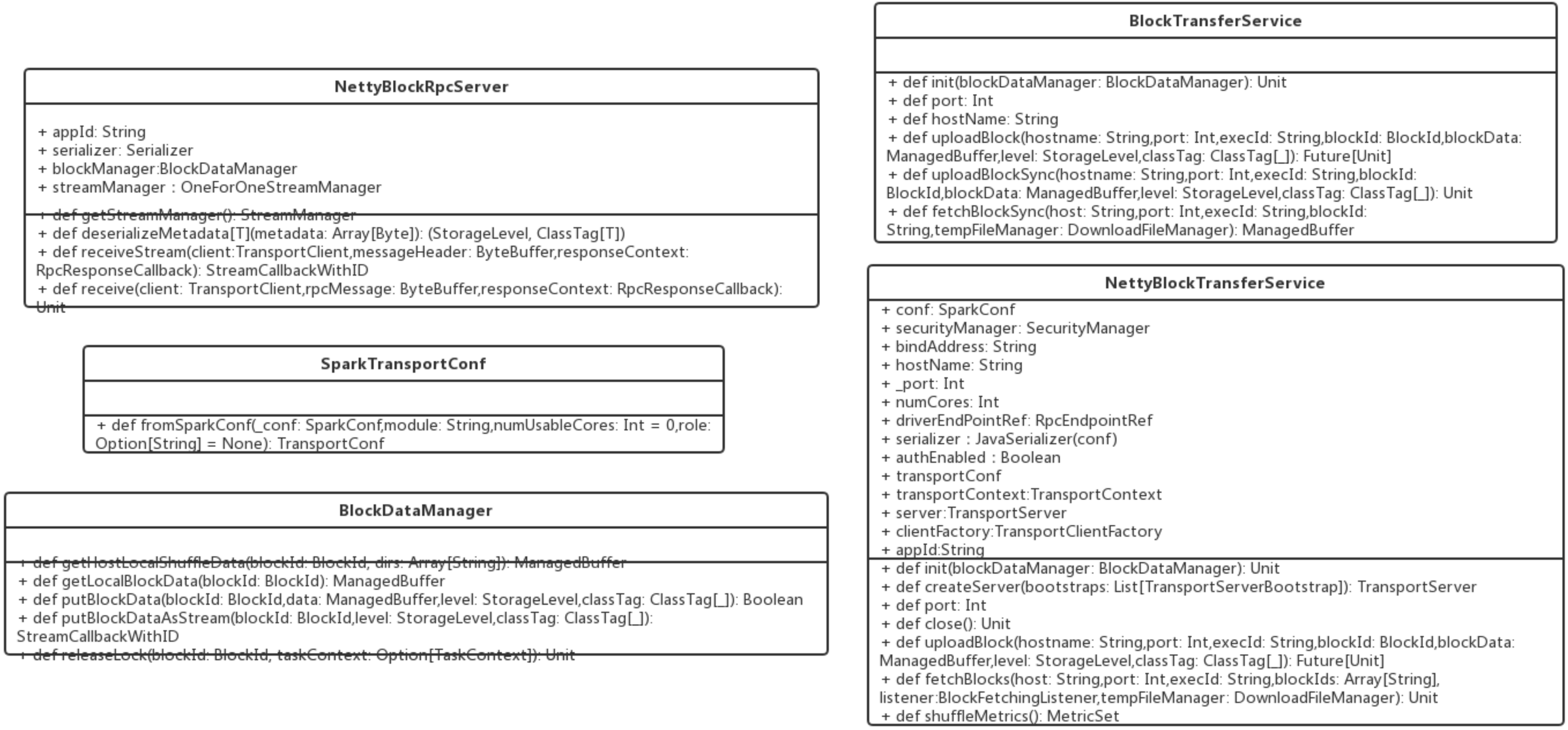
NettyBlockRpcServer
+ appId: String
+ serializer: Serializer
+ blockManager:BlockDataManager
+ streamManager : OneForOneStreamManager
+ def getStreamManager(): StreamManager
+ def deserializeMetadata[T](metadata: Array[Byte]): (StorageLevel, ClassTag[T])
+ def receiveStream(client:TransportClient,messageHeader: ByteBuffer,responseContext: RpcResponseCallback): StreamCallbackWithID
+ def receive(client: TransportClient,rpcMessage: ByteBuffer,responseContext: RpcResponseCallback): Unit
SparkTransportConf
+ def fromSparkConf(_conf: SparkConf,module: String,numUsableCores: Int = 0,role: Option[String] = None): TransportConf
BlockDataManager
+ def getHostLocalShuffleData(blockId: BlockId, dirs: Array[String]): ManagedBuffer
+ def getLocalBlockData(blockId: BlockId): ManagedBuffer
+ def putBlockData(blockId: BlockId,data: ManagedBuffer,level: StorageLevel,classTag: ClassTag[_]): Boolean
+ def putBlockDataAsStream(blockId: BlockId,level: StorageLevel,classTag: ClassTag[_]): StreamCallbackWithID
+ def releaseLock(blockId: BlockId, taskContext: Option[TaskContext]): Unit
BlockTransferService
+ def init(blockDataManager: BlockDataManager): Unit
+ def port: Int
+ def hostName: String
+ def uploadBlock(hostname: String,port: Int,execId: String,blockId: BlockId,blockData: ManagedBuffer,level: StorageLevel,classTag: ClassTag[_]): Future[Unit]
+ def uploadBlockSync(hostname: String,port: Int,execId: String,blockId: BlockId,blockData: ManagedBuffer,level: StorageLevel,classTag: ClassTag[_]): Unit
+ def fetchBlockSync(host: String,port: Int,execId: String,blockId: String,tempFileManager: DownloadFileManager): ManagedBuffer
NettyBlockTransferService
+ conf: SparkConf
+ securityManager: SecurityManager
+ bindAddress: String
+ hostName: String
+ _port: Int
+ numCores: Int
+ driverEndPointRef: RpcEndpointRef
+ serializer : JavaSerializer(conf)
+ authEnabled : Boolean
+ transportConf
+ transportContext:TransportContext
+ server:TransportServer
+ clientFactory:TransportClientFactory
+ appId:String
+ def init(blockDataManager: BlockDataManager): Unit
+ def createServer(bootstraps: List[TransportServerBootstrap]): TransportServer
+ def port: Int
+ def close(): Unit
+ def uploadBlock(hostname: String,port: Int,execId: String,blockId: BlockId,blockData: ManagedBuffer,level: StorageLevel,classTag: ClassTag[_]): Future[Unit]
+ def fetchBlocks(host: String,port: Int,execId: String,blockIds: Array[String],listener:BlockFetchingListener,tempFileManager: DownloadFileManager): Unit
+ def shuffleMetrics(): MetricSet