Energy Budget Transaction Protocol for Distributed Robotic Systems

S.S. Groothuis and S. Stramigioli

Abstract—Passivity is a necessary condition for a system’s stability, meaning that when energy is generated, a system may readily become unstable. Passivity can be enforced using energy-aware actuation that monitors the amount of energy that is injected into or extracted out of a system. This type of actuation requires an energy budget to be used to execute a task. Communicating these energy budgets should be done carefully as to prevent accidental generation of energy. Therefore, this paper proposes an energy transaction protocol to communicate energy budgets in a distributed robotic system to guarantee that passivity is kept in a system. Simulations are performed with a model of the protocol with a highly unreliable communication channel to verify that the protocol indeed keeps passivity.

I. INTRODUCTION

Guaranteeing safety in the interaction between humans and robots is one of the most important criteria in the development of robotic systems. Safe physical human-robot interaction (pHRI) is one of the topics that is treated in the SoftPro project [1], in which novel prostheses and wearable robotic systems are developed as well as their control systems. A big step towards achieving safety is to ensure that a system is unconditionally stable. It has been proved in [2] that a necessary condition to ensure stability during interaction with any environment is that the total system is passive. A system is said to be passive if its energy content is always less than or equal to the energy that has been added through a power port (an interface through which energy can be exchanged) plus the amount of energy that was initially present [3]. Passivity can be enforced in several ways [4], [5], [6], among which is energy-aware actuation [7]. This type of actuation monitors the amount of energy that is injected into or extracted out of a system by an actuator. It has been shown that a high level supervisor can allocate and communicate (virtual) energy packages to lower level (actuation) controllers, and that an amount of (physical) energy up to a certain budget is used by the actuation system, after which task executed is halted to not violate passivity [8].

Communication is never instantaneous and always incurs a delay on a message that has been sent. If this communication channel is part of a real-time control loop, instability can easily occur. Passivity is, again, a way to guarantee stability

Fig. 1: Energy in a system is composed of the energy of separate systems Σ_A and Σ_B , and the energy in the communication channel (E_T).

of a system using an unreliable communication channel with varying jitter [9]. It is also important to realize that energy in a distributed system is composed of the energy of separate systems Σ_A and Σ_B , and the energy in the communication channel (E_T), as shown in Figure 1. Its network architecture may be assumed to be fault-free. However, communicating actuation setpoints with corresponding energy budgets may be a source for the loss of passivity depending on the resilience of the system to network errors like, e.g., data corruption, duplication, packet loss, and out of order delivery. If, due to network errors, an energy budget is delivered more than once and is accepted by the client as valid energy, passivity is lost since energy is generated. Also, sending an energy budget to a client but waiting for that budget to arrive at the client before the sender subtracts that budget from its own budget, will violate passivity during the transfer. Communicating energy budgets in a naive way may also result in inefficient use of (virtual) energy to accomplish tasks. This happens, for example, if a transmitted energy package is completely dismissed after it has been lost in the network, despite that energy not being used as valid budget energy and not being injected in the system.

These issues may partly be solved by choosing a network protocol capable of handling the network errors. The Transmission Control Protocol (TCP), for example, guarantees packet delivery in the correct order, but is not suited for timely (real-time) delivery. The User Datagram Protocol (UDP) can be used in real-time situations since it requires less overhead, but there is no guarantee of packet delivery. EtherCAT is a widely used Ethernet-based fieldbus system [10], [11] that can be used in real-time control systems. It uses standard 100BASE-TX Ethernet communication and associated hardware to achieve that latencies are bounded, such that a certain operating frequency can be guaranteed. This is only possible with communication over cables. That means that it cannot be used in a distributed system, since it cannot make use of other communication channels like WiFi. Hence, telemanipulation over longer distances, or communication between multiple disjoint systems through various communication channels is not possible.

To be able to passively communicate energy budgets in

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 688857 (SoftPro). The content of this publication is the sole responsibility of the authors. The European Commission or its services cannot be held responsible for any use that may be made of the information it contains.

The authors are affiliated with the Robotics and Mechatronics group, Technical Medical Centre, University of Twente, The Netherlands, and the second author also with the Bio-mechatronics and Energy-Efficient Robotics group, ITMO University, St. Petersburg, The Russian Federation. Email: {s.s.groothuis,s.stramigioli}@utwente.nl

a distributed robotic system, independent of the specific network architecture, a formal message transaction has to be used. An example of such a transaction mechanism is a consensus protocol. A consensus protocol [12] facilitates the reaching of agreement among agents participating in a transaction about a proposal. Therefore, all agents commit to a change of the same value. These protocols have been used in the context of distributed database systems [13], [14], and multi-agent robot control [15], [16]. These consensus protocols can be employed whenever many separate agents need to agree on a proposal done by a coordinator. In a distributed robotic system with energy budgets being communicated for separate actuation systems, there is no consensus on energy since it will be different for each actuation system.

Therefore, this paper proposes an energy allocation transaction protocol for proper book-keeping of communicated energy budgets for ensuring passivity of a system. The protocol consists of two state machines: one for the supervisor, and one for the client to which energy is to be allocated. These state machines define a sequential procedure followed by the supervisor and client such that the system remains passive after a transaction of energy budgets is complete, making the protocol independent of the specific network architecture.

This paper continues with a treatment on the closely related financial transactions and their similarity with energy budget transactions in Section II. The Energy Transaction Protocol is elaborated in Section III. A case study showing the workings of the proposed protocol through simulations with a model is presented in Section IV, while the results are discussed in Section V. The paper concludes with Section VI.

II. FINANCIAL TRANSACTIONS

The communication of energy while ideally keeping the total amount of energy in the distributed robotic system constant is similar to digital money transfer in which money should be transferred from one account to the other, and where the notion of passivity also holds.

A. Passivity of financial transactions

Suppose €1 needs to be transferred from account A to account B, with each account having a positive balance of €10. The total amount of money is €20. When the transfer is done (ignoring transfer fees etc.), €1 should be deducted from the balance of account A resulting in €9 and be added to the balance of account B, resulting in €11. The total amount is still €20, meaning the “system” is passive. If €1 would be deducted from account A, but along the way the message to increase the balance of account B is lost, the total amount would be €19. This is still a passive system, and more precisely a dissipative system, but this is clearly not acceptable since money is lost. If the balance of account A is held at €10 while the balance of account B increases to €11 and in between the message that account B has indeed received €1 meaning that €1 can be deducted from account A is lost, the total amount of money in the accounts would be €21. Although not immediately problematic and actually quite beneficial for the account holders (where one of them

has received the money without the other losing money), this situation of money generation by a transfer of funds is not desired for economic reasons. The system is then not passive (or even dissipative), but actually active (generative).

B. Transaction properties

To ensure valid transactions, the so-called ACID transaction properties can be used, which stands for Atomicity, Consistency, Isolation, and Durability, which is a paradigm originating from data manipulation in databases [17], [18]. Atomicity is the property that operations are indivisible and irreducible, and that either all or none occur. Consistency means that a consistent valid state of the system should be kept, so, for instance, that cells in a database will have the correct datatype. Isolation means that a concurrent execution of operations on a single entity are treated as a sequential execution, i.e., partial effects of one transaction are not visible by another transaction. Durability means that a transaction is only successful if it cannot be changed by external influences anymore, e.g., a power loss.

Besides the forming of the Society for Worldwide Interbank Financial Telecommunication (SWIFT) to standardize the interbanking messaging network [19], various protocols to facilitate money transfers have been developed [20], [21], [22]. To adhere to the atomicity property, the Netbill protocol [23], for example, implements a procedure in which a customer is only charged after receiving the encrypted desired information and the corresponding decryption key is only delivered to the customer after payment is successful.

C. Energy transactions in this context

The ACID properties can be placed in the context of energy transactions. Atomicity can be translated as the property that energy cannot be “given” without it also being subtracted somewhere, i.e., one action is coupled to the other. It will later be seen that, in case of uncertainty, energy may be subtracted while it has not actually been “given”. Consistency can be interpreted as the property that the total energy contents in the system should ideally be constant after the transaction. Isolation is achieved when partial effects of an energy transaction, e.g., receiving a new energy budget, are not visible to another transaction, i.e., concurrent transactions are to be treated as sequential transactions. An exception is that when a transaction has started, the energy it wants to communicate should be reserved and be made inaccessible to other transactions, otherwise energy could be generated. Durability is achieved when local energy budgets cannot be changed by some influence after a transaction is final. A power loss, as considered before, is less applicable here since the system would then be reinitialized.

III. ENERGY TRANSACTION PROTOCOL

A. General principle

The envisioned transaction protocol in a distributed robotic system to communicate allocated energy budgets is shown in Figure 2. It is assumed that a supervisor communicates with multiple (j) clients through communication channels.

Fig. 2: Communication in a distributed robotic system to transfer energy budgets from a supervisor to clients.

The supervisor has a usable budget and a transaction budget. The usable part of the supervisor budget is the energy that the supervisor can allocate to clients. The transaction budget of the supervisor is the reserved energy that is intended as a client's allocated energy, i.e., the transaction intention. Therefore, during a transaction that part of the budget is unusable for other transactions with other clients. On the client side a similar distinction between the usable budget and transaction budget can be made. The client budget is the usable energy budget that can be used by the local actuation control system to perform a task. The transaction budget is merely the budget that the supervisor intends as new usable budget for the client. The amount of "energy" in the transaction budget can only be used after the supervisor's intention has been committed in the client and the content has been transferred to the client's usable budget, where it becomes usable energy. In this implementation, energy cannot be transferred from one transaction supervisor budget (corresponding to a client) to another (corresponding to another client). The content of the supervisor's transaction budget needs to be deposited back in the usable part before it can be reallocated. The energy content of the supervisor is the sum of both (usable and transaction) budgets, while the energy in the client is only the usable budget.

B. Transaction protocol

The protocol state machine diagram of the supervisor is shown in Figure 3, while the protocol state diagram of a client (of which there can be many) is shown in Figure 4.

An intention is initiated by the supervisor, by transmitting an intended budget to the appropriate client. The client waits until it receives a new intention and, when it does, it sends an acknowledge. If the supervisor receives an acknowledge which does not correspond to the most recent transmitted intention, it defines a new intention and transmits that to the client. If the supervisor did receive the correct acknowledge, it can decide whether to commit or abort the transaction, which, in the nominal situation without timeouts and lost messages, is properly received and executed by the client. A confirmation of the action is sent by the client, after which the supervisor may execute a post-commit or post-abort process, and send an acknowledge to the client to indicate that the transaction is complete (although this latter step is not strictly necessary). Post-commit processing in the supervisor include updating the usable and unusable budgets. Post-abort processing includes moving the contents of the

transaction budget in the supervisor back to the usable budget for reallocation. This procedure uses the communication channel five times per transaction (or four times, if the last acknowledge is not implemented).

It is assumed that both the supervisor and the client have a mechanism to keep track of the time that has passed. When considering messages that arrive too late and messages that are lost, timeouts can be triggered. When a transaction intention has been transmitted, the supervisor can timeout when waiting for an acknowledge. It will then define a new intention and transmit it to the client. If the client's acknowledge has been received by the supervisor, and the supervisor decides to commit or abort but that command is never received by the client (and therefore the client encounters a timeout), the client commits the intention, sends a commit acknowledgement, and waits for the supervisor to send a new command. Proceeding with committing an intention after a timeout ensures that at least the budget that was initially intended is indeed allocated, which means that the client can use that energy to perform its task, thereby not limiting task execution further. If, on the other hand, the timeout results in reinitializing a transaction without the successful delivery of a new energy budget to be used by the client system, the performance of the complete system may suffer due to inappropriate budgeting at that time. The supervisor has to transmit a status request (number $n = 1$) to the client to verify what the client has done since the last communicated message. After a properly received status report (Intention committed), the supervisor can continue with post-commit processing and complete the transaction. If the status report is not received by the supervisor (or, similarly, the status request is not received by the client), the supervisor may again transmit a status request ($n = 2$). After N timeouts the supervisor has to assume, by a passivity timeout, that the client has committed the proposed transaction intention for passivity to be kept. The parameter N can be chosen according to latency specifications and the amount of confidence one requires in that a transaction has been completed in a valid way, instead of assuming that an intention had been committed and risking unnecessary dissipation (if it had not been committed). If no status requests were received by the client within the time associated to N supervisor timeouts, it will assume that the transaction is complete by Timeout_N and that the supervisor knows how to handle the situation unilaterally.

Fig. 3: Supervisor state machine of the energy transaction protocol.

Another scenario is that the acknowledgement of a new intention is sent from the client to the supervisor, but that acknowledgement is never received by the supervisor and a newly generated intention, due to a supervisor timeout, is in turn never received by the client. Then the client commits the intention due to a timeout, acknowledges its action, and waits for a supervisor command. If the supervisor has transmitted yet another new intention, the client understands that the timeout-commit should not have been done, resulting in a roll back and the treatment of a new intention.

A protocol may be resilient in case of supervisor or client failures and may still successfully accomplish a transaction [24]. Here, the concern is that passivity may be lost which can endanger the stability of the system. When a transaction is committed at the client the transaction budget is relocated to the usable budget. The supervisor's and client's usable budget then add up to the initial energy level in

Fig. 4: Client state machine of the energy transaction protocol.

the supervisor's usable budget that was present before the transaction. Therefore, passivity regarding the usable energy is never lost. One may incorrectly consider the traveling energy, i.e., the transaction budget of the supervisor or client, to be a source for the loss of passivity, because when an intention has been committed to the usable budget of the client the transaction intention energy is still present in the supervisor's transaction budget. However, one needs to consider a complete transaction after which this state will have been resolved. Moreover, the usable budgets should be implemented in non-volatile memory, while the transaction budgets should be implemented in volatile memory. A power loss will then cause the transaction budgets to reset and be empty, meaning that that energy cannot be reused for new transactions. This may cause additional dissipation, but always guarantees passivity.

After a transaction, the client can return the energy that was not used until a new usable budget was committed. Here it was chosen to not make it part of the protocol, because returning energy doesn't risk loss of passivity.

IV. SIMULATION CASE STUDY

A model of the energy transaction protocol has been made in MATLAB and Simulink, to perform simulations to verify the validity of the protocol and its effectiveness in keeping passivity while communicating energy budgets (Figure 5).

The state machines of Figure 3 and Figure 4 are implemented using Stateflow charts, and Simevents is used to model event-driven message generation (transaction intentions, command messages, etc.) and message transmission

Fig. 5: Model of the communication between a supervisor and a client, implementing the transaction protocol for passive communication of energy budgets. The supervisor and client submodels are Stateflow charts, while the communication mechanism between them is made with Simevents blocks.

over a communication channel. The communication channel is modeled to unreliably transmit messages with varying communication delays. For illustration purposes, the delay is set to $\Delta\tau = 1 + \text{rand}(-0.5, 0.5)$ seconds, where $\text{rand}()$ generates a random number between its arguments. This random delay is renewed for each new message that is transmitted. The timeouts in the state machines are set to 1 second, which means that a message arriving on time and a message arriving too late becomes equally likely. In the latter case, a timeout is triggered after which the state machine handles the timeout actions. The supervisor and client both sample the incoming message queue once a second. Message interrupts to allow processing of new incoming messages when they arrive has not been implemented, but can be incorporated straight-forwardly. The number of status requests and failed status responses has been set to $N = 1$.

A multitransaction simulation with the proposed protocol is done, in which a new transaction is started after the previous transaction has completed. The supervisor's decision to commit or abort a transaction has been randomized. The energy budgets are initialized such that the usable supervisor budget is at 10 Joule, while all other budgets are at 0 Joule. The result of the simulation can be seen in Figure 6, showing the communication procedure, and Figure 7, showing the individual energy budget levels. According to Figure 6, timeouts (grey lines) are frequently causing intentions (cross markers) to be redefined and retransmitted. If an intention is received on the client side (cross marker), an acknowledgement is sent (circular marker) and may or may not be received by the supervisor (circular marker). The supervisor (randomly) chooses to commit (green line) or abort (red line) the transaction, may do a status request (yellow line), and transmit an acknowledge (light blue line) to indicate the transaction is complete. Solid lines have been sent, while

dashed lines have been received. According to Figure 7, the total energy level of the usable budgets is always smaller than or equal to the initial energy in the system (10 Joule), i.e., the transactions ensure a passive system. The decrease of the energy is caused by the returning energy messages from the client not arriving at the supervisor due to the unreliable communication channel. At 20.5 seconds, however, the returned client energy is received by the supervisor and is stored in the usable energy budget.

When a more naive communication strategy for communicating energy budgets is followed using the same communication channel and identical sequential intentions and corresponding timeout moments, the energy budgets of the supervisor, client, and their sum will change. Because there is less overhead in this situation, the simulation time has been decreased to obtain the same number of communicated transaction intentions as with the transaction protocol in effect. When the supervisor is "naively optimistic", it can be seen in Figure 8 that the system remains passive, but that it has become much more dissipative. The supervisor assumes messages arrive and are acted upon, and because there is no two-way agreement whether energy has arrived, all intentions done by the supervisor are immediately and irreversibly subtracted from the supervisor budget. A similar situation holds for a "naively pessimistic" communicating supervisor, which considers energy intentions that are not acknowledged to have not arrived at the client, as shown in Figure 9. The supervisor's transaction budget is then added to the usable budget, while energy may still be returned by the client to the supervisor, which possibly generates energy (for instance at 8 seconds).

V. DISCUSSION

This paper proposes a protocol for communicating control and energy setpoints to ensure passivity of a system is kept. It

Fig. 6: Communication procedure of several consecutive transactions between the supervisor and client.

Fig. 7: Energy budgets of the supervisor and the client after several consecutive transactions.

can guarantee passivity even when using a highly unreliable communication channel, as was simulated. It has been shown that energy book-keeping becomes more dissipative or even generative if the protocol is not used and a more naive way of communicating energy budgets is employed instead.

The simulations that were done are quite extreme in the sense that the communication channel is highly unreliable. In real setups the connected components will likely share a more reliable network, thereby often allowing the supervisor and client to communicate in the nominal way as elaborated in Section III. The point is that it is important to keep track of energy budgets that are communicated throughout the system, which the protocol has shown to be able to do.

Using such protocols inevitably introduces a higher latency when compared to merely “sending and forgetting”, because of the overhead to confirm certain actions. In this case at least two round-trips over the communication channel are required to communicate intentions and to complete a transaction.

Fig. 8: Supervisor and client energy budgets after several consecutive transactions, for a “naively optimistic” energy communication strategy.

Fig. 9: Supervisor and client energy budgets after several consecutive transactions, for a “naively pessimistic” energy communication strategy.

For accurate verification of the workings of the protocol, this preliminary study should be advanced and tested on a setup with multiple clients, varying communication delays, and even communication loss and restoration.

VI. CONCLUSION

In this paper an energy transaction protocol has been presented that is able to communicate energy budgets in unreliable communication networks while keeping a system’s passivity. Inspiration has been found in financial transactions which are similar to energy budget transactions. A model of the protocol consisting of a supervisor, that initiates a transaction intention, a client, and a communication channel has been presented, and simulations have shown that the protocol is able to maintain passivity in the system. Without using the protocol, dissipation in the system increases and generation of energy may even occur. The protocol should be further investigated for accurate verification of its effectiveness.

REFERENCES

- [1] SoftPro, "SoftPro website: <http://softpro.eu>," Online, 2018.
- [2] S. Stramigioli, "Energy-aware robotics," in *Mathematical Control Theory I: Nonlinear and Hybrid Control Systems*, ser. Lecture Notes in Control and Information Sciences, M. K. Camlibel, A. A. Julius, R. Pasumathy, and J. M. Scherpen, Eds. Springer International Publishing, 2015, vol. 461, pp. 37–50.
- [3] J. C. Willems, "Dissipative dynamical systems part i: General theory," *Archive for rational mechanics and analysis*, vol. 45, no. 5, pp. 321–351, 1972.
- [4] J.-H. Ryu, C. Preusche, B. Hannaford, and G. Hirzinger, "Time domain passivity control with reference energy following," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 5, pp. 737–742, September 2005.
- [5] D. Lee and K. Huang, "Passive-set-position-modulation framework for interactive robotic systems," *Robotics, IEEE Transactions on*, vol. 26, no. 2, pp. 354–369, April 2010.
- [6] C. Schindlbeck and S. Haddadin, "Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 440–447.
- [7] G. A. Folkertsma, S. S. Groothuis, and S. Stramigioli, "Safety and guaranteed stability through embedded energy-aware actuators," in *IEEE International Conference on Robotics and Automation*, 2018.
- [8] S. Groothuis, G. Folkertsma, and S. Stramigioli, "A general approach to achieving stability and safe behavior in distributed robotic architectures," *Frontiers in Robotics and AI*, 2018.
- [9] A. Y. Mersha, X. Hou, R. Mahony, S. Stramigioli, P. Corke, and R. Carloni, "Intercontinental haptic teleoperation of a flying vehicle: a step towards real-time applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.
- [10] K. Langlois, T. van der Hoeven, D. R. Cianca, T. Verstraten, T. Bacek, B. Convens, C. Rodriguez-Guerrero, V. Grosu, D. Lefeber, and B. Vanderborght, "EtherCAT tutorial: An introduction for real-time hardware communication on windows," *IEEE Robotics & Automation Magazine*, vol. 25, no. 1, pp. 22–122, 2018.
- [11] EtherCAT Technology Group, "Ethercat - the ethernet fieldbus," Tech. Rep., 2018.
- [12] B. A. Nahas, S. Duquenooy, and O. Landsiedel, "Network-wide
- [22] C. i Fan, Y. kuang Liang, and B. wei Lin, "Fair transaction protocols based on electronic cash," in *Seventh International Conference on* consensus utilizing the capture effect in low-power wireless networks," in *Proceedings of SenSys '17*, 2017.
- [13] R. H. Thomas, "A majority consensus approach to concurrency control for multiple copy data bases," Bolt Beranek and Newman Inc., Tech. Rep., 1977.
- [14] M. Rafailescu and M. S. Petrescu, "Fault tolerant consensus protocol for distributed database transactions," in *Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences*, 2017.
- [15] D. Graff, D. Röhrig, and R. Karnapke, "Systemic support for transaction-based spatial-temporal programming of mobile robot swarms," in *40th Annual IEEE Conference on Local Computer Networks*, 2015.
- [16] B. R. Trilaksono, "Distributed consensus control of robot swarm with obstacle and collision avoidance," in *2nd International Conference on Information Technology, Computer, and Electrical Engineering*, 2015.
- [17] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. San Francisco, USA: Morgan Kaufmann Publishers Inc., 1992.
- [18] B. Medjahed, M. Ouzzani, and A. Elmagarmid, "Generalization of acid properties," *Cyber Center Publications*, 2009.
- [19] S. V. Scott and M. Zachariadis, *The Society for Worldwide Interbank Financial Telecommunication (SWIFT) - Cooperative governance for network innovation, standards, and community*. Routledge, 2014.
- [20] S. Lu and S. Smolka, "Model checking the secure electronic transaction (set) protocol," in *Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 1999.
- [21] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herreweghen, and M. Waidner, "Design, implementation, and deployment of the iKP secure electronic payment system," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 611–627, 2000.
- Parallel and Distributed Computing, Applications and Technologies*, 2006.
- [23] B. Cox, J. Tygar, and M. Sirbu, "Netbill security and transaction protocol," in *Proceedings of the First USENIX Workshop in Electronic Commerce*, 1995.
- [24] D. Skeen and M. Stonebraker, "A formal model of crash recovery in a distributed system," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 3, pp. 219–228, 1983.