Issues in Computational Resource Allocation in Cooperative Control

John P. Murphy, *Student Member, IEEE*, Luke M. Wachter, and Laura E. Ray, *Member, IEEE*

Abstract—This paper considers the problem of computational resource allocation for a team of dynamic mobile robots subject to limited communication and processing bandwidth, and uncertain state information. We describe a robot test bed, the Dynabot, that exhibits these constraints and characterize its information flow and state estimation accuracy. These models are used for simulating formation of a group of Dynabots, while illustrating resource allocation tradeoffs.

***Index Terms*—Cooperative systems, Mobile robot dynamics, Mobile robot navigation.**

I. Introduction

Distributed control requires each robot in a team to access information about the state of other robots. Information is subject to delays and uncertainty due to the limited processing resources, communication resources, and imperfect state knowledge. These limitations lower system stability and performance by increasing likelihood of collisions and time-to-goal. Optimization of communication, state estimation, and control are generally considered separately in the literature. Yoshida et al. [1] address communication topology design to minimize transmission time between mobile robots. Yook et al. [2] investigate the use of state observers at each node to estimate the state at other nodes, thereby trading computation and communication bandwidth. Sweeney et al. [3] examine the limits on scalability of a multi-robot system due to limited processing on individual robots. They show that when the system operates at or near its maximum processing bandwidth, tradeoffs in the design of the system architecture become important. Therefore, issues of limited communication bandwidth and uncertain state information must be considered together.

We identify three fundamental tradeoffs in computational resource allocation for individual robots in light of limited processing, communication, and state information:

1. State estimation reduces the uncertainty of shared information at the expense of increased processing and lower update frequency.
2. Increased control frequency reduces the delay between when information is received and when it is used at the expense of computational bandwidth and thus reduces time for other tasks such as state estimation.
3. Enhanced content of exchanged state information compensates for aged information from other robots at the expense of increased communication error and delay.

We investigate the first of these tradeoffs as a means of informing the latter two. The tradeoffs guide the development of a scheduling architecture for a low-cost, high-speed dynamic robot (Dynabot) test bed. The robots are used to characterize communication latency empirically and to determine state estimation error for three estimator schemes. Using models developed from physical testing, we simulate formation control of a team of Dynabots using a potential function controller to illustrate the effects of computational resource allocation in state estimation on stability and performance.

II. Dynabot Robot

The Dynabot is low cost mobile robot for studying high speed distributed control. The robot, described in detail in [4,5] and shown in Fig. 1, is a four-wheel drive, suspensionless vehicle. Sensors include motor currents and wheel speeds, 5 Hz GPS, and a nano-inertial measurement unit (nIMU) with 3-axes of acceleration, angular rates and magnetic bearing. An 802.11.b wireless communication card provides communication between robots and with human operators. The robot can achieve 10 m/s speed and 0.5g acceleration on hard surfaces.

The custom software running on a Dynabot is divided into four parallel modules, with information asynchronously placed on a bus for use by other modules. The task progression for an individual robot is depicted in Fig. 2 along with a breakdown of the task time and update period parameter definitions. The *Incoming Com Module* processes incoming communication -small UPD packets broadcast over an ad hoc 802.11.b wireless network. The *Control Module* uses the incoming messages as they become available. This module updates the motor commands based on the current information (the robot's knowledge of its own state, the state of all other robots and the goal) according to an artificial potential control law described in Section IV. The Control Module is invoked

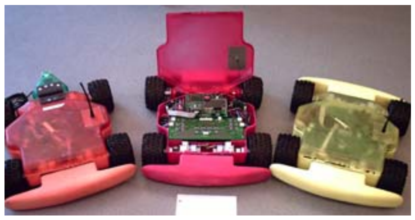
Fig. 1 Three Dynabots from a fleet of seven

J.P. Murphy is a Ph.D. candidate at the Thayer School of Engineering, Dartmouth College (e-mail: john.p.murphy@dartmouth.edu).

L. M. Wachter is an M.S. candidate at the Thayer School of Engineering, Dartmouth College (e-mail: luke.wachter@dartmouth.edu).

L.E. Ray is an Associate Professor at the Thayer School of Engineering, Dartmouth College Hanover, NH 03755 USA (e-mail: lray@dartmouth.edu).
This research is supported by NIST, Grant No. 60NANB4D1144 and 60NANB6D6130 awarded through the Institute for Security Technology Studies, and by the Army Research Office, contract No W911NF-06-1-0153.

ROBOCOMM 2007, 14th – 16th Oct 2007, Athens, Greece.
Copyright © 2011 – 2012 ICST
ISBN 978-963-9799-08-0
DOI 10.4108/ICST.ROBOCOMM2007.2146

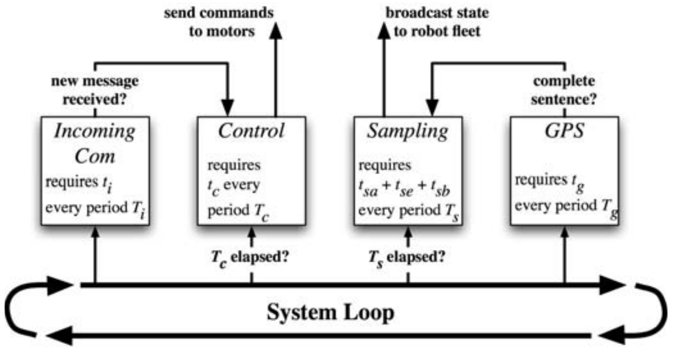
send commands to motors
broadcast state to robot fleet
new message received?
complete sentence?
Incoming Com
requires t_i every period T_i
Control
requires t_c every period T_c
Sampling
requires $t_{sa} + t_{se} + t_{sb}$ every period T_s
GPS
requires t_g every period T_g
T_c elapsed?
T_s elapsed?
System Loop

Fig. 2 Dynabot software architecture overview broken down into the fundamental modules with definition of timing parameters.

at a period T_c. The *Sampling Module* collects raw sensor data, computes a state estimate, and then broadcasts that estimate to the rest of the robot fleet every T_s ms. The *GPS Module* collects the continuous stream of raw data from the GPS unit. It notifies the *Sampling Module* any time a complete sentence is received and parsed, which occurs every T_g = 200ms for the 5Hz GPS unit. The time t_g spent in this module is not continuous but rather is distributed over a number of small tasks wherein a single character from the sentence is received.

In order for these tasks to be schedulable, the sum of the time expenses of the tasks cannot exceed unity [6]:

$$\sum_{j=1}^{n_t} \frac{t_j}{T_j} = U \le 1 \qquad (1)$$

n_t is the number of tasks, t_j and T_j are the execution time and period of the jth task, respectively, and U is the processor utilization. This relationship imposes a constraint on the values of sample period T_s and control period T_c. Because each robot broadcasts state information during each sampling module invocation, the incoming message period $T_i = T_s$ defines the minimum time required to receive one state update from each robot, and $t_i = (n-1)t_m$ where t_m is the time to process one incoming message, and n is the number of robots in the system. The time required to sample data, t_s, can be broken down into its individual components $t_s = t_{sd} + t_{se} + t_{sb}$ where t_{sd} is the time to retrieve and convert data from the sensors, t_{se} is the state estimation time, and t_{sb} is the time required to broadcast state information. The control time t_c depends on n because the potential function controller depends on the position of every other robot in the system. Specifically, $t_c = nt_{cp}$ where t_{cp} is the processing time required per potential field computation. Inserting these expressions into eq. 1 gives:

$$\frac{nt_{cp}}{T_c} + \frac{t_g}{T_g} + \frac{t_{sd} + t_{se} + t_{sb} + t_m(n-1)}{T_s} = U \qquad (2)$$

The worst-case values of the fixed timing parameters t_{cp}, t_g, T_g, t_{sb}, t_m, and t_{sd} as well as U were measured from benchmark tests of the Dynabot and are given in Table I. The state estimation time t_{se} depends on the choice of state observer (section III). t_{sb} and t_m depend on which state elements are broadcast; Table I assumes Cartesian position is broadcast to meet requirements of a potential function controller (section IV). While choosing $T_c < T_s$ is possible, as information from some robots may be received between sample periods, we restrict our investigation to the case where $T_c \ge T_s$ such that the controller is guaranteed to update on new information. Moreover, the only advantage of choosing $T_c > T_s$ would be to allow for faster sampling rates. However, examination of eq. 2 reveals that increasing T_c from T_s to ∞ results in at most a 10% decrease in T_s. Therefore, we constrain $T_c = T_s$. Eq. 2 gives an expression for T_c and T_s that is dependent only upon the number of robots and the choice of state estimation scheme:

$$T_c = T_s = \frac{t_{sd} + t_{se} + t_{sb} + t_m(n-1) + nt_{cp}}{U - t_g / T_g} \qquad (3)$$

TABLE I TIMING PARAMETER VALUES FOR THE DYNABOT

t_m	t_{sd}	t_{sb}	t_{cp}	t_g	T_g	U
1.23 ms	8.86 ms	3.40 ms	0.7 ms	32.0 ms	200 ms	0.95

III. TEST BED CHARACTERIZATION

A. Communication

Experiments were conducted using four Dynabot processors to produce empirical models of latency and packet loss ("error") rate. Although the model is particular to the Dynabot hardware, any system using UDP over 802.11 should exhibit similar properties. All communication is broadcast. A simplifying assumption is made that latency and error are direction- and distance-independent over the range used (roughly 150m). In testing to-date, this assumption has proven reasonable.

Latency is modeled as a stochastic number of constant delays generated by a Poisson distribution plus a non-stochastic offset time seen with every message. Both the offset time and the expected value (λ) are functions of packet length N and the number of robots n. Through repeated experiments with different numbers of nodes and message lengths between 1 and 125 bytes, an empirical model was found to characterize latency $L_m(n, N)$ of message m and λ as functions of N and n for $N \le 125$ bytes and $n \le 7$:

$$L_m(n,N) = t_k[\,floor(0.08N + 21 + 5.25(n-1) + k_m] \qquad (4)$$

$$\lambda(n,N) = (0.018n - 0.02)N + \left(0.06n^2 - 0.15n + 3\right) \qquad (5)$$

$t_k = 6.5 \times 10^{-4}$s and is limited by the accuracy of the microcontroller clock. The delay k_m is modeled as a Poisson-distributed random variable

$$P(k_m \mid n,N) = \frac{e^{-\lambda(n,N)} \lambda(n,N)^{k_m}}{k_m!} \,. \qquad (6)$$

Latency is determined per-message, i.e., all receivers are considered to have received a given message at the same time if they receive it at all.

The missed messages observed in experimentation were rarely missed by all of the nodes. This suggests that buffer overflow and low-level localized noise are the predominant error sources, and so a receiver-side error model is used. Although wireless communication is frequently modeled as bursty (due to RF noise), the error rates observed here, including other sources of error, did not display strong burstiness. A simple memoryless Bernoulli process is therefore used to determine whether a message is received. The probability of a given receiver r missing a given packet m is given purely in terms of the number of communicating nodes n , with $n \le 7$, by the empirically-determined model

$$p_{rm} = (1.01n^2 - 2.82n + 2.04) \times 10^{-3} \,. \qquad (7)$$

B. State Estimation

The basic state information of a given robot consists of absolute position X and Y, body fixed velocities v_x and v_y, and bearing ϕ. State uncertainty depends on fusion of inertial measurements and GPS data. Three data fusion strategies are considered to unearth tradeoffs between state estimation accuracy and processing time: GPS extrapolation, coupled Extended Kalman Filter (EKF), and decoupled EKF.

In GPS extrapolation, X, Y, v_x, and v_y are set to GPS position and velocity measurements every time a new GPS reading is available (every 200ms). Between readings, the velocity is assumed to be constant and a two-state Kalman filter tracks ϕ based on noisy bearing and yaw rate measurements from the nIMU. The simplicity of this filter makes it very fast but limits its accuracy to that of the GPS measurement.

The coupled EKF incorporates acceleration and inertial measurements with GPS to reduce state estimation error. In addition to X, Y, v_x, v_y, ϕ, it tracks biases on the yaw rate ($b_{\dot{\phi}}$), accelerations (b_{ax}, b_{ay}) and GPS position measurements (b_x, b_y), bringing the size of the state to 10. The large number of state variables makes this method computationally expensive.

A decoupled EKF is developed to reduce processing time by breaking the full state into sub-states tracked by individual EKFs. This approach is similar to the method described in [7, 8]. The choice of sub-states is guided by the magnitude of the filter gains within the coupled EKF. State variables with the largest associated Kalman gain for a given measurement are grouped together. The resulting state estimator preserves near-optimality and reduces computational burden by separating state elements from measurements that do not significantly influence them. The estimator consists of three two-state Kalman filters and one four-state EKF. For Kalman filters with more state variables than measurements, the computational complexity is $O(n_s^3)$ where n_s is the size of the state [9]. Therefore, the decoupled EKF ($n_s = 4$), is an order of magnitude less computationally expensive than the coupled EKF.

In order to bound the uncertainty of each estimation scheme, the Dynabot was driven around a circular path (constrained by a tether) while sensor data were collected. State estimates were computed offline using each algorithm. Trajectories are shown in Fig. 3 for each estimation scheme. Triangles along the trajectory indicate the estimated bearing. In each case, the uncertainty is taken as the maximum error between the estimated path and the best-fit actual path. The measured uncertainty is consistent with position covariance estimates derived from the EKF.

TABLE II. SUMMARY OF ESTIMATOR ACCURACY AND TIMING

Scheme	Uncertainty (m)	t_{se} (ms)
GPS interpolation	3.5	5.9
Decoupled EKF	1.1	13.8
Coupled EKF	0.7	114.2

Table II summarizes the measured accuracy and computation time for each method. The decoupled EKF improves state estimation accuracy over GPS interpolation by a factor of three. The corresponding expense is an increase in estimation time by just over a factor of two. In contrast, the coupled EKF offers a modest increase in accuracy at the cost of an increase in estimation time by a factor of over eight.

The communication and state estimation uncertainty models provide simple but useful models of the flow and uncertainty of information within the Dynabot fleet.

IV. SIMULATION METHODOLOGY AND RESULTS

We treat the vehicles as particles in order to distinguish between the effects of vehicle dynamics and information dynamics on computational resource allocation in cooperative control. A dynamic vehicle can be made to behave similarly to a point-mass robot given local traction and steering control laws [4], thus rigid-body dynamics are neglected in the results that follow.

The control method is based on artificial potential functions, in which one or more leaders manage flocking behavior of a group of robots [10]. Each robot and each leader is the center of its own radially symmetric potential function with a circular well of radius h_0 and d_0, respectively, and an attractive radius of h_1 and d_1, respectively. The force commanded on each robot is the gradient of the potential function plus a velocity-dependent dissipative force. For one robot and leader, the potential function V_h is

$$V_h = \begin{cases} \alpha_h\left(\ln(r)+\dfrac{h_0}{r}\right) & r \le h_0 \\ \alpha_h\left(\ln(h_0)+\dfrac{h_0}{h_0}\right) & h_0 < r \le h_0+\varepsilon \\ \alpha_h\left(\ln(r-\varepsilon)+\dfrac{h_0}{r-\varepsilon}\right) & h_0+\varepsilon < r \le h_1 \\ \alpha_h\left(\ln(h_1-\varepsilon)+\dfrac{h_0}{h_1-\varepsilon}\right) & r > h_1 \end{cases} \tag{8}$$

α_h is a scalar gain governing the gradient and the floor of the potential well is a ring of width ε that accommodates position uncertainty. A gain α_d governs inter-robot potentials.

Figure 4 shows trajectories in formation control of three

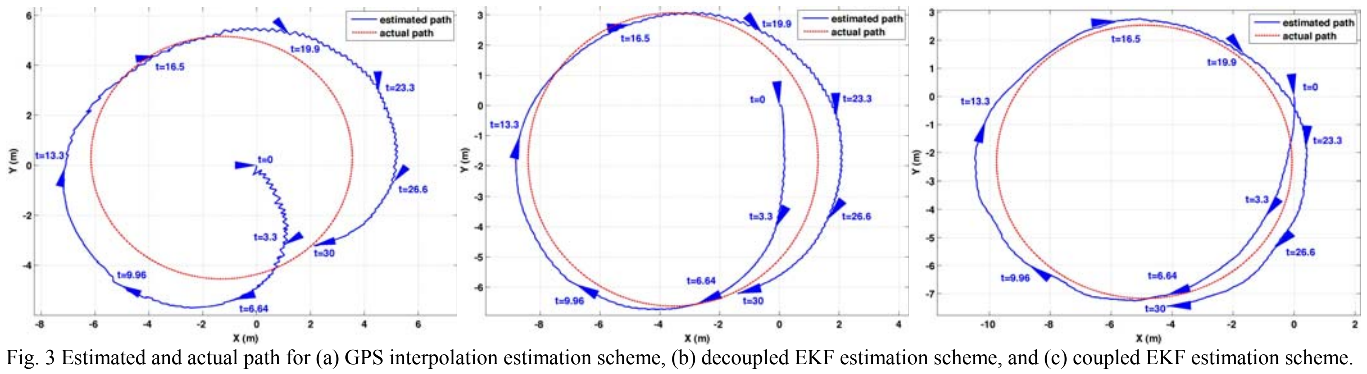

Fig. 3 Estimated and actual path for (a) GPS interpolation estimation scheme, (b) decoupled EKF estimation scheme, and (c) coupled EKF estimation scheme.

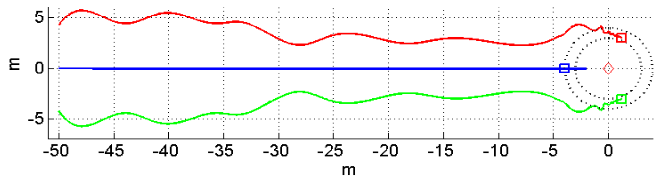

Fig. 4. Trajectories of three robots (in color) and one leader (diamond)

point-mass robots and one leader (diamond). The potential well is marked by circles $h_0 \pm 0.5\varepsilon$ around the leader, with h_0 = 3 m and ε = 1 m. The robots start at rest at a distance of 50 m from the leader and 4 m from each other and come to rest equally spaced in the potential well at $d_0 = h_0\sqrt{3}$, clustering as they proceed towards the leader. Potential function gains α_h and α_d are set to emphasize inter-robot interaction while limiting applied force magnitudes to values that can be solicited with the Dynabot hardware. Fig. 4 shows best-case performance for the given control parameters when perfect, instantaneous position information is assumed. Robots reach maximum speeds of 3.7 m/s along the ~50-m trajectory.

We use the configuration of Fig. 4 to illustrate the relative stability and performance of potential function control with the three state estimation methods. $T_s = T_c$ according to eq. 3 for each state estimation scheme. α_h and α_d are determined for each T_c so as to maximize bandwidth given control rate and position uncertainty, while observing physical limits on force commands. Only the decoupled EKF can use the gains of the no-uncertainty case without adjusting for position uncertainty, communication latency, or control rate.

We simulate the state estimation error e_{xk} as a steady-state Gauss-Markov sequence driven by Gaussian white noise w_k shaped so $E[{e_{xk}}^2] = \sigma^2$, where σ is the uncertainty measured experimentally for each observer (Table II). This model takes the place of simulating GPS and the EKFs in order to decouple observer dynamics from tradeoffs between performance and computational resource allocation. Communication latency is modeled on the message level. When position is estimated onboard each robot, the nodes that receive position information are determined with probability given in eq. 7. Then the latency is determined for all receivers using eq. 4-6, and the nodes receive the message after the delay has passed.

Table III presents settling time, collision rate, and completion rate for each observer based on 100 trials. Numbers in parentheses denote standard deviations. Collision rate is the percentage of trials that end in collision and is a measure of stability. Settling time is the time to reach 1% of the steady-state position averaged over the three robots and measures transient response. Completion rate measures the frequency with which all robots settle within a radius of $\pm\sigma$ of the goal position, with σ from Table II for each method. It measures ability to achieve a formation given the position uncertainty.

TABLE III. COMPARISON OF ESTIMATION METHODS

Estimation Method	$T_C = T_S$ (ms)	Gains α_h and α_d	1% settling time (s)	Collision rate (%)	Completion rate (%)
Perfect information	10	650, 300	33	0.0	100
Interpolated GPS	27	300, 150	68 (11)	15(4)	42(5)
Decoupled EKF	36	650, 300	37(8)	0	58(5)
Coupled EKF	156	500, 300	41(4)	0	24(4)

For comparison, metrics are reported for no position uncertainty and t_{se} = 0. In Table III, interpolated GPS exhibits poor stability, even with gains set for cautious navigation, which is the result of large position uncertainty. The decoupled EKF performance is modestly better than the coupled EKF, as control gains must be reduced for T_C = 156 ms to avoid control saturation. The decoupled EKF best completes the task, as measured by the completion rate.

V. CONCLUSION

For multi-robot systems subject to limited communication, processing and sensing, an improvement in formation control performance is achieved through consideration of computational resource allocation. In this paper, we have focused on the tradeoff between state estimation accuracy and processing time for a parallel task control architecture that seeks to maximize processor utilization. For a system based on empirical models from the Dynabot hardware, decoupled EKF provides the best balance of accuracy and update frequency of three state observers considered. While the differences in performance are subtle for a system of three robots, increasing the number of robots is expected to amplify these differences.

For each method, it is possible that increasing the content of exchanged state information can compensate for aged information from other robots, uncertainty, and control delay at the expense of increased communication latency and error. In future work, we will investigate the stability and performance tradeoffs when the content of exchanged information (and corresponding communication delay) increase.

REFERENCES

[1] E. Yoshida, T. Arai, M. Yamamoto, and J. Ota. Local communication of multiple mobile robots: Design of optimal communication area for cooperative tasks. *J. Robotic Systems*, 15(7):407–419, 1998.

[2] J. K. Yook, D. M. Tilbury, and N. R. Soparkar. Trading computation for bandwidth: Reducing communication in distributed control systems using state estimators. *IEEE Trans. Control Sys. Tech.* 10(4):503–518, 2002.

[3] J. D. Sweeney, H. Li, R. A. Grupen, and K. Ramamritham. Scalability and schedulability in large, coordinated, distributed robot systems. *Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation, Taipei, Taiwan*, volume 3, 4074–4079, 2003.

[4] L. E. Ray, D. Brande, J. Murphy, and J. Joslin. Cooperative control of autonomous mobile robots in unknown terrain. *Proc. ASME Int. Mech. Eng. Conf. and Exposition, Chicago, Illinois*, 2006.

[5] J. Joslin. The design, construction, and control of the DynaBot testbed. Master of Science thesis, Thayer School of Engineering, 2007.

[6] K. Jeffay, D.F. Stanat, and C.U. Martel.On Non-Preemptive Scheduling of Perirodic and Sporadic Tasks. *Proc. Real-Time Systems Symposium*, 129-139, 1991.

[7] F. E. Daum and R. J. Fitzgerald. Decoupled Kalman filters for phase array radar tracking. *IEEE Trans. Automatic Control*, 28(3):269–283, 1983.

[8] S. R. Rogers. Steady-state performance of the decoupled kalman filter. *Proc. of the Aerospace and Electronics Conference*, 334–339, 1988.

[9] M. Gioris, D. Gray, and J. Mareels. Reducing the computational load of a kalman filter. *Electronics Letters*, 33(18):1540–1541, 1997.

[10] N. E. Leonard and E. Fiorelli. Vitrual leaders, artificial potentials and coordinated control of groups. *Proc. 40th IEEE Conf. on Decision and Control*, 2968–2973, 2001.