Exhibit 596-08

Invalidity of the ’596 Patent by Hammer ’954 and Other Prior Art

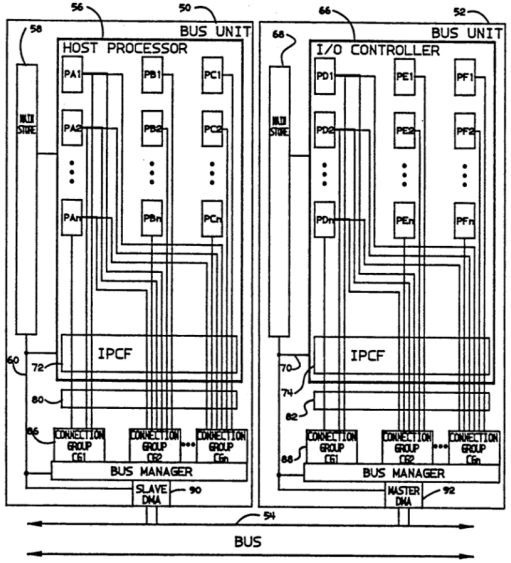
The asserted claims of U.S. Patent No. 8,924,596 are anticipated and/or rendered obvious in view of US5204954A (“Hammer ’954”) alone or in combination with knowledge of one of skill in the art or other prior art.

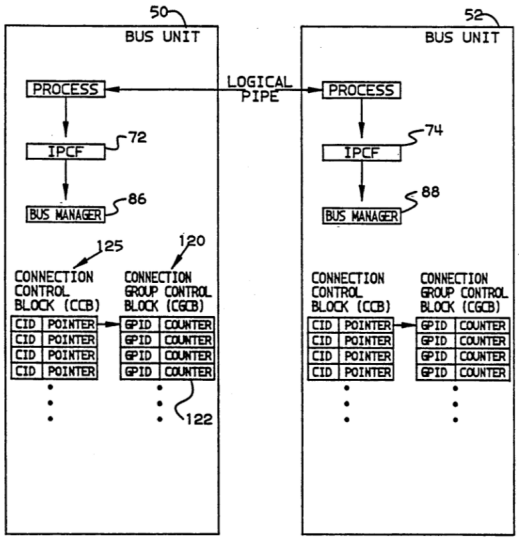
This claim chart is based on Defendants’ present understanding of the asserted claims, Plaintiffs’ vague and overbroad infringement contentions (and corresponding, implicit claim constructions), and Defendants’ investigation to date. Defendants are not adopting Plaintiffs’ apparent constructions, nor are Defendants admitting to the correctness of any particular construction. Defendants reserve all rights to amend this invalidity claim chart in light of any claim construction developments or any amendments to Plaintiffs’ infringement contentions. Further, as discovery is ongoing and Defendants continue to seek discovery from third parties regarding the references identified in Defendants’ invalidity contentions as well as other potential prior art, Defendants reserve the right to amend or supplement this claim chart in connection with Defendants’ continuing investigation. At a minimum, Defendants intend to revise these invalidity contentions as appropriate based on discovery from third parties as discovery progresses.

The claim chart below identifies where each limitation of each asserted claim of the ’596 patent can be found in Hammer ’954. This analysis is based on Plaintiff’s interpretation of the claims as indicated in their Infringement Contentions. The citations provided below are exemplary, rather than exhaustive, and Defendants reserve the right to rely upon any other portion of the cited references.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	To the extent the preamble is considered a claim limitation, Hammer ’954 teaches “[a] system for dividing and synchronizing a processing task across a plurality of processing elements,” including at least the following exemplary citations. For example, Hammer ‘954 teaches a system for dividing and synchronizing processing tasks by disclosing, as non–limiting examples, a send queue, a receive queue, an intermediate queue, bus units, and multiple processing units that coordinate, divide, and synchronize messages and data requests. *See, e.g.:* “The flow of work requests in a server driven process to process communication environment is described. Logical connections between processes and bus managers interfacing bus units to an I/O bus are assigned to connection groups for management by the bus managers. Each bus unit has its own connection groups for the logical connections. Bus unit resources are assigned to each connection group based on performance factors, and a series of bus unit messages are used to control the flow of work so that a group which has no more resources will not accept further work requests. The originator of the work requests will resequence rejected work requests and resend them when the connection group has freed up resources. A further mechanism is provided to facilitate work consistent with the server driven architecture when bus units do not have adequate DMA capabilities. Two ways of reversing control of transfer of work requests and data so that the server need not have master DMA capability are presented. Management of storage in a remote processor is used to transfer work and its associated data into storage accessible by a bus unit with slave DMA capability. The slave DMA bus unit then transfers the information into storage is manages. In another way of reversing the

IPR2025-00223
Concurrent Ventures LLC, XtreamEdge Inc. EX2032

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	flow, a bus unit message is used to make the original server a requestor. The bus unit message contains information which varies the request sent by the requestor. In this manner, the server, which was the original requestor transfers information using its master DMA capability flow." Hammer '954 at Abstract.  FIG. 2 Hammer '954 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

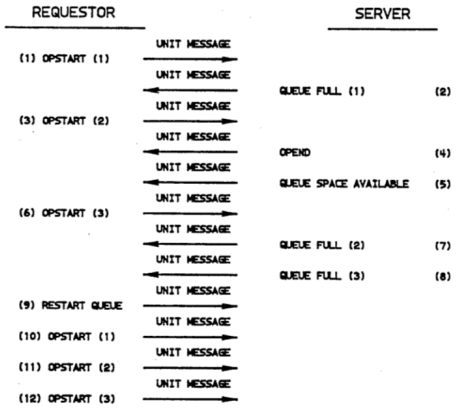


FIG. 12

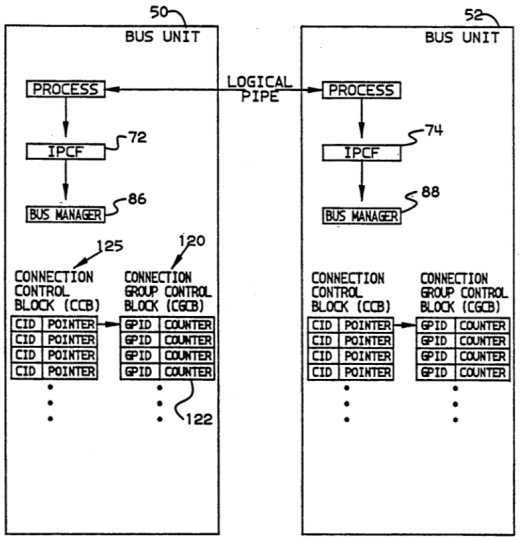
Hammer ’954 at Figure 12.

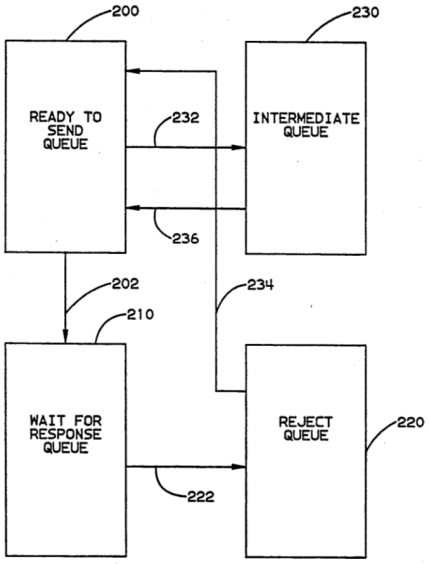
“A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35.

“In one preferred embodiment, the requester bus unit is a storage controller having master DMA capability, and the server bus unit is a host having slave DMA capability. A bus unit is a processing unit attached to the bus in a distributed processor network. It may include one or more host processors for example. Master DMA capability is simply the ability of a bus unit to access the main storage of a bus unit with slave DMA without interrupting the processor in the bus unit. The host is not able to access the controller main storage to obtain the work request nor the associated data. The controller, when a process within it generates a work request, will then place the work request and the associated data into storage in the host which is managed by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller. Management responsibility includes the right to read and write the remote storage as desired. The host may also read the remote storage and write responses in controller specified locations. The controller then sends an Opstart message to the host. The Opstart message is a bus message which indicates that there is a work request to be acted on by a server process and identifies the location in remote storage (actually local storage to the server) where a control block is. The control block identifies the remote storage location of the work request and associated data. The host then transfers the information into storage it manages for a server process to access. The host writes the response into the storage indicated in the control block." Hammer '954 at col. 2, ll. 36–63. "In FIG. 2, two bus units, 50 and 52 are coupled by a physical path such as an I/O (input and output) bus 54. In one preferred embodiment, bus unit 50 comprises a host processing system with a processor 56 having a main storage 58 coupled thereto by a line 60. Host processor 56 executes a plurality of programs, from accounting programs to operating system programs. An executing instance of a program is referred to as a process. Several processes PA1-PAn, PB1-PBn and PC1-PCn are shown in host processor 56." Hammer '954 at col. 6, ll. 11–20. "Bus unit 52, in one preferred embodiment comprises an I/O processor 66 having a main storage 68 coupled by a line 70. I/O processor 66 also has a number of processes PD1-PDn, PE1-PEn and PF1-PFn which may be executing. While a large number of processes are shown in each of the processors 56 and 66, there may be situations where a processor will have only one process. In further embodiments, processor 66 is a peer processor in a network of processors. Processes communicate with each other, regardless of the locations of two communicating processes through IPCF 72 in bus unit 50 and IPCF 74 in bus unit 52. Each pair of communicating processes is logically connected through IPCF by logical connections indicated by lines 80 in bus unit 50 and 82 in bus unit 52." Hammer '954 at col. 6, ll. 21–35. "The establishment of a connection between a process in host processor 56 and a process in I/O processor also 66 results in bus manager 88 completing the connection and assigning its processes connection to one of multiple connection groups CG1, CG2 . . . CGn. Thus, for each process in the system comprising the bus units, that has a connection to another process, the bus manager of the processor where the target process is located assigns its connection to a group, and allocates resources to the connection group to be used to serve work requests on the connection assigned to that group." Hammer '954 at col. 10, ll. 46–56. "In FIG. 12 an interchange of messages in response to a full queue for a connection group in a target bus unit is shown. The target bus unit is labeled as the server, and the source bus unit is labeled as the requestor. Messages sent and received by the server are shown on the server side of the Figure because that is where the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	flow is controlled. Arrows between the requestor and server point to the bus unit which received each message. The sequence of messages is indicated in numerical order. The events are now listed, in the order they appear in FIG. 12:" Hammer '954 at col. 12, ll. 29–39. "As discussed above, the normal flow of work requests is from the host processor 56 to the I/O processor 66. They include work such as reading and writing data to secondary storage devices coupled to the I/O processor 66, or initiating communications through the I/O processor 66. Having an I/O processor 66 with master DMA capability and a host processor 56 with slave DMA capability is ideally suited to this relationship. The server being the I/O processor drives the transfer of data without interrupting the host." Hammer '954 at col. 18, ll. 44–53.
[1a] an input queue implemented in hardware;	Hammer '954 teaches "an input queue implemented in hardware," including at least the following exemplary citations. For example, Hammer '954 teaches an input queue implemented in hardware, by disclosing, as non–limiting examples, a ready to send queue. *See, e.g.:* Defendants have proposed a construction that "queue implemented in hardware" means "queue implemented in physical circuits without software intervention." The input queue in Hammer '954 meets Defendants' construction because, for example, Hammer '954 teaches a hardware implementation of its teachings, which includes its teachings of queues. *See, e.g.*, Hammer '954 at 7:12–14 ("The bus hardware also controls queueing of messages from the bus units in main storage queues.").

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Hammer ’954 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

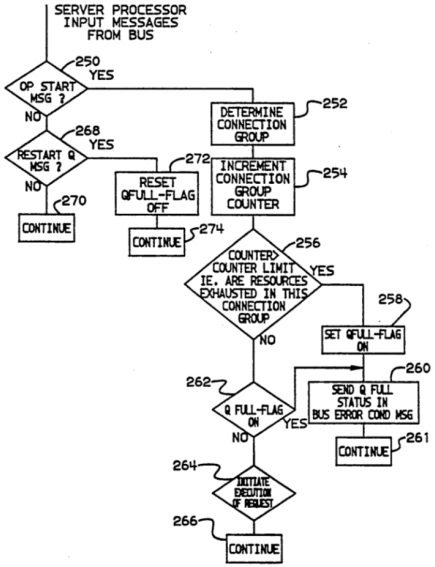


FIG. 7

Hammer ’954 at Figure 7.

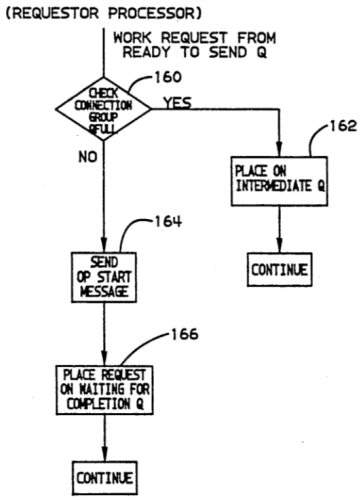
“A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35.

“FIG. 4 is a block flow diagram showing a requesting bus unit responding to a Queue Full status for work requests destined for a connection group which does not have sufficient resource to accept more work requests.

FIG. 5 is a further flow diagram of the requesting bus unit of FIG. 4 for responding to a Queue Full status.

FIG. 6 is a block diagram of queues used by the requesting bus unit of FIG. 4 for responding to a Queue Full status.” Hammer ’954 at col. 3, ll. 23–32.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26.
[1b] an output queue implemented in hardware;	Hammer '954 teaches "an output queue implemented in hardware," including at least the following exemplary citations. For example, Hammer '954 teaches an output queue implemented in hardware, by disclosing, as non–limiting examples, a wait for response queue or queueing of messages from the bus units. *See, e.g.:* Defendants have proposed a construction that "queue implemented in hardware" means "queue implemented in physical circuits without software intervention." The output queue in Hammer '954 meets Defendants' construction because, for example, Hammer '954 teaches a hardware implementation of its teachings, which includes its teachings of queues. *See, e.g.*, Hammer '954 at 7:12–14 ("The bus hardware also controls queueing of messages from the bus units in main storage queues.").

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Hammer ’954 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

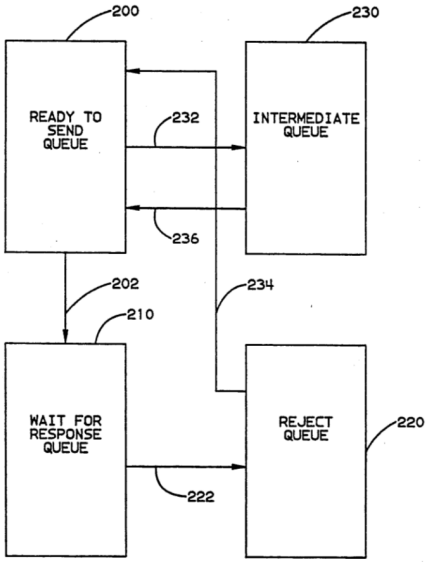


FIG. 6

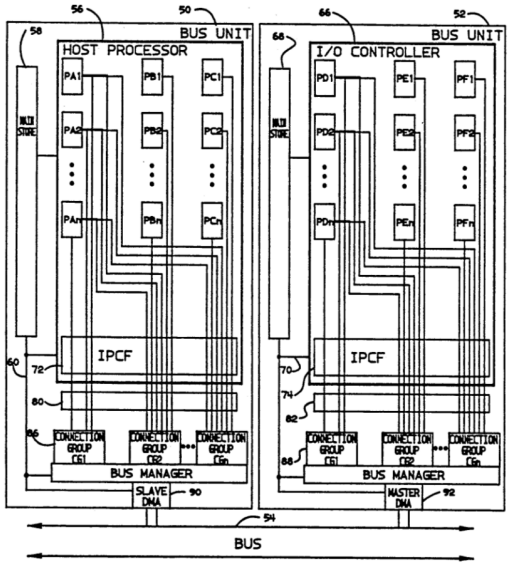
Hammer ’954 at Figure 6.

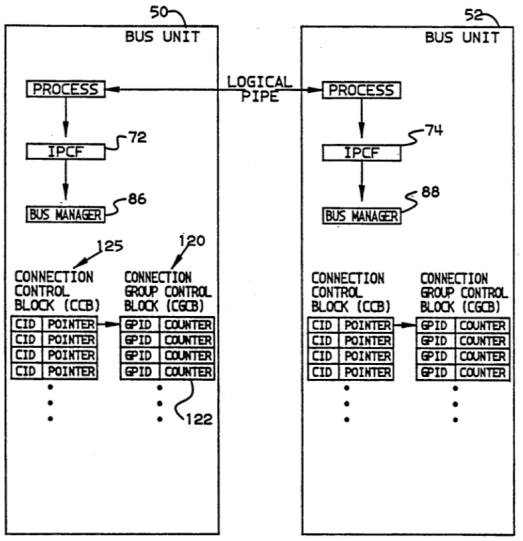
“In FIG. 2, two bus units, 50 and 52 are coupled by a physical path such as an I/O (input and output) bus 54. In one preferred embodiment, bus unit 50 comprises a host processing system with a processor 56 having a main storage 58 coupled thereto by a line 60. Host processor 56 executes a plurality of programs, from accounting programs to operating system programs. An executing instance of a program is referred to as a process. Several processes PA1-PAn, PB1-PBn and PC1-PCn are shown in host processor 56.” Hammer ’954 at col. 6, ll. 11–20.

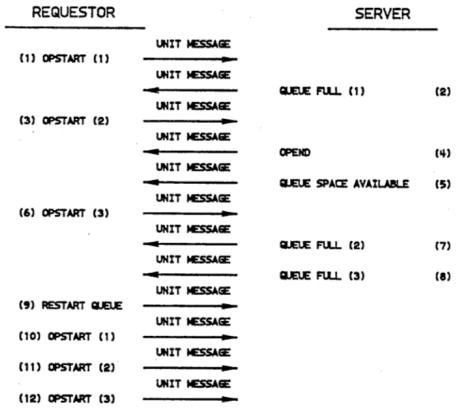
“In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "1. The requestor processor sends a message that initiates work, which is called an OPSTART(1) message, (defined later in this description) to the server processor. 2. The server processor has recognized a QUEUE FULL condition for the connection group that the OPSTART message was intended for, and returns an error message with QUEUE FULL status. 3. Because of the asynchronous nature of the bus units, the requestor is not yet aware of the QUEUE FULL status, and it sends a second message, OPSTART(2) to the server. 4. The server will have completed some previous work request, and will signal that to the requestor by sending an OPEND message." Hammer '954 at col. 12, ll. 40–54. "In the preferred embodiment of FIG. 2, the bus hardware 90 and 92 of bus units 50 and 52 are labeled as having direct memory access (DMA) capability. This capability is a standard hardware capability that exists most hardware today. Master DMA capability allows the bus unit which has it to directly access the main storage of a bus unit with slave DMA capability without interrupting the processor in the slave DMA bus unit. Hence, the details of its operation will not be described, as they are not necessary for a full understanding of the invention. Bus unit 50 has a slave DMA capability 90. In the preferred embodiment, bus unit 50 is a host processor. Slave DMA capability permits a bus unit to let other bus units access its main store 58 without interrupting the host processor 56. Thus, line 60, which gives host processor 56 access to main store 58 is also connected to bus manager 86, and to slave DMA hardware to permit 90 access directly to main store 58. This in turn permits another another bus unit, such as bus unit 52 which has master DMA hardware 92 capability to access main store 58 of bus unit 50 without interrupting host processor 56. A bus unit with only slave DMA capability cannot directly access the main store of any other bus unit, while a unit with only master DMA capability cannot enable any other bus unit trying to directly access its main store." Hammer '954 at col. 13, ll. 13–39. "In normal flow, where it is assured that each bus unit has full DMA capability, bus manager 88 would receive information from IPCF verbs from IPCF 74, which indicates that a process desires to send a work request. Bus manager 88 would then alert bus manager 86 that it has work to do by sending an OPSTART message. The OPSTART bus message format is shown in FIG. 13, and has sufficient information for bus manager 86 in server bus unit 50 to move a Request/Response Control Block (RRCB) that specifies control

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	information and data addresses from bus unit 52 main storage 68 into main storage 58 in bus unit 50, if it could. Further detail of the RRCB is shown in FIG. 14 and 15. Bus manager 86, could then alert the intended process in host processor 56 through IPCF 72 that there is a pending work request, which would be queued by the process. The process would then execute the request, which would require data movement between the bus units. The copy of the RRCB, that would now be in main storage 58 would be used by the bus manager 86 to control the transfer of data between the bus units. The requested operation would be signalled complete by an OPEND (see FIG. 16) message being sent to bus manager 88, which would alert the requesting process through IPCF 74." Hammer '954 at col. 13, l. 49 – col. 14, l. 5. "The problem with the above scenario is that as implemented in FIG. 2, bus hardware 90 cannot directly access main store 68. Even if bus hardware 90 had master DMA capability, bus hardware would have to have slave DMA capability as well. The problem is solved by using a storage list control block and several new bus messages to give management of buffers in main store 58 of host bus unit 50 to the bus manager 88 of bus unit 52. This permits the bus manager 88 in the requestor to transfer data related to the request to the buffers in main store 58 of the server, and then the server to transfer the data from the buffers into storage usable by the server process, in accordance with the normal flow of work. Thus, data flow appears normal to IPCF 72. The RRCB is used to indicate, as usual, where the data is that the server must access. The requestor bus manager 88 simply ensures that the data is located in buffers in main store 58. A more detailed description of the RRCB and messages will now be given." Hammer '954 at col. 14, ll. 6–24.
[1c] a first processing element having access to said input queue and said output queue;	Hammer '954 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. For example, Hammer '954 teaches a first processing element having access to said input queue and said output queue, by disclosing, as non–limiting examples, a I/O processor or a requestor processor. Both processors have access to both their respective input queue and output queues. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Hammer ’954 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “In one preferred embodiment, the requester bus unit is a storage controller having master DMA capability, and the server bus unit is a host having slave DMA capability. A bus unit is a processing unit attached to the bus in a distributed processor network. It may include one or more host processors for example. Master DMA capability is simply the ability of a bus unit to access the main storage of a bus unit with slave DMA without interrupting the processor in the bus unit. The host is not able to access the controller main storage to obtain the work request nor the associated data. The controller, when a process within it generates a work request, will then place the work request and the associated data into storage in the host which is managed by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller. Management responsibility includes the right to read and write the remote storage as desired. The host may also read the remote storage and write responses in controller specified locations. The controller then sends an Opstart message to the host. The Opstart message is a bus message which indicates that there is a work request to be acted on by a server process and identifies the location in remote storage (actually local storage to the server) where a control block is. The control block identifies the remote storage location of the work request and associated data. The host then transfers the information into storage it manages for a server process to access. The host writes the response into the storage indicated in the control block." Hammer '954 at col. 2, ll. 36–63. "In FIG. 1 a high level view of a distributed process environment is indicated generally at 10. A processor A indicated at 12 is coupled by a physical path indicated by a line 14 to a processor B indicated at 16. Processor A is indicated as having a process A indicated at 18 and a Process B indicated at 19 residing therein. A storage area 20 is associated with process A and process B as represented by lines 21 and 22 respectively to provide the process control of and access to data storage." Hammer '954 at col. 4, ll. 46–54. "Processes, or executing programs within the processors need to communicate with each other. In processors of differing configurations, or in the same processor as it changes over time, two processes that communicate may be in different relative locations and may have different physical paths between them." Hammer '954 at col. 4, ll. 61–66. "An inter process communication facility (IPCF) is provided within processor A and processor B at 30 and 32 respectively to accommodate inter process communication which is location transparent to the communicating processes. IPCF 30 is coupled to process A in processor A as represented by a line 34 and to process B as represented by a line 36. Lines 34 and 36 represent interfaces between process A and process B to the IPCF 30. These interfaces permit communication between process A and process B provided appropriate data paths are established. IPCF 30 is also coupled through a transport mechanism 40 in processor B to IPCF 32. IPCF 32 is in turn coupled as represented by interface lines 42 and 44 to process C and process D. These interfaces with the IPCFs and the transport mechanisms permit establishment of communication between all the processes indicated, without process knowledge of the location of the process it is communicating with. The transport mechanisms 38 and 40 preferably comprise a plurality of transport mechanisms such as local transport mechanisms for use when process A and process B or process C and process D communicate within a single processor. If processor A and processor B reside in the same machine, a bus transport mechanism is used to facilitate communication between processes on processor A and

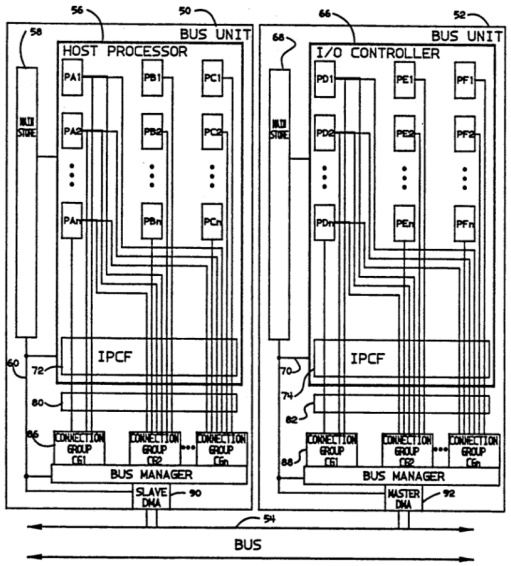
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processor B. For intermachine communication, a communication protocol such as SNA is suitable for use." Hammer '954 at col. 4, l. 67 – col. 5, l. 26. "The IPCF to the process attempting to communicate chooses the transport mechanism for the communication. The communicating processes need not be aware of the mechanism used. The process attempting to communicate supplies the name of the target process, as it is known to the process attempting to communicate, to the IPCF which uses an appropriate directory service to locate it. The IPCF then selects the appropriate transport mechanism and uses system-supplied services to set up the connection between the processes in a standard manner. IPCF can be used by all levels of processes, from applications to basic system services such as a paging manager." Hammer '954 at col. 5, ll. 37–49. "Communication between processes is in terms of sending and receiving messages over a connection between them as established by IPCF. The messages contain work requests and/or data. Relative to a particular work request, a process assumes the role of a requestor or a server. The requestor initiates a work request by sending a request to a server that carries it out. Requests contain a work request (a command and its parameters) and optionally some data. Both the request and the data are of variable length." Hammer '954 at col. 5, l. 61 – col. 6, l. 2. "In FIG. 2, two bus units, 50 and 52 are coupled by a physical path such as an I/O (input and output) bus 54. In one preferred embodiment, bus unit 50 comprises a host processing system with a processor 56 having a main storage 58 coupled thereto by a line 60. Host processor 56 executes a plurality of programs, from accounting programs to operating system programs. An executing instance of a program is referred to as a process. Several processes PA1-PAn, PB1-PBn and PC1-PCn are shown in host processor 56." Hammer '954 at col. 6, ll. 11–20. "Bus unit 52, in one preferred embodiment comprises an I/O processor 66 having a main storage 68 coupled by a line 70. I/O processor 66 also has a number of processes PD1-PDn, PE1-PEn and PF1-PFn which may be executing. While a large number of processes are shown in each of the processors 56 and 66, there may be situations where a processor will have only one process. In further embodiments, processor 66 is a peer processor in a network of processors. Processes communicate with each other, regardless of the locations of two communicating processes through IPCF 72 in bus unit 50 and IPCF 74 in bus unit 52. Each pair of communicating processes is logically connected through IPCF by logical connections indicated by lines 80 in bus unit 50 and 82 in bus unit 52." Hammer '954 at col. 6, ll. 21–35. "The establishment of connections is described in further detail in the above incorporated patent. In essence, an IPCF-Open verb establishes an IPCF connection between two processes. The connection is unique for each

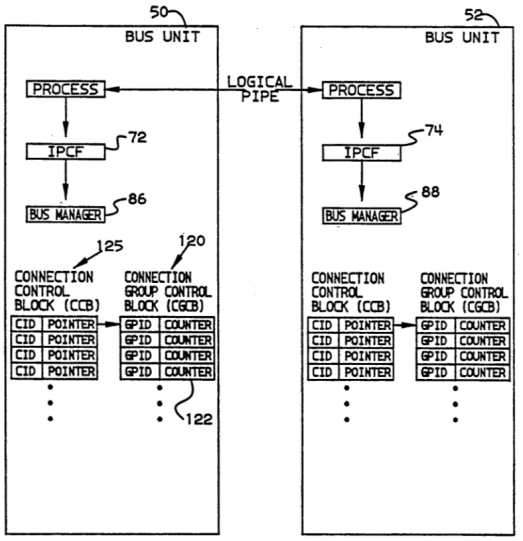
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	pair of processes communicating. Thus, each of lines 80 match to one of lines 82. Since there may be more I/O processors or host processors than shown in FIG. 2, more lines 80 than lines 82 would signify connections to processes in processors not shown. The Open verb, when issued by a process desiring to start a communication causes IPCF to create a logical connection between the process that issued the Open verb and the target process identified by the Open verb. The target of the Open verb is identified through an entity name. The Open verb establishes the connection to a new or already executing instance of a program based on the entity name supplied on the Open verb." Hammer '954 at col. 6, ll. 36–52. "The bus managers 86 and 88 manage the connections, and issue messages to control the flow of work requests on the bus 54. Hardware supporting the bus indicated at 90 and 92 provide arbitration to the bus for each of the bus units, DMA capability to another bus unit's main storage, and a queueing of messages from the respective bus managers until control of the bus has been obtained to permit transmission of messages and data. The bus hardware also controls queueing of messages from the bus units in main storage queues." Hammer '954 at col. 7, ll. 5–14. "The establishment of a connection between a process in host processor 56 and a process in I/O processor also 66 results in bus manager 88 completing the connection and assigning its processes connection to one of multiple connection groups CG1, CG2 . . . CGn. Thus, for each process in the system comprising the bus units, that has a connection to another process, the bus manager of the processor where the target process is located assigns its connection to a group, and allocates resources to the connection group to be used to serve work requests on the connection assigned to that group." Hammer '954 at col. 10, ll. 46–56. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications

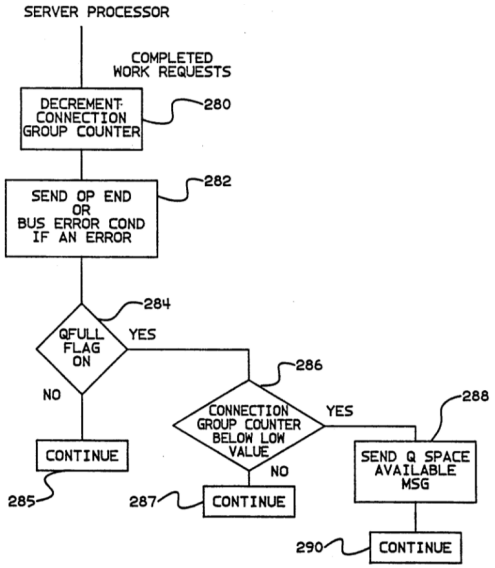
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "In FIG. 12 an interchange of messages in response to a full queue for a connection group in a target bus unit is shown. The target bus unit is labeled as the server, and the source bus unit is labeled as the requestor. Messages sent and received by the server are shown on the server side of the Figure because that is where the flow is controlled. Arrows between the requestor and server point to the bus unit which received each message. The sequence of messages is indicated in numerical order. The events are now listed, in the order they appear in FIG. 12:" Hammer '954 at col. 12, ll. 29–39. "In the preferred embodiment of FIG. 2, the bus hardware 90 and 92 of bus units 50 and 52 are labeled as having direct memory access (DMA) capability. This capability is a standard hardware capability that exists most hardware today. Master DMA capability allows the bus unit which has it to directly access the main storage of a bus unit with slave DMA capability without interrupting the processor in the slave DMA bus unit. Hence, the details of its operation will not be described, as they are not necessary for a full understanding of the invention. Bus unit 50 has a slave DMA capability 90. In the preferred embodiment, bus unit 50 is a host processor. Slave DMA capability permits a bus unit to let other bus units access its main store 58 without interrupting the host processor 56. Thus, line 60, which gives host processor 56 access to main store 58 is also connected to bus manager 86, and to slave DMA hardware to permit 90 access directly to main store 58. This in turn permits another another bus unit, such as bus unit 52 which has master DMA hardware 92 capability to access main store 58 of bus unit 50 without interrupting host processor 56. A bus unit with only slave DMA capability cannot directly access the main store of any other bus unit, while a unit with only master DMA capability cannot enable any other bus unit trying to directly access its main store." Hammer '954 at col. 13, ll. 13–39. "In normal flow, where it is assured that each bus unit has full DMA capability, bus manager 88 would receive information from IPCF verbs from IPCF 74, which indicates that a process desires to send a work request. Bus manager 88 would then alert bus manager 86 that it has work to do by sending an OPSTART message. The OPSTART bus message format is shown in FIG. 13, and has sufficient information for bus manager 86 in server bus unit 50 to move a Request/Response Control Block (RRCB) that specifies control information and data addresses from bus unit 52 main storage 68 into main storage 58 in bus unit 50, if it could. Further detail of the RRCB is shown in FIG. 14 and 15. Bus manager 86, could then alert the intended

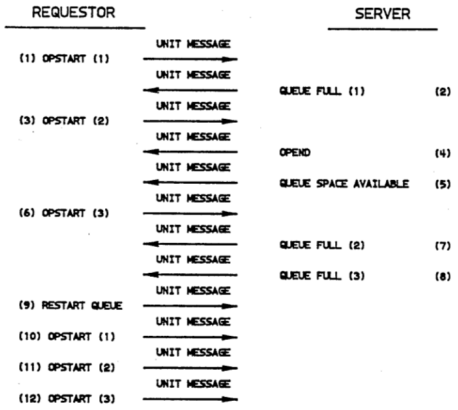
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	process in host processor 56 through IPCF 72 that there is a pending work request, which would be queued by the process. The process would then execute the request, which would require data movement between the bus units. The copy of the RRCB, that would now be in main storage 58 would be used by the bus manager 86 to control the transfer of data between the bus units. The requested operation would be signalled complete by an OPEND (see FIG. 16) message being sent to bus manager 88, which would alert the requesting process through IPCF 74." Hammer '954 at col. 13, l. 49 – col. 14, l. 5. "The problem with the above scenario is that as implemented in FIG. 2, bus hardware 90 cannot directly access main store 68. Even if bus hardware 90 had master DMA capability, bus hardware would have to have slave DMA capability as well. The problem is solved by using a storage list control block and several new bus messages to give management of buffers in main store 58 of host bus unit 50 to the bus manager 88 of bus unit 52. This permits the bus manager 88 in the requestor to transfer data related to the request to the buffers in main store 58 of the server, and then the server to transfer the data from the buffers into storage usable by the server process, in accordance with the normal flow of work. Thus, data flow appears normal to IPCF 72. The RRCB is used to indicate, as usual, where the data is that the server must access. The requestor bus manager 88 simply ensures that the data is located in buffers in main store 58. A more detailed description of the RRCB and messages will now be given." Hammer '954 at col. 14, ll. 6–24. "A RRCB is shown in FIGS. 14 and 15. It is used to identify a work request, and the data associated with it. The RRCB is a control block that is used by the bus manager in the requestor bus unit and the bus manager in the server bus unit to control the movement of data between the bus units. Information in the RRCB is used for the physical DMA process. The contents of the RRCB are preferably read only. The contents are not altered or modified by the bus manager in the server bus unit." Hammer '954 at col. 14, ll. 25–34. "As discussed above, the normal flow of work requests is from the host processor 56 to the I/O processor 66. They include work such as reading and writing data to secondary storage devices coupled to the I/O processor 66, or initiating communications through the I/O processor 66. Having an I/O processor 66 with master DMA capability and a host processor 56 with slave DMA capability is ideally suited to this relationship. The server being the I/O processor drives the transfer of data without interrupting the host." Hammer '954 at col. 18, ll. 44–53.
[1d] at least one second processing element in communication with said first processing element;	Hammer '954 teaches "at least one second processing element in communication with said first processing element," including at least the following exemplary citations.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	For example, Hammer '954 teaches at least one second processing element in communication with said first processing element, by disclosing, as non–limiting examples, a host, a server processor, a bus manager of the server, a Request/Response Control Block (RRCB) of a server, or an inter process communication facility (IPCF) of a server. A host processor, a server processor, a bus manager of a server, or a bus manager of a requestor, the second processing element is in communication with the first processing element. *See, e.g.:* "The flow of work requests in a server driven process to process communication environment is described. Logical connections between processes and bus managers interfacing bus units to an I/O bus are assigned to connection groups for management by the bus managers. Each bus unit has its own connection groups for the logical connections. Bus unit resources are assigned to each connection group based on performance factors, and a series of bus unit messages are used to control the flow of work so that a group which has no more resources will not accept further work requests. The originator of the work requests will resequence rejected work requests and resend them when the connection group has freed up resources. A further mechanism is provided to facilitate work consistent with the server driven architecture when bus units do not have adequate DMA capabilities. Two ways of reversing control of transfer of work requests and data so that the server need not have master DMA capability are presented. Management of storage in a remote processor is used to transfer work and its associated data into storage accessible by a bus unit with slave DMA capability. The slave DMA bus unit then transfers the information into storage is manages. In another way of reversing the flow, a bus unit message is used to make the original server a requestor. The bus unit message contains information which varies the request sent by the requestor. In this manner, the server, which was the original requestor transfers information using its master DMA capability flow." Hammer '954 at Abstract.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Hammer '954 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “In one preferred embodiment, the requester bus unit is a storage controller having master DMA capability, and the server bus unit is a host having slave DMA capability. A bus unit is a processing unit attached to the bus in a distributed processor network. It may include one or more host processors for example. Master DMA capability is simply the ability of a bus unit to access the main storage of a bus unit with slave DMA without interrupting the processor in the bus unit. The host is not able to access the controller main storage to obtain the work request nor the associated data. The controller, when a process within it generates a work request, will then place the work request and the associated data into storage in the host which is managed by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller. Management responsibility includes the right to read and write the remote storage as desired. The host may also read the remote storage and write responses in controller specified locations. The controller then sends an Opstart message to the host. The Opstart message is a bus message which indicates that there is a work request to be acted on by a server process and identifies the location in remote storage (actually local storage to the server) where a control block is. The control block identifies the remote storage location of the work request and associated data. The host then transfers the information into storage it manages for a server process to access. The host writes the response into the storage indicated in the control block." Hammer '954 at col. 2, ll. 36–63. "In FIG. 1 a high level view of a distributed process environment is indicated generally at 10. A processor A indicated at 12 is coupled by a physical path indicated by a line 14 to a processor B indicated at 16. Processor A is indicated as having a process A indicated at 18 and a Process B indicated at 19 residing therein. A storage area 20 is associated with process A and process B as represented by lines 21 and 22 respectively to provide the process control of and access to data storage." Hammer '954 at col. 4, ll. 46–54. "Processes, or executing programs within the processors need to communicate with each other. In processors of differing configurations, or in the same processor as it changes over time, two processes that communicate may be in different relative locations and may have different physical paths between them." Hammer '954 at col. 4, ll. 61–66. "An inter process communication facility (IPCF) is provided within processor A and processor B at 30 and 32 respectively to accommodate inter process communication which is location transparent to the communicating processes. IPCF 30 is coupled to process A in processor A as represented by a line 34 and to process B as represented by a line 36. Lines 34 and 36 represent interfaces between process A and process B to the IPCF 30. These interfaces permit communication between process A and process B provided appropriate data paths are established. IPCF 30 is also coupled through a transport mechanism 40 in processor B to IPCF 32. IPCF 32 is in turn coupled as represented by interface lines 42 and 44 to process C and process D. These interfaces with the IPCFs and the transport mechanisms permit establishment of communication between all the processes indicated, without process knowledge of the location of the process it is communicating with. The transport mechanisms 38 and 40 preferably comprise a plurality of transport mechanisms such as local transport mechanisms for use when process A and process B or process C and process D communicate within a single processor. If processor A and processor B reside in the same machine, a bus transport mechanism is used to facilitate communication between processes on processor A and

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processor B. For intermachine communication, a communication protocol such as SNA is suitable for use." Hammer '954 at col. 4, l. 67 – col. 5, l. 26. "The IPCF to the process attempting to communicate chooses the transport mechanism for the communication. The communicating processes need not be aware of the mechanism used. The process attempting to communicate supplies the name of the target process, as it is known to the process attempting to communicate, to the IPCF which uses an appropriate directory service to locate it. The IPCF then selects the appropriate transport mechanism and uses system-supplied services to set up the connection between the processes in a standard manner. IPCF can be used by all levels of processes, from applications to basic system services such as a paging manager." Hammer '954 at col. 5, ll. 37–49. "Communication between processes is in terms of sending and receiving messages over a connection between them as established by IPCF. The messages contain work requests and/or data. Relative to a particular work request, a process assumes the role of a requestor or a server. The requestor initiates a work request by sending a request to a server that carries it out. Requests contain a work request (a command and its parameters) and optionally some data. Both the request and the data are of variable length." Hammer '954 at col. 5, l. 61 – col. 6, l. 2. "In FIG. 2, two bus units, 50 and 52 are coupled by a physical path such as an I/O (input and output) bus 54. In one preferred embodiment, bus unit 50 comprises a host processing system with a processor 56 having a main storage 58 coupled thereto by a line 60. Host processor 56 executes a plurality of programs, from accounting programs to operating system programs. An executing instance of a program is referred to as a process. Several processes PA1-PAn, PB1-PBn and PC1-PCn are shown in host processor 56." Hammer '954 at col. 6, ll. 11–20. "Bus unit 52, in one preferred embodiment comprises an I/O processor 66 having a main storage 68 coupled by a line 70. I/O processor 66 also has a number of processes PD1-PDn, PE1-PEn and PF1-PFn which may be executing. While a large number of processes are shown in each of the processors 56 and 66, there may be situations where a processor will have only one process. In further embodiments, processor 66 is a peer processor in a network of processors. Processes communicate with each other, regardless of the locations of two communicating processes through IPCF 72 in bus unit 50 and IPCF 74 in bus unit 52. Each pair of communicating processes is logically connected through IPCF by logical connections indicated by lines 80 in bus unit 50 and 82 in bus unit 52." Hammer '954 at col. 6, ll. 21–35. "The establishment of connections is described in further detail in the above incorporated patent. In essence, an IPCF-Open verb establishes an IPCF connection between two processes. The connection is unique for each

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	pair of processes communicating. Thus, each of lines 80 match to one of lines 82. Since there may be more I/O processors or host processors than shown in FIG. 2, more lines 80 than lines 82 would signify connections to processes in processors not shown. The Open verb, when issued by a process desiring to start a communication causes IPCF to create a logical connection between the process that issued the Open verb and the target process identified by the Open verb. The target of the Open verb is identified through an entity name. The Open verb establishes the connection to a new or already executing instance of a program based on the entity name supplied on the Open verb." Hammer '954 at col. 6, ll. 36–52. "The bus managers 86 and 88 manage the connections, and issue messages to control the flow of work requests on the bus 54. Hardware supporting the bus indicated at 90 and 92 provide arbitration to the bus for each of the bus units, DMA capability to another bus unit's main storage, and a queueing of messages from the respective bus managers until control of the bus has been obtained to permit transmission of messages and data. The bus hardware also controls queueing of messages from the bus units in main storage queues." Hammer '954 at col. 7, ll. 5–14. "Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "The establishment of a connection between a process in host processor 56 and a process in I/O processor also 66 results in bus manager 88 completing the connection and assigning its processes connection to one of multiple connection groups CG1, CG2 . . . CGn. Thus, for each process in the system comprising the bus units, that has a connection to another process, the bus manager of the processor where the target process is located assigns its connection to a group, and allocates resources to the connection group to be used to serve work requests on the connection assigned to that group." Hammer '954 at col. 10, ll. 46–56. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "In FIG. 12 an interchange of messages in response to a full queue for a connection group in a target bus unit is shown. The target bus unit is labeled as the server, and the source bus unit is labeled as the requestor. Messages sent and received by the server are shown on the server side of the Figure because that is where the flow is controlled. Arrows between the requestor and server point to the bus unit which received each

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	message. The sequence of messages is indicated in numerical order. The events are now listed, in the order they appear in FIG. 12:" Hammer '954 at col. 12, ll. 29–39. "A RRCB is shown in FIGS. 14 and 15. It is used to identify a work request, and the data associated with it. The RRCB is a control block that is used by the bus manager in the requestor bus unit and the bus manager in the server bus unit to control the movement of data between the bus units. Information in the RRCB is used for the physical DMA process. The contents of the RRCB are preferably read only. The contents are not altered or modified by the bus manager in the server bus unit." Hammer '954 at col. 14, ll. 25–34. "As discussed above, the normal flow of work requests is from the host processor 56 to the I/O processor 66. They include work such as reading and writing data to secondary storage devices coupled to the I/O processor 66, or initiating communications through the I/O processor 66. Having an I/O processor 66 with master DMA capability and a host processor 56 with slave DMA capability is ideally suited to this relationship. The server being the I/O processor drives the transfer of data without interrupting the host." Hammer '954 at col. 18, ll. 44–53.
[1e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Hammer '954 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations. For example, Hammer '954 teaches a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element, by disclosing, as non–limiting examples, a connection group counter. The connection group counter stores a value of indicative space and is accessible by both the first and second processing elements discussed above. Plaintiffs have cited the use of data structures in memory storing a value indicative of available space, e.g., qstate, as allegedly satisfying limitation [1e]. *See* Am. Compl. at ¶62; '596 Infringement Contentions at 100–137. To the extent that Plaintiffs argue that Hammer '954's connection group counter is in memory, Hammer '954 discloses or renders obvious claim limitation [1e] under Plaintiffs' apparent interpretation of the claim. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

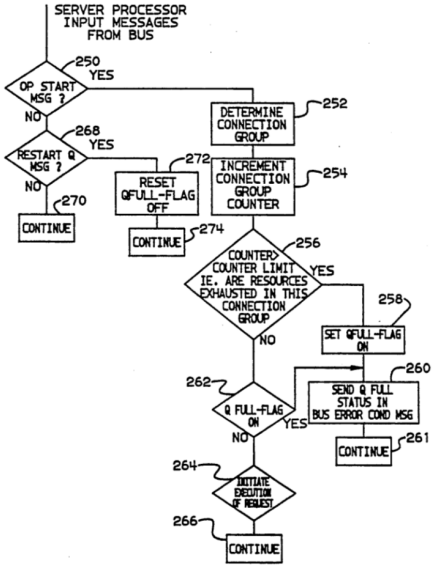


FIG. 7

Hammer ’954 at Figure 7.

“A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35.

“FIG. 4 is a block flow diagram showing a requesting bus unit responding to a Queue Full status for work requests destined for a connection group which does not have sufficient resource to accept more work requests.

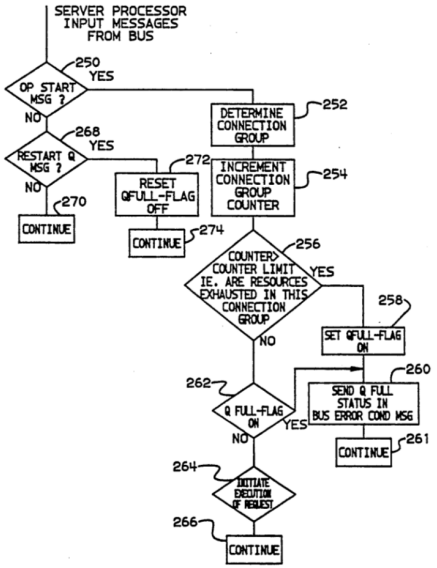
FIG. 5 is a further flow diagram of the requesting bus unit of FIG. 4 for responding to a Queue Full status.

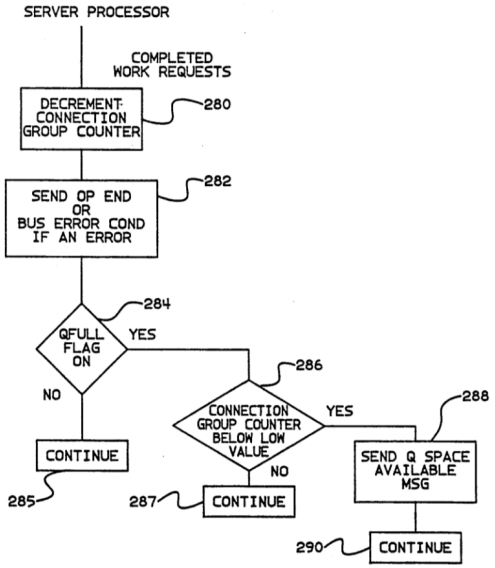
FIG. 6 is a block diagram of queues used by the requesting bus unit of FIG. 4 for responding to a Queue Full status.” Hammer ’954 at col. 3, ll. 23–32.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor. 6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message.

IPR2025-00223
Concurrent Ventures LLC, XtreamEdge Inc. EX2032

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5.
[1f] computer storage storing instructions, which when executed by said at least one second processing element:	Hammer '954 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. For example, Hammer '954 teaches computer storage storing instructions that are executed by said at least one second processing element, by disclosing, as non–limiting examples, main storage of the bus units. A host processor, a server processor, a bus manager of a server, or a bus manager of a requestor executes instructions those instructions stored in the computer storage. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

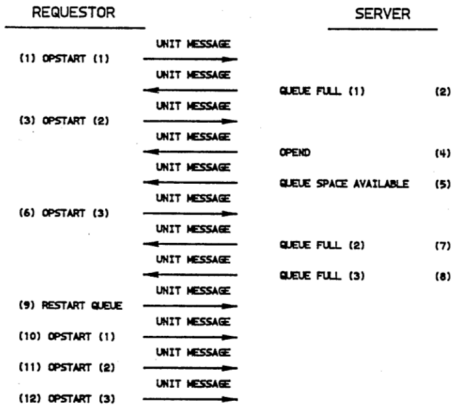


FIG. 12

Hammer ’954 at Figure 12.

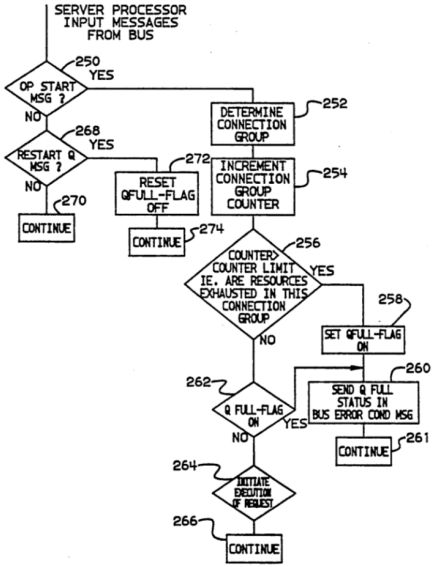
“A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35.

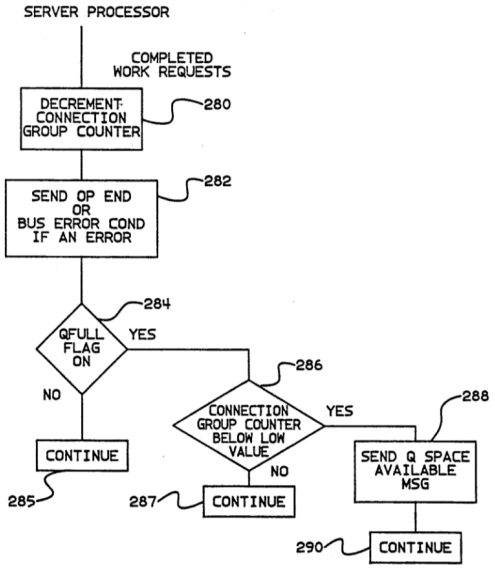
“Back in the flow of FIG. 4, when a Queue Space Available message is received by the requestor processor at 168, the Q SPACE FLAG is set on at 170, and the waiting to complete queue is checked for any entries after the one that QUEUE FULL was received for in the identified connection group. If queue full has not been returned for all requests waiting to complete, the bus unit continues to wait for the next communication at 174.” Hammer ’954 at col. 9, ll. 10–18.

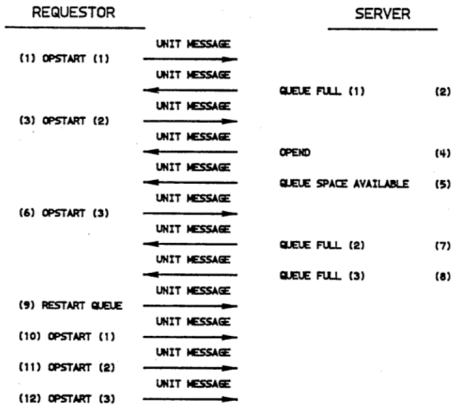
“If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor. 6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message. 7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5.
[1g] accesses said reservation register and reads said stored value;	Hammer '954 teaches "accesses said reservation register and reads said stored value," including at least the following exemplary citations. For example, Hammer '954 teaches that the second processing element executes instructions that causes it to access said reservation register and reads said stored value, by disclosing, as non–limiting examples, a host or server processor checking the queue space state, buffer counter, and/or checking Q space flag. A host or server processor accesses and reads the stored value in the reservation register. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

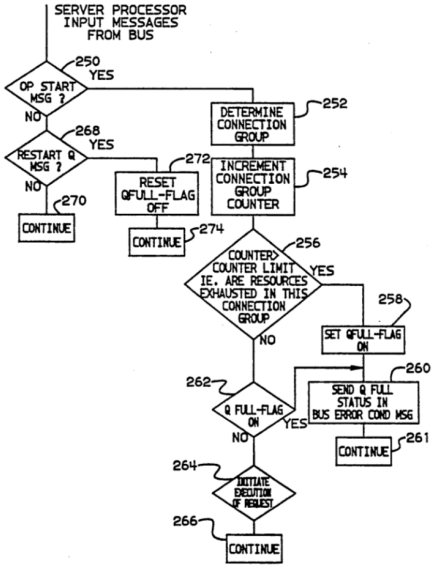
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

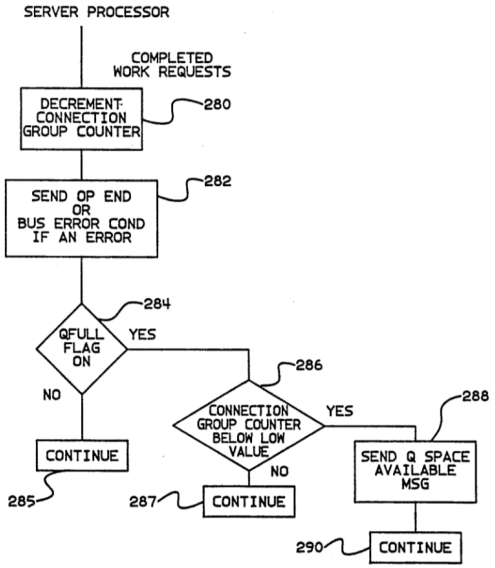
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “FIG. 4 is a block flow diagram showing a requesting bus unit responding to a Queue Full status for work requests destined for a connection group which does not have sufficient resource to accept more work requests. FIG. 5 is a further flow diagram of the requesting bus unit of FIG. 4 for responding to a Queue Full status. FIG. 6 is a block diagram of queues used by the requesting bus unit of FIG. 4 for responding to a Queue Full status.” Hammer ’954 at col. 3, ll. 23–32.

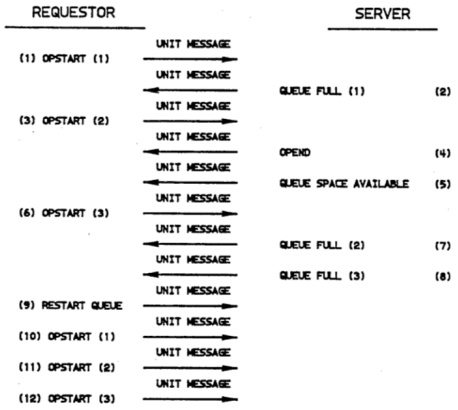
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Back in the flow of FIG. 4, when a Queue Space Available message is received by the requestor processor at 168, the Q SPACE FLAG is set on at 170, and the waiting to complete queue is checked for any entries after the one that QUEUE FULL was received for in the identified connection group. If queue full has not been returned for all requests waiting to complete, the bus unit continues to wait for the next communication at 174." Hammer '954 at col. 9, ll. 10–18. "If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message. 7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5.
[1h] determines when said read value indicates available space in said input queue for said first processing element to issue a command;	Hammer '954 teaches "determines when said read value indicates available space in said input queue for said first processing element to issue a command," including at least the following exemplary citations. For example, Hammer '954 teaches that the second processing element executes instructions that causes it to determine when said read value indicates available space in said input queue for said first processing element to issue a command, by disclosing, as non–limiting examples, a host processor, a server processor, a bus manager of a server, or a bus manager of a requestor checking the reservation register. A host processor, a server processor, a bus manager of a server, or a bus manager of a requestor determines if the stored value indicates available space in the input queue for the first processing element to issue a command. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “FIG. 4 is a block flow diagram showing a requesting bus unit responding to a Queue Full status for work requests destined for a connection group which does not have sufficient resource to accept more work requests. FIG. 5 is a further flow diagram of the requesting bus unit of FIG. 4 for responding to a Queue Full status. FIG. 6 is a block diagram of queues used by the requesting bus unit of FIG. 4 for responding to a Queue Full status.” Hammer ’954 at col. 3, ll. 23–32.

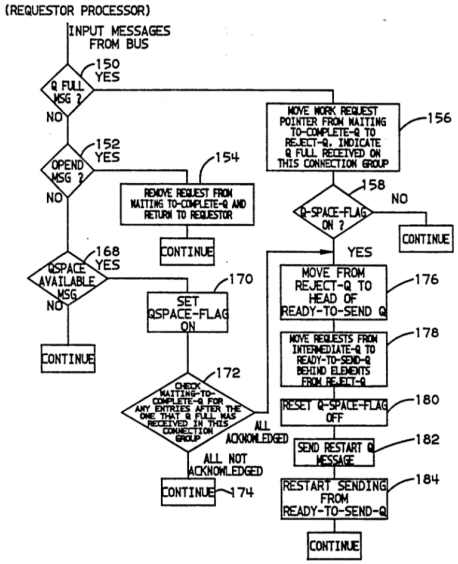
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Back in the flow of FIG. 4, when a Queue Space Available message is received by the requestor processor at 168, the Q SPACE FLAG is set on at 170, and the waiting to complete queue is checked for any entries after the one that QUEUE FULL was received for in the identified connection group. If queue full has not been returned for all requests waiting to complete, the bus unit continues to wait for the next communication at 174." Hammer '954 at col. 9, ll. 10–18. "If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it

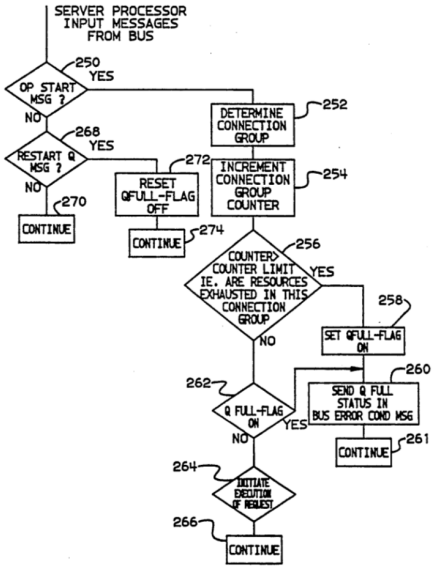
IPR2025-00223
Concurrent Ventures LLC, XtreamEdge Inc. EX2032

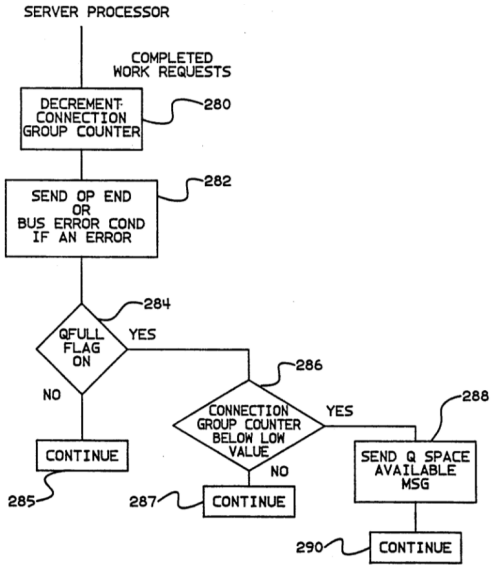
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor.

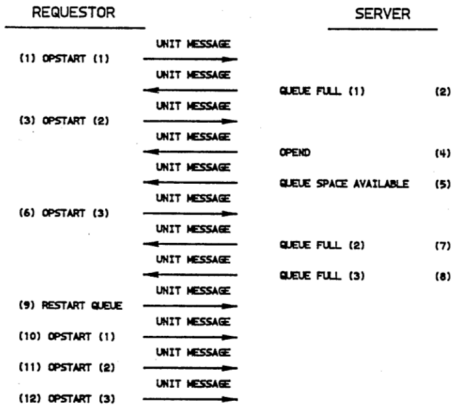
IPR2025-00223
Concurrent Ventures LLC, XtreamEdge Inc. EX2032

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message. 7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5.
[1i] notifies said first processing element to issue said command to said input queue; and	Hammer '954 teaches "notifies said first processing element to issue said command to said input queue," including at least the following exemplary citations. For example, Hammer '954 teaches that the second processing element executes instructions that causes it to notify said first processing element to issue said command to said input queue, by disclosing, as non–limiting examples, a second processing element (e.g., a host processor, a server processor, a bus manager of a server, or a bus manager of a requestor) sending a storage available message, or a queue space available message to the first processing element. The second processing element notifies the first processing element to issue the command to the input queue. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Hammer ’954 at Figure 4.

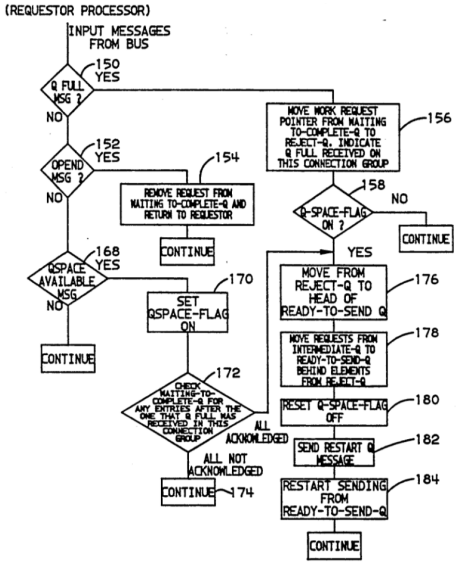
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

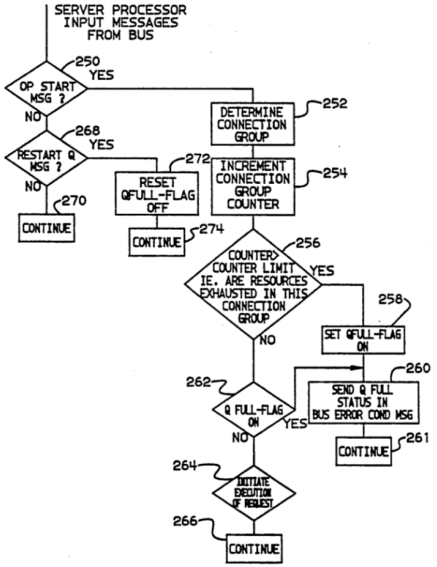
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

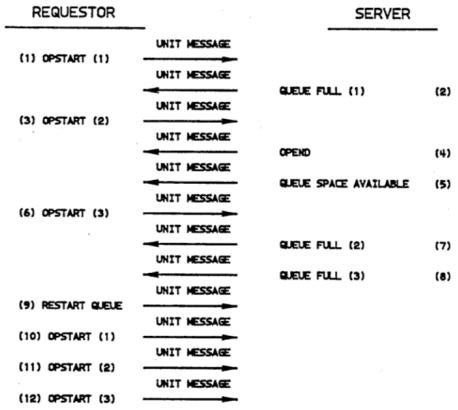
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “FIG. 4 is a block flow diagram showing a requesting bus unit responding to a Queue Full status for work requests destined for a connection group which does not have sufficient resource to accept more work requests. FIG. 5 is a further flow diagram of the requesting bus unit of FIG. 4 for responding to a Queue Full status. FIG. 6 is a block diagram of queues used by the requesting bus unit of FIG. 4 for responding to a Queue Full status.” Hammer ’954 at col. 3, ll. 23–32.

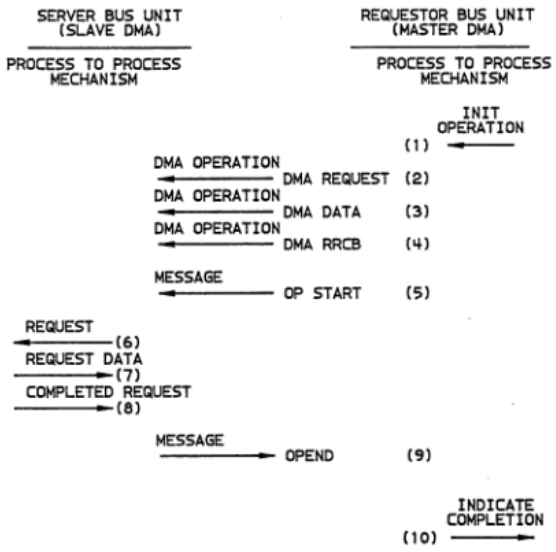
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Back in the flow of FIG. 4, when a Queue Space Available message is received by the requestor processor at 168, the Q SPACE FLAG is set on at 170, and the waiting to complete queue is checked for any entries after the one that QUEUE FULL was received for in the identified connection group. If queue full has not been returned for all requests waiting to complete, the bus unit continues to wait for the next communication at 174." Hammer '954 at col. 9, ll. 10–18. "If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor. 6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message. 7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1j] wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue.	Hammer ’954 teaches “wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue,” including at least the following exemplary citations. For example, Hammer ’954 teaches notifies said first processing element to issue said command to said input queue, by disclosing, as non–limiting examples, a firs processing element (e.g., a requestor processor or a I/O processor), after receiving the storage or queue available message, moving work requests to the input queue. The first processing element receives the notification, issues the command, and receives a corresponding response from the output queue. *See, e.g.:*  FIG. 4 Hammer ’954 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12.

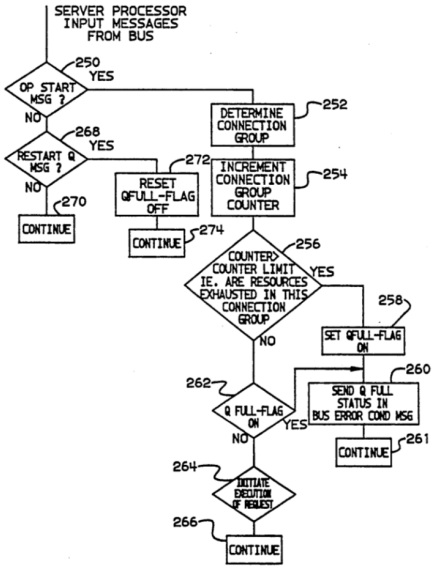
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 25 Hammer '954 at Fig. 25. "If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for

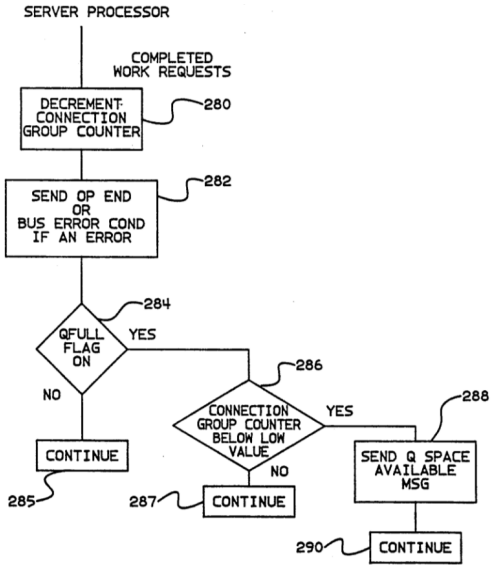
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "1. The requestor processor sends a message that initiates work, which is called an OPSTART(1) message, (defined later in this description) to the server processor. 2. The server processor has recognized a QUEUE FULL condition for the connection group that the OPSTART message was intended for, and returns an error message with QUEUE FULL status. 3. Because of the asynchronous nature of the bus units, the requestor is not yet aware of the QUEUE FULL status, and it sends a second message, OPSTART(2) to the server. 4. The server will have completed some previous work request, and will signal that to the requestor by sending an OPEND message." Hammer '954 at col. 12, ll. 40–54. "10, 11, and 12. The requestor processor resends the OPSTART messages in the correct order. When work is completed by a server, an OPEND message is sent to the requesting bus unit to indicate that work has completed. Its format is described later." Hammer '954 at col. 13, ll. 6–10. "1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
2. The system of claim 1, wherein said input queue is a first-in-first-out (FIFO) buffer.	

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
3. The system of claim 1, wherein said system is part of a storage area network (SAN) controller.	Hammer '954 teaches "[t]he system of claim 1, wherein said system is part of a storage area network (SAN) controller," including at least the following exemplary citations. For example, Hammer '954 teaches wherein said system is part of a storage area network (SAN) controller, by disclosing, as non–limiting examples, bus units operating over remote storage, remote buffer, and group of tape drive devices. The bus units and the first and second processing elements are used in a SAN controller. *See, e.g.:* "One benefit of the present invention is that management responsibility for portions of storage may be transferred between different bus units in a dynamic manner using predefined bus messages. The remote storage so managed has use in masking non-symmetrical hardware without the expense of added hardware. It provides symmetry at the process to process interface, even when the underlying hardware is not symmetric. A further benefit of the present invention is that remote storage can be used to balance performance of bus units in a system. If one bus unit is performance bound due to lack of main storage, remote buffers can be used to increase its performance. Thus, resources can be shifted from a bus unit with extra resources to a bus unit with too few resources to increase overall system performance." Hammer '954 at col. 2, l. 64 – col. 3, l. 11. "One example of a device which requires a large amount of main storage resource is a tape drive device. In order to keep a tape device operating in streaming mode, its most efficient mode of operation, a large main storage buffer is required. If it is desired to attach multiple tape drives, and each requires large amounts of storage to buffer data when it is operating in a streaming mode, the assignment of sufficient resources to each tape drive would leave few resources available to be allocated to direct access storage devices. By combining the tape drives into one connection group, and allocating resources sufficient for only one or two to operate simultaneously, many more resources are available for direct access storage devices. A guaranteed level of support is provided for tape drive devices, and also for direct access storage devices. Since it is unlikely that five tape devices will be operating at the same time, there is no reduction in overall service." Hammer '954 at col. 7, ll. 36–53.
4. The system of claim 1, wherein said system is part of a network attached storage (NAS) controller.	Hammer '954 teaches "[t]he system of claim 1, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. For example, Hammer '954 teaches wherein said system is part of a storage area network (SAN) controller, by disclosing, as non–limiting examples, bus units operating over a network of remote storage, remote buffer,

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	and group of tape drive devices. The bus units and the first and second processing elements are used in a NAS controller. *See, e.g.:* "One benefit of the present invention is that management responsibility for portions of storage may be transferred between different bus units in a dynamic manner using predefined bus messages. The remote storage so managed has use in masking non-symmetrical hardware without the expense of added hardware. It provides symmetry at the process to process interface, even when the underlying hardware is not symmetric. A further benefit of the present invention is that remote storage can be used to balance performance of bus units in a system. If one bus unit is performance bound due to lack of main storage, remote buffers can be used to increase its performance. Thus, resources can be shifted from a bus unit with extra resources to a bus unit with too few resources to increase overall system performance." Hammer '954 at col. 2, l. 64 – col. 3, l. 11. "One example of a device which requires a large amount of main storage resource is a tape drive device. In order to keep a tape device operating in streaming mode, its most efficient mode of operation, a large main storage buffer is required. If it is desired to attach multiple tape drives, and each requires large amounts of storage to buffer data when it is operating in a streaming mode, the assignment of sufficient resources to each tape drive would leave few resources available to be allocated to direct access storage devices. By combining the tape drives into one connection group, and allocating resources sufficient for only one or two to operate simultaneously, many more resources are available for direct access storage devices. A guaranteed level of support is provided for tape drive devices, and also for direct access storage devices. Since it is unlikely that five tape devices will be operating at the same time, there is no reduction in overall service." Hammer '954 at col. 7, ll. 36–53.
5. The system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Hammer '954 teaches "[t]he system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations. For example, Hammer '954 teaches wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element by disclosing, as non–limiting examples, incrementing connection group counter and decrementing connection group counter prior to said stored value to said at least one second processing element. The connection group counter decrements the stored value by a predefined value. *See, e.g.:*

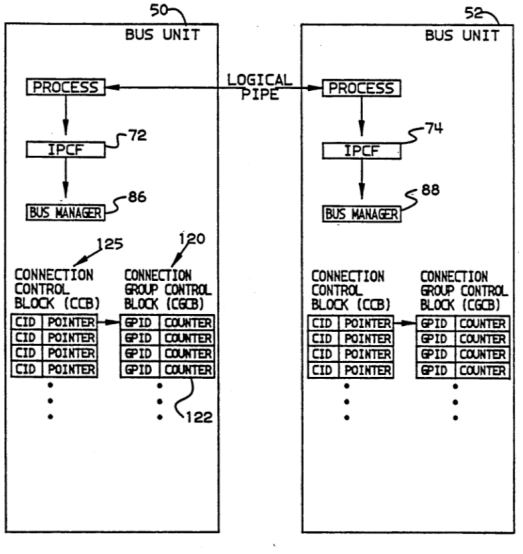
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

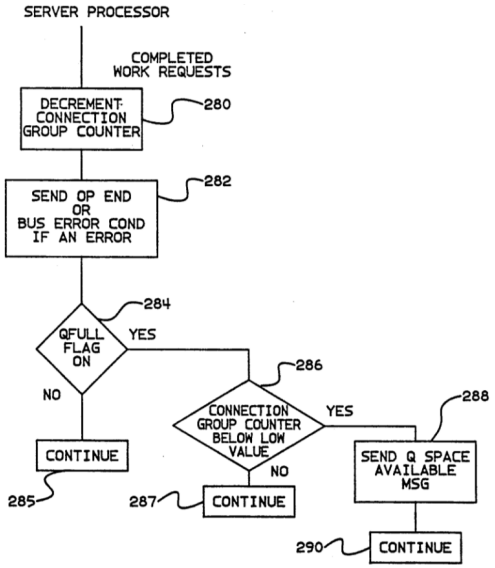
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8. “To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290.” Hammer ’954 at col. 10, ll. 4–26.

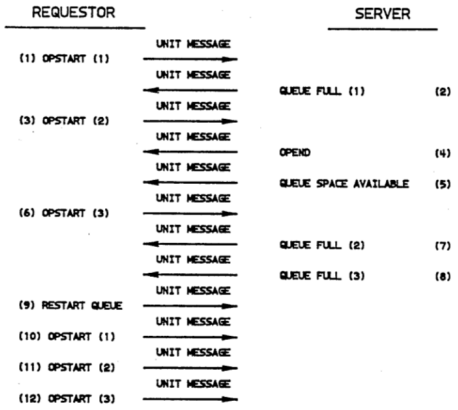
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
6. The system of claim 1, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Hammer ’954 teaches “[t]he system of claim 1, wherein at least one third processing element executes said command and notifies said output queue to issue said response,” including at least the following exemplary citations. For example, Hammer ’954 teaches wherein at least one third processing element executes said command and notifies said output queue to issue said response, by disclosing, as non–limiting examples, a server, or a server processor. The server, or server processor executes the command and notifies the output queue to issue a response. *See, e.g.:* “1. The requestor processor sends a message that initiates work, which is called an OPSTART(1) message, (defined later in this description) to the server processor. 2. The server processor has recognized a QUEUE FULL condition for the connection group that the OPSTART message was intended for, and returns an error message with QUEUE FULL status. 3. Because of the asynchronous nature of the bus units, the requestor is not yet aware of the QUEUE FULL status, and it sends a second message, OPSTART(2) to the server. 4. The server will have completed some previous work request, and will signal that to the requestor by sending an OPEND message.” Hammer ’954 at col. 12, ll. 40–54. “1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72.

IPR2025-00223
Concurrent Ventures LLC, XtreamEdge Inc. EX2032

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
[7pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Hammer '954 teaches "[a] system for dividing and synchronizing a processing task across a plurality of processing elements," including at least the following exemplary citations. *See* above at row [1pre].
[7a] an input queue implemented in hardware;	Hammer '954 teaches "an input queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1a].
[7b] an output queue implemented in hardware;	Hammer '954 teaches "an output queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1b].
[7c] a first processing element having access to said input queue and said output queue;	Hammer '954 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. *See* above at row [1c].

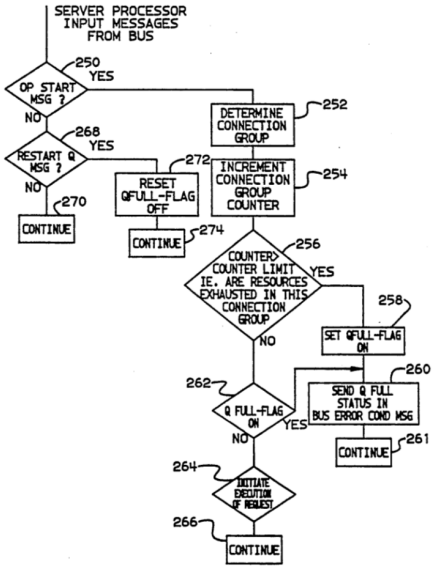
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7d] at least one second processing element in communication with said input queue;	Hammer ’954 teaches “at least one second processing element in communication with said input queue,” including at least the following exemplary citations. For example, Hammer ’954 teaches at least one second processing element in communication with said input queue, by disclosing, as non–limiting examples, a host processor, a server processor, a bus manager of the server, or a bus manager of a requestor. A host processor, a server processor, a bus manager of the server, or a bus manager of a requestor, the second processing element is in communication with the input queue. *See, e.g.:*  FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
[7e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Hammer '954 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations. *See* above at row [1e].
[7f] computer storage storing instructions, which when executed by said at least one second processing element:	Hammer '954 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. *See* above at row [1f].
[7g] receives a command from said first processing element;	Hammer '954 teaches "receives a command from said first processing element," including at least the following exemplary citations.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	For example, Hammer '954 teaches that the second processing element executes instructions that causes it to receive a command from said first processing element, by disclosing, as non–limiting examples, the second processing element (e.g., a host processor, a server processor, a bus manager of a server, or a bus manager of a requestor) receiving, for example, a work-initiation message, a signal, or storage/buffer management messages. The second processing element receives a command from the first processing element. *See, e.g.:*  FIG. 7 Hammer '954 at Figure 7.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

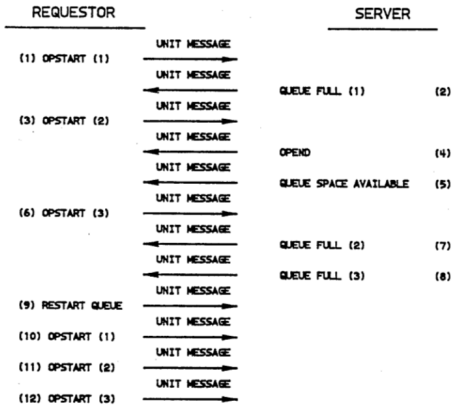


FIG. 12

Hammer ’954 at Figure 12.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness

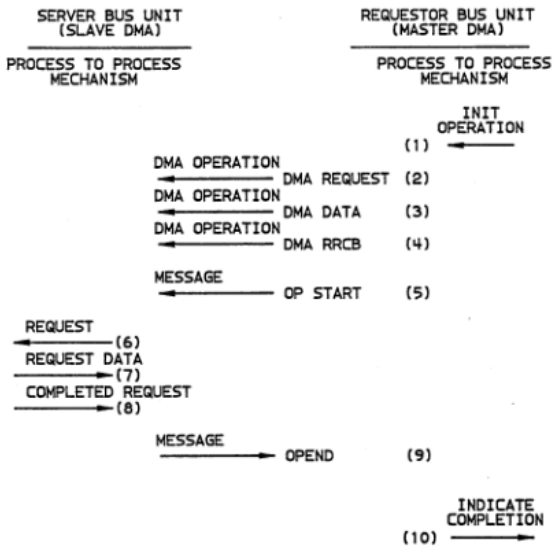


FIG. 25

Hammer ’954 at Fig. 25.

“A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35.

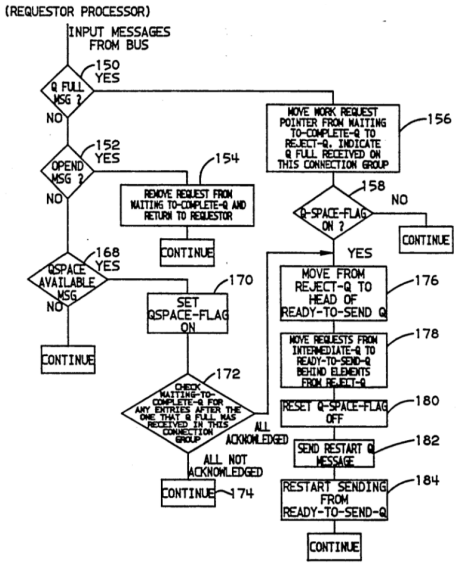
“Back in the flow of FIG. 4, when a Queue Space Available message is received by the requestor processor at 168, the Q SPACE FLAG is set on at 170, and the waiting to complete queue is checked for any entries after the one that QUEUE FULL was received for in the identified connection group. If queue full has not been returned for all requests waiting to complete, the bus unit continues to wait for the next communication at 174.” Hammer ’954 at col. 9, ll. 10–18.

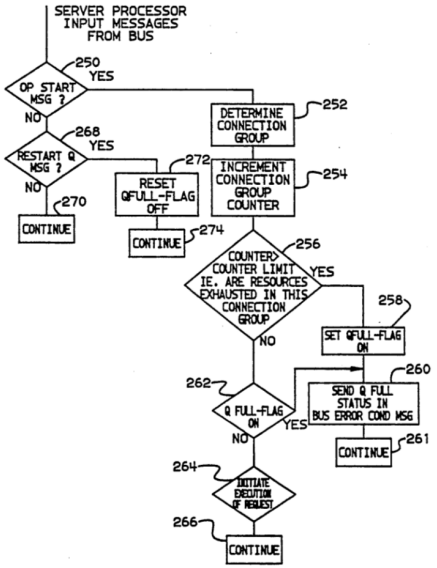
“If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue

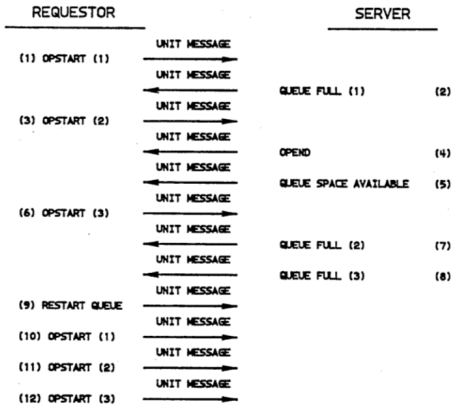
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184." Hammer '954 at col. 9, ll. 19–26. "In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "After sending a QUEUE FULL status to a source bus unit, the bus manager of the target bus unit which sent it monitors the appropriate connection group to determine when sufficient resources are available to serve the connection group. When the target bus unit has space available in that particular connection group, it sends a Queue Space Available message to the source bus unit. The Queue Space Available bus unit message is used to indicate to

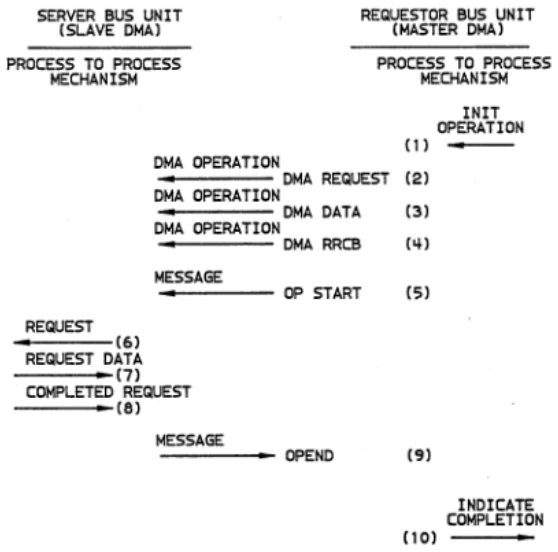
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the source bus manager that queue space is available. It is sent by a bus unit only after the bus unit has sent a QUEUE FULL status to the source bus unit. It indicates what connection group has queue space available. The format of the Queue Space Available message is shown in FIG. 10. There are four RESERVED fields, a MESSAGE ID field, and a GROUP field which uniquely defines the connection group for which there is queue space available. The source bus unit will receive the Queue Space Available message and will determine the number of work requests which were sent to that particular connection group on the target bus unit. Since communications occur asynchronously, and there may be internal queue delays in the hardware of the source bus unit, the source bus unit may have already sent more messages to initiate work." Hammer '954 at col. 11, l. 45 – col. 12, l. 2. "5. Since the server has completed some previous work request, the server will have resources available or queue space available, and will send a Queue Space Available message to the requestor. 6. The requestor has not recognized the Queue Full condition because of hardware queueing delays in the requestor, and will send an OPSTART(3) message. 7. The server must continue to return QUEUE FULL status to the requestor until the server receives a Restart Queue message. It therefore returns a QUEUE FULL(2) status in response to recognizing the OPSTART(2) message. 8. The server now recognizes the OPSTART(3) message and returns a QUEUE FULL(3) message. 9. The requestor now recognizes the QUEUE FULL status for OPSTART (1), (2) and (3), and Queue Space Available message and sends a Restart Queue message after determining which messages it must resend, and the order in which to send them." Hammer '954 at col. 12, l. 55 – col. 13, l. 5. "1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
[7h] accesses said reservation register and reads said stored value;	Hammer '954 teaches "accesses said reservation register and reads said stored value," including at least the following exemplary citations. *See* above at row [1g].
[7i] determines when said read value indicates available space in said input queue to issue said command;	Hammer '954 teaches "determines when said read value indicates available space in said input queue to issue said command," including at least the following exemplary citations. *See* above at row [1h].

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7j] issues said command to said input queue on behalf of said first processing element; and	Hammer ’954 teaches “issues said command to said input queue on behalf of said first processing element,” including at least the following exemplary citations. For example, Hammer ’954 teaches that the second processing element executes instructions that causes it to issue said command to said input queue on behalf of said first processing element, by disclosing, as non–limiting examples, the second processing element (e.g., a host processor, a server processor, a bus manager of a requestor, or a bus manager of a server), after receiving the request and data from the first processing element, transferring the request and data into the input queue. The second processing element issues the command to the input queue on behalf of the first processing element.  FIG. 4 Hammer ’954 at Figure 4.

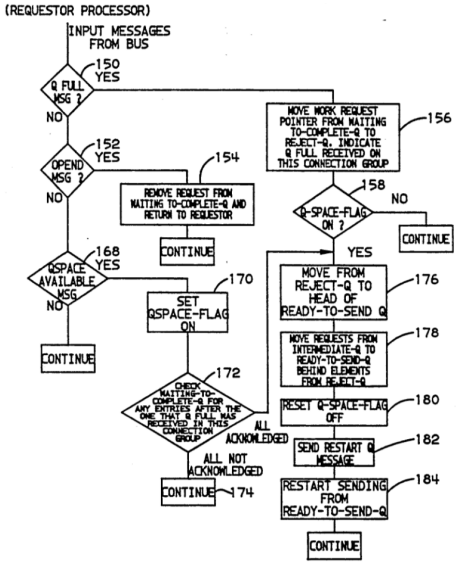
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Hammer ’954 at Figure 7.

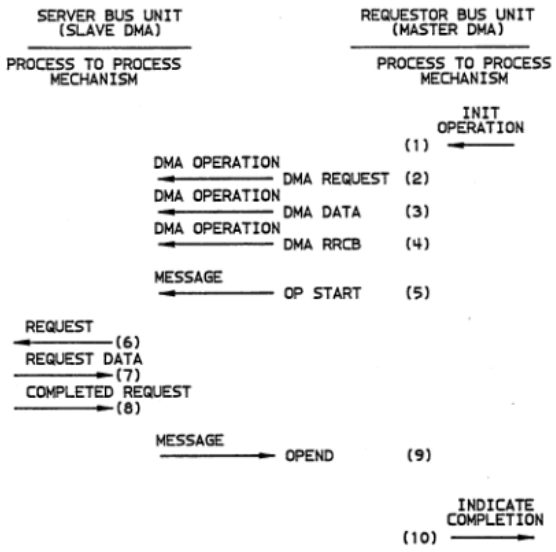
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 25 Hammer ’954 at Fig. 25. “If all are acknowledged at 172, the entries are moved from the reject-queue to the beginning of the ready to send queue at 176. Then, request entries are moved from the intermediate queue to the ready-to-send-queue behind elements from the reject queue at 178, and the Q SPACE FLAG is reset at 180. A Restart Queue message is sent at 182, and the bus manager begins sending from the ready to send queue at 184.” Hammer ’954 at col. 9, ll. 19–26. “In FIG. 6, the queues used by the above flow are shown, with the flow of queue entries corresponding to work requests shown by arrows. The ready to send queue 200 normally transfers entries by line 202 to the wait for response queue 210. If a QUEUE FULL status is received, entries are transferred from queue 210 to the reject queue 220 via a line 222. Meanwhile, as new work in queue 200 is being initiated, if a work request to a connection group already full is detected, it is transferred to the intermediate queue 230 via line 232. When all entries from the wait for response queue 210 which are intended for the full connection group have been transferred to the reject queue 220, and the Restart Queue message has been sent in response to a Queue Space Available message, the rejected requests are sent back to the top of the ready to send queue 200 via a line 234. Then, on line 236, other requests intended for the group which now has resources available for

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	further work are transferred to the ready to send queue 200 behind the previously rejected requests." Hammer '954 at col. 9, ll. 27–46. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "10, 11, and 12. The requestor processor resends the OPSTART messages in the correct order. When work is completed by a server, an OPEND message is sent to the requesting bus unit to indicate that work has completed. Its format is described later." Hammer '954 at col. 13, ll. 6–10. "1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage.

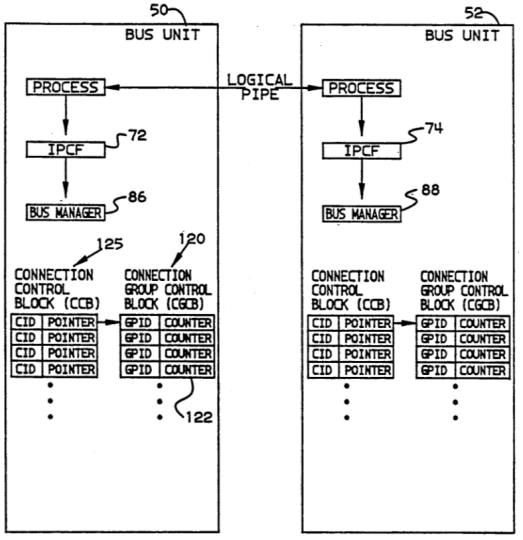
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
[7k] wherein said first processing element receives a response corresponding to said command from said output queue.	Hammer '954 teaches "wherein said first processing element receives a response corresponding to said command from said output queue," including at least the following exemplary citations. For example, Hammer '954 teaches wherein said first processing element receives a response corresponding to said command from said output queue, by disclosing, as non–limiting examples, a first processing element (e.g., a host processor, a server processor, a bus manager of a requestor, or a bus manager of a server) receiving responses (e.g., OPEND messages or completion indication) from the output queue. The first processing element receives a response from the output queue corresponding to the command. *See, e.g.:*

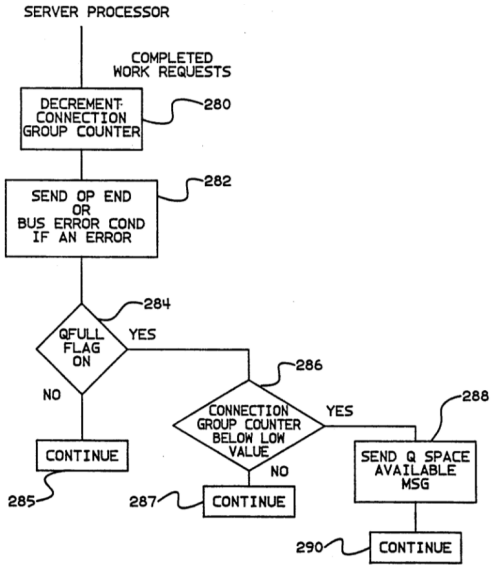
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Hammer ’954 at Figure 4.

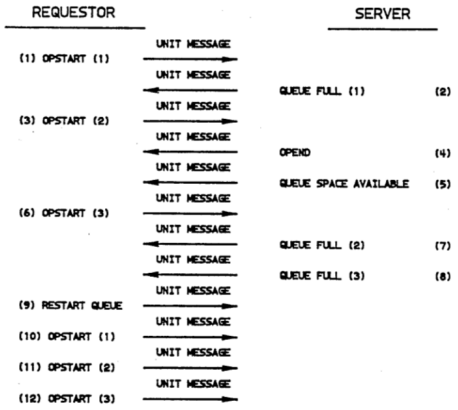
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 25 Hammer ’954 at Fig. 25. “1. The requestor processor sends a message that initiates work, which is called an OPSTART(1) message, (defined later in this description) to the server processor. 2. The server processor has recognized a QUEUE FULL condition for the connection group that the OPSTART message was intended for, and returns an error message with QUEUE FULL status. 3. Because of the asynchronous nature of the bus units, the requestor is not yet aware of the QUEUE FULL status, and it sends a second message, OPSTART(2) to the server. 4. The server will have completed some previous work request, and will signal that to the requestor by sending an OPEND message.” Hammer ’954 at col. 12, ll. 40–54. “1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
8. The system of claim 7, wherein said input queue is a first-in-first-out (FIFO) buffer.	
9. The system of claim 7, wherein said system is part of a storage area network (SAN) controller.	Hammer '954 teaches "[t]he system of claim 7, wherein said system is part of a storage area network (SAN) controller," including at least the following exemplary citations. *See* above at row 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
10. The system of claim 7, wherein said system is part of a network attached storage (NAS) controller.	Hammer '954 teaches "[t]he system of claim 7, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. *See* above at row 4.
11. The system of claim 7, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Hammer '954 teaches "[t]he system of claim 7, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations. *See* above at row 5.
12. The system of claim 7, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Hammer '954 teaches "[t]he system of claim 7, wherein at least one third processing element executes said command and notifies said output queue to issue said response," including at least the following exemplary citations. *See* above at row 6.
[13pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Hammer '954 teaches "[a] system for dividing and synchronizing a processing task across a plurality of processing elements," including at least the following exemplary citations. *See* above at row [1pre].
[13a] an input queue implemented in hardware;	Hammer '954 teaches "an input queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1a].
[13b] an output queue implemented in hardware;	Hammer '954 teaches "an output queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1b].

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[13c] a first processing element having access to said input queue and said output queue;	Hammer ’954 teaches “a first processing element having access to said input queue and said output queue,” including at least the following exemplary citations. *See* above at row [1c].
[13d] at least one second processing element in communication with said input queue and said output queue;	Hammer ’954 teaches “at least one second processing element in communication with said input queue and said output queue,” including at least the following exemplary citations. For example, Hammer ’954 teaches at least one second processing element in communication with said input queue and said output queue, by disclosing, as non–limiting examples, the second processing element (e.g., a host processor, a server processor, a bus manager of a requestor, or a bus manager of a server) having access and managing both the input and output queue to control submission of work requests and receipt of corresponding responses. A host processor, a server processor, a bus manager of a requestor, or a bus manager of a server, the second processing element is in communication with the input and output queue. *See, e.g.:*  FIG. 3 Hammer ’954 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Hammer ’954 at Figure 8.

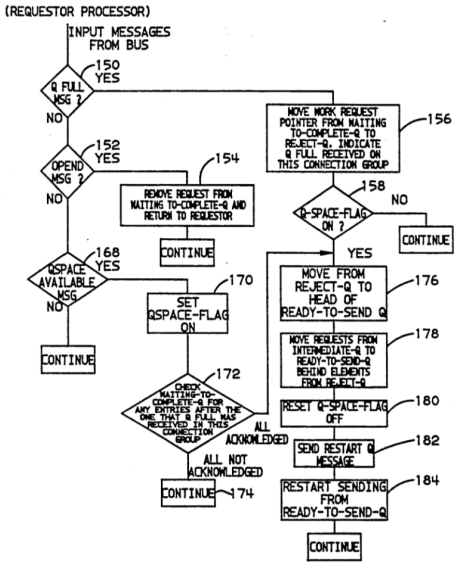
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 12 Hammer ’954 at Figure 12. “A requester bus unit uses a Storage List Request bus message to request management of remote storage in a server bus unit main storage. The Storage List Request contains fields which indicate the total amount of remote storage requested, and the length of buffers it desires. The server bus unit receives the message, and checks for buffers of at least the length indicated. It creates a Storage List Control Block which identifies the starting address and length of each buffer found and for which management responsibility is to be transferred. The server bus unit then sends a Storage List Available message to the requester bus unit which indicates if long enough buffers were found, and where to find the Storage List Control Block.” Hammer ’954 at col. 2, ll. 22–35. “In FIG. 2, two bus units, 50 and 52 are coupled by a physical path such as an I/O (input and output) bus 54. In one preferred embodiment, bus unit 50 comprises a host processing system with a processor 56 having a main storage 58 coupled thereto by a line 60. Host processor 56 executes a plurality of programs, from accounting programs to operating system programs. An executing instance of a program is referred to as a process. Several processes PA1-PAn, PB1-PBn and PC1-PCn are shown in host processor 56.” Hammer ’954 at col. 6, ll. 11–20.

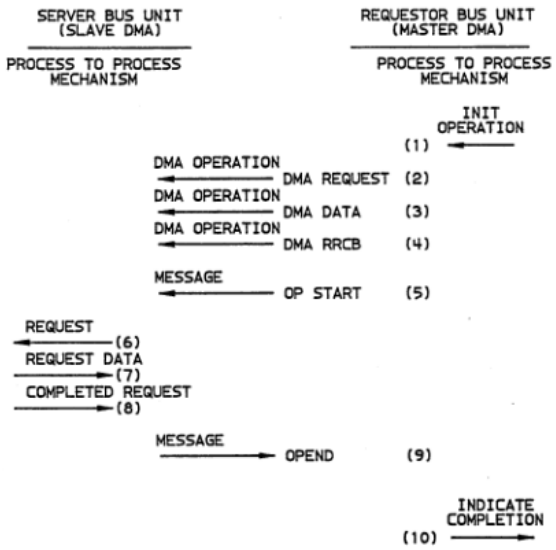
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Corresponding flow in the server bus unit is shown in FIG. 7 and 8. On receipt of an OPSTART message at 250 by a server bus unit, the connection group is determined at 252, and its counter is incremented at 254. A counter limit is used to define the number of work requests which a connection group may have outstanding at any one time. The value of the counter is compared to the counter limit at 256, and if the value is greater than the limit, a Q FULL FLAG is set at 258 and QUEUE FULL status is sent in a Bus Error Condition Message to the originating bus unit at 260, and processing continues at 261. In a further embodiment, actual resources available are monitored and compared to the resources required for a work request indicated by the OPSTART. If the counter is not greater than the limit, but the Q FULL FLAG is detected as on at 262, the flow returns to 260 and the QUEUE FULL status is sent in response to the OPSTART, otherwise, processing of the request continues at 264, and flow continues at 266. If the message received at 250 is not an OPSTART message, it is checked to see if it is a Restart Queue message at 268. If not, processing continues at 270. If it is a Restart Queue message, the Q FULL FLAG is reset to off at 272, and processing continues at 274." Hammer '954 at col. 9, l. 47 – col. 10, l. 3. "To clear up a queue full condition, the server processor must either allocate more resources, and increase the limit of outstanding requests, or complete work on at least one of the requests on a full connection. This flow is indicated in FIG. 8. When a work request is completed, the relevant connection group counter is decremented at 280. Either an OPEND, or a Bus Error Condition message is sent at 282. The Q FULL FLAG is then checked at 284, and if it is off, processing continues at 285. If the Q FULL FLAG was set on, the connection group counter is checked to see if it is below the limit at 286. This limit, referred to as a "low value" in block 286 need not be the same as the previously discussed limit. It may be desirable to establish an even lower limit to ensure that sufficient resources will be available for a number of requests. This may help to eliminate a constant sequence of QUEUE FULL flows where multiple requests may be sent quickly following restart of the requestors queue. If the counter is not below the low value, processing continues at 287. If the counter is below the low value at 286, a Queue Space Available message is sent at 288, and processing continues at 290." Hammer '954 at col. 10, ll. 4–26. "In the preferred embodiment of FIG. 2, the bus hardware 90 and 92 of bus units 50 and 52 are labeled as having direct memory access (DMA) capability. This capability is a standard hardware capability that exists most hardware today. Master DMA capability allows the bus unit which has it to directly access the main storage of a bus unit with slave DMA capability without interrupting the processor in the slave DMA bus unit. Hence, the details of its operation will not be described, as they are not necessary for a full understanding of the invention.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Bus unit 50 has a slave DMA capability 90. In the preferred embodiment, bus unit 50 is a host processor. Slave DMA capability permits a bus unit to let other bus units access its main store 58 without interrupting the host processor 56. Thus, line 60, which gives host processor 56 access to main store 58 is also connected to bus manager 86, and to slave DMA hardware to permit 90 access directly to main store 58. This in turn permits another another bus unit, such as bus unit 52 which has master DMA hardware 92 capability to access main store 58 of bus unit 50 without interrupting host processor 56. A bus unit with only slave DMA capability cannot directly access the main store of any other bus unit, while a unit with only master DMA capability cannot enable any other bus unit trying to directly access its main store." Hammer '954 at col. 13, ll. 13–39. "In normal flow, where it is assured that each bus unit has full DMA capability, bus manager 88 would receive information from IPCF verbs from IPCF 74, which indicates that a process desires to send a work request. Bus manager 88 would then alert bus manager 86 that it has work to do by sending an OPSTART message. The OPSTART bus message format is shown in FIG. 13, and has sufficient information for bus manager 86 in server bus unit 50 to move a Request/Response Control Block (RRCB) that specifies control information and data addresses from bus unit 52 main storage 68 into main storage 58 in bus unit 50, if it could. Further detail of the RRCB is shown in FIG. 14 and 15. Bus manager 86, could then alert the intended process in host processor 56 through IPCF 72 that there is a pending work request, which would be queued by the process. The process would then execute the request, which would require data movement between the bus units. The copy of the RRCB, that would now be in main storage 58 would be used by the bus manager 86 to control the transfer of data between the bus units. The requested operation would be signalled complete by an OPEND (see FIG. 16) message being sent to bus manager 88, which would alert the requesting process through IPCF 74." Hammer '954 at col. 13, l. 49 – col. 14, l. 5. "The problem with the above scenario is that as implemented in FIG. 2, bus hardware 90 cannot directly access main store 68. Even if bus hardware 90 had master DMA capability, bus hardware would have to have slave DMA capability as well. The problem is solved by using a storage list control block and several new bus messages to give management of buffers in main store 58 of host bus unit 50 to the bus manager 88 of bus unit 52. This permits the bus manager 88 in the requestor to transfer data related to the request to the buffers in main store 58 of the server, and then the server to transfer the data from the buffers into storage usable by the server process, in accordance with the normal flow of work. Thus, data flow appears normal to IPCF 72. The RRCB is used to indicate, as usual, where the data is that the server must access. The requestor bus manager 88 simply ensures that the data is located in buffers in main store 58. A more detailed description of the RRCB and messages will now be given." Hammer '954 at col. 14, ll. 6–24.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50. 3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
[13e] a reservation register implemented in hardware storing a value indicative of available space	Hammer '954 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	*See* above at row [1e].
[13f] computer storage storing instructions, which when executed by said at least one second processing element:	Hammer ’954 teaches computer storage storing instructions that are executed by said at least one second processing element including at least the following exemplary citations. *See* above at row [1f].
[13g] receives a command from said first processing element;	Hammer ’954 teaches “receives a command from said first processing element,” including at least the following exemplary citations. *See* above at row [7g].
[13h] accesses said reservation register and reads said stored value;	Hammer ’954 teaches “accesses said reservation register and reads said stored value,” including at least the following exemplary citations. *See* above at row [1g].
[13i] determines when said read value indicates available space in said input queue to issue said command;	Hammer ’954 teaches “determines when said read value indicates available space in said input queue to issue said command,” including at least the following exemplary citations. *See* above at row [1h].
[13j] issues said command to said input queue on behalf of said first processing element; and	Hammer ’954 teaches “issues said command to said input queue on behalf of said first processing element,” including at least the following exemplary citations. *See* above at row [7j].
[13k] receives a response corresponding to said command from said output queue.	Hammer ’954 teaches “receives a response corresponding to said command from said output queue,” including at least the following exemplary citations. For example, Hammer ’954 teaches that the second processing element executes instructions that causes it to receive a response corresponding to said command from said output queue, by disclosing, as non–limiting

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	examples, a second processing element (e.g., a host processor, a server processor, a bus manager of a requestor, or a bus manager of a server) receiving responses (e.g., OPEND messages or completion indication) from the output queue that correspond to the said command. The second processing element receives a corresponding response from the output queue. *See, e.g.:*  FIG. 4 Hammer ’954 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 25 Hammer ’954 at Fig. 25. “1. The requestor processor sends a message that initiates work, which is called an OPSTART(1) message, (defined later in this description) to the server processor. 2. The server processor has recognized a QUEUE FULL condition for the connection group that the OPSTART message was intended for, and returns an error message with QUEUE FULL status. 3. Because of the asynchronous nature of the bus units, the requestor is not yet aware of the QUEUE FULL status, and it sends a second message, OPSTART(2) to the server. 4. The server will have completed some previous work request, and will signal that to the requestor by sending an OPEND message.” Hammer ’954 at col. 12, ll. 40–54. “1. The requestor process issues a request at the process to process verb interface to the process to process mechanism, IPCF 74 in FIG. 2. 2. The bus manager 88 will obtain a buffer of adequate size in the host processor main store 58 if it already does not have one, and initiates a master DMA operation through bus hardware 92 to move the request into remote storage in the server storage bus unit 50.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	3. The request bus manager 88 DMAs data into the buffer. 4. The bus manager 88 will then DMA the RRCB into the buffers in the server's storage. The RRCB at this point uses addresses in the servers storage of the request and data that were DMA'd into the remote storage in the server bus unit in steps 2 and 3. 5. The bus manager 88 in the requestor processor 66 sends an Opstart bus unit message to the server processor indicating that the RRCB and data are in the server storage. At data associated with the process using the process to process mechanism has been DMA'd into the server's storage. 6. The request is passed to the server process by the process to process mechanism 72. 7. The server process requests the necessary data, which is in the storage local to the server. The bus manager 88 still has control of the buffers but bus manager 86 has access to these buffers while processing the Opstart message. The bus manager 86 of the server process transfers the data into an area of main store 58 which is accessible to the server process. 8. When the server process has completed the requested operation, the process to process mechanism 72 is notified by use of IPCF verbs. 9. The bus manager 86 in the server processor 56 issues an Opend message with the status information. This informs the bus manager 88 in the requestor processor 66 of any response put in the buffers by bus manager 86. After retrieving such response, the bus manager 88 frees the buffers for further use and indicates to the requesting process that the requested operation has been completed. 10. The process to process mechanism 74 then notifies the requesting process that the operation has been completed." Hammer '954 at col. 19, ll. 1–44.
14. The system of claim 13, wherein said input queue is a first-in-first-out (FIFO) buffer.	
15. The system of claim 13, wherein said system is either part of a storage area network (SAN) controller.	Hammer '954 teaches "[t]he system of claim 13, wherein said system is either part of a storage area network (SAN) controller," including at least the following exemplary citations. *See* above at row 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
16. The system of claim 13, wherein said system is part of a network attached storage (NAS) controller.	Hammer '954 teaches "[t]he system of claim 13, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. *See* above at row 4.
17. The system of claim 13, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Hammer '954 teaches "[t]he system of claim 13, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations. *See* above at row 5.
18. The system of claim 13, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Hammer '954 teaches "[t]he system of claim 13, wherein at least one third processing element executes said command and notifies said output queue to issue said response," including at least the following exemplary citations. *See* above at row 6.