

adapters may, in turn, comprise processing elements and/or logic circuitry configured to manipulate these data structures. According to an illustrative embodiment, mbufs stored in the buffer cache 255 may contain one or more data streams that may be requested by the plurality of clients 120.

[0038] The network adapter 220 sends and receives data to/from other nodes in the network 100, e.g., over an Ethernet link, a synchronous optical network (SONET) link, a wireless connection, etc. Specifically, the network adapter comprises the mechanical, electrical and signaling circuitry needed to connect the streaming media cache 200 to a client 120 over the computer network 100. The adapter may also include specialized processing elements, such as logic or processors that format in-coming and out-going packets consistent with a predetermined network communication protocol. The network adapter may be implemented by components and circuitry on a network interface card (NIC), as known in the art.

[0039] The storage adapter 260 cooperates with the operating system 300 to access client-requested data streams stored on the disks 270. The storage disks (or other storage devices) are attached, via the storage adapter 260, to the streaming media cache 200. The storage adapter includes input/output (I/O) interface circuitry that couples to the disks over an I/O interconnect arrangement, such as a conventional high-performance, Fibre Channel serial link topology. The client-requested data streams are retrieved by the storage adapter and processed by the processor 230 (or the adapter 260 itself) in accordance with the storage operating system 300. The data streams are then forwarded over the system bus 210 to the network adapter 220, where they are formatted into packets and sent to their requesting clients 120.

[0040] FIG. 3 is a schematic block diagram illustrating the exemplary storage operating system 300, which may represent the streaming media cache 200. The operating system communicates with the network 100 through a series of software layers, which are organized as a network protocol engine 310. The layers of the network protocol engine 310 process network headers appended to data packets transmitted and received to/from the network. For example, the engine 310 may comprise a data link layer, an IP layer, a TCP/UDP layer, and so forth.

[0041] According to the illustrative embodiment, the network protocol engine 310 associates a unique "port" number with each streaming media protocol, such as the RTSP or the MMS protocol, that may be processed by the streaming media cache 200. The engine 310 identifies a received data packet as being formatted according to a particular streaming media protocol when a port number stored in a designated header of the received packet equals the port number associated with the protocol. For example, if the RTSP and MMS protocols are respectively associated with TCP port numbers 554 and 1755, then the 310 identifies data packets addressed to TCP port number 554 as RTSP data packets, whereas packets addressed to TCP port number 1755 are identified as MMS data packets. Those skilled in the art will appreciate that the port number identifying the streaming media protocol need not be associated with a TCP port and may be, e.g., a UDP port number instead. Furthermore, as mentioned above, the RTSP and MMS requests may be identified by a respective "mms" or "rtsp" portion of the request URL.

[0042] When the network protocol engine 310 identifies that a received data packet is addressed to a port number associated with a streaming media protocol, the packet is passed from the engine 310 to the protocol's corresponding streaming media protocol engine 330. For example, a packet addressed to TCP port 554 may be passed from the network protocol engine 310 to an RTSP protocol engine. Each streaming media protocol engine 330 is configured to process data packets formatted in accordance with its corresponding streaming media protocol. For instance, the RTSP protocol engine processes data packets containing RTSP requests for a data stream (e.g., requests to PLAY, PAUSE, or RECORD the stream).

[0043] The streaming media protocol engines 330 are interfaced with a packet pacing sub-system (SMPACER) 320 and a streaming media disk sub-system (SMDISK) 340. The SMDISK sub-system, which may also be referred to as a streaming media persistent storage sub-system, receives instructions from the streaming media protocol engines to write and retrieve data packets to/from the storage devices 270. To that end, SMDISK sub-system 340 issues functional calls to a file system layer 350, which writes or retrieves data to/from the storage devices through a storage access engine 360. The storage access engine 360 comprises a series of software layers that facilitate data transfers between the file system and the storage devices. For instance, these layers may include, e.g., a disk storage layer to manage a redundant array of independent disks (RAID), a disk driver layer to manage communications over a small computer system interface (SCSI), and so forth.

[0044] The SMDISK sub-system is preferably configured to process data packets that are stored in one or more memory buffers (mbufs), e.g., located in the buffer cache 255. Accordingly, the SMDISK sub-system 340 may pass mbuf pointers referencing the data packets to other layers in the storage operating system 300. For example, the SMDISK sub-system may forward mbuf pointers referencing data packets retrieved from the storage devices 270 to the SMPACER sub-system. Similarly, the SMDISK sub-system may receive mbuf pointers corresponding to data packets received by the network protocol engine 310.

[0045] The SMPACER sub-system 320 is responsible for determining the rate at which data packets are sent from the streaming media cache 200 to their requesting clients 120. In one embodiment, the SMPACER sub-system 320 waits to receive a predetermined number of mbuf pointers referencing packets of a client-requested data stream. Once the predetermined number of mbuf pointers has been received (or, optionally, once a predetermined period of time has elapsed), the SMPACER sub-system 320 makes a "call-back" (i.e., a function call) to an appropriate streaming media protocol engine 330, which returns a "delivery time" that defines when a copy of the data packet should be transmitted to its requesting client. The choice of which streaming media protocol engine 330 is called by the SMPACER sub-system depends on which streaming media protocol, e.g., RTSP or MMS protocol, is used by the requesting client. For instance, SMPACER 320 makes a call-back to an MMS protocol engine when the requesting client is configured to communicate via the MMS protocol.

[0046] FIG. 4 illustrates a method 400 for unified caching of media content, according to one embodiment of the