# SYSTEM AND METHOD FOR DETERMINING A DESIGNATED CONNECTION BETWEEN COMPONENTS OF COMPUTING DEVICES

**[0001]** Embodiments of the invention relate generally to data communications between components of computing devices, and more specifically to a system and method for identifying a connection between the components to be used for such communications.

**[0002]** In a typical distributed computing environment, different components and objects that comprise one or more applications reside on different computing devices, connected to each other in a network. Information and processing capability may be distributed amongst the computing devices in the network, and each of the computing devices may be permitted access to the others as needed. Components may communicate with each other through a connection for data communications (e.g. a Transmission Control Protocol (TCP) connection), or using some other communications protocol (e.g. User Datagram Protocol (UDP)).

**[0003]** Consider, for example, the situation where peer-to-peer (P2P) communication is to take place between two components residing on different computing devices. A connection may be established between the two components to facilitate communications. Since the components are in a peer-to-peer relationship and not in a client-server relationship, both components have equal status in the arrangement and each of the two components is independently capable of initiating the establishment of such connections. Since each of the two components may establish a connection with the other independently, it is possible that multiple connections between the same two components may be established and may exist simultaneously.

**[0004]** However, managing multiple connections between the two components may be unnecessary and may result in a waste of resources, particularly when each of the connections is adapted to facilitate bi-directional data communications (e.g. TCP).

**Summary**

**[0005]** Embodiments of the invention relate generally to a technique for determining a single, designated connection to be used for peer-to-peer communications between two components, each residing on a computing device.

**[0006]** In a broad aspect of the invention, there is provided a system for determining a designated connection between a first component residing on a computing device and a second component residing on a computing device, each component capable of initiating the establishment of connections that facilitate data communication to the other, wherein the system comprises a first application and a second application associated with the first and second components respectively, and wherein the first and second applications are programmed to co-operate to perform the steps of: generating a component identifier for each of the first and second components in accordance with an identifier generating algorithm; identifying a current connection between the first and second components; transmitting and receiving component identifiers between each of the first and second components to and from the other; comparing the component identifiers for the first and second components to identify one of the first and second components as a controller and the other as a responder; and determining a designated connection between the first and second components, the step initiated by the component identified as the controller and comprising the substeps of determining whether at least one connection other than the current connection exists between the first and second components that neither the first component nor the second component knows to be broken, if the at least one connection other than the current connection does not exist, identifying the current connection as the designated connection, and if the at least one connection other than the current connection exists, identifying a connection as the designated connection from a group that includes the current connection and the at least one connection other than the current connection; wherein the identifier generating algorithm is adapted to ensure that component identifiers generated by the algorithm are orderable and substantially unique.

**[0007]** In another broad aspect of the invention, there is provided a method of determining a designated connection between a first component residing on a computing device and a second component residing on a computing device, each

component capable of initiating the establishment of connections that facilitate data communication to the other, and wherein the method comprises the following steps performed at each of the first and second components: generating a component identifier for the respective component in accordance with an identifier generating algorithm; identifying a current connection between the respective component and the other of the first and second components; transmitting the component identifier for the respective component to the other component, and receiving a component identifier for the other component from the other component; performing a comparison of the component identifiers for the respective component and the other component, to identify the respective component as one of a controller and a responder; and determining a designated connection between the respective component and the other component, the step initiated by the respective component if the respective component is identified as the controller and comprising the substeps of determining whether at least one connection other than the current connection exists between the respective component and the other component that neither the respective component nor the other component knows to be broken, if the at least one connection other than the current connection does not exist, identifying the current connection as the designated connection, and if the at least one connection other than the current connection exists, identifying a connection as the designated connection from a group that includes the current connection and the at least one connection other than the current connection; wherein the identifier generating algorithm is adapted to ensure that component identifiers generated by the algorithm are orderable and substantially unique.

## Brief Description of the Drawings

**[0008]** For a better understanding of embodiments of the invention, and to show more clearly how they may be carried into effect, reference will now be made, by way of example, to the accompanying drawings in which:

FIG. 1 is a schematic diagram illustrating a subset of elements of computing devices in an example network configuration;

FIG. 2 is a schematic diagram of an example of a distributed system comprising four components;

FIGS. 3A and 3B are flowcharts that illustrate steps in a method of determining a designated connection between a first component and a second component in an embodiment of the invention; and

FIG. 4 is a diagram illustrating different examples of connection states between two components.

**Detailed Description**

**[0009]** Embodiments of the invention relate generally to a technique for determining a single, designated connection to be used for peer-to-peer communications between two components, each residing on a computing device. More specifically, embodiments of the invention are generally directed to a technique for identifying a designated connection from one or more connections that may have been established between the two components.

**[0010]** Where the relationship between the two components is peer-to-peer, it is possible that two (or more) connections may be established between them. In order for one connection to be determined as the designated connection, a reliable way of choosing the connection to be selected is required. To facilitate this, in accordance with at least one embodiment of the invention, each component is associated with a component identifier that is generated by an identifier generating algorithm adapted to ensure that component identifiers generated by the algorithm are orderable and substantially unique. This allows a comparison of the component identifiers for any two given components in a distributed system to be made, and allows one component to be identified as a controller and the other component to be identified as a responder. Therefore, when a method of determining a designated connection is performed, serialization of operations can be enforced and race conditions that might otherwise be caused by the two components can be avoided, with each component maintaining its role as controller or responder throughout the performance of the method.

**[0011]** Furthermore, in order for one connection to be determined as the designated connection, a reliable way of tracking connections that have been established between a given component and other components is required. The component identifier with which each component is associated allows components in a distributed system to be easily identified for this purpose. Since

the component identifiers generated by the identifier generating algorithm are orderable and unique, each component in the distributed system can be uniquely identified.

**[0012]** In an embodiment of the invention, each component identifier is a Universally Unique Identifier (UUID). In contrast, an Internet Protocol ("IP") address or identifier, for example, would not be reliable for the purpose of tracking connections between two specific components, as an IP identifier may be unable to uniquely identify each component in the distributed system. For example, a computing device can host multiple components, and therefore, different components may be assigned the same IP identifier. Moreover, any given component may be assigned multiple IP identifiers.

**[0013]** In an embodiment of the invention, each connection that is established between the two components is a TCP connection. While peer-to-peer communications may be implemented using other communication protocols (e.g. UDP), the functionality provided by TCP with respect to flow control, network adaptation, and ensuring reliable delivery of transmitted data, for example, may provide particular benefits where TCP connections are employed. Furthermore, TCP almost always operates full-duplex, in that bi-directional data communications are supported. In contrast to an implementation where two TCP connections between two components are used for data communications (e.g. where one TCP connection is used for data transmission for each direction), the use of a single, designated TCP connection can simplify software application development, making it easier to monitor traffic between a given component and other components, and making it easier to manage the establishment and termination of connections between those components.

**[0014]** Referring now to FIG. 1, a schematic diagram illustrating a subset of elements of computing devices in an example network configuration is shown generally as 10.

**[0015]** In the example network configuration 10, two computing devices 12 are connected to each other by a network connection 14 over which computing devices 12 can communicate. Although only two computing devices are shown in network 10 of FIG. 1, more than two computing devices may be connected in a network, and they may be arranged in any of a number of network topologies in

variant implementations. It will be understood by persons skilled in the art that network connection 14 need not be a wired connection, as any given connection in the network may instead be a wireless communication link, for example. It will also be understood that other devices may be connected either directly or indirectly to computing devices 12 in network 10 that are not shown in FIG. 1.

**[0016]** A subset of elements of each computing device 12 is shown in FIG. 1. It will be understood by persons skilled in the art that the elements shown in FIG. 1 are provided by way of example only, and computing device 12 will generally comprise other elements in addition to those shown in FIG. 1. Each computing device 12 comprises a processor 16, an optional user interface 18 for receiving user input and/or providing output, memory 20, and a network interface 24 that facilitates network communications with other computing devices in network 10. Processor 16 executes instructions in accordance with system software and other applications residing in memory 20. It will be understood that the term "memory" as used herein is not limited to a single storage device and may comprise volatile and non-volatile memory elements, as well as secondary storage elements, residing on computing device 12 or otherwise coupled thereto.

**[0017]** Referring to FIG. 2, a schematic diagram of an example of a distributed system comprising four components is shown generally as 30.

**[0018]** In FIG. 2, distributed system 30 comprises four components 32 that have connections 34 for data communications established between them. In this example implementation, connections 34 are TCP connections. It will be understood by persons skilled in the art that distributed system 30 may comprise more or less connected components 32, and that connections 34 may be of different types in variant implementations.

**[0019]** It will be understood by persons skilled in the art that a "component" as described herein need not refer to a particular computing device, machine, or device associated with an IP identifier, for example. In at least one embodiment of the invention, a "component" is a process executing on a computing device. For instance, in the example of FIG. 2, each component 32 is a process executing on a computing device (e.g. computing device 12 of FIG. 1). In this specification and in the claims, a component 32 that is a process executing on a particular computing device is considered to reside on that computing device.

**[0020]** Several components 32 may reside on the same computing device or on different computing devices. For instance, in the example of FIG. 2, the two leftmost components 32 shown schematically in the Figure may reside on one computing device while the two rightmost components 32 shown schematically in the Figure may both reside on a different computing device. Where two given components 32 reside on different computing devices and are to communicate with each other, a connection 34 may be established between them for this purpose. In this embodiment of the invention, peer-to-peer communications between components 32 that reside on different computing devices take place via connections 34.

**[0021]** Although the embodiments herein, provided by way of example, are described as relating generally to connections between components that reside on different computing devices, in variant embodiments, a connection 34 may also be established between components 32 residing on the same computing device. For example, TCP may also be used to connect components residing on the same computing device, as it may also function as an interprocess protocol. In this case, TCP may interface with users and applications through an interface (e.g. a loop-back interface) that consists of a set of calls that are similar to the calls that an operating system provides to an application process for manipulating files.

**[0022]** In one embodiment, each component 32 is a process that may be considered as having at least two layers: an application processing layer 36 and a TCP library (TCPLIB) layer 38. The TCPLIB layer 38 of a component 32 is used to set up an actual TCP connection 34 between the component 32 and another component when such a connection is required for use by the application processing layer 36. The components 32 can then communicate with each other over the established connections 34.

**[0023]** In accordance with an embodiment of the invention, a method of determining a designated connection between two components 32, as will be described with reference to FIGS. 3A and 3B for example, can be implemented to ensure that only one TCP connection 34 is used to facilitate data communications between any two given components 32. In the example of FIG. 2, the method is implemented through the TCPLIB layer 38 of the components 32. The application

processing layer 36 of a particular component 32 uses the respective designated TCP connection 34 set up by the TCPLIB layer 38 of that component 32 to communicate with another component 32 in distributed system 30.

**[0024]** Referring to FIG. 3A, a flowchart illustrating steps in a method of determining a designated connection between a first component and a second component in an embodiment of the invention is shown generally as 50.

**[0025]** Method 50 is directed to a technique for determining a single, designated connection to be used for peer-to-peer data communications between two components (e.g. component 32 of FIG. 2). In this embodiment, each of the two components resides on a different computing device (e.g. computing device 12 of FIG. 1); however, as noted above, in variant embodiments, the two components may reside on the same computing device.

**[0026]** The components are in a peer-to-peer relationship, and each of the two components is independently capable of initiating the establishment of a connection to the other. In this embodiment, connections established between the two components are TCP connections.

**[0027]** In one example software implementation, each of the first and second components is implemented as a process on a computing device, as generally described with reference to FIG. 2. In particular, each process is an executing instance of an application that has been programmed to perform the steps of method 50 that are to be carried out by the respective process on the respective computing device. Further, the applications are programmed to co-operate in order to exchange data between the first and second components and perform the necessary processing at each of these components, in order to facilitate the identification of a single connection between the components as the designated connection.

**[0028]** Some of the steps of method 50 are performed at the first component (identified as component "A"), while other steps of method 50 are performed at the second component (identified as component "B"), as shown in FIG. 3A and as described in further detail below.

**[0029]** At step 52a, a component identifier is generated for component A in accordance with an identifier generating algorithm that is adapted to ensure that component identifiers generated by the algorithm are orderable and unique.

Similarly, at step 52b, a component identifier is generated for component B in accordance with the same identifier generating algorithm. These component identifiers will typically be stored in memory (e.g. memory 20 of FIG. 1) of the computing device (e.g. computing device 12 of FIG. 1) on which the respective component resides. These component identifiers may be generated upon activation of the TCPLIB layer (e.g. 38 of FIG. 2) of each component, for example. In this embodiment, a component identifier is to be generated for each component, and remains associated with the respective component for its entire lifetime.

**[0030]** In an embodiment of the invention, each component identifier is generated in accordance with an algorithm adapted to generate Universally Unique Identifiers (UUIDs), also known as Globally Unique Identifiers (GUIDs). A number of known UUID specifications and formats exist, including the Open Software Foundation's (OSF) (currently known as The Open Group) Distributed Computing Environment specification, for example.

**[0031]** A UUID is typically a 128-bit long identifier that requires no central registration process. A UUID is substantially unique across both space and time, with respect to the space of all UUIDs. Each UUID is considered to be "substantially" unique as it is possible for UUID values to rollover (e.g. around A.D. 3400, depending on the specific algorithm used). However, for practical purposes, each generated UUID can be considered to be unique.

**[0032]** While the concept of UUIDs is generally known, the UUIDs are used in a novel and inventive manner in this embodiment of the invention to uniquely identify components in a distributed system and to facilitate the identification of designated connections between components. Each component (e.g. process) in the distributed system may be assigned one UUID for the term of its life. These unique identifiers enable components to be uniquely identified, irrespective of whether the components reside on the same computing device or whether the components are associated with the same IP identifier.

**[0033]** UUIDs are typically of a fixed size that is reasonably small relative to other alternative identifiers, and accordingly, they are easily sorted, ordered, and stored, for example. Since the UUIDs are orderable, this allows a component that has had a UUID associated with it to be assigned a relative rank or position vis-à-

vis other components. In this embodiment, as between two components for example, the UUIDs facilitate the consistent identification of one component as a controller and the other component as a responder in the performance of method 50. Accordingly, situations where both components compete to perform specific steps or tasks can be avoided.

**[0034]** In one embodiment of the invention, a connection table is maintained by each component (e.g. in memory 20 of the computing device 12 upon which the component resides as shown in FIG. 1) of the distributed system to store data for tracking all connections connected to the respective component. In variant embodiments, other data structures may be used to store information associated with the connections.

**[0035]** More specifically, the connection table maintained by each component stores, for each connection, the component identifier (e.g. UUID) of the component to which the respective component is connected, an identifier associated with the connection itself (e.g. a socket handle for the TCP connection), and a status flag. At any given time, the status flag may be set to one of three values: "created", "pending", or "connected", in this embodiment. Different values may be assigned to the status flag, and other connection data may also be stored in the connection table in variant embodiments.

**[0036]** When a connection is established between any two components at the TCP level, as may be initiated by either of the two components, steps 54a and 54b are performed. At step 54a, at component A, the TCP connection that has been established with component B is identified, and a record is created in the connection table maintained by component A. This record includes the connection's identifier, and the status flag is set to "created".

**[0037]** Similarly, at step 54b, the connection is also identified at component B, and a record is created in the connection table maintained by component B. This record also includes the connection's identifier, and the status flag is set to "created".

**[0038]** Performance of subsequent steps of method 50 is triggered by the establishment of the connection identified at component A (step 54a) and at component B (step 54b). In respect of the present instance of method 50, the connection is referred to as the "current" connection.

**[0039]** It will be understood by persons skilled in the art that multiple instances of method 50 may be performed concurrently where multiple established connections can co-exist. Consider an example scenario where a first connection is established (e.g. at the TCP level) between two components. This first connection would be identified at steps 54a and 54b by the two components in a first instance of method 50, and at each component, the existence of this first connection is recorded in the connection table maintained by the respective component, with this first connection being initially flagged as "created". Performance of subsequent steps of this first instance of method 50 (e.g. 56a/56b to 62) is triggered so that a determination of whether this first connection should be identified as the designated connection can be made.

**[0040]** Now suppose that a second connection between the same two components were to be established while the determination with respect to the first connection is still pending. A number of steps of a second instance of method 50 may be performed while the first instance of method 50 continues to be performed. For example, the second connection would be identified at steps 54a and 54b by the two components in the second instance of method 50, and at each component, the existence of this second connection is recorded in the connection table maintained by the respective component, with the second connection being initially flagged as "created". Performance of subsequent steps of this second instance of method 50 is triggered so that a determination of whether this second connection should be identified as the designated connection can be made. As the first connection takes the role of the "current" connection in respect of the first instance of method 50, the second connection takes the role of the "current" connection in respect of the second instance of method 50.

**[0041]** Similarly, further instances of method 50 may be performed with respect to additional connections concurrently as further connections are established, and each of those additional connections will take the role of the "current" connection in respect of its own instance of method 50.

**[0042]** While certain steps of multiple instances of method 50 may be performed concurrently, step 52a will be performed only once by component A and step 52b will be performed only once by component B, since only one component identifier is to be generated for each component.

**[0043]** Referring again to FIG. 3A, at step 56a, the component identifier for component A is transmitted to component B. At step 58b, the component identifier for component A is received by component B and stored in component B's connection table, in the "created" record associated with the current connection. Similarly, at step 56b, the component identifier for component B is transmitted to component A. At step 58a, the component identifier for component B is received by component A and stored in component A's connection table, in the "created" record associated with the current connection. The dashes in FIG. 3A illustrate that there is a first exchange of data (i.e. the component identifiers are exchanged) between components A and B when these steps are performed. In one embodiment, the component identifiers are exchanged over the current connection identified at steps 54a and 54b, although other means for exchanging the component identifiers may be employed in variant embodiments.

**[0044]** At step 60a, at component A, the component identifier for component A is compared to the component identifier for component B received at step 58a, to identify component A as either the controller or the responder in the performance of subsequent steps of method 50. Similarly, at step 60b, at component B, the component identifier for component B is compared to the component identifier for component A received at step 58b, to identify component B as either the controller or the responder in the performance of subsequent steps of method 50. In the specification and in the claims, the component identified as the controller is referred to generally as "the controller", and the component identified as the responder is referred to generally as "the responder", in respect of the particular current connection.

**[0045]** In one embodiment, the component identifiers are UUIDs. Since the UUIDs for components A and B are orderable and unique across the space of all UUIDs, it is possible to consistently identify one component as the controller and the other component as the responder. Any technique to assign the controller and responder roles to the components based on the relative positions of the identifiers in the ordering may be employed, so long as the techniques used at both components provide consistent results.

**[0046]** For example, in one embodiment, the component having the higher-valued UUID as determined by a comparison operation may be identified as the

controller, while the component having the lower-valued UUID as determined by the same comparison operation may be identified as the responder. Conversely, the component having the lower-valued UUID may be identified as the controller, while the component having the higher-valued UUID may be identified as the responder in a variant embodiment. In variant embodiments, other techniques and algorithms to assign the controller and responder roles to any two components in respect of a particular connection may be used, and the ordering of component identifiers may be defined in different ways.

**[0047]** At step 62, a determination is made as to whether the current connection is to be identified as the single, designated connection for subsequent data communication between components A and B. This generally requires initially determining whether or not there are any established connections between components A and B, other than the current connection, that neither component A nor component B knows to be broken. If no such connections exist, then the current connection, being the only connection between components A and B not known to be broken, can be identified as the designated connection for subsequent data communication.

**[0048]** On the other hand, if an established connection between components A and B, other than the current connection, exists that is not known to be broken (i.e. by component A, by component B, or by both components), then both the current connection and the other connection are candidates for the designated connection, and a choice must be made to identify either the current connection or the other established connection as the designated connection for subsequent data communication. The other established connection may be a connection that has previously been identified as the designated connection. In one embodiment, the current connection would not be identified as the designated connection if a designated connection between components A and B not known to be broken already exists. However, in variant embodiments, other protocols may be implemented. For example, in variant embodiments, the current connection may always be identified as the designated connection, taking precedence over any earlier established connection.

**[0049]** The connection identified as the designated connection will have its associated record in the connection tables of both components updated (if

necessary) to reflect a "connected" status, by setting the status flag for that connection in both tables appropriately. Any other established connection between the two components that is considered as a candidate for the designated connection, but is not ultimately identified as the designated connection, may be terminated at step 62, and the records in the connection tables of both components associated with that terminated connection removed. The terminated connection may include the current connection if it is not identified as the designated connection, or it may include a previously designated connection, for example. A previously designated connection that is terminated may be one that has become broken since it was designated, for example, as discussed in greater detail below.

**[0050]** Referring again to step 62, in this embodiment of the invention, the candidates for the single, designated connection are connections that neither component A nor component B knows to be broken. If either component A or component B knows that a particular connection is broken, it would be inefficient to identify that connection as the designated connection for data communication.

**[0051]** The term "broken" is used herein to describe one possible state of connection between two components. A connection between two components, including in particular a designated connection between the components, may become broken in any of a number of ways. For example, a network cable at one of the computing devices on which a component resides may become physically disconnected from the computing device, making it impossible (at least until the network cable is reconnected to the computing device) for data to be communicated between the devices over a connection using that cable. In the normal course of communications between the components, a component may discover the connection to be broken. For example, when an attempt is made by one component to send data to the other component using the connection and where a response from the other component is expected, failure to receive the response in a timely manner may indicate to the component sending the data that the connection is broken. Connections that are discovered by a component to be broken can be terminated unilaterally by that component, and the record of the terminated connection is removed from the connection table maintained by that component [steps not explicitly shown in Figures].

**[0052]** However, depending on whether a given component has made a failed attempt to communicate with the other component or tested a connection while the connection is broken, it is possible that the connection would be known to be broken (and subsequently terminated) by only one of the two components. It is also possible that neither component may know that a connection is broken if no communication by either component has been attempted over that connection while the connection is broken. For the purposes of the specification and the claims, a broken connection is still considered to be a connection between two components, so long as the broken connection is not known to be broken and has not been terminated by at least one of the two components.

**[0053]** Accordingly, in this embodiment, step 62 requires confirmation that a connection that is to be considered as a candidate for the designated connection is one that neither component A nor component B knows to be broken. Referring to FIG. 3B, step 62 of FIG. 3A is described in greater detail as implemented in accordance with one embodiment of the invention.

**[0054]** Some of the substeps that comprise step 62 in this embodiment are performed by the component identified as the controller (which is one of component A and component B), while other substeps are performed by the component identified as the responder (which is the other of component A and component B) as illustrated in FIG. 3B. The identification of one of the components as the controller and the other as the responder was performed in steps 60a and 60b of FIG. 3A. As shown in FIG. 3B, the following substeps of step 62 are initiated by the controller.

**[0055]** At substep 64, a determination is made by the controller as to whether an earlier designated connection to the responder is known by the controller to exist (i.e. has not been terminated by the controller). In one embodiment, this substep comprises determining whether a record of a connection, other than the current connection, to a component identified by the UUID of the responder exists in the connection table maintained by the controller, and where the flag for that other connection is set to "connected".

**[0056]** If no earlier designated connection is known by the controller to exist, then the current connection is identified as the designated connection at substep 66, and the record of the current connection to the responder (the

responder being identified by the responder's UUID) is updated in the connection table maintained by the controller, with the value of the status flag being changed from "created" to "connected".

**[0057]** Where multiple connections may be established and exist simultaneously, a number of instances of method 50 (and step 62) may be performed concurrently. In this case, implementations of this embodiment of the invention should ensure that access to the connection table maintained by the controller in order to make the determination in substep 64 is restricted, and that substeps 64 and 66 are completed in a single transaction if substep 66 is performed, so as to prevent more than one "current" connection from being identified as the designated connection. Different programming constructs may be employed (e.g. locks, etc.) to achieve this control in variant implementations.

**[0058]** An indication that the current connection has been identified as the designated connection is then transmitted to the responder at substep 68. This indication is received by the responder at substep 70 as illustrated by the dashed line, depicting a second transmission of data from component A to component B (the first transmission was part of a data exchange made earlier as discussed with reference to FIG. 3A).

**[0059]** Even where no earlier designated connection is known by the controller to exist, it may be that a broken connection exists, previously detected by the controller and terminated, but which is not yet known by the responder to be broken and has not yet been terminated by the responder. From the responder's point of view, the broken connection may still be the designated connection, and that broken connection may be identified as such in the connection table maintained by the responder. Accordingly, at substep 72, the responder checks whether such a broken connection exists by identifying whether a record of an established connection (other than the current connection) to a component with the same UUID as the controller is recorded and flagged as "connected" in the connection table maintained by the responder. If so, this broken connection can be terminated at substep 72, and the record of the broken connection removed from the connection table. The record of the current connection to the controller in the connection table maintained by the responder is

updated, with the value of the associated status flag being changed from "created" to "connected".

**[0060]** Referring again to the determination made at substep 64, if, on the other hand, an earlier designated connection to the responder is known by the controller to exist, then at substep 74, a determination is made as to whether an earlier pending connection to the responder is also known by the controller to exist.

**[0061]** When an earlier designated connection is known by the controller to exist, it may ultimately be replaced by a current connection as the designated connection if the earlier connection is subsequently discovered to be broken (i.e. the responder terminated the connection). Until it is confirmed that the earlier connection is not known by the responder to be broken, the current connection remains an active candidate for the designated connection. However, there may be multiple "current" connections active at a given time, being considered by the controller. Therefore, in one embodiment, at most one of the "current" connections is flagged as "pending", to prevent multiple connections from being identified as the designated connection, in the event that the earlier designated connection is in fact, ultimately discovered to be broken. In effect, the first "current" connection to be flagged as pending will be considered as the "first in line" to replace the earlier connection as the designated connection. All other connections that are neither "pending" nor "connected" would no longer be considered as candidates for the designated connection, and may be terminated.

**[0062]** Accordingly, if an earlier established connection is pending, then the current connection is terminated at substep 76 in favor of the pending connection, and the record of the current connection in the connection table maintained by the controller is removed. When the current connection is terminated by the controller, the underlying TCP will transmit a disconnect signal automatically to the responder, and the current connection will be terminated by the responder, with the record of the current connection in the connection table maintained by the responder being removed [steps not shown].

**[0063]** In this manner, newer connections will typically be eliminated from consideration in favor of earlier established connections, as instances of the steps of method 50 with respect to the earlier connection first flagged as "pending" will

continue to be performed. This approach may generally be considered as efficient, since the longest pending connection is likely to be closer to resolution in the determination of the designated connection, relative to more recently established connections.

**[0064]** However, in a variant embodiment of the invention, the current connection may be given priority over any earlier established pending or designated connections, with the current connection being identified as the designated connection, and with all earlier pending or designated connections being terminated.

**[0065]** Referring again to FIG. 3B, if it is determined (at substep 64) that an earlier established designated connection known by the controller exists, and that no additional connections are currently pending (as determined at substep 74), then the current connection is marked as pending by updating the value of the associated status flag in the connection table maintained by the controller, from "created" to "pending" at substep 78. Where multiple connections may be established and exist simultaneously, implementations of this embodiment of the invention should ensure that access to the connection table maintained by the controller in order to make the determination in substep 74 is restricted, and that substeps 74 and 78 are completed in a single transaction if substep 78 is performed, so as to prevent more than one "current" connection from being identified as a pending connection. Different programming constructs may be employed (e.g. locks, etc.) to achieve this control in variant implementations.

**[0066]** It is noted that between any two given components, it will not be possible that one connection will be considered as pending according to one component's connection table and that a different connection between the same two components will be considered as pending according to another component's table. If this were permitted to occur, then it would be possible that both pending connections may ultimately be identified as the designated connection, which would be an unsatisfactory result. However, this is avoided since orderable and unique identifiers are used to identify all components, and between any two components, one component will always be identified as the controller while the other component will always be identified as the responder vis-à-vis those two components. Accordingly, in method 50, connections will only be marked or

flagged as "pending" in the connection table maintained by the controller, at substep 78.

**[0067]** At substep 80, the responder is instructed to confirm that the earlier established connection known by the controller and identified as the designated connection still exists from the responder's perspective.

**[0068]** The confirmation is made by the responder at substep 82, after receiving the instruction from the controller, as indicated by the dashed line representing a first part of a second data exchange between component A and component B (a first data exchange was made earlier as discussed with reference to FIG. 3A). The responder checks its records of existing connections stored in the connection table that it maintains, to determine if a connection to a component with the same UUID as the controller has been flagged as "connected" and has not yet been terminated by the responder. If the earlier designated connection is known by the responder to still exist (i.e. the earlier designated connection is not known by the responder to be broken), then at substep 84, the current connection is terminated in favor of the existing designated connection, and the record of the current connection in the connection table maintained by the responder is removed. When the current connection is terminated by the responder, the underlying TCP will transmit a disconnect signal automatically to the controller, and the current connection will be terminated by the controller, with the record of the current connection in the connection table maintained by the controller being removed [steps not shown].

**[0069]** However, if the earlier designated connection is no longer known by the responder to exist, then at substep 86, the current connection is identified as the designated connection. The record of the current connection to the controller in the connection table maintained by the responder is updated, with the value of the status flag being changed from "created" to "connected". This substep would be performed when the controller has not been made aware that the earlier established connection, which had previously been identified as the designated connection, was broken and subsequently terminated by the responder.

**[0070]** A result of the confirmation that the current connection is to be identified as the new designated connection is returned to the controller at substep 88, which is received by the controller at substep 90, as indicated by the

dashed line representing a second part of the second data exchange between component A and component B. The record of the current connection to the responder in the connection table maintained by the controller is updated, with the value of the status flag being changed from "created" to "connected". The previously designated connection, now known to be broken, is terminated at substep 92, and the record of the broken connection is removed from the connection table maintained by the controller.

**[0071]** With reference to FIGS. 3A and 3B, it may be noted that whenever the steps of an instance of method 50 are performed to determine if a particular connection between any two components is to be identified as the designated connection, at most two data exchanges between the components are required to make the determination in accordance with this embodiment of the invention. Accordingly, method 50 may be implemented with relatively minimal overhead.

**[0072]** Upon completion of method 50, each component can consult the connection table that it maintains to determine the currently designated connection to any other component (e.g. the other component identified by its UUID, and the connection being flagged as "connected"), and use the designated connection for subsequent data communication to the other component.

**[0073]** Referring to FIG. 4, a diagram illustrating different examples of connection states between two components is provided to illustrate the application of method 50 described with reference to FIGS. 3A and 3B in a number of scenarios.

**[0074]** It will be understood by persons skilled in the art that this diagram is provided by way of example only, and other connection states not shown in FIG. 4 are possible in variant embodiments.

**[0075]** Connection state 100 depicts the current connection as being the only connection between components A and B. Application of method 50 in one embodiment will result in the current connection being identified as the designated connection.

**[0076]** Connection state 102 depicts a pending connection between components A and B in addition to the current connection. The pending connection is a candidate for the designated connection in respect of a different instance of method 50, along with a previously designated connection (broken or

unbroken) [not shown]. Application of method 50 in one embodiment in respect of the current connection will result in the current connection being terminated in favor of the pending connection. The pending connection may subsequently be identified as the designated connection when method 50 is applied in respect of that pending connection.

**[0077]** Connection state 104 depicts a previously designated connection between components A and B that is not broken, in addition to the current connection. Application of method 50 in one embodiment in respect of the current connection will result in the current connection being flagged as a pending connection by the controller (component A or B depending on their component identifiers), and then subsequently terminated in favor of the existing designated connection.

**[0078]** Connection state 106 depicts a broken connection between components A and B in addition to the current connection, where component B does not know that the connection has been broken and has already been terminated by component A. Application of method 50 in one embodiment will result in the current connection being identified as the designated connection, and the broken connection will be terminated by component B. It is noted that if component B were identified as the controller, then the current connection would be flagged as a pending connection by component B before it is ultimately identified as the designated connection.

**[0079]** Connection state 108 depicts a broken connection between components A and B in addition to the current connection, where component A does not know that the connection has been broken and has already been terminated by component B. Application of method 50 in one embodiment will result in the current connection being identified as the designated connection, and the broken connection will be terminated by component A. It is noted that if component A were identified as the controller, then the current connection would be flagged as a pending connection by component A before it is ultimately identified as the designated connection.

**[0080]** Connection state 110 depicts a broken, previously designated connection between components A and B in addition to the current connection, where neither component A nor component B knows that the connection is

broken. This scenario might arise if no communication by either component has been attempted over that connection while the connection is broken, for example. Application of method 50 in the embodiments previously described will result in the current connection being terminated in favor of the broken, previously designated connection. This is not necessarily an unsatisfactory result, as it may be considered reasonable to expect that a previously designated connection should be deemed to be "good" and should continue to be used, until at least one of the components discovers that the connection is actually broken, through the normal course of data communications using the connection.

**[0081]** However, in a variant embodiment of the invention, before a current connection is to be terminated in favor of a previously designated connection, the previously designated connection may be tested (e.g. ping/echo) to ensure that it is not broken prior to the termination of the current connection, in the event that the connection has not yet been discovered by either component to be broken. This may require at least one additional data exchange between the two components. In the event that the previously designated connection is discovered to be broken as a result of this test, it can then be terminated in favor of the current connection, which would then be identified as the new designated connection.

**[0082]** The steps of a method for determining a designated connection between a first component and a second component in embodiments of the invention may be provided as executable software instructions stored on computer-readable media, which may include transmission-type media.

**[0083]** The invention has been described with regard to a number of embodiments. However, it will be understood by persons skilled in the art that other variants and modifications may be made without departing from the scope of the invention as defined in the claims appended hereto.

**Claims:**

1. A system for performing a process between a first component residing on a computing device and a second component residing on a computing device, wherein the system comprises a first application and a second application associated with the first and second components respectively, and wherein the first and second applications are programmed to co-operate to perform the steps of:

associating a respective component identifier with each of the first and second components;

transmitting and receiving component identifiers between each of the first and second components to and from the other;

at each component, comparing the received component identifier with the identifier associated with that component in order to identify one of the first and second components as a controller and the other as a responder; and

maintaining the identified one of the first and second components as a controller and the other as a responder throughout the performance of the process.

2. The system of claim 1, wherein the respective component identifiers associated with the first and second components are substantially unique, orderable identifiers.

3. The system of claim 1 or claim 2, wherein the respective component identifiers associated with the first and second components are generated by an identifier generating algorithm.

4. The system of claim 3, wherein the same identifier generating algorithm is utilized at each of the first and second components to generate the respective identifiers for the first and second components.

5. The system of claim 3 or claim 4, wherein each component is capable of initiating the establishment of connections that facilitate data communication to the other and the first and second applications are programmed to co-operate to perform the further steps of:

a) identifying a current connection between the first and second components; and

b) determining a designated connection between the first and second components, said step initiated by the component identified as the controller and comprising the substeps of

(i) determining whether at least one connection other than the current connection exists between the first and second components that neither the first component nor the second component knows to be broken,

(ii) if said at least one connection other than the current connection does not exist, identifying the current connection as the designated connection, and

(iii) if said at least one connection other than the current connection exists, identifying a connection as the designated connection from a group that includes the current connection and the at least one connection other than the current connection.

6. The system of any one of claims 1 to 5, wherein the first and second components reside on different computing devices.

7. The system of any one of claims 1 to 5, wherein the first and second components reside on the same computing device.

8. The system of any one of claims 1 to 7, wherein each of the first and second components is a process executing on the computing device on which each respective component resides.

9. The system of any one of claims 1 to 8, wherein each component identifier is a Universally Unique Identifier.

10. The system of any one of claims 1 to 9, wherein the component of the first and second components with the higher-valued component identifier therefore is identified as the controller at the comparing step, and wherein the other component is identified as the responder at the comparing step.

11. The system of any one of claims 1 to 10, wherein any connection between the first and second components is a Transmission Control Protocol connection.

12. The system of claim 5, wherein the connection identified as the designated connection at substep iii) is a connection established before the current connection.

13. The system of claim 12, wherein the connection identified as the designated connection at substep iii) is a connection that has been previously identified as a designated connection.

14. The system of any one of claims 5, 12 or 13, wherein the first and second applications are programmed to co-operate to perform the further steps of:

c) identifying at least one connection not determined as the designated connection at step b), where such at least one connection exists; and

d) terminating the at least one connection identified at step c).

15. The system of claim 14, wherein the at least one connection identified at step c) and terminated at step d) includes a broken connection.

16. The system of any one of claims 5 and 12 to 15, wherein the system comprises first and second connection tables maintained by the first and second components respectively, and wherein the first and second connection tables store data pertaining to connections to other components from the first and

second components respectively, and wherein step b) comprises accessing said first and second connection tables when performing said determining step.

17. A method for performing a process between a first component residing on a computing device and a second component residing on a computing device, wherein the method comprises the following steps performed at each of the first and second components:

associating a component identifier with the respective component;

transmitting the component identifier for the respective component to the other component, and receiving a component identifier for the other component from the other component;

performing a comparison of the component identifiers for the respective component and the other component, to identify the respective component as one of a controller and a responder; and

maintaining the respective component as the identified one of a controller and responder throughout the performance of the process.

18. The method of claim 17, wherein the respective component identifiers associated with the first and second components are substantially unique, orderable identifiers.

19. The method of claim 17 or claim 18, wherein the respective component identifiers associated with the first and second components are generated by an identifier generating algorithm.

20. The method of claim 19, wherein the same identifier generating algorithm is utilized at each of the first and second components to generate the respective identifiers for the first and second components.

21. The method of claim 19 or claim 20, wherein each component is capable of initiating the establishment of connections that facilitate data communication to the other and method includes the further steps of:

a) identifying a current connection between the first and second components; and

b) determining a designated connection between the first and second components, said step initiated by the component identified as the controller and comprising the substeps of

(i) determining whether at least one connection other than the current connection exists between the first and second components that neither the first component nor the second component knows to be broken,

(ii) if said at least one connection other than the current connection does not exist, identifying the current connection as the designated connection, and

(iii) if said at least one connection other than the current connection exists, identifying a connection as the designated connection from a group that includes the current connection and the at least one connection other than the current connection.

22. The method of any one of claims 17 to 21, wherein the first and second components reside on different computing devices.

23. The method of any one of claims 17 to 21, wherein the first and second components reside on the same computing device.

24. The method of any of claims 17 to 23, wherein each of the first and second components is a process executing on the computing device on which each respective component resides.

25. The method of any one of claims 17 to 24, wherein the component identifier generated for the respective component is a Universally Unique Identifier.

26. The method of one of claims 17 to 25, wherein the respective component is identified as the controller at the comparing step if the component identifier for the respective component is higher-valued than the component identifier for the other component.

27. The method of one of claims 17 to 26, wherein any connection between the forst and second components is a Transmission Control Protocol connection.

28. The method of claim 21, wherein the connection identified as the designated connection at substep iii) is a connection established before the current connection.

29. The method of claim 28, wherein the connection identified as the designated connection at substep iii) is a connection that has been previously identified as a designated connection.

30. The method of any of claims 21, 28 or 29, further comprising the steps of:

c) identifying at least one connection not determined as the designated connection at step f), where such at least one connection exists; and

d) terminating the at least one connection identified at step g).

31. The method of claim 30, wherein the at least one connection identified at step c) and terminated at step d) includes a broken connection.

32. The method of any one of claims 21 or 28 to 31, wherein step b) comprises accessing first and second connection tables maintained by the first and second components respectively when performing said determining step, wherein the first and second connection tables store data pertaining to connections to other components from the first and second components respectively.

33. A computer-readable medium upon which a program executable on at least one of the first and second computing devices is stored, wherein the medium

comprises instructions for performing the steps of the method as claimed in any of claims 17 to 32.