (3) the "worker" threads to process incoming requests. We have accomplished challenging tasks of COM-based multi-threaded programming, where the COM interface pointer needs to be marshaled per thread.

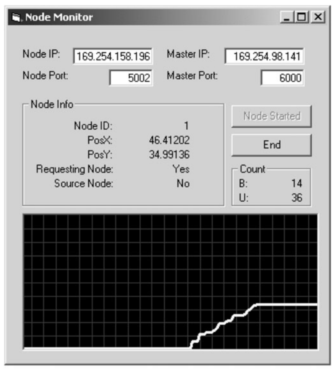
With respect to the exposed interfaces for the application to invoke, and the delivered COM events for the application to handle, we briefly show precise definitions of example methods and events in the iFlow interface, defined in the Microsoft Interface Definition Language (IDL):

```
interface iFlowWrapper : IDispatch
{
  typedef struct tagPacket { ... } Packet;

  \\ methods
  \\ send a packet for local broadcast through iFlow
  HRESULT BroadcastPacket([in] Packet* Msg,
                [out,retval] int* pCount);
  \\ initialize the iFlow middleware algorithms
  HRESULT StartServer([in] BSTR srcAddress,[in] int srcPort
                [in] int IsMaster);
  \\ parse incoming message
  HRESULT ParseMessage([in] int pPacket,[in] UINT pSocket,
                [out, retval] int* pVal);
  ...
};

\\ events
dispinterface _iFlowWrapperEvents
{
  HRESULT OnDiffuse();
  HRESULT OnRecvBroadcast([in] short PacketNumber);
  HRESULT OnReBroadcast([in] short PacketNumber,
                    [in] short reach);
  ...
};
```

To demonstrate the results of such an implementation, we have deployed iFlow in a 30-node network with one data item of interest. Fig. 10 shows graphical user interfaces on a regular node and the omniscient observer

[4]Since node mobility is simulated, it may be more appropriate to refer to our testbed as an *emulation* testbed rather than implementation. However, with modest modifications, we believe that the implementation of iFlow may still be readily be deployed in real-world scenarios.

during a diffusion session. The results have mostly agreed with our simulations, which we choose not to show repeatedly. For completeness, Fig. 11(a) shows the per-node progress recorded within iFlow on each of the nodes (10 nodes are shown as examples), and Fig. 11(b) shows the number of consumers that have reconstructed the requested data item over time. To conclude, even though the implementation is a proof-of-concept prototype, it has demonstrated the feasibility of real-world deployment of iFlow on wireless devices; we have especially shown ready support for COM-based Windows applications that are ubiquitously available.

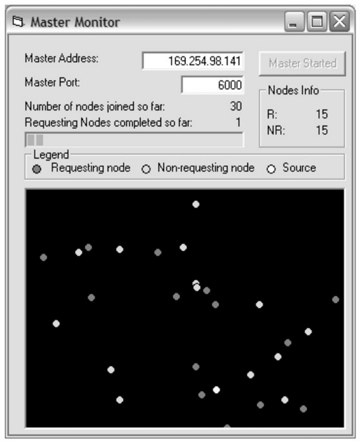

Node Monitor

Node IP: 169.254.158.196 Master IP: 169.254.98.141
Node Port: 5002 Master Port: 6000

Node Info
Node ID: 1
PosX: 46.41202
PosY: 34.99136
Requesting Node: Yes
Source Node: No

Count
B: 14
U: 36

(a) The monitoring graphical user interface of the *iFlow* middleware architecture on regular nodes, showing the number of symbols accumulated so far. (Windows 2000)


Master Monitor

Master Address: 169.254.98.141
Master Port: 6000
Number of nodes joined so far: 30
Requesting Nodes completed so far: 1

Nodes Info
R: 15
NR: 15

(b) The monitoring graphical user interface of the *iFlow* architecture on the "omniscient observer", showing a global view of the entire network. (Windows XP)

Figure 10: The middleware implementation of iFlow architecture: controlled diffusion session in action

6 Related Work

The design of the iFlow architecture and algorithms has been inspired by various exciting work from recent literature. We position iFlow in light of these work and highlight our original contributions in comparative studies.

Byers *et al.* studied the problem of delivering bulk data (on the order of gigabytes) to a large number of users across the Internet through an overlay network [11]. Their approach is similar to ours in that both employ Tornado codes to resolve the problem of obtaining the final data segments, as well as to provide robustness. Also, intermediate nodes have recoding capability and collaborate with each other actively. However, the focus of their work is on the problem of reconciliation between peer nodes, which is complicated by the application of Tornado codes. The design objective of their system is to deliver content to end users in a timely fashion in broadband wireline networks, while the design of iFlow focuses on improving bandwidth efficiency of information access in mobile ad hoc networks.

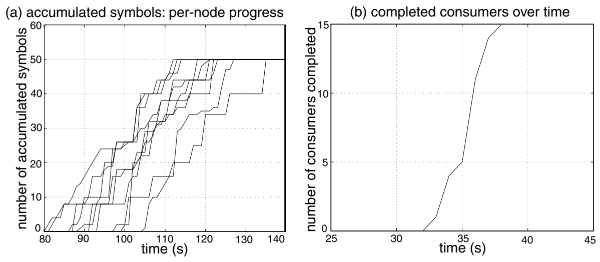


Figure 11: Live progress on consumers with the iFlow middleware testbed

The theoretical work of Grossglauser *et al.* [12] first reveals the fact that *node mobility*, which is usually treated as a negative factor in wireless networking, can play a *positive* role. Their main result is that, when nodes are mobile, the available end-to-end capacity for each source-destination pair in the network can remain constant, rather than approaches zero, as the size of the network grows. Although this result remains largely theoretical due to its strong assumptions, it does suggest that in certain scenarios in practice, it is possible to devise algorithms to trade off delay for better network performance, which is throughput in their case, and bandwidth efficiency in ours.

Network coding was first proposed and studied by Ahlswede *et al.* in the context of wireline networks [7]. It is shown that, applying network coding (usually linear codes suffice[8]) on intermediate nodes over a multicast network may increase its capacity. The problem of whether a given throughput can be achieved in a given multicast network was studied subsequently using an algebraic approach, with sufficient and necessary conditions provided for some cases [9]. Although exciting insights are provided, the existing studies on network coding has remained to be largely theoretical, and we are not aware of any published work that studies network coding in wireless networks. Due to the unique spatial contention of wireless transmissions in the local neighborhood, the effects of network coding is dramatically different from its counterpart in broadband wireline networks. In Sec. 3, we have shown that network coding can lead to higher bandwidth efficiency in wireless transmissions as well.

The 7DS system proposed by Papadopouli *et al.* [4] presents a practical study of data sharing among nodes in an ad hoc network that is sparse and less mobile. A node that loses Internet connection may acquire a desired data item from its neighbors, if it has been cached by one or more of them. The authors focus on the issues of how various network dynamics and design choices affect the performance of the 7DS protocol. The design of 7DS concentrates on data availability rather than bandwidth efficiency. With iFlow, although data availability is also improved due to its nature of distributed content caching, our main interests are on utilizing node mobility to disseminate popular data items in a bandwidth efficient way, subjecting to delay requirements imposed by applications.

A recent short paper by Goel *et al.* has first proposed to use Tornado codes to facilitate data dissemination in wireless networks [13]. Their simulation results show that the time it takes for all requesting nodes to obtain the data item using Tornado codes may be significantly reduced compared to not using Tornado codes. However, there does not exist any analytical work to support the results, and the results are limited to pre-defined mobility models. Further, the paper did not examine the issue of bandwidth efficiency in wireless networks. In the design of iFlow, we have brought the separate pieces together, including the use of Tornado codes that was previously mentioned [13], and also the algorithms facilitating information rendezvous and network coding on third-party nodes. We design and assemble the strategies with one unified objective: improving bandwidth efficiency, and study the effects of various tradeoffs and parameters pertaining this goal. We have not been able to identify such analysis in previous studies.

7 Concluding Remarks

This paper has presented the architecture, algorithms and analysis of *iFlow*, a middleware framework to facilitate information access in mobile ad hoc applications. We have shown that, iFlow is able to transparently provide a bandwidth-efficient way of information flow from suppliers to consumers, with strategies that include information rendezvous, erasure codes and network coding. We note that a high degree of node mobility actually contributes to achieving and improving bandwidth efficiency, and a relaxed delay requirement in delay-insensitive applications is an ideal scenario to deploy iFlow.

We are convinced that the full potential of iFlow with respect to efficient information access has yet to be realized. As an example, we may devise a mechanism for requests from consumers to be self-routed to the reservoir nodes (or suppliers) that hold the missing symbols, so that the requests may be satisfied earlier, with a slight penalty on bandwidth efficiency. Such a mechanism may be invoked when the consumers are about to activate the panic mode to directly contact the suppliers. Other improvements are also possible, including more in-depth integration of Tornado codes and network coding. We believe that, by extending our prototype implementation of iFlow as a middleware layer, iFlow may be rapidly deployed to assist emerging applications in mobile ad hoc networks, and, subsequently, redefine the communication patterns of such applications. Such patterns may further be studied to facilitate the design of lower-layer ad hoc network protocols.

References

[1] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks,” in *Proc. of ACM MobiCom*, September 2001, pp. 61–69.

[2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, “Instrumenting the World With Wireless Sensor Networks,” in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2001.

[3] J. Chang and L. Tassiulas, “Energy Conserving Routing in Wireless Ad hoc Networks,” in *Proceedings of IEEE INFOCOM*, 2000.

[4] M. Papadopouli and H. Schulzrinne, “Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices,” in *Proc. of ACM MobiHoc*, 2001.

[5] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data,” in *Proc. of ACM SIGCOMM*, 1998, pp. 56–67.

[6] J. Wieselthier, G. Nguyen, and A. Ephremides, “On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks,” in *Proc. of IEEE INFOCOM*, 2000.

[7] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, “Network Information Flow,” *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[8] S.-Y. R. Li and R. W. Yeung, “Linear Network Coding,” *IEEE Transactions on Information Theory, to appear*, 2002.

[9] R. Koetter and M. Medard, “Beyond Routing: An Algebraic Approach to Network Coding,” in *Proc. of IEEE INFOCOM*, 2002.

[10] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: A Media Access Protocol for Wireless LANs,” in *Proc. of ACM SIGCOMM*, 1994, pp. 212–225.

[11] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, “Informed Content Delivery Across Adaptive Overlay Networks,” in *Proc. of ACM SIGCOMM*, 2002.

[12] M. Grossglauser and D. Tse, “Mobility Increases the Capacity of Ad-hoc Wireless Networks,” in *Proc. of IEEE INFOCOM*, 2001.

[13] S. K. Goel, M. Chai, D. Xu, and B. Li, “Efficient Peer-to-Peer Data Dissemination in Mobile Ad-hoc Networks,” in *Proc. of International Workshop on Ad Hoc Networking*, August 2002.