WHAT IS CLAIMED IS:

1. A port switch service (PSS) system, comprising a server cluster and a client cluster, wherein the server cluster elects a master node in a current cluster through a quorum algorithm and is guaranteed to be unique within a specified period in a lease form; the server cluster employs a mode of one master node plus a plurality of salve nodes, or a mode of one master node plus a plurality of slave nodes plus a plurality of arbiter nodes, and all data are stored in a memory (RAM) of the master node only (full-in-memory), thereby ensuring all data are stored in a single node; the client cluster contains various client nodes needing to use the PSS system, and each client node can establish connection with the master node as needed; after connected to the master node, the client node can register any number of ports thereon; after the client node goes offline, the corresponding registered ports to the client node are released; after the master node is offline, the corresponding registered ports thereto become invalid; the un-registered (invalid) ports can be re-registered by the client nodes.
2. The port switch service (PSS) system according to claim 1, wherein each of the client nodes maintains at least one Keep-Alive connection with the port switch service system, and any number of ports can be registered for each Keep-Alive connection; each of the client nodes needs to maintain a heartbeat signal with the port switch service system; each of the client nodes is identified in the server cluster through a unique node ID.
3. The port switch service (PSS) system according to claim 1, wherein the name of the port is described using a UTF-8 character string and must be globally unique; registering a port will fail if the port is already registered by another client node; and the port contains a message caching queue and a port release notification list.

4. The port switch service (PSS) system according to claim 1, wherein the PSS system offers the following application programming interface (API) primitives: Waiting for Message (WaitMsg), Relet, Port Registration (RegPort), Port Un-registration (UnRegPort), Message Sending (SendMsg), Port Query (QueryPort), Node Query (QueryNode) and Clear; the message registration primitive of the Port Registration permits that one communication request contains multiple port registration commands simultaneously; the message un-registration primitive of

the Port Un-registration permits that one communication request contains multiple port un-registration commands simultaneously; and the message sending primitive permits that one communication request contains multiple messages simultaneously (batch message sending).

5. The port switch service (PSS) system according to claim 4, wherein the connection of the client cluster and the port switch service system includes message receiving connections and message sending connections; the message receiving connection (1:1) uses the WaitMsg method for the node registration and message pushing, keeps occupying all ports belong to current node using Relet, and uses the Clear primitive to clean up before normal disconnection; each node within the cluster should keep and only keep a single message receiving connection, which is a Keep-Alive connection; the connection active is always kept and Relet is completed in a timely manner, because re-establishing a receiving connection will require service electing again (port registration); with respect to the message sending connection (1:N): all connections that are not upgraded using WaitMsg API are deemed as sending connections, uses primitives like RegPort, UnRegPort, SendMsg and QueryPort for non-pushing requests, without the need for using Relet to keep heartbeat, and does not need to use the Clear command to clean up; and each node within the cluster maintains a message sending connection pool, so that worker threads can stay in communication with the port switch service system.

6. The port switch service (PSS) system according to claim 1, wherein the server cluster can be segmented into sub server clusters by name spaces, and the sub server clusters achieve horizontal scaling through a tree cascade structure; and each of

the client nodes is registered on ports under a local name space and a superior name space of the corresponding client node.