Stony Brook University

From Sensor Networks to the Cloud: Smart System for Data Sensing, Matching and Storage

A Dissertation presented

by

Ying Li

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2015

Stony Brook University

The Graduate School

Ying Li

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Xin Wang - Dissertation Advisor
Associate Professor in department of Electrical & Computer Engineering

Alex Doboli - Chairperson of Defense Committee
Professor in department of Electrical & Computer Engineering

Thomas Robertazzi - Member of Defense Committee
Professor in department of Electrical & Computer Engineering

Samir Das - Member of Defense Committee
Professor in department of Computer Sciences

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

From Sensor Networks to the Cloud: Smart System for Data Sensing, Matching and Storage

by

Ying Li

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2015

With the drastic growth of attention in crowed sensing and wireless based social network applications, it is in desperate need to establish a comprehensive infrastructure that can efficiently sense the data, then accurately matches and delivers the gathered information to the various parties of interests in a timely manner. On the other end of the picture, the huge amount of user data needs to be reliably stored with easy and fast access at anytime and from anywhere. These inter-connected challenging problems form a complete information service framework of my thesis. In this thesis, we first introduce a set of adaptive sampling schemes based on improved compressive sensing technique for efficient information sensing and data gathering. Then in the second, we provide a storage efficient and traffic light-weighted fast content based information matching overlay for proper data dissemination and future processing. At last, we propose a space cost-effective and fast cloud based storage system using data deduplication and coding techniques for fast and reliable data storage. The proposed components work seamlessly towards a highly efficient and reliable framework that outperforms most peer systems for the various emerging applications.

Dedication Page

I am dedicating this work to my parents who always love and support me, and to my wife's patient moral support through countless days and nights. Wish them happy and healthy.

Contents

Acknowledgements

I would like to give my deepest thanks to my advisor-Professor Xin Wang, an admirable lady in the field of science. She has been patient with my research progress and always encourages me. She is to me an academic tutor, an elder in career and a friend in life.

I would also like to give my special thanks to Dr. Katherine Guo from Bell Labs, who has helped me a lot during my internships at Bell Labs, and work with me for several papers and projects. She is as kind as she is wise and professional. Wish her all the best.

1 Framework Overview

The whole framework is depicted in Figure 1.

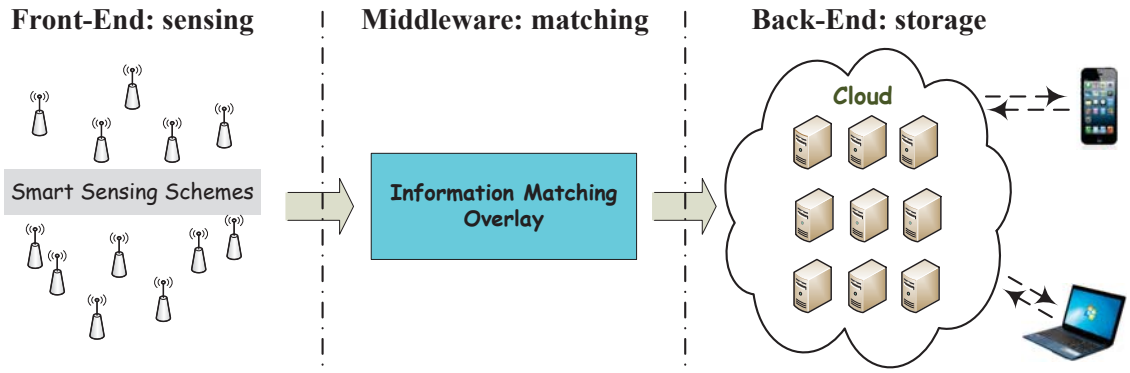


Figure 1: The framework overview.

At the front-end, the smart sensing component collects sensing data with high accuracy at extremely low resource cost. We have two independent sensing schemes to efficiently collect the information based on different application scenarios.

In the middle, we propose our information matching overlay that matches the data producer with potential consumers. The uniquely designed binary range vector supports rich and accurate content expression, and at the same time reduces management traffic and storage overhead, making it extremely suitable for wireless networks.

A robust system needs strong support from its storage strategy. At the back-end, we designed a cloud based storage system that utilizes both deduplication and coding technology to help reliably and efficiently maintain the huge amount of data. Our flexible and configurable design enables customized performance balance between access speed and space efficiency for different application needs. The cloud platform also features easy and everywhere access. Most importantly, all three components of our framework can be seamlessly connected via this cloud based storage system without dedicated interfaces.

In the following sections, each of the three components will be individually introduced.

2 Front-End: information collecting

For this part of research, we propose two independent works. The first one focuses on reducing the number of sensors while maintaining the sensing quality by adaptively learning the previous sensing results. On top of the same sensing efficiency as achieved by the first work, the second work is able to support a flexibly mixed level of sensing resolution without extra sensing cost. Both works outperform peer works in the literature.

2.1 Work 1: Adaptive Chasing Method in Compressive Sensing

2.1.1 Introduction

Efficient information collection is critical for many engineering problems, such as medical imaging, remote sensing and radar detection. Practical signals are generally continuous and can be sampled into digital form for more efficient storage, processing and communications. To achieve high efficiency, the fundamental challenge is to determine the minimum number of samples needed so that the signal can be acquired to meet the desired degree of fidelity in an often noisy environment.

For several decades, the Nyquist sampling theorem has been considered to be fundamental in the information theory area where it states that a band-limited signal can be completely recovered if it is sampled at a rate larger than two times its bandwidth. Recently, a new sampling theory, called Compressive Sensing (CS) [7,8,10,20] has attracted a lot of attentions. This theory enables the reconstruction of signal with the number of measurements much lower than that of the Nyquist sampling rate if the signal has a sparse nature in some domain.

The reconstruction of an under-sampled signal requires solving an under-determined set of linear equations which has more unknowns than the number of equations and may generally have an infinite number of solutions. In order to find the solution in this case, one must impose extra constraints. In CS theory, such constraint is the requirement of the signal to be sparse, meaning only a small part of the components are nonzero. If there is a unique sparse solution to the under-determined system, the CS theory guarantees to find it.

The fundamental works of CS include the introduction of the l_1-minimization method to reconstruct the signal. Later works on the reconstruction techniques provide some greedy approaches [40] [21] [39] [46] to recover the components of the signal gradually. Another thrust of the research attempts to directly apply CS in different application areas. CS is applied to reduce traffic volume in the process of signal acquisition [34] [6] [28]. In the area of sensor networks and RFID, CS is also applied to find target locations and numbers [17] [46] [23].

Complementary to above efforts, in this work, we would like to systematically study how the sensing matrix that defines the sensing behavior can be adaptively modified thus the subsequent measurements can focus on the proper signal subspaces based on the information from previous observations to improve the signal recovery accuracy.

Particularly, to interpret the principle of our adaptive schemes, we instanti-

ate an RF signal strength detection scenario, where the signals attenuate over distance before reaching a sensor. This setting allows us to map the tuning of sensing matrix in theory to the choice of sensor locations for sampling in practice in achieving better sensing quality while reducing the sensor usage.

The contributions of our work include:

- Our proposed algorithms significantly reduce the number of samples needed to accurately reconstruct a signal.
- Under the same level of noise interference, for the same number of measurements allowed, our schemes achieve much lower reconstruction error.
- Featuring on the adaptive learning process, our schemes do not have a requirement on the choice of underlying reconstruction methods.

Compared to conventional sampling methods, the simulation results demonstrate that our algorithms allow $45\% - 46\%$ less number of samples needed for accurate signal reconstruction and achieve up to 57% smaller signal reconstruction error under the same noise condition.

2.1.2 Fundamentals of Compressive Sensing

Recent research shows that a sparse signal can be reconstructed through Compressive Sensing with a high probability at much lower sampling rate. Moreover, most signals that are not sparse enough can also be projected to other domains to achieve the desired sparsity.

Let vector $x \in \mathbb{R}^N$ be a signal not sparse enough. Given an $N \times N$ orthogonal basis $\Psi = [\Psi_1, \Psi_2, ..., \Psi_N]$ with each Ψ_i being a column vector, we have:

$$x = \Psi s = \sum_{i=1}^{N} s_i \Psi_i \tag{1}$$

where s is the coefficients of x in the transformed domain Ψ. s is said to be k-sparse if it has at most k nonzero entries and $k \ll N$. The samples are then

$$y = \Phi x = \Phi \Psi s = As \tag{2}$$

where Φ is an $M \times N$ measurement matrix which will be defined later with $k \ll M \ll N$, A is the $M \times N$ sensing matrix, and y is the sample vector of $M \times 1$.

According to CS theory, instead of acquiring N samples of x, only $M = 2k$ of measurements are needed to reconstruct x by l_0 minimization method

when the measurements are noise-free [16]. Unfortunately, the l_0 optimization problem is NP-hard.

Matrix A is said to satisfy the Restricted Isometry Property (RIP) with parameters (k, δ) for $\delta \in (0, 1)$ if

$$(1 - \delta)\|v\|_2^2 \leq \|Av\|_2^2 \leq (1 + \delta)\|v\|_2^2 \tag{3}$$

holds for all k-sparse vector v.

As long as matrix A satisfies RIP, the following l_1-minimization problem, which is much easier to solve than the l_0 minimization, gives the accurate recovery of the original signal:

$$\min \mid s \mid_{l_1} \text{ subject to } y = As \tag{4}$$

The literature work [9] has pointed out that a randomly formed $M \times N$ matrix obeys the RIP with overwhelming probability provided that

$$M \geq c \cdot k \cdot \log(N/k) \tag{5}$$

where c is a constant, and k is the sparsity of the original signal vector.

2.1.3 Main Scheme

To investigate the possibility and methods of improving the sampling efficiency and accuracy through the tuning of sensing matrix and accordingly the adaptation of sensor positions or sampling points, in this paper, we example our theory into a real problem in a general sensor network scenario where a set of sensors are randomly distributed in a sensing field to detect the strength and locations of some signal sources, mapping the modification of sensing matrix in math on to the adjustment of sensor positions/sampling points in practice.

The strength at location j for a signal sourcing at location i is roughly approximated as:

$$P_{ij} = \frac{P_0 G_{ij}}{d_{ij}^{\beta}}, \tag{6}$$

where P_0 is the signal strength at its source location i, d_{ij} is the Euclidian distance between the signal source at i and the location j. G_{ij} captures the Raleigh Fading of the signal. β is the decay factor with the range $[2.0, 5.0]$ depending on the environment. The real and imaginary component of the Raleigh distribution are both independent and identically distributed Gaussian variables with zero mean and variance of σ_0^2 [46].

For the convenience of location reference of a sensing area and to facilitate scalable monitoring, the sensing domain is partitioned into N grids, each could

have no or several signal sources inside it. We do not differentiate individual signals within a grid, but rather regard them as one aggregate signal located at the geographical center of the grid and refer a **signal source** as a grid which has aggregated signals inside it.

Let $s = [s(1), s(2), ..., s(N)]^T$ be an $N \times 1$ column vector, where the i^{th} entry $s(i)$ is the aggregate signal strength of grid i. s is k-sparse with $k \ll N$, which means at most k grids out of N actually have signal sources. Figure 2 shows an example system of 16 grids. Some of the grids have sensors deployed, and some have signal sources with different level of energy indicated by the dot of different sizes. The sensors can be activated to take sampling measurements or in the sleeping state to save energy. The adaptation of the sensing process can be performed through the selection of the sensors for samplings.

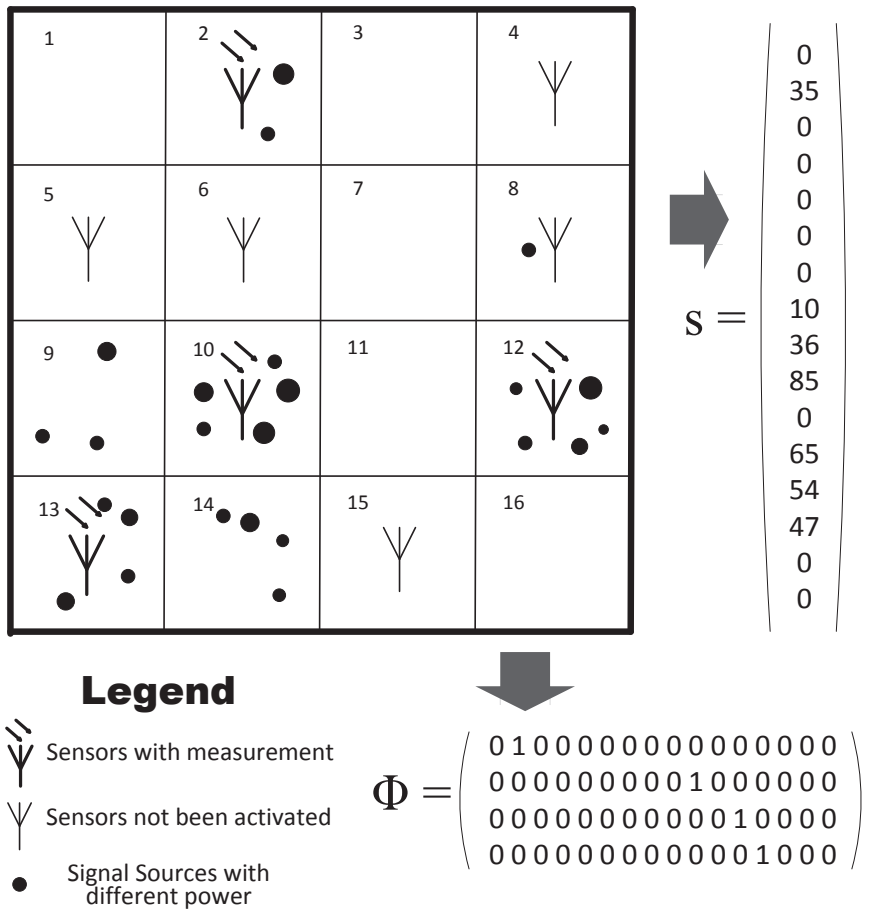


Figure 2: A demo of the system with the corresponding signal vector s and measurement matrix Φ.

To monitor the energy of signals, traditionally, a large number of sensors need to be placed uniformly across the whole monitoring domain and kept active to maintain the coverage need [27] [22]. In reality, there will be only very small number of signal sources appearing in the domain at a time and some may be clustered in certain area. We could reduce the number of sensors and samplings needed by applying compressive sensing in the spatial domain. Instead of taking a number of samplings [46] randomly at one time in the field to apply the CS theory, in this paper, we would like to start with a small amount of samplings at random locations, then adaptively adding new measurements at locations that are not previous sampled based on the learning of previous observations to gradually concentrate the sensing resource to positions that are more likely to have signal sources, in order to increase the recover accuracy while reducing the total number of samples taken.

Let Ψ be an $N \times N$ transformation matrix embodying the signal energy decaying process defined in Eq.(7)

$$\Psi = \begin{pmatrix} \frac{G_{11}}{d_{11}^{\beta}} & \frac{G_{21}}{d_{21}^{\beta}} & \cdots & \frac{G_{N1}}{d_{N1}^{\beta}} \\ \frac{G_{12}}{d_{12}^{\beta}} & \frac{G_{22}}{d_{22}^{\beta}} & \cdots & \frac{G_{N2}}{d_{N2}^{\beta}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{G_{1N}}{d_{1N}^{\beta}} & \frac{G_{2N}}{d_{2N}^{\beta}} & \cdots & \frac{G_{NN}}{d_{N1}^{\beta}} \end{pmatrix} \tag{7}$$

Then $x = \Psi s$ is the received signal strength vector with $x(j)$ denoting the aggregate signal strength received by the sensor at grid j from all the signal sources.

Given the setting above, it is left for us to choose the best observation points at which samples are taken. To benefit from the grid management and maximize the sensing efficiency for each sensor, we assume one grid has one sensor, or none. The sensors chosen to take samples can be specified, or "selected", by the measurement matrix Φ, which is an $M \times N$ matrix of the format shown in Figure 2. The i^{th} row of the matrix $\Phi(i)$ is a $1 \times N$ vector with all elements equaling to zero except $\Phi(i, j) = 1$, where j is the index of the grid at which the i^{th} sensor is located. Given that each entry $x(j)$ denotes the total received strength at grid j of all the signals, the effect of left-multiplying Φ to vector x in Eq.(2) is to select M out of N rows of x, or equivalently choosing the samples taken by sensors at M specific grid positions as illustrated in Figure 2.

Under this formulation, adjusting the number of samples to be used in one reconstruction process is extremely convenient. It can be done by simply adding or removing a few rows in Φ corresponding to the sensors at required

sampling positions. More excitingly, only the samples from sensors at new locations need to be collected in the upcoming round. In addition, the samples already acquired previously by sensors at certain grid locations can be directly combined with the new samples to form a more informative sample vector y as shown in Eq.(8), which can be applied to recover the data using the combined Φ.

$$y = \begin{pmatrix} y' \\ y'' \end{pmatrix} = \Phi\Psi s = \begin{pmatrix} \Phi' \\ \Phi'' \end{pmatrix} \Psi s \tag{8}$$

y' is the vector of samples already collected by sensors selected previously in Φ', and y'' contains the new samples taken by sensors newly specified in Φ''. Thus for each sensor ever chosen to take a sample, it only needs to sample once during the whole process, and its sample is kept for future use if sampling at this sensor location is needed again. This reuse ability enabled by our unique formulation helps greatly preserve the sensing resource consumption, and is the base of our algorithms to be proposed.

In Eq.(2), the sensing matrix $A = \Phi\Psi$ has been proven in [46] to obey RIP condition as long as matrix Φ and Ψ are constructed as defined above. Therefore a signal can be uniquely recovered with enough samples under our formulation by applying the l_1-minimization method.

Restricted by the deployment and maintenance cost, in the practical sensing application, it is preferable to use as few sensors as possible to meet the acceptable sensing requirements. In a practical monitoring domain, the sensors either are static once planted, or generally cannot move as fast as signal sources do to track them. Given abundant but not excessive number of sensors have been planted in a detection area, a possible way of improving the sensing quality while cutting down the sensing cost is to selectively activate only a portion of sensors at which positions samplings are needed while keeping the rest of sensors in energy saving mode. In this work, a better set of sensors will be activated in each round of sensing based on the reconstruction results from the previous round. We use $\hat{s}^{(i)}$ to denote the reconstructed signal after the i^{th} round of sensing. As we start at a number of samples that is smaller than necessary for compressive sensing, there exists inaccuracy for each intermediate $\hat{s}^{(i)}$. Although neither the positions nor the values of the nonzero entries of $\hat{s}^{(i)}$ may be accurate, however to some extent, the actual nonzero entries of the original vector may be close to or around these nonzero positions indicated by $\hat{s}^{(i)}$. In other words for the signal detection example, the signal sources are probably in the region close to the grid locations corresponding to the nonzero positions of $\hat{s}^{(i)}$. So by "moving" sensors (which in our case equivalent to activating sensors at the desired locations) towards the estimated locations of the signal sources step by step, the algorithms will improve the

sensing results until the positions as well as the values of the nonzero entries no longer change. This way we can find the accurate positions along with the energy level of the signal resources. This is the fundamental principle of our adaptive algorithms.

Algorithm 1 outlines the details of Individual Chasing in each iteration. In the i^{th} iteration, according to the previous reconstruction vector $\hat{s}^{(i-1)}$, for each of its nonzero entry $\hat{s}^{(i-1)}(n)$, a sample is taken at a sensor whose location is closest to grid n. After each non-zero position n is ensured to have one sampling in the corresponding grid, the l_1-minimization process is invoked to get the reconstruction $\hat{s}^{(i)}$ based on the combined samples y and combined Φ as in Eq.(8). The reconstruction result $\hat{s}^{(i)}$ is fed into the Algorithm 2 for termination condition check to determine whether the algorithm should end or continue with more iterations.

Algorithm 1 Individual Chasing

1: In the i^{th} iteration:
2: **for** each nonzero position n of $\hat{s}^{(i-1)}$ **do**
3: find a grid position p in $\mathbb{P}$ with the smallest euclidian distance to grid n.
4: **end for**
5: combine new samples with existing ones for y.
6: do l_1-minimization on y and Φ to get $\hat{s}^{(i)}$.
7: call Algorithm 2 to check the termination condition.
8: **if** algorithm does not terminate in this iteration **then**
9: $i = i + 1$, go back to Line 1 and start the next iteration.
10: **end if**

Algorithm 2 Termination Condition Check

1: **if** the nonzero positions of $\hat{s}^{(i)}$ are all the same as $\hat{s}^{(i-1)}$ **then**
2: **if** the numeric difference of each nonzero value between $\hat{s}^{(i)}$ and $\hat{s}^{(i-1)}$ is smaller than a percentage threshold Δ of $\hat{s}^{(i-1)}$ **then**
3: reconstruction process terminates in this iteration.
4: **else**
5: continue with the next iteration of reconstruction.
6: **end if**
7: **else**
8: continue with the next iteration of reconstruction.
9: **end if**

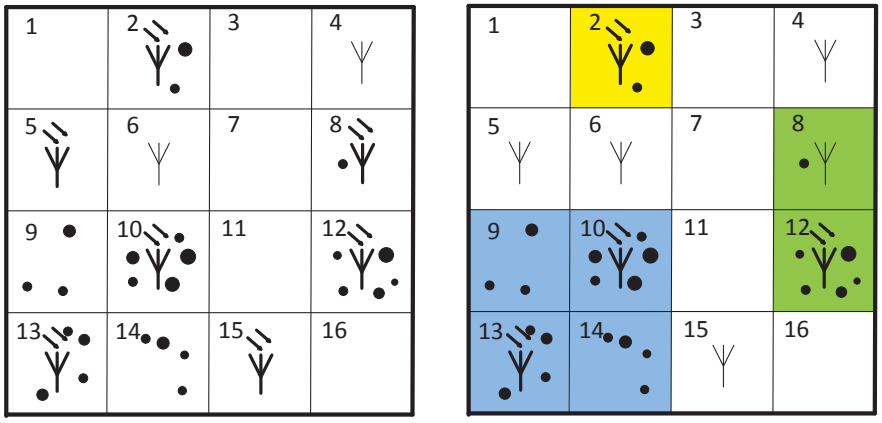
The Individual Chasing (IC) scheme adapts well when signal sources are uniformly distributed in the monitoring field. Figure 3-(a) shows the sensor locations when the Individual Chasing algorithm terminates. It can be observed that sampling measurements have been taken at each grid with signal source that also has a sensor inside. For the grids with signal sources but without sensors deployed, samples are taken at the closest grids, i.e. samples have been taken at sensors in grid 5 and 15 for nearby signals in grid 9 and 14.

In addition, We proposed another algorithm - Centroid Chasing (CC), for the application scenarios where signal sources are located in concentrated clusters. Due to the quick growth of social network applications, the signal sources tend to locate closely and form clusters. For example, cell phone users are often observed in hot public areas such as shopping malls, movie theaters, restaurants, airports, etc. Portable devices such as laptops or tablet PCs are highly concentrated in residential quarters, office buildings or coffee shops. The clustering patterns of signal sources may be exploited to guide the sampling locations to facilitate the finding of signal sources with even fewer sensor measurements.

The Centroid Chasing scheme initializes similarly as Individual Chasing, then at each iteration, possible grid positions of signal sources are clustered and sensors closest to the center of each cluster are activated for sampling. Figure 3-(b) shows the sensing condition at the end of the Centroid Chasing algorithm. Grids with signals are grouped into 3 clusters. Within each cluster, a portion of the closest sensors are activated for sampling. Compared with Individual Chasing scheme on the left of the figure, fewer number of sensors are used given the signal sources have a nice clustering feature. Both algorithm perform significantly better than state-of-the-art literature works, as shown later in the main result section.

Local Optimum Avoidance - Random Exploration

Very unlikely but possible, adaptive algorithms could converge at local optimums. To our problem, local optimum does not give accurate reconstruction at algorithm termination. Because in an iterative scenario, one can only look at the intermediate results to decide whether the iteration should stop. The Individual Chasing and Centroid Chasing algorithms will stop when consecutive iterations present no more change in the results. In order to break possible local optimums, we introduce an extra step called Random Exploration. To be specific, for both algorithms, when the Termination Condition Alg.2 is satisfied at certain iteration, we do not terminate the program. Instead, we randomly pick up some sensors that have not been activated before to take samples, then re-enter the adaptive process and let it converge again.



(a) Individual Chasing (b) Centroid Chasing

Figure 3: Illustration of Individual Chasing and Centroid Chasing.

2.1.4 Main Results

Performance Metrics:

We first give the definitions of several performance metrics.

- **Reconstruction Error**: defined as the Sum of Absolute Difference (SAD) between the recovered and the original signal vector:

$$SAD = \|\hat{s} - s\|_{l_1} = \sum_{i=1}^{N} |\hat{s}_i - s_i| \tag{9}$$

  This measurement metric evaluates the accuracy of vector reconstruction. It reflects not only the degree of error due to the position mismatch of nonzero entries, but also the difference in magnitude for each unmatched entry.

- **Number of Samplings Needed (M)**: in our adaptive algorithms, more samples may be added in each new iteration. The number of samples needed for our algorithms are defined as the total number of sampling measurements during the whole sensing process, in order for fair comparison with other non-iterative algorithms whose numbers of samples needed are defined as the one-time choice made before the algorithms start.

Simulation Set-Up:

Based on [5], the approximate transmitting power at the signal source location is, 80 dBm for a FM radio station with 50-kilometer range, 27 dBm for typical 3G cellular phone, 20 dBm for IEEE 802.11b/g Wireless LAN 20MHz-wide channels in the 2.4 GHz ISM band, and 4 dBm for Bluetooth Class 2 radio, etc. We adopt dBm as the measurement unit of signal strength in our simulation. Since in our case each grid could have multiple signal sources, the overall signal strength for a grid is the aggregate of these basic measures in different combinations. Being aware that the numeric scale of the nonzero entries of signal vector is not critical to the problem of compressive sensing recovery, we assume a range of 30-500 dBm for the possible aggregated signal strength inside any single grid.

The simulation is carried out with MATLAB. The signal energy decay model follows Eq.(6) with $\beta = 2$. Both real and imaginary parts of Rayleigh fading follow an independent and identical Gaussian distribution with the mean of 0 and the variance of 0.5 as in [46] for fair comparison that follows. 150 sensors are randomly deployed in an area of $N = 30 \times 30 = 900$ grids, with at most one sensor inside one grid. 30 meters for the size of each square grid is a good choice to reflect distance effect in signal propagation. The termination condition threshold Δ in Alg.2 of 5% will generally guarantee the recovery result to be accurate at the algorithm termination time with the recovery error in the order of 10^{-4} even for real valued signal vectors, thus our default choice on Δ is 5%. All the sensors are static and can be individually activated from the sleep mode for sampling, and then be turn off again. There are $k << N$ grids with signal sources. k is the sparsity value, which is varied in different studies. The positions of k grids with signal sources can be either randomly distributed in a clustered fashion, or uniformly distributed. To examine the reliability performance of our schemes, Gaussian White noise $N(0, \sigma^2)$ is added to the observed sample vector y in some of the simulation runs, and SNR measure is exploited to quantify the noise strength. Each presented result is the average of many runs.

Our proposed two algorithms Individual Chasing and Centroid Chasing, also referred to as IC and CC from now on, do not depend on any specific CS reconstruction technique. Thus we chose two fundamental and most prevalent types of work for performance evaluations-l_1 minimization based CS and greedy based CS. GMP (INFOCOM'11) [46] provides a greedy based reconstruction algorithm for CS, and also exploits the received signal strength at different grid positions to help solve the target localization and counting problem. l_1-magic is a concise and dominant realization of l_1 minimization based CS scheme, which can be directly applied to and is thus worth comparing with our simulation scenario of signal strength vector reconstruction.

Number of Samples Needed:

Pursuing a minimum number of samples needed for an accurate signal reconstruction is the major challenge and research focus in the closely related research fields. We evaluate the minimum number of samples needed for accurate signal vector reconstruction (zero reconstruction error) under different levels of signal sparsity for each scheme in Figure4. As expected, the number of samples needed increases as k grows for all the algorithms. Particularly, Figure4-(a) is under the scenario where the signal sources are randomly uniformly distributed across the network grids. IC performs slightly better than CC as expected. The clustering function of CC performs poorly when the signal sources are uniformly distributed, which leads to more iterations to converge and more samples needed. Compared to GMP, IC requires 45% fewer samples when k is small, and about 23% fewer samples when k gets bigger. IC requires $46\% - 25\%$ fewer samples than l_1-magic for different k.

In Figure4-(b), the signal sources are distributed in clustered fashion across grids, which benefits the clustering process of CC algorithm. Thus CC is observed to require fewer samples than IC as the number of signal sources exceeds certain value. While the performance difference between IC and CC is small, CC requires $40\% - 33\%$ fewer samples than GMP, and $40\% - 30\%$ fewer than l_1-magic, which are big improvements. In general, both IC and CC work extremely well no matter the signal sources are concentrated or scattered, and far outperform the other two schemes under the same k.

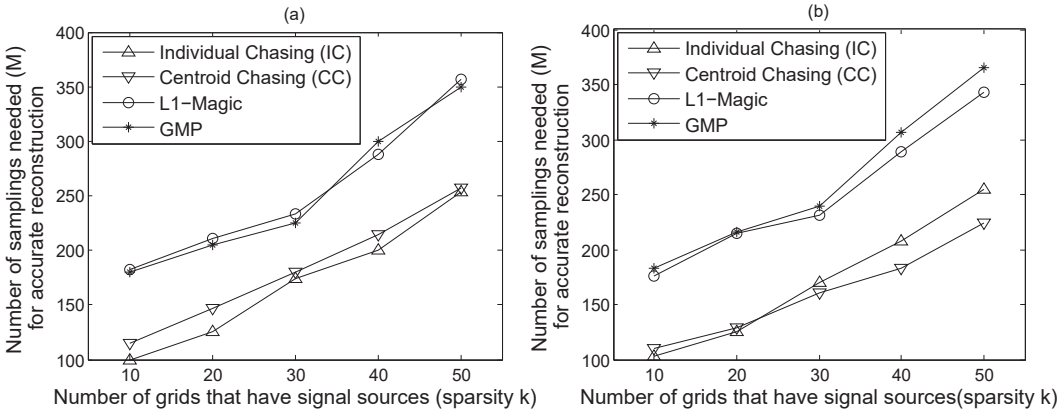


Figure 4: (a) scattered signal sources. (b) clustered signal sources.

Given the performance difference between IC and CC is small and both algorithms follow the same principle, we only study and compare IC with the other two schemes and assume the signal sources are randomly and uniformly distributed in the simulations that follow.

Reconstruction Error:

A. Convergency Study

We study the convergency of IC in Figure 5-(a). It is clear that under all k, IC is able to converge within 3-6 iterations to get accurate reconstruction with 0 error, and it exhibits a rather steady (i.e., approximate-linear) improvement in reducing the reconstruction error in each iteration. It converges faster for larger k. This is due to the fact that we initialize $2k$ number of samples for the optimal performance. With a larger k, there are more samples taken at the beginning, therefore it needs fewer iterations to get enough overall samples for accurate recovery.

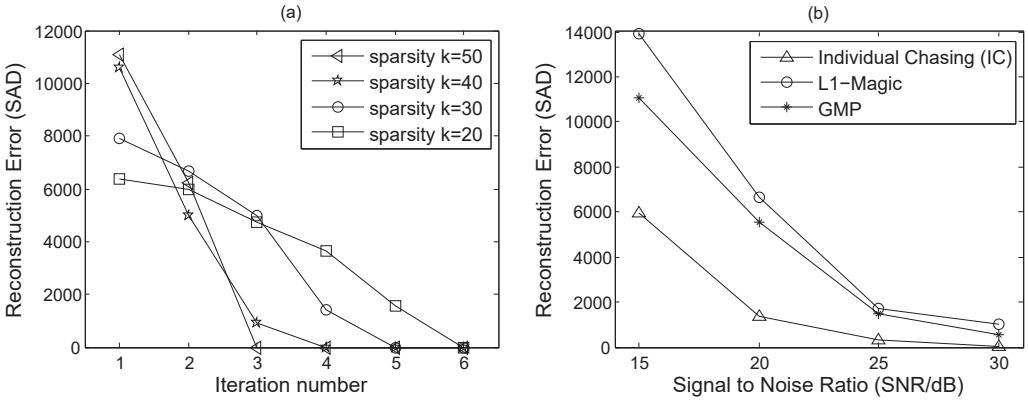


Figure 5: (a) Convergency study of Individual Chasing algorithm. (b) The reconstruction error comparison due to noise under the same $k = 50$ and $M = 250$.

B. Performance Under Noise

The adaptive algorithms can always find the accurate reconstruction given enough iterations and proper handling with local optimum avoidance. However under noisy environments where the sample y is contaminated, the final reconstructed result is different from the actual signal vector.

The reconstruction errors due to sampling noise for different algorithms are compared in Figure 5-(b). The reconstruction error reduces as signal-to-noise ratio increases for all three schemes. Under the same sensing condition of $k = 50$ and $M = 250$, at each SNR level, IC gives much more accurate result than the other two. In the worst scenario with the strongest noise at 15dB SNR in our test setting, the reconstruction error for IC is approximately $(14000 - 6000)/14000 = 57\%$ smaller than that of l_1-magic. GMP is slightly more accurate than l_1-magic under the same sensing setting, because it enumerates all possible values for each possible nonzero position of the vector at the cost of higher computational overhead.

2.1.5 Conclusion

In this work, we observe and theoretically prove that by adjusting the structure of the sensing matrix, the number of samples needed for high-quality signal recovery in compressive sensing can be further reduced. We propose two adaptive algorithms, Individual Chasing and Centroid Chasing, for different signal source distribution scenarios. Both schemes adaptively concentrate sensing resources to proper signal subspace towards better acquisition of signals, and do not depend on any specific CS reconstruction methods. The instantiation of our algorithms on the general signal strength sensing problem with distance fading can be conveniently generalized to various similar applications. Extensive simulations demonstrate that our algorithms can achieve as much as 57% more accurate signal recovery under noisy conditions, and require up to 46% fewer samples than state-of-the-art related works.

2.2 Work 2: Weighted Zoom-in on Sensing Resolution

2.2.1 Introduction

There have been increasing interests and efforts in applying a network of sensors to monitor a large and complex space for surveillance and other applications. Among various types of events to monitor, many are related to the detection of energy sources. For example, there is a need to monitor the radio spectrum usage conditions of a network area and enable cognitive network transmissions over unused spectrum for significantly higher wireless network capacity. Some other applications include the monitoring of the radiation strength of a suspecting region, and finding regions of noise sources to prevent against riots in urban sensing.

In these applications, besides detecting events, there is often a need to find the location ranges of signal sources to guide further actions. Depending on the application types and importance, the location resolution requirements for monitoring are different. As a major application, there are growing interests in finding the locations of RF devices based on their received signal strength (RSS). This includes finding the region of a wireless jamming source to protect against the communication attack, and identifying an RFID location for object tracking. Rather than finding the location range of specific signal sources, the sensing data can also be applied to form a signal distribution map. For example, secondary radio devices in the network may collaboratively form a spectrum sensing map of primary signals to facilitate cognitive network transmissions. In fact, any form of signals that attenuate along distance are able to reveal the location information given that enough samples are taken by surrounding sensors.

Conventionally, the localization of wireless targets is transformed into geometrical problems based on the mutual distances between targets or between targets and known reference points determined through the measurement of signal strength [15, 26, 30, 31, 41]. Obviously, there is a tradeoff between the resolution of location ranges of targets/events and the number of sensors needed for monitoring. It would require a large number of sensors to achieve fine location resolution, which is difficult if the monitoring domain is large.

In many practical wireless applications, targets are often distributed sparsely in a large area. The emerging Compressive Sensing (CS) theory [8, 14, 19] sheds some light on the monitoring of signals using a much smaller number of sensors. Specifically, by mapping the magnitudes and locations of a set of targets into a monitoring vector whose items are corresponding to the signal strength of a specific location, the sparsity of targets in the spatial domain

will be translated into the value sparsity of the vector (i.e., many items are zero) so the vector can be reconstructed from sensing data of a smaller number of sensors. The locations of targets can be determined from the positions of non-zero values in the vector [23, 46].

Despite the potential, it is challenging to directly apply the CS-based method in practical sensing scenarios. First, the number of sensors needed depends on the number of targets, which is often unknown and varies over time. The sensing is made even harder with the presence of noise. An insufficient number of sensing samples and large sensing noise would render the CS recovery accuracy very low and even result in the failure of recovering the monitoring vector. In addition, in the above formulation, the dimension of the vector is determined based on the desired resolution of the location. The vector dimension will increase as the monitoring domain becomes larger, which would make the computational complexity of the CS problem prohibitively high.

In light of above problems, the aims of this work are to perform efficient and accurate signal monitoring with much lower number of sensors, and design a fast and efficient method to reconstruct the strength of wireless network signals with their location resolution at the desired level.

Specifically, to more accurately find the signal locations in a practical monitoring domain with varying number of signal sources and measurement noise, we propose a novel *weighted zoom-in* algorithm that adaptively finds the distributions of signal sources at the desired location accuracy through two iterative procedures: 1) a **focusing** procedure to fully use the available sensor resources to coarsely detect the regions where signals are located, and 2) a **zoom-in** procedure to *virtually* divide each of the valid regions into sub-grids to achieve a finer level of sensing resolution. The overall sensing process iterates with the interactive operation of the two procedures until the desired sensing resolution is reached for all the areas in the field.

By reducing the original full-range signal localization problem into several rounds of CS reconstruction problems with lower vector dimension, in each round it requires much smaller number of samples and lower computation complexity as compared to those needed in the conventional CS-based schemes that try to tackle the whole problem in full scale at once. On the other hand, the same number of samples will appear to be larger when applied to reconstruct a smaller dimension vector, which would allow for higher reconstruction accuracy. The benefits of our algorithms can be summarized as follows:

- Our proposed algorithm significantly reduces the total number of samples

needed for accurate target localization.

- Under the same level of noise and for the same number of samples allowed to take, our scheme achieves much higher reconstruction accuracy.
- For solving the same scale of localization problems, our algorithm has a much shorter running time.
- Despite the use of iterative reconstruction, our design only requires taking samples once at the beginning of the whole process, without having to take additional samples as the algorithm iterates.

Different from existing adaptive CS algorithms which generally vary the sampling rate to improve the sensing quality, our algorithm will adapt the detection resolution based on the results from the previous rounds of processing to reduce the uncertainty and ambiguity for more accurate detection in a dynamic and varying physical environment. The adaptation of the sensing granularity and vector size are carried out through a ***virtual*** process during the signal reconstruction process based on the same sample data set. This allows the sensing matrix to be quickly and flexibly reformulated in response to each round of zoom-in process without the need of taking additional samples at sensors.

Besides its applications for signal localization and forming signal distribution map, we expect the proposed algorithm will help advance the sparse signal processing field by improving upon the conventional compressive sensing theory with more intelligent use of measurement data. Our algorithm can also facilitate non uniform monitoring with the sensing resolution at different regions set differently based on the priority and need. Our performance results prove that the proposed scheme allows for the number of samples taken to be much smaller than that needed based on conventional CS sampling and signal reconstruction schemes.

2.2.2 System Overview

The strength or energy of real-world signals decays over distance, which is the core basis for most localization methods. Different than utilizing geometrical principles, such as trigonometry, to deduce the location of a single target source based on the samples of this signal at multiple positions, we virtually divide the area with grids, then abstract the geographical distribution of targets across the field into numerical vectors by making a one-to-one mapping between the grid sequence number and the entry sequences number of the vector. The location information of all the targets over the monitoring area can

be retrieved all at once by back-translating the vector representation that we can accurately reconstruct using compressive sensing technique out of a small number of aggregated samples. The advantages are obvious-fast and cost less sensing resources.

To be realistic, we consider a scenario where a set of sensors are deployed in a monitoring region to estimate the location and number of targets that transmit radio signals. Generally accurate monitoring of a large area requires more sensing resources, thus traditionally, a large number of sensors need to be placed uniformly across the whole monitoring domain and kept active to maintain the coverage need [27] [22]. In reality however, the number of targets and the room they will take in the large area at a given time is small and limited. In other words, they are geographically sparse. To save installation and maintenance cost while preserving the same level of sensing quality, we could reduce the number of sensors and samplings needed by applying compressive sensing (CS) technique in the spatial domain. Instead of triggering a number of sensors in the field that are required theoretically by conventional compressive sensing methods as will be introduced in the next section, in this paper, we propose to activate only an small number of sensors to take sample only once, then adaptively zoom-in at locations that we are certain about signal source existence. The whole process only requires one time sensor sampling, and areas with no targets are filtered out of focus so that the reconstruction efforts are dedicated to sub-regions with targets to achieve extremely high reconstruction accuracy at low sensing resource cost.

For a specific sensing application, we assume that the transmitting signal power of every individual target is approximately the same in magnitude, so that it is feasible to distinguish the targets and count the number of targets in an area by dividing the total signal strength of this area with the unit amount. The signal at the receiver (i.e. the sensor) end has been attenuated along the path. The basic signal fading model and the construction rule of the sensing matrix follow the similar pattern as specified in Eq.(6)(7)(8) of Work 1 in section 2.1.

Under our formulation, when construct sensing matrix $\mathbf{A}$ for a sensing task, the number of sensors allowed to be used decides the row dimension, while the column dimension depends on the size of the problem, or the number of grids N to be estimated. The distances between the center of each grid and each of the sensors need to be calculated accordingly in order to get each entry of $\mathbf{A}$. The **shape** or **size** of the grids does **not matter** at all, since we always take the center of each grid to approximate every other point in the grid when calculating the distances.

Specifically, in our target localization application, when we fix the sensors

to be used while deliberately change the number and size of the grids to focus on particular sub-regions. By re-calculating the distance between each new grid and the chosen sensors to re-built the matrix $\mathbf{A}$, we can **reuse the same set of samples y** initially taken to reconstruct $\mathbf{s}$, without having to re-sampling each time the grid changes. When the changes of grids are local and minor, the modification of matrix $\mathbf{A}$ is even more convenient by simply adding, removing or replacing a few of its columns corresponding to the affected grid positions. This is the most important and innovative feature of our design which helps greatly preserve the sensing resource consumption, and is the base of our algorithms to be proposed.

In order to be able to accurately refer the location and for conveniently formulating the problem into our mathematical model, we place virtual grids on top of the monitoring area, which does not actually exist, but rather as a reference to coordinates. The grid is not of a fixed size, actually, the feature that the grid size can be freely adjusted as needed is just the unique and most important characteristic and advantage of our design. Moreover, as we will show later in this section, the real flexibility of the virtual grid infrastructure allows the grid size of different regions of the monitoring area to be different in order to best suit the resolution need for each part of the area. Once the area is marked by grids, the localization problem can then be formulated into a vector format with each entry of the vector represents one grid position in the field and the value of this entry denotes the total aggregated signal strength emitted from all the targets within the corresponding grid. We define this vector as $\mathbf{s}$. Its dimension will be N if it represents a mapping to N grids. As long as the mapping rule from the grid locations to the entry positions of vector $\mathbf{s}$ is enforced consistently, we can always translate the content of $\mathbf{s}$ back to the solution of localization problem correctly.

Figure 6 illustrates the grid concept with an example. The monitoring area is covered by virtual grids of different sizes, or resolutions, at different parts depending on the density of targets. There are several versions of vector $\mathbf{s}$ reflecting the location distribution of targets for each round of grid change, which will be introduced later as the zoom-in process.

2.2.3 Main Scheme

Restricted by the deployment and maintenance cost, in practical sensing applications, it is preferable to use as few sensors as possible to meet the acceptable sensing resolution requirements. Generally, the sensors are static once planted at a practical monitoring area. The optimal placement of a given number of sensors for a specific monitoring area is not the focus of this work. Rather, we

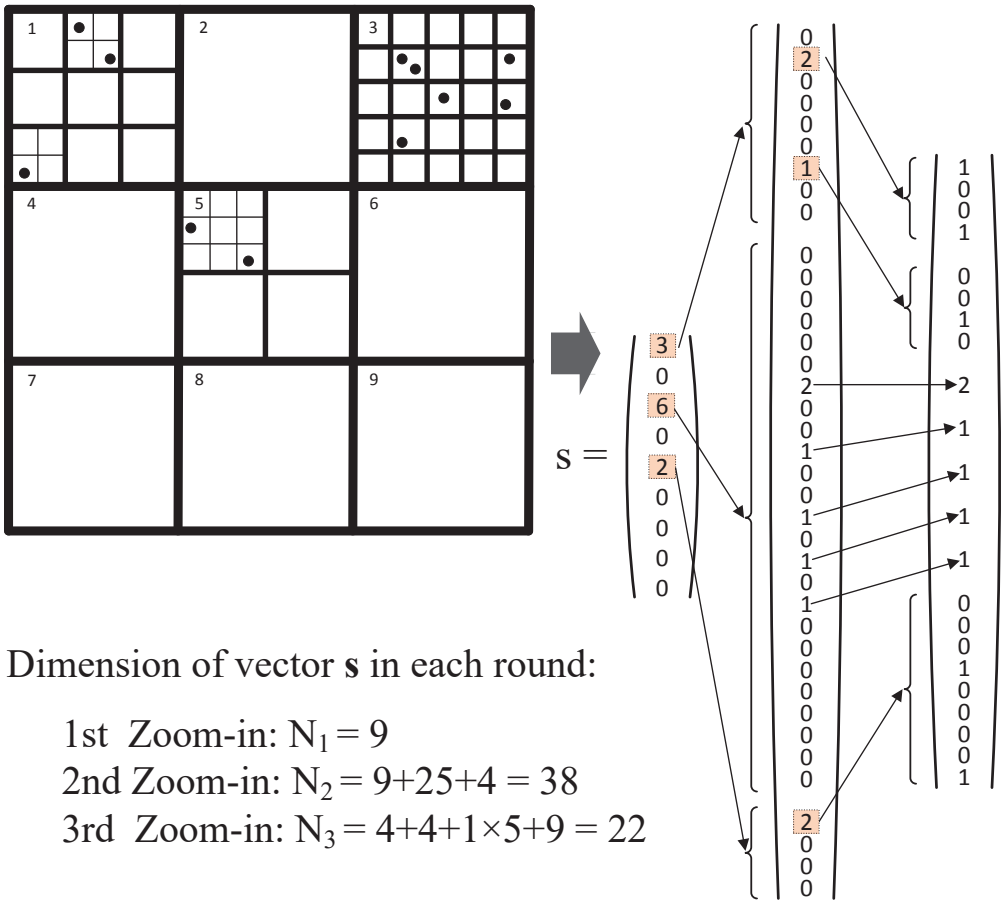


Figure 6: Example problem and illustration of Zoom-in process.

aim to propose a smart sensing method, which does not depend on the underlying sensor placements, to improve the speed and quality of sensing processes. Given abundant but not excessive number of sensors have been planted in a detection area, we are going to show that by randomly picking a small portion of the installed sensors, a number smaller than conventional compressive sensing technique would require, and taking sample only once at these sensors, the signals can be accurately reconstructed through our chasing zoom-in procedure which at the same time is computationally lighter compared to conventional schemes.

Instead of taking samples at a large number of sensors and try to reconstruct the original signal vector with equal weight on each of its entries, then bet on the number of samples to be enough, a possible way of improving the sensing quality while cutting down the sensing cost is to adaptively reconstruct

the vector from the samples by iteratively filter out the entries which do not hold information and concentrate the reconstruction "attention" to those carry valid components. In this work, the nonzero entries of the reconstructed vector will be re-evaluated in a finer resolution level on the next round until the desired resolution is reached. The adaptive sensing process iteratively divide the current virtual grids into smaller grids of different sizes at different regions of the whole area, and terminates when the grid of each sub-region is smaller than or equal to a fixed threshold value, which is defined as the desired **final resolution level** for the localization of targets that we must achieve.

As previously analyzed, the process requires sampling only once on a small number of sensors at the beginning, then the subsequent adaptive reconstruction procedure involves no more sampling but pure math in rapidly reducing scales, and converges quickly.

A. Focusing Operation

In our system model, we use a vector to represent the grid location of signal sources across the monitoring areas. Since the signal sources are usually sparsely distributed across the monitoring domain, after each round of reconstruction, the recovered vector $\mathbf{s}$ should have many zero entries corresponding to grids detected without signal sources. We propose a **focusing** procedure to filter out the zero entries after each round of reconstruction, and put signal detection and recovery effort on non-zero entries in the next round.

Instead of being ideally zero, the recovered vector $\mathbf{s}$ in each iteration may contain some entries with extremely small or negative values, which are insignificant but may be considered as non-zero and mislead the further recovery process. We thus introduce a two-step pruning procedure before the zoom-in operation that follows: 1) Setting all the negative entries of $\mathbf{s}$ to zero, and 2) For positive entries, setting all the entries with values below a percentage - α that of the largest $\mathbf{s}$ entry to zeros. α can be customized by the system administrator.

B. Zoom-in Operation

After the focusing operation has filtered out the zero entries of $\hat{s}$, for all the non-zero entries left, their corresponding grids, referred as the **valid grids**, are going to be further divided into smaller grids for finer resolution. This action is called the **zoom-in** operation. The zoom-in operation allows the sensed samples to be contributed to the location of targets during the reconstruction calculation, therefore for a smaller number of samples taken, our scheme can still achieve the accurate reconstruction. Generally, the change of focusing scope and resolution scale would ask for appropriate sampling adjustment accordingly, because the dimension (N) of the vector to be recovered (s) has changed. Innovatively for our design, benefit from the proper construction

of the sensing matrix, the change of problem scale does not require the similar adjustment for the samples. The adaptive reconstruction process involves only one time sampling at the beginning no matter how many zoom-in operations are needed before termination at convergence. This unique feature helps greatly reduce the sensing cost for taking samples and transmitting sampling results among energy constraint sensors.

Another innovative advantage endorsed by the virtual grid infrastructure and the supporting sensing matrix mathematical model is the flexible density based sub-division of zoom-in operation. For each valid grid (i.e. corresponding non-zero entry of vector $\hat{s}$), it does not necessarily have to be divided into equal number of sub-grids. In our design actually, the number of sub-grids to form within each valid grid is properly quantified based on the density of targets inside to ensure most accurate allocation of reconstruction effort for each individual region. The number of sub-grids to form within valid grid j is denoted as n_j and defined by Eq.(10):

$$n_j = \frac{\hat{s}_j / P_0}{\eta} \tag{10}$$

$\hat{s}_j$ is the non-zero value for entry j of vector $\hat{s}$, which is the aggregated signal strength within the corresponding grid. This value, when divided by the unit target transmitting power P_0, approximates the number of targets inside the grid corresponding to the j^{th} entry of $\hat{s}$. η is a percentage which indicates the ratio of sub-grids that have targets inside over the total number of sub-grids within a valid grid. When mapping to vector representation, η is an equivalent measure to the vector sparsity level. The underlying compressive sensing technique we exploited for reconstruction requires the vector to be reconstructed is sparse. Eq.(10) ensures that after each zoom-in operation, every region of the monitoring area is of proper sparsity for reconstruction. Usually a η that is smaller than 50% is sparse enough for accurate reconstruction using compressive sensing method. Note that although η and the vector sparsity k are related, they are different. k is defined as the number of non-zero entries of the vector which we use to describe the target localization in the grids. Since multiple targets could fall in a single grid, η is usually larger than the ratio of k over the total number of entries of the vector. We study the impact on the choice of η in the simulation. Based on target density, intuitively, by dividing more sub-grids in the grids with more targets and less sub-grids in those with less targets, the average proportion of sub-grids with signal sources inside (i.e. the sparsity of the corresponding vector) can be maintained at a steady level.

It is not hard to observe that the position as well as the resolution (i.e. grid size) of each valid grid is not critical and actually does not matter at all

to the reconstruction process. Because although the resolutions or sizes of the valid grids at different regions are different after several zoom-in operations, the distance from the center of each valid grid to each sensor is easy to get by simple calculation since all the grids are referred to and managed by virtual coordinates. Once the distances are decided, the sensing matrix A can be immediately constructed according to the number of samples we have taken initially and the number of valid grids at the current stage by following Eq.(7) Although the resolutions, or the number of sub-grids inside, for different regions of the monitoring area at a certain time point could all be different, as long as the mapping from geographical grid locations to entry positions of the vector is kept consistent, we can always safely apply compressive sensing to reconstruct the vector and continue the iterations until the final resolution is reached at each part of the monitoring area.

C. Main Algorithm

The principle of focusing and zoom-in processes is different from the classical "divide and conquer" method. We cannot simply ignore the zero entries and take each non-zero entry as a standing alone sub-problem to solve. The sample taken at each sensor ($\mathbf{y}_j$) represents the strength of the aggregate signal received from all the signal sources, i.e. the summation of the received signal strength from all target sources. Separating the whole monitoring domain and trying to tackle each sub-region individually would not be able to differentiate signals received from different regions and avoid their interference. Instead, with flexible reformulation of the sensing matrix in each iteration, our proposed scheme applies focusing and zoom-in operations to reconstruct the signals based on all valid grids.

Algorithm 3 Valid Component Chasing Zoom-in

1: Initial zoom-in, construct the sensing matrix $\mathbf{A}$
2: Reconstruct $\mathbf{s}$ using $l1$-minimization subject to $\mathbf{y} = \mathbf{As}$
3: Focusing
4: **if** Not all the valid grids have reached the required final resolution **then**
5: Zoom-in at the valid grids that have not reached the final resolution
6: **else**
7: Terminate and return $\mathbf{s}$
8: **end if**
9: Re-calculate distances, adjust sensing matrix $\mathbf{A}$
10: Go to Line 2

Algorithm 3 outlines the complete reconstruction process. At each iteration, the vector $\mathbf{s}$ is reconstructed using the $l1$-minimization method of com-

pressive sensing. The focusing step that follows filters out the zero entries of the vector $\mathbf{s}$ (i.e. the grids that contain no targets). On Line 5 of the algorithm, the zoom-in operation is applied to help reconstruct the vectors based on only the valid grids that have not yet reached the required final resolution. After zoom-in, since each region has been zoomed into different resolution levels, the distance from each grid to each sensor has to be re-calculated to form the updated sensing matrix $\mathbf{A}$ accordingly. The process terminates when each sub-region reaches the desired final resolution of sensing.

Samples are taken by M sensors in the monitoring domain. M could be the total number of sensors deployed or a subset of sensors activated to conserve energy. In the later case, the M sensors could be randomly selected from the field. The choice of M could be based on the coarse knowledge of the number of grids that have signals inside for a given resolution level, which we denote as T. Our adaptation of the virtual grid number makes our algorithm more robust to the inaccuracy of the M setup. The impact of M on the reconstruction performance and the optimal choice of M with respect to T is studied in our performance evaluations. The initial zoom-in determines the number of grids to divide the whole original monitoring area into. It can be based on the estimated density of targets following the same principle in Eq. (10). In each new iteration, the dimension of vector $\mathbf{s}$ needs to be determined first. The *zoom-in* procedure helps to fulfill this task. Based on Eq. (10), the dimension of the vector to recover can be obtained by summing up the number of sub-grids from each valid grid, as illustrated in Figure 6.

In the example shown in Figure 6, the whole area is first divided into 9 grids. After the first round of reconstruction, the focusing operation only keeps grids 1, 3 and 5 which are valid grids, and filters out the rest of grids that contain no targets. Based on the values of the vector entries corresponding to these grids, each of the grids is further divided into different number of sub-grids. Grid 3 has a larger value in the corresponding vector entry and is thus divided into more sub-grids than the other two. At this point, grid 3 has already reached the desired final resolution, therefore the sub-grids inside grid 3 will not be further divided in the subsequent iterations. However for grid 1 and 5, the process will continue one more round of focusing, zoom-in and reconstruction before the whole reconstruction process terminates, where each part reaches the desired sensing resolution level.

2.2.4 Main Results

We carry out the simulation under similar setting as in Work 1 in section 2.1. We first give the definitions of several performance metrics.

- **Localization Accuracy**: defined as the ratio below:

$$LA = \frac{\#\text{ of targets localized correctly}}{\text{total }\#\text{ of targets actually exist}} \tag{11}$$

  where the numerator is the number of targets being accurately estimated about their locations. The location accuracy is calculated based on the desired final resolution.

- **Number of Samples Needed (M)**: defined as the minimum number of samples needed for a scheme to accurately estimate the locations of targets at the desired final resolution. It is the same as the number of sensors needed for our proposed scheme. Each device senses the spectrum for a short duration, and the average measurement signal strength is used as one sample.

- **Algorithm Running Time**: defined as the execution time of each algorithm for a complete sensing task. This metric reflects the time complexity of each algorithm compared, and does not include the time for the sample capturing and transmission.

We compare our proposed weighted zoom-in (WZI) algorithm with two fundamental and most prevalent types of CS recovery schemes-l_1 minimization and greedy schemes. l_1-magic is a concise and dominant CS scheme based on l_1 minimization, which can be directly applied to the localization problem. GMP (INFOCOM'11) [46] provides a greedy reconstruction approach, and also exploits the received signal strength at different grid positions to help solve the target localization and counting problem.

A. Number of Samples Needed

Achieving the desired sensing quality with the minimum number of samples is always desirable, and is the major challenge and research focus. We evaluate the minimum number of samples needed for accurate vector reconstruction (i.e. accurate localization for targets) for each scheme in Figure 7. As expected, the number of samples needed increases as the total number of targets T grows for all three algorithms. The two reference schemes have similar performance, while WZI greatly outperforms the other two. For a given T, WZI requires approximately 60% fewer samples compared with the other two schemes. From the figure, it is not difficult to observe that the minimum number of samples needed for WZI to accurately reconstruct the problem falls approximately within the range $1.5T - 1.75T$, i.e., 1.5 to 1.75 times that of the number of targets. This range is much smaller than the theoretical number of samples

needed for M, which is generally much larger than $2T$ and the empirical values are between 3T and 4T [35, 48]. This range could serve as a guidance for the optimal selection of M in our algorithm.

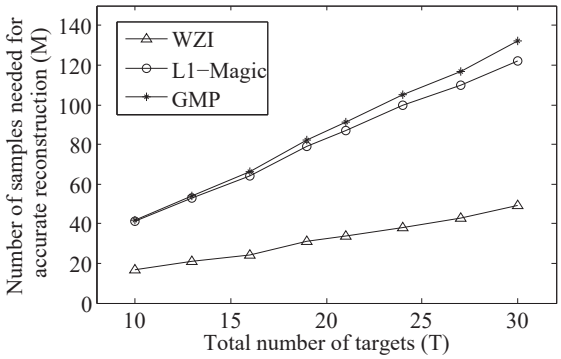


Figure 7: The number of samples needed for accurate reconstruction.

B. Localization Accuracy under Small M

When the number of samples M is limited, there will be inaccuracy associated with the reconstruction process, so does the solution to the target localization problem. We study the Localization Accuracy in Figure 8. The number of targets T is fixed at 20. It is clear that when M is large enough, there are enough number of samples, all three schemes are able to give accurate reconstruction results. As the number of samples used M continues to decrease, the Localization Accuracy of l_1-magic and GMP deteriorate rapidly. In contrast, WZI is still able to accurately reconstruct the locations of targets for a relatively small number of samples. When M gets even smaller, WZI maintains a slowly dropping curve of the accuracy, which is still much higher than those of the other two methods. In the lowest number of samples studied, the recovery accuracy of our WZI almost doubles that of the two reference schemes.

C. Performance Under Noise

Under noisy environments where the sample $\mathbf{y}$ is contaminated, precise reconstruction and completely accurate localization are difficult to realize. It is thus important to compare the level of accuracy each algorithm can achieve under the noise.

The reconstruction errors due to sampling noise for different algorithms are compared in Figure 9. The Localization Accuracy increases as the signal-to-noise ratio increases for all three schemes. Under the same sensing condition of $T = 20$ and $M = 50$, at each SNR level, WZI achieves much more accurate result than the other two. In the worst scenario at 15dB SNR with the

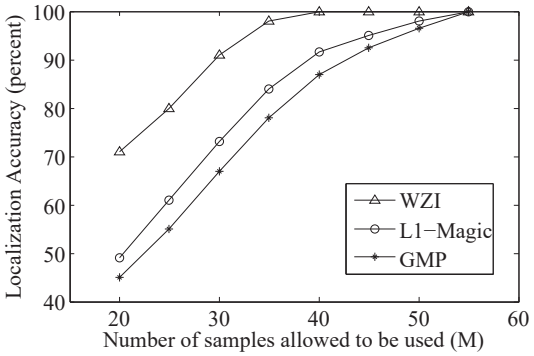


Figure 8: Localization Accuracy v.s. M ($T = 20$).

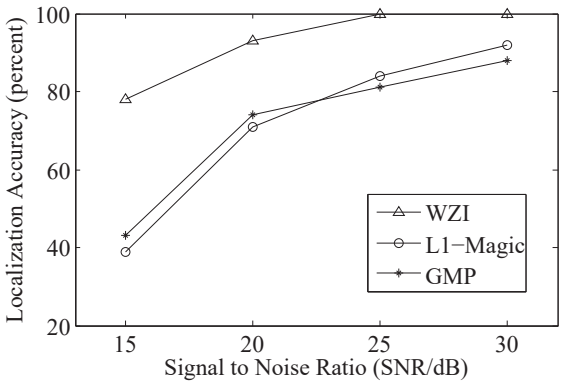


Figure 9: Localization Accuracy under noise ($T = 20$, $M = 50$).

strongest noise in our test setting, the Localization Accuracy for WZI almost doubles that of l_1-magic. GMP is slightly more accurate than l_1-magic under the same sensing setting, because it enumerates all possible values for each possible nonzero position of the vector at the cost of higher computational overhead.

D. Algorithm Running Time

Fast response and short processing time is another critical metric for a sensing task, especially for real time applications or when the targets are moving. In this part of evaluation, we examine and compare the running time for each algorithm. This metric only considers the actual execution time of each algorithm, and does not include the time for taking sensing samples and transmitting the samples to the fusion center. The number of samples used is fixed at $M = 80$ in this test.

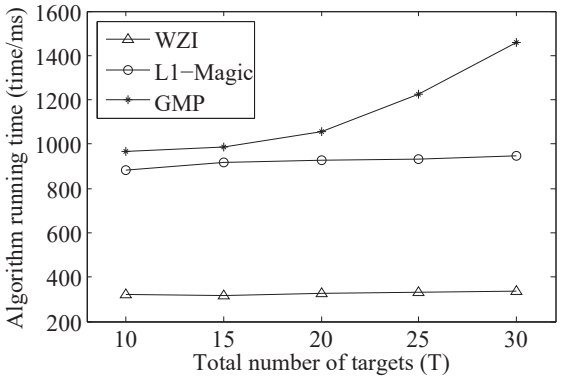


Figure 10: Comparison of algorithm running time ($M = 80$).

WZI transforms a large problem into several sub-problems with much smaller scales, which helps to greatly reduce the processing complexity thus time. In Figure 10, it executes more than three times faster on average than both l_1-magic and GMP under any given number of targets in the scenarios studied. When the number of targets T is relatively large, there are more valid grids after each round of reconstruction, and more sub-grids are formed after each zoom-in operation. However, our proposed algorithm can terminate with fewer rounds of iteration because the fast sub-grids generation rate helps the target detection process to quickly reach the desired final resolution. With the counteraction of the two factors, the running time of WZI remains relatively stable with the increase of T. Both l_1-magic and GMP attempt to recover all signals at once, thus it takes much longer time to run compared with WZI. Specially, the running time of l_1-magic depends only on the scale of problem input – the number of grids at the finest resolution, which does not change with the number of targets T. On the contrary, GMP greedily recovers each entry of the vector, thus it takes longer time to finish when there are more targets. Its running time rises up to five times that of WZI at the highest T tested.

2.2.5 Conclusion

In this paper, we demonstrate that by adaptively focusing the reconstruction effort into regions with signals and gradually increasing the resolution at these regions, a higher signal localization accuracy can be achieved with a smaller number of measurement samples. We exploit variable size of virtual grids to help flexibly adjust the resolution based on the target density and utilize Compressive Sensing technique to reconstruct the vector that bears the location

information. The combined *focusing* and *zoom-in* process works extremely efficient in reducing the large original problem into smaller scales, which enables our algorithm to achieve extremely high reconstruction accuracy at high execution speed with a low requirement on the number of samples needed. Extensive evaluation tests demonstrate that our algorithm can achieve 2 times more accurate localization recovery under noisy conditions in less than one third of the running time, while requiring up to 60% fewer samples than state-of-the-art related works.

3 Middleware: matching the data with potential consumers

3.1 Work 3: Efficient and Content Expressive Information Matching

3.1.1 Introduction

Efficient and flexible information matching over wireless networks has become increasingly important and challenging with the popularity of smart devices and the growth of social-network-based applications. As an example of new era information service, a smart-phone user in a downtown block wants to obtain a recommendation for some restaurants while people close-by may be also searching for the same type of information. Another user just stepping out of a Thai cuisine is satisfied with the dining experience and would like to share this place with others. Other applications include traffic information posting and retrieval where users cooperatively contribute to and benefit from the real-time traffic reports. These applications can be better met by a "contribute-and-benefit" pattern system. Publish/Subscribe (Pub/Sub) system is one of this type, in which subscribers specify their interests and publishers post advertisements. The system matches subscriptions with publications. Unlike client/server models, the Pub/Sub model decouples time, space, and flow between publishers and subscribers to provide flexibility in information distribution.

Gryphon [42] and SIENA [12] were once popular Pub/Sub models in wireline networks, however, their tree-based structure are not scalable in dynamic wireless network whose topology may constant change due to mobility and connection broken. Many later attempts have been made to apply Pub/Sub infrastructure for wireless networks [45] [11] [36], where the information in the systems is roughly divided into several basic types. These platforms cannot efficiently support heterogeneous user application needs.

Different from conventional Pub/Sub systems which mainly categorize information into a few types for ease of implementation, the modern information system is expected to better meet the customized information needs of individual users. Besides the difference in categories, the heterogeneity of information is more generally resulted from different values or contents for the same type of information. Simply ascribing information into coarse types (food, movie, car, etc.) cannot meet most application needs. On the other hand, completely expressing every detail of the information in words and matching over them is not feasible in reality.

In this work, we proposed a reliable and scalable binary range vector summary tree (BRVST) infrastructure for flexible information expression support, effective content matching and timely information dissemination over the dy-

namic wireless network. A novel attribute range vector structure has been introduced for efficient and accurate content representation and a summary tree structure to facilitate information aggregation. For robust and scalable operations over dynamic wireless network, the proposed overlay system exploits a virtual hierarchical geographic management framework. Extensive simulations demonstrate that BRVST has a significantly faster event matching speed, while incurs very low storage and traffic overhead, as compared with peer schemes tested.

The main contributions of our work are:

- We propose a mechanism to flexibly and efficiently represent information with the combination of a set of elementary tuples for numerical expression of the content.
- We propose a novel Attribute Range Vector that allows flexible vector length adjustment based on the information accuracy requirement, and supports a unique simple bit-wise operation for quick content matching check, to facilitate accurate content representation as well as low-overhead in storage and transmission.
- We propose a Summary Tree structure to facilitate efficient aggregation of information, which significantly reduces the overhead for storing and transmitting information updates.

3.1.2 Model Background and System Overview

In this work, we adopt the notion of Publication and Subscription to distinguish information from the generators and to the consumers. The whole information space is built up with the basic element - attribute $(A_i, i = 1, 2, ...)$, which contains attribute name (a_n) specifying the identification of an attribute (numeric ID in realization), and attribute value (a_v) that specifies the content and is usually a numeric point or range. i.e. $A_i = \{a_n, a_v\}$. A subscription $\mathbf{s}$ is a conjunction of n attributes: $\mathbf{s} = \{A_1 \wedge \cdots \wedge A_n\}$. A publication $\mathbf{p}$ is a disjunction of attributes: $\mathbf{p} = \{A_1 \vee \cdots \vee A_n\}$, and is also referred to as an event. Conventionally the attribute value of a subscription could either be a numeric point or a range, while that for publication is assumed only to be a numeric point [47] [29]. However, very often some attributes of the information, when generated, are not absolute point values. For example, the video surveillance data could have its *time* attribute as a range which confines the start and end points of a video segment. So our design also supports range value for a publication attribute. In Figure 11 example, a user submits a subscription

specifying the criteria of interested restaurants, while another user publishes a review on his dining experience.

Subscription

Attribute Name	Attribute Value
Food style	3 (i.e. Italian)
Price per person	0 ~ 100
Open hours	15:00 ~ 23:00
Review rates	3 ~ 5

Publication

Attribute Name	Attribute Value
Price per person	30 ~ 50
Review rates	4
Longitude	-73.98
Latitude	40.72

Figure 11: Examples of Subscription and Publication.

We consider a publication and a subscription to match each other iff: for each attribute existing in the subscription, the same attribute must also exist in the publication; and for the common attributes, those from the publications must have their value ranges contained by the value ranges of the corresponding attributes in the subscription. i.e.$\forall A^s \in s, \exists A^p \in p : (a_n^p = a_n^s, a_v^p \subseteq a_v^s)$, where the superscript ^{s} denotes the subscription, while ^{p} denotes the corresponding terms for a publication.

In order to make the infrastructure scalable and more robust to the network dynamics, we introduce a virtual management infrastructure where the network space is mapped into virtual zones each consisting of a set of virtual grids (Figure 12). With many wireless devices equipped with GPS receivers or having other methods of localization [11], the grid and zone which a node belongs to can be easily determined. The grid size can be determined by the system based on the application scenarios and performance tradeoffs. Its effects is studied in the simulation.

Each grid can elect [44] a Grid Manager (GM) for Pub/Sub message collection, aggregation and matching within the grid. Each zone also has a Zone Manager (ZM) responsible for Pub/Sub aggregation, matching, data catching over grids within the zone. The schemes for leader election and maintenance have been proposed by many literature work [44] which can be leveraged in our system, and the election can take into account factors such as the power and resources of the nodes as well as the node distance to the center of the grid or zone. The managers can be static or mobile, depending on the system application scenarios.

Event matching and Pub/Sub message update are both performed on demand. Subscriptions and publications in a grid are collected and aggregated.

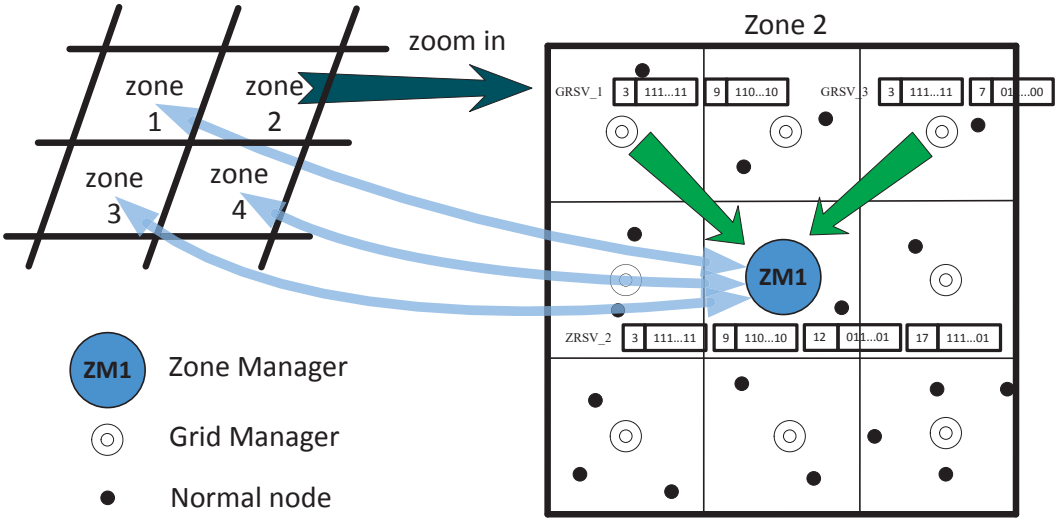


Figure 12: An example system where each zone has 9 grids. The zone manager collects Pub/Sub messages from grids within the zone and aggregates them into control messages to exchange with other zones.

Although nodes may frequently move in and out of a grid, the aggregated messages may stay unchanged. Messages are sent to the upper level ZM only upon the change of aggregate filter. This will significantly reduce the overhead for Pub/Sub message transmission and matching in a dynamic wireless network. A ZM maintains the Pub and Sub information of the grids within its zone with efficient data structures to be introduced in Section 3.1.3, and the Pub/Sub information of the whole zone can be similarly further aggregated. As many mobile users have interests in close-by information, the aggregate filters only need to be shared among nearby zones or zones identified with Pub/Sub relationship.

Any new subscription or publication will trigger the event matching process within its own zone first, then matching at other zones whose aggregate filters imply potential chance of match will initiate. This will significantly reduce the data matching and distribution overhead. Once a publication is matched with one or more subscribers, the overlay structure will then deliver the data to these destinations using the stateless geographic multicasting, RSGM [44], for reliable and low overhead transmissions. The detailed routing process is beyond the scope of this work.

3.1.3 Main Structures and Scheme

A. Binary Vector and its Operations

We propose a binary bit vector named Attribute Range Vector (ARV) to flexibly represent the numeric range values of an attribute, referred as the target range. The target range could be a single point value as well. An ARV has a small size and is easy to process. The numeric value of an attribute is generally limited within predefined boundaries, which can be determined in advance by the system based on some common knowledge. For example, the temperature of the weather has an lower and upper limit in physical world. A subscriber could indicate her interest by setting a target range within the limit defined by the system. To facilitate flexible range matching, the predefined limit range is divided into N smaller equal segments, while the value of N can vary based on the matching accuracy requirement. An N-bit ARV is formed by representing whether a segment matches a content range, following the steps below:

Step0: Set the initial segment to be the whole predefined limit range.

Step1: Check if the target attribute value range falls into some existing segments with each occupied more than α (percentage) of the segment range, an accuracy threshold desired. If so, goes to the next step; otherwise divide each of the current segments into equal halves, and continue this step.

Step2: Make an N-bit vector with N equal to the current number of segments, with each bit indicating if the attribute range overlaps the corresponding segment range, 1 yes, and 0 no.

From the above ARV construction process, we can see that the number of bits of the vector can only be the power of 2, i.e., $N = 2^i$, $i = 0, 1, 2, 3...$, and the length of ARV can be continuously doubled until a desired representation accuracy is achieved. The threshold α trades off between accuracy and simplicity of the message representation.

For example, the attribute *Age*, often involved in social network applications, is limited within 0 to 100. Three subscriptions that contain the attribute *Age* are: Age_{Sub1} 1-48, Age_{Sub2} 26-47, and Age_{Sub3} 38-60. Their corresponding ARVs are obtained by constructing a split tree following the above steps as shown in Figure 13, with the level i having 2^i segments. Suppose the threshold α is set to 90% in this example. Age_{Sub1} falls into the segment 0-50 and the fitting ratio of the target range 1-48 is 48/50, which is larger than the threshold $\alpha = 90\%$. So this segment is accurate enough to represent the target range

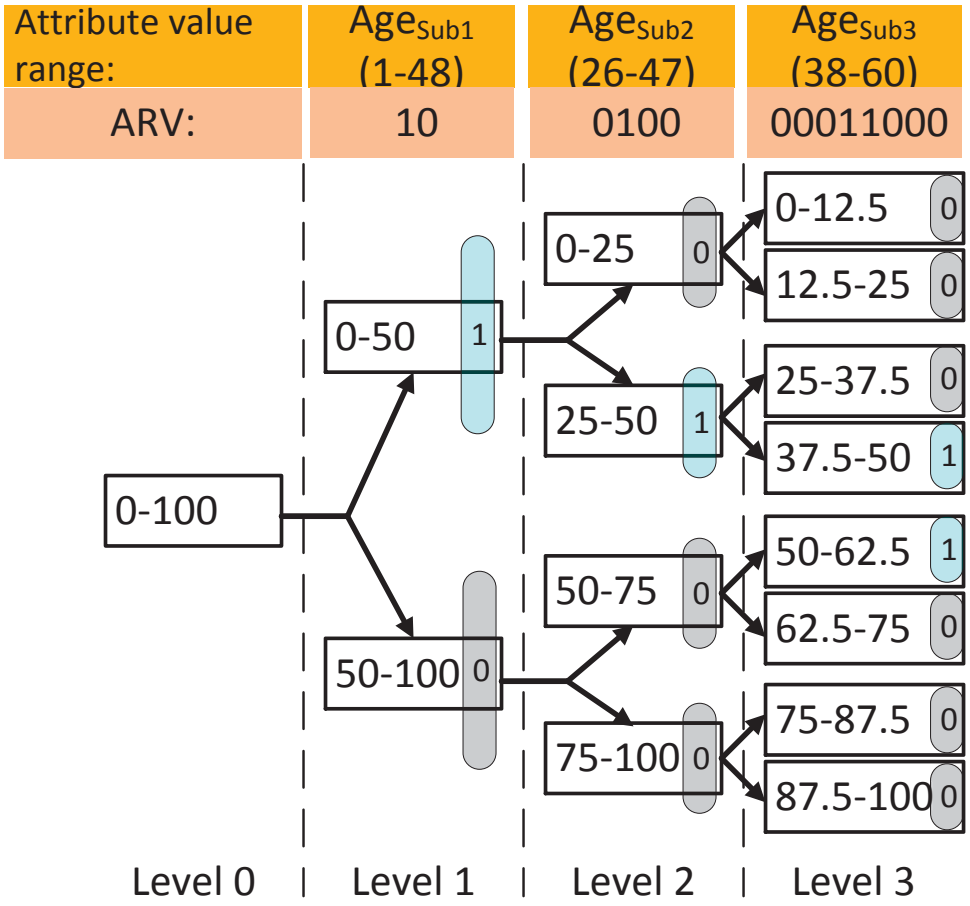


Figure 13: The segment division procedure in constructing an ARV.

and the ARV for Age_{Sub1} is 10. Age_{Sub2} apparently falls into the 0-50 segment of level 1, however, this range is not very accurate. We further divide the overall range into 4 new segments at the level 2, so the range 26-47 falls into the segment 25-50. We can use 4-bit vector 0100 to represent this 4-segment coverage, with the left most bit standing for the segment of the lowest value. The target range 38-60 of Age_{Sub3} spans across the 0-50 segment and the 50-100 segment at the first-level of the split tree, but these two segments are inaccurate in representing the target range. If we go deeper into the level 3, the segment 37.5-50 & 50-62.5 will be accurate enough with the resulting ARV 00011000.

A shorter ARV is always preferable to reduce the transmission and storage overhead. The ARV bit vector is checked after each modification for the potential of simplification. Except level 0, the number of bits in an ARV is

always even and in the power of 2. When the length of ARV is larger than 1, starting from one side of the vector, if *every* consecutive 2-bit has the same value (both '1' or both '0'), the length of the vector can be reduced into half by taking every other bit to form a new ARV. For example, 1100 can be reduced to 10, but not 0110 nor 0111 which does not have the same value for consecutive 2-bit. The simplification operation will continue without losing the accuracy of the information until the vector cannot be further simplified.

Likewise, a given vector could also be extended by 2^i (i=1,2,3...) times when needed by simply doubling the bit patterns. This feature is extremely useful in the matching process we will discuss later, where two or more ARVs need to be adjusted to have the equal length before they can be compared or merged.

The proposed ARV is the elementary component of Subs and Pubs, and some other aggregated management structures at different hierarchical levels are composed of ARVs.

ARV Merge: A merge operation is needed for information aggregation. As the length of the vectors could only be the power of 2, two vectors of different lengths can always be made equal by doubling the length of the shorter one several times as previously mentioned. Then the merge can be completed by only a simple bitwise "OR" between two ARVs. The accuracy of the ARV will not be impacted when it is scaled up or down. The merge operation is always carried at the length of longest ARV thus over the finest level of segments, and the merge of ARV will maintain the accuracy level.

Match of ARVs: For differentiation and ease of referral, an subscription and publication attribute range vector are called respectively an S-ARV and P-ARV. If one or more attributes of the subscription are not included by the publication, we can immediately claim they do not match each other, given the conditions above. Otherwise they are further checked. First all the S-ARVs and P-ARVs are respectively concatenated following the corresponding order as shown in Figure 14, with all the redundant P-ARVs ignored and each corresponding pair of P-ARV and S-ARV scaled to the same length. Then the Sub and Pub are considered to match each other if and only if all bits after the following operations are 0: The cascaded P-ARVs vector and S-ARVs vector first have the bitwise AND operation, and the result XOR with the original cascaded P-ARVs vector.

Figure 14 gives an example. The subscription has 2 attributes, and the publication has 3 attributes. To perform the matching, the attribute 1 (Attr.1) of the publication is scaled to 4 bits, while the Attr.5 is omitted because it is not involved in the subscription. Then bitwise operations are carried out: (P-ARVs `AND` S-ARVs) `XOR` P-ARVs, and the result is not all '0' thus is not a

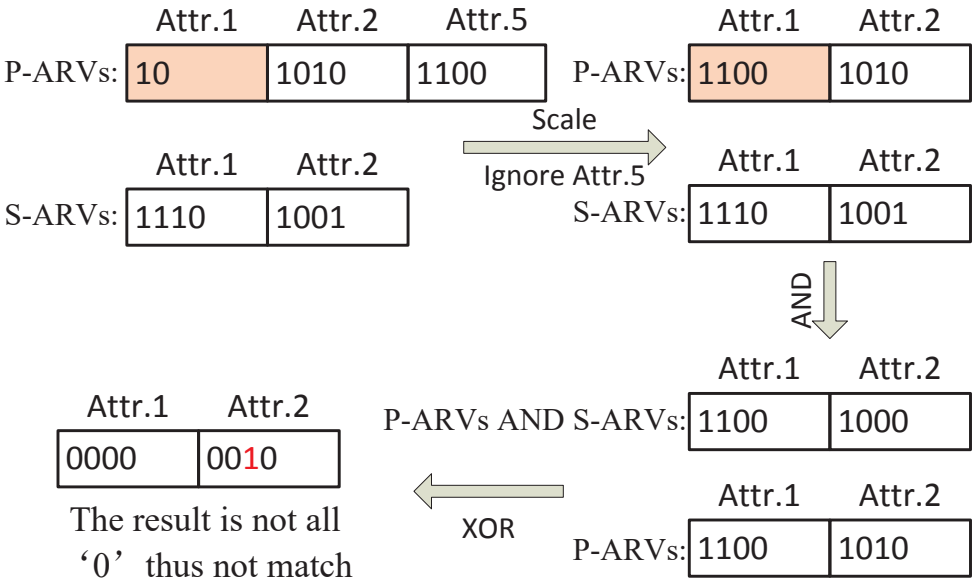


Figure 14: The bit-wise operations for matching evaluation of a Pub and Sub.

match. Because Attr.2 of the publication has a '1' in the bit position where the subscription Attr.2 does not, which means the attribute 2 value range of the publication is out of that of the subscription.

B. Subscription Maintenance at the Grid Manager

A subscriber sends its subscription to its grid manager on demand, following the format shown in Figure 15. There are possibly many subscriptions in an information-dense area. Simply storing and transmitting all subscriptions would not only incur a large overhead in traffic and storage but also difficult to track the frequent subscription changes due to the user mobility and frequent user interest changes. On the other hand, selectively ignoring some of the subscriptions would compromise the system performance. In our system, the GM will aggregate the subscriptions by finding the minimum representative subscription set to represent all the subscriptions within the grid before recording them and sending them to the upper level.

Two subscriptions could share some common attributes, and the attribute set of a subscription could contain all the attributes of another subscription. In the second case, if the value ranges of the common attributes overlap each other to some extent, we could take the subscription which has all its attributes contained by the other subscription as the representative subscription of both subscriptions. However, if the value ranges of the common attributes do not have any intersection, then using one subscription to represent the other is not appropriate. We use an example to illustrate this aggregation principle.

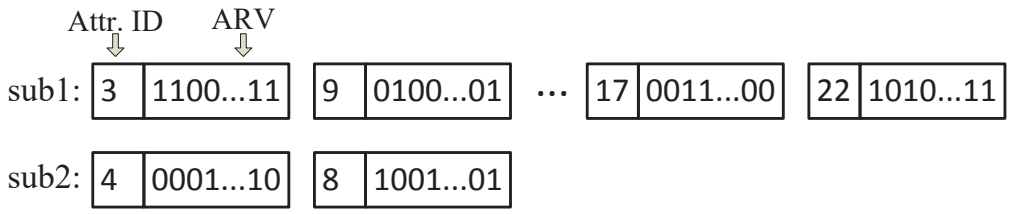


Figure 15: Example of a subscription request.

Suppose there are 2 subscriptions in a grid, SUB1: A and SUB2: A∧B∧C, where A, B and C are different attributes. According to our scheme, since all publications that contain the attribute A including the ones that also contain B and/or C will all be routed to this grid for further matching, thus taking SUB1 as the representative subscription, compared to otherwise having both SUB1 and SUB2, will help reduce the subscription information storage and control traffic without sacrificing the completeness of subscription information in this grid. Once receiving the information based on the aggregate filter, the GM will further match the information with individual subscription to determine if the information matches all the criteria of a subscriber. Thus aggregation reduces the message and data transmission between the ZM and GM, but does not sacrifice the accuracy requirement of each subscriber.

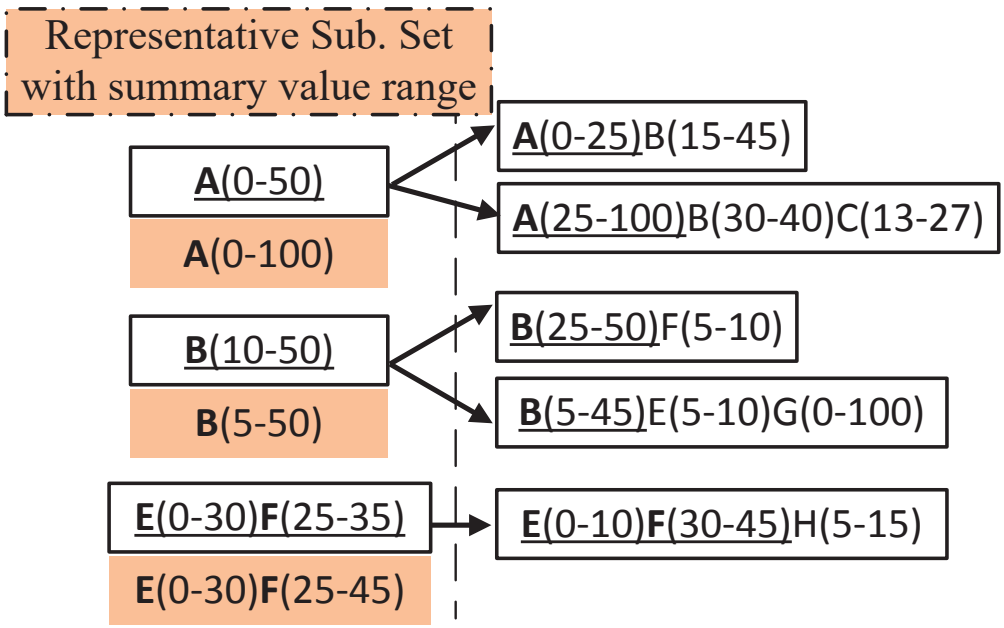


Figure 16: The summary forest with attribute summary value range in shade.

The subscription aggregating process can be realized through a summary tree, which is actually a forest containing several separate trees as shown horizontally in Figure 16. All the subscriptions of a tree will be represented by its root, and a tree node will contain all attributes of the root. There is also a summary range attached to each root shown as the shaded block in Figure 16, obtained by merging ('OR' operation) the value range of common attributes (underscored in Figure 16) of all the subscriptions on a tree. When determining if a node should be inserted into a tree, we will check if some of its attributes are the same as the root and if the attribute ranges overlap the current summary ranges. The summary ranges of all trees form the representative subscription set of the grid as shown on the left side of the dash line in Figure 16.

Algorithm 4 shows how to add a subscription into the current summary forest. On lines 3-10, a new subscription will become either the child or the parent of an existing root, depending on whether it contains all the attributes of a root or all of its attributes are contained by a root of the forest, with the value ranges of corresponding common attributes overlapping each others. Otherwise, the subscription will be made a new stand alone root, as shown on lines 12 and 16. On line 18, after inserting the new subscription, the summary value range attached to the root of the affected tree will be updated. Line 19 checks whether trees can be merged to one another to reduce the number of trees in the forest, i.e., the size of the forest, every time the summary value range of a tree is changed, by examining whether one tree root can be inserted as the child of another tree root following the similar criteria.

For illustration, suppose a grid has the following subscriptions with letters representing different attribute names: A(0-50), B(10-50), A(0-25)B(15-45), A(25-100)B(30-40)C(13-27), B(25-50)F(5-10), E(0-30)F(25-35), B(5-45)E(5-10)G(0-100), E(0-10)F(30-45)H(5-15). Applying them one after another with Algorithm 4 will generate a summary forest as shown in Figure 16.

Algorithm 5 works to remove a node in response to unsubscription. On lines 1-5, if the subscription to be deleted is the root of a tree, then this whole tree is removed with all the non-root nodes reinserted into the forest by applying algorithm 4 one by one. If the subscription is not a root, it is simply deleted from the tree as shown on lines 6-7. Then the affected trees will have their summary value ranges updated accordingly on line 9. Line 10 works similarly as the last line of Algorithm 4 to reduce the forest size.

Each GM will maintain a subscription summary forest, and updates the trees in response to the changes of subscription from individual subscribers within the grid. When a node wants to send a new subscription, modify or unsubscribe its existing subscription, it will send a message as in Figure 15 through on-demand light-weight geographic routing [43] to the GM. The GM

Algorithm 4 Adding a subscription s into the summary forest

```
1: if there are already nodes in the forest then
2:    for each root node R_i of the forest do
3:       if the subscription s contains all the attributes in R_i then
4:          if the summary value range of each attribute in R_i overlaps that of s
            then
5:             insert s as the child of R_i into the summary tree;
6:          end if
7:       else if R_i contains all the attributes of s then
8:          if each attribute value range of s overlaps the summary value range of
            the same attribute in R_i then
9:             make s the parent of R_i as the new root;
10:         end if
11:      else
12:         make s a new root of the forest;
13:      end if
14:   end for
15: else
16:   make s a new root of the forest;
17: end if
18: Adjust the summary value range of the affected tree.
19: Check whether the forest can be reduced by merging trees.
```

Algorithm 5 Removing a subscription s from the forest

```
1: if s is a root of the forest then
2:    delete the tree originated from root s;
3:    for each children node of s do
4:       apply Algorithm 4;
5:    end for
6: else
7:    delete s from the summary tree;
8: end if
9: Adjust the summary value range for each affected tree.
10: Check whether the forest can be reduced by merging trees.
```

will either insert or delete the subscription following the Algorithm 4 or 5. A new action may change the representative set. In many cases, however, *individual subscription changes will not lead to the change of the aggregated information summary at the root level of the tree.* This feature is very important. It helps to increase the stableness and significantly reduce the information maintenance overhead in a wireless environment with possible constant node movement and thus frequent subscription changes. The representative set is forged into a vector, named Grid Representative Set Vector (GRSV) as shown in Figure 17 by cascading each subscription from the representative set. The GRSV will be sent to the ZM upon its change to reduce the update overhead.

C. Subscription Maintenance at the Zone Manager

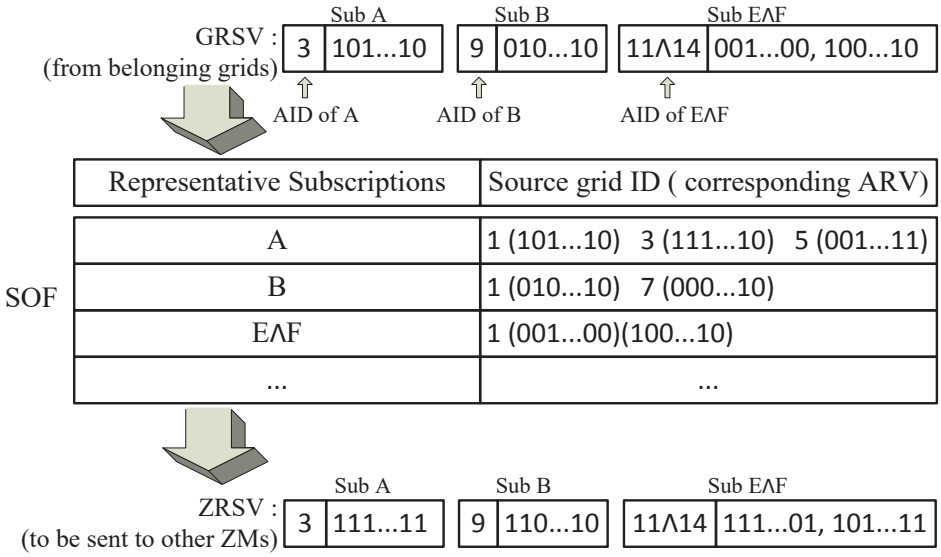


Figure 17: The ZM converts the GRSVs received from belonging grids into SOF, then converts it into ZRSV by summary tree scheme.

Each zone manager maintains a subscription origin form (SOF) generated based on the GRSVs sent by grids with subscriptions within its zone, as shown in Figure 17. The representative subscriptions from the grids will again be aggregated through the summary tree scheme similar to that at the grid level. We cascade each subscription of the resulting representative set to form a long vector - Zone Representative Set Vector (ZRSV). The ZRSVs are exchanged among ZMs to guide the publication distributions. The SOF will be updated if there is a GRSV update, but similar to the grid level aggregation, an individual update in SOF may not lead to ZRSV change. The aggregation helps to reduce

the message distribution and simplify the information matching process, which is more critical for dynamic wireless networks.

D. Match a Publication over Subscriptions

When a node generates a publication, it will send the data along with the publication ARVs describing the data to its GM. GM will perform a match within its grid by comparing the publication ARVs with its representative Sub set, i.e., the roots with summary ranges of the summary forest. If a root is matched, each of its tree node is further examined to precisely find the subscribers. The data will be forwarded to the identified subscribers. No matter local matches are found or not, GM will forward the data along with the P-ARVs and the grid ID to the zone manager. The ZM will match the P-ARVs against its SOF, to decide which grids within the zone to forward the data to for further matching at GM level. It also matches against all ZRSVs for other zones it maintains. The data along with the publication P-ARVs and the zone ID will be multicasted towards the centers of the zones that match this publication, where matching at finer level happens thereafter.

3.1.4 Main Results

We implement BRVST using NS2.34. The underlying routing scheme follows SOGR [43] and RSGM [44] for on-demand robust unicast and multicast respectively. 400 nodes are randomly distributed initially in a network region of size 1000m x 1000m to reflect the real-world mobile user density. In our default setting, the network is divided into 4 equal zones with 4 equal grids inside each. These numbers will vary when studying the impact of grid size on system performances. The node movement follows the improved Random Waypoint model [38]. The wireless channel propagation model is set to be TwoRayGround, and 802.11a is adopted as the MAC protocol with an average transmission range of 80m. Publications and subscriptions are generated by randomly selected nodes. Each publication or subscription has one to three attributes, which are randomly selected from a predefined set of 15. The range of an attribute is also randomly generated within a predefined range limit based on the attribute type. If not otherwise specified, the average node moving speed is set to 5 m/s, the Pub and Sub generation rates are both set to 200/minute, and the accuracy threshold α is set to 90%.

There is very limited number of studies closely related to ours. For performance references, we select two existing Pub/Sub schemes, DRIP and TAMA, that are partly comparable to our work. DRIP [45] (INFOCOM'08) is proposed for wireless networks which group nodes into Voronoi regions to manage the network, while BRVST introduces geographic zones to facilitate manage-

ment and information distribution. TAMA [47] (ICDCS'11) is a middleware for content matching, but is not specified for wireless networks. To be fair, we compare the impact of node mobility on the matching time for DRIP and BRVST in wireless environment, without including TAMA. The number of Voronoi regions for DRIP is also set to 16 under the same region area and node density. The management overhead involved for storing and transmitting publication and subscribe messages are compared among all three schemes.

Matching Time:

It is equally important for both the information provider and consumer to be served as fast as possible, so we evaluate the time for an emergent publication and an emergent subscription to get matched separately.

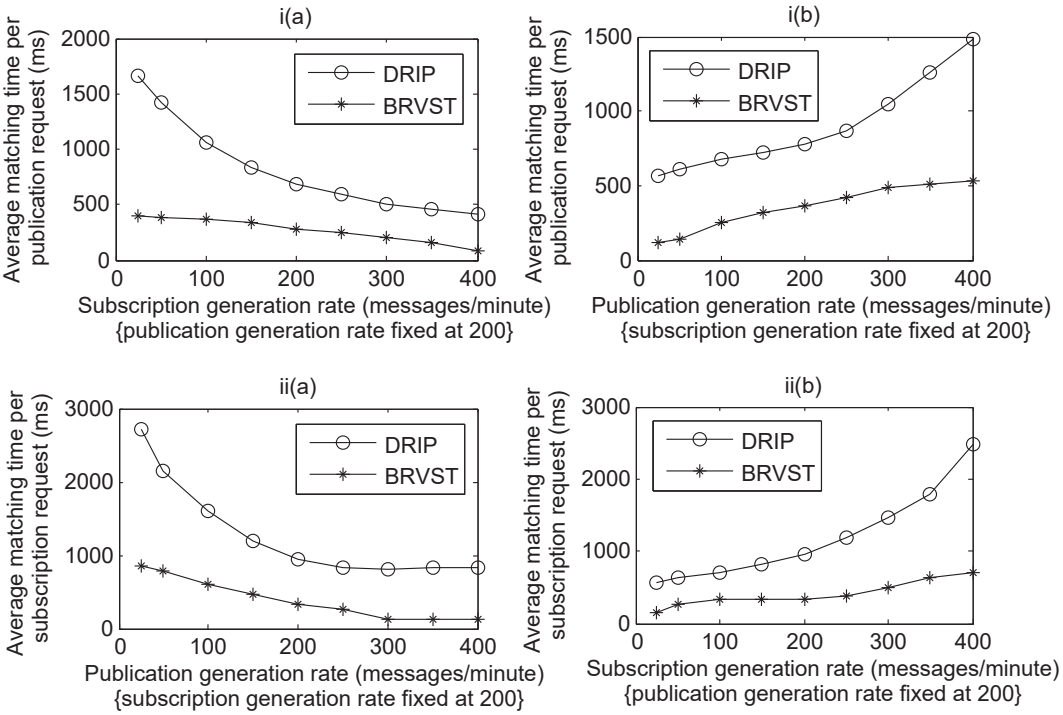


Figure 18: i(a) Matching time per Pub request as Sub rate increases; i(b) Matching time per Pub request as Pub rate increases; ii(a) Matching time per Sub request as Pub rate increases; ii(b) Matching time per Sub request as Sub rate increases.

For each newly published event, we evaluate the average time taken to match it with the subscribers. We allow publication to be matched with a later generated subscription and vice versa, so the delay is also affected by the subscription and publication generating frequency, as shown in Figure 18-i. In Figure 18-i(a) the publications rate is fixed at 200/min, while the subscription

rate is varied. In Figure 18-i(b), the subscription rate is fixed at 200/min, while the publication rate is varied. Similarly, we evaluate the average time duration for a newly generated subscription to match the publication in Figure 18-ii(a) and (b), with the subscription and publication rate fixed at 200/min respectively.

We can observe that BRVST has a much shorter average matching time as compared to DRIP under all test scenarios. A publication (or subscription) request has a shorter time to be matched when there is a higher subscription (or publication) rate as shown in Figures 18-i(a) and ii(a). The reduction of matching time reaches a limit, beyond which the matching time may slightly increase as a result of higher processing overhead.

On the contrary, as the publication (or subscription) rate becomes larger, the time to match a publication (or subscription) increases as a result of competitions, which deteriorate the average matching time, as shown in Figures 18-i(b) and ii(b). As DRIP involves network-wide broadcast to establish and maintain Voronoi regions, the matching time increases exponentially, while BRVST has only a sub-linear increasing time, which indicates its better scalability to system load.

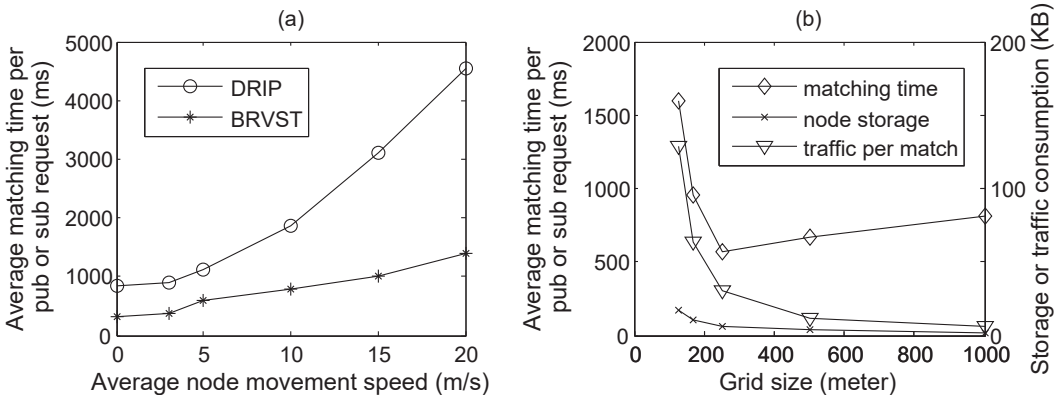


Figure 19: (a)Mobility impact on average matching time; (b)Grid size impact on BRVST's average matching time per message, system average node storage consumption and traffic volume incurred per match. The setting of grid size variation corresponds to the number of grids varying from 64 downto 1.

Figure 19-(a) tests and compares the reliability of BRVST and DRIP in terms of matching time performance under high node mobility, with the average node speed varying from 0 to 20m/s. The average matching time per message (including either the publication match or subscription match) of DRIP increases significantly as a result of its broadcast-based management

overhead. The delay becomes more severe when the average moving speed is higher than 10m/s, where nodes could move across regions within the average matching duration. Based on light-weight virtual management infrastructure, BRVST has much more stable performance in the mobility case.

In Figure 19-(b), the matching time is seen to first reduce with grid size and then increase. As the grid size increases, the number of grids decreases so does the number of zones, while the number of nodes in a grid increases. In a larger grid, messages are more likely to get matched within the grid or zone, and there are fewer other zones to check with. However when the grid size gets too large, messages need to interact over longer distance with GMs and ZMs. In addition, a large number of nodes also result in more filters in a grid which incurs a longer matching time.

System Maintenance Overhead:

We compare the overhead for storing and transmitting management messages at broker nodes and regular network nodes respectively. In Figure 20, the publication and subscription rates increase at the same speed.

In Figure 20-i(a), TAMA and BRVST both have lower storage overhead at regular nodes, as these nodes do not store publication and subscription information. Specifically, BRVST only requires each node to keep a few ID numbers which are very small in volume. With the need of storing a delay list of brokers and neighboring information, DRIP has much higher storage overhead, and the overhead increases quickly with the load.

In Figure 20-i(b), the storage overhead at brokers for all three schemes increase linearly with the load. DRIP has a much higher increasing rate with its need of maintaining information of both non-broker nodes and other brokers, as well as the subscriptions and publications of all the nodes in the network. Both TAMA and BRVST exploit range-based content representation to reduce the storage space. BRVST exploits space efficient aggregate scheme, so its storage space is 60% lower than that of TAMA.

We compare DRIP and BRVST on the overhead for transmission of management messages. In Figure 20-ii(a), the overhead of DRIP increases exponentially due to its inefficient broadcast mechanism. BRVST does not requirement significant overhead to maintain its zone and grid infrastructure, and only sends highly aggregated publish or subscribe information, thus it has a much lower transmission overhead.

In Figure 20-ii(b), when the message rate is low, BRVST and DRIP have similar matching overhead. At a higher load, however, the overhead of DRIP increases exponentially, while the overhead of BRVST is compensated as each publication can match multiple subscriptions with its aggregate subscription mechanism.

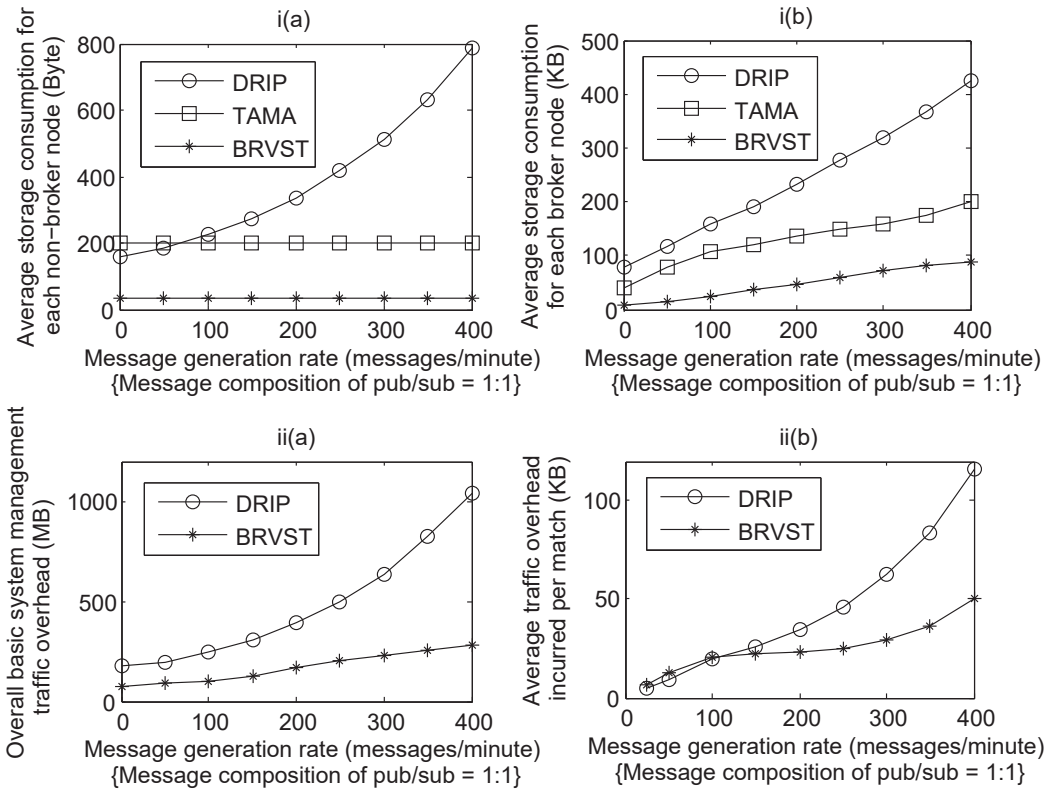


Figure 20: Storage consumption for i(a)non-broker node; i(b)broker node; i-i(a)Basic system traffic overhead; ii(b)Traffic overhead incurred per match.

In Figure 19-(b), as grid size increases, both the average node storage space and the traffic volume incurred for each match reduce. With a larger grid size, nodes are less likely to move out of the grid, thus the overhead associated with grid change will be lower. A larger grid also allows better information aggregation, thus reducing the matching traffic.

3.1.5 Conclusion

In this work, we present BRVST, an information content matching and forwarding engine in wireless network, which supports maximum flexibility in the expression of information content. The most valuable contributions of BRVST are its introduction of a novel attribute range vector that can accurately represent information content with extreme efficiency both in space and computationally, and the summary tree concept that enables effective extraction and aggregation of information. All these proposed structures help signifi-

cantly reduce storage and communication consumption as well as computation overhead, and ensure stable performance. Extensive simulations demonstrate that BRVST is reliable and scalable in large and dynamic wireless network conditions even under very high information load.

4 Back-End: reliable storage on the cloud

4.1 Work 4: Fast, Reliable & Space-Efficient Cloud Storage System

4.1.1 Introduction

Data from connected devices today are flowing into data centers with an unprecedented rate. More than half of the companies in the survey of global enterprise market currently store at least 100 TB of data and one-third expect their data to double in the next two to three years [3].

The cloud infrastructure enables low-cost and scalable file storage that provides global file access. Any file system must offer reliable storage whether through file duplication that requires more space but less computation complexity such as GFS [25] or through erasure coding that requires less space but more computation complexity such as RAID systems [13]. At the same time, raw data exhibit redundancy across files. This redundancy can be explored to reduce storage cost mainly for backup systems [32, 37, 49]. Using these techniques, data are divided into chunks and unique data chunks are stored once and referenced multiple times. Different from archival systems, cloud-based storage systems are required to support interactive user access with reasonable response time.

We propose a cloud-based file system named **SEARS**-Space Efficient And Reliable Storage system that exploits the deduplication technique to reduce storage and traffic cost as well as the erasure coding technique to increase both the data reliability and the file retrieval speed. Given a file, there are different ways to associate data chunks with available storage servers and retrieve data. Archive-based backup systems mainly care about storage efficiency and reliability. However, interactive cloud storage systems also care about file retrieval speed. To meet different application needs, we propose two data-server binding schemes with different performance goals: (1) faster file access speed or (2) higher storage efficiency.

We aim for SEARS to serve as a reference design for a flexible cloud storage framework that can support customized level of deduplication, modes of coding and server binding, and the mix of different modes. Its flexibility handles different application scenarios, from batch-centric archival to real-time.

4.1.2 SEARS Architecture

Figure 21 shows the SEARS system architecture consisting of storage server nodes operating in a data center. Users use SEARS as any file system by storing (or uploading) files to server nodes; and retrieving (or downloading) files from server nodes. Each user accesses SEARS through a designated storage

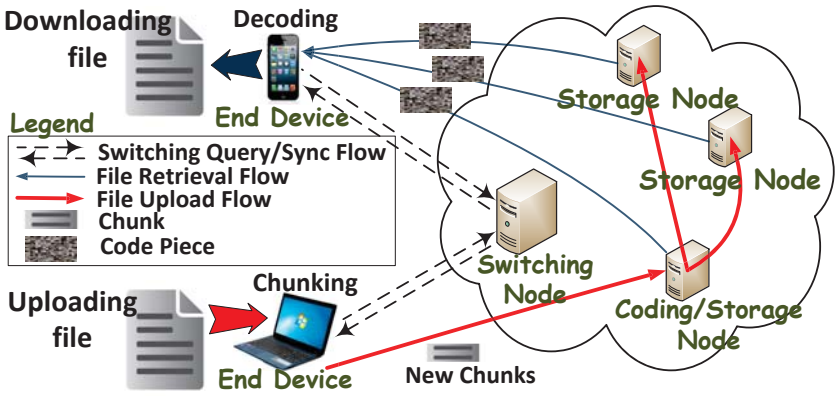


Figure 21: SEARS system overview. One end device (laptop) uploads a file where the file is chunked at end device and the meta-data for the file is uploaded to the **switching node** for the user. Unique chunks for the file missing from SEARS are sent to and coded at the **coding node** which is one of the server nodes in the cluster storing code pieces of the chunks for the file. Another end device (smart phone) downloads a file where code pieces of each unique chunk are retrieved from multiple storage nodes in SEARS concurrently.

server node we call **switching node** for the user and all the user's files. Each user end device is configured with host name or the IP address of the the user's switching node since it is the first node to reach SEARS. We consider a total of N nodes in SEARS divided into non-overlapping **clusters** of size n. The reason of forming cluster of nodes is due to the need of storing coded chunks at multiple nodes for reliability. We assign each cluster with a unique **cluster id**. We focus on the single data center configuration in this work. However, the concept of SEARS can be naturally extended to multiple data centers.

Content-based Chunking Operation: Before storing data, SEARS first removes redundant content. Files are divided into chunks and unique chunks are stored only once. We use content-based chunking to better capture redundancy [24]. Using smaller chunk size can result in more duplicate chunks thus achieving higher levels of deduplication. However, it also results in larger number of chunks and therefore larger overhead in meta-data management and reduced system performance. Furthermore, disk operations benefit from continuous data access, while smaller chunks lead to less efficient random access pattern. To balance the tradeoff, we choose average chunk size of 4 KB [18] [37] and enforce the minimum and maximum chunk sizes to be 1 KB and 8 KB respectively. For each chunk, we apply the 160-bit SHA-1 hash function [18] to generate a fixed-size hash value to serve as the **chunk id**.

File Storage Operation: Ahead of data storage, SEARS explores both intra-file and inter-file content redundancy and eliminates all redundant content. In the first step, SEARS eliminates intra-file redundancy as follows. Before a user file is uploaded into SEARS, the end device applies content-based chunking to the file, and generates chunk id for each chunk, and produces **file chunk-meta-data** for the file, which is composed of a sequence of entries for all chunks in the file and each entry consists of a chunk id and a cluster id specifying the cluster that stores the chunk. The file chunk-meta-data is stored at (1) the user's end device and uploaded to (2) the SEARS switching node serving the user. After this process, only non-repeating chunks will be kept so that intra-file redundancy can be eliminated.

A file in SEARS is represented by its file chunk-meta-data. Each unique chunk is stored as n code pieces in an n-node cluster. The user's switching node keeps a **chunk-meta-data-table** that stores one file chunk-meta-data for each file belonging to the user. As a chunk can appear in multiple files, we define the **reference count** for a chunk as the number of files in SEARS that the chunk appears in. The chunk reference count is updated as SEARS evolves with file addition, removal and update.

In the second step, SEARS eliminates inter-file redundancy across the set of nodes responsible for storing the file as follows. The user's switching node in SEARS removes chunk ids already in the set of nodes and forms a list of ids of missing chunks for the end device to upload directly to the set of storage nodes. This means only unique chunks that are not present in the set of SEARS nodes are uploaded from the user's end device. As a result, bandwidth between the user's end device and SEARS is only required to transfer non-redundant data.

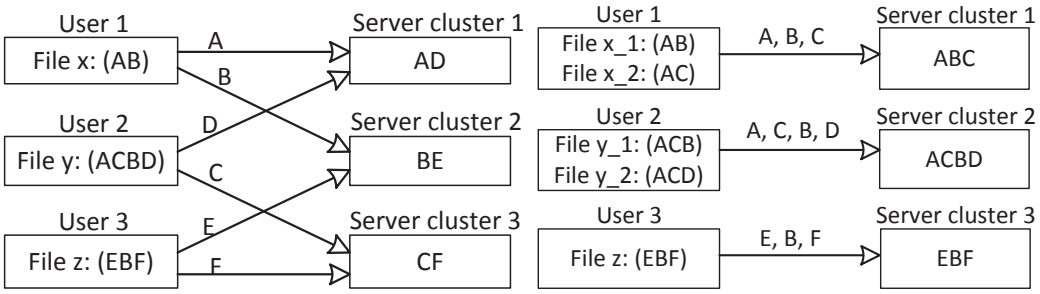
File Retrieval Operation: Whenever an end device retrieves a file from SEARS for the first time, the requesting end device does not have the file chunk-meta-data and the retrieval request is sent to the user's switching node. The switching node first sends back the file chunk-meta-data. The end device then checks the list of chunk ids in the file chunk-meta-data against the list of chunk ids already in its local storage, and determines the missing chunks needed to construct the file. The end device then only requests the missing chunks from SEARS.

File Chunk-Meta-Data Synchronization Operation: In the case when the end device and its responsible switching node in SEARScloud each has a version of the file chunk-meta-data, synchronization is required to resolve any conflicts. We follow the policy for the copy with the latest time-stamp to overwrite the one with an earlier time-stamp. We assume clock synchronization between the user's end device and SEARS is provided with mechanisms such as NTP [4].

Erasure Coding and Decoding Process: In SEARS, each unique chunk first reaches a node in the cluster that stores the code pieces of the chunk, we call **coding node**. The coding node then divides the chunk into k equal-sized pieces and codes it into n code pieces through (n, k) erasure coding with $n \geq k$. These n code pieces are associated with a cluster of n storage nodes and exactly one piece is stored in one node in the cluster. Note that any node in the cluster can serve as the coding node for a chunk to be stored at the cluster.

Whenever the user's end device requests a missing chunk in a file based on the file chunk-meta-data, it issues n concurrent requests to the n nodes in the cluster identified by the cluster id and as soon as k code pieces are received, it reconstructs the chunk and terminates any ongoing connection to the remaining $n - k$ nodes. This design benefits from parallel download of data to reduce SEARS response time as we show in Section 4.1.4.

4.1.3 Server Binding Schemes



(a) Chunk Level Binding (CLB) (b) User Level Binding (ULB)

Figure 22: Illustration of the binding schemes.

Consider SEARS nodes grouped into M clusters of size n. A file to be stored in SEARS is divided into chunks and each chunk is coded into n code pieces to be stored in a cluster. A key design question for SEARS is to determine how to associate data to clusters. We call this the **binding** process. Different applications have different requirements for cloud-based storage services, including fast file retrieval, small space usage in order to reduce storage cost. We design binding schemes across the spectrum of application requirements namely Chunk Level Binding and User Level Binding with examples in Figure 22(a) and 22(b) respectively.

Chunk Level Binding (CLB): For archival applications that runs in the background and demands storage efficiency, the binding process must offer system wide data deduplication. The **Chunk Level Binding (CLB)** scheme selects the best cluster to store each chunk. CLB is ideal for large media content repository like YouTube and NetFlix where users share the same or similar content. Each unique chunk entering SEARS is assigned to a cluster such that storage space of all clusters are evenly consumed as time passes. Note that all storage and retrieval requests must pass through the user's switching node. To distribute load evenly to clusters, we use a greedy algorithm to assign a chunk to the cluster with the largest amount of free storage space.

User Level Binding (ULB): For interactive applications with emphasis on promptness of file retrieval, the binding scheme must offer simplicity in chunk retrieval. The **User Level Binding (ULB)** scheme binds each user with a fixed cluster and simplifies file retrieval process as all chunks of this user are stored in the same cluster. Initially each user is assigned a fixed cluster. When storage capacity is exhausted at the cluster assigned for the user, a new cluster is assigned to future files from the user. This is equivalent to assigning a subset of user files to a separate user and only intra-set redundancy within the subset of files can be captured. ULB incurs at most one extra cluster id for a subset of user files, offers simple retrieval process but sacrifices space efficiency, as the chunks stored in different clusters belonging to different users (or even the same user) can not be exploited globally during the deduplication process.

The two binding schemes described so far offer different tradeoffs in space saving and file retrieval response time. However, they are just examples to showcase the flexibility in the design of SEARS . We design SEARS to be a powerful platform that use both deduplication and erasure coding in the best combination to fit various application needs.

4.1.4 Performance Evaluation

We evaluate the performance of our prototype implementation of SEARS over Amazon EC2 [1]. We generate a data set reflecting real-time data access of 10 users during a span of 3 weeks in 2014 containing three parts. (1) User Personal Data of 1.6 TB consisting of various common types of files from 10 users; (2) System Log of 132 GB consisting of major system log files (e.g. files under */var/log* directory) of Amazon EC2 Ubuntu server machines recorded every hour; and (3) System Backup Image of 3.5 TB consisting of the complete backup image files for Linux systems created once a day.

We evaluate SEARS in terms of storage usage with deduplication ratio and

time performance with the average file retrieval time. **Deduplication Ratio** is defined as the ratio of the total size of original files over the total space consumption for SEARS including the indexing overhead for storing them. This metric captures the combined effect of deduplication (reduce space usage) and erasure coding (increase space usage). **Average File Retrieval Time** is defined as the average time duration from the moment the user issues a request for a file to the moment the file is ready at end device. This involves downloading and decoding of all necessary chunks and reconstruction of the file from all chunks.

We employ 10 Amazon EC2 instances as driver machines to generate the log files, system backup images in addition to making users upload their own personal data. We fix cluster size at $n = 10$ thus use 10 EC2 instances for each cluster. We use $E = 20$ clusters.

We compare SEARS with the existing storage system R-ADMAD [33] which packs variable-length data chunks into fixed size objects of 8 MB which are encoded with erasure code and distributed among storage nodes called redundancy groups. To fairly evaluate R-ADMAD with SEARS, we implement it on EC2 cloud, and follow the same chunking process as SEARSas specified in Section 4.1.2 for all files in our data set to generate chunks of 4 KB average size. Furthermore, the same set of nodes are used for the SEARS cluster and the R-ADMAD redundancy group.

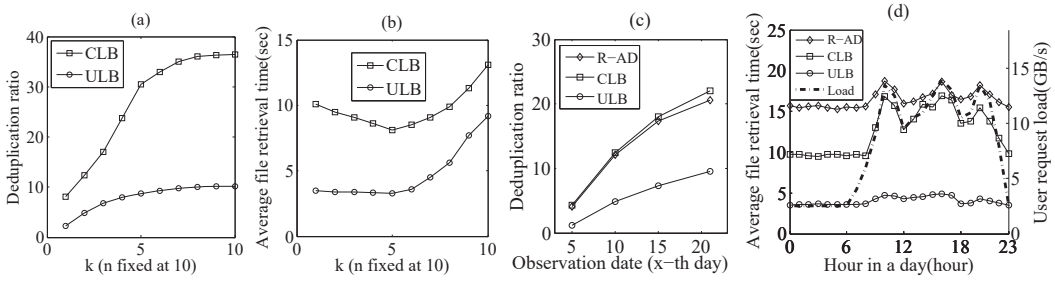


Figure 23: (a) k/n effect on Dedup ratio; (b) k/n effect on retrieval time; (c) Dedup ratio; (d) file retrieval time

Effect of k/n Ratio: The ratio k/n has profound performance impact on any scheme using erasure coding. To illustrate this, we fix n at 10 and vary k for the data set. As each chunk requires n/k times as much space as before the coding process, deduplication ratio increases with k as shown in Figure 23(a). Increases of k also lead to larger numbers of code pieces with smaller sizes for each chunk. This implies more parallel retrieval processes,

each with smaller bandwidth requirement. With smaller k ($k < 5$), both factors contribute to reduced chunk and file retrieval time. However, after k increase beyond a threshold, $k = 5$ for the data sets, the larger number of concurrent retrieval processes and the decoding process with more code pieces become the bottleneck and increase retrieval time as shown in Figure 23(b). CLB exploits redundancy across all chunks in all files and achieves a higher deduplication ratio. However, the process of searching for chunks across all clusters leads to the higher file retrieval time. On the other hand, ULB can only exploit intra-user redundancy which leads to a lower redundancy ratio. However all chunks in a file are easily retrieved from one cluster, which leads to the faster file retrieval time. We use $k = 5$ and $n = 10$ from now on.

Deduplication Ratio: To see how the ratio changes as data volume evolves over time, we plot the cumulative deduplication ratio on the 5^{th}, 10^{th}, 15^{th}, and 21^{st} day in Figure 23(c). The ratio improves for all schemes over time as data volume increases, for more redundancy can be exploited. It also shows deduplication ratio decreases in the order of CLB, R-ADMAD, and ULB. R-ADMAD is essentially same as CLB in data deduplication as it can exploit system wide redundancy just as CLB. But R-ADMAD uses slightly more space than CLB because of its indexing structure is more complex than CLB.

Time Performance: To examine interactive user experience, we replay the request pattern captured in the user personal data trace of our data set. We use 10 desktop machines residing in the eastern region of the US. Each desktop replays the file access trace for each of the 10 users. We report the file retrieval time for files accessed during each hour of the day averaged over 21 days over 10 users. To retrieve a file, the user's end device directly requests data chunks from 10 nodes storing the code pieces of each chunk in the three schemes. Figure 23(d) presents file retrieval time in relation to user request load averaged over each hour of the day over 21 days. Users' data request volume per hour in these figures reflect work activity during a day, that is, light activity at night (0:00 midnight to 8:00 am) and heavy and fluctuating activity for the rest of the day. ULB offers the fastest and relatively flat retrieval time because requests from the same user are handled by one cluster and there are no multiple requests for the same data chunk at the same time. CLB offers slower file retrieval than ULB, and large fluctuation during the working hours closely matching data request volume. This is because a unique chunk is stored only once in the entire system, and multiple users can request the same unique chunk at the same time, which leads to congestion at the cluster hosting the chunk in demand. R-ADMAD follows the data volume fluctuation during the day but with larger retrieval time than SEARS.

To compare with a commercial system, we note that downloading 3 MB files from the same set of 10 desktops residing in the eastern part of the US takes an average of 7 s from Amazon EC2 service in us-east-1 region [2]. With ULB in SEARS, the download time is 2.5 s throughout the day.

4.1.5 Conclusion and Future Work

We describe the design and implementation of a space efficient, data reliable and fast retrieving cloud-based storage system SEARS which integrates data deduplication and erasure coding. SEARS provides a flexible combination of various binding schemes to associate server nodes with data to be stored at different level based on application needs. Evaluation over Amazon EC2 shows that SEARS outperforms related systems with lower storage usage while ensuring fast and reliable data access.

As future work, we plan on examining the location of cluster nodes inside data centers to future improve data reliability and reduce retrieval time. We are evaluating the system with more data sets with additional metrics such as storage balance, file upload time and file retrieval success rate. Various system design parameters in SEARS and performance under flexible configuration of SEARS with multiple binding schemes, chunk size and erasure codes also need further investigation.

References

[1] Aws ec2. In *http://aws.amazon.com/ec2*.

[2] Cloud match. In *https://cloudharmony.com/speedtest*.

[3] Global enterprise big data trends:2013. In *http://www.microsoft.com/en-us/news/download/presskits/bigdata/docs/bigdata_021113.pdf*.

[4] Network time protocol. In *RFC 1305*.

[5] wikipedia.

[6] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006.

[7] E.J. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006.

[8] E.J. Candes and T. Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[9] E.J. Candes and M.B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[10] Emmanuel J. Candes and Justin Romberg. Practical signal recovery from random projections. In *SPIE Computational Imaging*, volume 5674, pages 76–86, 2005.

[11] Nuno Carvalho, Filipe Araujo, and Luis Rodrigues. Reducing latency in rendezvous-based publish-subscribe systems for wireless ad hoc networks. ICDCSW '06. IEEE Computer Society, 2006.

[12] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19:332–383, August 2001.

[13] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2), 1994.

[14] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, 2001.

[15] Jose A. Costa, Neal Patwari, and Alfred O. Hero, III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):39–64, February 2006.

[16] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inf. Theor.*, 55(5):2230–2249, May 2009.

[17] Min Ding, Fang Liu, Andrew Thaeler, Dechang Chen, and Xiuzhen Cheng. Fault-tolerant target localization in sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2007(1):19–19, January 2007.

[18] Wei Dong, Fred Douglis, Kai Li, Hugo Patterson, Sazzala Reddy, and Philip Shilane. Tradeoffs in scalable data routing for deduplication clusters. In *Proceedings of the 9th USENIX conference on File and stroage technologies*, FAST'11, pages 2–2, 2011.

[19] D. L. Donoho and Y. Tsaig. Fast solution of l1-norm minimization problems when the solution may be sparse. *Information Theory, IEEE Transactions on*, 54(11):4789–4812, 2008.

[20] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[21] D.L. Donoho, Y. Tsaig, I. Drori, and J-L Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 58(2):1094–1121, 2012.

[22] Qing Fang, Feng Zhao, and Leonidas Guibas. Lightweight sensing and communication protocols for target enumeration and aggregation. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 165–176.

[23] Chen Feng, W.S.A. Au, S. Valaee, and Zhenhui Tan. Compressive sensing based positioning using rss of wlan access points. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[24] Davide Frey, Anne-Marie Kermarrec, and Konstantinos Kloudas. Probabilistic deduplication for cluster-based storage systems. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 17:1–17:14, 2012.

[25] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, 2003.

[26] U. Grossmann, M. Schauch, and S. Hakobyan. Rssi based wlan indoor positioning with personal digital assistants. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on*, pages 653–656, 2007.

[27] Yan Guo, Bei Hua, and Lihua Yue. Energy-based target numeration in wireless sensor networks. In *Future Generation Communication and Networking (FGCN 2007)*, volume 2, pages 380–385, 2007.

[28] J. Haupt, W.U. Bajwa, M. Rabbat, and R. Nowak. Compressed sensing for networked data. *Signal Processing Magazine, IEEE*, 25(2):92–101, 2008.

[29] Zbigniew Jerzak and Christof Fetzer. Bloom filter based routing for content-based publish/subscribe. In *Proceedings of the second international conference on Distributed event-based systems*, DEBS '08, pages 71–81.

[30] Xiang Ji and Hongyuan Zha. Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2652–2661, 2004.

[31] A. Kushki, K.N. Plataniotis, and A.N. Venetsanopoulos. Kernel-based positioning in wireless local area networks. *Mobile Computing, IEEE Transactions on*, 6(6):689–705, 2007.

[32] Mark Lillibridge, Kave Eshghi, Deepavali Bhagwat, Vinay Deolalikar, Greg Trezise, and Peter Camble. Sparse indexing: Large scale, inline deduplication using sampling and locality. In *Proccedings of the 7th Conference on File and Storage Technologies*, FAST '09, pages 111–123, Berkeley, CA, USA, 2009. USENIX Association.

[33] Chuanyi Liu, Yu Gu, Linchun Sun, Bin Yan, and Dongsheng Wang. R-admad: high reliability provision for large-scale de-duplication archival storage systems. In *ICS*, 2009.

[34] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the*

15th annual international conference on Mobile computing and networking, MobiCom '09, pages 145–156.

[35] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, MobiCom '09, pages 145–156, New York, NY, USA, 2009. ACM.

[36] José Mocito, J. Alfonso Briones-García, Boris Koldehofe, Hugo Miranda, and Luís Rodrigues. Geographical distribution of subscriptions for content-based publish/subscribe in manets. In *Proceedings of the ACM/IFIP/USENIX Middleware'08*, pages 102–103. ACM, 2008.

[37] Athicha Muthitacharoen, Benjie Chen, and David Mazières. A low-bandwidth network file system. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 174–187, 2001.

[38] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, 2004.

[39] D. Needell and R. Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):310–316, 2010.

[40] Y.C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, volume 1, pages 40–44, 1993.

[41] N. Patwari and A.O. Hero. Manifold learning algorithms for localization in wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 3, pages 857–860, 2004.

[42] Robert E. Strom, Guruduth Banavar, Tushar Deepak Chandra, Marc Kaplan, Kevan Miller, Bodhi Mukherjee, Daniel C. Sturman, and Michael Ward. Gryphon: An information flow based approach to message brokering. *CoRR*, 1998.

[43] X. Xiang, X. Wang, and Z. Zhou. Self-adaptive on-demand geographic routing for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 1:99, 2011.

[44] Xiaojing Xiang, Xin Wang, and Yuanyuan Yang. Stateless multicasting in mobile ad hoc networks. *IEEE Transactions on Computers*, 59(8):1076 –1090, aug. 2010.

[45] Quan Yuan and Jie Wu. Drip: A dynamic voronoi regions-based publish/subscribe protocol in mobile networks. In *INFOCOM 2008*, pages 2110 –2118, april 2008.

[46] Bowu Zhang, Xiuzhen Cheng, Nan Zhang, Yong Cui, Yingshu Li, and Qilian Liang. Sparse target counting and localization in sensor networks based on compressive sensing. In *INFOCOM, 2011 Proceedings IEEE*, pages 2255–2263, 2011.

[47] Yaxiong Zhao and Jie Wu. Towards approximate event processing in a large-scale content-based network. ICDCS '11, pages 790–799.

[48] Yuanqing Zheng and Mo Li. P-mti: Physical-layer missing tag identification via compressive sensing. In *INFOCOM, 2013 Proceedings IEEE*, pages 917–925, April 2013.

[49] Benjamin Zhu, Kai Li, and Hugo Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, pages 18:1–18:14, Berkeley, CA, USA, 2008. USENIX Association.