Transversal Issues in Real-Time Sense-and-Respond Systems

Ahmad T. Al-Hammouri
Case Western Reserve University
E-mail: ata5@case.edu

Vincenzo Liberatore
Case Western Reserve University
E-mail: vxl11@case.edu

Huthaifa A. Al-Omari
Case Western Reserve University
E-mail: haa6@case.edu

Stephen M. Phillips
Arizona State University
E-mail: stephen.phillips@asu.edu

Abstract

Networked S&R systems extend human capabilities beyond temporal and spatial barriers with useful applications in broad areas. Involving the physical-world environments, S&R systems must fulfill the intrinsic real-time requirements of these physical environments. However, communication networks lack of QoS can hamper performance and effectiveness of S&R. Therefore, end system strategies must be deployed to retain effectiveness and to enhance performance of S&R. In this paper, we survey transversal issues pertaining to the development of agile S&R systems to work over a geographically scalable network.

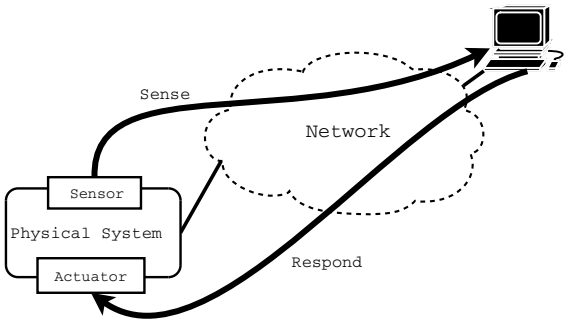

Figure 1: Simplest example of S&R system (with a feedback loop between sense and respond).

1 Introduction

Networked sense-and-respond systems (S&R) extend human reach beyond temporal and spatial barriers. Remote physical environments can then be monitored, controlled, and affected through communication networks (Fig. 1). Examples of potential applications include industrial automation [15], automatic asset management [7], distributed instrumentation [1, 16], disaster recovery [14], unmanned vehicles [11], and home robotics [16].

S&R systems involve a physical environment from which they inherit the real-time characteristics. To ensure operation correctness and to attain maximal performance levels, delivery timelines of sense and respond messages must fulfill the real-time constraints mandated by the specific physical environments. On the other hand, communication networks have inherent non-deterministic behavior, and provide no timeliness or QoS guarantees on data delivery. Consequently, S&R must deploy strategies to alleviate networks' non-determinism and lack of QoS, such as bandwidth limitations, packet losses, delays, and delay jitter. Due to the Internet's end-to-end principle, most of the complexity associated with these strategies must be pushed to end-systems.

Related attempts to solve these problems have focused on control theory, middleware, and databases; whereas networking research has been conspicuously absent in the arena of S&R. Nonetheless, existing methods, such as play-back buffers and congestion control, stress the fact that these problems fall within the scope of networking and thus networks are critical to the resolution of these problems. Since these methods cannot be used the way they are—because they were not developed specific for S&R systems—they must be adapted to work with the S&R systems. This paper presents our conceptual framework for understanding these problems and the open issues that we think are the most pressing and critical for further advancements in the area of S&R. Based on our extensive experiences in the fields of Internet robotics (e.g., [1, 16, 2]), networked control systems (e.g., [5, 8]), industrial automation[17], and COTS middleware platforms (e.g., [2, 11]), we will draw and identify transversal issues related to the development of S&R systems to work effectively over best effort geographically scalable networks.

Heterogeneity of S&R. S&R control environments can differ radically in complexity and in applications. Such environments can range from simple linear systems as in the case of a thermostat to very complex ones, which might include systems of subsystems, as in the case of unmanned autonomous vehicles (UAVs) [11] and in the case of value-chains in manufacturing [9]. Moreover, some systems may include different hierarchical levels of complexity abstraction. For example in UAVs, there are several hierarchical levels. At the lowest level is the direct force level. At this level, the on-board controller issues tasks, such as rotating motors forward or reverse, based on feedback information supplied by sensors (this represents local sense and respond). At the highest level of abstraction are software agents. Software agent carry out high-level tasks, and are responsible of coordinating multiple UAVs into task-oriented teams, which can have impact on different applications, e.g., military transformation [12]. On the other hand, an online auction S&R system would comprise only the software-agent level.

In spite of the striking heterogeneity across different systems' typologies and levels, there are several transversal issues common to the development of agile S&R systems. Transversal issues arise because they address the necessity of providing network QoS for S&R systems to meet their inherent real-time requirements. This paper presents these transversal issues with illustrative examples. These transversal issues are: encapsulation of inner loops (Section 2.1), adaptability and tolerance (Section 2.2), and congestion control (Section 2.3). This paper also highlights differences exist between real-time S&R systems and other real-time applications.

2 Transversal issues

This section discusses the various transversal issues pertaining to real-time S&R systems. It is worth noticing that there are other issues that are critical to distributed systems and applications in general and are not specific to real-time S&R. These issues, such as security, are not covered in this paper.

2.1 Encapsulation of inner loops

An S&R system can include an internal sense-and-respond loop whose actions, i.e., sense or respond, are local to a particular system and are not exchanged over the network, as shown in Fig. 2. This nested sense-and-respond loop establishes a particular form of hierarchical control that highlights the boundary between local and global S&R. Such a configuration often arises in practice. For example, in an *online auction*, a bidder chooses

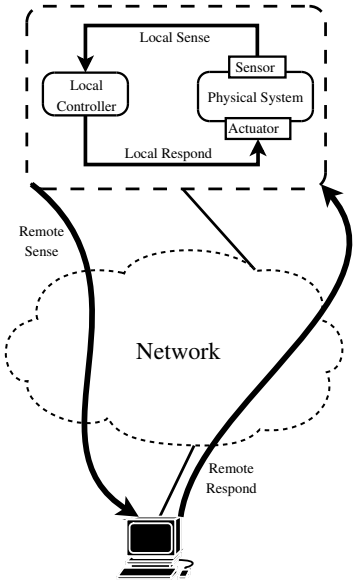

Figure 2: Local and remote sense and respond in S&R.

a maximum price for a particular item (the outer loop). Based on other bidders' behaviors, the auction's agent places bids on this bidder's behalf up to the maximum price (the inner loop). Whenever the bidder's maximum price is outbidded, the auction's agent conveys this to the bidder, who then reacts by choosing a higher bid or by withdrawing from the auction (again, the outer loop). In general, an S&R system in a networked environment will *expose* a certain interface (in the example, the maximum price a bidder is willing to pay for an item), and will *encapsulate* locally other functionality that can include complex decision strategies (in the example, the placing of higher bids based on other bidders reactions). The decision on what to expose globally and what to encapsulate locally is transversal to all applications and taxonomies of S&R and it is critical for the system behavior. In our experience, we have developed the following two general principles to guide the design process regarding encapsulation [1, 11].

In the first place, a system behavior should be controlled locally whenever possible. In many cases, this implies that the remote control, i.e., the outer S&R loop, has access only to the variables that affect the global coordination of the system. Conversely, the remote control would not have access to those variables that are not relevant for distributed control. For example, placing a higher bid on behalf of a bidder is accomplished locally at the auction site, however the maximum bid depends on the willingness of the bidder on how much to pay and so it must be obtained remotely. There are several reasons why S&R loop should be as local as possible. First, the

system's responsiveness and performance are typically superior when a local loop is used. Second, the safety of the system often depends on failback measures that need to be implemented locally to avoid potential failures or unreliable levels of service in the network infrastructure. Finally, a local loop reduces the demands on the communication network. However, some communication is required if cooperative and coordinated action is needed. In general, different variables are relevant in the global setting depending on the application and therefore different variables are exposed or encapsulated depending on the application. Furthermore, power or computational constraints can require certain variables to be remotely controlled even though they could have been otherwise encapsulated locally.

Second, the local control can be often implemented in many different ways and the local implementation can significantly affect the behavior of the globally coordinated system. For example, certain manipulation robots can follow a reference arm location through either *position control* or *force control*. Position control usually results in the robot executing its task quickly. However, a position-controlled robot can apply forces large enough to cause damage to its environment, to the robot, or to both. An alternative to position control is a version of *force control*, whose advantage is that the robot can be compliant with its environment and does not introduce damaging forces as long as certain parameters are appropriately chosen [1]. For example, in the workspace shown in Fig. 3, the ParaDex robot is required to set the position of levers and switches. Such tasks were achieved in force control [1] but position control would have introduced a significant damage in this task space. Force control and position control present a similar interface to a remote controller, but in this application force control was more appropriate for a networked environment in that the robot remains gentle even when connectivity is poor.

2.2 Adaptability and tolerance

To operate effectively over best-effort wide-area networks, S&R systems with tight real-time requirements must be adaptable to network levels of service as expressed, for example, by delays, jitter, packet losses, and bandwidth. A general S&R model is shown in Fig. 1. In the figure, the physical system generates a *sensed* sample i, wraps it inside a packet, and sends it to the other end host, which processes the sensed data and then generates a *respond* message i. The respond message arrives back the physical system after a round-trip delay d_i since when the the sensed sample i was first generated. Due to the nondeterministic nature of communication networks,

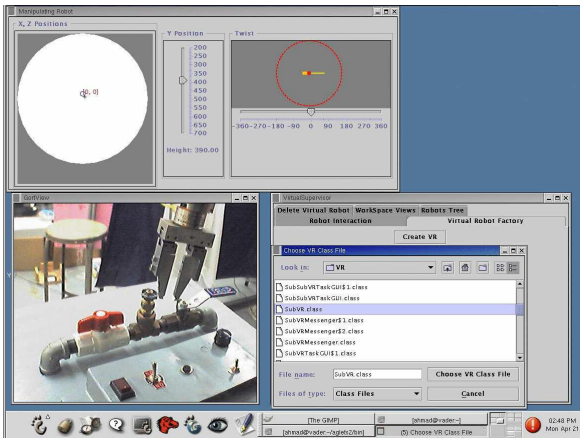

Figure 3: Robot manipulates switches, levers, and valves through force control [1].

delays (d_i's) vary randomly over time. Because of this, we decompose d_i into two components as: $d_i = \tau + \xi_i$, where τ is a nominal round-trip time between the two end hosts and is constant across i's; whereas, ξ_i is the variation, i.e., the *jitter*, in the round-trip delay i from the nominal value. This decomposition allows us to partition the problem of time-varying delays into two subproblems: fixed delays and jitter. Apart from delays and jitter, either sense or respond packets may be dropped out from the network. In the subsections to follow, we thus discuss methods to address these three issues, i.e., fixed delays, jitter, and packet losses, while bandwidth allocation is deferred to Section 2.3.

2.2.1 Round-trip delay

S&R systems could involve systems with inherent physical dynamics that require upper limits on delays. Consequently, long delays degrade performance and may result in S&R systems' misbehavior. In the online auction example of Section 2.1, an item's current bidded price and participants' bids must be conveyed in a timely manner and must not be delayed longer (say, after the auction's time elapses) such that the auction proceeds in an efficient way.

When delays are fixed, i.e., $\xi_i = 0$, several methods can be deployed to ameliorate the adverse effect of delays. One method is by sacrificing performance to preserve effectiveness. For instance, when teleoperating a robot in presence of transmission delays, an operator would issue commands in smaller steps, for example, the operator would command a robot to move only a couple of inches, then he would hold back waiting to see the visual feedback before issuing another command, and so

on. A second method is to use observers. An *observer* is an approximate model of the physical system used to simulate system's dynamics. Based on the latest sensed information, the observer extrapolates the system state to predict a future state after one round-trip time, and then the respond signal is issued accordingly. Not all system dynamics can be captured and predicted by observers because some systems have complex dynamics and/or complex interactions with their environments, one such example being the online auction. A third method is using encapsulation in that the system parts that necessitate short delays are implemented locally—if this is possible, see Section 2.1.

2.2.2 Delay jitter

Jitter leads to unpredictable delivery-times of sense and respond messages. Whereas some S&R are not sensitive to jitter as long as round-trip delays are within some range, others are. Examples of the first is the online auction; and of the second is networked control systems [18]. In addition, jitter causes inaccurate predictions in those systems that use delay-compensation techniques based on observers (see Section 2.2.1). Play-back buffers can be deployed at the physical system side to smooth jitter and to apply control signals at predictable times, i.e., ensuring the situation of $\xi_i = 0$ and thus $d_i = \tau$. Play-back buffers have been used in multimedia streaming [10]. However, in multimedia streaming, jitter is addressed in terms of one-way delays, and play-back delays are determined by the end-system that receives the stream. In S&R, on the other hand, the situation is different. First, jitter in the round-trip delays is the concern. Second, play-back delays are determined by the controller, which is away from the host that plays back (i.e., applies) the signal. Furthermore, multimedia and S&R possess different performance metrics. Therefore, methods and algorithms used in multimedia play-back must be adapted or changed to work for the case of S&R.

The idea of play-back buffers is that packets are delayed by the receiver a certain amount of time, called play-back delay, before being processed. Packets are processed only if they arrive before the scheduled processing times; otherwise, i.e., a packet arrives after the scheduled time, it is dropped and is considered lost. Therefore, choosing an appropriate value for the play-back delay should compromise between delay (see Section 2.2.1) and losses (see Section 2.2.3).

2.2.3 Packet losses

S&R traffic carries vital data and hence packet losses are undesirable. Although packet losses can be minimized with methods like *Forward Error Correction* [10], they cannot be entirely eliminated. Retransmission of lost packets is an inappropriate solution, either. This is true because by the time a packet has been discovered to be lost and retransmitted, the system would have evolved to a newer state—and thus the retransmitted packet would have been based on stale information.

When a packet carrying either a sense or a respond information is lost, then the system behavior is equivalent to the case when the corresponding sensed sample was not generated at all. Effectiveness of an S&R system can still be attained if subsequent packets arrive intact and that the rate of generating and sending sensed data is higher than the rate of variation of the system's state. Let us illustrate with an example. A thermostat is installed to measure the temperature of an environment. The temperature varies as 1^o per minute. Suppose the whole system is sensitive to variations of 0.1^o and the temperature measurements are made 50 times a minute. In every five consecutive packets, if four are dropped out from the network, the system will continue to function properly.

Oversampling, which is sensing at a rate higher than what is actually needed, makes S&R systems more tolerant to packet losses. However, two important issues need to be addressed when oversampling. First, oversampling increases demand on the bandwidth and may cause congestion, which in turn leads to more packet delays and losses. Second, there must be a criterion to ascertain how much to oversample. In Section 2.3, we address both of theses issues.

2.3 Congestion control

The introduction of congestion control into TCP solved the problem of congestion collapses that were occurring during 1980s. Congestion control was one of the reasons that the Internet scaled up to its size today. Due to the original philosophy of Internet—the end-to-end principle—the entire implementation of the congestion control scheme was delegated to end-systems, which are senders and receivers.

The main objective of congestion control is to match senders' transmission rates to network capacity. Congestion control is transversal among all the levels discussed in Section 1. At the agent level, agents are applications built atop the transport layer and in most cases the conventional TCP congestion control is involved when agents move from a place to another or when they exchange high-level messages. Therefore, we will not cover it here. For lower level layers, congestion control is used to provide a *fair* bandwidth allocation:

- Between S&R real-time traffic and other non-real-time traffic, and

- Among different S&R traffic.

We emphasize that *fair*, here, should not be mistakenly understood to mean even or equal; rather, it means that allocation is related to intrinsic bandwidth requirements and is affected by the environment to be controlled. Allocating bandwidth between S&R and other non-real time traffic is necessary because S&R traffic is less elastic than other traffic and it requires minimum bandwidth guarantees. Allocating bandwidth among different S&R traffic is necessary because different physical environments differ in speed of physical dynamics. As a result, how frequent sense-and-respond messages must be exchanged—and thus the bandwidth allocation—differ from one system to another. We elaborate on this with an example.

Assume a hypothetical scenario where two S&R's used to control the process of filling two irrigation tanks with water up to a certain level. Assume further the two tanks have the same capacity but being supplied with two different water pipes: one supplies water at rate 20 liters per minute and the other supplies at rate 180 liters per minute. The level of the second tank needs to be sensed at least 180 times per minute so that the error in the water level may not exceed 1 liter. On the other hand, it is sufficient for the first tank to be sensed at least 20 times per minute so that not to exceed an error of 1 liter. If the two S&R's share one network link that can transmit data at a maximum rate of 300 sensed samples per minute, the question then becomes: how to divide the bandwidth among these two systems in an efficient way and to obey the aforementioned requirements. One feasible solution is to give each system its minimum requirements, i.e., 20 samples per minute and 180 samples per minute for the first and the second respectively, and then to divide the remaining unused bandwidth in anyway between the two. Another natural and better way is to divide the bandwidth proportional to the speed of water flow, i.e., 30 samples per minute and 270 samples per minute for the first and the second systems respectively.

This particular example is easy because of two reasons. First, the environment is static, that is, we have fixed number of systems and fixed network parameters. Second, the dynamics of each system, i.e., the rate of water flow, is globally known. In practice, however, systems use and relinquish network resources in a dynamic manner and one system may not know dynamics of others. In [4], an admission-control protocol is proposed to solve these issues. Systems need to register with a master node before gaining admission to the network. The bandwidth is time-divided among active systems, that is, each systems is given specific time slots during which it can send or receive data. The disadvantages of this approach are that it is fully centralized and it requires clock synchronization among all entities in the network.

Our contribution in this area is to devise a distributed approach that is scalable, flexible and reconfigurable. We measure the performance of each S&R system, i, by a performance function, $U(A_i, w_i)$. A_i captures the physical dynamics of system i, and w_i is the bandwidth allocated to system i. Bandwidth w_i can be thought of as the rate at which a S&R system can generate and transmit sensed samples. We mathematically formulate the overall objective as follows:

$$\begin{aligned} \max \quad & \sum_i U(A_i, w_i), \\ \text{s. t.} \quad & \sum_{i \in S(l)} w_i \leq C_l, l = 1, \ldots, L \end{aligned} \tag{1}$$

where $S(l)$ is the set of systems whose end-to-end flow paths use link l, B_l is the capacity of link l, and L is the total number of links in the network. In other words, the objective is to allocate the network links capacities in such a way to maximize the aggregate performance of all the systems. We use the results and the techniques published in [13] to achieve the objective expressed in (1) in a distributed manner. Specifically, the authors in [13] formulated the problem as a convex optimization problem and they used the Lagrange multipliers method to decompose the problem into separable subproblems. Based on their approach, each link computes a so-called *price* variable, which measures the mismatch between aggregate flow rates and the capacity of the link, i.e., the congestion at the link. Links report prices back to end-systems whereby end-systems adjust their sending rates (i.e., the frequency of generating and sending sense and respond messages) appropriately. For the solution to converge to an equilibrium, the performance function must be concave, monotonically increasing, and twice continuously differentiable; and its second derivative must be greater than zero. The advantage of this approach can be manifested in two scenarios. First, when there are few S&R systems competing for the network, this approach allocates the bandwidth in a way that every system operates with high performance. Second, when there are too many S&R systems, the network bandwidth may not be sufficient to ensure high performance for each system. However, the approach allocates the bandwidth in order to achieve the highest aggregate performance for all S&R systems.

In [3], we demonstrated the applicability of this approach for the class of scalar linear control systems whose system dynamics evolve according to the following differential equation:

$$\dot{x}(t) = ax(t) + u(t) \tag{2}$$

where $x(t)$ is the system state; a is the system constant—a is larger for faster system dynamics; $u(t) = -kx(t_j)$ is the *respond* signal, which is simply a constant, k, multiplied by the last *sensed* signal.

A representative performance function would take the form of:

$$U(A_i, w_i) = \frac{a_i - k_i}{a_i} e^{\frac{a_i}{w_i}} \quad (3)$$

This definition is based on the error, derived in [6], that the system develops when using bandwidth equal to w_i. Note that $U(A_i, w_i)$ in (3) satisfies all conditions mentioned before.

3 Conclusion

In this paper, we have discussed fundamental issues for designing effective real-time S&R systems over communication networks. Such issues stem from two facts. First, communication networks are typically best-effort media with no QoS provisioning guarantees. Second, operation correctness and performance of real-time S&R systems depend on their ability to adapt to networks levels of service. Thus, intelligent end-system strategies must be deployed to conceal the adverse effects of networks' lack of QoS on the effectiveness of real-time S&R systems.

Acknowledgments

We thank Michael S. Branicky and Nathan Wedge for helpful conversations. This Work has been supported in part under NSF grant number CCR-0329910, Department of Commerce grant number TOP 39-60-04003, and NASA contract number NNC05CB20C.

References

[1] A. Al-Hammouri, A. Covitch, D. Rosas, M. Kose, W. S. Newman, and V. Liberatore. Compliant control and software agents for internet robotics. In *Eighth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS)*, 2003.

[2] Ahmad T. Al-Hammouri. A distributed framework to facilitate human-robot remote interaction. Master's thesis, Case Western Reserve Universiy, Cleveland, Ohio, Januray 2004. Advisor: Vincenzo Liberatore.

[3] Ahmad T. Al-Hammouri and Vincenzo Liberatore. Optimization congestion control for networked control systems. In *IEEE Infocom Student Workshop*, Miami, FL, March 2005. Abstract.

[4] L. Almeida, J.A. Fonseca, and P. Fonseca. A flexible time-triggered communication system based on the controller area network: Experimental results. *FeT'99, Int. Symposium on Fieldbus Technology*, September 1999.

[5] M. S. Branicky, V. Liberatore, and S. Phillips. Co-simulation for co-design of networked control systems. In *American Control Conference*, 2003.

[6] Michael S. Branicky, Stephen M. Phillips, and Wei Zhang. Scheduling and feedback co-design for networked control systems. In *Proc. IEEE Conf. on Decision and Control*, Las Vegas, December 2002.

[7] Alexander Brewer, Nancy Sloan, and Thomas L. Landers. Intelligent tracking in manufacturing. *Journal of Intelligent Manufacturing*, 10(3-4):245–250, September 1999.

[8] Justin R. Hartman, Michael S. Branicky, and Vincenzo Liberatore. Time-dependent dynamics in networked sensing and control. In *American Control Conference*, Portland, OR, June 2005.

[9] Albert Jones and Abhijit Deshmukh. Test beds for complex systems. *Commun. ACM*, 48(5):45–50, 2005.

[10] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley Longman, Inc., 2001.

[11] V. Liberatore et al. IP communication and distributed agents for unmanned autonomous vehicles. In *AIAA-UAV*, 2003.

[12] Grace Y. Lin and Jr. Robert E. Luby. Transforming the military through sense and respond, January 2005. IBM White paper.

[13] Steven H. Low and David E. Lapsley. Optimization flow control—I:basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.

[14] L. Matthies et al. A portable, autonomous, urban reconnaissance robot. In *The 6th International Conference on Intelligent Autonomous Systems*, July 2000.

[15] W. S. Newman et al. Design lessons for building agile manufacturing systems. *IEEE Trans. Robotics and Automation*, 16(3):228–238, 2000.

[16] Marco L. Ngai, Vincenzo Liberatore, and Wyatt S. Newman. An experiment in remote robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2190–2195, 2002.

[17] B. P. Robinson and V. Liberatore. On the impact of bursty cross-traffic on distributed real-time process control. In *Workshop on Factory Communication Systems (WFCS)*, 2004.

[18] Z. Wei, M. Branicky, and S. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84–99, February 2001.