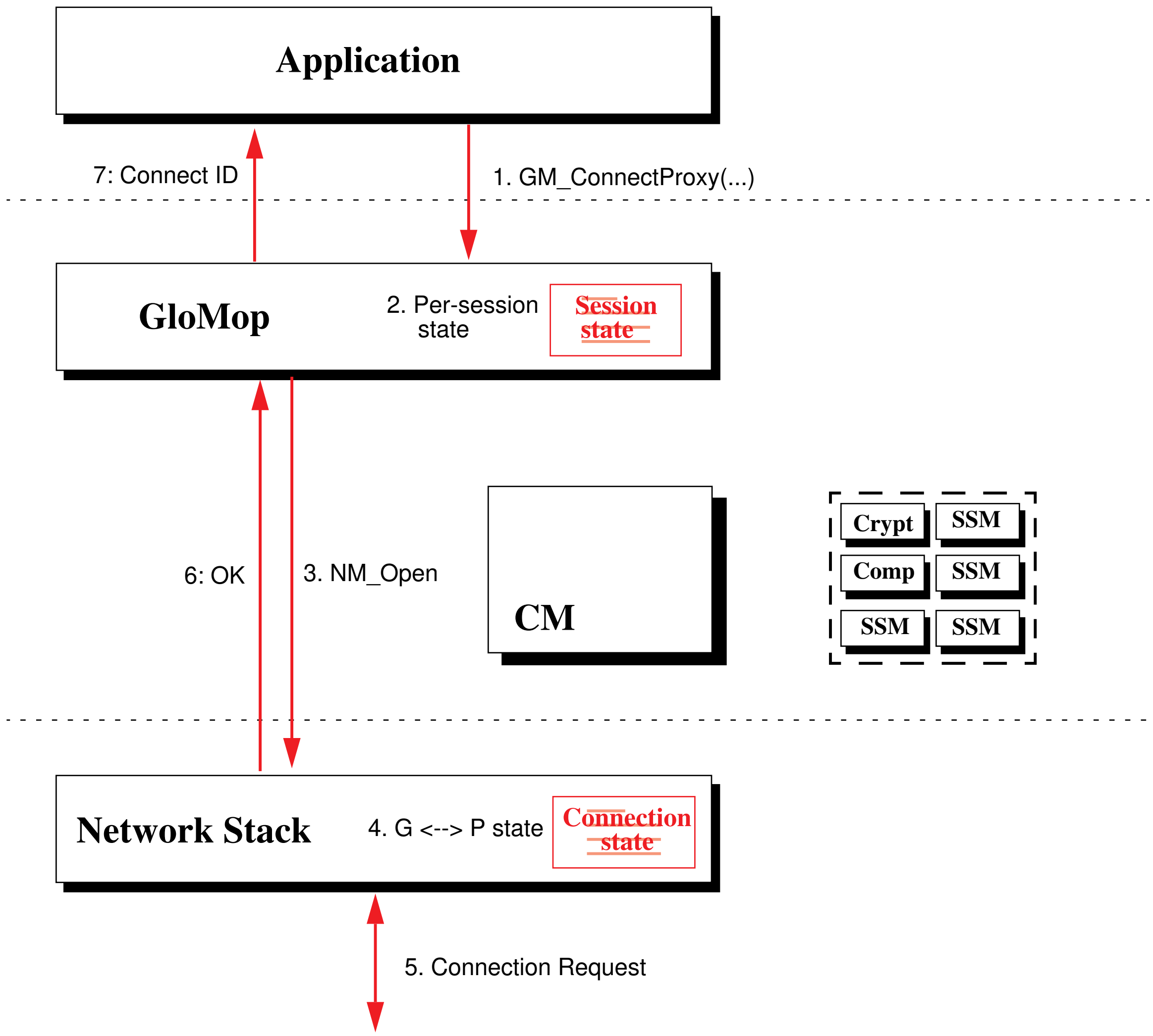
Example Scenarios of GloMop/Proxy Interaction

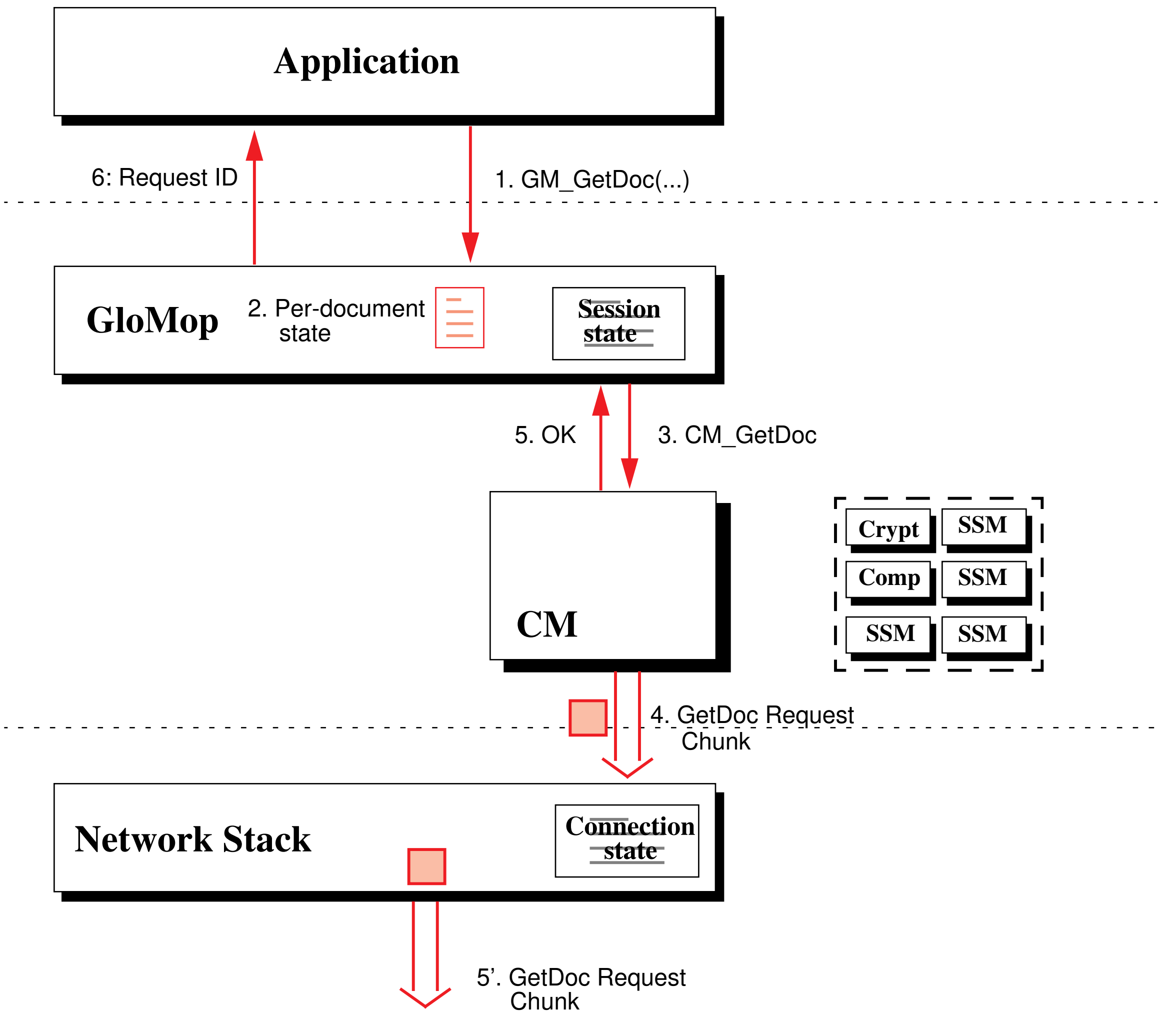
0.1 Connecting to a Proxy



1. GM_ConnectProxy: The client application issues a "GM_ConnectProxy" request to the GloMop layer. Parameters to this request include the proxy address. This request is a blocking request.
2. Session state: The GloMop layer begins to store state information associated with this new session
3. NM_Open: The GloMop layer issues an NM_Open signal to the network stack, asking it to forge a new transport-level connection with the proxy.
4. G <--> P state: The network stack initializes state information associated with this particular proxy connection.
5. The network stack issues the transport-level connection request to the proxy.
6. After the connection is forged, a success message is propagated back to GloMop.

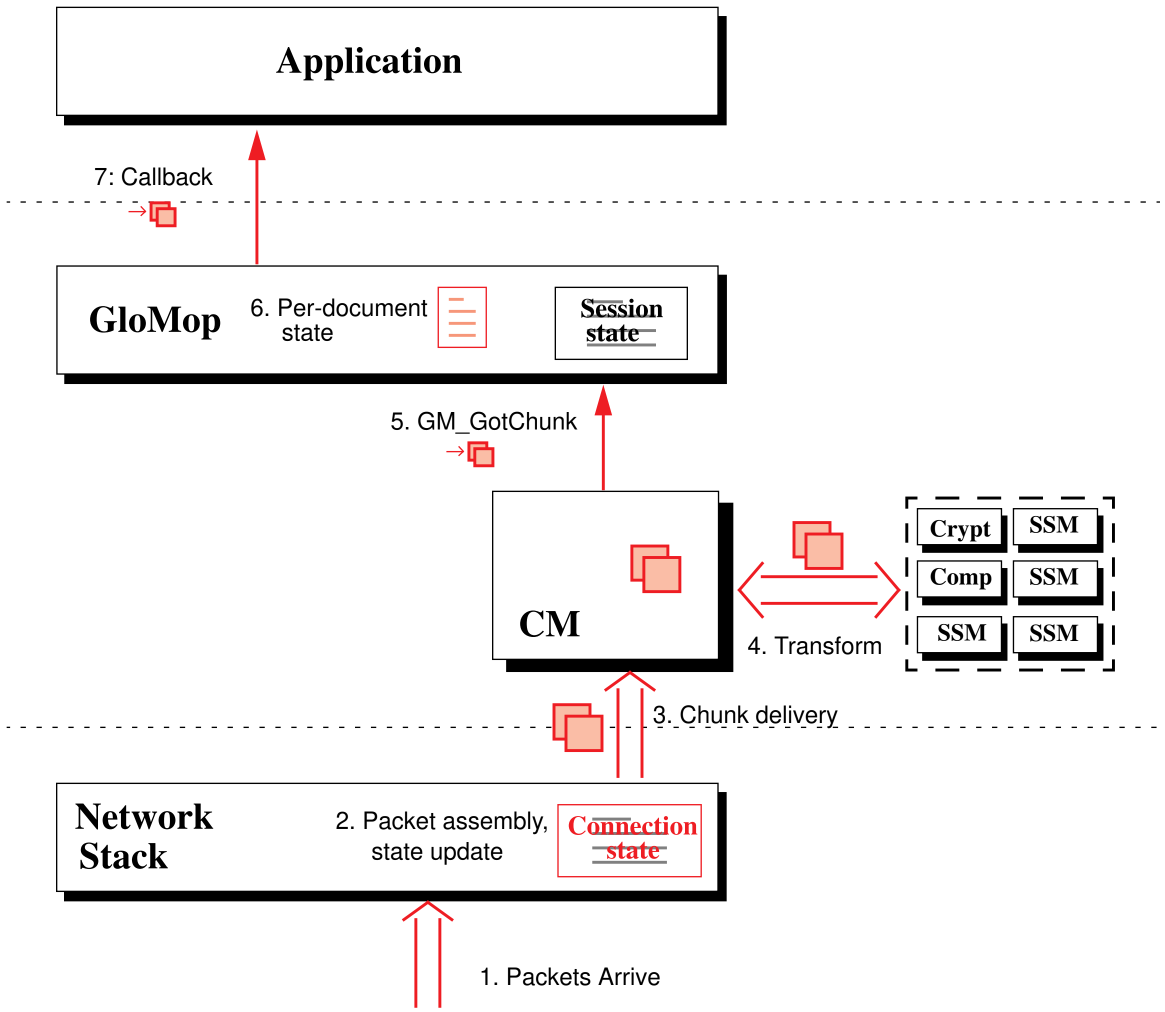
7. The success message and a new connection ID are returned to the client application.

0.2 A Document Request



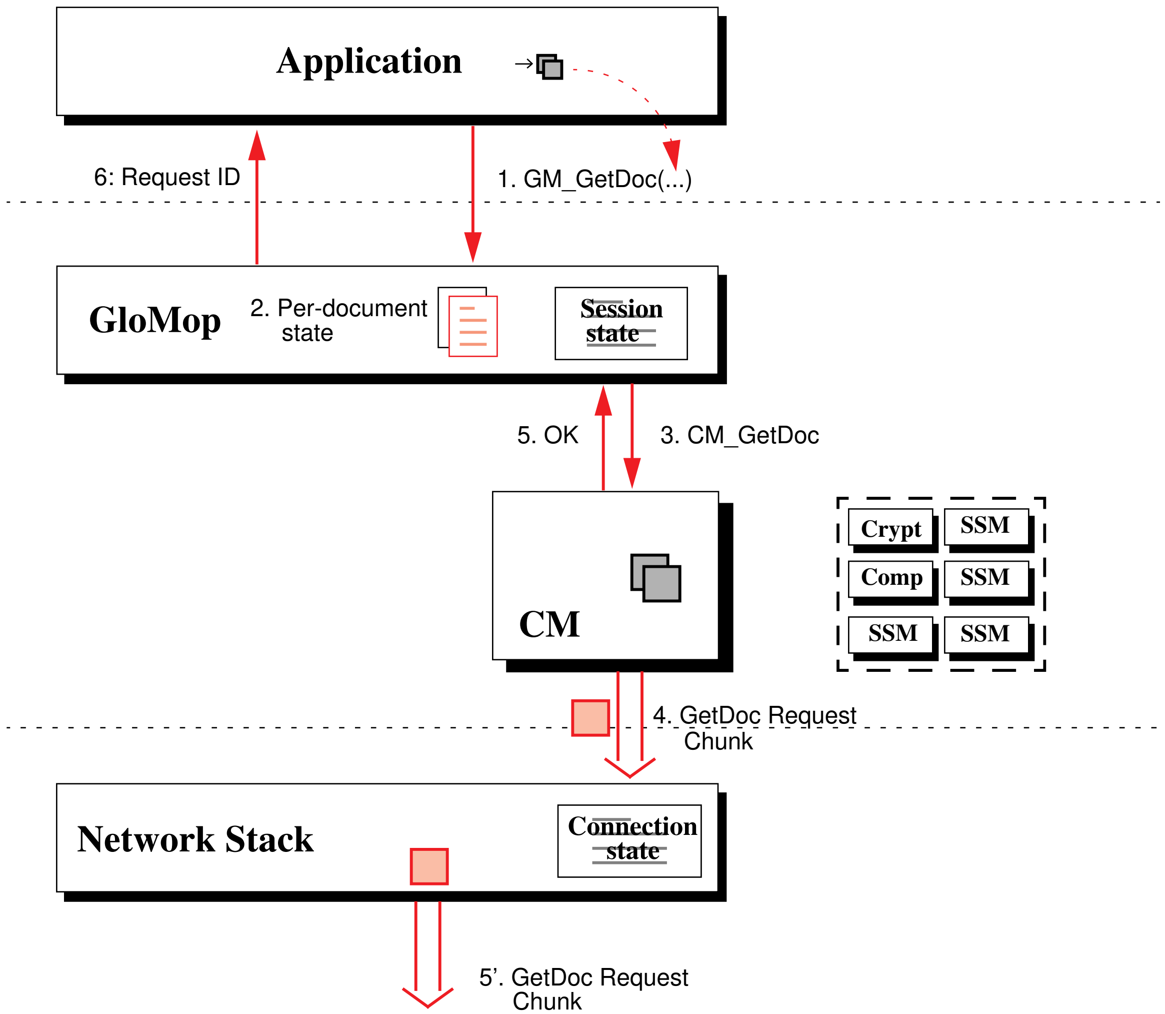
1. GM_GetDoc: The client application issues a "GM_GetDoc" request to GloMop. Arguments include the document locator, QOS and encryption preferences, a callback routine, and the callback threshold and preload count. This is an asynchronous (non-blocking) call.
2. Per-document state is initialized in the GloMop layer. This state includes QOS preferences, the callback threshold and preload count, callback routines for outstanding chunks, etc.
3. A CM_GetDoc request is forwarded to the chunk manager, giving it the opportunity to initialize relevant data structures for the expected incoming chunks.
4. The GetDoc request parameters are packaged into a chunk and given to the network stack.
5. The application thread returns from the chunk manager once this chunk has been given to the network stack. Concurrently, network stack queues, schedules, and delivers the outgoing chunk to the proxy.
6. The application thread returns from GloMop; a requestID for the document is passed back to the application.

0.3 Chunk arrival



1. The network stack receives packets.
2. If fragmentation of chunks has occurred at the source, the packets are reassembled into a chunk, otherwise the chunk is simply extracted from a single packet. The network stack's state information is modified to reflect the arrival of the new chunk.
3. The extracted chunk is forward to the chunk manager.
4. The generic transformation libraries and SSMs are used to transform the chunk into a native format or object.
5. The chunk manager passes the GloMop layer a handle (or pointer, depending on the particular OS) to the chunk in a CM_GotChunk message.
6. The per-document state information associated with the new chunk's document is updated.
7. If the threshold specified in the original OpenDoc call has been exceeded, then the GloMop layer notifies the client application of the new chunk via a callback. The handle (or pointer) to the new chunk is delivered in this callback.

0.4 Requesting a Refinement



1. Requesting a refinement is identical to requesting a new document, except the document locator used is embedded in the chunk to be refined. The client application issues a "GM_GetDoc" request to GloMop, using this embedded document locator as an argument.
2. Per-document state is initialized in the GloMop layer. This state includes QOS preferences, the callback threshold and preload count, callback routines for outstanding chunks, etc.
3. A CM_GetDoc request is forwarded to the chunk manager, giving it the opportunity to initialize relevant data structures for the expected incoming chunks.
4. The GetDoc request parameters are packaged into a chunk and given to the network stack for delivery to the proxy.
5. The application thread returns from the chunk manager once this chunk has been given to the network stack. Concurrently, the network stack queues, schedules, and delivers the outgoing chunk.
6. The application thread returns from GloMop; a requestID for the document is passed back to the application.