Kalman Filter Based Dead Reckoning Algorithm for Minimizing Network Traffic Between Mobile Nodes in Wireless GRID

Seong-Whan Kim and Ki-Hong Ko

Department of Computer Science, Univ. of Seoul, Jeon-Nong-Dong, Seoul, Korea
Tel.: +82-2-2210-5316; fax: +82-2-2210-5275
swkim7@uos.ac.kr, jedigo@venus.uos.ac.kr

Abstract. Conventional GRID service is static (no mobility), and it has many drawbacks such as continuous connection, waste of bandwidth, and service overloading. Wireless GRID supports mobility, however it should consider geographic position to support efficient resource sharing and routing. When the devices in the GRID are highly mobile, there will be much traffic to exchange the geographic position information of each mobile node, and this makes adverse effect on efficient battery usage. To minimize the network traffic between mobile users, we use dead reckoning algorithm for each mobile nodes, where each node uses the algorithm to estimates its own movement (also other node's movement), and when the estimation error is over threshold, the node sends the UPDATE (including position, velocity, etc) packet to other devices. As the estimation accuracy is increased, each node can minimize the number of UPDATE packet transmission. To improve the prediction accuracy of dead reckoning algorithm, we propose Kalman filter based dead reckoning approach. To experiment our scheme, we implement a popular network game (BZFlag) with our scheme added on each mobile node, and the results show that we can achieve better prediction accuracy and reduction of network traffic by 12 percents.

Keywords: Dead reckoning, Kalman filter, Wireless GRID.

1 Introduction

Conventional GRID service supports no mobility, and results in many drawbacks such as continuous connection, waste of bandwidth, and service overloading. Wireless GRID supports mobility and it should consider geographic position to support efficient resource sharing and routing [1]. However, if the device in the GRID is highly mobile, there will be much traffic to manage the geographic position of each mobile node, and this make adverse effect on efficient battery usage. To minimize the network traffic between networking mobile devices, dead reckoning technique is used [2]. Each mobile device uses the algorithm to estimates its movement and other devices' movement, thereby, each device can minimize the transmission of its information (position, velocity, etc) to other entities. R. Gossweiler and R. J. Laferriere introduced the dead reckoning algorithm for the multi-user game [2], and S. Aggarwal and H.

E. Sha et al. (Eds.): EUC 2006, LNCS 4096, pp. 162 – 170, 2006.

Banavar proposed the use of globally synchronized clocks among the participating players and a time-stamp augmented dead reckoning vector that enables the receiver to render the entity accurately [3]. In addition, W. Cai and F. B.S. Lee proposed a multi-level threshold scheme that adaptively adjusted, based on the relative distance between entities to reduce the rate of transmitting UPDATE packets [4].

To improve the prediction accuracy of dead reckoning algorithm, we propose the Kalman filter based dead reckoning approach. To simulate the mobility of mobile device scenarios in wireless GRID, we use a simple analogy, network game (BZFlag). In section 2, we review the dead reckoning and Kalman filter. In Section 3, we propose a Kalman filter based dead reckoning algorithm. In Section 4, we apply our Kalman approach on BZFLAG game; show the experimental results with minimized UPDATE packets between game players. We conclude in section 5.

2 Related Works

The networking server technique can be implemented using three methods: (1) peer to peer, (2) client server architecture, and (3) distributed server architecture. In peer to peer, each entity transmits the occurred information to each other. It is suitable for the small-scale network. In client server (CS) architecture, the server collects all of the data from the clients, stores the changes in some data, and then sends the results to each participating client. For large-scale networks, we need distributed server architecture. To distribute server, we can use load distribution method or map server method. Networking techniques should minimize (1) network bandwidth and (2) network delay.

2.1 Location Awareness in Wireless Mobile Networks

In the wireless mobile GRID, the GRID protocol's core concept partitions the geographic area into several squares in GRIDs. Each GRID is elected the GRID's leader (so called gateway), and GRID leaders perform routing GRID by GRID. This protocol is location awared because it exploits location information in routing. Geographic location awareness can be GeoCast, GeoTora, and GeoGrid methods [10].

- GeoCast: GeoCast sends a message to all mobile devices within a designated geographic area (so called a geo-cast region). This protocol differs from traditional multicast, because the it uses two zones: forwarding zone and multicast region. In the forwarding zone, the data packet is sent by unicast to each other's device, and in the multicast region, the data packet is sent by multicast to each other's device.
- GeoTora: GeoTora derives from TORA (temporally ordered routing algorithm). TORA maintains a DAG (directed acyclic graph) with the destination device as sink; the data packet is forwarded by the DAG's direction to sink. GeoTora divides into TORA (DAG region) and GeoCast region. In GeoCast regions, mobile devices perform the flooding, and in the DAG region, mobile devices perform an anycast from the source to any host.
- GeoGrid: GeoGrid is derived by the GRID protocol that have the GRID leader. GeoGrid uses two methods such as the flooding-based geo-casting and ticket-based geo-casting. The flooding-based geo-casting allows any grid leader in the forwarding zone to rebroadcast the messages, and the ticket-based geo-casting allows only ticket-holding grid leaders to rebroadcast.

2.2 Reviews on Dead Reckoning Algorithms

Since each mobile device is physically distributed, updating states (e.g. each mobile device's position, etc) of the mobile devices may generate a large amount of communication and thus saturate network bandwidth. To reduce the number of state UPDATE packets, the DR technique is used [4]. In addition to the high fidelity model that maintains the accurate position about its entities, each mobile device also has a dead reckoning model that estimates the position of all entities (both local and remote). Therefore, instead of transmitting state UPDATE packets, the estimated position of a remote mobile device can be readily available through a simple and localized computation [4]. The mobile device compares real position with DR position. If the difference between real position and DR position is greater than a threshold, the mobile device informs others remote entities to update their ghost object position [2]. We can describe the simple dead reckoning algorithm as follows.

```
Algorithm : Dead Reckoning

for every received packet of remote entity do
  switch received packet type {
    case UPDATE
      fix ghost position of remote entity
      break;
    case PLAYER_QUITING
      remove remote entity
      break;
  }

[Extrapolation] Extrapolate all ghost position based
on the past state information;

if (local entity's true position - local entity's
extrapolated position) > Threshold {
  Broadcast an UPDATE packet to the group
}
Draw all ghost
```

3 Kalman Filter Based Dead Reckoning Algorithm

In wireless GRID environment, each mobile device is geographically distributed. A technique referred to as dead reckoning (DR) is commonly used to exchange information about movement among the mobile devices [6, 7, 8]. Each mobile device sends information about its movement as well as the movement of the entities it controls to the other mobile devices using a dead reckoning vector (DRV). A DRV typically contains information about the current position of the entity in terms of x, y and z coordinates (at the time the DRV sent) as well as the trajectory of the entity in terms of the velocity component in each of the dimensions [3].

In this paper, we use the mobility of network game users to simulate the real geographically distributed mobile device environments. For the network game, we present a Kalman filter based dead reckoning to optimize the network traffic. A Kalman filter is a recursive procedure to estimate the states s_k of a discrete-time controlled process governed by the linear stochastic difference equation, from a set of measured observations t_k. The mathematical model is shown in in Equation (1) and Equation (2).

$$\boldsymbol{s}_k = \boldsymbol{A}\boldsymbol{s}_{k-1} + \boldsymbol{w}_{k-1} \tag{1}$$

$$\boldsymbol{t}_k = \boldsymbol{H}\boldsymbol{s}_k + \boldsymbol{r}_k \tag{2}$$

The NxN matrix A represents an state transition matrix, w_k is an Nx1 process noise vector with N(0, σ_w^2), t_k is Mx1 measurement vector, H is MxM measurement matrix, and r_k is Mx1 measurement noise vector with N(0, σ_r^2). To estimate the process, Kalman filter uses a form of feedback control as shown in Figure 1 [5]. We define $\hat{\boldsymbol{s}}_k^-$, $\hat{\boldsymbol{s}}_k$, p_k^- and p_k as the priori state estimate, posteriori state estimate, priori estimate error covariance, and posteriori estimate error covariance, respectively. K is the Kalman gain.

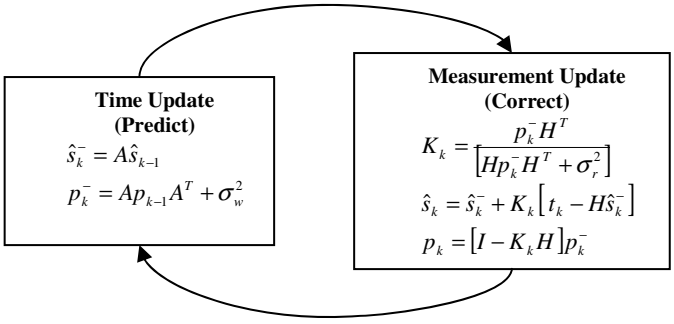

Fig. 1. Kalman filter cycle [5].

To evaluate our scheme, we used simple dead reckoning scenarios (scheme 1) and optimized dead reckoning algorithm for game logic (scheme 3) for comparison. For scheme 1 and scheme 3, we use Kalman filter approach (scheme 2 and scheme 4) to improve the prediction performance of scheme 1 and scheme 3 as shown in Figure 2.

Scheme 1 and scheme 2 use DRV, which includes only position and velocity information of each mobile device. In scheme 3 and scheme 4, we added the angle which is a direction of mobile device for prediction improvements, and the DRV is (x, y, z, vx, vy, vz, angle, t). Scheme 3 is real dead reckoning algorithm, which is optimized for BZFlag game logic. The details of each schemes are as follows.

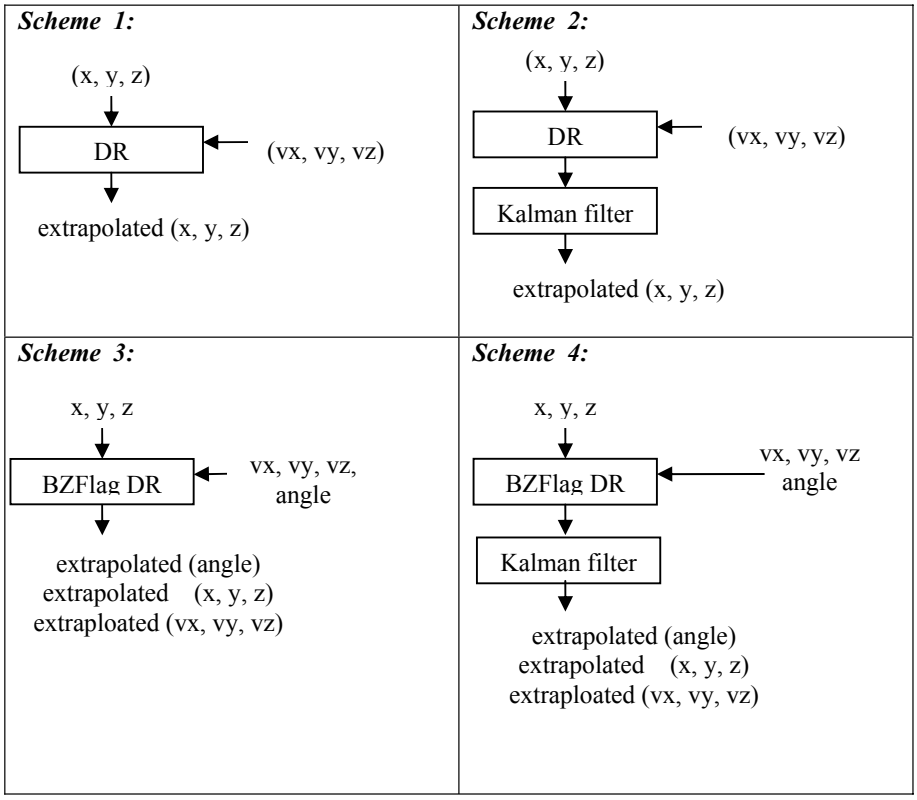

Fig. 2. Kalman filter approach for dead reckoning algorithm.

Scheme 1*:* We compute the extrapolated position using last position, last velocity, and time step as follows. We performed the extrapolation until the difference between the extrapolated position and the true position is under threshold.

```
Extrapolated position = last position + last velocity *
time step;
```

Scheme 2*:* Scheme 2 uses Kalman filter after computing the extrapolated position as scheme 1. We performed the extrapolation until the difference between the extrapolated position and the true position is under threshold.

```
Extrapolated position = Kalman Filter (last position +
last velocity * time step);
```

Scheme 3*:* To get a new extrapolated position, the scheme uses two equations depending on the game entity's motion type as follows. We performed the extrapolation until the difference between the extrapolated position and the true position is under threshold.

```
if (linear motion) {
  extrapolated position = last position + last velocity
* time step;
} else {
  extrapolated position = BZFlag function(angle);
}
```

Scheme 4*:* Scheme 4 adds Kalman filter after computing the extrapolated (position, velocity, and angle) as scheme 3. Our dead reckoning algorithm (scheme 4) is described as follows.

```
float speed = (vx * cosf(angle)) + (vy * sin(angle));
// speed relative to the tank's direction
radius = (speed / angular_velocity);

float inputTurnCenter[2]; // tank turn center
float inputTurnVector[2]; // tank turn vector
inputTurnVector[0] = +sin(last_angle) * radius;
inputTurnVector[1] = -cos(last_angle) * radius;
inputTurnCenter[0] = last_position-inputTurnVector[0];
inputTurnCenter[1] = last_position-inputTurnVector[1];

// compute new extrapolated angle using Kalman filter
float angle = Kalman (time step * angular_velocity);
float cos_val = cosf(angle);
float sin_val = sinf(angle);

// compute new extrapolated position
const float* tc = inputTurnCenter;
const float* tv = inputTurnVector;
new_x = tc[0]+((tv[0] * cos_val) - (tv[1] * sin_val));
new_y = tc[1]+((tv[1] * cos_val) + (tv[0] * sin_val));
new_z = last_position + (vz * time step);

// compute new extrapolated velocity
float vx = Kalman ((vx * cos_val) - (vy * sin_val));
float vy = Kalman ((vy * cos_val) + (vx * sin_val));
float vz = Kalman (vz);
```

4 Experimental Results

In this paper, we use a simple analogy: a popular on-line game BZFlag to simulate geographically distributed mobile devices. BZFlag (Battle Zone Flag) is a first-person shooter game where the players in teams drive tanks and move within a battlefield. The aim of the players is to navigate and capture flags belonging to the other team and bring them back to their own area. The players shoot each other's tanks using "shooting bullets" The movements of the tanks (players) as well as that of the shots (entities) exchanged among the players using DR vectors [3, 9].

The experimental data are the position value and the velocity value gotten in BZFlag game. We used the experimental data of 8301 numbers, and the threshold to 0.09. We compared the number of DRV packet transmission and the average

prediction error E as shown in (3). (x, y, z) represent the true position, (newx, newy, newz) represent the extrapolated position, and (n) represent the number of data.

$$E = \frac{\sum_{i=1}^{n=8301} \sqrt{(x_i - newx_i)^2 + (y_i - newy_i)^2 + (z_i - newz_i)^2}}{n} \tag{3}$$

Table 1 shows the experimental result. Table 1 shows that the number of DRV transmission of scheme 2 and scheme 4 is smaller than that of scheme 1 and scheme 3, respectively.

Table 1. Font sizes of headings. Table captions should always be positioned *above* the tables.

	Scheme 1	Scheme 2	Scheme 3	Scheme 4
# of DRV transmission	4657	3965	703	611
E	4.511	2.563	0.4745	0.4048

In BZFlag game, it uses the game optimized dead reckoning algorithm, which means that it considers the two more vectors (orientation and angle) to predict the position more accurately. Scheme 3 improves the simple dead reckoning approaches: scheme 1 and scheme 2. In scheme 4, we used Kalman filter prediction on velocity and angle, and Figure 3 compares the scheme 3 and scheme 4 over 8000 time steps.

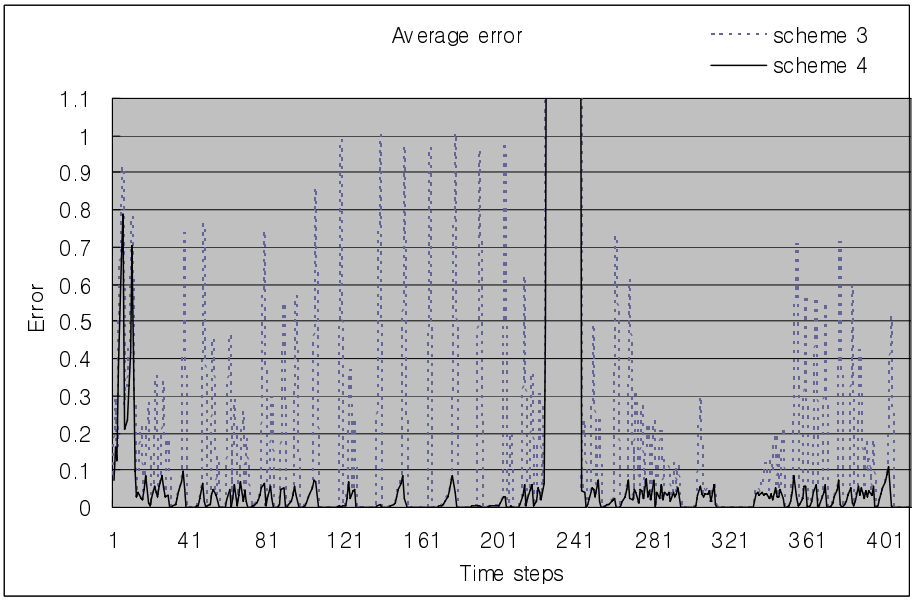

Fig. 3. Comparison of prediction accuracy for 8301 time step duration

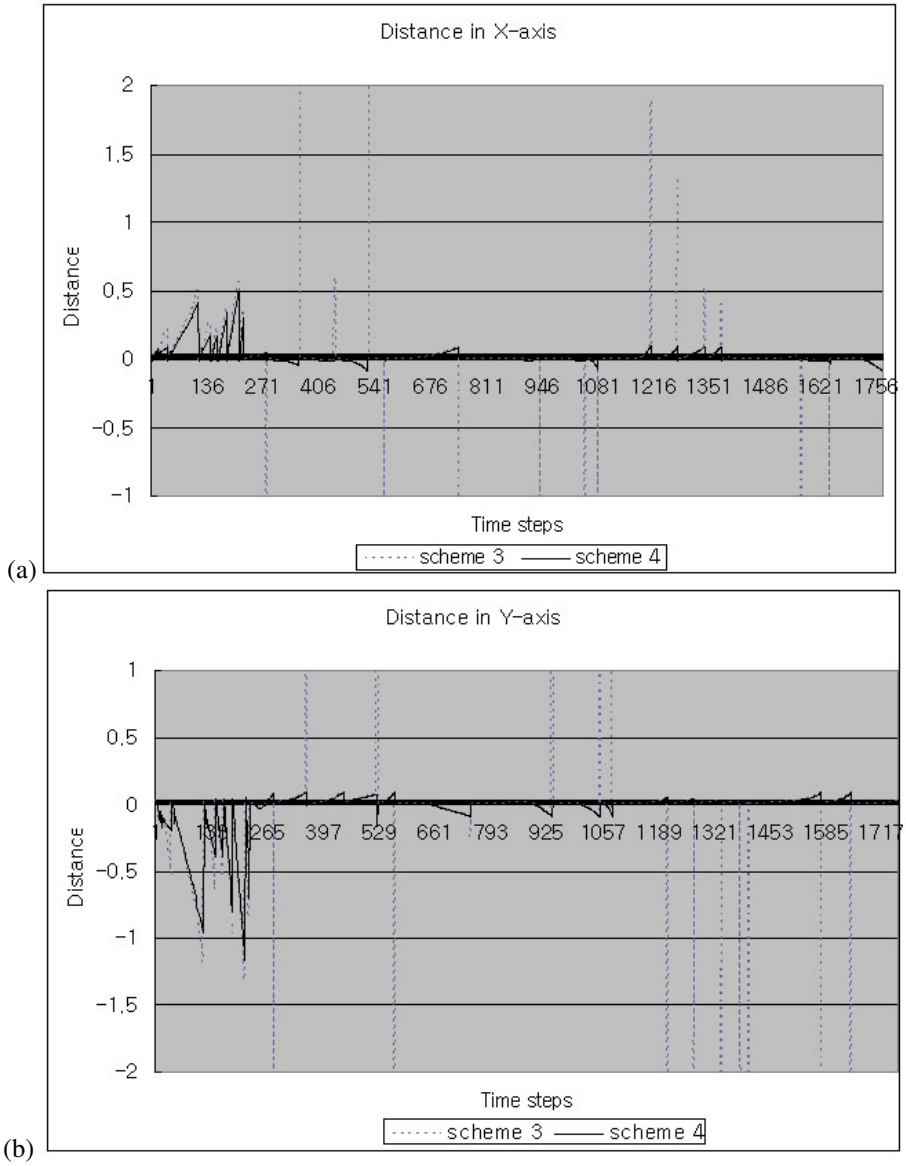

Fig. 4. Error in X and Y prediction: (a) errors in X direction, (b) errors in Y direction

For better comparison, we computed moving average for each 20 samples. *The dotted line* and *the solid line* show the result of scheme 3 and scheme 4, respectively.

Figure 4 shows the prediction errors in X and Y direction, respectively. Scheme 3, which uses BZFlag game optimized logic, shows fluctuations, and when the prediction error is over than 0.9, the BZFlag clients should send dead reckoning packets. Minimizing dead reckoning packets also minimized network latency and the

game responses time. Even in the detailed view, the prediction errors of scheme 4 are smaller than the prediction errors of scheme 3.

5 Conclusions

In this paper, we propose the Kalman filter approach to improve the dead reckoning algorithm for geographically oriented networking between mobile nodes in wireless GRID environments. Our scheme improves the accuracy of dead reckoning prediction, and minimizes the network traffic among the mobile devices. Instead of experimenting geographically distributed mobile devices, we use a popular on-line game BZFlag, and compare our scheme with the state-of-the-art dead reckoning algorithm optimized for game logic. Our Kalman filter based dead reckoning scheme reduces more than 10% of network traffic over game optimized dead reckoning algorithms. Reduced network traffic can make efficient battery usage.

References

1. Zhang W., Zhang J., Ma D., Wang B., Chen Y.: Key technique research on GRID mobile servie. Proc. 2nd Int. Conf. Information Technology (2004)
2. Gossweiler, R., Laferriere, R.J., Keller, M.L., Pausch, R.: An introductory tutorial for developing multi-user virtual environments. Tele-operators and Virtual Environments, vol. 3. no. 4 (1994) 255-264
3. Aggarwal, S., Banavar, H., Khandelwal, A., Mukherjee, S., Rangarajan, S.: User experience: accuracy in dead-reckoning based distributed multi-player games. Proc. ACM SIGCOMM 2004 Workshops on Net-Games. Network and System Support for Games (2004)
4. Cai, W., Lee, F.B.S., Chen, L.: An auto-adaptive dead reckoning algorithm for distributed interactive simulation. Proc. of the thirteenth Workshop on Parallel and Distributed Simulation (1999)
5. Welch, G., Bishop, G.: An introduction to the Kalman filters. available in http://www.cs.unc.edu/~welch/Kalman/index.html
6. Gautier, L., Diot, C.: Design and Evaluation of MiMaze, a Multiplayer Game on the Internet. Proc. IEEE Multimedia. ICMCS (1998)
7. Mauve, M.: Consistency in Replicated Continuous Interactive Media. Proc. of the ACM Conference on Computer Supported Cooperative Work (2000) 181–190
8. Singhal, S.K., Cheriton, D.R.: Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality. Tele-operators and Virtual Environments. vol. 4. no. 2 (1995) 169-193
9. Schoeneman, C., Riker, T.: BZFlag (Battle Zone capture Flag), available in http:// www.bzflag.org
10. Tseng, Y.-C., Wu, S.-L., Liao, W.-H., Chao, C.-M.: Location awareness in ad hoc wireless mobile networks. IEEE Computer. vol. 34, no. 6, (2001) 46-52