

In at least some embodiments, the successor node in the consistent hash ring serves as the secondary flow tracker for the packet flow.

[0054] Figure 3 shows an example load balancer (LB) node 110 that includes components that implement all three roles (ingress, egress, and flow tracker), according to at least some embodiments. In this example, an ingress server 112 component performs the ingress role of receiving inbound TCP packets from client(s) and sending the TCP packets as encapsulated packets to the server(s). An egress server 114 component performs the egress role of receiving outbound encapsulated packets from the server(s) and sending the decapsulated TCP packets on to the client(s). A flow tracker 116 component performs as a primary or secondary flow tracker for one or more packet flows that are established between the client(s) 160 and the server(s) 134. The ingress server 112 may also communicate with the flow tracker 116 on load balancer node 110, or with the flow tracker 116 on another load balancer node 110 to initiate a TCP connection between a client and one of the servers 134 in response to a connection request received from the respective client 160, or to obtain mapping information for the packet flow. If the ingress server does

Load balancer nodes

[0055] Referring again to Figure 1, in at least some embodiments, the load balancer nodes 110 in the load balancer node layer receive client traffic (packets, e.g. TCP packets) from one or more routers 104 on the network and encapsulate the packets according to a protocol (e.g., the User Datagram Protocol (UDP)) used by the distributed load balancer system on the fabric 120. The load balancer node layer then forwards the encapsulated packets to the destination server nodes 130 over fabric 120. Each server node 130 includes a local module 132 that is a component of the load balancer system. The module 132 may be referred to herein as a load balancer module or simply LB module, and may be implemented in software, hardware, or a combination thereof on the server node 130. At each server node 130, the respective load balancer module 132 decapsulates the packets and sends the TCP packets to a local TCP stack for normal TCP processing. In at least some embodiments, the load balancer node layer may maintain state information for every client-server TCP flow; however, the load balancer nodes 110 in the load balancer node layer may not interpret anything about the TCP flow. Each flow is managed between the server 134 on the respective server node 130 and the client 160. The distributed load balancer system insures that the TCP packets arrive at the correct destination server 134. The load balancer module 132 at each server node 130 makes the decision as to

a consistent hash function to determine a flow tracker node for the data flow according to a consistent hash ring and recover the mapping from the flow tracker node.

Asymmetric packet flow

[0080] In at least some embodiments, to efficiently utilize ingress node bandwidth and CPU usage when the ratio of outbound traffic to inbound data is greater than 1, the distributed load balancing system forwards outbound packets from the server nodes 130 to multiple egress nodes as shown in Figure 7. In at least some embodiments, for each connection, the load balancer module 132 on the respective server node 130 hashes the client endpoint/public endpoint tuple and uses a consistent hash algorithm to select a load balancer node 110 to serve as the egress server 114 for the respective outbound packet flow. However, in some embodiments other methods and/or data may be used to select the egress servers 114 for connections. The selected egress server 114 may typically be, but is not necessarily, a different load balancer node 110 than the load balancer node 110 that serves as the ingress server 112 for the connection. In at least some embodiments, unless there is a failure of that load balancer node 110 / egress server 114, all of the outbound packets for the particular connection will be forwarded to the same egress server 114 in order to avoid out-of-order packets.

[0081] In at least some embodiments, the method and data used for selecting an egress server 114 by the server nodes 130 may be different than the method and data used for selecting an ingress server 112 performed by the edge router(s) 104. Using the different methods and data may generally result in a different load balancer node 110 being selected as the egress node for a given connections than the load balancer node 110 selected as the ingress node for the connection, and may also result in multiple load balancer nodes 110 being selected as egress nodes to handle the outgoing traffic for connections that pass through a single load balancer node 110 serving as an ingress node.

[0082] Figure 7 graphically illustrates asymmetric packet flow, according to at least some embodiments. At least one connection has been established from clients 160 on external network 150 through ingress server 112 to each of server nodes 130A, 130B, 130C, and 130D. In at least some embodiments, to select egress nodes for the connections, for each connection, the load balancer module 132 on the respective server node 130 hashes the client endpoint/public endpoint tuple and uses a consistent hash algorithm to select a load balancer node 110 to serve as the egress server 114 for the respective outbound packet flow. For example, server node 130A has selected egress server 114A for a connection, and server node 114B has selected egress server 114A for one connection and egress server 114B for another connection. However, in

processed by one or more modules on the load balancer node 110 before being delivered to the BGP client, including processing by the Linux kernel to manage the TCP connection and effectively translate the packets into a TCP stream. This design allows the BGP client to be written using standard Java TCP socket libraries.

5 [0213] Figure 27 illustrates processing of incoming BGP TCP packets by a load balancer (LB) node process 650, according to at least some embodiments. A packet from the edge router 104 arrives at the north-facing NIC 640 and goes into input queue 640 for the receive core 652. The receive core 652 reads the packet from the queue 640, identified the packet as a BGP packet, and places the packet on an input queue 654 for the slow worker core 656. The slow worker core
10 656 validates the packet and places it on the JNI output queue 658. JNI packet receiver 660 reads the packet from the queue 658 via JNI, mangles the source/destination addresses, and writes the packet to a raw socket 644. The Linux kernel 646 receives the raw packet, handles it according to the TCP protocol, and appends the payload data to the TCP socket InputStream. The data from the packet is then delivered to the Java TCP socket in the BGP client 662.

15 [0214] Figure 28 illustrates processing of outgoing BGP TCP packets by a load balancer (LB) node process 650, according to at least some embodiments. The BGP client 662 writes data to a Java TCP socket of Linux kernel 646. The Linux kernel 646 handles the data according to the TCP protocol and converts the data into TCP packet(s). In at least some embodiments, the TCP packet(s) match a 127.x.x.x iptables rule. The TCP packet(s) are placed on an output queue
20 648, for example a Netfilter LOCAL_OUT queue. A Java thread of JNI packet receiver 670 monitoring the queue 648 via JNI receives the TCP packet(s) and marks each NF_STOLEN to make the kernel 646 forget about them. The Java thread mangles the source/destination addresses and adds the packets(s) to a JNI input queue 672 for slow worker core 656 via JNI. The slow worker core 656 receives the TCP packet(s) from its JNI input queue 672 and places the packets
25 on the outbound queue 664 for the north-facing NIC 640 transmit core 666. Transmit core 666 reads the TCP packets(s) from its input queue 664 and writes them to the north-facing NIC 640. The TCP packets are sent by NIC 640 the edge router 104

Distributed load balancer simulation and testing

[0215] The load balancer described herein is a distributed system that requires the interaction
30 of many independent components (e.g., routers, load balancer nodes, load balancer modules, etc.). To perform testing of the distributed components, logic, and protocols, as well as to simulate scenarios such as node failures, message drops, and delays, embodiments of a test system are described that enable the distributed load balancer to be run in a single process where

In at least some embodiments, the successor node in the consistent hash ring serves as the secondary flow tracker for the packet flow.

[0054] Figure 3 shows an example load balancer (LB) node 110 that includes components that implement all three roles (ingress, egress, and flow tracker), according to at least some embodiments. In this example, an ingress server 112 component performs the ingress role of receiving inbound TCP packets from client(s) and sending the TCP packets as encapsulated packets to the server(s). An egress server 114 component performs the egress role of receiving outbound encapsulated packets from the server(s) and sending the decapsulated TCP packets on to the client(s). A flow tracker 116 component performs as a primary or secondary flow tracker for one or more packet flows that are established between the client(s) 160 and the server(s) 134. The ingress server 112 may also communicate with the flow tracker 116 on load balancer node 110, or with the flow tracker 116 on another load balancer node 110 to initiate a TCP connection between a client and one of the servers 134 in response to a connection request received from the respective client 160, or to obtain mapping information for the packet flow.

Load balancer nodes

[0055] Referring again to Figure 1, in at least some embodiments, the load balancer nodes 110 in the load balancer node layer receive client traffic (packets, e.g. TCP packets) from one or more routers 104 on the network and encapsulate the packets according to a protocol (e.g., the User Datagram Protocol (UDP)) used by the distributed load balancer system on the fabric 120. The load balancer node layer then forwards the encapsulated packets to the destination server nodes 130 over fabric 120. Each server node 130 includes a local module 132 that is a component of the load balancer system. The module 132 may be referred to herein as a load balancer module or simply LB module, and may be implemented in software, hardware, or a combination thereof on the server node 130. At each server node 130, the respective load balancer module 132 decapsulates the packets and sends the TCP packets to a local TCP stack for normal TCP processing. In at least some embodiments, the load balancer node layer may maintain state information for every client-server TCP flow; however, the load balancer nodes 110 in the load balancer node layer may not interpret anything about the TCP flow. Each flow is managed between the server 134 on the respective server node 130 and the client 160. The distributed load balancer system insures that the TCP packets arrive at the correct destination server 134. The load balancer module 132 at each server node 130 makes the decision as to

a consistent hash function to determine a flow tracker node for the data flow according to a consistent hash ring and recover the mapping from the flow tracker node.

Asymmetric packet flow

[0080] In at least some embodiments, to efficiently utilize ingress node bandwidth and CPU usage when the ratio of outbound traffic to inbound data is greater than 1, the distributed load balancing system forwards outbound packets from the server nodes 130 to multiple egress nodes as shown in Figure 7. In at least some embodiments, for each connection, the load balancer module 132 on the respective server node 130 hashes the client endpoint/public endpoint tuple and uses a consistent hash algorithm to select a load balancer node 110 to serve as the egress server 114 for the respective outbound packet flow. However, in some embodiments other methods and/or data may be used to select the egress servers 114 for connections. The selected egress server 114 may typically be, but is not necessarily, a different load balancer node 110 than the load balancer node 110 that serves as the ingress server 112 for the connection. In at least some embodiments, unless there is a failure of that load balancer node 110 / egress server 114, all of the outbound packets for the particular connection will be forwarded to the same egress server 114 in order to avoid out-of-order packets.

[0081] In at least some embodiments, the method and data used for selecting an egress server 114 by the server nodes 130 may be different than the method and data used for selecting an ingress server 112 performed by the edge router(s) 104. Using the different methods and data may generally result in a different load balancer node 110 being selected as the egress node for a given connections than the load balancer node 110 selected as the ingress node for the connection, and may also result in multiple load balancer nodes 110 being selected as egress nodes to handle the outgoing traffic for connections that pass through a single load balancer node 110 serving as an ingress node.

[0082] Figure 7 graphically illustrates asymmetric packet flow, according to at least some embodiments. At least one connection has been established from clients 160 on external network 150 through ingress server 112 to each of server nodes 130A, 130B, 130C, and 130D. In at least some embodiments, to select egress nodes for the connections, for each connection, the load balancer module 132 on the respective server node 130 hashes the client endpoint/public endpoint tuple and uses a consistent hash algorithm to select a load balancer node 110 to serve as the egress server 114 for the respective outbound packet flow. For example, server node 130A has selected egress server 114A for a connection, and server node 130B has selected egress server 114A for one connection and egress server 114B for another connection. However, in

processed by one or more modules on the load balancer node 110 before being delivered to the BGP client, including processing by the Linux kernel to manage the TCP connection and effectively translate the packets into a TCP stream. This design allows the BGP client to be written using standard Java TCP socket libraries.

5 **[0213]** Figure 27 illustrates processing of incoming BGP TCP packets by a load balancer (LB) node process 650, according to at least some embodiments. A packet from the edge router 104 arrives at the north-facing NIC 640 and goes into input queue 640 for the receive core 652. The receive core 652 reads the packet from the queue 640, identified the packet as a BGP packet, and places the packet on an input queue 654 for the slow worker core 656. The slow worker core
10 656 validates the packet and places it on the JNI output queue 658. JNI packet receiver 660 reads the packet from the queue 658 via JNI, mangles the source/destination addresses, and writes the packet to a raw socket 644. The Linux kernel 646 receives the raw packet, handles it according to the TCP protocol, and appends the payload data to the TCP socket InputStream. The data from the packet is then delivered to the Java TCP socket in the BGP client 662.

15 **[0214]** Figure 28 illustrates processing of outgoing BGP TCP packets by a load balancer (LB) node process 650, according to at least some embodiments. The BGP client 662 writes data to a Java TCP socket of Linux kernel 646. The Linux kernel 646 handles the data according to the TCP protocol and converts the data into TCP packet(s). In at least some embodiments, the TCP packet(s) match a 127.x.x.x iptables rule. The TCP packet(s) are placed on an output queue
20 648, for example a Netfilter LOCAL_OUT queue. A Java thread of JNI packet receiver 670 monitoring the queue 648 via JNI receives the TCP packet(s) and marks each NF_STOLEN to make the kernel 646 forget about them. The Java thread mangles the source/destination addresses and adds the packets(s) to a JNI input queue 672 for slow worker core 656 via JNI. The slow worker core 656 receives the TCP packet(s) from its JNI input queue 672 and places the packets
25 on the outbound queue 664 for the north-facing NIC 640 transmit core 666. Transmit core 666 reads the TCP packets(s) from its input queue 664 and writes them to the north-facing NIC 640. The TCP packets are sent by NIC 640 to the edge router.

Distributed load balancer simulation and testing

30 **[0215]** The load balancer described herein is a distributed system that requires the interaction of many independent components (e.g., routers, load balancer nodes, load balancer modules, etc.). To perform testing of the distributed components, logic, and protocols, as well as to simulate scenarios such as node failures, message drops, and delays, embodiments of a test system are described that enable the distributed load balancer to be run in a single process where

CLAIMS

1. A distributed load balancer system, comprising:
a plurality of load balancer nodes; and
a plurality of server nodes each comprising a server and a load balancer module;
wherein the plurality of load balancer nodes is configured to distribute data communications protocol packet flows from one or more clients among the plurality of server nodes, wherein, to distribute the packet flows among the plurality of server nodes, the plurality of load balancer nodes is configured to:
select server nodes from among the plurality of server nodes to receive connection requests for the packet flows from the clients; and
send the connection requests to the selected server nodes;
wherein the load balancer module on each server node is configured to:
receive a connection request for a packet flow from one of the plurality of load balancer nodes;
determine if a data communications protocol connection for the respective packet flow is to be accepted by the server on the server node;
if the server cannot accept the data communications protocol connection, reject the connection request; and
if the server can accept the data communications protocol connection, cooperate with the plurality of load balancer nodes to establish the data communications protocol connection for the packet flow between the respective client and the respective server;
wherein, after establishing a data communications protocol connection for a packet flow between a respective client and a respective server, the load balancer module on each server node is further configured to:
select one of the plurality of load balancer nodes to act as an egress server for the data communications protocol connection;
receive incoming data communications protocol packets for the data communications protocol connection from one of the load balancer nodes

and send the incoming data communications protocol packets to the server on the server node; and

intercept outgoing data communications protocol packets for the data communications protocol connection and send the outgoing data communications protocol packets to the selected egress server for delivery to the respective client.

2. The distributed load balancer system as recited in claim 1, further comprising a router configured to distribute the packet flows from the one or more clients among the plurality of load balancer nodes according to a hashed multipath routing technique.

3. The distributed load balancer system as recited in claim 1, wherein the plurality of load balancer nodes is further configured to select the server nodes from among the plurality of server nodes to receive the connection requests according to a random selection technique.

4. The distributed load balancer system as recited in claim 1, wherein the plurality of load balancer nodes is further configured to select another server node from among the plurality of server nodes to receive a rejected connection requests and send the connection request to the other server node.

5. The distributed load balancer system as recited in claim 1, wherein each packet flow is a Transmission Control Protocol (TCP) packet flow, and wherein each data communications protocol connection established between a client and a server is a TCP connection.

6. The distributed load balancer system as recited in claim 1, wherein each data communications protocol connection established between a client and a server is a direct data communications protocol connection that originates at the client, passes through one or more of the plurality of load balancer nodes, and is terminated by the server, wherein the one or more

load balancer nodes do not act as proxies for the data communications protocol connection and do not terminate the data communications protocol connection.

7. A method, comprising:

performing, by one or more of a plurality of load balancer nodes:

receiving a data communications protocol packet in a packet flow for a client; and
sending a connection request for the packet flow to a server node selected from
among a plurality of server nodes;

performing, by the selected server node:

determining if a server on the server node can or cannot accept a data
communications protocol connection for the respective packet flow;

rejecting the connection request upon determining that the server cannot accept
the data communications protocol connection; and

accepting the connection request upon determining that the server can accept the
data communications protocol connection, wherein said accepting the
connection request comprises:

cooperating with the one or more load balancer nodes to establish the data
communications protocol connection between the respective client
and the respective server for the packet flow; and

selecting one of the plurality of load balancer nodes to act as an egress
server for the data communications protocol connection, wherein
outgoing data communications protocol packets on the data
communications protocol connection are sent to the respective
client via the selected egress server.

8. The method as recited in claim 7, wherein said determining, said rejecting, and
said accepting are performed by a load balancer module on the selected server node, the method
further comprising performing, by the load balancer module after accepting the connection
request:

receiving incoming data communications protocol packets for the data communications protocol connection from one of the load balancer nodes and sending the incoming data communications protocol packets to the server on the server node; and

intercepting the outgoing data communications protocol packets for the data communications protocol connection and send the outgoing data communications protocol packets to the selected egress server for delivery to the respective client.

9. The method as recited in claim 8, wherein the packet flow is a Transmission Control Protocol (TCP) packet flow, and wherein the data communications protocol connection established between the client and the server is a TCP connection.

10. The method as recited in claim 8, wherein the established data communications protocol connection is a direct data communications protocol connection that originates at the client, passes through one or more of the plurality of load balancer nodes, and is terminated by the server, wherein the one or more load balancer nodes do not act as proxies for the data communications protocol connection and do not terminate the data communications protocol connection.

11. The method as recited in claim 7, wherein the packet is received from a router that distributes packet flows from one or more clients among the plurality of load balancer nodes according to a hashed multipath routing technique.

12. The method as recited in claim 7, further comprising the one or more load balancer nodes selecting the server node from among the plurality of server nodes according to a random selection technique.

13. The method as recited in claim 7, further comprising, if the selected server node rejects the connection request, the one or more load balancer nodes sending the connection request to another server node selected from among the plurality of server nodes.

14. A non-transitory computer-accessible storage medium storing program instructions computer-executable to implement a load balancer module on each of a plurality of server nodes, wherein each load balancer module is configured to:

receive, from one of a plurality of load balancer nodes, a connection request for a packet flow from a client;

determine if a server on the server node can or cannot accept a data communications protocol connection for the respective packet flow;

reject the connection request upon determining that the server cannot accept the data communications protocol connection; and

upon determining that the server can accept the data communications protocol connection:

communicate with the load balancer node and with the server to establish the data communications protocol connection between the client and the server for the packet flow; and

select one of the plurality of load balancer nodes to act as an egress server for the data communications protocol connection, wherein outgoing data communications protocol packets on the data communications protocol connection are sent to the respective client via the selected egress server.

15. The non-transitory computer-accessible storage medium as recited in claim 14, wherein the program instructions are further computer-executable to implement the load balancer node randomly selecting the server node from among the plurality of server nodes to receive the connection request.

16. The non-transitory computer-accessible storage medium as recited in claim 14, wherein, to reject the connection request, the load balancer module is configured to decrement a time to live (TTL) counter in the connection request and return the connection request to the load

balancer node, wherein the program instructions are further computer-executable to implement the load balancer node:

- checking the TTL counter in the returned connection request;
- if the TTL counter is above a threshold, selecting another server node from among the plurality of server nodes to receive the connection request; and
- if the TTL counter is at or below the threshold, dropping the connection request.

17. The distributed load balancer system as recited in claim 1, wherein, to determine if the data communications protocol connection is to be accepted by the server on the server node, the load balancer module is configured to analyze one or more current resource usage metrics of the server on the server node to determine if the server can accept the data communications protocol connection, wherein the one or more current resource usage metrics include one or more of CPU utilization, bandwidth consumption, server latency, and number of established data communications protocol connections.

18. The method as recited in claim 7, wherein said determining if a server on the server node can or cannot accept the data communications protocol connection comprises analyzing one or more current resource usage metrics of the server to determine if the server can accept the data communications protocol connection.

19. The non-transitory computer-accessible storage medium as recited in claim 14, wherein, to determine if a server on the server node can or cannot accept the data communications protocol connection, the load balancer module is configured to analyze one or more current resource usage metrics of the server to determine if the server can accept the data communications protocol connection.

20. The non-transitory computer-accessible storage medium as recited in claim 14, wherein the one or more current resource usage metrics include one or more of CPU utilization, bandwidth consumption, server latency, and number of established data communications protocol connections.

21. A distributed load balancer system, comprising:
a router configured to receive packets in packet flows from one or more clients according to a single public IP address of the router;
a plurality of server nodes; and
a plurality of load balancer nodes each configured as an ingress server in the distributed load balancer system, wherein the ingress servers all advertise the same single public IP address to the router;
wherein the router is further configured to distribute the packet flows among the plurality of ingress servers according to a hashed multipath routing technique applied to source and destination address information of the packets in the packet flows; and
wherein each ingress server is further configured to receive packets in one or more packet flows from the router and distribute the packets to one or more of the plurality of server nodes that are mapped to the respective packet flows.
22. The distributed load balancer system as recited in claim 21, wherein the hashed multipath routing technique is an equal-cost multipath (ECMP) routing technique.
23. The distributed load balancer system as recited in claim 21, wherein each load balancer node is advertised to the router by one or more of the other load balancer nodes.
24. The distributed load balancer system as recited in claim 23, wherein the one or more other load balancer nodes each establishes a Border Gateway Protocol (BGP) session with the router to advertise the load balancer node to the router.
25. The distributed load balancer system as recited in claim 24, wherein each of the one or more other load balancer nodes advertising the load balancer node to the router is further configured to:
detect that the load balancer node being advertised to the router is down; and

in response to said detecting, close the BGP session that advertises the load balancer node to the router.

26. The distributed load balancer system as recited in claim 25, wherein the router is further configured to redistribute the packet flows among the plurality of ingress servers according to the hashed multipath routing technique in response to the one or more other load balancer nodes closing the BGP sessions.

27. The distributed load balancer system as recited in claim 21, wherein the source and destination address information of a packet includes a client IP address, a client port, a server public IP address, and a server port.

28. A method, comprising:

receiving, by a router, packets in packet flows from one or more clients according to a public IP address;

distributing, by the router, the packet flows among a plurality of load balancer nodes according to a hashed multipath routing technique applied to source and destination address information of the packets in the packet flows, wherein the plurality of load balancer nodes share one or more public IP addresses; and

distributing, by each of the one or more load balancer nodes, the packets in one or more packet flows received from the router to one or more of a plurality of server nodes that are mapped to the respective packet flows.

29. The method as recited in claim 28, wherein the hashed multipath routing technique is an equal-cost multipath (ECMP) routing technique.

30. The method as recited in claim 28, further comprising each load balancer advertising at least one other load balancer node to the router, wherein each load balancer node is advertised to the router by one or more of the other load balancer nodes.

31. The method as recited in claim 30, wherein the one or more other load balancer nodes that advertise a load balancer node include the load balancer node's left and right neighbor load balancer nodes according to a specified ordering of the load balancer nodes.

32. The method as recited in claim 30, wherein said advertising comprises each load balancer node establishing a Border Gateway Protocol (BGP) session with the router for each other load balancer node that the load balancer node advertises to the router.

33. The method as recited in claim 32, further comprising:
detecting, by a load balancer node, that another load balancer node being advertised to the router by the load balancer node is down; and
in response to said detecting, closing the BGP session with the router that advertises the other load balancer node.

34. The method as recited in claim 33, further comprising redistributing, by the router, the packet flows among the plurality of load balancer nodes according to the hashed multipath routing technique in response to determining that one or more BGP sessions advertising a load balancer node have been closed.

35. The method as recited in claim 28, wherein the source and destination address information of a packet includes a client IP address, a client port, a server public IP address, and a server port.

36. A non-transitory computer-accessible storage medium storing program instructions computer-executable to implement:
advertising, by a plurality of load balancer nodes, one or more public IP addresses shared by the load balancer nodes to a router;
receiving, by the router, packets in packet flows from one or more clients according to a single public IP address;

distributing, by the router, the packet flows among the plurality of load balancer nodes according to a hashed multipath routing technique applied to source and destination address information of the packets in the packet flows; and distributing, by each of the one or more load balancer nodes, the packets in one or more packet flows received from the router to one or more of a plurality of server nodes.

37. The non-transitory computer-accessible storage medium as recited in claim 36, wherein, in said advertising, each load balancer node is advertised to the router by one or more of the other load balancer nodes.

38. The non-transitory computer-accessible storage medium as recited in claim 37, wherein each of the one or more other load balancer nodes establishes a Border Gateway Protocol (BGP) session with the router to advertise the load balancer node to the router.

39. The non-transitory computer-accessible storage medium as recited in claim 37, wherein the program instructions are further computer-executable to implement detecting, by a load balancer node, that another load balancer node being advertised to the router by the load balancer node is down; and in response to said detecting, closing a Border Gateway Protocol (BGP) session with the router that advertises the other load balancer node to the router.

40. The non-transitory computer-accessible storage medium as recited in claim 37, wherein the program instructions are further computer-executable to implement redistributing, by the router, the packet flows among the plurality of load balancer nodes according to the hashed multipath routing technique in response to determining that one of the plurality of load balancer nodes is not advertised by the other load balancer nodes.

41. A distributed load balancer system, comprising:

a plurality of load balancer nodes, wherein at least two of the plurality of load balancer nodes are configured as ingress servers, and wherein at least two of the plurality of load balancer nodes are configured as flow tracker nodes;
a plurality of server nodes; and
a router configured to distribute packet flows from one or more clients to the ingress servers according to a hashed multipath routing technique;
wherein each ingress server is configured to:
 receive a packet in a packet flow for a client from the router;
 determine that the ingress server does not have a mapping for the packet flow to the plurality of server nodes;
 determine at least one flow tracker node for the packet flow according to a consistent hash function applied to source and destination address information of the packet;
 obtain a mapping of a connection to a particular one of the plurality of server nodes for the packet flow from the at least one flow tracker node; and
 send one or more packets of the packet flow to the particular server node.

42. The distributed load balancer system as recited in claim 41, wherein the packet flow is a Transmission Control Protocol (TCP) packet flow.

43. The distributed load balancer system as recited in claim 41,
wherein at least two of the plurality of load balancer nodes are configured as egress servers configured to send outgoing packets from the server nodes to the one or more clients,
wherein the server node is configured to:
 select one of the egress servers for the packet flow; and
 send one or more outgoing packets for the packet flow to the selected egress server;
wherein the egress server is configured to send the outgoing packets to the client; and

wherein the selected egress server for the packet flow is a different load balancer node than the ingress server for the packet flow.

44. The distributed load balancer system as recited in claim 43, wherein the ingress server encapsulates the one or more packets according to User Datagram Protocol (UDP) prior to sending the packets to the server node, wherein the server node encapsulates the outgoing packets according to UDP prior to sending the outgoing packets to the egress server, and wherein the egress server strips the UDP encapsulation from the outgoing packets prior to sending the outgoing packets to the client.

45. The distributed load balancer system as recited in claim 44, wherein the server node includes a load balancer module configured to:

- select the egress server for the packet flow;
- receive the incoming encapsulated packets from the ingress server;
- strip the UDP encapsulation from the packets and deliver the packets to a server on the server node;
- obtain the outgoing packets from the server on the server node;
- encapsulate the outgoing packets according to UDP; and
- send the encapsulated outgoing packets to the egress server.

46. The distributed load balancer system as recited in claim 41, wherein, to obtain a mapping of a connection to a particular one of the plurality of server nodes for the packet flow from the at least one flow tracker node,

- the ingress server sends a message that includes information for the packet flow to a primary flow tracker for the packet flow;
- the primary flow tracker sends a message to a secondary flow tracker for the packet flow that includes the information for the packet flow, wherein the primary and secondary flow trackers for the packet flow are different load balancer nodes;
- the secondary flow tracker sends an acknowledgement for the packet flow to the client;

the ingress server receives an acknowledgement packet from the client and forwards the acknowledgement packet to the primary flow tracker;

the primary flow tracker randomly selects the particular server node from among the plurality of server nodes as the server node to receive the packet flow and sends a message to the secondary flow tracker indicating the particular server node;

the secondary flow tracker fabricates a synchronization message and sends the fabricated synchronization message to the particular server node;

the secondary flow tracker receives connection information for the packet flow from the particular server node and sends a message including the connection information to the primary flow tracker; and

the primary flow tracker sends a message including the connection information for the packet flow to the ingress server, wherein the connection information maps the packet flow to the particular server node.

47. The distributed load balancer system as recited in claim 46, wherein the server node includes a load balancer module configured to:

receive the fabricated synchronization message from the secondary flow tracker;

determine that a server on the server node can accept a connection;

generate a synchronization packet according to the fabricated synchronization message and deliver the synchronization packet to the server on the server node;

intercept an acknowledgement packet generated by the server on the server node; and

send a message including the connection information to the secondary flow tracker.

48. A method, comprising:

performing, by an ingress server on one of a plurality of load balancer nodes:

receiving a packet in a packet flow for a client, wherein the packet is received from a router that distributes packet flows from one or more clients to the plurality of load balancer nodes according to a consistent hash function;

determining a load balancer node that serves as a flow tracker node for the packet flow according to a consistent hash function applied to source and destination address information of the packet;
obtaining a mapping of a connection to one of a plurality of server nodes for the packet flow from the flow tracker node for the packet flow; and
sending one or more packets of the packet flow to the server node indicated by the mapping.

49. The method as recited in claim 48, wherein the packet flow is a Transmission Control Protocol (TCP) packet flow.

50. The method as recited in claim 48, further comprising the ingress server encapsulating the packets according to User Datagram Protocol (UDP) prior to said sending the one or more packets of the packet flow to the server node.

51. The method as recited in claim 48, further comprising:
selecting, by the server node, one of the plurality of load balancer nodes as an egress server for the packet flow, wherein the selected egress server for the packet flow is on a different load balancer node than the ingress server for the packet flow;
sending, by the server node, one or more outgoing packets for the packet flow to the selected egress server; and
sending, by the egress server, the outgoing packets to the client of the packet flow.

52. The method as recited in claim 51, further comprising:
the server node encapsulating the outgoing packets according to User Datagram Protocol (UDP) prior to said sending the outgoing packets to the egress server; and
the egress server stripping the UDP encapsulation from the outgoing packets prior to said sending the outgoing packets to the client.

53. The method as recited in claim 48, wherein the flow tracker node is a primary flow tracker node for the packet flow, and wherein a next load balancer node in a consistent hash ring according to the consistent hash function is a secondary flow tracker node for the packet flow.

54. The method as recited in claim 53, wherein said obtaining a mapping of a connection to one of a plurality of server nodes for the packet flow from the flow tracker node for the packet flow comprises:

- sending, by the ingress server, at least one message to the primary flow tracker node for the packet flow, each message including a packet of the packet flow received from the router;
- selecting, by the primary flow tracker node, the server node for the packet flow from among the plurality of server nodes;
- sending, by the primary flow tracker node, packet flow information including an indication of the selected server node to the secondary flow tracker node;
- facilitating, by the secondary flow tracker node, establishment of the connection to the selected server node for the packet flow by communicating with the server node and the client; and
- sending, by the secondary flow tracker node, connection information for the packet flow to the ingress server via the primary flow tracker node, wherein the connection information maps the packet flow to the selected server node.

55. The method as recited in claim 54, wherein the server node includes a load balancer module, wherein said facilitating establishment of the connection to the selected server node for the packet flow by communicating with the server node and the client comprises:

- sending, by the secondary flow tracker node, a fabricated synchronization message to the load balancer module on the server node; and
- performing, by the load balancer module on the server node:
 - determining that a server on the server node can accept a connection;

generating a synchronization packet according to the fabricated synchronization message;
delivering the synchronization packet to the server on the server node;
intercepting an acknowledgement packet generated by the server on the server node; and
sending a message including the connection information to the secondary flow tracker node.

56. A non-transitory computer-accessible storage medium storing program instructions computer-executable to implement an ingress server and a flow tracker on each of a plurality of load balancer nodes, each ingress server configured to:

receive a packet in a packet flow for a client, wherein the packet is received from a router that distributes packet flows from one or more clients to the plurality of load balancer nodes according to a consistent hash function;
determine one of the plurality of load balancer nodes that serves as a flow tracker node for the packet flow according to a consistent hash function applied to source and destination address information of the packet;
obtain a mapping of a connection to one of a plurality of server nodes for the packet flow from the flow tracker node for the packet flow; and
send one or more packets of the packet flow to the server node indicated by the mapping.

57. The non-transitory computer-accessible storage medium as recited in claim 56, wherein the program instructions are further computer-executable to implement an egress server on each of the load balancer nodes and a load balancer module on each of the plurality of server nodes, each load balancer module configured to:

select one of the plurality of load balancer nodes as an egress server for a packet flow, wherein the selected egress server for the packet flow is on a different load balancer node than an ingress server for the packet flow; and
send one or more outgoing packets for the packet flow to the selected egress server;

wherein each egress server is configured to send outgoing packets in a packet flow received from a load balancer module to a client of the packet flow.

58. The non-transitory computer-accessible storage medium as recited in claim 57, wherein each ingress server is further configured to encapsulate the packets according to User Datagram Protocol (UDP) prior to said sending the one or more packets of the packet flow to the server node;

wherein each load balancer module is further configured to:

strip the UDP encapsulation from the packets received from an ingress server and

deliver the packets to a server on the respective server node;

intercept the outgoing packets from the server on the respective server node; and

encapsulate the outgoing packets according to UDP prior to said sending the outgoing packets to an egress server;

wherein each egress server is further configured to strip the UDP encapsulation from the outgoing packets prior to said sending the outgoing packets to the client of the packet flow.

59. The non-transitory computer-accessible storage medium as recited in claim 56, wherein the flow tracker node is a primary flow tracker node for the packet flow, and wherein a next load balancer node in a consistent hash ring according to the consistent hash function is a secondary flow tracker node for the packet flow, and wherein, to obtain a mapping of a connection to one of a plurality of server nodes for the packet flow from the flow tracker node for the packet flow:

the ingress server is configured to send at least one message to the primary flow tracker node for the packet flow, each message including a packet of the packet flow received from the router;

the primary flow tracker node is configured to:

select the server node for the packet flow from among the plurality of server nodes; and

send packet flow information including an indication of the selected server node
to the secondary flow tracker node;
the secondary flow tracker node is configured to:
facilitate establishment of the connection to the selected server node for the packet
flow by communicating with the server node and the client; and
send connection information for the packet flow to the ingress server via the
primary flow tracker node, wherein the connection information maps the
packet flow to the selected server node.

60. The non-transitory computer-accessible storage medium as recited in claim 59,
wherein the program instructions are further computer-executable to implement a load balancer
module on each of the plurality of server nodes, and wherein, to facilitate establishment of the
connection to the selected server node for the packet flow by communicating with the server
node and the client:

the secondary flow tracker node is configured to send a fabricated synchronization
message to the load balancer module on the server node; and
the load balancer module on the server node is configured to:
determine that a server on the server node can accept a connection;
generate a synchronization packet according to the fabricated synchronization
message;
deliver the synchronization packet to the server on the server node;
intercept an acknowledgement packet generated by the server on the server node;
and
send a message including the connection information to the secondary flow
tracker node.

CLAIMS

1. A distributed load balancer system, comprising:
a plurality of load balancer nodes; and
a plurality of server nodes each comprising a server and a load balancer module;
wherein the plurality of load balancer nodes is configured to distribute data communications protocol packet flows from one or more clients among the plurality of server nodes, wherein, to distribute the packet flows among the plurality of server nodes, the plurality of load balancer nodes is configured to:
select server nodes from among the plurality of server nodes to receive connection requests for the packet flows from the clients; and
send the connection requests to the selected server nodes;
wherein the load balancer module on each server node is configured to:
receive a connection request for a packet flow from one of the plurality of load balancer nodes;
determine if a data communications protocol connection for the respective packet flow is to be accepted by the server on the server node;
if the server cannot accept the data communications protocol connection, reject the connection request; and
if the server can accept the data communications protocol connection, cooperate with the plurality of load balancer nodes to establish the data communications protocol connection for the packet flow between the respective client and the respective server;
wherein, after establishing a data communications protocol connection for a packet flow between a respective client and a respective server, the load balancer module on each server node is further configured to:
select one of the plurality of load balancer nodes to act as an egress server for the data communications protocol connection;
receive incoming data communications protocol packets for the data communications protocol connection from one of the load balancer nodes

and send the incoming data communications protocol packets to the server on the server node; and

intercept outgoing data communications protocol packets for the data communications protocol connection from the server on the server node and send the outgoing data communications protocol packets to the selected egress server for delivery to the respective client.

2. The distributed load balancer system as recited in claim 1, further comprising a router configured to distribute the packet flows from the one or more clients among the plurality of load balancer nodes according to a hashed multipath routing technique.

3. The distributed load balancer system as recited in claim 1, wherein the plurality of load balancer nodes is further configured to select the server nodes from among the plurality of server nodes to receive the connection requests according to a random selection technique.

4. The distributed load balancer system as recited in claim 1, wherein the plurality of load balancer nodes is further configured to select another server node from among the plurality of server nodes to receive a rejected connection requests and send the connection request to the other server node.

5. The distributed load balancer system as recited in claim 1, wherein each packet flow is a Transmission Control Protocol (TCP) packet flow, and wherein each data communications protocol connection established between a client and a server is a TCP connection.

6. The distributed load balancer system as recited in claim 1, wherein each data communication protocol connection established between a client and a server is a direct data communications protocol connection that originates at the client, passes through one or more of the plurality of load balancer nodes, and is terminated by the server, wherein the one or more

load balancer nodes do not act as proxies for the data communications protocol connection and do not terminate the data communications protocol connection.

7. A method, comprising:

performing, by one or more of a plurality of load balancer nodes:

receiving a data communication protocol packet in a packet flow for a client; and

sending a connection request for the packet flow to a server node selected from among a plurality of server nodes;

performing, by a load balancer module on the selected server node:

determining if a server on the server node can or cannot accept a data communication protocol connection for the respective packet flow;

rejecting the connection request upon determining that the server cannot accept the data communication protocol connection; and

accepting the connection request upon determining that the server can accept the data communication protocol connection, wherein said accepting the connection request comprises:

cooperating with the one or more load balancer nodes to establish the data communications protocol connection between the respective client and the respective server for the packet flow; and

selecting one of the plurality of load balancer nodes to act as an egress server for the data communications protocol connection; and

performing, by the load balancer module after accepting the connection request:

receiving incoming data communications protocol packets for the data communications protocol connection from one of the load balancer nodes and sending the incoming data communications protocol packets to the server on the server node; and

intercepting the outgoing data communications protocol packets for the data communications protocol connection from the server on the server node and sending the outgoing data communications

protocol packets to the selected egress server for delivery to the respective client.

8. The method as recited in claim 7, wherein the packet flow is a Transmission Control Protocol (TCP) packet flow, and wherein the data communications protocol connection established between the client and the server is a TCP connection.

9. The method as recited in claim 7, wherein the established data communications protocol connection is a direct data communications protocol connection that originates at the client, passes through one or more of the plurality of load balancer nodes, and is terminated by the server, wherein the one or more load balancer nodes do not act as proxies for the data communications protocol connection and do not terminate the data communications protocol connection.

10. The method as recited in claim 7, wherein the packet is received from a router that distributes packet flows from one or more clients among the plurality of load balancer nodes according to a hashed multipath routing technique.

11. The method as recited in claim 7, further comprising the one or more load balancer nodes selecting the server node from among the plurality of server nodes according to a random selection technique.

12. The method as recited in claim 7, further comprising, if the selected server node rejects the connection request, the one or more load balancer nodes sending the connection request to another server node selected from among the plurality of server nodes.

13. A non-transitory computer-accessible storage medium storing program instructions computer-executable to implement a load balancer module on each of a plurality of server nodes, wherein each load balancer module is configured to:

receive, from one of a plurality of load balancer nodes, a connection request for a packet flow from a client;

determine if a server on the server node can or cannot accept a data communications protocol connection for the respective packet flow;

reject the connection request upon determining that the server cannot accept the data communications protocol connection; and

upon determining that the server can accept the data communications protocol connection:

- communicate with the load balancer node and with the server to establish the data communications protocol connection between the client and the server for the packet flow;
- select one of the plurality of load balancer nodes to act as an egress server for the data communications protocol connection;
- receive incoming data communications protocol packets for the data communications protocol connection from one of the load balancer nodes and send the incoming data communications protocol packets to the server on the server node; and
- intercept outgoing data communications protocol packets for the data communications protocol connection from the server on the server node and send the outgoing data communications protocol packets to the selected egress server for delivery to the respective client.

14. The non-transitory computer-accessible storage medium as recited in claim 13, wherein the program instructions are further computer-executable to implement the load balancer node randomly selecting the server node from among the plurality of server nodes to receive the connection request.

15. The non-transitory computer-accessible storage medium as recited in claim 13, wherein, to reject the connection request, the load balancer module is configured to decrement a time to live (TTL) counter in the connection request and return the connection request to the load

balancer node, wherein the program instructions are further computer-executable to implement the load balancer node:

- checking the TTL counter in the returned connection request;
- if the TTL counter is above a threshold, selecting another server node from among the plurality of server nodes to receive the connection request; and
- if the TTL counter is at or below the threshold, dropping the connection request.

16. The distributed load balancer system as recited in claim 1, wherein, to determine if the data communications protocol connection is to be accepted by the server on the server node, the load balancer module is configured to analyze one or more current resource usage metrics of the server on the server node to determine if the server can accept the data communications protocol connection, wherein the one or more current resource usage metrics include one or more of CPU utilization, bandwidth consumption, server latency, and number of established data communications protocol connections.

17. The method as recited in claim 7, wherein said determining if a server on the server node can or cannot accept the data communications protocol connection comprises analyzing one or more current resource usage metrics of the server to determine if the server can accept the data communications protocol connection.

18. The non-transitory computer-accessible storage medium as recited in claim 13, wherein, to determine if a server on the server node can or cannot accept the data communications protocol connection, the load balancer module is configured to analyze one or more current resource usage metrics of the server to determine if the server can accept the data communications protocol connection.

19. The non-transitory computer-accessible storage medium as recited in claim 13, wherein the one or more current resource usage metrics include one or more of CPU utilization, bandwidth consumption, server latency, and number of established data communications protocol connections.