Revised Appendix A-1: Written Description Support for Revised, Proposed, Substitute Claims

Claim I.D.	Claim Limitation	Exemplary Support in Provisional App. No. 61/870,815	Exemplary Support in Nonprovisional App. No. 14/468,836
29 pre, pt. 1	29. (proposed substitute for claim 15) ~~15.~~ A method for fetching a fresh content over the Internet, requested by a first device, identified in the Internet by a first identifier, from a first server identified in the Internet by a second identifier via a group of multiple tunnel devices, each identified in the Internet by an associated group device identifier, …	See, e.g., Fig. 58 and associated description on Page 2 : "Figure 58 is a diagram of how Tunnels can be used by a Client. The "Tunnel" 16 is used by the Client 2 to fetch data from the Data Server 5, when such data is not found in the acceleration network's elements Caches, or the data is found in the caches but does not make sense for the Client 2 to fetch it for practical reasons, such as because the time to fetch it from the acceleration network's caches would be longer than to fetch the data directly from the Data Server 5. In such a case where it is beneficial for the Client 2 to fetch the data directly from the Data Server 5, the Client 2 may use one or more Tunnels 16 to carry out the request to the Data Server 5 on its behalf." See, e.g., Figure 55 and associated description on Page 1: "Figure 55 describes how an element takes on one of the roles that it can assume: 5506, 5508 and 5510 show that when the system is starting up, a connection is established with the Acceleration Server 1, and a list	See, e.g., original claim 134 See, e.g., Figs. 5, 5a, and 5b; see also Figs. 11, 11a, 11b E.g., Page 81: "A method is disclosed for fetching a content over the Internet by a first device identified in the Internet by a first identifier, from a first server identified in the Internet by a second identifier [v]ia a group of multiple devices, each identified in the Internet by an associated group device identifier, …" E.g., Abstract: "A method for fetching a content from a web server to a client device is disclosed, using tunnel devices serving as intermediate devices…. The tunnel devices in turn fetch the slices from the data server, and send the slices to the client device, where the content is reconstructed from the received slices… The partition into slices may be overlapping or non-overlapping, and the same slice (or the whole content) may be fetched via multiple tunnel devices."

		of Tunnels for this communication device is requested. Once received, in 5510 the device establishes connections (e.g. TCP connect) to each of the Tunnels in the list, so that once messages are needed to be sent to/from these Tunnels, it can be done quickly due to this pre-connection." See, e.g., Fig. 50 and associated description on Page 1: "In the network of figure 50, there is a new type of element called a Tunnel." E.g., Page 2: "…the Tunnels receives a data request from a Client, including the data server to get the request from and the associated other data required for such a data request, and fetches this from the data server and provides back to the Client[.]" See, e.g., Fig. 69 and associated description on Page 7: "Figure 69 describes the PATH_DIRECT- i.e. how to retrieve the data from the Tunnels 16 when this option is chosen." See, e.g., Fig. 69 and associated description on Page 7: "In 6902, the requested data is split in to ranges, such that there is one range per participating Tunnel. It is possible to simply choose ranges of equal size to implement this. A	E.g., Page 82: "In one example, the second device may consist of, comprise, or be part of, a tunnel device, such as the tunnel device #1." E.g., Page 118: "A flowchart 100 relating to the client device #1 31 a when employing three tunnel devices is shown in FIG. 10, based on the flowchart 60 described above. Upon receiving a list of available tunnel in a 'Receive Tunnels List' step 62 b from the Acceleration server 32, the client device #1 31 a selects multiple tunnels from the received list, rather than selecting a single tunnel as described in the 'Select Tunnel' step 62 c described above." E.g., Page 114: "The request is received at the tunnel device #1 33 a at a 'Request Received' state 54 c, corresponding to a 'Receive Content Request' 73 b in the flowchart 70). In response, the tunnel device 33 a sends a 'Content Request' message 56 h to the data server #1 22 a (corresponding to a 'Send Request To Server' step 73 c), requesting the content that was requested by the client device #1 31 a." E.g., Page 118: "In turn the data server #1 replies and sends the content in a 'Send Content' message 111 c (corresponding to

		possibly more efficient implementation is to provide each Tunnel a range that is proportional to that Tunnel's ability to provide the data (for example, based on that Tunnel's upstream and downstream traffic to the Client, as measured in previous communications between them)." See also, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client."	the 'Send Content' message 56 i in the timing chart 50) to the requesting tunnel device #1 33 a, which in turn forward the fetched content to the asking client device #1 31 a using a 'Send Content' message 111 d (corresponding to the 'Send Content' message 56 j in the timing chart 50)." See also, e.g., Page 73: "A client device may be a first device identified in the Internet by a first identifier, executing a method for fetching over the Internet a first content, identified by a first content, from a second server identified in the Internet by a third identifier, via a second device identified in the Internet by a second identifier, using a first server."

29 pre, pt. 2	the method comprising the step of partitioning the fresh content into a plurality of content slices, each content slice containing at least part of the fresh content, and identified using a content slice identifier, and for each of the content slices, comprising the steps of:	E.g., Page 2: "The Client 2 will also determine how much of the data it will request from each of the Tunnels 16 based on their speed, and thus assign a 'range' of the data (for example for a data of 1200 bytes where two agents A and Bare selected, a range assignment may be that A is assigned the bytes between byte 0 and byte 856, and B is assigned bytes 857 to 1200). The Client 2 will then send the request to all these Tunnels 16 along with the requested range for them to fetch from the Data Server." E.g., Page 2: "…where a range is also provided to the Tunnel, and the Tunnel only fetches this range of data from the data server." See, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along with the ***segment*** that this Tunnel needs to fetch from the Data Server." (emphasis added) E.g., Page 23: "Data servers typically limit the number of active connections that a host can keep open with them concurrently. This limitation is detrimental in various scenarios. One such scenario as an example is where a communications device with a web	See, e.g., original claim 134 See, e.g., Fig. 10, 11, 11a, 11b and 21 E.g., Page 81: "…the method comprising the step of partitioning the content into a plurality of content slices, each content slice containing at least part of the content, and identified using a content slice identifier, and for each of the content slices." E.g., Page 125: "The content requested by the client device #1 31 a may be partitioned into multiple parts or 'slices'. Any number of slices may be used…" E.g., Page 138: "The content stored in a data server, such as the data server #1 22a, which may be requested by a client device such as the client device #1 201a, may be partitioned into multiple parts or 'slices'. Any number of slices may be used. The slicing may be in a bit, nibble (4-bits), byte (8-bits), word (multiple bytes), character, string, or file level. For example, in a case wherein the content includes 240 bytes designated byte #1 to byte #240, using a byte level partitioning into two slices results in a first slice (slice #1) including byte #1 to byte #120, and a second slice (slice #2) including byte #121 to byte #240.

		browser needs to load ***many URLs from a Data Server in order to display a web page***…" (emphasis added) See, e.g., Fig. 136 and associated description on Page 23: "Figure 136 describes such a system. In it, the Client 2 would like to open more connections with the Data Server 5 than the Data Server 5 will allow (***where each of these connections is loading one or more requests, where each request is a resource, a range of a resource, or multiple resources***). The Client then communicates with participating Tunnels 4 (i.e. network devices on which software is installed for this purpose), and requesting from each of these Tunnels to load ***several of these requests***, such that the each of the participating elements (the Client and the Tunnels from which it requested the loads) does not open more connections vs. the Data Server 5 that the Data Server 5 allows." (emphasis added) See, e.g., Fig. 73 See also, e.g., Page 21: "a system that comprises of a software module, a database, where upon getting a uri as an input, the system looks up the domain of the URL within the database, retrieves the schema of the URL and can identify the	E.g., Page 138: "In one example, the content itself is made of inherent or identifiable parts or segments, and the partition may make use of these parts. In one example, the content may be a website content composed of multiple webpages, and thus the partition may be such that each slice includes one (or few) webpages. Further, the partitioning may be sequential or non-sequential in the content."

		identity of the video, and the offset to view within the video according to the schema[.]" See also, e.g., Page 2: "…wherein a Client chooses one or more Tunnels and a range per Tunnel, sends the request to these Tunnels, and receives back the information from the Tunnel that the Tunnel received from the Data Server[.]" See also, e.g., Page 2: "This approach has multiple benefits, including the ability to fetch multiple segments of this data in parallel (which may result in a faster load time) and also by creating multiple paths to the Data Server 5 (which may result in a more effective use of the network's resources)." See also Pages 33-34 ("Microchunks" section)	
29a	(a) selecting a tunnel device from the group so long as the tunnel device has an idle resource that satisfies a criterion;	E.g., Page 2: "Therefore, once the Client 2 has determined that it is not worthwhile to use the caches within the various acceleration systems' network elements, it will determine how many Tunnels 16 to use for this request based on the amount of data to fetch, and the speed of the available Tunnels 16. The Client 2 will also determine how much of the data it will request from each of the Tunnels 16	See, e.g., original claim 134 See, e.g., Figs. 5, 5a, 5b, and 6 E.g., Page 81: "The method may comprise the steps of selecting a device from the group…" E.g., Page 113: "Based on pre-set criteria, a tunnel device (or multiple tunnel

		based on their speed, and thus assign a 'range' of the data (for example for a data of 1200 bytes where two agents A and Bare selected, a range assignment may be that A is assigned the bytes between byte 0 and byte 856, and B is assigned bytes 857 to 1200). The Client 2 will then send the request to all these Tunnels 16 along with the requested range for them to fetch from the Data Server." E.g., Page 7: "In 6901 the Client 2 chooses which of the Tunnels 16 that were allocated to it by the Acceleration Server 1 it will use for this current transaction." See also, e.g., Page 7: "…alternative tunnel can be chosen by several methods; a random tunnel from the list of Tunnels for this client, or a Tunnel that is currently not in use, or a Tunnel that the Client has the highest BW/RTI ratio to, etc." See also, e.g., Page 5: "[I]n a system where one or more network elements (tunnels) are used to load portions of data on behalf of another network element (a client), a method of choosing tunnels for the clients, where the network elements that are the closest to the Client are chosen as tunnels[.]"	devices) is selected by the client device #1 31 a in a 'Tunnel Select' step 53 c (corresponding to a 'Select Tunnel' step 62 c in the flowchart 60). For example, the tunnel device #1 33 a may be selected." E.g., Page 76: "A distinct device may be selected for each content slice." See also, e.g., Page 118: "Upon receiving a list of available tunnel in a 'Receive Tunnels List' step 62 b from the Acceleration server 32, the client device #1 31 a selects multiple tunnels from the received list, rather than selecting a single tunnel as described in the 'Select Tunnel' step 62 c described above." See also, e.g., Page 118: "In the described example, three distinct tunnel devices are selected from the list, such as the tunnel device #1 33 a (as before), the tunnel device #2 33 b, and the tunnel device #3 33 c. The client device 31 a executes three pre-connection processes in a 'Pre-Connection Tunnel #1' step 64 a, a 'Pre-Connection Tunnel #2' step 64 b, and a 'Pre-Connection Tunnel #3' step 64 c (each corresponding to the 'Pre-connection' flow chart 64 above), followed by a 'Content Fetch Tunnel #1' step 65 a, a 'Content Fetch Tunnel #2'

		E.g., Page 26: "… the Client receives a list of Agents to use, uses them for a specific transaction, and when the network is idle, sends back to the Acceleration Server information about these agents for the server to use in recommending these agents in the future." E.g., Page 26: "… where the information is sent only when the network is idle." E.g., Page 26: "… where the acceleration server uses this information by providing back this information to the client to use in transactions with the agent." E.g., Page 27: "… where the acceleration server uses this information to decide which agent to recommend in future requests by clients." See, e.g., Fig. 700 and associated description on page 84: "… In a system monitoring the idleness of a computing device, the following sensors may be included in a list of devices to be monitored for idleness: Bits being sent or received to/from communication device , Storage device read or write, Temperature changes in any of the device's temperature gauges, CPU busy over certain threshold, Camera senses motion, Light sensor senses changes in light intensity or	step 65 b, and a 'Content Fetch Tunnel #3' step 65 c, respectively (each corresponding to the 'Content Fetch' flow chart 65 above)." See also, e.g., Page 112: "Then the tunnel device #1 33 a sign in with the acceleration server 32 in a step 'Sign-in as Tunnel' 71 b, which corresponds to a message 'Sign In' 56 a in the chart 50. The message comprises the device functionality as 'tunnel', and the device 33 a identification on the Internet 113, such as its IP address (for example 125.12.67.0)." E.g., Page 97: "A client device may receive and use a list of agent devices for a specific transaction, and when the network is idle, the client device may update the acceleration server regarding the used agent devices, allowing the acceleration server to later recommend or use these agent devices. The updating of the acceleration server may include the IP address of each of the agent devices, the communication session information such as RTT, BW and speed, the ports used in the communication sessions, the latency and speed for each of the connection phases, and whether the required file or data was stored in each of the agent devices. The acceleration server may use

		structure, Accelerometer that senses movement or orientation change, HD accelerometer that senses movement, GPS sensor that senses movement in location or hight." See, e.g., Fig. 214 and associated description on page 40: ""When network elements use BW & RTT tables, they could benefit from 'teaching' each other about the information they already have about BW & RTI between them and other network elements. Figure 214 is a flowchart showing a method by which network elements can share BW & RTI information so that a host that wishes to communicate with a network element that it has not recently communicate with can assess the BW & RTT to it by learning from other network elements that have recently communicated with that network element." E.g., Page 40: "The method starts with scheduling an update of other network elements in 21404. The scheduling can be set to either happen at constant time intervals, or when the local host is idle, or when both the local host and other network element to be updated are idle, or when finished communicating with another network element, etc."	this information to decide which agent devices to recommend and include in a future agent devices list requested by client devices." E.g., Page 113: "In order to make the communication between a client device and a tunnel device faster and more efficient, a pre-connection phase is defined, where a preparation for communication such as a TCP connection is established, allowing for quick data transfer afterwards. The pre-connection phase starts at a 'Start Pre-Connection' state 53 b in the chart 50, followed by the 'Request List' message 56 c (corresponding to the 'Request Tunnels List' step 62 in the flowchart 60), being part of the Pre-connection client flowchart 64, where the client 31 a requests the list of the available tunnels that may be used, from the acceleration server 32. The tunnel device #1 33 a at this point is idling in an 'IDLE' step 72 c shown in the flowchart 70, being part of the Pre-connection tunnel flowchart 72. In response to the client device 31 a request, the acceleration server 32 prepares in a step 'Prepare List' 52 b the list of current available tunnels, and sends the list as a 'Send List' message 56 d to the client device 31 a, which in turn receives the list

		See, e.g., Fig. 610 and associated description on page 69: "Figure 610 is a flowchart for how this can be done, by validating invalid (or soon to be invalid) cache entries while the communication device is idle." E.g., Page 69: "61002 checks if the communication device's resources are idle (or idle enough) to update entries in its cache without degrading system performance (i.e. checks that CPU is below a certain threshold for example of 30%, that the disk is not in use, that there is excess bandwidth that can be used, etc.)." See, e.g., Fig. 158 and associated description on page 30: "In 15808 the Acceleration Server receives a request from a Client that seeks to communicate with a Data Server, and the Client requests to receive a list of Agents through which to communicate. The Acceleration server then creates 15810 a list of Agents that are assigned to this particular Data Server according to its database. It then checks 15814 the number of Agents in this list. If there are under 5 (example implementation- can be other numbers as well) Agents assigned to this Data Server, then the Acceleration server allocates	as part of a 'Receive Tunnels List' step 62 b." E.g., Page 114: "The received content is prepared in a 'Content Prepared' state 54 d, and then sent, in a 'Send Content' message 56 j (corresponding to a 'Send Content To Client' step 73 e), to the client device #1 31 a. The tunnel device 33 a may then revert to idling in the 'IDLE' step 73 a, until a new request is received." E.g., Page 146: "In a 'Request List Agent #1' step 234 a in the flowchart 230 a, the client device #1 201 a send to the agent device #1 103 a (using its identifier from the list received from the acceleration server 202) a request for a list of peers associated the requested content identifier (such as a URL), such as these peer devices that are known or expected to store chunks of the requested content (or any part of it), corresponding to the 'Request List' message 226 g in the chart 220. The agent device #1 103 a, which may be idling in an 'IDLE' step 241 c, receives the request from the client device #1 201 a in a 'Receive List Request' step 241 d." E.g., Page 195: "A network element may thus execute a flowchart 490 b shown in FIG. 49 b. The validation process is

		15816 more Agents to this Data Server to complete the list to 5 Agents, and sends 15818 the list to the client, and resumes waiting for additional messages. If the list created in 15814 is of more than 5 Agents, then the Acceleration Server checks 15820 how many of these Agents in the list that are assigned to this Data Server have transmitted a congestion signal to the Acceleration Server in the past 10 seconds (10 is an example, could be other times as well). If 80% (for example) of the Agents in the list have not transmitted a congestion message within this timeframe, then the Acceleration Server removes 15822 an Agent from this list, and marks it within the database as not assigned to this particular Data Server any longer, and then sends 15818 the list of Agents to the Client. If in 15814 there are exactly 5 Agents in the list then these are sent 15818 to the Client." See, e.g., Fig. 160 and associated description on page 31: "In 16004 the Agent checks its own resources (storage, 1/0, CPU, Bandwidth, etc.) and if they are being used over a certain threshold, (e.g. above 10% of the CPU for example), then the agent sends a 'congested' signal to the acceleration server[.]"	considered as a low-priority task, so in an 'Idle?' step 495 a, the activity of the network element is checked, such as checking the CPU utilization, the available storage size, or the available communication bandwidth. In the case the activity is above a set threshold, the higher-priority activities are given precedence, and the validation activity is not activated, and the element remains in the 'Idle?' step 495 a. Upon availability of enough resources and determination that no other more important tasks are to be activated, the device scans the local memory (or cache) in a 'Scan Cache' step 495 b, and the entries of the various content copies are checked for validity. In the case where all the content parts are found to be valid in a 'Non-Valid Content?' step 495 c, the device resumes to the idling of the validation process in the 'Idle?' step 495 a, since no validation activity is required." E.g., Pages 210-211: "There are applications that may benefit from understanding when the computer's resources are idle. For example, a screensaver monitors the mouse and keyboard movements for idleness, and after a preset amount of idle time, it activates its display program. Further, the idling period may be utilized for

		E.g., Page 31: "In 16006 the acceleration server checks for congestion by pinging the Agents (or sending them some other type of request) and marking them as congested if they do not reply above a certain time (e.g. above 300ms)[.]"	performing non-time sensitive activities, such as updating and maintenance. For example, idling is detected as part of the 'Idle?' step 495 a in the flowchart 490 b. Other cases are for a web site to monitor inactivity of a keyboard or mouse input, and to log out as a consequence." See, e.g., Fig. 77 and associated description on page 211: "***FIG. 77 is a flowchart of an idle monitor that uses new inputs to define idleness of a computing device or its operator***. Such a flowchart may be executed by any network element herein. In a step 70002, an application wishes to be notified of when idleness has occurred, and so notifies the idle monitor program, provides a CallBack function (CB)—a function to call when the idleness occurs—and defines the type of idleness on which to call this function (PARAMS) and the duration of time they should be idle for the CB to be called. ***The program then scans in a step 70004 'idleness' resources and tracks for how long each has been idle. These resources include the following***: Bits being sent or received to/from communication device—i.e. idleness of communication. Storage device read or write—i.e. idleness of the IO with storage devices. Temperature changes in any of the device's temperature

			gauges—i.e., idleness of the environment. Non-idleness of the environment could signal that there is movement in the environment of this computing device, or that the temperature around is shifting. CPU busy over certain threshold (could be 5% for example)—i.e. idleness to a degree of the computing resources. Camera senses motion—i.e. idleness of the physical surrounding of the computing device (movement of persons for example can trigger this to not be idle)" E.g., Page 211: "Light sensor senses changes in light intensity or structure—i.e., idleness of the physical surrounding of the computing device (movement of persons, for example, can trigger this to not be idle): Accelerometer that senses movement or orientation change HD accelerometer that senses movement GPS sensor that senses movement If any of the above sensors is idle in a step 70006, then check in a step 70008 whether all sensors included in PARAMS are idle and have been idle for the amount of time as described in T. If they have all been idle for this time, call the CB." E.g., Pages 211-212: "Hence, in a system monitoring the idleness of a computing device, the following sensors may be

			included in a list of devices to be monitored for idleness: Bits being sent or received to/from communication device, storage device read or write, temperature changes in any of the device's temperature gauges, CPU busy over certain threshold, camera senses motion, light sensor senses changes in light intensity or structure, Accelerometer that senses movement or orientation change, HD accelerometer that senses movement, GPS sensor that senses movement in location or height" See, e.g., Fig. 94 and associated description on page 223: "When network elements use BW & RTT values from the tables, they could benefit from 'teaching' each other about the information they already have about BW & RTT between them and other network elements. FIG. 94 is a flowchart showing a method where network elements can share BW & RTT information so that a network element that wishes to communicate with a network element that it has not recently communicate with, may assess the BW & RTT to it by learning from the experience of other network elements that have recently communicated with that network element. The method starts with scheduling an update of other network elements in a step 21404. The scheduling can be set to either happen at constant

			time intervals, when the local host is idle, when both the local host and other network element to be updated are idle, or when finished communicating with other network element. If the scheduled event of updating other network elements with the table from the local network element, then a list is created in a step 21408 of which of the network elements to update. This can be all the network elements that have been recently communicated with, or a central server that will be updated and update other network elements, or by getting a list from such a central server. The BW & RTT of this local host are then communicated to the elements in this list." See, e.g., Fig. 30 and associated description on pages 160-161: "In normal operation, the device is in an 'ONLINE' state 302, where the device is connected to the Internet, and may receive messages from, and send messages to, the Internet. Further, a resource (or few resources) in the device may in time become congested in a 'CONGESTED' state 303. The device monitors its resources and performance, and upon detecting a resource utilization that is above a set threshold, declares itself as congested. The congestion detection scheme serves as a mechanism to measure the device performance and quality of service, and may be used to

			alert other devices in the system that the device may not be capable to handle additional tasks or services. The detection of congestion may be further used for load balancing, such as for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives, for optimizing resource use, maximizing throughput, minimizing response time, and avoiding overload of any one of the resources. Further, using multiple components in a device with load balancing instead of a single component may increase reliability through redundancy. Similarly, using multiple devices in a system or network with load balancing instead of a single (or few) device may increase reliability through redundancy." E.g., Page 161: "In the case a congestion is detected, the device shifts to the 'CONGESTED' state 303, as depicted by an arrow 304 e. Upon detecting that the detected congestion has elapsed, the device may resume to normal operation in the 'ONLINE' state 302, as depicted by an arrow 304 d. The device may also shift from the 'CONGESTED' state 303 to the 'OFFLINE' state 301, as depicted by an arrow 304 c."

			E.g., Page 161: "In one example, the congestion decision may be based on a CPU utilization, where CPU time or CPU usage is reported either for each thread, for each process, or for the entire system. The CPU utilization relates to the relative time that the CPU is not idling (for example, the amount of time it not executing a system idle process). In the case the CPU utilization is above a predetermined threshold, such as 80%, the device declares itself as congested."
29b	(b) sending over the Internet a first request from the first device to the selected tunnel device using a tunneling protocol and the group device identifier of the selected tunnel device, the first request including the content slice identifier and the second identifier and indicating that the content slice is to be fetched directly from the first server;	See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel	See, e.g., original claim 134 See, e.g., Figs. 5, 5a, 5b, and 7 E.g., Page 81: "…the first device sending a first request to the selected device using the selected device identifier, the first request including the content slice identifier and the second identifier…" E.g., Page 114: "Once the client device #1 31 a determines that external content from a data server is required, as shown in a 'Content Required' state 53 d, a 'Content Request' message 56 g (shown in the messaging chart 50) is sent (corresponding to a 'Send Content Request' step 63 b in the flowchart 60) to the selected tunnel device #1 33 a. The request is received at the tunnel device #1

		sends the data (7010) back to the requesting Client." See, e.g., Fig. 58 and associated description on Page 2: "Figure 58 is a diagram of how Tunnels can be used by a Client. The "Tunnel" 16 is used by the Client 2 to fetch data from the Data Server 5, when such data is not found in the acceleration network's elements Caches, or the data is found in the caches but does not make sense for the Client 2 to fetch it for practical reasons, such as because the time to fetch it from the acceleration network's caches would be longer than to fetch the data directly from the Data Server 5. In such a case where it is beneficial for the Client 2 to fetch the data directly from the Data Server 5, the Client 2 may use one or more Tunnels 16 to carry out the request to the Data Server 5 on its behalf. ... Therefore, once the Client 2 has determined that it is not worthwhile to use the caches within the various acceleration systems' network elements, it will determine how many Tunnels 16 to use for this request based on the amount of data to fetch, and the speed of the available Tunnels 16. … The Client 2 will then send the request to all these Tunnels 16 along with the requested range for them to fetch from the Data Server."	33 a at a 'Request Received' state 54 c, corresponding to a 'Receive Content Request' 73 b in the flowchart 70). In response, the tunnel device 33 a sends a 'Content Request' message 56 h to the data server #1 22 a (corresponding to a 'Send Request To Server' step 73 c), requesting the content that was requested by the client device #1 31 a." E.g., Pages 117-118: "A 'Content Request' message 111 a (corresponding to the 'Content Request' message 56 g in the timing chart 50) is first sent from the client device #1 31 a to the selected tunnel device #1 33 a, which responds by forwarding the request to the data server #1 22 a using a 'Content Request' message 111 b (corresponding to the 'Content Request' message 56 h in the timing chart 50)." E.g., Page 145: "A content, such as an URL (or a web-page, or a web-site) which is typically stored in a data server, such as the data server #1 22 a, may be requested by the client device, such as the client device 201 a, as shown in a state 'Content Needed' 223 b in the chart 220. The client device sends a 'Request List' message 226 e to the acceleration server 202, corresponding to a 'Request Agents List' step 231 c in the flowchart 230. This

		E.g., Page 2: "Tunnels receives a data request from a Client, including the data server to get the request from and the associated other data required for such a data request, and fetches this from the data server and provides back to the Client." E.g., Page 3: "…it is always possible for the Client 2 to fetch the information through Tunnels 16 by requesting that they fetch the information on the Client's behalf from the Data Server 5." See, e.g., Figs. 61 and 62 and associated description on page 3: "Figures 61 and 62 are flowcharts that suggest an implementation of a method of choosing the optimal method- whether to fetch the data from the acceleration network (called in the figures "PATH_ACCEL"), or ***directly via the Tunnels (called in the figures "PATH_DIRECT")***." See, e.g., Fig. 69 and associated description on Page 7: "Figure 69 describes the PATH_DIRECT- i.e. how to retrieve the data from the Tunnels 16 when this option is chosen." E.g., Page 7: "In 6904 the data request is sent to each of the tunnels along with the segment that this Tunnel needs to fetch	request includes the URL or any other identifier of the requested content." E.g., Pages 34-35: "Layer 3 (Network Layer) and lower layer encryption based protocols include IPsec, L2TP (Layer 2 Tunneling Protocol) over IPsec, and Ethernet over IPsec. The IPsec is a protocol suite for securing IP communication by encrypting and authenticating each IP packet of a communication session." E.g., Page 85: "Alternatively or in addition, the first device may be communicating with the second device using a VPN or a tunneling protocol, and the connection may be established using authentication." E.g., Page 99: "Any communication between any two network elements may use TCP, and wherein the connection may be established by performing 'Active OPEN' or 'Passive OPEN', may use a VPN, or may use a tunneling protocol." E.g., Page 69: "The client device may then select one (or more) tunnel device, and then executes a pre-connection process with the selected tunnel device. Upon determining the need for a content to be fetched from the data server, the

		from the Data Server. The Tunnel will then make that request to the Data Server and return the information received to the Client." E.g., Page 1: "Once received, in 5510 the device establishes connections (e.g. TCP connect) to each of the Tunnels in the list, so that once messages are needed to be sent to/from these Tunnels, it can be done quickly due to this pre-connection." See, e.g., Figs. 136 and 138 and associated description on Page 23: "In it, the Client 2 would like to open more connections with the Data Server 5 than the Data Server 5 will allow (where each of these connections is loading one or more requests, where each request is a resource, a range of a resource, or multiple resources). The Client then communicates with participating Tunnels 4 (i.e. network devices on which software is installed for this purpose), and requesting from each of these Tunnels to load several of these requests, such that the each of the participating elements (the Client and the Tunnels from which it requested the loads) does not open more connections vs. the Data Server 5 that the Data Server 5 allows."	client device sends a request to the tunnel device, which in turn fetches the required content from the data server, and sends the fetched content to the client device." See, e.g., Fig. 33 and associated description on pages 167-168: "In the case the requested content is not locally available at the device, the client device #1 341 may check in a 'Direct Fetch ?' step 331 d the possibility of directly accessing a data server (such as the data server #1 22 a) storing the content. In one example, such directly approaching the data server without using any intermediate devices such as using tunnel devices, or fetching the content from peer devices, may result in less overhead and handling, and sometimes may be faster. In the case of direct fetching, the client device #1 341 accesses and fetch the requested content directly from the data server in a 'Fetch from Server' step 331 e. … If the direct fetching is not selected, then in a 'Method Select' step 331 f, the device selects which content fetching method to use. The selection of which method to use may be based on estimation of the latency associated with each method until the content is fully fetched. In one example, a method may be selected when the estimated latency using the other method is substantially longer. The client device

		See also, e.g., Fig. 226 and associated description on Page 44: "It is thus useful to create a system or a method that would allow to always send the data from an available socket. Figure 226 is a diagram of a system that makes it possible to move data between sockets so that if data is waiting to be sent through a socket and during that time a different socket is ready for sending data, the data is moved to that other socket."	#1 341 may select to only use tunnel devices ('Tunnels Only'), and in this scenario, it will execute the tunnels-using client device flowchart (such as the flowchart 60 in FIG. 6) as part of a 'Tunnel Flowchart' step 331 i." E.g., Page 88: "A tunnel device may receive from a client device a request for content from a data server. Upon receiving such a request, the tunnel device fetches the requested content from the data server and sends the retrieved content to the client device. The request may specify a range of, or any portion of, the content, and then only the specified portion or range is retrieved from the data server and sent to the requesting client device." See, e.g., Fig. 11 and associated description on pages 117-118: "A schematic messaging flow diagram 110 describing the client device #1 31 a related 'content fetch' flowchart 65 and the tunnel device #1 33 a related flowchart 73 is shown in FIG. 11. A 'Content Request' message 111 a (corresponding to the 'Content Request' message 56 g in the timing chart 50) is first sent from the client device #1 31 a to the selected tunnel device #1 33 a, which responds by forwarding the request to the data server

			#1 22 a using a 'Content Request' message 111 b (corresponding to the 'Content Request' message 56 h in the timing chart 50). In turn the data server #1 replies and sends the content in a 'Send Content' message 111 c (corresponding to the 'Send Content' message 56 i in the timing chart 50) to the requesting tunnel device #1 33 a, which in turn forward the fetched content to the asking client device #1 31 a using a 'Send Content' message 111 d (corresponding to the 'Send Content' message 56 j in the timing chart 50)." See, e.g., Fig. 10a and associated description on page 119: "Such a flowchart 100 a is shown in FIG. 10 a, where a 'Content Fetch Direct' step 65 d is added. In this step 65 d, the client device #1 31 a directly accesses the data server #1 22 a, as typically known, and in the same way, or in a similar way, the tunnel devices are accessing the data server #1 22 a for fetching content therefrom."
29c	(c) in response to receiving the ~~sent~~ first request ~~by the selected device~~, sending the content slice identifier from the selected tunnel device to	See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the	See, e.g., original claim 108 and original claim 120 See, e.g., Figs. 5, 5a, 5b, and 7

	the first server using the second identifier;	standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client." See, e.g., Fig. 58 and associated description on Page 2: "Figure 58 is a diagram of how Tunnels can be used by a Client. The "Tunnel" 16 is used by the Client 2 to fetch data from the Data Server 5, when such data is not found in the acceleration network's elements Caches, or the data is found in the caches but does not make sense for the Client 2 to fetch it for practical reasons, such as because the time to fetch it from the acceleration network's caches would be longer than to fetch the data directly from the Data Server 5. In such a case where it is beneficial for the Client 2 to fetch the data directly from the Data Server 5, the Client 2 may use one or more Tunnels 16 to	E.g., Pages 76-77: "…in response to receiving the second request, the second device sending the first content identifier to the second server using the third identifier…" E.g., Page 81: "…in response to receiving the first request, the selected device sending a second request to the first server using the second identifier, the second request including the content slice identifier…" E.g., Page 114: "The request is received at the tunnel device #1 33 a at a 'Request Received' state 54 c, corresponding to a 'Receive Content Request' 73 b in the flowchart 70). In response, the tunnel device 33 a sends a 'Content Request' message 56 h to the data server #1 22 a (corresponding to a 'Send Request To Server' step 73 c), requesting the content that was requested by the client device #1 31 a."

		carry out the request to the Data Server 5 on its behalf." See, e.g., Fig. 62 and associated description on Page 4: "6128 sends the resulting data (received directly from the Data Server 5 through the Tunnels 16, or from the acceleration network through the Agents 4 and Peers 3, to the requesting Application." See, e.g., Fig. 69 and associated description on Page 7: "Figure 69 describes the PATH_DIRECT- i.e. how to retrieve the data from the Tunnels 16 when this option is chosen."	
29d	(d) following step (c), receiving over the Internet the content slice from the first server through ~~from~~ the selected tunnel device; and	See, e.g., Fig. 70 and associated description on Pages 7-8: The Tunnel the creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client. See, e.g., Fig. 58 and associated description on Page 2 : "Figure 58 is a diagram of how Tunnels can be used by a Client. The "Tunnel" 16 is used by the Client 2 to fetch data from the Data Server 5, when such data is not found in the acceleration network's elements Caches,	See, e.g., original claim 134 See, e.g., Figs. 5, 5a, 5b, and 7 E.g., Page 81: "…in response to receiving the first request, the selected device sending a second request to the first server using the second identifier, the second request including the content slice identifier; in response to receiving the second request, the first server sending the content slice to the selected device; and in response to receiving the content slice, the selected device sending the content slice to the first device."

		or the data is found in the caches but does not make sense for the Client 2 to fetch it for practical reasons, such as because the time to fetch it from the acceleration network's caches would be longer than to fetch the data directly from the Data Server 5. In such a case where it is beneficial for the Client 2 to fetch the data directly from the Data Server 5, the Client 2 may use one or more Tunnels 16 to carry out the request to the Data Server 5 on its behalf." See, e.g., Fig. 62 and associated description on Page 4: "6128 sends the resulting data (received directly from the Data Server 5 through the Tunnels 16, or from the acceleration network through the Agents 4 and Peers 3, to the requesting Application." See, e.g., Fig. 69 and associated description on Page 7: "Figure 69 describes the PATH_DIRECT- i.e. how to retrieve the data from the Tunnels 16 when this option is chosen." See also, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along with the segment that this Tunnel needs to fetch from the Data Server. The Tunnel	E.g., Page 114: "The request is received at the tunnel device #1 33 a at a 'Request Received' state 54 c, corresponding to a 'Receive Content Request' 73 b in the flowchart 70). In response, the tunnel device 33 a sends a 'Content Request' message 56 h to the data server #1 22 a (corresponding to a 'Send Request To Server' step 73 c), requesting the content that was requested by the client device #1 31 a." E.g., Page 118: "In turn the data server #1 replies and sends the content in a 'Send Content' message 111 c (corresponding to the 'Send Content' message 56 i in the timing chart 50) to the requesting tunnel device #1 33 a, which in turn forward the fetched content to the asking client device #1 31 a using a 'Send Content' message 111 d (corresponding to the 'Send Content' message 56 j in the timing chart 50)." E.g., Page 114: "The data server #1 22 a receives the request and prepares the requested content in a 'Content Prepared' state 55 a, and sends the requested content back to the tunnel device #1 33 a in a 'Send Content' message 56 i, received by the tunnel device #1 33 a in a 'Receive Content from Server' step 73 d. The

		will then make that request to the Data Server and return the information received to the Client."	received content is prepared in a 'Content Prepared' state 54 d, and then sent, in a 'Send Content' message 56 j (corresponding to a 'Send Content To Client' step 73 e), to the client device #1 31 a."
29e	wherein the method further ~~comprising~~comprises the step of constructing the fresh content from the received plurality of content slices, and	See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client." E.g., Page 21: "a system that comprises of a software module, a database, where upon getting a uri as an input, the system looks up the domain of the URL within	See, e.g., original claim 134 See, e.g., Figs. 5 and 5b E.g., Page 81: "…receiving the content slice from the selected device; and constructing the content from the received content slices." See, e.g., Fig. 21 and associated description on Page 139: "For example, multiple chunks may be combined to reconstruct the original content, such as website or content, as schematically shown in an arrangement 210 shown in FIG. 21." E.g., Page 139: "In one example, the content is a website or a webpage, or may be identified as a URL, and consists of, or comprises, non-overlapping and equally-sized parts, referred to as chunks. For example, multiple chunks may be combined to reconstruct the original content, such as website or content."

		the database, retrieves the schema of the URL and can identify the identity of the video, and the offset to view within the video according to the schema[.]" See, e.g., Fig. 62 and associated description on Page 4: "6128 sends the resulting data (received directly from the Data Server 5 through the Tunnels 16, or from the acceleration network through the Agents 4 and Peers 3, to the requesting Application." See, e.g., Fig. 69 and associated description on Page 7: "Figure 69 describes the PATH_DIRECT- i.e. how to retrieve the data from the Tunnels 16 when this option is chosen." See also, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along with the segment that this Tunnel needs to fetch from the Data Server. The Tunnel will then make that request to the Data Server and return the information received to the Client." See also, e.g., Page 13: "Next, the agent decides on a chunk size {8006) which will be the size of the pieces in to which the data is broken down to, for the sake of assigning pieces to the Peers to deliver to	E.g., Page 126: "It is noted that in such a partition, the content may be fully reconstructed from any two of the slices, hence providing a degree of redundancy. For example, in case of carrying the three slices over the Internet and a failure to receive one of the slices, the remaining two slices may be used to fully reconstruct the whole content."

		the Client (chunk_size). The chunk_size can either be a set value (for example, 16KB), or a number that changes according to the size of the data that is requested (for example 1/10th of the size of the data itself)." E.g., Page 33: "In 17402 the client is going to load a file which is available bia multiple sources (source(1) ... source(n_sources). In 17404, the requested file is split up in toN chunks of equal size (chunk_size) that was pre-determined in the system. For example, this chunk size can be 16KB." See also Pages 33-34 ("Microchunks" section) See also Fig. 73	
29f	wherein each of the tunnel devices in the group is a client device comprising a consumer electronic device with a client operating system, and	E.g., Page 1: "In the network of figure 50, there is a new type of element called a Tunnel. As in patent 12836059, each of the communication devices in the network may store the same software that under different conditions enables that communication device to assume a different role, and may also take multiple roles at the same time (e.g. the communication device can be both a Client and a Tunnel for example), and thus a device may for example	E.g., Page 56: "A device herein (such as device 11) may consist of, be part of, or include, a Personal Computer (PC), a desktop computer, a mobile computer, a laptop computer, a notebook computer, a tablet computer, a server computer, a handheld computer, a handheld device, a Personal Digital Assistant (PDA) device, or a cellular handset."

		communicate with itself under different roles (for example a Client may use itself also as a Tunnel)." See, e.g., Fig. 234 and associated description on Page 46: "Figure 234 is a diagram that shows such an architecture. A network element 23402 has an operating system 10 and applications 8, with a virtual application gateway module 23404 running on it as well." See also, e.g., Pages 47-48: "In a computing device that is connected to a network and includes an operating system and applications, a method by which a program on the device provides the communication configuration to the operating system instead of the operating system getting it from a gateway on the network, where the program communicates with the external gateway to get the configuration information so that it can communicate to other network elements, and provides separate configuration information to the operating system, thereby having the operating system communicate with the program, and the program communicating with the external network elements[.]"	E.g., Page 100: "Each of the network elements herein, such as the first, second, and third devices, may store, operate, or use, a client operating system, that may consist or, comprise of, or may be based on, Microsoft Windows 7, Microsoft Windows XP, Microsoft Windows 8, Microsoft Windows 8.1, Linux, or Google Chrome OS. The client operating system may be a mobile operating system, such as Android version 2.2 (Froyo), Android version 2.3 (Gingerbread), Android version 4.0 (Ice Cream Sandwich), Android Version 4.2 (Jelly Bean), Android version 4.4 (KitKat)), Apple iOS version 3, Apple iOS version 4, Apple iOS version 5, Apple iOS version 6, Apple iOS version 7, Microsoft Windows® Phone version 7, Microsoft Windows® Phone version 8, Microsoft Windows® Phone version 9, or Blackberry® operating system." E.g., Abstract: "A client device may also serve as a tunnel device, serving as an intermediate device to other client devices. Similarly, a tunnel device may also serve as a client device for fetching content from a data server." E.g., Page 167: "A client device such as a client device #1 341 may assume the role

			of the tunnel-using client device #1 31 a, the role of the agent/peers-using client device #1 201 a, or both."
29g	wherein the criterion is satisfied when an idleness of the idle resource is above or below a set threshold value, and	E.g., Page 26: "… the Client receives a list of Agents to use, uses them for a specific transaction, and when the network is idle, sends back to the Acceleration Server information about these agents for the server to use in recommending these agents in the future." E.g., Page 26: "… where the information is sent only when the network is idle." E.g., Page 26: "… where the acceleration server uses this information by providing back this information to the client to use in transactions with the agent." E.g., Page 27: "… where the acceleration server uses this information to decide which agent to recommend in future requests by clients." See, e.g., Fig. 700 and associated description on page 84: "… In a system monitoring the idleness of a computing device, the following sensors may be included in a list of devices to be monitored for idleness: Bits being sent or received to/from communication device , Storage device read or write, Temperature	E.g., Page 97: "A client device may receive and use a list of agent devices for a specific transaction, and when the network is idle, the client device may update the acceleration server regarding the used agent devices, allowing the acceleration server to later recommend or use these agent devices. The updating of the acceleration server may include the IP address of each of the agent devices, the communication session information such as RTT, BW and speed, the ports used in the communication sessions, the latency and speed for each of the connection phases, and whether the required file or data was stored in each of the agent devices. The acceleration server may use this information to decide which agent devices to recommend and include in a future agent devices list requested by client devices." E.g., Page 113: "In order to make the communication between a client device and a tunnel device faster and more efficient, a pre-connection phase is defined, where a preparation for communication such as a TCP connection is established, allowing for quick data

		changes in any of the device's temperature gauges, CPU busy over certain threshold, Camera senses motion, Light sensor senses changes in light intensity or structure, Accelerometer that senses movement or orientation change, HD accelerometer that senses movement, GPS sensor that senses movement in location or hight." See, e.g., Fig. 214 and associated description on page 40: ""When network elements use BW & RTT tables, they could benefit from 'teaching' each other about the information they already have about BW & RTI between them and other network elements. Figure 214 is a flowchart showing a method by which network elements can share BW & RTI information so that a host that wishes to communicate with a network element that it has not recently communicate with can assess the BW & RTT to it by learning from other network elements that have recently communicated with that network element." E.g., Page 40: "The method starts with scheduling an update of other network elements in 21404. The scheduling can be set to either happen at constant time intervals, or when the local host is idle,	transfer afterwards. The pre-connection phase starts at a 'Start Pre-Connection' state 53 b in the chart 50, followed by the 'Request List' message 56 c (corresponding to the 'Request Tunnels List' step 62 in the flowchart 60), being part of the Pre-connection client flowchart 64, where the client 31 a requests the list of the available tunnels that may be used, from the acceleration server 32. The tunnel device #1 33 a at this point is idling in an 'IDLE' step 72 c shown in the flowchart 70, being part of the Pre-connection tunnel flowchart 72. In response to the client device 31 a request, the acceleration server 32 prepares in a step 'Prepare List' 52 b the list of current available tunnels, and sends the list as a 'Send List' message 56 d to the client device 31 a, which in turn receives the list as part of a 'Receive Tunnels List' step 62 b." E.g., Page 114: "The received content is prepared in a 'Content Prepared' state 54 d, and then sent, in a 'Send Content' message 56 j (corresponding to a 'Send Content To Client' step 73 e), to the client device #1 31 a. The tunnel device 33 a may then revert to idling in the 'IDLE' step 73 a, until a new request is received."

		or when both the local host and other network element to be updated are idle, or when finished communicating with another network element, etc." See, e.g., Fig. 610 and associated description on page 69: "Figure 610 is a flowchart for how this can be done, by validating invalid (or soon to be invalid) cache entries while the communication device is idle." E.g., Page 69: "61002 checks if the communication device's resources are idle (or idle enough) to update entries in its cache without degrading system performance (i.e. checks that CPU is below a certain threshold for example of 30%, that the disk is not in use, that there is excess bandwidth that can be used, etc.)." See, e.g., Fig. 158 and associated description on page 30: "In 15808 the Acceleration Server receives a request from a Client that seeks to communicate with a Data Server, and the Client requests to receive a list of Agents through which to communicate. The Acceleration server then creates 15810 a list of Agents that are assigned to this particular Data Server according to its database. It then checks 15814 the number	E.g., Page 146: "In a 'Request List Agent #1' step 234 a in the flowchart 230 a, the client device #1 201 a send to the agent device #1 103 a (using its identifier from the list received from the acceleration server 202) a request for a list of peers associated the requested content identifier (such as a URL), such as these peer devices that are known or expected to store chunks of the requested content (or any part of it), corresponding to the 'Request List' message 226 g in the chart 220. The agent device #1 103 a, which may be idling in an 'IDLE' step 241 c, receives the request from the client device #1 201 a in a 'Receive List Request' step 241 d." E.g., Page 195: "A network element may thus execute a flowchart 490 b shown in FIG. 49 b. The validation process is considered as a low-priority task, so in an 'Idle?' step 495 a, the activity of the network element is checked, such as checking the CPU utilization, the available storage size, or the available communication bandwidth. In the case the activity is above a set threshold, the higher-priority activities are given precedence, and the validation activity is not activated, and the element remains in the 'Idle?' step 495 a. Upon availability of enough resources and determination that

		of Agents in this list. If there are under 5 (example implementation- can be other numbers as well) Agents assigned to this Data Server, then the Acceleration server allocates 15816 more Agents to this Data Server to complete the list to 5 Agents, and sends 15818 the list to the client, and resumes waiting for additional messages. If the list created in 15814 is of more than 5 Agents, then the Acceleration Server checks 15820 how many of these Agents in the list that are assigned to this Data Server have transmitted a congestion signal to the Acceleration Server in the past 10 seconds (10 is an example, could be other times as well). If 80% (for example) of the Agents in the list have not transmitted a congestion message within this timeframe, then the Acceleration Server removes 15822 an Agent from this list, and marks it within the database as not assigned to this particular Data Server any longer, and then sends 15818 the list of Agents to the Client. If in 15814 there are exactly 5 Agents in the list then these are sent 15818 to the Client." See, e.g., Fig. 160 and associated description on page 31: "In 16004 the Agent checks its own resources (storage, 1/0, CPU, Bandwidth, etc.) and if they are	no other more important tasks are to be activated, the device scans the local memory (or cache) in a 'Scan Cache' step 495 b, and the entries of the various content copies are checked for validity. In the case where all the content parts are found to be valid in a 'Non-Valid Content?' step 495 c, the device resumes to the idling of the validation process in the 'Idle?' step 495 a, since no validation activity is required." E.g., Pages 210-211: "There are applications that may benefit from understanding when the computer's resources are idle. For example, a screensaver monitors the mouse and keyboard movements for idleness, and after a preset amount of idle time, it activates its display program. Further, the idling period may be utilized for performing non-time sensitive activities, such as updating and maintenance. For example, idling is detected as part of the 'Idle?' step 495 a in the flowchart 490 b. Other cases are for a web site to monitor inactivity of a keyboard or mouse input, and to log out as a consequence." See, e.g., Fig. 77 and associated description on page 211: "***FIG. 77 is a flowchart of an idle monitor that uses new inputs to define idleness of a***

		being used over a certain threshold, (e.g. above 10% of the CPU for example), then the agent sends a 'congested' signal to the acceleration server[.]" E.g., Page 31: "In 16006 the acceleration server checks for congestion by pinging the Agents (or sending them some other type of request) and marking them as congested if they do not reply above a certain time (e.g. above 300ms)[.]"	***computing device or its operator***. Such a flowchart may be executed by any network element herein. In a step 70002, an application wishes to be notified of when idleness has occurred, and so notifies the idle monitor program, provides a CallBack function (CB)—a function to call when the idleness occurs—and defines the type of idleness on which to call this function (PARAMS) and the duration of time they should be idle for the CB to be called. ***The program then scans in a step 70004 'idleness' resources and tracks for how long each has been idle. These resources include the following***: Bits being sent or received to/from communication device—i.e. idleness of communication. Storage device read or write—i.e. idleness of the IO with storage devices. Temperature changes in any of the device's temperature gauges—i.e., idleness of the environment. Non-idleness of the environment could signal that there is movement in the environment of this computing device, or that the temperature around is shifting. CPU busy over certain threshold (could be 5% for example)—i.e. idleness to a degree of the computing resources. Camera senses motion—i.e. idleness of the physical surrounding of the computing device (movement of persons for example can trigger this to not be idle)"

			E.g., Page 211: "Light sensor senses changes in light intensity or structure—i.e., idleness of the physical surrounding of the computing device (movement of persons, for example, can trigger this to not be idle): Accelerometer that senses movement or orientation change HD accelerometer that senses movement GPS sensor that senses movement If any of the above sensors is idle in a step 70006, then check in a step 70008 whether all sensors included in PARAMS are idle and have been idle for the amount of time as described in T. If they have all been idle for this time, call the CB." E.g., Pages 211-212: "Hence, in a system monitoring the idleness of a computing device, the following sensors may be included in a list of devices to be monitored for idleness: Bits being sent or received to/from communication device, storage device read or write, temperature changes in any of the device's temperature gauges, CPU busy over certain threshold, camera senses motion, light sensor senses changes in light intensity or structure, Accelerometer that senses movement or orientation change, HD accelerometer that senses movement, GPS sensor that senses movement in location or height"

			See, e.g., Fig. 94 and associated description on page 223: "When network elements use BW & RTT values from the tables, they could benefit from 'teaching' each other about the information they already have about BW & RTT between them and other network elements. FIG. 94 is a flowchart showing a method where network elements can share BW & RTT information so that a network element that wishes to communicate with a network element that it has not recently communicate with, may assess the BW & RTT to it by learning from the experience of other network elements that have recently communicated with that network element. The method starts with scheduling an update of other network elements in a step 21404. The scheduling can be set to either happen at constant time intervals, when the local host is idle, when both the local host and other network element to be updated are idle, or when finished communicating with other network element. If the scheduled event of updating other network elements with the table from the local network element, then a list is created in a step 21408 of which of the network elements to update. This can be all the network elements that have been recently communicated with, or a central server that will be updated and

			update other network elements, or by getting a list from such a central server. The BW & RTT of this local host are then communicated to the elements in this list." See, e.g., Fig. 30 and associated description on pages 160-161: "In normal operation, the device is in an 'ONLINE' state 302, where the device is connected to the Internet, and may receive messages from, and send messages to, the Internet. Further, a resource (or few resources) in the device may in time become congested in a 'CONGESTED' state 303. The device monitors its resources and performance, and upon detecting a resource utilization that is above a set threshold, declares itself as congested. The congestion detection scheme serves as a mechanism to measure the device performance and quality of service, and may be used to alert other devices in the system that the device may not be capable to handle additional tasks or services. The detection of congestion may be further used for load balancing, such as for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives, for optimizing resource use, maximizing throughput, minimizing response time, and avoiding overload of any one of the resources.

			Further, using multiple components in a device with load balancing instead of a single component may increase reliability through redundancy. Similarly, using multiple devices in a system or network with load balancing instead of a single (or few) device may increase reliability through redundancy." E.g., Page 161: "In the case a congestion is detected, the device shifts to the 'CONGESTED' state 303, as depicted by an arrow 304 e. Upon detecting that the detected congestion has elapsed, the device may resume to normal operation in the 'ONLINE' state 302, as depicted by an arrow 304 d. The device may also shift from the 'CONGESTED' state 303 to the 'OFFLINE' state 301, as depicted by an arrow 304 c." E.g., Page 161: "In one example, the congestion decision may be based on a CPU utilization, where CPU time or CPU usage is reported either for each thread, for each process, or for the entire system. The CPU utilization relates to the relative time that the CPU is not idling (for example, the amount of time it not executing a system idle process). In the case the CPU utilization is above a predetermined threshold, such as 80%, the device declares itself as congested."

29h	wherein the selected tunnel device does not cache a copy of the content slice, and	E.g., Page 18: "[A] method by which when the Client sends the request to the Tunnel, the Tunnel ***may*** provide back results from its cache[.]" (emphasis added) E.g., Page 17: "When the Tunnel receives 10402 a request for data from a Client it first checks 10404 whether this Tunnel already has received such a request in the past or if its currently serving such a request, in which case it ***may have some of the data cached***, and thus may provide it from the cache instead of fetching it from the Data Server." (emphasis added) See, e.g., Figs. 61 and 62 and associated description on Page 3: "In 6103 it checks whether the data required is located in the local cache of this communication device…If the data does not exist in the local cache…" See also, e.g., Page 8: "Often Peers will either have all of the file within their cache, or none of it."	E.g., Page 94: "The content ***may be stored*** in a device in each location…" (emphasis added) E.g., Page 117: "Upon receiving a request for content from a client device, a tunnel device (such as the tunnel device #1 33 a) first checks if the requested content is stored locally (in the tunnel device itself), such as in its cache memory, in a 'Locally Available ?' step 75 a. The requested content ***may be stored*** in the tunnel device as a result of a former accessing the respective data server, for example by a web browser (or any other application) that is part of the tunnel device." (emphasis added) E.g., Page 117: "Alternatively or in addition, upon receiving the requested content from the data server in the 'Receive Content From Server' step 73d, the receive content may be stored locally in the tunnel device for future use, in the 'Store Content From Server' step 75b." E.g., Pages 129-130: "The fetched content may be stored in the client device in any volatile or non-volatile memory, or ***may be stored*** in a local cache as described in U.S. Pat. No. 8,135,912… For example, the stored content may be used when the

			same content is required at any later stage by the same client…" (emphasis added)
29i	wherein the first device remains anonymous as to the first server.	See, e.g., Fig. 58 and associated description on Page 2 : "In such a case where it is beneficial for the Client 2 to fetch the data directly from the Data Server 5, the Client 2 may use one or more Tunnels 16 to carry out the request to the Data Server 5 ***on its behalf***." (emphasis added) E.g., Page 3: "[I]t is always possible for the Client 2 to fetch the information through Tunnels 16 by requesting that they fetch the information ***on the Client's behalf*** from the Data Server 5." (emphasis added) See, e.g., Figs. 61 and 62 and associated description on Page 3: "Figures 61 and 62 are flowcharts that suggest an implementation of a method of choosing the optimal method- whether to fetch the data from the acceleration network (called in the figures "PATH_ACCEL"), or ***directly via the Tunnels*** (called in the figures "PATH_DIRECT")." (emphasis added) See, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along	E.g., Page 117: "Since the data server #1 22 a is accessed by, and sends information only to, tunnel devices (such as the tunnel device #1 33 a), and is not aware of the final content destination being the client device #1 31 a, the identity (such as the IP address) of the client device #1 31 a is concealed from the data server #1 22 a, thus providing anonymity and untraceability. Further, in a case where the data server #1 22 a is a web server, the method and system described may provide for an anonymous web browsing." E.g., Page 128: "The service allows the client device (such as client device #1 31 a) to quickly and anonymously fetch content from the data server #1 22 a." See, e.g., Fig. 13 and associated description on Page 128 : "In one example shown as a system 130 in FIG. 13, the client device #1 31 a functionality is implemented in as client device #1 system 133, comprising a computer 132 (may be used for GUI or as a client), is communicating with a proxy server 131."

		with the segment that this Tunnel needs to fetch from the Data Server. The Tunnel will then make that request to the Data Server and return the information received to the Client." See, e.g., Fig. 234 associated description on Page46: "When the network element 23402 is connected to a gateway 23410, the virtual application gateway 23404 intercepts the operating system's request to the network for receiving configuration information, and receives on its own the configuration information from the gateway 23410 and provides that information back to the operating system, thereby acting as a sort of proxy for the initial configuration information. Some of the requests that the virtual application gateway module 23404 receives may require it to do actions on its own which are not in the form of being a proxy. One such action is the IP request from the operating system in that case the virtual application gateway module 23404 will need to provide a local IP to the operating system." See, e.g., Fig. 236 and associated description on Page 47: "Then, the VAGM sets 23604 itself up for intercepting the IP requests of the operating system that would normally	

		have gone out to the gateway... Once such a request has been trapped 23610 the VAGM provides an IP to the OS in a similar way to how NAT devices provide IPs to requesting clients. The VAGM then acts 23606 as a proxy/nat for the operating system as NAT devices do."	
30	30. (proposed substitute for claim 16) ~~16.~~ The method according to claim ~~15~~[[29]], wherein the fresh content is composed of bits, nibbles, bytes, characters, words, or strings, and the partitioning is based on bit, nibble, byte, multi-byte, number, character, word, or string level.	E.g., Page 68: "If the cached URL has expired, then in 60012 a range request for X bytes is sent to the external source (could be 16kb for example, or could be a random range of 5% of the size (in bytes) of the cached URL's size." E.g., Page 2: "The Client 2 will also determine how much of the data it will request from each of the Tunnels 16 based on their speed, and thus assign a 'range' of the data (for example for a data of 1200 bytes where two agents A and Bare selected, a range assignment may be that A is assigned the bytes between byte 0 and byte 856, and B is assigned bytes 857 to 1200). The Client 2 will then send the request to all these Tunnels 16 along with the requested range for them to fetch from the Data Server." See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be	See, e.g., original claim 135 E.g., Page 82: "A content herein may be composed of bits, nibbles, bytes, characters, words, or strings, and the partitioning may be based on bit, nibble, byte, multi-byte, number, character, word, or string level." E.g., Page 138: "The slicing may be in a bit, nibble (4-bits), byte (8-bits), word (multiple bytes), character, string, or file level. For example, in a case wherein the content includes 240 bytes designated byte #1 to byte #240, using a byte level partitioning into two slices results in a first slice (slice #1) including byte #1 to byte #120, and a second slice (slice #2) including byte #121 to byte #240. In the case of byte-level partitioning into three slices (referred as slice #1, slice #2, and slice #3), a first slice (slice #1) may be including byte #1 to byte #80, a second slice (slice #2) may be including byte #81

		implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client." E.g., Pages 18-19: "Here are some examples of who may benefit from different treatment as described: 1. by URL: specific files on the Internet (e.g. "Wikipedia.org/contact.html") 2. by Domain: specific web sites (e.g. "Wikipedia.org") 3. by Data server IP: specific servers (e.g. "208.80.152.201") 4. by Type of file: specific file types (e.g. ".flv files") 5. by time of day: specific handling of all files or a group of files during certain hours of the day (e.g. "all files between 11pm to 4am") 6. by Geography of Client: specific handling	to byte #160, and a third slice (slice #3) may be including byte #161 to byte #240. Similarly, in a case wherein the content include 3 bytes designated byte #1 to byte #3 representing 24 bits, using a bit-level partitioning into four slices results in a slice #1 including the first 6 bits, slice #2 including the next 6 bits, slice #3 including the next 6 bits, and slice #4 including the last 6 bits." E.g., Page 138: "In one example, the content itself is made of inherent or identifiable parts or segments, and the partition may make use of these parts. In one example, the content may be a website content composed of multiple webpages, and thus the partition may be such that each slice includes one (or few) webpages. Further, the partitioning may be sequential or non-sequential in the content." See also, e.g., Fig. 21

		according to where the Client is (e.g. "for all Clients in Germany")" See, e.g., Fig. 73 See also, e.g., Page 13: "Next, the agent decides on a chunk size {8006) which will be the size of the pieces in to which the data is broken down to, for the sake of assigning pieces to the Peers to deliver to the Client (chunk_size). The chunk_size can either be a set value (for example, 16KB), or a number that changes according to the size of the data that is requested (for example 1/10th of the size of the data itself)."	
31	31. (proposed substitute for claim 17) ~~17.~~ The method according to claim ~~15~~[[29]], wherein the fresh content is composed of files, or programs, and the partitioning is based on file or program level.	E.g., Pages 18-19: "Here are some examples of who may benefit from different treatment as described: 1. by URL: specific files on the Internet (e.g. "Wikipedia.org/contact.html") 2. by Domain: specific web sites (e.g. "Wikipedia.org") 3. by Data server IP: specific servers (e.g. "208.80.152.201") 4. by Type of file: specific file types (e.g. ".flv files") 5. by time of day: specific handling of all files or a group of files during certain hours of the day (e.g. "all files between 11pm to 4am") 6. by Geography of Client: specific handling	See, e.g., original claim 136 E.g., Page 82: "Alternatively or in addition, a content herein may be composed of files or programs, and the partitioning may be based on file or program level." E.g., Page 138: "The slicing may be in a bit, nibble (4-bits), byte (8-bits), word (multiple bytes), character, string, or file level…" E.g., Page 138: "In one example, the content itself is made of inherent or

		according to where the Client is (e.g. "for all Clients in Germany")" See, e.g., Fig. 73 See, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along with the ***segment*** that this Tunnel needs to fetch from the Data Server." (emphasis added) E.g., Page 23: "Data servers typically limit the number of active connections that a host can keep open with them concurrently. This limitation is detrimental in various scenarios. One such scenario as an example is where a communications device with a web browser needs to load ***many URLs from a Data Server in order to display a web page***…" (emphasis added)	identifiable parts or segments, and the partition may make use of these parts. In one example, the content may be a website content composed of multiple webpages, and thus the partition may be such that each slice includes one (or few) webpages. Further, the partitioning may be sequential or non-sequential in the content." See also, e.g., Fig. 21
32	32. (proposed substitute for claim 18) ~~18.~~ The method according to claim ~~17~~[[31]], wherein the fresh content is a website content comprising multiple webpages, and the partitioning is based webpages level.	E.g., Page 80: "68002 are a group of pages in the physical memory of a computer (page #1, Page #2, ... Page #N), in this example each of the pages is a range of 4KB." E.g., Pages 18-19: "Here are some examples of who may benefit from different treatment as described: 1. by	See, e.g., original claim 137 E.g., Page 82: "Further, the content may be a website content comprising multiple webpages, and the partitioning may be based on webpages level." E.g., Page 138: "In one example, the content itself is made of inherent or

		URL: specific files on the Internet (e.g. "Wikipedia.org/contact.html") 2. by Domain: specific web sites (e.g. "Wikipedia.org") 3. by Data server IP: specific servers (e.g. "208.80.152.201") 4. by Type of file: specific file types (e.g. ".flv files") 5. by time of day: specific handling of all files or a group of files during certain hours of the day (e.g. "all files between 11pm to 4am") 6. by Geography of Client: specific handling according to where the Client is (e.g. "for all Clients in Germany")" See, e.g., Fig. 73 See, e.g., Fig. 69 and associated description on Page 7: "In 6904 the data request is sent to each of the tunnels along with the ***segment*** that this Tunnel needs to fetch from the Data Server." (emphasis added) E.g., Page 23: "Data servers typically limit the number of active connections that a host can keep open with them concurrently. This limitation is detrimental in various scenarios. One such scenario as an example is where a communications device with a web browser needs to load ***many URLs from a Data Server in order to display a web page***…" (emphasis added)	identifiable parts or segments, and the partition may make use of these parts. In one example, the content may be a website content composed of multiple webpages, and thus the partition may be such that each slice includes one (or few) webpages. Further, the partitioning may be sequential or non-sequential in the content." See also, e.g., Fig. 21

33	33. (proposed substitute for claim 19) ~~19.~~ The method according to claim ~~15~~[[29]], wherein all of the parts of the fresh content are included in all of the content slices.	E.g., Page 7: "…the range of the data that is requested (e.g. ***all of the data***, or only a certain byte range, for example bytes 1000 to 1020)…" (emphasis added) See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. ***all of the data***, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client." (emphasis added) E.g., Page 50: "In figure 247, a method of transmitting packets in parallel over multiple routes is described, in which one communications device (NE1) wishes to	See, e.g., original claim 138 E.g., Page 82: "All parts of the content may be included in all of the content slices, and two or more, or all of the content slices, may be having the same size." E.g., Page 82: "Two or more of the content slices may include the same information. Further, the same part of the content may be included in two or more content slices." E.g., Page 84: "Part of, or all of, the content parts may be having the same size, that may be 8 KB, 16 KB, 32 KB, or 64 KB. Two or more content parts may be identical and may contain the same data." E.g., Page 126: "The same content may be requested and fetched using multiple tunnel devices as exampled above. Alternatively or in addition, the content may be partitioned into multiple slices (overlapping or non-overlapping), where each slice is requested and fetched using a distinct tunnel device (or via the client device serving as its own tunnel)."

		communicate with another communications device (NE2), sending the ***same data*** over multiple routes in parallel." (emphasis added) See, e.g., Fig. 136 and associated description on Page 23: "Figure 136 describes such a system. In it, the Client 2 would like to open more connections with the Data Server 5 than the Data Server 5 will allow (***where each of these connections is loading one or more requests, where each request is a resource, a range of a resource, or multiple resources***). The Client then communicates with participating Tunnels 4 (i.e. network devices on which software is installed for this purpose), and requesting from each of these Tunnels to load several of these requests, such that the each of the participating elements (the Client and the Tunnels from which it requested the loads) does not open more connections vs. the Data Server 5 that the Data Server 5 allows." (emphasis added)	E.g., Page 139: "The partitioning may be non-overlapping, wherein each slice includes a distinct part of the content, as is exampled above in a case wherein the content includes 240 bytes designated byte #1 to byte #240, where using a byte level partitioning into three slices (referred as slice #1, slice #2, and slice #3), results in a first slice (slice #1) including byte #1 to byte #80, a second slice (slice #2) including byte #81 to byte #160, and a third slice (slice #3) including byte #161 to byte #240. Alternatively or in addition, an overlapping partitioning may be applied, where the same part of the content is included in multiple slices."
34	34. (proposed substitute for claim 20) ~~20.~~ The method according to claim ~~15~~[[29]], wherein all of the content slices have the same size.	E.g., Page 13: "For example, this can be done according to the BW/RTT from the Peer to the Client. Next, the agent decides on a chunk size {8006) which will be the size of the pieces in to which the data is	See, e.g., original claim 139 E.g., Page 82: "All parts of the content may be included in all of the content slices, and two or more, or all of the

		broken down to, for the sake of assigning pieces to the Peers to deliver to the Client (chunk_size). The chunk_size can either be a set value (for example, 16KB), or a number that changes according to the size of the data that is requested (for example 1/10th of the size of the data itself)." E.g., Page 33: "In 17402 the client is going to load a file which is available bia multiple sources (source(1) ... source(n_sources). In 17404, the requested file is split up in toN chunks of ***equal size*** (chunk_size) that was pre-determined in the system. For example, this chunk size can be 16KB." (emphasis added) See, e.g., Fig. 69 and associated description on Page 7: "In 6902, the requested data is split in to ranges, such that there is one range per participating Tunnel. It is possible to simply choose ranges of ***equal size*** to implement this." (emphasis added)	content slices, may be having the same size." E.g., Page 82: "Two or more of the content slices may include the same information. Further, the same part of the content may be included in two or more content slices." E.g., Page 84: "Part of, or all of, the content parts may be having the same size, that may be 8 KB, 16 KB, 32 KB, or 64 KB. Two or more content parts may be identical and may contain the same data." E.g., Page 126: "The same content may be requested and fetched using multiple tunnel devices as exampled above. Alternatively or in addition, the content may be partitioned into multiple slices (overlapping or non-overlapping), where each slice is requested and fetched using a distinct tunnel device (or via the client device serving as its own tunnel)." E.g., Page 139: "The partitioning may be non-overlapping, wherein each slice includes a distinct part of the content, as is exampled above in a case wherein the content includes 240 bytes designated byte #1 to byte #240, where using a byte level partitioning into three slices (referred as slice #1, slice #2, and slice

			#3), results in a first slice (slice #1) including byte #1 to byte #80, a second slice (slice #2) including byte #81 to byte #160, and a third slice (slice #3) including byte #161 to byte #240. Alternatively or in addition, an overlapping partitioning may be applied, where the same part of the content is included in multiple slices."
35	35. (proposed substitute for claim 23) ~~23.~~ The method according to claim ~~15~~[[29]], wherein the partitioning is sequential in the fresh content.	E.g., Page 2: "The Client 2 will also determine how much of the data it will request from each of the Tunnels 16 based on their speed, and thus assign a 'range' of the data (for example for a data of 1200 bytes where two agents A and Bare selected, a range assignment may be that A is assigned the bytes between byte 0 and byte 856, and B is assigned bytes 857 to 1200). The Client 2 will then send the request to all these Tunnels 16 along with the requested range for them to fetch from the Data Server." E.g., Page 7: "In 6902, the requested data is split in to ranges, such that there is one range per participating Tunnel. It is possible to simply choose ranges of equal size to implement this. A possibly more efficient implementation is to provide each Tunnel a range that is proportional to that Tunnel's ability to provide the data (for example, based on that Tunnel's	See, e.g., original claim 142 E.g., Page 76: "The partitioning may be sequential or non-sequential in the content…" E.g., Page 224: "The steps described herein may be sequential, and performed in the described order…" E.g., Page 120: "The tasks relating to the different data paths, such as shown in a flowchart 100 a, relating to communicating with the multiple tunnel devices and/or direct access, may be executed sequentially or in parallel." E.g., Page 82: "The partitioning may be sequential or non-sequential in the content, and the number of the content slices may be equal to, higher than, or lower than, the number of devices in the group."

		upstream and downstream traffic to the Client, as measured in previous communications between them).” E.g., Page 13: “The Agent then assigns the chunks of data to each Peer in the following manner: starting with the first Peer in the list {8010}, which is the fastest Peer because of the sorting in 8004, it then assigns a number of chunks to the Peer {8012) by using a metric of allocation. For example, by providing a set number of chunks per Peer, or by assigning for example BW/RTT chunks to the Peer. If there are more chunks of the data left to assign {8014), then the next Peer in the list is selected {8016), and more chunks are assigned to the Peer in 8012, and so on until all chunks are assigned, and the allocation function ends {8018).” E.g., Page 33: “In 17402 the client is going to load a file which is available bia multiple sources (source(1) ... source(n_sources). In 17404, the requested file is split up in toN chunks of equal size (chunk_size) that was pre-determined in the system. For example, this chunk size can be 16KB.” E.g., Page 7: “…the range of the data that is requested (e.g. all of the data, or only a	E.g., Page 101: “The steps described herein may be sequential, and performed in the described order. For example, in a case where a step is performed in response to another step, or upon completion of another step, the steps are executed one after the other.” E.g., Page 139: “The partitioning may be non-overlapping, wherein each slice includes a distinct part of the content, as is exampled above in a case wherein the content includes 240 bytes designated byte #1 to byte #240, where using a byte level partitioning into three slices (referred as slice #1, slice #2, and slice #3), results in a first slice (slice #1) including byte #1 to byte #80, a second slice (slice #2) including byte #81 to byte #160, and a third slice (slice #3) including byte #161 to byte #240. Alternatively or in addition, an overlapping partitioning may be applied, where the same part of the content is included in multiple slices.”

		certain byte range, for example bytes 1000 to 1020)…"	
36	36. (proposed substitute for claim 24) ~~24.~~ The method according to claim ~~15~~[[29]], wherein the partitioning is non-sequential in the fresh content.	E.g., Page 13: "The Agent then assigns the chunks of data to each Peer in the following manner: starting with the first Peer in the list {8010}, which is the fastest Peer because of the sorting in 8004, it then assigns a number of chunks to the Peer {8012) by using a metric of allocation. For example, by providing a set number of chunks per Peer, or by assigning for example BW/RTT chunks to the Peer. If there are more chunks of the data left to assign {8014), then the next Peer in the list is selected {8016), and more chunks are assigned to the Peer in 8012, and so on until all chunks are assigned, and the allocation function ends {8018)." E.g., Page 7: "In 6902, the requested data is split in to ranges, such that there is one range per participating Tunnel. It is possible to simply choose ranges of equal size to implement this. A possibly more efficient implementation is to provide each Tunnel a range that is proportional to that Tunnel's ability to provide the data (for example, based on that Tunnel's upstream and downstream traffic to the	See, e.g., original claim 143 E.g., Page 76: "The partitioning may be sequential or non-sequential in the content…" E.g., Page 101: "However, in case where two or more steps are not explicitly described as being sequentially executed, these steps may be executed in any order, or may be simultaneously performed." E.g., Page 120: "The tasks relating to the different data paths, such as shown in a flowchart 100 a, relating to communicating with the multiple tunnel devices and/or direct access, may be executed sequentially or in parallel." E.g., Page 82: "The partitioning may be sequential or non-sequential in the content, and the number of the content slices may be equal to, higher than, or lower than, the number of devices in the group." E.g., Page 139: "The partitioning may be non-overlapping, wherein each slice

		Client, as measured in previous communications between them)."	includes a distinct part of the content, as is exampled above in a case wherein the content includes 240 bytes designated byte #1 to byte #240, where using a byte level partitioning into three slices (referred as slice #1, slice #2, and slice #3), results in a first slice (slice #1) including byte #1 to byte #80, a second slice (slice #2) including byte #81 to byte #160, and a third slice (slice #3) including byte #161 to byte #240. Alternatively or in addition, an overlapping partitioning may be applied, where the same part of the content is included in multiple slices."
37	37. (proposed substitute for claim 27) ~~27.~~ The method according to claim ~~15~~[[29]], wherein the number of content slices is higher than the number of tunnel devices in the group.	E.g., Page 33: "In this example, the client 2 loads these chunks from 3 available sources, fetching more chunks from faster sources, so that 3 chunks are requested from source 117002, 1 chunk from source 2 17004 and 2 chunks from source 3 17006." E.g., Page 13: "The Agent then assigns the chunks of data to each Peer in the following manner: starting with the first Peer in the list {8010}, which is the fastest Peer because of the sorting in 8004, it then assigns a number of chunks to the Peer {8012) by using a metric of allocation. For example, by providing a set number of chunks per Peer, or by assigning for	See, e.g., original claim 146 E.g., Page 82: "The partitioning may be sequential or non-sequential in the content, and the number of the content slices may be equal to, higher than, or lower than, the number of devices in the group." E.g., Page 82: "Two or more of the content slices may include the same information. Further, the same part of the content may be included in two or more content slices." E.g., Page 84: "Part of, or all of, the content parts may be having the same

		example BW/RTT chunks to the Peer. If there are more chunks of the data left to assign {8014), then the next Peer in the list is selected {8016), and more chunks are assigned to the Peer in 8012, and so on until all chunks are assigned, and the allocation function ends {8018)." See also, e.g., Fig. 136 and associated description on Page 23: "Figure 136 describes such a system. In it, the Client 2 would like to open more connections with the Data Server 5 than the Data Server 5 will allow (***where each of these connections is loading one or more requests, where each request is a resource, a range of a resource, or multiple resources***). The Client then communicates with participating Tunnels 4 (i.e. network devices on which software is installed for this purpose), and requesting from each of these Tunnels to load several of these requests, such that the each of the participating elements (the Client and the Tunnels from which it requested the loads) does not open more connections vs. the Data Server 5 that the Data Server 5 allows." (emphasis added) See, e.g., Fig. 70 and associated description on Pages 7-8: "Figure 70 shows an example of how a tunnel may be implemented. The Tunnel receives a	size, that may be 8 KB, 16 KB, 32 KB, or 64 KB. Two or more content parts may be identical and may contain the same data." E.g., Page 126: "The same content may be requested and fetched using multiple tunnel devices as exampled above. Alternatively or in addition, the content may be partitioned into multiple slices (overlapping or non-overlapping), where each slice is requested and fetched using a distinct tunnel device (or via the client device serving as its own tunnel)." See also, e.g., Fig. 21 See also, e.g., Page 126: "It is noted that in such a partition, the content may be fully reconstructed from any two of the slices, hence providing a degree of redundancy. For example, in case of carrying the three slices over the Internet and a failure to receive one of the slices, the remaining two slices may be used to fully reconstruct the whole content."

		request (7002) from another network element (e.g. a Client) which includes the standard information of a data request (DR) including the address of the data server (such as the name or IP of a web server), the information requested (such as a URL), the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020), and additional information about the request (such as the cookies in HTIP). The Tunnel the[n] creates a request (7004) to the data server based on the DR and sends it to the data server. Once the response is received (7006}, the Tunnel sends the data (7010) back to the requesting Client." See also Pages 33-34 ("Microchunks" section)	
38	38. (proposed substitute for claim 28) ~~28.~~ The method according to claim ~~15~~[[29]], wherein the number of content slices is lower than the number of tunnel devices in the group.	E.g., Page 7: "…the range of the data that is requested (e.g. all of the data, or only a certain byte range, for example bytes 1000 to 1020)…" E.g., Page 50: "In figure 247, a method of transmitting packets in parallel over multiple routes is described, in which one communications device (NE1) wishes to communicate with another communications device (NE2), sending	See, e.g., original claim 147 E.g., Page 82: "The partitioning may be sequential or non-sequential in the content, and the number of the content slices may be equal to, higher than, or lower than, the number of devices in the group." E.g., Page 82: "Two or more of the content slices may include the same

		the same data over multiple routes in parallel." E.g., Page 33: "In 17402 the client is going to load a file which is available bia multiple sources (source(1) ... source(n_sources). In 17404, the requested file is split up in toN chunks of equal size (chunk_size) that was pre-determined in the system. For example, this chunk size can be 16KB." See also, e.g., Fig. 136 and associated description on Page 23: "Figure 136 describes such a system. In it, the Client 2 would like to open more connections with the Data Server 5 than the Data Server 5 will allow (***where each of these connections is loading one or more requests, where each request is a resource, a range of a resource, or multiple resources***). The Client then communicates with participating Tunnels 4 (i.e. network devices on which software is installed for this purpose), and requesting from each of these Tunnels to load several of these requests, such that the each of the participating elements (the Client and the Tunnels from which it requested the loads) does not open more connections vs. the Data Server 5 that the Data Server 5 allows." (emphasis added)	information. Further, the same part of the content may be included in two or more content slices." E.g., Page 84: "Part of, or all of, the content parts may be having the same size, that may be 8 KB, 16 KB, 32 KB, or 64 KB. Two or more content parts may be identical and may contain the same data." E.g., Page 126: "The same content may be requested and fetched using multiple tunnel devices as exampled above. Alternatively or in addition, the content may be partitioned into multiple slices (overlapping or non-overlapping), where each slice is requested and fetched using a distinct tunnel device (or via the client device serving as its own tunnel)." See also, e.g., Fig. 21 See also, e.g., Page 126: "It is noted that in such a partition, the content may be fully reconstructed from any two of the slices, hence providing a degree of redundancy. For example, in case of carrying the three slices over the Internet and a failure to receive one of the slices, the remaining two slices may be used to fully reconstruct the whole content."

		See also Pages 33-34 (“Microchunks” section)	