NonStop: Continuous Multimedia Streaming in Wireless Ad Hoc Networks with Node Mobility

Baochun Li, *Member, IEEE,* Karen H. Wang, *Member, IEEE*

***Abstract*— Guaranteeing continuous streaming of multimedia data from service providers to the users is a challenging task in wireless ad hoc networks, particularly when node mobility is considered. The topological dynamics introduced by node mobility are further exacerbated by the natural grouping behavior of mobile users, which leads to frequent *network partitioning*. Network partitioning poses significant challenges to the provisioning of continuous multimedia streaming services in wireless ad hoc networks, since the partitioning disconnects many mobile users from the centralized streaming service. In this paper, we propose *NonStop*, a collection of novel middleware-based run-time algorithms that ensures the continuous availability of such multimedia streaming services, while minimizing the overhead involved. The network-wide continuous streaming coverage is achieved by partition prediction and service replication on the streaming sources, and assisted by distributed selection of streaming sources on regular mobile nodes and users. The proposed algorithms are validated by extensive results from performance evaluations.**

***Index Terms*— Multimedia streaming, service replication, wireless ad hoc networks.**

I. Introduction

The main appeal of wireless cellular and ad hoc networks is that they allow both user mobility and untethered connectivity. However, user mobility poses significant challenges to network operations such as routing, resource management, and Quality of Service (QoS) provisioning, especially when it comes to the QoS provisioning of multimedia services. The problem is more challenging in wireless ad hoc networks, since the mobile nodes constitute the communication infrastructure — a node acts as both a packet router and an end host. Node mobility leads to frequent disconnections of wireless links and dynamic changes of the network topology.

To improve network connectivity, many mobility prediction schemes have been proposed [1], [2], [3] to predict the future availability of wireless links, for the purpose of building more stable end-to-end connections at the network layer. However, there exist fewer studies on the effect of dynamic network topology on prominent problems at the application layer. One of such problems is the provisioning of *continuous streaming services of multimedia data* in wireless networks, especially ad hoc networks, such that streaming interruptions may be avoided or minimized as much as possible in the user experiences when consuming continuous media. The broad category of continuous media streaming include video-on-demand services and complex processing based on multimedia streaming, such as visual tracking.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are {bli,karen}@iqua.ece.toronto.edu.

With respect to such continuous streaming services that are critical to the Quality of Perception of a user working with a mobile node, we note the following two characteristics that serve as the baseline of this paper: (1) The continuous multimedia streaming service is inherently centralized; and (2) every node in the wireless network may need access to such a service, and once initiated, such a service should not be unnecessarily interrupted. Such an ideal situation is in sharp contrast when compared with the dynamics of wireless ad hoc networks with mobile users, in which case the network may partition into several disjoint “islands”, where the “islands” or *partitions* are completely disconnected from each other. For example, when mobile visitors to a museum show wish to access an audio/video stream (*e.g.*, information about upcoming events broadcast by the museum guides), the ad hoc network may become partitioned from time to time due to user mobility. Such partitioning may be more likely to occur when visitors tend to move towards different directions (e.g., different points of interests), approximately (and naturally) in *groups*. When the network partitions, those mobile users that are not in the same partition as the streaming service will suffer streaming interruptions or service unavailability until the partitions eventually merge. Depending on the degree of node mobility, such streaming interruptions are in the order of tens of seconds or even minutes.

Naturally, the *only* countermeasure to such streaming service interruptions is to make the service available by *replicating* to the separate partitions *a priori* before the partitioning occurs. There exist no other ways to solve the problem. However, such replications will not be performed without costs of bandwidth, and instant replications are impossible in the case of multimedia services. For a realistic example, for a one-minute audio/video streaming service of 100Kbps to user PDAs, approximately 750K bytes of data needs to be replicated. Over an end-to-end wireless connection with a capacity of 200K bytes per second, such replication may be performed in about four seconds. This requires that the replication event, as well as the timing of detecting partitions, needs to be four seconds ahead of the actual partitioning. In the museum scenario where mobile users move slowly, and with a timely algorithm to predict partitions well in advance, replicating the multimedia streaming data is still a feasible solution. In such solutions, *early predictions* and *high replication bandwidth* are required, the latter is contingent on the number of wireless hops between the nodes that replication may occur. With a shorter replicating distance, the available wireless bandwidth is higher [4]. In the best scenario of single-hop replications, the maximum bandwidth from the channel (*e.g.*, 54 Mbps or 6M bytes per second in the case of IEEE 802.11a) can be utilized for timely replications. Apparently, the success of replication-based solutions require that the time between prediction and partition events should be longer than the time required for replications.

In this paper, we present *NonStop*, a collection of middleware-based run-time algorithms that collectively guarantee the continuous availability of multimedia streaming services from the point of view of any mobile users in the network. Such continuous streaming availability is achieved by periodic monitoring of the network status, meticulous selection of replication candidates, as well as timely replications of the streaming service to the new partition, *before* network partitioning (and hence the service interruption) occurs. Such partition prediction at the global scale is achieved by implementing a prediction model based on the ideal case of *group mobility*, where mobile users exhibit correlated mobility patterns in their movements. Though such a grouping behavior of the mobile users has been observed in actual field trials of local area wireless networks [5], we show in simulation results that, even without such assumptions (with weak or no group mobility), our prediction model may still operate reasonably well, but without hard guarantees of continuous streaming.

We observe that network partitioning events (and service interruptions) are caused by group mobility. Consider an ad hoc network that consists of many mobility groups whose nodes are initially dispersed and intermixed. The distinct mobility pattern of each group causes the groups to separate, and the network eventually partitions. On the other hand, for a fully connected network to partition into completely disconnected components, such large-scale and structured topology changes can only be caused by correlated movements of a group of nodes, whereas independent movement of individual nodes can only cause random and sporadic link breakage. This insight agrees with the simulation results from [6], [7] which have shown that, the group mobility behavior of mobile users causes frequent network partitioning, and the resulting partitions are the separate mobility groups.

The novel contribution of *NonStop* is that, it captures the essential characteristics that represent such correlated mobility patterns, derives information about the changing network topology, and ultimately *predicts* future network partitioning events. With such predictions, the multimedia streaming services may be replicated onto the nodes of the anticipated partitions *in advance*, in order to ensure the continuous availability of the streams to existing in-progress nodes. The highlight of this work is the introduction of two algorithms that capture the network mobility status using *pattern recognition* algorithms in the *velocity space* of mobile nodes. We support our theoretical analysis and claims with extensive simulation results, which show that our algorithms perform effectively in real-world scenarios (even without the notion of group mobility), and comparatively better than alternative solutions. We believe this is a seminal contribution towards achieving a middleware-based hybrid service model that blends peer-to-peer communications and continuously available multimedia streaming services in wireless ad hoc networks.

The remainder of the paper is organized as follows. In Sec. II, we describe our group mobility model. *NonStop* is presented in Sec. III and Sec. IV. The performance of our algorithms are investigated in Sec. V. Sec. VI reviews related work and puts our work in a comparative perspective. Finally, we conclude the paper in Sec. VII.

II. System Models

We begin with the presentation of the *group mobility model* that we use throughout the paper. Most existing node mobility models used for the ad hoc networks are variations of the *random walk* model that defines individual node movements. Very few mobility models reflect group-based movements of nodes, one of which is the *Reference Point Group Mobility (RPGM)* model [6]. In this model, the nodes in the network are organized into mobility groups. Each mobility group has a logical group center, the *reference point*, which defines the movement of the entire group. The member nodes of the group are physically distributed in the vicinity of the *reference point*. The RPGM model describes the group membership of a mobile node by its physical displacement from the group's *reference point*. For example, at time t, the location of the ith node in the jth group is given by:

- Reference center location: $\mathbf{Y}_j(t)$
- Local displacement: $\mathbf{Z}_{j,i}(t)$
- Node location: $\mathbf{X}_{j,i}(t) = \mathbf{Y}_j(t) + \mathbf{Z}_{j,i}(t)$

The RPGM model generates the physical locations of the mobile nodes[1], but *it may not be used to accurately **identify** mobility groups*. As an illustration, Fig. 1(a) shows the snapshot of a network topology generated by the RPGM model: there are three mobility groups with common reference points (shown by the symbol $\circ$), and their coverage areas overlap — the member nodes (marked by their group symbols) are all intermixed. It is impossible to recognize the mobility groups, based on the *node physical location*. Since the nodes exhibit grouping behavior in their movements, naturally, a more distinguishing characteristic of nodes within the same mobility group is the *node velocity*. In other words, the mobility patterns are correlated based on the velocity of nodes. When the node velocities are plotted in the *velocity (v_x, v_y) space* as shown in Fig. 1(b), the mobility groups are most apparent.

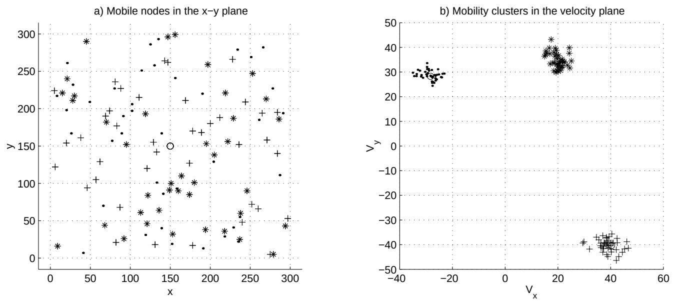

Fig. 1. Mobile nodes represented by their (a) physical locations; and (b) velocities

Therefore, we extend the RPGM model and propose a **Reference Velocity Group Mobility (RVGM) model** [8].

[1]The RPGM model is used in generating network topologies for ad hoc network simulations.

In our model, we represent each mobile node by its velocity $\mathbf{v} = (v_x, v_y)^T$, where v_x and v_y are the velocity components in the x and y directions. Each mobility group has a characteristic *mean group velocity*. The velocity of each member node may be slightly different from the characteristic mean group velocity. Therefore, the membership of ith node in the jth group is described by the addition of two velocity vectors:

– Mean group velocity: $\mathbf{W}_j(t) \sim \mathbf{P}_{j,t}(w)$
– Local velocity deviation: $\mathbf{U}_{j,i}(t) \sim \mathbf{Q}_{j,t}(u)$
– Node velocity: $\mathbf{V}_{j,i}(t) = \mathbf{W}_j(t) + \mathbf{U}_{j,i}(t)$

We further model the group velocity $\mathbf{W}_j(t)$ and the local velocity deviation of the member nodes $\mathbf{U}_{j,i}(t)$ as random variables each drawn from the distribution $\mathbf{P}_{j,t}(w)$ and $\mathbf{Q}_{j,t}(u)$, respectively. The distributions can be any arbitrary type, in order to model the various mobility patterns that may exist for different mobility groups and for the nodes within a mobility group. As an example, for the suitability of applying the Kalman Sequential Clustering algorithm for partition predictions (Sec. III-B), we may model the node velocity distribution in each mobility group by a *Gaussian distribution* parameterized by the *mean group velocity*:

$$\boldsymbol{\mu} = (\mu_{v_x}, \mu_{v_y})$$

and the variance:

$$\mathbf{S} = \begin{pmatrix} \sigma^2_{v_x} & \rho_{v_x v_y}\sigma_{v_x}\sigma_{v_y} \\ \rho_{v_y v_x}\sigma_{v_y}\sigma_{v_x} & \sigma^2_{v_y} \end{pmatrix}$$

where σ_{v_x} and σ_{v_y} represent the amount of variation in the x and y component of the node velocities of the group, respectively. The correlation coefficients $\rho_{v_x v_y}$ and $\rho_{v_y v_x}$ measure the relation between v_x and v_y. The v_x and v_y are often related due to the contour of the path the node is traveling, e.g., turning a corner or moving along a curve. The variance represents the amount of variation that exists in the member node velocities.

As we have seen in Fig. 1(b), the mobility groups form clusters in the velocity space where the member node velocities concentrate around the mean group velocity. We can take advantage of this cluster pattern to detect the mobility groups and identify the membership of every mobile node in an ad hoc network. In addition, using the mean group velocity and the variance parameters given by our RVGM model, we can calculate the movements and locations of the mobility groups, and then estimate the occurrence of network partitioning. The RVGM model provides the basis for the *NonStop* algorithms presented in this paper.

Throughput the paper, we consider wireless ad hoc networks where each mobile node has a unique identifier and is able to monitor its position (as a two-dimensional Cartesian coordinate) via GPS devices or through measurements of other signal sources. With the history of its successive locations, each node can measure its current velocity, expressed by $\mathbf{v} = (v_x, v_y)^T$. We further assume that each node is aware of the neighboring nodes within its transmission range, by means of periodic local broadcast of beacon signals.

The mobile nodes follow the group-based movements and their velocities obey the RVGM model[2]. A mobile node's *group membership is dynamic*, that is, it may switch mobility group at any time. Furthermore, to model realistic situations, each node does not know its mobility group nor the group memberships of other nodes.

With respect to the provisioning of multimedia streaming services, we consider one or more of these services in the network. The hosting nodes of such streaming services are referred to as the *service instances*, or simply the *servers*. Servers may be further replicated or subsequently terminated. A node becomes a server when it receives a service instance replication. The regular mobile users (nodes) in the network, hereafter referred to as *clients*, need continuous streaming access to at least one of the existing servers to obtain multimedia data. Finally, the clients piggyback their *identifier*, *location*, and *velocity* information when they initiate streaming requests with the servers.

III. NonStop: Partition Prediction

We present the core algorithm in *NonStop*: partition prediction. To guarantee continuous streaming service availability to all its clients in a partitionable ad hoc network, a server must replicate the service onto the partitioned nodes before they completely separate. This is illustrated in Fig. 2.

A single server S is serving two mobility groups (C_j and C_k) that are moving at different speeds and directions, and S belongs to group C_j. Initially in Fig. 2(a), the coverage areas of the two groups overlap, thus server S is accessible to all nodes. However, after the groups separate in Fig. 2(c), the nodes in group C_k are without service. Hence, S must replicate service when it passes the boundary of C_k's coverage area, as shown in Fig. 2(b).

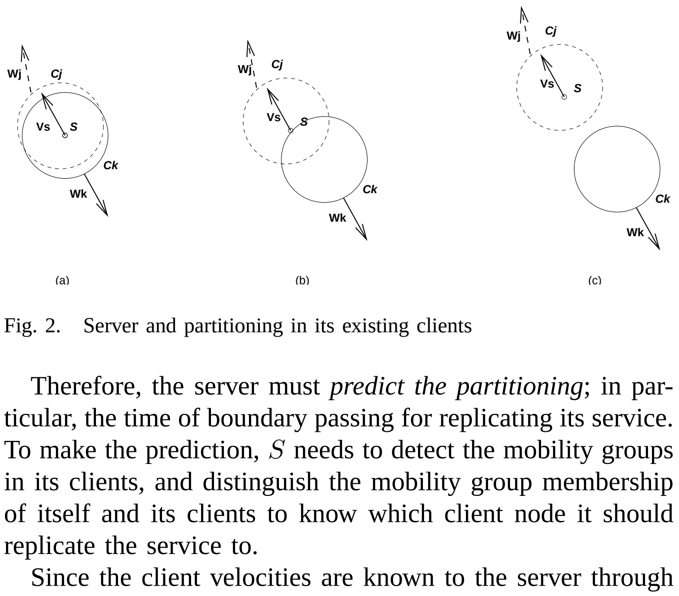

Fig. 2. Server and partitioning in its existing clients

Therefore, the server must *predict the partitioning*; in particular, the time of boundary passing for replicating its service. To make the prediction, S needs to detect the mobility groups in its clients, and distinguish the mobility group membership of itself and its clients to know which client node it should replicate the service to.

Since the client velocities are known to the server through piggybacking in the service requests, we propose a *centralized online algorithm* run by the server to determine these necessary information.

[2]We will show that the algorithms designed under such an assumption perform well even when the assumption is invalid.

A. Basic Sequential Clustering

We first propose to use the sequential clustering (SC) algorithm from *pattern recognition*, which exploits the cluster patterns formed by the mobile nodes to identify mobility groups [8] [9]. The SC algorithm classifies the mobile nodes into mobility groups based on the similarity between the node's velocity and the mean velocity of each mobility group.

The algorithm sequentially processes each mobile node x_i in three steps: (1) on the $(v_x,\, v_y)$ velocity plane, it measures the Cartesian distance between the velocity of x_i and the mean velocity of each group C_j, $(1 \leq j \leq m)$. (2) If x_i has the least distance from group C_k, given by $d(x_i, C_k)$, and the distance is less than a pre-set distance threshold α, then x_i is classified into group C_k; otherwise, a new group is created with node x_i as the first member. (3) Each time a new node is classified into a existing mobility group, the mean group velocity is recalculated. The algorithm is summarized in Table I.

TABLE I

SEQUENTIAL CLUSTERING (SC) ALGORITHM

```
$m = 1$
$C_m = \{x_1\}$
**for** $i = 2$ **to** end of data set
  Find $C_k$: $d(x_i, C_k) = \min_{1 \leq j \leq m} d(x_i, C_j)$
  **if** $d(x_i, C_k) > \alpha$ **and** $(m < m_{\max})$ **then**
     $m = m + 1$
     $C_m = \{x_i\}$
  **else**
     $C_k = C_k \cup \{x_i\}$
     recalculate the mean of $C_k$
  **end**
**end**
```

The SC algorithm boot-straps itself by classifying the first mobile node x_1 into the first mobility group C_1. The parameter $m_{\max}$ is the maximum number of groups allowed, which prevents too many mobility groups to be created. The details and the performance of the SC algorithm are discussed in [8] and [9].

The SC algorithm identifies the clusters formed by the mobile nodes in the velocity space as those depicted in Fig. 1(b). We can obtain: (1) the number of mobility groups from the number of clusters found; (2) the mean group velocities from the cluster centers; and (3) the mobility group membership of every mobile node from which cluster its node velocity belongs to. Since the algorithm presented in Table I is the baseline in our studies, it is henceforth referred to as the *Basic Sequential Clustering Algorithm* (BSCA).

B. Sequential Clustering based on Kalman Filter Estimation

To reduce the sensitivity to the order of data presentation and to identify clusters of various shapes, we take an *estimation* approach in the sequential clustering, where an extra estimation component using a *suboptimal Kalman filter* is added (Fig. 3).

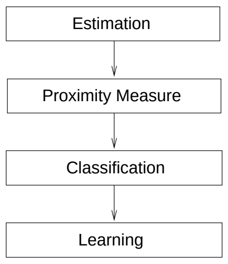

Fig. 3. Main steps in a Kalman Filter estimation clustering algorithm

We first describe the general idea of such an estimation-based sequential clustering. The RVGM model states that, every client node belongs to a mobility group and the velocities of each mobility group follow a Gaussian distribution. Hence, rather than vectors forming n clusters in the two-dimensional velocity space, the node velocity data points ($\mathbf{x}_i$'s) can be viewed as the outcomes of trials governed by a mixture of n Gaussian probability densities:

$$P(\mathbf{x}_i) = \sum_{j=1}^{n} P(C_j) P(\mathbf{x}_i | C_j; \boldsymbol{\mu}_j, \mathbf{S}_j)$$

where $\boldsymbol{\mu}_j$ and $\mathbf{S}_j$ are the mean and the variance-covariance matrix of the jth Gaussian distribution C_j. For convenience, we assume all $P(C_j)$s are equiprobable:

$$P(\mathbf{x}_i) = \frac{1}{n} \sum_{j=1}^{n} P(\mathbf{x}_i | C_j; \boldsymbol{\mu}_j, \mathbf{S}_j).$$

For our classification purpose, for each observed data point $\mathbf{x}_i$, the conditional probability $P(\mathbf{x}_i|C_j)$ can be calculated for every probability density C_j given its current $\boldsymbol{\mu}_j$ and $\mathbf{S}_j$. If $P(\mathbf{x}_i|C_q) = \max_{j=1,\ldots,n} P(\mathbf{x}_i|C_j)$ is greater than some threshold α, then $\mathbf{x}_i$ is assigned to C_q, and the parameters $\boldsymbol{\mu}_q$ and $\mathbf{S}_q$ are updated accordingly. However, the number of Gaussian distributions and their $\boldsymbol{\mu}$ and $\mathbf{S}$ are *not known a priori*, and must be determined with each classification. Thus, our cluster identification problem becomes an estimation problem: given the observations, estimate the mean $\boldsymbol{\mu}$ and covariance $\mathbf{S}$ of each Gaussian distribution. In this algorithm, a suboptimal Kalman filter is used to estimate the parameters of the Gaussian distribution of each mobility group.

B.1 Dynamic Stochastic Systems

The premise for using a Kalman filter (or a suboptimal version of the filter) is to model the Gaussian distribution of each mobility group as a *dynamic stochastic system*. The details are presented as follows. Each mobility group's Gaussian distribution or each *class* (using a more familiar clustering terminology) under estimation has dynamics associated with it since its state (mean $\boldsymbol{\mu}$) evolves as new elements are acquired. Each class is a stochastic random variable that has uncertainty or variance $\mathbf{S}$. Furthermore, each data point used in the classification may or may not be a member of the class, thus may provide valid or false inference about the state of the class.

Hence, we model each class under estimation as a dynamic stochastic system, and describe the system model using conventional mathematical notations. Specifically, for class C_i, $\mathbf{x}_k^i$ is the state of the class which represents $\boldsymbol{\mu}$, the mean of the Gaussian distribution, at discrete time step k:

$$\mathbf{x}_k^i = \mathbf{x}_{k-1}^i + \mathbf{u}_{k-1}^i \tag{1}$$

where $\mathbf{u}_k^i$ models the system noise (variance of the Gaussian distribution) due to disturbances. It is a white, zero-mean Gaussian random sequence:

$$E[\mathbf{u}_k^i] = 0,\ E[\mathbf{u}_k^i {\mathbf{u}_k^i}^T] = \mathbf{Q},\ E[\mathbf{u}_k^i {\mathbf{u}_j^i}^T] = 0,\ \forall k \neq j.$$

The expected initial state $\hat{\mathbf{x}}_0^i$ and its associated variance $\mathbf{P}_0^i$ are known:

$$\hat{\mathbf{x}}_0^i = E[\mathbf{x}_0^i], \qquad \mathbf{P}_0^i = E[(\mathbf{x}_0^i - \hat{\mathbf{x}}_0^i)(\mathbf{x}_0^i - \hat{\mathbf{x}}_0^i)^T].$$

The data point $\mathbf{y}_k$ is an imperfect observation of $\mathbf{x}_k^i$:

$$\mathbf{y}_k = \mathbf{x}_k^i + \mathbf{w}_k^i \tag{2}$$

due to the measurement error $\mathbf{w}_k^i$ which models the variability of the observation. The $\mathbf{w}_k^i$ is also a white, zero-mean Gaussian random sequence:

$$E[\mathbf{w}_k^i] = 0,\ E[\mathbf{w}_k^i {\mathbf{w}_k^i}^T] = \mathbf{R},\ E[\mathbf{w}_k^i {\mathbf{w}_j^i}^T] = 0,\ \forall k \neq j.$$

The system noise and measurement error are uncorrelated

$$\forall k\, \forall j, \qquad E[\mathbf{u}_k^i \mathbf{w}_j^i] = 0.$$

Also, $\mathbf{Q}$ and $\mathbf{R}$ are the variance of the system noise and the measurement error, respectively.

B.2 Kalman Filter

For a stochastic system, given its system model (Eq. (1) and (2)) and noise-corrupted measurements ($\mathbf{y}_k$), a Kalman filter can be used to estimate the system state, $\hat{\mathbf{x}}_k^i$ ($\wedge$ indicates it is an estimate of $\mathbf{x}_k^i$) such that the mean-square estimation error $\mathbf{P}_k^i$ is minimized,

$$\mathbf{P}_k^i = E[(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)(\mathbf{x}_k^i - \hat{\mathbf{x}}_k^i)^T].$$

The Kalman filter is a linear data processing algorithm that uses all the available information about the system: a) knowledge of the system dynamics model, b) measurements of all precisions, c) statistical information about system and measurement noise, and d) initial system state, to generate an estimated system state with the minimum estimation error.

The algorithm of the **discrete time Kalman filter**[3], for the stochastic system defined by Eq. (1) and (2), is expressed by the five steps shown in Fig. 4.

B.3 Suboptimal Kalman Filter

Since our classes or Gaussian distributions can be modeled as dynamic stochastic systems, and the node velocity data points are the observed measurements of the Gaussian distributions, we attempt to use the Kalman filter to estimate the Gaussian distribution of each mobility group.

We first examine the requirements for applying the Kalman filter. Our system model (Eq. 1 and 2) satisfies the required assumptions of the Kalman filter. For the required initial variables, $\mathbf{x}_0$, $\mathbf{P}_0$, $\mathbf{Q}$ and $\mathbf{R}$, we have $\mathbf{x}_0$ as the first data point processed and the associated variance $\mathbf{P}_0$ can be set to some reasonable estimate. However, we do not have the variances of the system and measurement noise, $\mathbf{Q}$ and $\mathbf{R}$. Therefore, a **suboptimal Kalman filter** is used to estimate the parameters of each Gaussian distribution.

1. State estimate prediction (propagation)
   $\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1}$
2. Covariance estimate prediction (propagation)
   $\mathbf{P}_k^- = \mathbf{P}_{k-1} + \mathbf{Q}$
3. Weighted gain calculation
   $\mathbf{K}_k = \frac{\mathbf{P}_k^-}{\mathbf{P}_k^- + \mathbf{R}}$
4. State estimate update
   $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{x}}_k^-)$
5. Covariance estimate update
   $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_k^-$

Fig. 4. Equations of the discrete time Kalman Filter

We initialize $\mathbf{Q}_0$ and $\mathbf{R}_0$, and estimate the $\mathbf{Q}_k$ and $\mathbf{R}_k$ simultaneously with the system state $\mathbf{x}_k$. Since $\mathbf{Q}_k$ is the variance of the system noise, it is estimated as the iterative mean of incremental difference between successive system state estimate $\hat{\mathbf{x}}_k$,

$$\hat{\mathbf{Q}}_k = \frac{1}{k}\sum_{j=1}^{k} \mathbf{q}_j \mathbf{q}_j^T, \qquad \text{where} \qquad \mathbf{q}_j = \hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j-1}$$

Similarly, $\mathbf{R}$ is the variance of the measurement error, it is calculated as the iterative mean of the discrepancy between the measurement $\mathbf{y}_k$ and the propagated state estimate $\hat{\mathbf{x}}_k^-$ at every time instant.

$$\hat{\mathbf{R}}_k = \frac{1}{k}\sum_{j=1}^{k} \mathbf{r}_j \mathbf{r}_j^T, \qquad \text{where} \qquad \mathbf{r}_j = \mathbf{y}_j - \hat{\mathbf{x}}_j^-$$

The outline of the suboptimal Kalman filter algorithm is illustrated in Fig. 5. For clarity, the original 5 steps of the Kalman filter are labeled in the figure.

At each time step k, with a measurement of the class state $\mathbf{y}_k$, the suboptimal Kalman filter estimates the class state $\hat{\mathbf{x}}_k$, the system and measurement variances $\hat{\mathbf{Q}}_k$, and $\hat{\mathbf{R}}_k$, and calculates the estimation error $\hat{\mathbf{P}}_k$, and further propagates them to the next time step.

B.4 Algorithm Outline

The complete outline of the sequential clustering algorithm using the suboptimal Kalman filter estimation is given in Table II. It has the similar algorithmic structure as the basic sequential clustering algorithm (BSCA) shown in Table I, but with one extra step of the suboptimal Kalman filter added.

Line 1 to 3 bootstrap the clustering algorithm by classifying the first data point, and set up the initial state variables required by the suboptimal Kalman filter. For subsequent data points ($\mathbf{y}_t$'s), the algorithm sequentially processes each data point through the *estimation*, *proximity measure*, *classification*, and *learning* steps.

[3]Detailed and rigorous mathematical explanations of the Kalman filter can be found in reference [10] and [11].

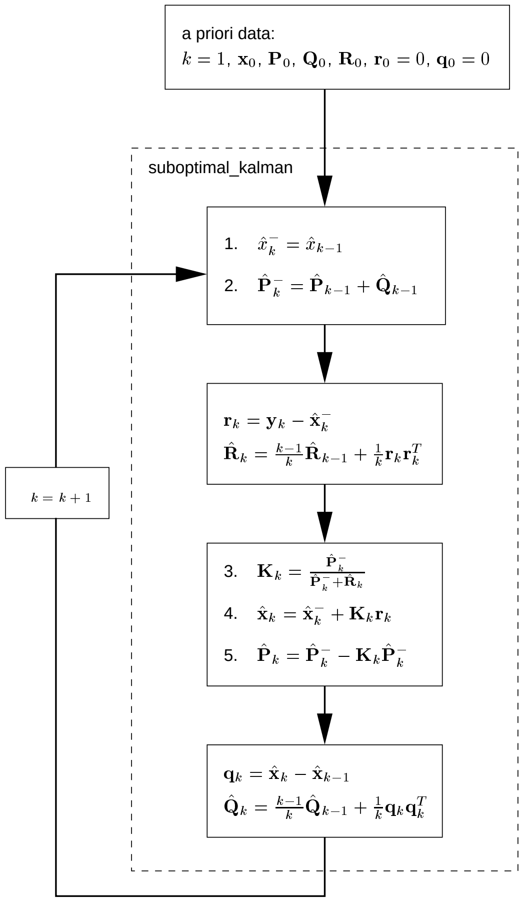
a priori data:
$k=1$, $\mathbf{x}_0$, $\mathbf{P}_0$, $\mathbf{Q}_0$, $\mathbf{R}_0$, $\mathbf{r}_0=0$, $\mathbf{q}_0=0$
suboptimal_kalman
1. $\hat{x}_k^- = \hat{x}_{k-1}$
2. $\hat{\mathbf{P}}_k^- = \hat{\mathbf{P}}_{k-1} + \hat{\mathbf{Q}}_{k-1}$
$\mathbf{r}_k = \mathbf{y}_k - \hat{\mathbf{x}}_k^-$
$\hat{\mathbf{R}}_k = \frac{k-1}{k}\hat{\mathbf{R}}_{k-1} + \frac{1}{k}\mathbf{r}_k\mathbf{r}_k^T$
3. $\mathbf{K}_k = \frac{\hat{\mathbf{P}}_k^-}{\hat{\mathbf{P}}_k^- + \hat{\mathbf{R}}_k}$
4. $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\mathbf{r}_k$
5. $\hat{\mathbf{P}}_k = \hat{\mathbf{P}}_k^- - \mathbf{K}_k\hat{\mathbf{P}}_k^-$
$\mathbf{q}_k = \hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k-1}$
$\hat{\mathbf{Q}}_k = \frac{k-1}{k}\hat{\mathbf{Q}}_{k-1} + \frac{1}{k}\mathbf{q}_k\mathbf{q}_k^T$
$k=k+1$

Fig. 5. Suboptimal Kalman Filter

For data point $\mathbf{y}_t$, Line 7 to 9 use the suboptimal Kalman filter to **estimate** the Gaussian probability density $P(\mathbf{y}_t|C_i)$, for $i=1$ to n, where n is the number of classes that currently exist. The suboptimal Kalman filter balances the uncertainty about the class state and the possible error introduced by the data point, hence determines the likelihood the data point belonging to the class while minimizing the uncertainty about the class state or the estimation error. The estimated values of the class state variables are temporarily stored in $(\hat{\mathbf{x}}_{k_i}^{i*}, \hat{\mathbf{P}}_{k_i}^{i*}, \hat{\mathbf{Q}}_{k_i}^{i*}, \hat{\mathbf{R}}_{k_i}^{i*})$

The **proximity measure** (Line 11) is calculated from the Gaussian pdf $P(\mathbf{x}_i|C_j)$, with the estimated parameter $\hat{\mathbf{x}}_{k_i}^{i*}$ as the mean and $\hat{\mathbf{P}}_{k_i}^{i*}$ as the variance.

The **classification** rule (Line 13) determines which class the data point $\mathbf{y}_t$ belongs to. If the maximum probability of $\mathbf{y}_t$ belonging to any of the existing class is less than the threshold α, a new class is created (Line 14 to 16); otherwise, $\mathbf{y}_t$ is classified to the most probable class. With the newly acquired data point, the class **learns** its new state by updating its class state variables to the estimated $(\hat{\mathbf{x}}_{k_i}^{i*}, \hat{\mathbf{P}}_{k_i}^{i*}, \hat{\mathbf{Q}}_{k_i}^{i*}, \hat{\mathbf{R}}_{k_i}^{i*})$ and propagate them to its next classification instant (Line 19).

Compared to BSCA, the computation cost of this algorithm is higher while still manageable, since the suboptimal Kalman filter involves a few multiplication and one inversion of the

TABLE II
THE KALMAN SC ALGORITHM

```
1.   $n = 1$
2.   $C_n = \{\mathbf{y}_1\}$, $k_n = 1$, initialize
3.   $\hat{\mathbf{x}}_{k_n}^n = \mathbf{y}_1$, $\hat{\mathbf{P}}_{k_n}^n = \mathbf{P}_0$, $\hat{\mathbf{Q}}_{k_n}^n = \mathbf{Q}_0$, $\hat{\mathbf{R}}_{k_n}^n = \mathbf{R}_0$
4.
5.   **for** $t = 2$ **to** end of data set
6.
7.       **for** $i = 1$ **to** $n$
8.           $(\hat{\mathbf{x}}_{k_i}^{i*}, \hat{\mathbf{P}}_{k_i}^{i*}, \hat{\mathbf{Q}}_{k_i}^{i*}, \hat{\mathbf{R}}_{k_i}^{i*})$ =
                 **suboptimal_kalman**($\mathbf{y}_t$)
9.       **end**
10.
11.      Find $C_q$: $P(\mathbf{y}_t|C_q; \hat{\mathbf{x}}_{k_q}^{i*}, \hat{\mathbf{P}}_{k_q}^{i*}) =$
           $\max_{1 \le i \le n} P(\mathbf{y}_t|C_i; \hat{\mathbf{x}}_{k_i}^{i*}, \hat{\mathbf{P}}_{k_i}^{i*})$
12.
13.      **if** $P(\mathbf{y}_t, C_q) < \alpha$ **and** $(n < n_{\max})$ **then**
14.          $n = n + 1$
15.          $C_n = \{\mathbf{y}_t\}$, $k_n = 1$, initialize
16.          $\hat{\mathbf{x}}_{k_n}^n = \mathbf{y}_t$, $\hat{\mathbf{P}}_{k_n}^n = \mathbf{P}_0$, $\hat{\mathbf{Q}}_{k_n}^n = \mathbf{Q}_0$, $\hat{\mathbf{R}}_{k_n}^n = \mathbf{R}_0$
17.      **else**
18.          $C_q = C_q \cup \{\mathbf{y}_t\}$, $k_q = k_q + 1$, commit
19.          $(\hat{\mathbf{x}}_{k_q}^i, \hat{\mathbf{P}}_{k_q}^i, \hat{\mathbf{Q}}_{k_q}^i, \hat{\mathbf{R}}_{k_q}^i)$ = $(\hat{\mathbf{x}}_{k_q}^{i*}, \hat{\mathbf{P}}_{k_q}^{i*}, \hat{\mathbf{Q}}_{k_q}^{i*}, \hat{\mathbf{R}}_{k_q}^{i*})$
20.      **end**
21.
22.  **end**
```

2×2 matrix[4].

For simplicity, in the remainder of this paper, we refer to the algorithm as sequential clustering based on Kalman filter estimation (or simply *Kalman SC*), even though it in fact uses a suboptimal Kalman filter. We postpone performance comparisons between the BSCA and Kalman SC to Sec. V.

C. Service Replication

With the mobility groups accurately identified, the server S can calculate the time of service replication. S determines the two groups C_j and C_k are moving at the velocities of $\mathbf{W}_j$ and $\mathbf{W}_k$, and itself belongs to group C_j, while its velocity is $\mathbf{V}_S$. Then, the *effective velocity*, $\mathbf{V}_{S\leftrightarrow C_k}$, at which group C_k is separating from S is:

$$\mathbf{V}_{S\leftrightarrow C_k} = \mathbf{V}_S + (-\mathbf{W}_k), \quad \mathbf{V}_{S\leftrightarrow C_k} = (V_{x,S\leftrightarrow C_k}, V_{y,S\leftrightarrow C_k})$$

Fig. 6 illustrates a geometric view of S and the mobile nodes in C_k. The server can determine this view from the client's location information piggybacked on the service requests. The goal is for S to replicate its multimedia streaming service to a node in the departing group C_k, and to start the replication process as soon as possible. Therefore, the node *closest from S in the direction of $\mathbf{V}_{S\leftrightarrow C_k}$* should be selected. To find the closest node, let vector $\mathbf{L}_{x,S}$ denotes the distance vector between S and a node x, and $\theta_{x,S}$ is the angle between vectors $\mathbf{L}_{x,S}$ and $\mathbf{V}_{S\leftrightarrow C_k}$. Only the nodes *"ahead"* of S in the direction of $\mathbf{V}_{S\leftrightarrow C_k}$, with $|\theta_{x,S}| < 1/2\pi$ should be considered (shown as the shaded nodes in Fig. 6). Projecting the distance

[4]A more efficient form of the Kalman filter where the inverse matrix is propagated can be found in [10].

vector $\mathbf{L}_{x,S}$ of each node x onto $\mathbf{V}_{S\leftrightarrow C_k}$, the closest node from S has the smallest projection proj$(\mathbf{L}_{x,S})$. In Fig. 6, it is node x_6. Thus, the server selects

$$X_{C_k} = \text{the node in } C_k \text{ with } \min(\text{proj}(\mathbf{L}_{x,S}))$$

as the target node in group C_k for service replication. The corresponding time of service replication is

$$T_{S,C_k} = \frac{\text{proj}(\mathbf{L}_{X_{C_k},S})}{\sqrt{V^2_{x,S\leftrightarrow C_k} + V^2_{y,S\leftrightarrow C_k}}}.$$

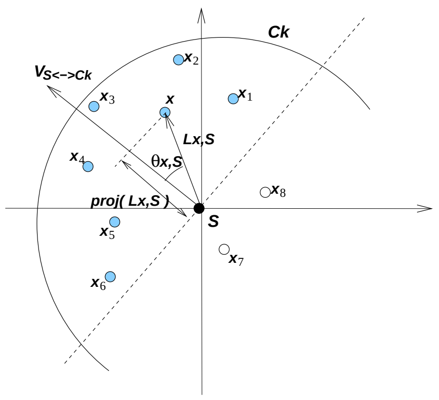

Fig. 6. Timing and node selection in service replication

Once T_{S,C_k} is computed, we may use it as an estimate of the duration between the timing of prediction and partitioning events. If this estimate is larger than the time required to replicate the streaming service (assuming known sizes of the media streams), we initiate the replication process. Otherwise, the replication may fail to complete before partitioning occurs, and we abort the replication attempt. If a replication should be initiated, the server first checks if the target node is reachable, since the node may have changed its movement path. Second, the server verifies the target node is still a client, because the node can switch to a better server (discussed in Sec. IV) after the replication is scheduled. If both conditions hold, the server initiates a replication to the target node. After the replication, the target node is referred to as the *child* server, corresponding to its *parent* server S.

IV. NonStop: Server Selection

In the previous section, we have addressed the problem of streaming service continuity during network partitioning from the point of view of the streaming services. On the other hand, from the point of view of the client nodes (users), a streaming server that was previously connected may become unreachable when the network partitions. Hence, it is essential for the clients to discover and select a reliable streaming service instance with a stable connectivity that is unlikely to be disconnected.

According to the RVGM mobility model, nodes of the same mobility group have similar velocities, and hence maintain a relatively steady distance from each other. Naturally, a node has stable connectivity with other nodes of its own mobility group. In addition, the mobility group is unlikely to be divided during the partitioning, and the server side service replication algorithm ensures that a service instance is always available in a disconnected partition. Therefore, the search for nodes of stable connectivity may be reduced to the problem of finding nodes of same group.

However, in a spontaneously deployed ad hoc network with no pre-configurations, the mobile node has no prior knowledge about the mobility groups, let alone its and other nodes' group memberships. To further complicate the case, the mobility group membership of a node can change dynamically, as the mobile host may decide to change its course of movement. Although the SC algorithms (either BSCA or Kalman SC) can identify mobility groups at run-time, it is centralized in nature and hence impractical for every client to run, because of the prohibitive communication cost of collecting velocities globally from the other nodes. We need a *fully distributed algorithm* for the individual client to find nodes of its mobility group at run-time, which only requires local information.

A. Stable Group

Before introducing the algorithm, we first define *stable connectivity* between mobile nodes and the term *stable group* [12].

Through periodic beaconing, a mobile node can estimate its distance (e.g. extrapolate from signal strength of received beacons) to each neighboring node[5]. Assuming symmetric transmissions, the nodes A and B are *neighbors* if the distance between them $\|AB\| \leq r$, where r is the transmission range. $\|AB\|$ varies since both A and B are mobile. However, if A and B are of the same mobility group, $\|AB\|$ is less variable and relatively steady. To formally define stable connectivity in terms of the distance $\|AB\|$, we define the following terms:

Definition 1: *Nodes A and B form an* ***Adjacent Group Pair*** *(AGP), denoted by $A \overset{0}{\sim} B$,* if $\|AB\|$ obeys a normal distribution with a mean $\mu \leq r$, and a standard deviation $\sigma < \sigma_{max}$.

Definition 2: *Nodes A and B form a* **k*-related Adjacent Group Pair*** *(k-AGP), denoted by $A \overset{k}{\sim} B$, $k \geq 1$, if there exist k intermediate nodes $C_1, C_2, \ldots, C_k$, such that $A \overset{0}{\sim} C_1, C_1 \overset{0}{\sim} C_2, \ldots, C_i \overset{0}{\sim} C_{i+1}, \ldots, C_k \overset{0}{\sim} B$.*

Definition 1 captures the fact that if two adjacent nodes are in the same group, the distance between them stabilizes around a mean value μ with small variations, while $\mu < r$ so that they are connected. Although they may be out of range from each other ($\|AB\| > r$) intermittently, the probability is low based on its probability density function. The standard deviation σ represents the degree of variations; nodes with correlated mobility pattern have small σ. Two nodes forming an AGP have stable connectivity, and Definition 2 extends the stable connectivity relation to non-neighboring nodes. Fig. 7 gives an example of AGP and k-related AGP.

We now formally define the relation between a group of nodes with stable inter-nodal distance.

[5]One may argue that it is non-trivial to estimate such distances accurately. In fact, the effectiveness of our algorithm does not rely on accurate estimates, since it derives properties from distance variations over time.

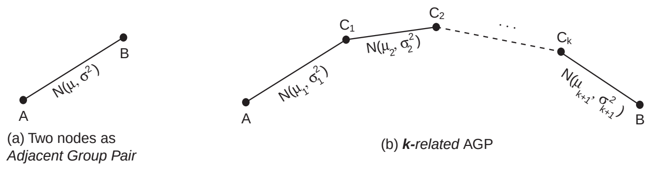

Fig. 7. Adjacent Group Pair and k-related AGP

Definition 3: *Nodes A and B form a* ***group pair****, denoted by $A \sim B$, if either $A \overset{0}{\sim} B$ or $A \overset{k}{\sim} B$, $k \geq 1$.*

Definition 4: *$A_1, A_2, \ldots, A_n$ are in one* ***stable group*** *$\mathbf{G}_s$, denoted by $A_i \in G_s$, if $A_i \sim A_j, \forall i, j, 1 \leq i, j \leq n$.*

The definition of a **stable group** identifies the nodes with stable connectivity in terms of relative stability in distance (or correlated mobility patterns) over time, rather than close geographic proximity at any given time instant. It excludes any mobile nodes that briefly connect but soon separate due to different movement patterns (belonging to different mobility group). The definition of a stable group coincides with that of a mobility group in the RVGM model, but it is based on the local distance between neighboring nodes.

B. Distributed Grouping Algorithm

We propose a fully distributed grouping algorithm for a node to identify its stable group — the group of nodes it has reliable connectivity with — at run-time, using only local observations.

On each mobile node A_i, the following local states are maintained for running the distributed grouping algorithm:

– *Profile of measurements* $[P(A_i)]$: a two-dimensional profile in which each row represents one of the neighbors, and each column represents distances to all neighbors obtained from one round of measurements. After l measurements, l samples of distances are obtained from A_i to each neighbor.
– *The set of AGP nodes* $[AGP(A_i)]$: the set of neighbors that has been identified to have AGP property, using $P(A_i)$.
– *The set of nodes in the same stable group* $[G_s(A_i)]$: updated regularly by exchanging information with AGP neighboring nodes. During boot-strapping, G_s in a node A_i is initialized to $\{A_i, AGP(A_i)\}$.
– *The set of servers in the stable group* $[N_s(A_i)]$: for each server S_i, its *id* and last-known velocity are stored as a tuple $\langle id(S_i), \mathbf{V}(S_i)\rangle$ in $N_s(A_i)$.

The distributed grouping algorithm allows the mobile nodes to find their AGP neighbors and construct their stable group G_s in a fully distributed fashion. It has the following steps:

1) **Measurements**: A_i measures the distance between itself and each neighbor for l times. The measurements are collected in the profile $P(A_i)$.
2) **Updates**: For each neighbor, the l distance measurements are used to obtain the mean distance μ, and the standard deviation σ around the mean. If μ and σ satisfy Definition 1, then the neighbor and A_i are added to each other's $AGP(\cdot)$ and $G_s(\cdot)$ set. Otherwise, if the neighbor is an existing AGP node, it is removed from $AGP(A_i)$ and $G_s(A_i)$, and likewise A_i is removed from the AGP and G_s set of the neighbor. If A_i is a server, its tuple $\langle id(A_i), \mathbf{V}(A_i)\rangle$ is stored in $N_s(A_i)$, and its $\mathbf{V}(A_i)$ is updated. If the service was terminated on A_i, its tuple is removed.
3) **Exchanges**: A_i exchanges its $G_s(A_i)$ and $N_s(A_i)$ with the neighbors that are in $AGP(A_i)$. It sends the following information to all nodes in $AGP(A_i)$: (1) its identifier; (2) its $G_s(A_i)$; and (3) its $N_s(A_i)$. Upon receiving information from other nodes, it constructs its local copy of G_s and N_s with the algorithm shown in Table III:

TABLE III
FORMATION OF $G_s(A_i)$ AND $N_s(A_i)$ ON NODE A_i

```
$G_s(A_i)$ initialized to $\{A_i, AGP(A_i)\}$, $N_s(A_i) = \phi$;
Let $G_s^p(A_j)$ and $N_s^p(A_j)$ be the previously received
$G_s(A_j)$ and $N_s(A_j)$, respectively, from $A_j$;

On receiving $G_s(A_j)$ and $N_s(A_j)$ from $A_j$:

foreach $A_k$ in $G_s(A_j)$
  if $A_k \notin G_s(A_i)$ then
      $G_s(A_i) = G_s(A_i) + A_k$
end
foreach $A_k$
 s.t. $A_k \in G_s^p(A_j), A_k \in G_s(A_i)$ and $A_k \notin G_s(A_j)$
    $G_s(A_i) = G_s(A_i) - A_k$
end
Identical processing applies to $N_s(A_i)$;
Remove all tuples $\langle id(S), \mathbf{V}(S)\rangle$ from $N_s(A_i)$
if $S \notin G_s(A_i)$.
```

Such exchange of G_s between the AGP nodes ensures that the local copy of G_s on all nodes of the same stable group eventually converge to an accurate group snapshot that includes all current members. This holds when new nodes are discovered and added to the stable group, or when existing members disconnect from the group and are subject to removal. The communication overhead and complexity of such exchanges are dependent on the size of the stable group, which is often much smaller than the total size of a partitionable ad hoc network. The local computation on each node only involves the calculation of mean and standard deviations, as well as set operations. They are, therefore, very computationally efficient algorithms.

C. Selection of Streaming Service

The mobile nodes, both clients and streaming servers, run the distributed grouping algorithm at a regular *service discovery interval*. After each run of the algorithm, the node constructs its stable group G_s and discovers a set of servers N_s.

The client selects its streaming server among the discovered servers. If there are several, the client selects the best server through velocity comparison. Since the nodes of the same mobility group can maintain longer lasting connections during network partitioning, the client selects the best as the server with the most similar velocity. The act of periodically discovering servers and selecting the best server allows the clients to actively pursue the more stable server, as a mean of adapting to the frequently changing network topology and network partitioning events.

D. Optimizing Service Efficiency

In addition to guaranteeing continuous availability of the streaming services, our second objective is to minimize the *service cost* — the number of streaming service instances deployed in the network, while maintaining the network-wide service coverage.

Because the stable connectivity among the nodes of the same mobility group, we use the mobility group as the basic unit of service coverage. At time t, $N_k(t)$ denotes the number of mobility groups that have access to a service instance, and $N_s(t)$ denotes the number of service instances deployed. We quantify the *service efficiency* as

$$S_{\textit{efficiency}}(t) = \frac{N_s(t)}{N_k(t)}$$

Since the nodes in a G_s group have stable connectivity, only one server is required to provide the service to the entire group, and other servers are redundant and can be turned off. The server also runs the distributed grouping algorithm at *service discovery intervals* to discover a set of stable servers N_s. By doing so, the servers in the same stable group monitor each other's presence. As an arbitration, the server with the highest *id* continues its service, and the others with lower *id*s automatically terminate their service instances.

The following condition must be checked before a server terminates its service. The server cannot be the parent or the child of the highest *id*ed server from a *recent* service replication. This prevents the parent and child servers, when they are in close vicinity shortly after the replication, from terminating each other, since each is intended for a different mobility group that will soon partition.

V. Simulation and Analysis

To evaluate the validity and the collective performance of *NonStop*, we perform extensive simulations using a large-scale mobile ad hoc network, with 130 mobile nodes moving in a square area of $750 \times 750\text{m}^2$. Nodes that move beyond an edge of the simulation area bounce back in their reflective directions[6] The transmission range of nodes is 60 meters. We configure 6 mobility groups, with their initial locations, movement paths and coverage areas shown in Fig. 8[7].

In addition, we simulate each mobility group's node velocities with a unique Gaussian distribution as in the Reference Velocity Group Mobility (RVGM) model. We also simulate the dynamic group memberships by introducing the notion of a *mobility epoch*. The mobility epoch is a time period during which the movement stays the same. Both mobile nodes and

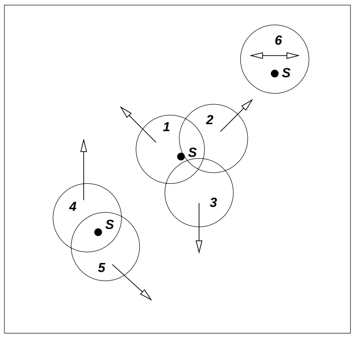

Fig. 8. Simulation of a mobile ad hoc network: initial scenario

the mobility groups have their mobility epochs, the lengths of which are exponentially distributed, with the mean $1/\lambda_n$ (90 time units in the simulations) and $1/\lambda_g$ (30 time units), respectively. At the end of one mobility epoch, the mobile node has a 0.3 probability of changing its group membership to another mobility group by following their neighbors, i.e., switching only to the mobility groups of its neighboring node. For the mobility group, at the end of its epoch, it randomly selects a new speed and direction.

Three streaming service instances are placed strategically at each concentration of mobile groups, so that initially all mobile nodes can access a service instance. However, the mobility groups are configured to partition and merge as soon as they start moving. We evaluate the performance of our algorithms under these dynamic network topology changes. All simulations are run for 1000 time units, t. The performance metrics we use to evaluate our algorithms are: (1) *service coverage*, in terms of the total number of nodes that can access a streaming service, normalized over the total number of nodes; and (2) *service cost*, in terms of the total number of streaming service instances in the network, normalized over the total number of mobility groups (which effectively reflects the service efficiency as defined in Sec. IV-D). Note that the normalized service cost may be less than 1 when one server is serving two or more temporarily merged mobility groups.

A. Comparison with Alternative Approaches

We compare the *NonStop* algorithms with three alternative approaches. (1) *No replications*: The baseline approach where the three servers move with their own mobility groups, and take no actions to ensure service coverage. Only the client nodes run the the distributed grouping algorithm to discover and select servers. (2) *Fixed Servers*: This is similar to the approach above, except four servers are fixed at specific points uniformly spaced in the $750 \times 750\text{m}^2$ simulation region, analogous to the cellular base stations. The servers do not replicate services. (3) *Probe for Replications*: Similar to the first approach where the three servers are mobile, and the clients run the distributed grouping algorithm to discover servers. However, if no servers are found in its stable group (*i.e.*, during a streaming interruption), the client regularly probes reachable neighbors for a service instance that it can replicate. Also, the redundant service instances in the same stable group are terminated. This is the approach proposed

[6]We have also implemented *wrap-around* border behavior so that nodes re-emerge at the opposite side, the simulation results do not change.

[7]The coverage areas of groups 1-3 and 4-5 are initially fully overlapped. The illustration is different for the sake of clarity.

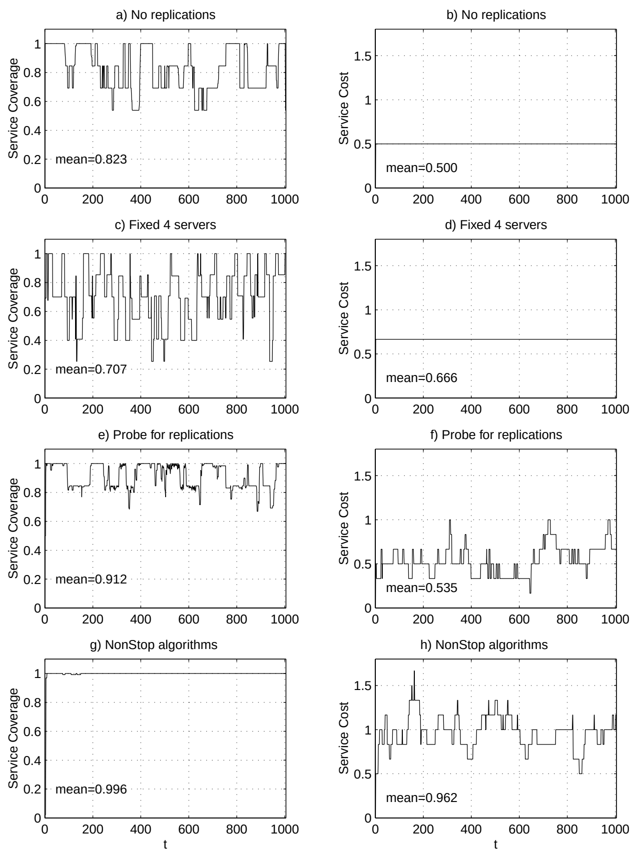

Fig. 9. Comparison of Different Approaches

in [12] for improving service accessibility during network partitioning, which this work is based upon.

The normalized service coverage and service cost of the three alternative approaches and our prediction-and-replication approach are plotted in Fig. 9. We also tabulate their mean values[8] in Table IV.

TABLE IV
SERVICE COVERAGE AND SERVICE COST COMPARISON

Approaches	Mean service coverage	Mean service cost
no replications	0.823	0.500 (const.)
fixed (4) servers	0.707	0.666 (const.)
probe for replications	0.912	0.535
prediction & replications	0.996	0.962

The first two approaches — no replication and fixed servers — have no service replications, and consequently have the lowest service coverage and constant service cost. The server's mobility affects the service coverage. When the servers are moving with the mobility groups (in the no-replication approach), the service coverage changes whenever the groups separate (network partitions) or recombine (network merges). When the servers are fixed (in the fixed-servers approach), the coverage decreases and increases as the mobile nodes move in and out of the transmission range of the fixed servers. Therefore, Fig. 9(c) shows more periodic and frequent rises and drops in service coverage than Fig. 9(a).

[8]The mean values are also marked within the figures in all illustrations.

The third approach, probe for replications, shown in Fig. 9(e)(f), greatly improves the service coverage to 91.2% because service instance can replicated onto disconnected nodes. However, the service coverage is still interrupted when the network partitions, and the service replication occurs by chance — only when a probing client encounters a server. In comparison, our NonStop approach using partition prediction by the server achieves *full service coverage*, shown in Fig. 9(g)(h), because the service is always replicated well before the partitioning, and the redundant service instances are terminated only after a period of careful monitoring. Naturally, our approach incurs a higher mean service cost of 0.962, which means on average 5.77 service instances are deployed. However, this is still less than the total number of mobility groups, indicating that our service efficiency algorithms effectively reduces the redundant servers. Comparing to 16 fixed servers, our approach achieve full service coverage with only $1/3$ of the service cost.

From the comparisons, we may conclude that the NonStop algorithm is effective in providing continuous service coverage, since the replications is decided at run-time based on the changing network topology.

B. BSCA, Kalman SC and Perfect Group Identification

We compare the performance of the basic sequential clustering algorithm (BSCA) with that of the Kalman filter estimation sequential clustering algorithm (Kalman SC), and further compare both algorithms with the performance of a perfect group identification.

Recall that the performance of both BSCA and Kalman SC are sensitive to the order in which the data points are presented, and to the shape of the clusters formed by the data points. In our simulation, the client velocities are collected by the server whenever the client sends a service request and piggybacks its velocity information. We model each client's service request rate as a Poisson process which is independent of other nodes. Therefore, the client velocities collected by the servers are random and are not ordered by their mobility groups. In addition, to vary the shape of the clusters in the velocity space, we adjust the variance $\mathbf{S}$ of the Gaussian distribution that generates the node velocities of each mobility group. We have two setups: in setup 1 (Fig. 10(a)), the mobility group velocities form well-separated and compact clusters, and in Setup 2 (Fig. 10(b)), they form scattered and elongated clusters. In Fig. 10(a), the mean group velocity $(v_x,\ v_y)$ of every mobility group is labeled corresponding to the physical layout shown in Fig. 8. Based on such setups, the initial parameters of both BSCA and Kalman SC are shown in Table V.

The simulation results of Setup 1 are shown in Fig. 11. In (a) the service coverage is initially 0 as all mobile nodes begin without any service, but increases as the nodes discover service nodes by running the distributed grouping and server selection algorithms. Because there are large separations between the node velocity clusters, it is found that both BSCA and Kalman SC correctly identify all necessary mobility groups, and initiate the server to replicate at the appropriate time.

TABLE V
BSCA AND KALMAN SC INITIAL PARAMETER VALUES

	α	$n_{\max}$	$\mathbf{P}_0$	$\mathbf{Q}_0$	$\mathbf{R}_0$
BSCA	2	4	—	—	—
Kalman SC	0.10	4	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}$	$\begin{pmatrix} 0.04 & 0 \\ 0 & 0.3 \end{pmatrix}$

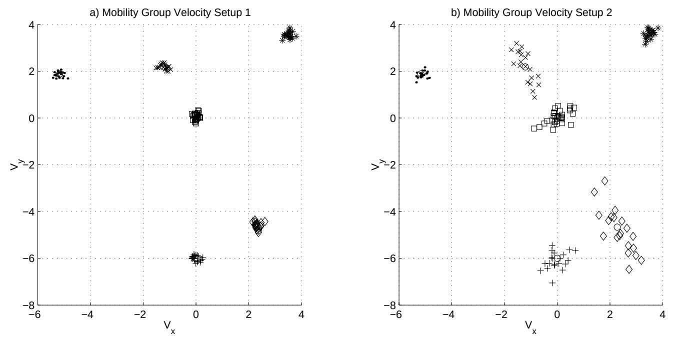

Fig. 10. Simulation Mobility Group Velocities Setup 1 and 2

Thus both sequential clustering based algorithms achieve full service coverage, identical to the perfect identification based algorithm, illustrated in Fig. 11(a).

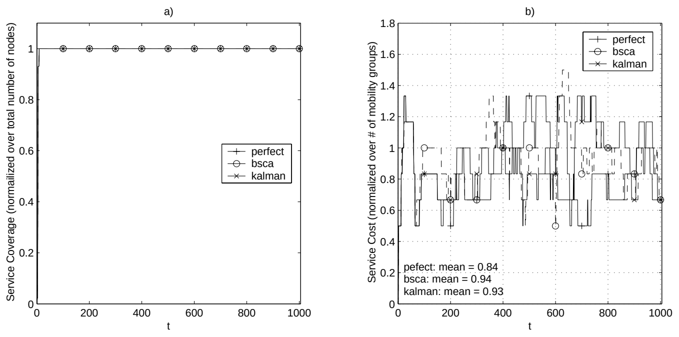

Fig. 11. Setup 1: Comparison of perfect mobility identification, BSCA, and Kalman SC

However, in Fig. 11(b), both the BSCA and Kalman SC algorithms incur higher service cost, as the sequential clustering algorithms tend to identify extra mobility groups and trigger more service replications. This is due to the algorithm's occasional misclassification, and also the clustering parameter $n_{\max}$ (the maximum allowed number of clusters) set to a larger value (4) to prevent a under-detection of mobility groups. Fortunately, the redundant service replications are quickly rectified by the service termination algorithm, shown as the high narrow spikes. Overall, both sequential clustering algorithms are able to achieve normalized service costs[9] of 0.93 and 0.94, which are less than 1.

Fig. 12 shows the simulation results of all three algorithms for Setup 2. In general, the complete service coverage is not constantly maintained, since the large variance in velocity distribution generates nodes with more sporadic velocity, that often stray from their mobility group and are disconnected from service, which explains the many narrow dips in the service coverage. The Kalman SC algorithm, when initialized

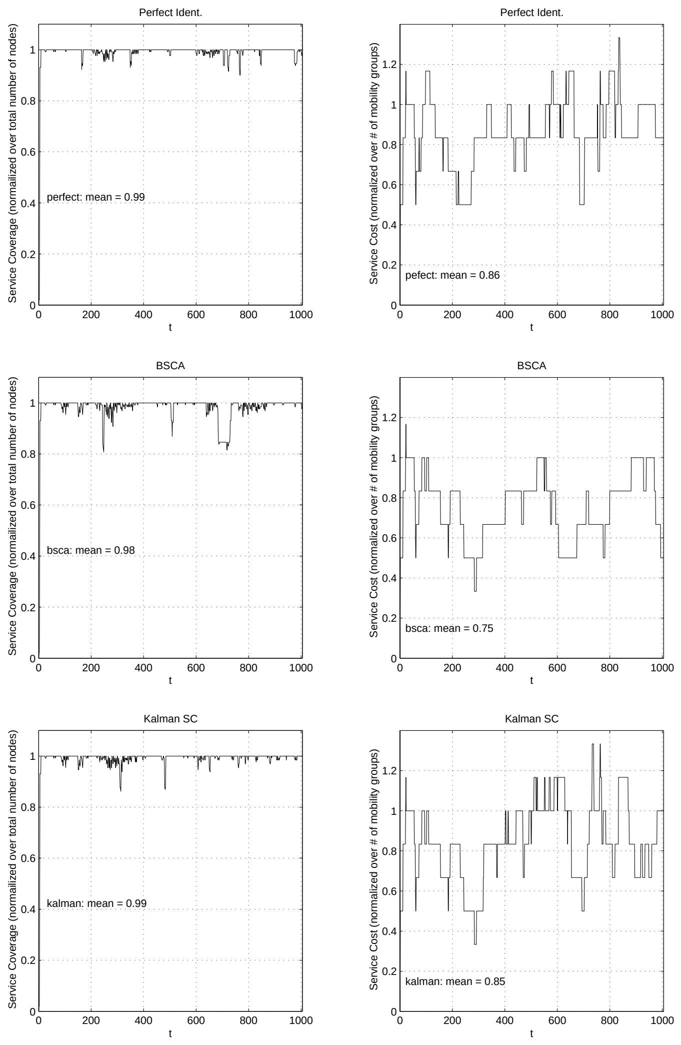

Fig. 12. Setup 2: Comparison of perfect mobility identification, BSCA, and Kalman SC

with suitable parameters ($\mathbf{P}_0$, $\mathbf{Q}_0$, and $\mathbf{R}_0$) as shown in Table V, can correctly identify the clusters of various shape and orientations, and attain the same level of service coverage (0.99) and at even slightly lower service cost (0.85) than the perfect mobility identification algorithm. In comparison, the BSCA algorithm does not recognize cluster orientation and hence is unable to distinguish neighboring velocity clusters oriented at different directions (such as those formed by group 2 and 5, and group 4 and 6 in Setup 2 shown in Fig. 10(b)) as separate clusters. Hence, BSCA under-detects the number of mobility group, and leads to failure of the server to predict partitioning and replicate service. This is reflected in its service coverage dropping almost 20% for a period of time and a lower service cost 0.75 compared to Kalman SC and the perfect

[9]For all illustrations, the mean values are marked within the figures.

group identification. Compared with BSCA, the advantages of Kalman SC are brought forth by the added computational complexity of applying the Kalman Filter, which may add to the computation load on the streaming servers. That said, in our simulations with Pentium-grade processors, we do not detect any noticeable extra computation time with Kalman SC compared with BSCA.

For later simulation results in order to evaluate the performance of other aspects of our algorithms, all simulations are run with the mobility group velocity Setup 1, to eliminate the presence of stray mobile nodes with sporadic velocity that causes drops in service coverage. In addition, since both BSCA and Kalman SC perform equally well for the compact clusters in Setup 1, for simplicity, we use the BSCA algorithm for mobility group identification in the simulations, and refer to BSCA simply as the sequential clustering (SC) algorithm.

C. Group vs. Random Walk Mobility Model

Our algorithms utilize the assumption of correlated mobility patterns of mobile nodes to predict partitioning, the results of which are used to replicate service and to achieve service availability. To test the robustness of our algorithms, we examine their performance when the assumption of group mobility no longer holds.

In this simulation, the mobile nodes move according to either RVGM or the *random walk* mobility model. For random walk movement, the mobile node's speed is selected uniformly between 0 and a maximum speed of $10\ m/t$, and a direction chosen uniformly between 0 and 2π. At the end of a mobility epoch, the nodes randomly select a new speed and direction. The random walk nodes are evenly distributed throughout the simulated network area. We define the *degree of group mobility* in the network as the percentage of network nodes that follow the RVGM model. We simulate four cases: 100%, 75%, 25% group mobility and 100% random walk.

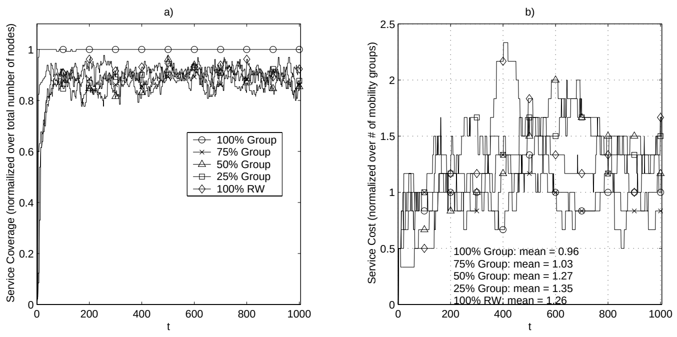

Fig. 13. Comparison between group and random walk node mobility

As expected, in Fig. 13, the service coverage is lower when there are random walk nodes in the network, compared to 100% in 100% group mobility, it drops to between 80% and 90%. However, the service coverage of 80% to 90% is unexpectedly high, and interestingly, it does not vary much for the different degree of random walk in the network. This is due to two balancing factors. First, higher percentage of random walk nodes gives rise to fewer occurrences of network partitioning, as the nodes are uniformly distributed throughout the network. Second, higher percentage of RVGM nodes allows the SC to better capture the movement pattern, predict partitioning and replicate service. Both factors afford continuous service coverage to the mobile nodes.

The accuracy of the SC mobility group identification decreases when the node velocities are random and uncorrelated, the SC identifies the maximum number of mobility groups allowed, triggering more replications, and hence, higher service cost. However, at 100% randomness, there are no clear distinctions between the node velocities, the SC algorithm tend to classify nodes into a single mobility group, this explains the drop in service cost.

VI. Related Work

In addition to the alternative approaches which we compared and discussed in Section V-A, other recent research works have also addressed the problem of service availability in frequently partitioned ad hoc networks, although with slightly different focuses. Here, we present a comparison between our work and the recent contributions.

Karumanchi *et al.* [13] has assumed that there are many designated servers throughout the network. However, the servers are pre-determined and fixed, so during network topology changes and network partitioning, their reachability changes. Hence, the work has developed run-time heuristics for clients to select servers with the highest likelihood of being accessible, in order to maximize the chances of successful service requests. In comparison, rather than relying only on client side heuristics, our approach aggressively ensures service accessibility to the clients by dynamically creating and placing servers based on the changing network topology. The service accessibility is further improved by client side selections of reliable servers.

The work by Hara [14] focuses on data accessibility in ad hoc networks. It assumes that all mobile nodes can store some data replicas. Hence the work is concerned with the optimal placement of data replicas around the network that achieves high data accessibility in the event of network partitioning, by considering data access frequencies of mobile nodes. Our approach is similar in replicating data or service instances; however, we consider topology changes and connection stability to replicate only when necessary and to strategically place the replicas. Further, redundant service replicas are eliminated through service efficiency algorithms. Thus, our approach achieves network wide data or service accessibility with much lower replica costs.

Liang and Haas in [15] have proposed the virtual service backbone. Similar to our approach, the servers are dynamically created and terminated as the network topology changes to ensure network wide service availability. Further, it is service efficient by having only one server serving a well-connected group of nodes, in this case a r-hop network zone, and redundant servers are merged. However, in their approach, when servers fail due to network partitioning, a new server is regenerated. This has two drawbacks. First it relies on the nature of the service being regenerable, which is unlikely for general network services. Without the service being regenerable, the service is lost in the partitioned network. Second, during the period of server failure detection and regeneration, the service is interrupted for the mobile nodes. Our approach averts these drawbacks by creating a new server through replication before the partitioning occurs.

The major difference between these schemes and our work is that they cannot guarantee service availability when the network is partitioned. This is because they treat the event of network partitioning as non-deterministic: [13] and [14] populate the network with redundant data replicas or servers to mitigate the impact of partitioning, and [15] regenerates servers after the failure. Our solution utilizes observed node mobility patterns to *predict the occurrence of partitioning*, and takes the necessary actions *in advance* to efficiently provide continuous service availability when the network partitions.

VII. Conclusions

In this paper, we have proposed *NonStop*, a collection of middleware-based on-line algorithms to address the problem of provisioning continuous streaming service of multimedia data in wireless ad hoc networks. We take the approach of exploiting correlated mobility patterns exhibited by mobile users. On the streaming servers, we present two variants of sequential clustering algorithms that can identify correlated mobility patterns, which is used to predict the time and location of network partitioning. On the clients, we show a fully distributed grouping algorithm that discovers mobility group membership based on the stability with respect to distances to neighboring nodes. Our simulations show that, under frequent network partitioning, our algorithms achieve continuous and network-wide streaming coverage with efficiency, whereas other alternative approaches only mitigate the effect of partitioning but cannot prevent streaming interruptions. However, *NonStop* does bring the overhead of replicating services, which we believe is inevitable when we demand continuous streaming services in a fundamentally disruptive network. In addition, we note that *NonStop* does not apply to the case where a high degree of user mobility and a large media stream co-exist, in which case the replication may not be accomplished between the time of prediction and partitioning.

References

[1] A. McDonald and T. Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1466–1486, August 1999.

[2] S. Jiang, D. He, and J. Rao, "A Prediction-based Link Availability Estimation for Mobile Ad Hoc Networks," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, April 2001.

[3] W. Su, S.-J. Lee, and M. Gerla, "Mobility Prediction and Routing in Ad Hoc Wireless Networks," *International Journal of Network Management*, 2000.

[4] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," in *The Seventh International Conference on Mobile Computing and Networking (MOBICOM 2001)*, September 2001, pp. 61–69.

[5] D. Tang and M. Baker, "Analysis of a Local-Area Wireless Network," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, August 2000.

[6] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," in *Proceedings of ACM/IEEE MSWiM*, Seattle, WA, August 1999.

[7] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, August 1999, pp. 195–206.

[8] K. H. Wang and B. Li, "Group Mobility and Partition Prediction in Wireless Ad-Hoc Ntworks," in *Proceedings of IEEE International Conference on Communications (ICC)*, NYC, NY, April 2002.

[9] K. H. Wang, "Adaptive Service Provisionings in Partitionable Wireless Mobile Ad-Hoc Networks," M.S. thesis, University of Toronto, Department of Electrical and Computer Engineering, October 2001.

[10] R. F. Stengel, *Optimal Control and Estimation*, chapter 4, pp. 342–351, Dover Publications, Inc., 1986.

[11] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, vol. 1, chapter 1, pp. 1–16, Academic Press, 1979.

[12] B. Li, "QoS-aware Adaptive Services in Mobile Ad-hoc Networks," in *Proceedings of the Nineth IEEE International Workshop on Quality of Service (IWQoS 01)*, Karlsruhe, Germany, June 2001, pp. 251–268.

[13] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information Dissemination in Partitionable Mobile Ad Hoc Networks," in *Proceedings of IEEE Symposium on Reliable Distributed Systems*, Lausanne, Switzerland, October 2000.

[14] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, April 2001.

[15] B. Liang and Z. J. Haas, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.