Bounding the End-to-End Execution Time in Distributed Real-Time Systems: Arguing the case for Deterministic Networks in Lingua Franca

Henrik Austad[*] [†]
henrik.austad@sintef.no
Institute for Mathematics and Cybernetics, SINTEF Digital
Trondheim, Norway

Geir Mathisen
geir.mathisen@ntnu.no
Department of Engineering Cybernetics, Norwegian University of Science and Technology
Trondheim, Norway

ABSTRACT

Designing and implementing distributed systems with real-time requirements quickly reveal the complexity of handling time and logic across multiple systems. As data traverse a network, it is subjected to variable delay due to interfering traffic and variable load on network components. This introduces an element of non-determinism in execution time for distributed algorithms, which translates into increased error logic and pessimistic worst-case estimates. Over the next few years, it is expected that Cyber-Physical Systems will see many new use cases, and the network connecting these will play an ever more important role. Combined with the onset of the fourth industrial revolution, IEEEs Time Sensitive Networking, IETFs Deterministic Networking, and 3GPPs Ultra Reliable Low Latency profile will play a vital role in realizing these systems. Coordination languages such as Lingua Franca can offer a substantial contribution to the design process and implementation of distributed systems such as Cyber-Phyiscal Systems, both through its model of computation which elevates time to a first-class citizen and with its support for distributed models. In this paper, we show that by introducing deterministic network channels with a fixed delay, the worst-case execution time is not increased whereas the variance in total execution time from start to finish is greatly reduced. For a coordination language such as LF, this means that we can analyze a system using much tighter delay bounds for network traffic, which in turn can yield better resource utilization.

CCS CONCEPTS

• **Networks → Sensor networks**; *Packet-switching networks*; • **Computer systems organization** → *Real-time system architecture*; Real-time operating systems; *Embedded systems.*

KEYWORDS

Deterministic Networks, TSN, Tactile Internet, Cyber-Physical Systems, Lingua Franca, Real-Time Systems

[*]Corresponding author
[†]This work was funded by the Norwegian Research Council under grant 323340

ACM Reference Format:
Henrik Austad and Geir Mathisen. 2023. Bounding the End-to-End Execution Time in Distributed Real-Time Systems: Arguing the case for Deterministic Networks in Lingua Franca. In *Cyber-Physical Systems and Internet of Things Week 2023 (CPS-IoT Week Workshops '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3576914.3587499

1 INTRODUCTION AND MOTIVATION

A traditional definition of a real-time system is a system in which correctness is evaluated not only on its logical result but also on the time in which it is produced. A classical example is the control loop for a robotic system where sensors read the angle of one or more joints. These are then used as input in a model that estimates the state of the robot and compares this to the desired state. This difference is used as an error input to a control algorithm that computes one or more actuator commands designed to bring the robot closer to the desired position. The value of such a control signal to a highly dynamic system decreases quickly with time. If the new control signal arrives late it may no longer improve the system, and may instead be detrimental and make the control loop unstable. Since neither state estimation nor computing the control signal happens in zero time, a certain delay is unavoidable, and depending on the system, other tasks may also share the same hardware. One crucial task is therefore to ensure that the control signal is computed and made available to the actuator in time. The computation has a *deadline*.

Reducing the interference of unrelated tasks and keeping deadlines has been an active field since the seminal paper of Liu and Layland [15]. Real-time scheduling has seen a continuous improvement from fixed priority, single core to dynamic priorities running on multicore systems [9]. Although meeting the deadlines is important, it is also important to reduce the *variance* of the delay, meaning making the time taken to compute a value as consistent as possible. A constant delay makes the total system more predictable as sudden surges in computation demand are less likely to occur. One caveat of this is that the delay is *constant*, i.e. the *jitter* should be as low as possible.

When a real-time system consists of components located on different physical machines, we say that the system is a distributed real-time system (DRTS). Independent components can easily be assigned to different nodes in the system, but for components that have a direct or indirect dependency, this quickly becomes more challenging. For a distributed system, unrelated traffic thus manifests itself as an indirect dependency on other, unknown entities.

The execution profile of a task can only be found if all preceding tasks on which it depends are also found, and in a distributed system, the delay of the interconnecting fabric (the network) must also be factored in. For systems with poor quality of service (QoS), this necessarily leads to pessimistic run-time estimates, if they can be derived at all.

When multiple components of a DRTS send critical traffic concurrently, or when non-critical traffic enters the network, critical traffic may experience delay jitter even if the extra delay is within the delay bounds. A distributed real-time system must consist of *composable objects*, i.e. elements that behave independently of each other and which can be combined without affecting the validity of the remaining system [13]. By always using the upper bound for the end-to-end delay we effectively remove the non-determinism of said interference without increasing the worst-case delay. Granted, the *best-case* delay grows closer to the worst case, but as the *range* of delays decreases, the total execution time for a distributed algorithm becomes stable, and the *variance* decreases. This allows us to calculate tight bounds on total execution time, reduce timeout used to detect network outages and make a decentralized, distributed system more predictable.

The rest of this paper is organized as follows. Relevant background information is covered in Section 2. This also presents relevant deterministic networks and Lingua Franca in more detail. Related work is covered in Section 3. In Section 4 we present a roadmap for including `net_chan` in LF and describe the current status of this work. We close the paper in Section 5 with a brief discussion before concluding in Section 6.

2 BACKGROUND

Figure 1 shows a small, distributed control system for a robotic arm. The robot interface receives state updates periodically from the arm, forwards this to the controller, and receives an updated control signal which is then used to adjust the robotic arm.

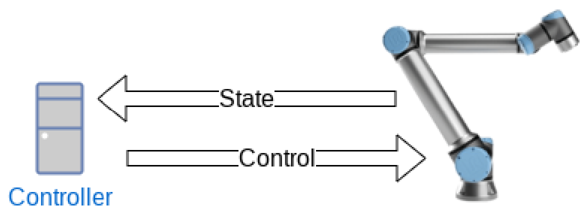


Figure 1: Distributing a robotic control algorithm
(credit: UR10e image, Universal Robots)

The nature of the connection between Robot IF and Controller is important once the nature of the traffic becomes apparent. Each state update must be processed before a new update arrives, and each control update must be sent to the robot in the correct order.

2.1 Time Sensitive Networks and the Tactile Internet

Time Sensitive Networking [1, 3] is a set of open IEEE standards that bring determinism and improved reliability to Commercial Off-the-Shelf (COTS) network hardware. A set of data packets that logically belong together is denoted *a stream* Critical streams are identified using a unique stream identifier and are protected by reserving both buffer capacity and transmission priority along the path from the sender ("Talker") to one or more receivers ("Listeners"). If all network switches (Bridges in TSN nomenclature) can meet the stream requirements, the reservation succeeds and the talker can begin transmitting data, otherwise the reservation fails and the network is unable to provide bounded delays. The original traffic classes, A and B, with the Credit Based Shaper (CBS) [2] inherited from Audio-Video Bridging have since been extended with scheduled traffic using the Time-Aware Shaper (TAS) [3] and Asynchronous Traffic Shaper (ATS) [4]. Class A traffic can be guaranteed to arrive within 2ms in a 100 Mbps network with a maximum of 7 hops. In a 1 Gbps network, TAS can give a 100 μs latency guarantee over a 5-hop path but requires substantial configuration and engineering compared to CBS.

Alongside this, ITU-T has defined Tactile Internet (TI) as an internet network that combines low latency with extremely high availability and security[12]. With a specified upper end-to-end latency of 1 ms and a 99.999% availability[12], there are currently no commercially available wireless technologies capable of meeting this demand. 3GPPs Ultra-Reliable Low-Latency (URLLC) profile from Release 17 (3GPP Rel17) can be seen as the best approach for this, especially when combined with TSN as a backhaul network [17]. When taken together, TSN and URLLC provide a deterministic network for both wired and wireless connections. It is expected that future Cyber-Physical Systems (CPS) with time-constrained behavior will rely on TI to realize their potential.

Note: IETF has specified Deterministic Networking (DetNet) [10] which targets larger networks than TSN. Where TSN is limited to subnets (i.e. not routable), DetNet targets WAN. It is important to note that DetNet does not standardize a single networking technology, but rather provides recommendations and requirements for underlying networks on top of which a DetNet can be served. DetNet can run over TSN, Multiprotocol Label Switching (MPLS) Packet Switched network and URLLC.

2.2 Lingua Franca

Lingua Franca (LF) is a coordination language that adds time and reactive concurrency to multiple target languages [16]. Execution of code is performed by actors that react to external events, which are named reactors in LF. To ensure the deterministic execution of multiple reactors, LF uses the concept of superdense time and events between reactors are ordered by "tags" which are advanced either globally (centralized coordination), or when each reactor can be sure that no other earlier event can arrive and thus advance the time in a decentralized fashion. Internal actions are ordered by tag and micro-steps such that another iteration with the same input will yield the same execution order. This comes as a contrast to a traditional multithreaded application where the interleaving of threads makes the actual execution order unpredictable [14].

From the specification files, which are a mix of LF notation and target language code, LF will generate a runtime with the target application embedded. This can be either a single system or a distributed application where an LF federation consists of a set of federate reactors, each capable of running on separate hosts. In the latter case, time can be coordinated using either a centralized approach where a runtime infrastructure (RTI) server handles the advancement of time or a decentralized approach. Reactors in the latter case adhere to a concept called "Safe to Process" (STP) where reactors are connected directly (and not via RTI) and will advance logical time on their own after a safe delay has been observed. To handle delays beyond what is expected, reactors can specify STP violation actions where excessive network delays will be handled should they occur.

3 RELATED WORK

Timed C [18] is an extension to the C programming language and a framework somewhat similar to Lingua Franca that elevates time as a first-order citizen in the system. In Timed C, it is possible to express directly timeouts and delays for how long subsets of an algorithm should take, and by expressing timing points as either soft, firm or hard, different actions can be taken should the system miss its target.

In [11], Gutiérrez et. al showed that TSN is capable of forwarding critical control signals to control a robotic arm. Recently, the Institute of Electrical and Electronics Engineers (IEEE) and International Electrotechnical Commission (IEC) have started on a joint standard for TSN in Industrial Automation [19].

The Robot Operating System 2 (ROS2) has chosen Distributed Data Service (DDS) as the underlying protocol, and Agarwal et. al showed how determinism and jitter are improved using TSN to safeguard the traffic [5].

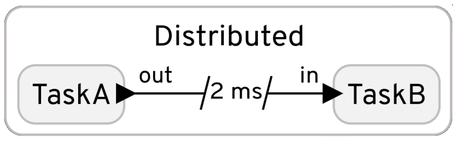
In a project known as "Federated Timed C" [8], Timed C was extended with net_chan [6] to design and implement a distributed real-time system. In the end, two separate tasks were synchronized to 18μs, which was the accuracy of the kernel scheduler running on the systems.

4 EXTENDING DECENTRALIZED LF WITH TSN AND NET_CHAN

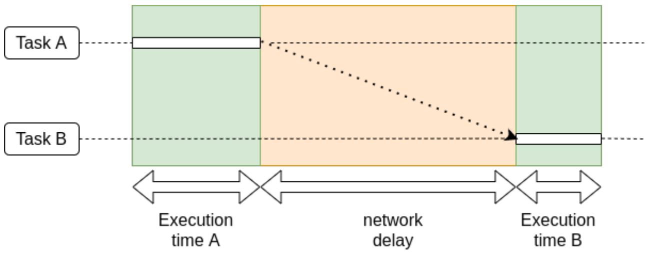
In a decentralized approach, LF will use STP to determine if it is safe to advance time. Due to the nature of standard packet-switched networks, the STP offset value that sets the expected worst-case delay must be set somewhat arbitrarily. To derive tighter bounds, explicit QoS measures must be in place and the network stack configured carefully.

In Fig. 2, we see a distributed system of two tasks, A and B connected by a channel. For this example, the delay can be selected arbitrarily but we chose the 2 ms delay guaranteed by TSN's Class A. Real-time systems need to verify the total execution times of its component, and as complex as this task already is, adding unknown and unpredictable network delays further complicates this matter. Fig. 2b) shows this, and the main focus of our work here is to make the network delay a known constant.

In addition, as shown in [8], when combining Timed C and net_chan, setting the delay to a fixed value, the network channel becomes composable and independent nodes in the system will not be affected by interfering traffic. With bounded delays on each link, it also becomes possible to extend this to larger federates, the total delay can then be summarized as the individual computational delay and the upper limit of each network link.



(a) A Lingua Franca model showing a logical channel with a defined delay to connect to reactors. This can map to a TSN Class A stream.



(b) Total execution time a distributed system consists of local computation and network delay.

Figure 2: Trivial distributed system of two hosts connected by a TSN Class A link.

The benefits of using fixed delay network channels in combination with LF are thus:

- Reduced deadline/timeout STP values as the network provides a known bounded and low latency.
- Guaranteed delivery allows for simplifying error handling logic. If a packet is delayed beyond the stated latency, it is safe to assume that the network has become partitioned.
- Nodes become composable as any jitter introduced (within the delay bounds) is effectively hidden.
- LF can generate TSN stream attributes such as required bandwidth, period and network path directly from the LF model.
- LF can reason about total execution time and schedule tasks accordingly, even in a distributed scenario.
- Critical traffic can coexist with normal, best-effort traffic on the same network, with no need to move critical and non-critical traffic into separate networks, making LF a good candidate for future CPS systems on converged networks.

4.1 Adding net_chan to Lingua Franca

Setting up and configuring TSN channels require a lot of system knowledge. In net_chan, most of these steps are already taken and a developer can specify a channel using a "manifest" where all channels are described. Currently, this file must be created manually as the approach taken in [8] cannot yield this information directly from the model.

Listing 1: net_chan manifest, all channels must be described in this format for net_chan, typically within a single file shared by all nodes.

```c
struct net_fifo net_fifo_chans[] = {{
  .dst       = {0x01, 0x00, 0x5E, 0x01, 0x11, 0x03},
  .stream_id = 3,
  .sc        = CLASS_A,
  .size      = sizeof(struct sensor_data),
  .freq      = 10,
  .name      = "sensor_data"}};
```

4.2 Challenges

Currently, this approach will only work on a decentralized LF federation. In the centralized case, the dependence on the RTI server would negate the benefit of fast and bounded latency through a net_chan channel.

To properly use TSN, only a few capable Network Integrated Cards (NIC) exist. A popular choice is the Intel I210 which comes in a PCIe form factor. Other NICs exist, both Renesas and NXP have TSN-capable systems. Currently, net_chan supports TSN on Linux and I210 and this limits the usage to a single OS and NIC. Furthermore, as LF is a large and complex framework, adding seamless integration for TSN is a daunting task and should be done carefully. Even though net_chan removes some of the complexities of managing TSN streams, it still adds additional software dependencies to LF, which could frustrate users.

4.3 Roadmap and Current Status

The current outline for testing LF and net_chan is as follows:

(1) The first step is planned as simply linking with net_chan from within an LF module. This will provide a network construct inside the reactor, but as far as LF is concerned, the reactor is only writing the value to a local variable. The results yielded from this step are comparable with what was derived in [8], and this approach can therefore be considered a "proven approach".
(2) The next step is thus to extend LF in a way such that a new channel is available, similarly to logical channels as the delay is known and bounded. Even though the network approach introduces non-determinism, using a deterministic channel in combination with a fixed delay such as READ_WAIT() from net_chan, using logical channels with fixed delays, is the most promising approach.
(3) Building on this, where net_chan requires a manifest to describe each network channel, once we have this available from within LF, extracting channel attributes directly from the model will give LF a tremendous expressive power.
(4) Finally, by using the fixed delay on each channel, we hope to use this to compute the total execution time for composed, federated reactors and provide this as additional input to the task scheduler in LF. As previously stated, estimating WCET is a tall order, but by introducing upper bounds on network delays, this at least brings distributed WCET closer to "single host" WCET estimation. In [20], Zhao et. al derived tighter bounds for total End-to-End latency in a TSN network using Network Calculus, and it seems promising to extend the results to LF and derive end-to-end (E2E) latency bounds. This remains an interesting venue to explore at a later stage.

4.4 Implementing the First Step - Direct Linking

Currently, LF support distributed computing using *federates*, and when configured to run in a decentralized setup, the RTI server is only used for initial setup and shutdown. It is natural to compare the current behavior of a federated LF model with a corresponding model using net_chan.

4.4.1 Initial Integration and Evaluation. We use the same approach as in [8] and include net_chan as an external library. To achieve a level of composability, we delay the traffic by 2 ms since TSN provides an upper bound of 2 ms of all traffic. The code framework used to demonstrate this approach and the following tests can be obtained from a public GitHub repository [7], net_chan must be installed locally. As the goal is to evaluate how deterministic channels behave conjoined with LF, both the sensor reactor and the accompanying logic are intentionally kept simple. In the aforementioned repository, a Lingua Franca federated system is available under distributed/. This is shown in Fig. 3.

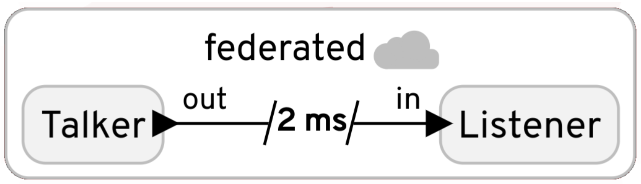


Figure 3: Distributed system used to compare federated LF with a TSN capable application

Similarly, talker/ and listener/ contains code for separate reactors that together implement the same behavior as the federated application. In other words, in this scenario, the two reactors in Fig. 3 each implement a single, standalone application that can run on separate hosts and is connected using a TSN stream via net_chan.

4.4.2 Small Scale Testing. For TSN and net_chan to behave as expected, the end stations and network must all support TSN. We rely on net_chan's integration with AvNUs mrpd for reservation messages. linuxptp provides ptp4l for synchronizing the Network clock and slaving the system clock to the network using phc2sys. Our end station uses the Intel I210 NIC alongside Linux's network subsystem that has supported CBS and TAS for some time and a pair of Cisco Catalyst switches is used to host the network.

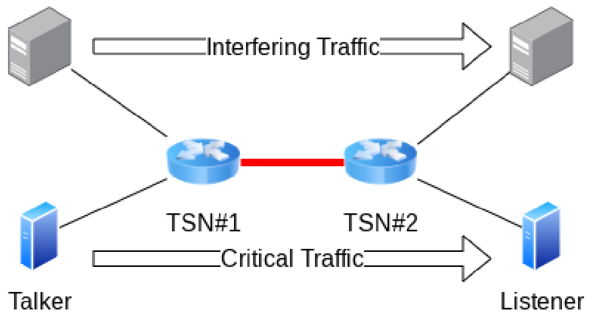


Figure 4: Test system setup where a common link in the network transports both critical and (from the real-time application's perspective) irrelevant traffic.

We capture timestamps[1] when a frame is sent and when it is received. To reduce unrelated timing errors, both test machines run

[1]we use lf_time_physical() which on Linux maps directly to CLOCK_REALTIME which is kept accurate with phc2sys and ptp4l

		Listener		Talker	
		PTP	*clock*	*PTP*	*clock*
F-LF	Max	17 ns	717 ns	29 ns	796 ns
	Avg.	11.1 ns	118.1 ns	16.6 ns	119.9 ns
	σ	2.0 ns	165.7 ns	3.9 ns	156.3 ns
NC-LF	Max	18 ns	561 ns	29 ns	512 ns
	Avg.	11.2 ns	43.9 ns	16.2 ns	40.6 ns
	σ	1.9 ns	82.3 ns	3.7 ns	51.8 ns

Table 1: PTP and CLOCK_REALTIME (PTP+phc2sys) accuracy during test runs. Federated Lingua Franca (F-LF) runs decentralized with RTI providing initial startup/shutdown synchronization. A `net_chan`-enabled LF model TSN streams for 2 similar, yet separate reactors.

	Federated LF		`net_chan` LF	
	Idle	Noise	Idle	Noise
Min	1 696.46 μs	1 703.31 μs	2 011.55 μs	2 017.61 μs
Max	2 044.30 μs	207 551.63 μs	2 037.43 μs	2 043.21 μs
Avg	1 812.78 μs	2 106.05 μs	2 026.18 μs	2 026.12 μs
σ	72.70 μs	5 973.12 μs	0.68 μs	0.74 μs

Table 2: E2E latency for federated and `net_chan` model. Noise is when a noise-generating application is filling all available best-effort capacities of the shared link between the two TSN switches.

a real-time Linux kernel (5.16.2-rt19), the applications are shielded from other tasks using cpusets, interrupts are moved away, memory is allocated at initialization and locked to avoid page faults, and the tasks run with a real-time scheduler (SCHED_RR) and priority 70.

We then run both the Federated LF model and the `net_chan`-alternative on an idle network as well as a highly saturated network where two unrelated systems saturate the single, shared link between the two switches as shown in Fig. 4. As these are early results, the tests are run for a short time, logging timestamps at 10 Hz for 300 seconds.

4.4.3 Results. The first step is to synchronize and evaluate the accuracy of the clocks so that we can evaluate the accuracy of the calculated E2E latency. Both `ptp4l` and `phc2sys` log the adjustment each second which can use to keep track of the clock accuracy. This is summarized in Table 1 The worst error seen for both tests is approximately 0.8μs which most likely stems from PCIe jitter, the network clock is synchronized to within 29ns.

The results of the tests are shown in Table 2. In the "Federated LF" (F-LF) case, the system shown in Fig. 3 is evaluated. The numbers under "Noise" is captured when the Noise Generator is active. In the idle case, F-LF performs well, it is apparent that the "after 2 ms" clause is an upper bound. On the other hand, due to the lack of any QoS mechanisms in play when the network is saturated in the F-LF case. The worst-case delay is severely affected, and both the maximum and the standard deviation are increased by 2 orders of magnitude. Impressively, even with the applied network noise, the delay added due to retransmissions of lost packets is not increased much beyond 200ms. A plot of F-LF's E2E delay (idle) can be viewed in the topmost plot in Fig. 5, the other F-LF plot was omitted due to the extreme outliers.

The tests using `net_chan` can be seen in the last two columns in Table 2. As can be seen, even during high network load, the delay hardly changes, but impressive as this may seem, this is also partly due to usage of READ_WAIT() in `net_chan`, as the network is loaded, the e2e invariably increases, just not beyond the guaranteed 2 ms. The lower plot in Fig. 5 shows the delay using `net_chan` under heavy network load. The single values are overlaid by a simple moving average of length 100.

5 DISCUSSION AND FUTURE WORK

From the results, protecting the traffic using QoS is self-evident, and TSN remains an interesting candidate. The amount of code needed to include `net_chan` in LF is not excessive, which makes `net_chan` an interesting library to experiment with. Although 3 lines of code *could* be enough, tighter integration with LF would simplify this even further. As discussed in Sec. 4.3, with the structure of LF models, deriving stream properties directly from a model is a tantalizing possibility.

One important takeaway when comparing a federated LF model to two reactors connected using `net_chan`, during our tests, the former ran completely without QoS and comparing the federated model directly with TSN is unfair. What we aimed to demonstrate was the usefulness of a TSN channel as the behavior is almost identical regardless of network load. The results in Table 2 and the lower plot in Fig. 5 clearly demonstrate this.

In the future, we plan to use Lingua Franca and `net_chan` to compose a distributed control system for a robotic arm, in essence, a Sensor, a controller and an actuator (looking back to Fig. 1). For each iteration, the error from the desired state is found and a new set of joint motor speeds are derived to bring the arm and its end-effector to the correct position. From the results obtained in this paper, we see that we can expect periodic data to arrive with low variance regardless of interfering traffic.

6 CONCLUSION

In this paper, we have seen how coordination languages such as Lingua Franca can benefit from having a clear interface to deterministic networking. By using TSN channels with a low, yet fixed delay, we can improve the accuracy of execution bounds for distributed tasks even in the face of severe network interference. We can also use the same guarantees to substantially reduce the timeout value used to detect network outages. This allows LF to operate with lower margins for network delay and can thus achieve higher resource utilization without sacrificing safety. Finally, by using an external library such as `net_chan` to provide such channels, LF can allow the library to address new networks (such as URLLC), new NICs and OS interfaces and instead focus on a fairly small, stable API.

REFERENCES

[1] 2010. IEEE Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)* (2010). https://doi.org/10.1109/IEEESTD.2010.8684664
[2] 2010. IEEE Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)* (2010). https://doi.org/10.1109/IEEESTD.2010.5594972

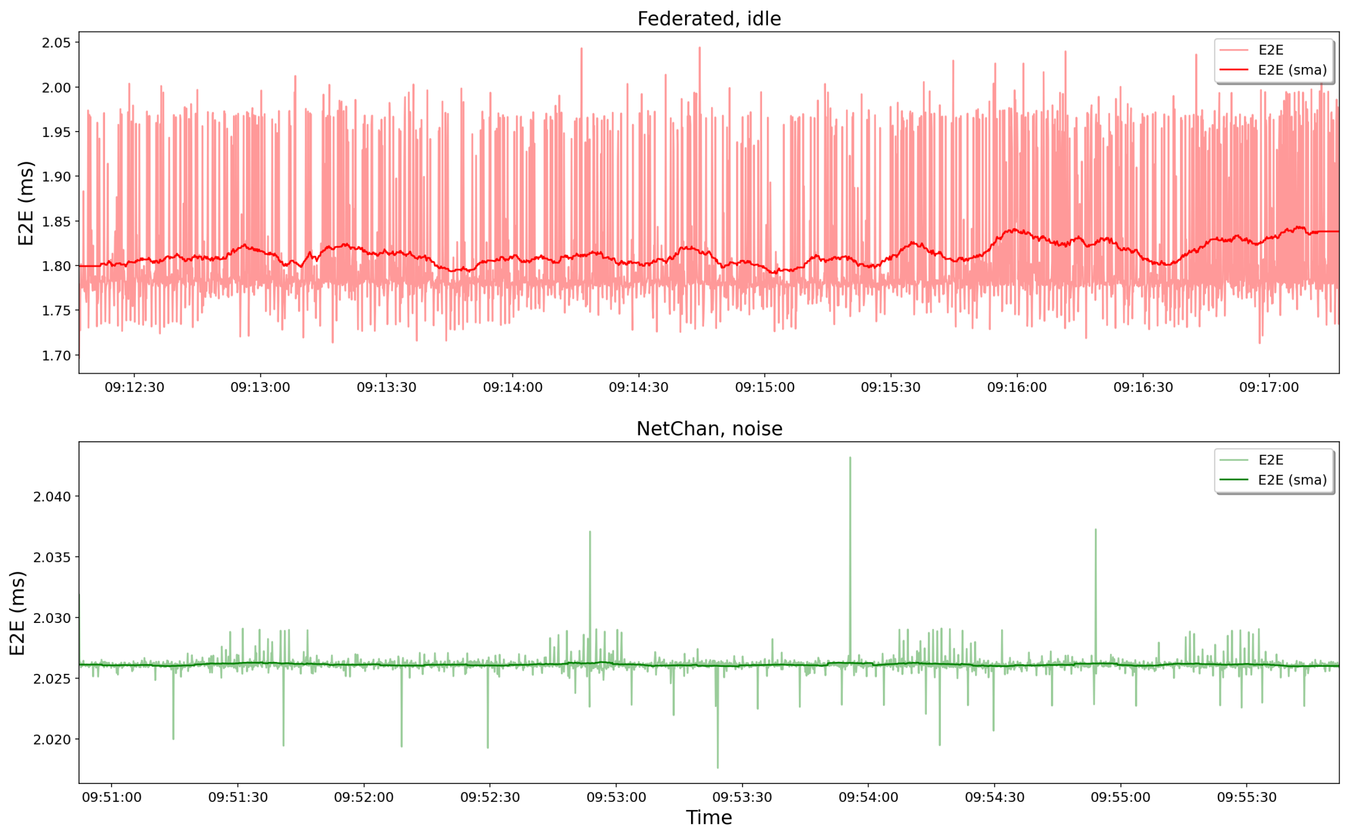


Figure 5: E2E comparison of an idle, federated system and LF using net_chan under severe network load. Same dataset as used to generate Table 2

[3] 2016. IEEE Enhancements for Scheduled Traffic. *IEEE Std 802.1Qbv-2015* (2016). https://doi.org/10.1109/IEEESTD.2016.8613095

[4] 2020. IEEE Asynchronous Traffic Shaping. *IEEE Std 802.1Qcr-2020* (2020). https://doi.org/10.1109/IEEESTD.2020.9253013

[5] Tanushree Agarwal, Payam Niknejad, M. R. Barzegaran, and Luigi Vanfretti. 2019. Multi-Level Time-Sensitive Networking (TSN) Using the Data Distribution Services (DDS) for Synchronized Three-Phase Measurement Data Transfer. *IEEE Access* 7 (2019), 131407–131417. https://doi.org/10.1109/ACCESS.2019.2939497

[6] Henrik Austad. 2023. *henrikau/net_chan: v0.1.2*. https://doi.org/10.5281/zenodo.7635611

[7] Henrik Austad. 2023. *Lingua Franca net_chan integration*. https://github.com/henrikau/tcrs_lf

[8] Henrik Austad, Erling Rennemo Jellum, Sverre Hendseth, Geir Mathisen, Torleiv Håland Bryne, Kristoffer Nyborg Gregertsen, Sigurd Mørkved Albrektsen, and Bjarne Emil Helvik. 2023. Composable distributed real-time systems with deterministic network channels. *Journal of Systems Architecture* 137 (2023), 102853. https://doi.org/10.1016/j.sysarc.2023.102853

[9] Dario Faggioli, Fabio Checconi, Michael Trimarchi, and Claudio Scordino. 2009. An EDF scheduling class for the Linux kernel. In *Proceedings of the 11th Real-Time Linux Workshop*.

[10] Norman Finn, Pascal Thubert, Balazs Varga, and János Farkas. 2019. Deterministic Networking Architecture. RFC 8655. https://doi.org/10.17487/RFC8655

[11] Carlos San Vicente Gutiérrez, Lander Usategui San Juan, Irati Zamalloa Ugarte, and Victor Mayoral Vilches. 2018. Time-Sensitive Networking for robotics. *CoRR* abs/1804.07643 (2018). https://dblp.org/rec/journals/corr/abs-1804-07643.bib

[12] ITU-T. 2014. The Tactile Internet. https://www.itu.int/dms_pub/itu-t/oth/23/01/T23010000230001PDFE.pdf

[13] Hermann Kopetz. 2011. *Real-Time Systems - Design Principles for Distributed Embedded Applications*. Springer. https://doi.org/10.1007/978-1-4419-8237-7

[14] Edward A Lee. 2006. The problem with threads. *Computer* 39, 5 (2006).

[15] C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* 20, 1 (1973), 46–61. https://doi.org/10.1145/321738.321743

[16] Marten Lohstroh, Christian Menard, Soroush Bateni, and Edward A. Lee. 2021. Toward a Lingua Franca for Deterministic Concurrent Systems. *ACM Trans. Embed. Comput. Syst.* 20, 4 (2021). https://doi.org/10.1145/3448128

[17] Ahmed Nasrallah, Akhilesh S. Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. 2019. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Communications Surveys & Tutorials* 21, 1 (2019), 88–145. https://doi.org/10.1109/COMST.2018.2869350

[18] Saranya Natarajan and David Broman. 2018. Timed C: An Extension to the C Programming Language for Real-Time Systems. In *24TH IEEE REAL-TIME AND EMBEDDED TECHNOLOGY AND APPLICATIONS SYMPOSIUM (RTAS 2018)*. IEEE, 227–239. https://doi.org/10.1109/RTAS.2018.00031 [ed] Pellizzoni, R.

[19] Ludwig Winkel. 2023. IEC/IEEE 60802 TSN Profile for Industrial Automation.

[20] Luxi Zhao, Paul Pop, Zhong Zheng, Hugo Daigmorte, and Marc Boyer. 2020. Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus. *IEEE Transactions on Industrial Electronics* 68, 10 (2020), 10291–10302.