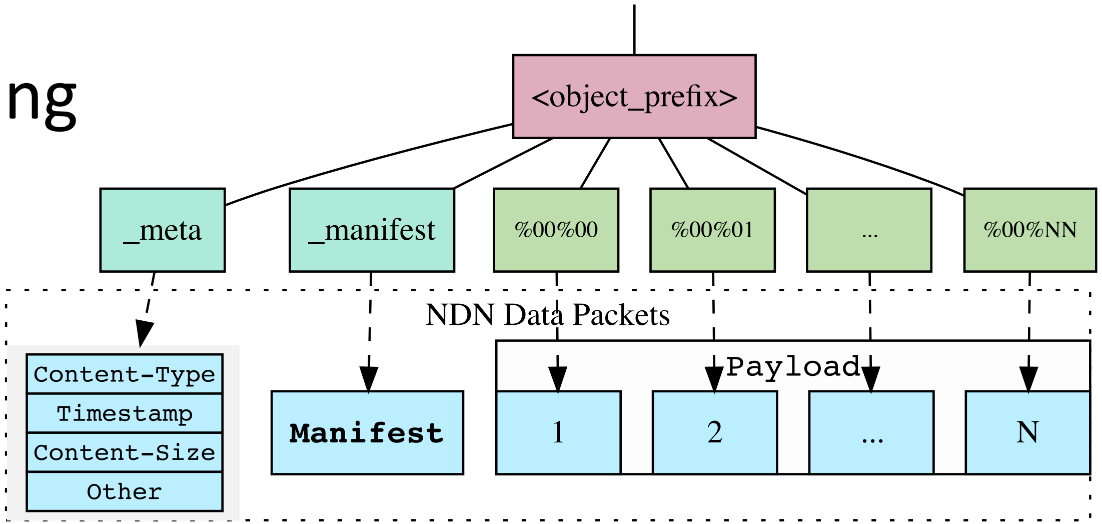
what is needed

- store generalized objects in persistent storage
- store generalized object **streams** in persistent storage
- store **ndnrtc streams** in persistent storage
- serve stored data **immediately**

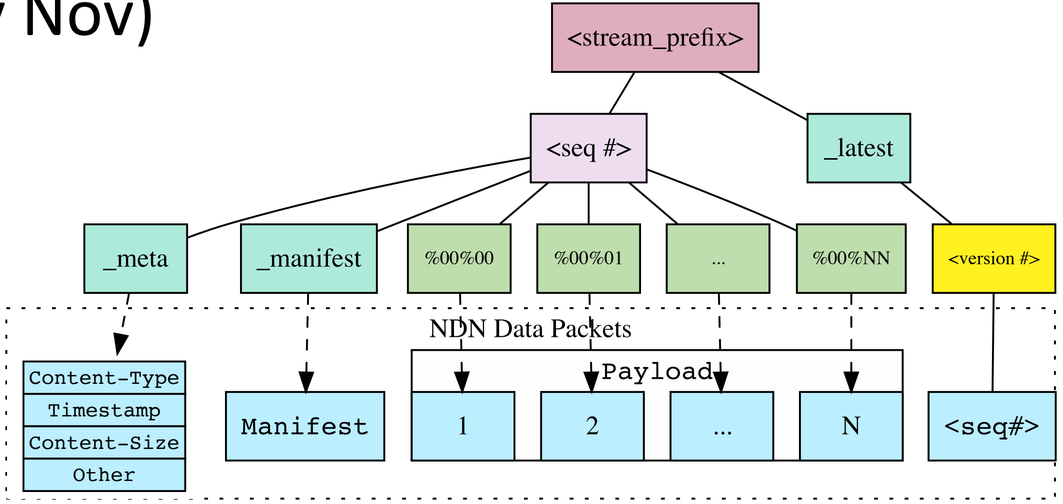
generalized object

- used for publishing annotations
- there is CNL implementation for fetching and publishing



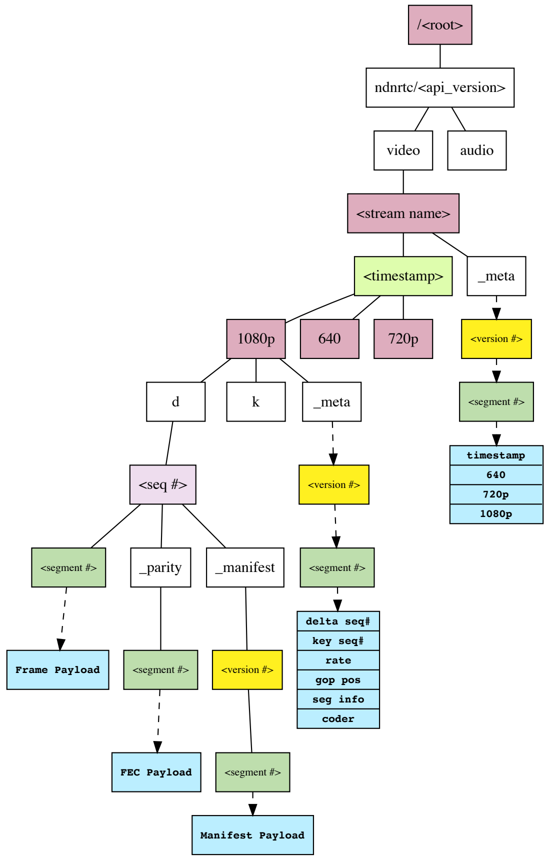
g.o. stream

- allows client to publish generalized objects as a stream
  - (i.e. annotations per frame)
- fetching uses RDR to get latest data
- g.o. stream fetching/publishing will be part of CNL (to be implemented by Nov)



ndnrtc stream

- custom namespace, data + metadata
- will be implemented by Peter and code provided as a module



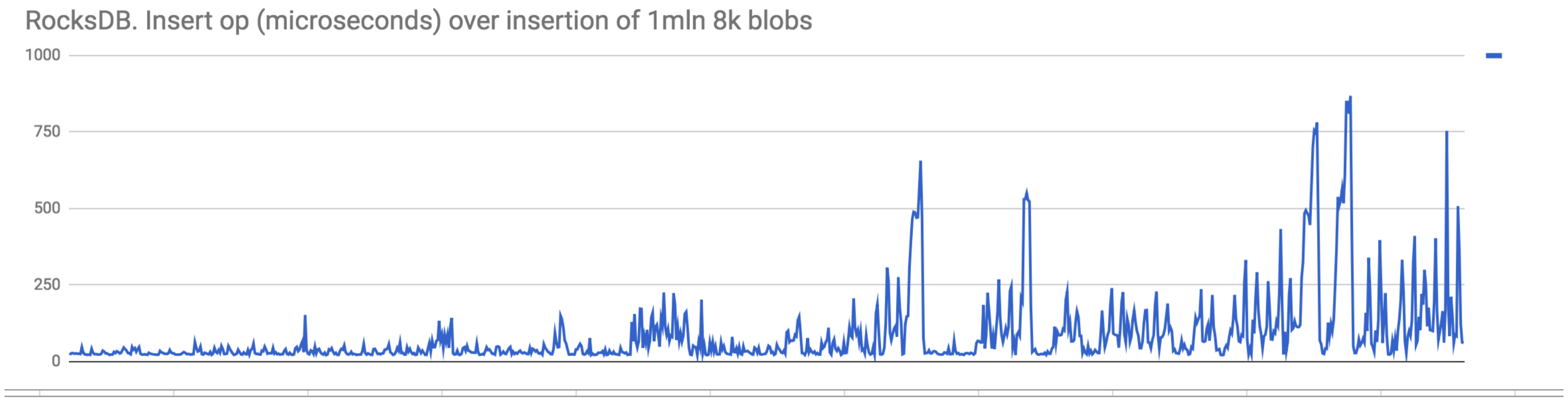
repo extension?

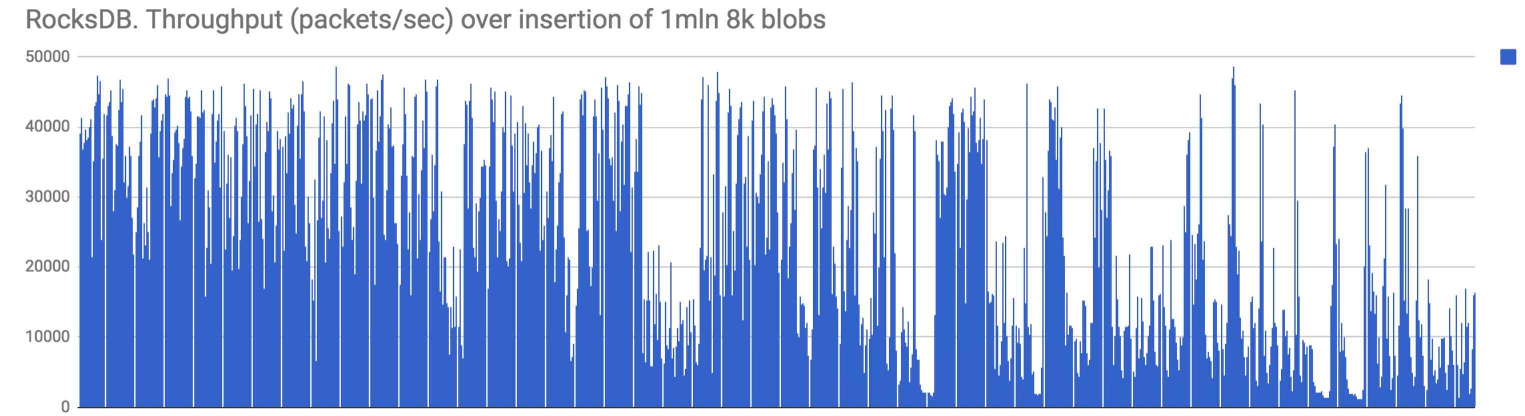
- we want repo to be a stand-alone process running 24/7
- producer "tells" repo:
  - what data to fetch (prefix)
  - how to fetch (fetching pattern)
- repo makes fetched data immediately available for incoming requests (under original prefix)
- repo uses fast key-value store (exact name match) -- RocksDB

implementation

- running code by November:
  - g.o. stream fetch pattern − CNL (JeffT)
  - ndnrtc stream fetch pattern − “ndnrtc”-module (Peter)
  - protocol extension − Xinyu?
- questions:
  - shall we use existing repo code base?
    - this will require adding new dependencies: RocksDB, CNL, ndn-cpp
    - may be the fastest way to go
  - or shall we create a separate codebase (project scaffold w/ dependencies can be provided by Peter quickly)

additional slide: storage benchmark


RocksDB. Insert op (microseconds) over insertion of 1mln 8k blobs


RocksDB. Throughput (packets/sec) over insertion of 1mln 8k blobs