Changelog (second revision)

- ➔ The XRootD ability to federate different sides through meta managers together with additional functionalities provided by the AAA (like logical file name translation to physical file name) allowed to achieve a global, multi-site environment for data storage and analysis.
  - I've replaced the old text with your suggestion 😉
- ➔ through a TCP implementation -> over TCP protocol.
  - changed
- ➔ I would rephrase: "The mechanism that allows the client to receive feedback from the TCP kernel is called event-loop. The feedback consists of communicating whether there is available space in the TCP-output buffer for writing data (i.e., requests which will be sent to the server) or if there is some data in the TCP-receive buffer for reading responses from the server." to something like: Using the epoll syscall the XRootD client runtime receives events from the kernel signaling whether there is available space in the TCP-output buffer for writing data (i.e., requests which will be sent to the server) or if there is some data in the TCP-receive buffer for reading responses from the server.
  - Changed accordingly 😉
- ➔ "During a read-event yielded by the event-loop, the client is informed that it can read from the socket, that is a server response." -> During a read-event yielded by the event-loop, the client is informed that it can readout from the socket the server response.
  - Changed accordingly 😉
- ➔ Remarks about figure 4 ✅
  - based on your observations, I've adjusted the figure and it should now look ok
- ➔ Listing 2 appears before Listing 1
  - Changed
- ➔ "It is in fact the response handler that takes care of the function callback once it has been executed; in other words, the handler controls the proper flow of the execution pipeline. The flow of operations follows works in such a way that each next function from the pipeline needs to be called within the handler of the previous function." ***I know what you are trying to say but it might be unclear for the reader ;-) I would go rather for something like:*** In case the user wishes to use only asynchronous operations, the subsequent operation needs to be called from the handler of the previous operation.
  - Replaced the original phrase with your suggestion 😉
- ➔ "Its response handler must have the second operation" -> must call
  - Fixed
- ➔ "The constructed API makes it so there is a communication protocol between the operations:" -> The proposed API provides a syntax for chaining consecutive operations:
  - Fixed
- ➔ The defined operations are connected -> are chained
  - Properly modified
- ➔ Listing 7 is missing the actual call to Parallel**:**
  - called the pipeline in parallel
- ➔ flexibility and fluidity -> fluidity is not the right word here ;-)
  - Dropped the word from the phrase
- ➔ (taking as an argument the lock file itself) -> I would drop this, by lock file you mean the file object named lock, right, this is confusing because previously lock file refers to the actual lock on the remote server
  - Dropped the parentheses.
- ➔ "The Declarative API is tested in the development of an Erasure Coding plug-in for the client." -> the main use case for the declarative API is the development of an erasure coding plugin for the client
  - Text was replaced as suggested
- ➔ The drawback -> the trade off

  - replaced
- "and that can translate into increased latency." -> and that error recovery might result in increased network traffic and latency.
  - Text was changed accordingly 😉
- I would drop this sentence: "In other words, erasure coding adds the redundancy to the system that tolerates failures."
  - Dropped in its entirety 😉
- "In terms of the workflow, EC takes the original data and encodes it in such a way that when needed, only a subset of all the chunks is required to recreate the original information." -> EC encodes n chunks of data (of equal size) in such a way that the result is the n original data chunks and additional m chunks of parity (n+m chunks in total). Every n chunks of the obtained n+m chunks are sufficient to recover the original n chunks.
  - The original phrase was changed as indicated
- "The process of writing the plug- in achieves a high degree of code readability," -> "The obtained code is much more readable, ...
  - Changed accordingly
- "The standard asynchronous operations hide the actual workflow of operations behind the first function callback (e.g. in the Open->Read->Close pipeline, the entire workflow is hidden in the callback of the Open() function)." -> In the contrary, the standard asynchronous operations hide ...
  - Changed accordingly
- I would decrease the font of the listing, so the original format is persisted:
  - The listing was extended full page. Also, some parts of the code were highlighted. 😉
- "Attention was focused on" -> "We focused on ..."
  - I've changed it to "Furthermore, a special focus was given on", since I didn't use the any pronouns in the text 😉
- "The next topic was devoted to the Declarative API," -> Subsequently, we discussed the Declarative API
  - Changed to: Subsequently, a discussion was made on the Declarative API
- "and its main feature is" -> with its main feature being
  - Changed accordingly
- "The Declarative API was also put into usage with the implementation" -> The Declarative API was adopted in the erasure coding plugin implementation ...
  - Changed accordingly
- I'm not sure Figure A1 is correct-> the response handlers are called in the client not the server
  - Moved the response handlers into the "client space" 😉

- **Overall improvement of the structure of the paper by:**
  - **Adding importance of Erasure Coding as a mechanism for data reliability**
  - **Mentioning the drawbacks of the standard async implementation (when compared to the Declarative API)**
  - **Re-structure of the conclusions, making them clearer and more concise.**