

**UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD**

AMAZON.COM, INC. and AMAZON WEB SERVICES, INC.,
Petitioner,

v.

SWARM TECHNOLOGY, LLC,
Patent Owner.

Case: IPR2023-00390
U.S. Patent No. 9,146,777

**PATENT OWNER'S PRELIMINARY RESPONSE
PURSUANT TO 35 U.S.C. § 313**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. THE PETITION LACKS A STATUTORY BASIS TO PROCEED	2
A. Petitioner was served with a Complaint more than one year before the Petition was filed.	2
III. THE '777 PATENT	3
A. Overview of the '777 Patent.....	3
B. Prosecution History	7
C. Claim Construction.....	8
IV. THE PETITION'S GROUNDS AND REFERENCES.....	8
A. Overview of Lurie	9
1. Direct Distribution Mode	9
2. Automatic Task Distribution Mode	11
3. Batch Mode	14
B. Overview of Ethernet Specification	15
C. Overview of Andrews.....	17
D. Overview of Lippett.....	17
V. PETITIONER FAILS TO SHOW A REASONABLE LIKELIHOOD THAT IT WOULD PREVAIL ON ANY CHALLENGED CLAIM	18
A. The Petition fails to show that the cited references teach a “first agent retrieves the <i>matching task</i> by ... <i>searching the task pool</i> ”	18

B.	Claims 1-14: Petitioner fails to show that the cited art teaches a “first agent configured to <i>proactively retrieve</i> , from the task pool . . . a matching task”	27
C.	Claims 1-14: Petitioner fails to show that the cited art teaches a first agent having a “ <i>payload compris[ing] a list of functions</i> ”	33
VI.	CONCLUSION	35

TABLE OF AUTHORITIES

	<u>Page(s)</u>
<u>Statutes</u>	
35 U.S.C. § 103	7
35 U.S.C. § 313	1
35 U.S.C. § 314	2, 17, 42
<u>Regulations</u>	
37 C.F.R. § 42.104	17
37 C.F.R. § 42.107	1

LIST OF EXHIBITS

<u>Exhibit</u>	<u>Description</u>
2001	Declaration of Dr. Brent E. Nelson in Support of Patent Owner's Preliminary Response
2002	Curriculum Vitae of Dr. Brent Nelson
2003	IPR2022-00283 Preliminary Guidance
2004	IPR2022-00633 Preliminary Guidance
2005	Complaint for Patent Infringement
2006	Motion for Leave to File First Amended Complaint
2007	Exhibit 2 to First Amended Complaint

Pursuant to 35 U.S.C. § 313 and 37 C.F.R. § 42.107, Patent Owner Swarm Technology, LLC (“Patent Owner”) hereby files its Preliminary Response to the Petition filed by Amazon.com., Inc. and Amazon Web Services, Inc., (collectively “Petitioner”) in IPR2023-00390 requesting *inter partes* review of claims 1-14 of U.S. Patent No. 9,146,777 (“the ’777 Patent”). In accordance with 37 C.F.R. § 42.107(b), this Preliminary Response is timely filed within three months of the mailing date (January 10, 2023) of the Notice of Filing Date Accorded to Petition and Time for Filing Patent Owner Preliminary Response. Paper 3.

I. INTRODUCTION

The Board should deny institution for at least the following reasons: 1) instituting *inter partes* review in this case is barred by law under 35 U.S.C. § 315(b) because the Petition was filed more than one year after the date on which Petitioner was served with a complaint alleging infringement of the ’777 Patent; and, 2) the Petition fails to show a reasonable likelihood that it will prevail with respect to at least one of the challenged claims. Petitioner fails to show that any of its cited references, alone or in combination, disclose or suggest numerous limitations required by the claims of the ’777 Patent.

Specifically, Petitioner fails to show that the cited references, alone or in combination, disclose or suggest:

- i. a “first agent retrieves [a] matching task by: . . . *searching the task pool* for a task” as called for in claims 1 - 14.
- ii. a “first solidarity cell comprising a first agent configured to *proactively retrieve*, from the task pool . . . *a matching task*” as called for in claims 1 - 14.
- iii. a first agent having a payload comprising a list of functions the agent’s solidarity cell is configured to perform as part of retrieving a task as called for in claims 1 – 14.

All of the above quoted claim limitations are recited in each and every claim of the ’777 Patent. Thus, it is only necessary for Patent Owner to prevail on any one of these elements to properly preclude institution.

Based on these three claim limitations alone, the Petition fails to satisfy the statutory burden under 35 U.S.C. § 314(a) as it fails to establish a reasonable likelihood that Petitioner will prevail with respect to at least one of the claims challenged in the Petition.

The Board should therefore deny institution.

II. THE PETITION LACKS A STATUTORY BASIS TO PROCEED

A. Petitioner was served with a Complaint more than one year before the Petition was filed.

On 15 March 2021, Patent Owner filed a Complaint against Petitioner in the U.S. District Court for the District of Arizona related to U.S. Patent Nos. 9,146,777,

9,852,004 (“the ’004 Patent”) and 10,952,275 (“the ’275 Patent”). Exhibit 2005 (“March 2021 Complaint”) at ¶¶76-80, 92-97. The March 2021 Complaint asserted specific counts for infringement of the ’004 Patent and ’275 Patent. March 2021 Complaint at ¶¶137-152. The March 2021 Complaint was dismissed, *without prejudice*, on 20 September 2021. On 20 October 2021, Patent Owner filed and served a Motion for Leave to File a First Amended Complaint (Exhibit 2006 (“Motion for Leave”)), along with a required exhibit containing the First Amended Complaint (Exhibit 2007 (“First Amended Complaint”)) which asserted specific counts for infringement of the ’777 Patent, the ’004 Patent, and the ’275 Patent against Petitioner. First Amended Complaint at ¶¶299-306, 283-290, 291-298.

Significantly, the First Amended Complaint, which included specific allegations of infringement of the ’777 Patent, was filed more than one year before the filing date of the instant Petition, namely, 21 December 2022.

Under 35 U.S.C. § 315(b), the Petition is time-barred and the Board should therefore deny institution.

III. THE ’777 PATENT

A. Overview of the ’777 Patent

Alfonso Íñiguez is the sole inventor of U.S. Patent No. 9,146,777 entitled “Parallel Processing with Solidarity Cells by Proactively Retrieving from a Task Pool a Matching Task for the Solidarity Cell to Process.” Prior to Mr. Íñiguez’

invention, conventional multiprocessor systems included a central controller or CPU, and one or more co-processors (responders). Under the conventional “controller/responder” paradigm, the controller partitions the system’s computational requirements into tasks (including threads of tasks) and distributes them to the co-processors. Completed tasks are reported to the controller, which continues to distribute additional task to the co-processors as needed. *See* ’777 Patent, 1:24-30.

Such parallel processing architectures can be disadvantageous in that “a significant amount of the CPU's processing time is consumed managing the co-processing tasks. The management may include: distributing the tasks, in sequential order when needed, to co-processors according to their capabilities; waiting for tasks to be completed before distributing result-dependent threads; responding to interrupts from co-processors every time a task is completed; and responding to other messages from co-processors. Second, a co-processor will remain idle as it waits for a thread to be assigned to it by the CPU.” *Id.* at 1:33-42.

The ’777 Patent describes novel and nonobvious parallel processing computing architectures that reduce CPU management overhead while more effectively exploiting co-processing resources. This architecture includes, among other things, a controller, one or more co-processors, and a task pool interposed between the controller and the co-processors. *See Id.* at Abstract, FIG. 1, 2:47-58.

The controller is configured to populate a task pool with tasks, for example, a set of first tasks and a set of second tasks. *Id.* at 3:62-3:64. The first tasks can be performed by certain co-processors that are capable of performing first task types. The second tasks can be performed by certain co-processors that are capable of performing second task types. *Id.* at 3:64-3:66. The one or more co-processors are configured to autonomously retrieve and complete a matching task from the task pool. Each co-processor is proactive in that it may interact with the task pool without being instructed to do so by the controller or the task pool. *Id.* at 5:19-25.

Fig. 1 of the '777 Patent (reproduced below) illustrates the operation of an exemplary embodiment of the parallel computing architecture described by the '777 Patent.

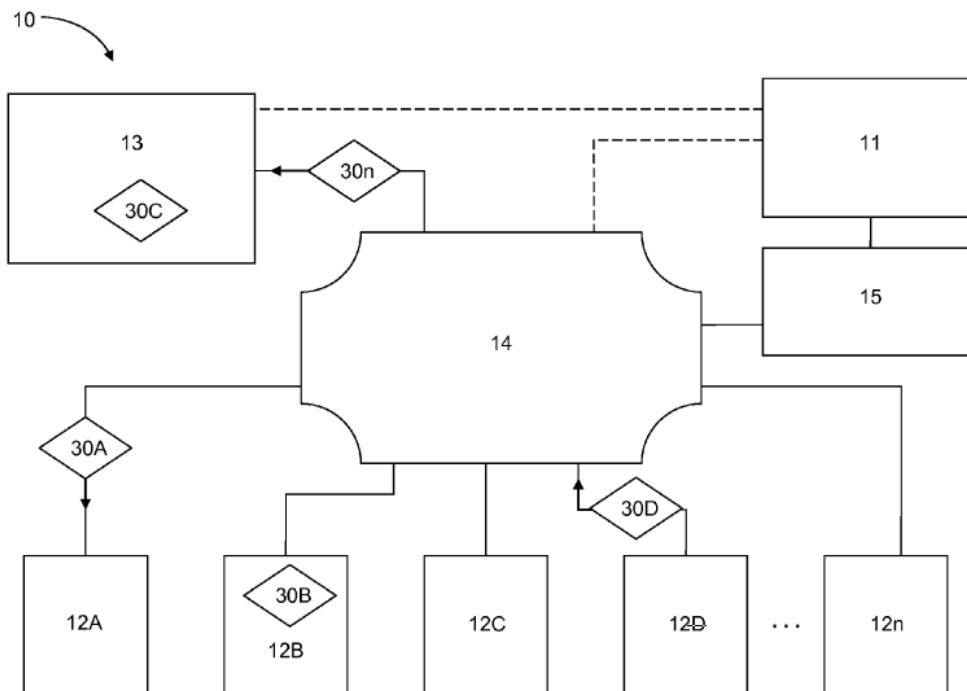


FIG. 1

As shown in Fig. 1, a central controller or CPU 11 populates the task pool 13 with particular sets of tasks or “threads.” *Id.* at Abstract, 2:1-4. The co-processors (12A to 12n) proactively dispatch their respective agents (30A to 30D) to the task pool 13 to identify tasks that match the capabilities of the co-processors. Each co-processor retrieves and processes a matching task without communicating with the CPU 11. *Id.* at 2:52-2:58, 7:27-7:29. The task pool and controller (via the task pool) are notified when the task is complete. *Id.* at Abstract, 5:11-5:13 and 6:39-6:42. Additional co-processor(s) can be dynamically incorporated into the system as needed. *Id.* at 7:06-7:09, 7:25-7:29.

Fig. 2 of the '777 Patent (reproduced below) is an exemplary embodiment of a task pool 13 containing threads 21, for example, thread 21A, comprising four tasks.

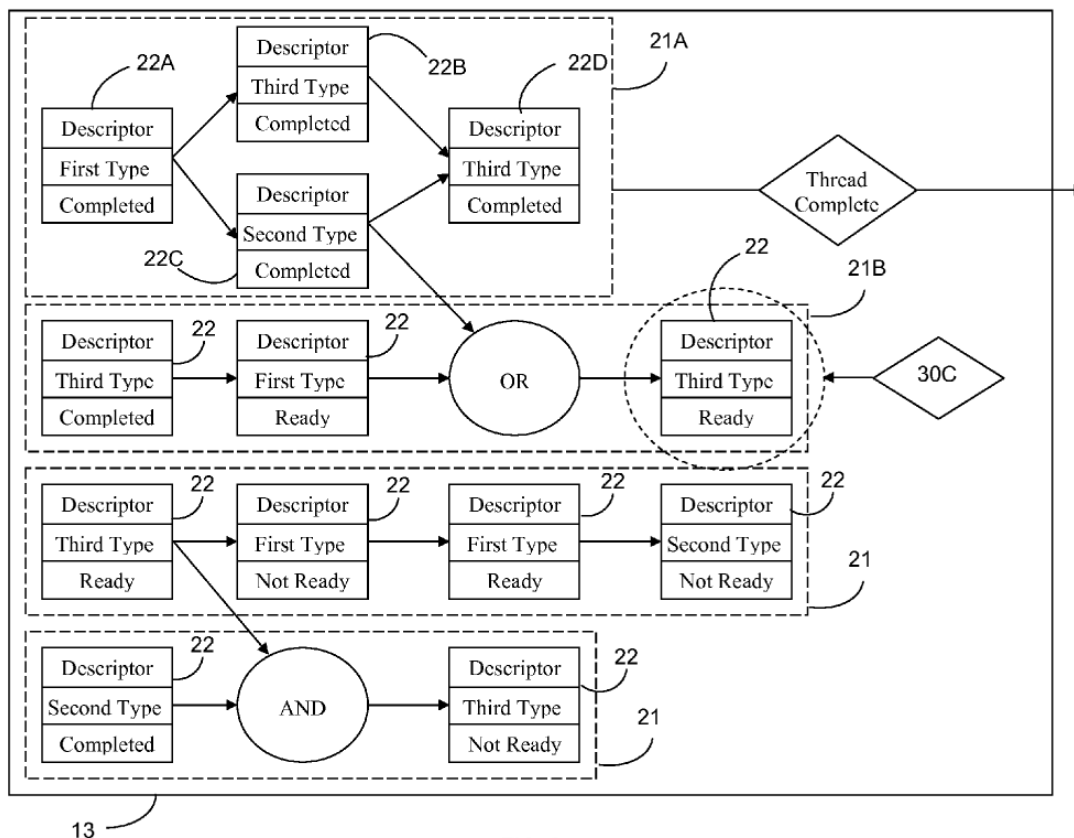


FIG. 2

The task pool 13 “preferably occupies a region of physical memory that is addressable by the CPU 11” and further “is also addressable by the cells 12A . . . n.” *Id.* at 4:12-15. The CPU 11 “may initialize and then populate the task pool 13 by dividing the large requirement into concurrently executable threads 21 and places the threads 21 in the task pool 13. A thread 21 is composed of one or more tasks 22.” *Id.* at 4:24-28.

B. Prosecution History

The '777 Patent was filed on January 25, 2013, from U.S. Application No. 13/750,696. The '777 Patent was examined by Examiner Wissam Rashid. The '777

Patent received its first office action on March 11, 2015 and was subsequently allowed by Examiner Rashid on June 29, 2015.

C. Claim Construction

All claim terms of the '777 Patent should be given their ordinary and customary meaning as understood by one of ordinary skill in the art based on their plain language in light of the specification and prosecution history.

Petitioner's misplaced reliance on Dr. Lowenthal's Declaration to construe the terms "solidarity cell" and "thread" "task pool" (*see* Petition at 7-8) should be disregarded by the Board as improper extrinsic evidence in view of the unambiguous intrinsic evidence set forth in the '777 Patent claims and specification.

IV. THE PETITION'S GROUNDS AND REFERENCES

Petitioner offers the following two Grounds challenging claims 1-14 of the '777 Patent as being obvious under 35 U.S.C. § 103:

Ground	Claims Challenged	35 U.S.C. (Pre AIA)	Reference[s]
1	1-10, 12-14	§ 103	Lurie (Ex. 1007) and Ethernet Specification (Ex. 1009) ¹
2	Dependent Claim 11	§ 103	Lurie, Ethernet Specification and Lippett (Ex. 1031)

¹ The Petition does not include Andrews as a Reference in support of Ground 1 (*see* Petition at i, 8), and therefore improperly relies on Andrews in arguing Ground 1 (*see* Petition at 19, 33, 35, 36).

The Petition relies principally on Exhibit 1007 (U.S. Patent Application Publication No. 2007/0124363 (“Lurie”)) as its primary reference for both Grounds. The Petition also reaches to a second reference, Exhibit 1009 (*The Ethernet – A Local Area Network, Data Link Layer and Physical Layer Specifications*, Version 1.0 (Sept. 30, 1980) (“Ethernet Specification,”)), in an effort to compensate for the deficiencies in Lurie.

Under Ground 2, Petitioner further looks to Exhibit 1031 (U.S. Patent Application Publication No. 2007/0220517 (“Lippett”)) to address deficiencies in Lurie and Ethernet Specification for dependent claim 11.

A. Overview of Lurie

Lurie discloses “an instrument-based distributed computing system” useful for “accelerat[ing] the measurement, analysis, verification and validation of data in [a] distributed computing environment.” Lurie, Abstract, [0002]. Lurie describes three different embodiments or modes of operation for its distributed computing system, namely: i) direct distribution mode, ii) automatic task distribution mode, and iii) batch mode. *See id.* [0067].

1. Direct Distribution Mode

Lurie’s direct distribution mode is generally described in relation to Fig. 3A (shown below) and Fig. 5A.

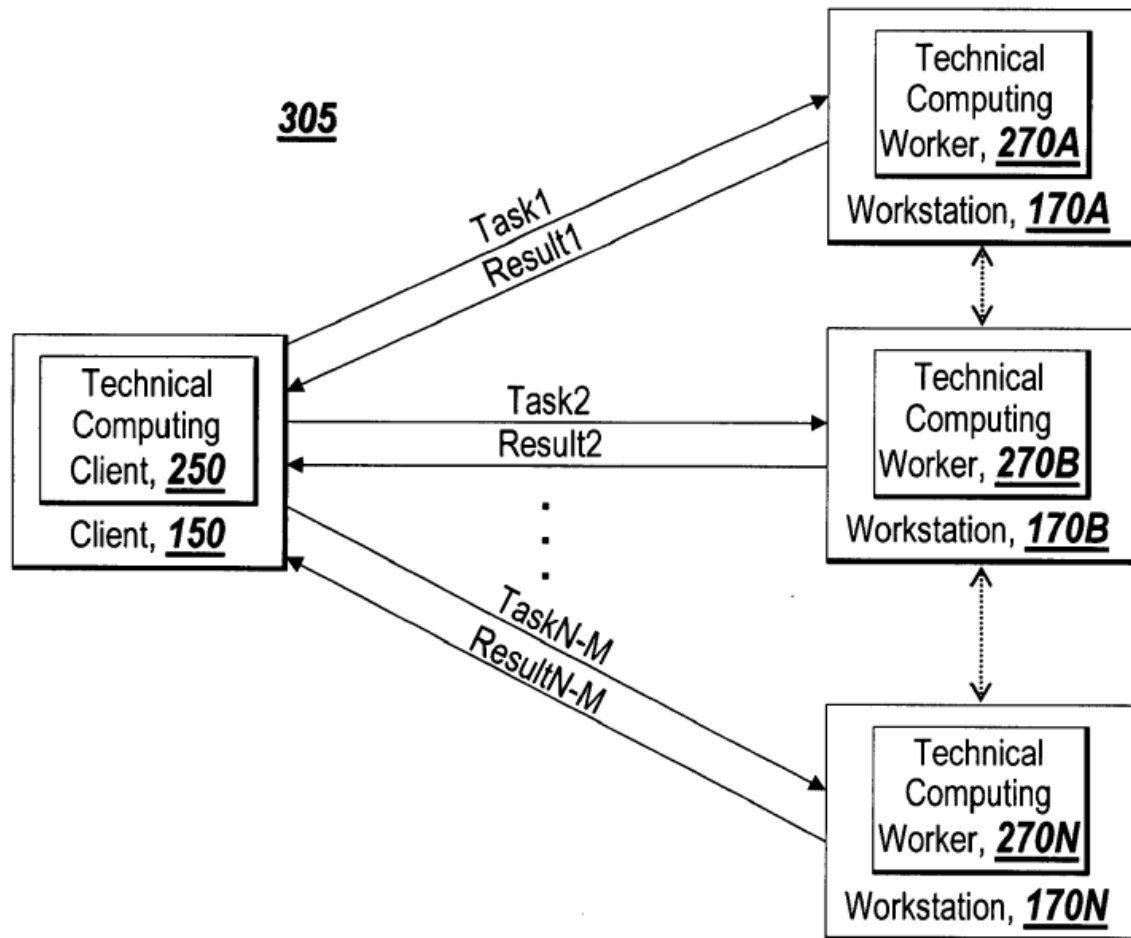


Fig. 3A

According to Lurie’s direct distribution mode, a Technical Computing Client (“TCC”) is in direct communication with a plurality of Technical Computing Workers (“TCWs”). *See id.* [0068]. The TCC can define “a task comprising an operation for technical computing” *Id.* at [0085]. The TCC “can submit multiple tasks (e.g., TaskN-M) to each of the multiple technical computing workers....” *Id.* at [0069]. As a result, the TCWs “can be performing technical computing of their

respective tasks independently and in parallel to each other.” *Id.* at [0069]. Under Lurie’s direct distribution mode, the TCC and TCWs implement a traditional controller/responder processing paradigm.

2. Automatic Task Distribution Mode

Lurie’s automatic task distribution mode is generally described in relation to Fig. 3B (shown below) and Fig. 5B.

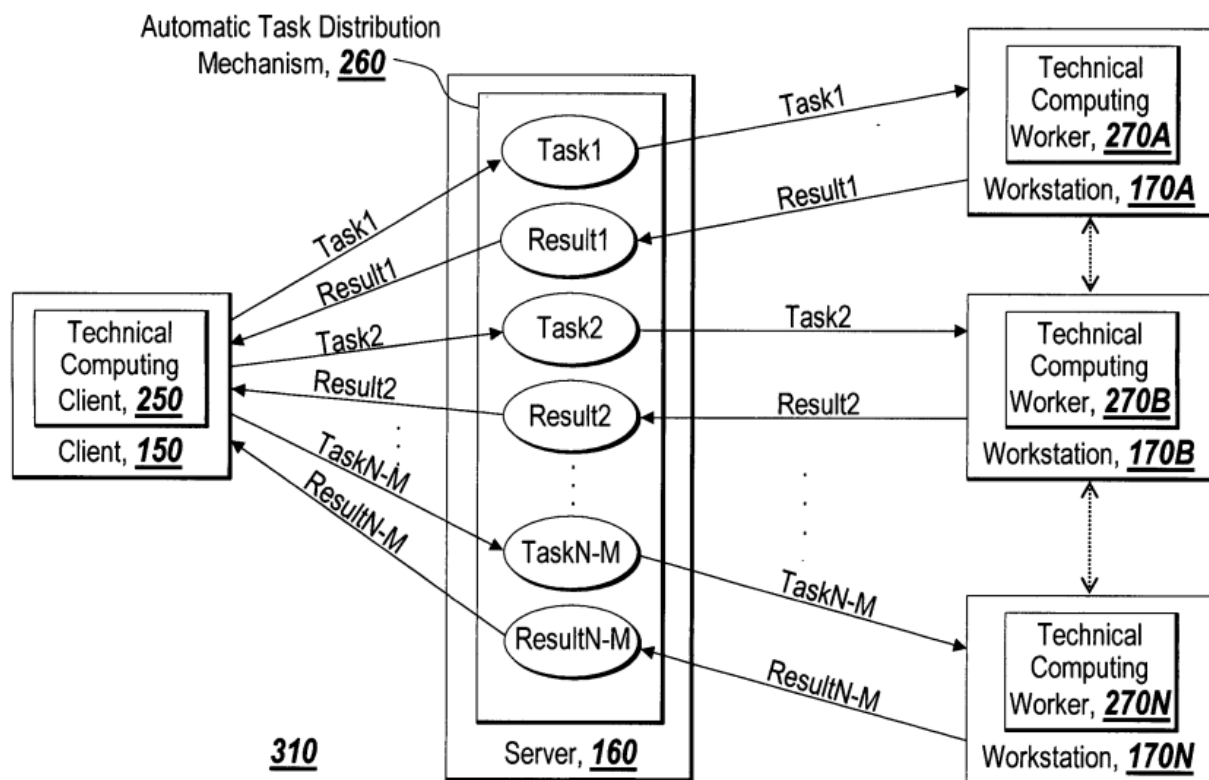


Fig. 3B

Lurie’s automatic task distribution mode includes “a technical computing client application 250, or technical computing client, running on a client 150 computer” and one or more “technical computing worker application 270, or

technical computing worker, running on a workstation 170.” *Id.* at [0054] (“TCW” or “TCWs”).

Lurie’s Automatic Task Distribution mechanism (“ATD Mechanism”) operates in a controller/responder fashion with the TCWs. Lurie’s ATD Mechanism is responsible for “distribut[ing] the one or more tasks automatically to one or more technical computing workers providing technical computing services.” *Id.* at [0048]. It is notable that Lurie’s ATD Mechanism and TCWs operate in a manner almost identical to that of the conventional controller/coprocessors described by the ’777 Patent. *See* ’777 Patent, 1:24-28.

In Lurie’s automatic task distribution system “the technical computing client 250 submits one or more tasks (Task1-TaskN-M) to the automatic task distribution mechanism 260.... The automatic task distribution mechanism 260 obtains the tasks (Task1-TaskN-M) to make then [sic] available for distribution to any of the technical computing workers 270A-270N.” Lurie [0072]. The ATD 15 mechanism “distribute[s] the one or more tasks automatically to one or more technical computing workers providing technical computing services.” *Id.* at [0044].

In Lurie’s automatic task distribution mode, the Technical Computing Client (“TCC”) does not “specify or have knowledge of the technical computing workers....” *Id.* However, Lurie does disclose that registration or availability information relating to the TCWs is communicated throughout “[t]he system (e.g.

315 or 320) [which] can compare the number of technical computing workers 270A-270N registered, or otherwise available, with the job manager 265 or automatic task distribution mechanism 260 against the configured setting of the minimum number of technical computing workers parameter.” *Id.* at [0080].

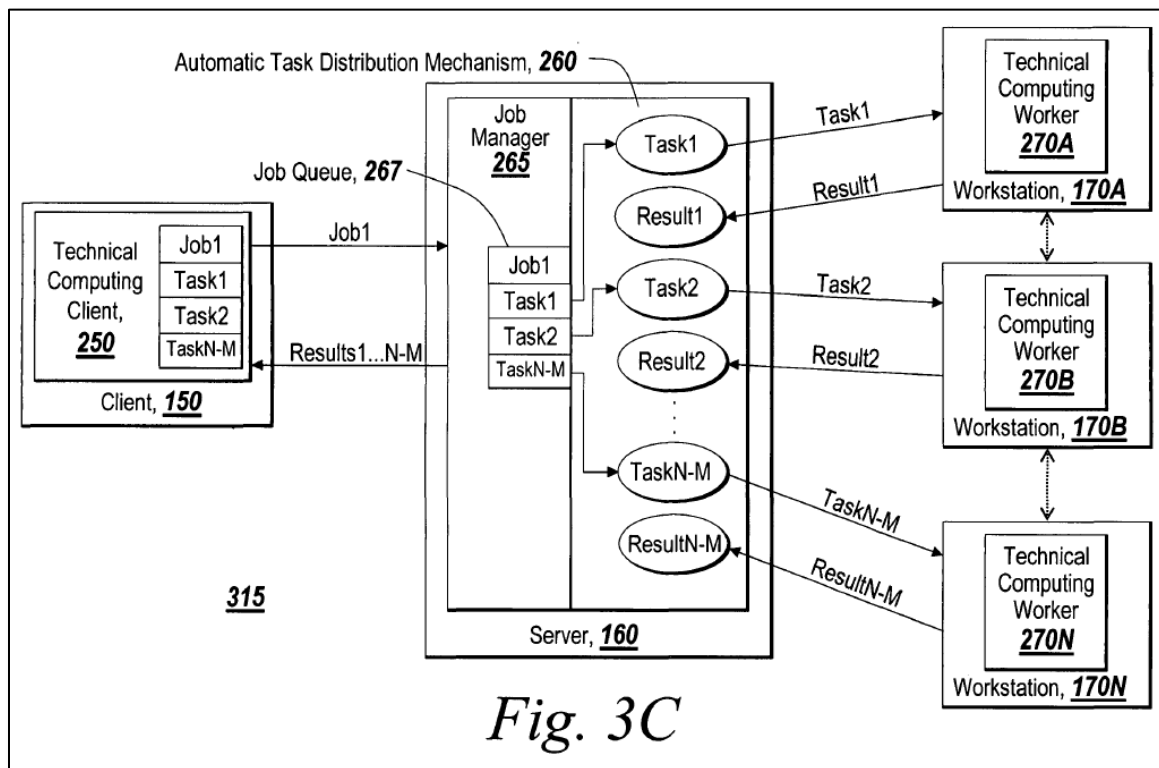
Each TCW registers with the automatic task distribution system “to receive notification of one or more tasks (step 527) becoming available, or appearing, in the automatic task distribution mechanism 260.” *Id.* at [0086]. Then, and only after a TCW registers with the ATD Mechanism, the “automatic task distribution mechanism 260 provides notification (step 538) to the technical computing worker 260 of the availability of the task. In response to receiving the notification (step 540), the technical computing worker obtains (step 544) the task provided (step 540) from the automatic task distribution mechanism 260.” *Id.*

Following the distribution of a task from the ATD Mechanism to a TCW, “[t]he technical computing worker 270 performs the requested technical computing on the function or command as defined by the task (step 508). In performing the technical computing on the task, an associated result may be generated (step 510).” *Id.* Finally, after generating the result from computing the task, “the technical computing worker 270 provides the result (step 512) to the automatic task distribution mechanism 260. After obtaining the result from the technical computing

worker 250 (step 550), the automatic task distribution mechanism 260 notifies (step 552) the technical computing client 250 that the result is available.” *Id.*

3. Batch Mode

Lurie’s Batch mode is generally described in relation to Fig. 3C (shown below) and Fig. 5C.



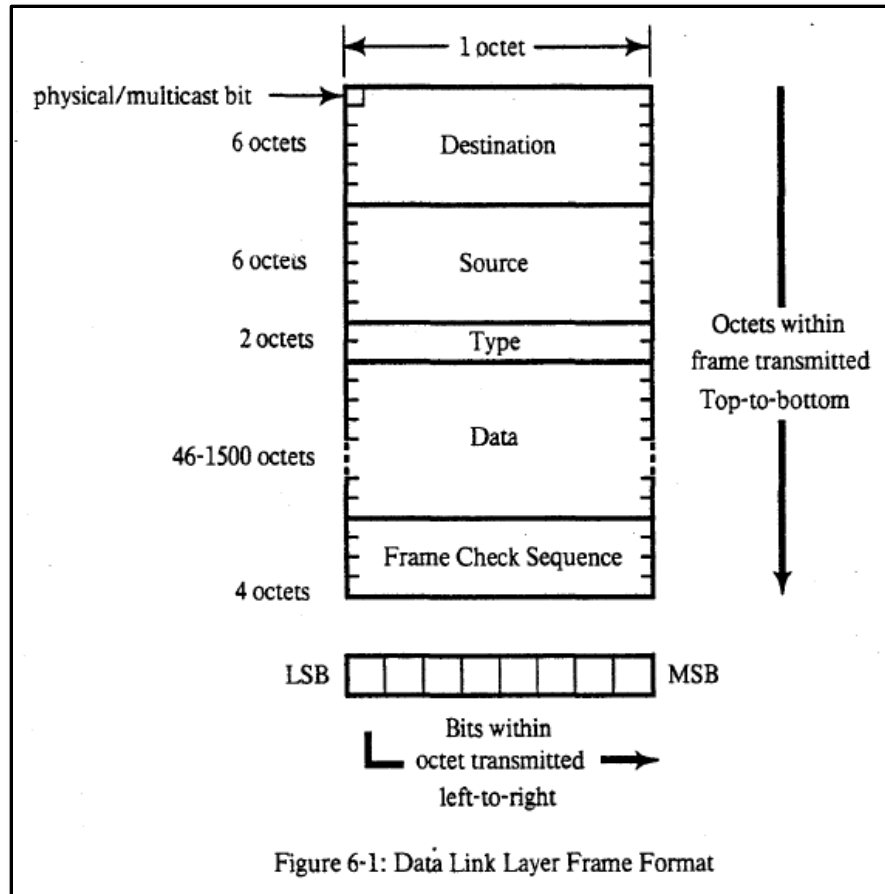
According to Lurie’s batch mode, “the technical computing client 250 defines the job.” *Id.* at [0074]. The TCC has a “programming language environment by which it can declare tasks, declare a job and associate the tasks with the job.” *Id.* Furthermore, “[i]nstead of submitting *each task* separately as depicted in FIG. 3B,

the technical computing client 250 submits *the job* containing all the associated tasks to the *job manager* 265.” *Id.* (emphasis added).

In batch mode and the job manager “obtains the tasks from the job and provides the tasks to the automatic task distribution mechanism...” *Id.* For example, the job manager “obtains Job1 and obtains each of the tasks, Task1-Taskn-M from Job1. Then, ... the job manager 265 submits each of the tasks to the automatic task distribution mechanism 260 for technical computing by a technical computing worker 270A-270N.” *Id.*

B. Overview of Ethernet Specification

The Ethernet Specification discloses a unified standard for establishing a local area network capable of providing “high speed data exchange among computers and other digital devices located within a moderate-sized geographic area.” Ethernet Specification at 1. Among other things, the Ethernet Specification discloses the use of a “Data Link Layer compris[ing] the construction and processing of frames.” *Id.* at 19.



As depicted in Figure 6-1 of the Ethernet Specification, a Data Link Layer Frame can include addressing fields “to identify the source and destination stations for the frame.” *Id.* at 19-20. Furthermore, the Data Link Layer Frame includes a Data Field which contains between 46 and 1,500 octets (bytes) of binary data. *Id.* at 20-22. The Ethernet Specification does not disclose or suggest what information is, or should be, represented by the data in a data field. Instead, the Ethernet Specification simply states that within the data field parameters “full data transparency is provided, in the sense that any arbitrary sequence of octet values may appear in the data field.” *Id.* at 22.

C. Overview of Andrews

Andrews appears to be a textbook directed to “programming multithreaded, parallel, and distributed computations.” Exhibit 1011 (“Andrews”) at vii. The relevant sections of Andrews provided by Petitioner² generally discuss an “adaptive quadrature” problem (integrating a function $f(x)$ from a to b) and include exemplary abstracted code for resolving an adaptive quadrature problem in the context of recursive parallelism. Andrews at 133.

D. Overview of Lippett

Lippett discloses “a method for scheduling executable transactions within a multicore processor comprising a plurality of processor elements.” Lippett, Abstract. Lippett’s executable transactions, referred to as threads, are scheduled by complex scheduling algorithms which more efficiently allocate the executable transactions to processing resources. Lippett’s scheduling method purportedly “improves the runtime performance of an application by more efficiently allocating the executable transactions, or threads, to processing resources.” Lippett, [0005].

Lippett discloses a dedicated, hard-coded preferred embodiment in a server that uses “pointers within a dedicated memory for scheduling allocations, storing values for decision purposes, and the like” where the pointers store parameters useful

² Petitioner segments include pages: v-vii, 2-3, 31-33, 132-139.

for making scheduling decisions, storing values for use in subsequent decisions, or the storing of values of interest in a dedicated memory. *Id.* at [0007].

V. PETITIONER FAILS TO SHOW A REASONABLE LIKELIHOOD THAT IT WOULD PREVAIL ON ANY CHALLENGED CLAIM

For an IPR to be instituted, the Petitioner bears the burden to establish “a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a). Furthermore, it is the Petitioner’s burden to establish, how “the construed claim is unpatentable under the statutory grounds,” and “where each element of the claim is found *in the prior art patents or printed publications* relied upon.” 37 C.F.R. § 42.104. As demonstrated below, neither Petitioner’s grounds (1 and 2) establish a reasonable likelihood that Petitioner will prevail on any challenged claim. Accordingly, Patent Owner respectfully requests that the Board deny institution of *inter partes* review of the ’777 Patent.

A. The Petition fails to show that the cited references teach a “first agent retrieves the *matching task* by ... *searching the task pool*”

Each and every claim (claims 1 – 14) of the ’777 Patent recites, expressly or by incorporation, wherein a “first agent retrieves the *matching task* by: ... *searching the task pool* for a task.”

Independent claim 1 of the ’777 Patent requires a “first solidarity cell comprising a first agent configured to proactively retrieve ... a matching task for the

solidarity cell to process ... wherein the first agent retrieves the matching task by ... searching the task pool for a task that is ready to be processed and has a function that the first solidarity cell can perform” ’777 Patent, 7:45-8:3. Claims 2-13 are dependent on claim 1 and thus include these elements by incorporation.

Independent claim 14 of the ’777 Patent requires similarly requires a “first solidarity cell comprising a first agent configured to proactively retrieve ... a matching task for the solidarity cell to process ... a second solidarity cell comprising a second agent that proactively retrieves matching tasks ... wherein each agent retrieves a matching task by ... searching the task pool for a task that is ready to be processed and has a function that the agent’s solidarity cell can perform” ’777 Patent, 8:55-9:11.

The ’777 Patent discloses that an “agent searches within the task pool for available tasks that match the cell’s instruction type.” *Id.* at Abstract. Furthermore, “it is an object [of this invention] that the co-processors include an agent that searches a task pool to acquire tasks for the co-processors to perform.” *Id.* at 1:50-52. Additionally, “The [solidarity cells] act proactively in that they seek a task 22 from the task pool 13 as soon as the [solidarity cells] become available to do further processing. To acquire a task 22, a [solidarity cell] sends an agent 30A to the task pool 13 to search for and retrieve an available task 22 that requires completion, is not locked, and has a task type that can be performed by the [solidarity cell].” *Id.* at

5:22-28. Moreover, “the agent 30A searches the task 22 descriptors for an executable instruction that matches one of the instructions that that [solidarity cell] is capable of executing. When a matching task 22 is found, the agent 30A delivers the descriptor of the matching task 22 to the [solidarity cell], which begins to process the task 22.” *Id.* at 6:26-31.

Both the plain language of the claims, as well as the detailed description in the specification, clearly indicate that each solidarity cell, through its agent, determines the task to be executed by the solidarity cell, and returns a descriptor of the task back to that solidarity cell.

The cited art fails to teach an agent configured to search the task pool for a task. Nonetheless, with no supporting evidence, Petitioner simply declares that “Lurie (alone or with Ethernet) teaches or suggests that the dispatched agent searches the task pool for a task function that the sending TCW can perform.” Petition at 28. Petitioner is mistaken. Under all modes of operation, Lurie teaches that it is the system (not the TCW) that selects the TCW to work on a task. *See* Exhibit 2001 (“Nelson Decl.”) ¶29. Consequently, Lurie’s TCWs do not search anything, much less the ATD Mechanism.

Petitioner argues that “[a]n obvious way to implement Lurie’s disclosure of matching the TCW based on its capabilities to a task function that it can perform would have been for the TCW to dispatch a data frame or agent carrying a payload

that includes, e.g., one or more indices, names or pointers/references associated with one or more (list of) task types that the TCW can perform—such as a particular version of a particular platform (e.g., MATLAB, LabVIEW, etc.), or a subset of the function/commands in that platform.” Petition at 26-27. Petitioner’s argument is conclusory, based on hindsight reasoning, and is incompatible with Lurie’s express teachings.

In each of Lurie’s Direct Distribution, Automatic Task Distribution, and Batch modes (as described in FIGS 3A-3D), Lurie states that “*the system can determine or select the [TCW] to work on a task by operational and/or performance characteristics of the [TCW].*” Lurie, [0081] (emphasis added). Describing *how* the system can determine the TCW to work on a task, Lurie explains that a TCW “may have a specification or profile ... which provides a description of any operational and performance characteristics of the version of [MATLAB] running on that specific [TCW].” Lurie, [0081]. Lurie further describes that “[t]his profile may be described in a file accessible over the network or retrievable through an interface mechanism of the [TCW].” *Id.* Finally, Lurie highlights that “[f]or determining a [TCW] to work on a task, the system may discover any configuration, properties, and operational and performance characteristics of the MATLAB®-based distributed computing application of a [TCW] through any interface of the [TCW], such as, for example, in a file, graphical user interface, command or message.” *Id.*

In view of the description provided in Lurie paragraph [0081], a POSITA would understand that the *TCW does not* determine which task to work on. For example, a POSITA would understand that it would be non-sensical for the TCW to “discover” its own properties by “interfacing” with itself. See Nelson Decl. ¶32. Contrary to Petitioner’s assertion, a POSITA reading Lurie in its entirety would understand instead that it is another component within the system that is responsible for “discovering” the profile of the TCWs through an interface with the TCW or via a file that is accessible over the network. *Id.* This is consistent with additional implementation details provided throughout Lurie.

For example, Lurie teaches that “the [ATD Mechanism] provides notification to the [TCW] of the availability of *the task* (step 538).” Lurie, [0088] (emphasis added). A POSITA would understand this to indicate that *the task* has already been identified by the ATD Mechanism prior to the notification which is sent to the TCW. See Nelson Decl. ¶33. Furthermore, a POSITA would understand this to indicate that the notification is task-specific. *Id.* As a result of the notification, the TCW simply retrieves the task identified by the notification. This understanding is further supported by the following exemplary embodiment in which “*the task* is sent, by the job manager 265 to the [TCW] *as notification to perform the task.*” Lurie, [0088] (emphasis added).

Again, the task is determined by the system (e.g., the ATD Mechanism or the job runner), prior to the notification being sent to the TCW. In the above Batch Mode example, it is the task itself that is sent to the TCW as notification to perform the task. A POSITA would not understand that a TCW would subsequently “select” which task to retrieve after the ATD Mechanism has already sent a notification to retrieve a particular task (or has sent the task itself) to the TCW. *See Nelson Decl.* ¶133.

This understanding is consistent with Lurie’s other descriptions of how tasks are assigned to TCWs. For example, Lurie teaches that “[t]he job runner 667 can have such logic to determine from the worker pool 668 which [TCW] should be assigned and sent a task from the job queue 267.” Lurie, [0100]. That is, the job runner that makes the determination in the example described in Lurie [0100]. Alternatively, Lurie teaches that “a separate automatic task distribution mechanism 260 can be responsible for determining the TCW to be assigned a task and to send the task to the assigned [TCW].” Lurie, [0100]. In this example the ATD Mechanism determines which task will be assigned to a TCW.

Nowhere does Lurie disclose or suggest that the TCWs participate in task selection or assignment, and therefore Lurie’s TCWs do not and cannot “search” for or “select” tasks.

Dr. Lowenthal argues that “a TCW able to perform only a certain type or types of tasks invokes *getTask* (via its agent) with the list of functions the TCW can perform as a parameter – thus searching the ATD Mechanism for a task that the TCW can perform.” Exhibit 1003 (“Lowenthal Decl.”) ¶169. Dr. Lowenthal is mistaken. Lurie neither describes nor suggests any basis for the parameter of the *getTask* method to include “[a] list of functions the TCW can perform.”

While Lurie mentions the *getTask* method only briefly, a POSITA would understand that the parameter passed with *getTask* is simply the specified “TaskID” which is a “unique task identifier.” Lurie, [0120]; See Nelson Decl. ¶¶36-40. This is consistent with Lurie’s description that “the [ATD Mechanism] provides notification to the [TCW] of the availability of *the task* In response to receiving the notification ... the [TCW] obtains (step 542) *the task* provided (step 540) from the [ATD Mechanism] or the [Job Manager].” Lurie, [0088] (emphasis added). See also Lurie, [0128] (describing that the result-specific *getResult* method “works the same as *getTask*, except a result will be taken rather than a task.”) (emphasis added).

Null is the only other parameter described by Lurie in conjunction with the *getTask* method. When the *getTask* method is called with *null*, the system will return “a task with any TaskID.” Lurie, [0128]. This remains consistent with Lurie’s fundamental mode operation in which it is the system that “determines or selects the [TCW] to work on a task...” Lurie, [0081]; see Nelson Decl., ¶38.

A POSITA reading Lurie would understand that the system selects and assigns tasks to the TCWs, that each notification sent from the ATD Mechanism to the TCW *is task-specific*, and that the retrieval of the task(s) from the ATD Mechanism by the TCW is also *task-specific*. See Nelson Decl. ¶37. In this way, the “[TCW] takes task 1 from the [ATD Mechanism]” Lurie, [0072]. In Lurie’s JavaSpace implementation, the *getTask* method is used to “take” the specific task. See Lurie, [0128] (describing *getTask* as “a task will be *taken* from the space”). Furthermore, when there are “multiple task submissions from the [TCC], the [ATD Mechanism] can manage and handle the delegations of the tasks to the same [TCW], or to other [TCWs]” Lurie, [0062].

In view of the above, Lurie fails to disclose any embodiment in which a TCW participates in the selection of tasks. Rather, in every embodiment, tasks are assigned to the TCWs before the TCWs receive a notification to retrieve a task (or before the TCW receives the task “as notification to perform the task.” Lurie, [0088]). Lurie discloses no embodiment in which a TCW searches for tasks.

Accordingly, Lurie does not teach a “first solidarity cell comprising a first agent configured to proactively retrieve ... a matching task for the solidarity cell to process ... wherein the first agent retrieves the matching task by ... searching the task pool for a task that is ready to be processed and has a function that the first solidarity cell can perform” ’777 Patent, 7:45-8:3; *see also id.*, 8:55-9:11.

Moreover, Petitioner has not shown that a POSITA would be motivated to modify Lurie to meet the claimed limitations. See Nelson Decl. ¶¶42, 51-56. As discussed in more detail below, Petitioner's proposed modifications render portions of Lurie's system inoperable and fundamentally alter Lurie's principles of operation. As a result, a POSITA would not find Petitioner's proposed modifications to be obvious. *Id.*

There is nothing to suggest that the remaining cited art can supply these missing limitations. Petitioner argues that Lurie with Ethernet Specification teach the claimed agent. Petition at 28. Petitioner is mistaken. Neither Lurie nor Ethernet Specification, alone or in combination, teach an agent as that term would be understood by a POSITA in the context of claims 1-14. With regard to Ethernet Specification, there is nothing to suggest that an Ethernet data frame can, by itself, provide the searching capabilities defined in the '777 specification and required by the claims.

Describing its own disclosure, the Ethernet Specification states, "this document provides precise detailed definitions of the lowest two layers of an overall network architecture It does not specify the higher levels of protocols needed to provide a complete network architecture. Such higher-level protocols would generally include such functions as internetwork communication, error recovery, flow control, security measures (e.g., encryption), and other higher-level functions

that increase the power of the communication facility and/or tailor it to specific applications.” Ethernet Specification at 1-2.

A POSITA would understand that while the Ethernet Specification provides methods for facilitating communication and data transfer between computing devices, it does not teach or suggest anything regarding the content of such communications or data transfers, such as any particular content of a payload. *See* Nelson Decl. ¶44.

As a result, the Ethernet Specification cannot supply the missing limitations such as a “first solidarity cell comprising a first agent configured to proactively retrieve ... a matching task for the solidarity cell to process ... wherein the first agent retrieves the matching task by ... searching the task pool for a task that is ready to be processed and has a function that the first solidarity cell can perform” ’777 Patent, 7:45-8:3. *See also* ’777 Patent, 8:55-9:11. Accordingly, a POSITA would not find the claims of the ’777 Patent obvious over the cited art. *See* Nelson Decl. ¶44.

B. Claims 1-14: Petitioner fails to show that the cited art teaches a “first agent configured to *proactively retrieve*, from the task pool . . . a matching task”

The ’777 Patent is entitled “Parallel Processing with Solidarity Cells by *Proactively Retrieving* from a Task Pool a Matching Task for the Solidarity Cell to Process” (emphasis added). Every claim of the ’777 Patent recites, expressly or by incorporation, a “first solidarity cell comprising a first agent configured to

proactively retrieve ... a matching task for the solidarity cell to process.” ’777 Patent, 7:46-49; *see also id.*, 8:56-58.

The ’777 Patent states that “it is an object of this invention to provide an apparatus and method for parallel processing in a multiprocessor system using co-processors that proactively seek threads to process.” *Id.* at 1:46-49. Describing proactivity of the solidarity cells, the ’777 Patent discloses that “[e]ach solidarity cell is proactive, in that it obtains a task to perform by sending its agent to the task pool when the solidarity cell has no processing to perform.” *Id.* at 2:8-10. Moreover, “the [solidarity cells] act proactively in that they seek a task 22 from the task pool 13 as soon as the [solidarity cells] become available to do further processing.” *Id.* at 5:22-25.

Succinctly, claims 1 – 14 of the ’777 Patent recite that the solidarity cells “proactively” retrieve tasks from the task pool without instruction from the CPU or the task pool; that is, they retrieve tasks without being notified, instructed, or otherwise prompted to do so. In contrast, each and every embodiment in Lurie requires the TCWs to receive *a predicate notification* from the ATD Mechanism before obtaining each task. As detailed herein, Lurie’s ATD and batch modes require a notification or prompt to the TCW as a predicate to task retrieval, and thus Lurie’s TCWs do not and cannot “proactively” retrieve matching tasks. Nothing about Lurie’s Javaspace implementation alters this conclusion.

The cited art fails to teach an agent configured to proactively retrieve a matching task. Yet Petitioner summarily states, without any support from Lurie, that “Lurie’s TCWs do not receive an instruction to retrieve a task from the ATD.... Lurie’s TCWs ‘register[]’ with the ATD ‘to receive notification of one or more tasks (step 527) becoming available, or appearing, in the [ATD]. Lurie, [0087], [0097] In other words, the ATD’s notification merely informs the TCWs when tasks first become available.” Petition at 23-24. Petitioner’s interpretation of Lurie is mistaken and contrary to Lurie’s plain language. While it is correct that the TCWs register to receive notifications about tasks, nowhere does Lurie teach that the task notifications are merely generic broadcasts to all TCWs for the sole purpose of identifying *when* tasks have become available.

Indeed, the very paragraphs cited by Petitioner belie Petitioner’s interpretation. For example, Lurie describes that “[w]hen *the task* is submitted to the object exchange repository 662 by the job manager 265, the technical computing worker 270 receives a notification and takes *the task* from the object exchange repository 662 by performing a take operation.” Lurie, [0097] (emphasis added). Lurie provides, in an exemplary embodiment, that the TCWs “receive *a task* from the job manager 265 or the automatic task distribution mechanism 260 as *notification* to perform computing on the task.” Lurie, [0087] (emphasis added). A

POSITA would understand that in all embodiments, the notifications are TCW-specific and task-specific. See Nelson Decl. ¶48.

This understanding is further supported by the fact that the task notifications are sent as a result of the system “determin[ing] or select[ing] the [TCW] to work on a task by operational and/or performance characteristics of the [TCW].” Lurie, [0081]. The fact that Lurie’s task notifications may identify “one or more tasks” does not change the previous analysis. And this is consistent with Lurie’s disclosure that “[t]he automatic task distribution mechanism 260 provides notification (step 538) to the technical computing worker 260 of the availability of the task,” and, “[i]n response to receiving the notification (step 540) the technical computing worker *obtains* (step 544) *the task provided* (step 540) from the automatic task distribution mechanism 260.” Lurie, [0086], (emphasis added). See also [0071]-[0072], Figs. 3B and 5B (relating to ATD mode), and [0087]-[0088], [0073]-[0076], Figs. 3C and 5C (relating to Batch Mode).

In all cases, a POSITA would understand that Lurie’s TCW is dependent upon a notification from the ATD Mechanism before obtaining each task and Lurie does not provide any disclosure where a TCW obtains a task without first being notified to do so. As previously observed by the Board, on two separate occasions, Lurie “fail[s] to teach a technical computing worker ‘proactively’ retrieving a task from the automatic task distribution mechanism.” Exhibit 2003 (“IPR2022-00283

Preliminary Guidance”) at 17. *See also* Exhibit 2004 (“IPR2022-00633 Preliminary Guidance”) at 17.

Petitioner argues that it would have been obvious to modify Lurie’s TCWs to proactively retrieve tasks to “reduce unnecessary message traffic” and to “eliminat[e] a deadlock problem.” Petition at 34. Petitioner’s rationale is based two unstated and faulty assumptions:

First, Petitioner assumes that the TCWs determine and select their own tasks; and

Second, Petitioner assumes that Lurie’s notifications are not task-specific.

That is, Petitioner argues that Lurie’s notifications are nothing more than vague, generic indicators. Having manufactured this “problem” of unnecessary notifications, Petitioner then proposes modifying Lurie to incorporate the functionality of the ’777 Patent, namely, modifying Lurie’s TCWs to proactively retrieve tasks. This is classic hindsight reconstruction. *See* Nelson Decl. ¶52.

A POSITA reading Lurie would understand the role of Lurie’s task-specific notifications in the context of “determine[ing] or select[ing] the [TCW] to work on a task by operational and/or performance characteristics of the [TCW].” *See* Nelson Decl. ¶52 (quoting Lurie, [0081]). A POSITA would not find Petitioner’s proposed modifications to be obvious because they fundamentally alter these principles of operation of Lurie. *Id.*

Furthermore, Petitioner’s proposed modifications render multiple components of Lurie’s system inoperable or moot. For example, Petitioner proposes modifying Lurie such that there exists only “one initial notification to kick off the procedure. The initial notification informs all registered TCWs that one or more tasks are available.” Lownethal Decl. ¶196. *See also* Petition at 33.

Petitioner’s “kick off” theory, however, would render Lurie’s job manager entirely inoperable. Lurie’s job manager is configured to “provide control of delegating tasks and obtaining results” Lurie, [0063]. The job manager “go[es] through the list of [TCWs] registered in the worker pool 667 and send[s] communications to each of the [TCWs] to determine their status and if they are available.” *Id.* at [0099]. The job manager ensures the completion of jobs (and their associated tasks) in a predetermined and ordered manner. “[F]or example, the job manager may handle jobs in a first-in first-out (FIFO) manner. In this case, the job manager 265 does not process the next job until all the tasks from the current job have been processed by the automatic task distribution mechanism 260.” Lurie, [0075]. Moreover, “[t]he system may not start a job unless there is a minimum number of [TCWs] registered or available to work on the job.” *Id.* at [0080].

A POSITA would understand that Petitioner’s proposed single notification “kick off” theory would change the above-described principles of operation of the

job manager and the resulting haphazard task completion would run counter to Lurie's express design. See Nelson Decl. ¶¶54-55.

In view of the above, a POSITA would not understand Lurie to teach or suggest a "first solidarity cell comprising a first agent configured to proactively retrieve ... a matching task for the solidarity cell to process." And there is nothing to suggest that the remaining cited art can supply these missing limitations. See Nelson Decl. ¶56.

Moreover, Petitioner has not shown that a POSITA would be motivated to modify Lurie to meet the claimed limitations. Petitioner's proposed modifications render portions of Lurie's system inoperable and fundamentally alter Lurie's principles of operation. As a result, a POSITA would not find the claims of the '777 Patent obvious over the cited art. *Id.*

C. Claims 1-14: Petitioner fails to show that the cited art teaches a first agent having a *"payload compris[ing] a list of functions"*

Every claim of the '777 Patent recites, expressly or by incorporation, a first agent having a *"payload compris[ing] a list of functions the [agent's] solidarity cell is configured to perform."* '777 Patent, 5:55-57; see also *id.* 5:35-37, 5:39-40. The payload containing the list of functions the cell is capable of performing facilitates the agent searching the task pool for a matching task.

Petitioner initially discusses the "list of functions" claim element in connection with claim Element 14[h.1] and incorporates those same arguments in its

discussion of Claim 1. Petition at 24 and 42-43. Petitioner purports to map Ethernet Specification's data frame payload to the claimed payload and argues that Ethernet Specification "allows for any type of data or payload." Petition at 25. However, nowhere does Ethernet Specification, or any other reference, teach the *contents* of the payload. This is consistent with Ethernet Specification being a standard for general ethernet communication, without regard to any particular implementation or application.

Petitioner asserts that Lurie teaches that the TCW retrieves tasks, via its agent, "based on 'operational and/or performance characteristics' or 'any configuration or properties' of the TCW." *Id.* at 26 (citing Lurie, [0081]). However, this disclosure in Lurie relates to how *the system* may determine the capabilities of the TCWs to facilitate assigning tasks to the TCS prior to sending a notification indicating that a task is available. Lurie does not disclose the TCW dispatching an agent having a list of functions in its payload as part of task retrieval.

The profile describing the capabilities of the TCW "may be described in a file accessible over the network or retrievable through an interface mechanism of the technical computing worker." Lurie, [0081]. There is nothing in Lurie that suggests that the TCW sends such information to the ATD *as part of the TCW retrieving a task*.

More fundamentally, Lurie only describes the TCW retrieving a task *after* first receiving a notification from the ATD that the task is available. And, as discussed above, Lurie teaches that the ATD completes task selection and assignment prior to sending out a notification for the TCW to retrieve a task.

Accordingly, the cited art fails to teach an agent having a payload which includes a list of tasks as called for in Claims 1-14.

VI. CONCLUSION

Patent Owner respectfully submits that institution of the requested *inter partes* review should be denied. Among other things, Petitioner fails to show that any of the cited references, alone or in combination, disclose or suggest:

1. a “first agent retrieves [a] matching task by: . . . *searching the task pool* for a task” as called for in claims 1 - 14.
2. “first solidarity cell comprising a first agent configured to *proactively retrieve*, from the task pool . . . *a matching task*” as called for in claims 1 - 14.
3. a first agent having a payload comprising a list of functions the agent’s solidarity cell is configured to perform as part of retrieving a task as called for in claims 1 - 14.

Together, these limitations address all of the claims of the ’777 Patent. None of the references cited by the Petition, singly or combined, disclose or renders

obvious these limitations. Therefore, the Petition fails to satisfy the statutory burden under 35 U.S.C. § 314(a) for instituting *inter partes* review.

For all of the foregoing reasons, the Board should deny institution.

Respectfully submitted,

Dated: April 10, 2022

By: /Michael K. Kelly/
Michael K. Kelly
Reg. No. 32,848
Beus Gilbert McGroder
701 North 44th Street
Phoenix, Arizona 85008
Phone: 480.652.0083
Email: mkelly@beusgilbert.com
Lead Counsel for Patent Owner

CERTIFICATE OF COMPLIANCE

This paper complies with the requirements of 37 C.F.R. § 42.24 and contains 6,993 words (exclusive of the table of contents, the table of authorities, the table of exhibits, the certificate of service, and this certification).

This paper complies with the typeface requirements of 37 C.F.R. § 42.6 and has been prepared in 14-point Times New Roman proportionally spaced typeface with normal spacing.

Respectfully submitted,

Dated: April 10, 2022

By: /Michael K. Kelly/
Michael K. Kelly
Reg. No. 32,848
Beus Gilbert McGroder
701 North 44th Street
Phoenix, Arizona 85008
Phone: 480.652.0083
Email: mkelly@beusgilbert.com
Lead Counsel for Patent Owner

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on the below date, I caused the forgoing to be served by electronic mail to the following:

Jonathan Strang
Adam Greenfield
David Zucker
Kimberly Li

Latham & Watkins LLP
555 Eleventh Street, NW,
Ste. 1000 Washington, D.C.
20004-1304

Jonathan.Strang@lw.com
Adam.Greenfield@lw.com
David.Zucker@lw.com
Kimberly.Li@lw.com

Dated: April 10, 2022

By: /Michael K. Kelly/
Michael K. Kelly
Reg. No. 32,848
Beus Gilbert McGroder
701 North 44th Street
Phoenix, Arizona 85008
Phone: 480.652.0083
Email: mkelly@beusgilbert.com
Lead Counsel for Patent Owner