Collage
C++ Network Library
Technical Overview

Collage in a Nutshell

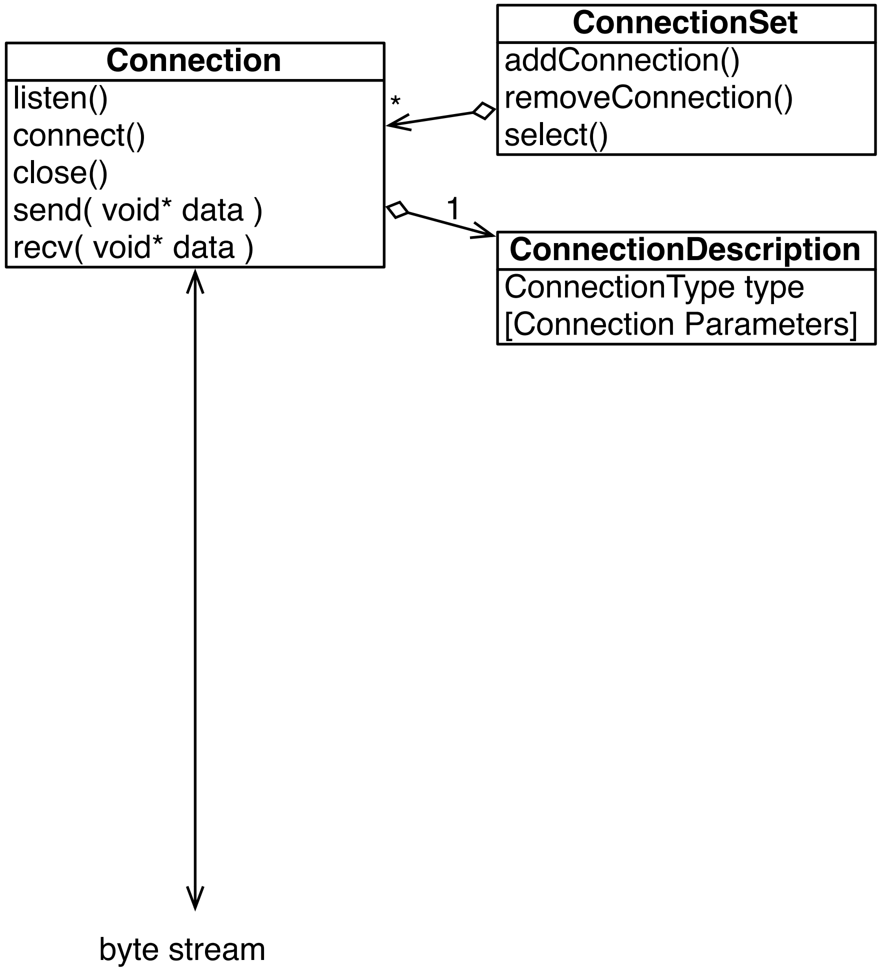
- Many network protocols
  - Unicast: TCP, SDP, RDMA, anon./named pipe
  - Multicast: Reliable Stream Protocol over UDP
- Peer-to-peer node communication
  - Extensible, message-oriented communication
- Distributed, versioned C++ objects
  - Push-based commit - sync protocol
  - Multicast, compression plugins

Collage Architecture

- **Connections** used by...
- **Node**s in a peer-to-peer network managing...
- Distributed, versioned **Object**s

Connection

- Stream-oriented C++ interface
- Abstracts different implementations
- Could be replaced by boost::asio or 0MQ
  - Need support for RSP and RDMA

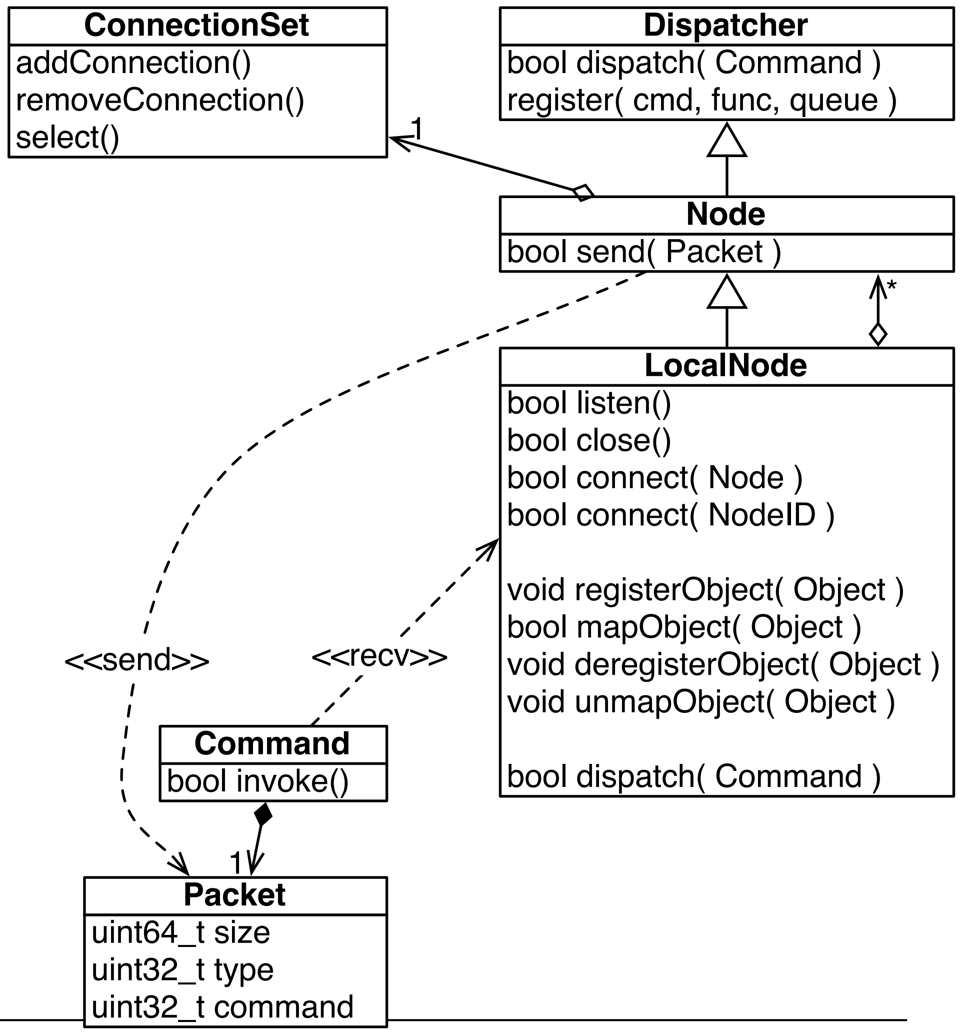


RSP Connection

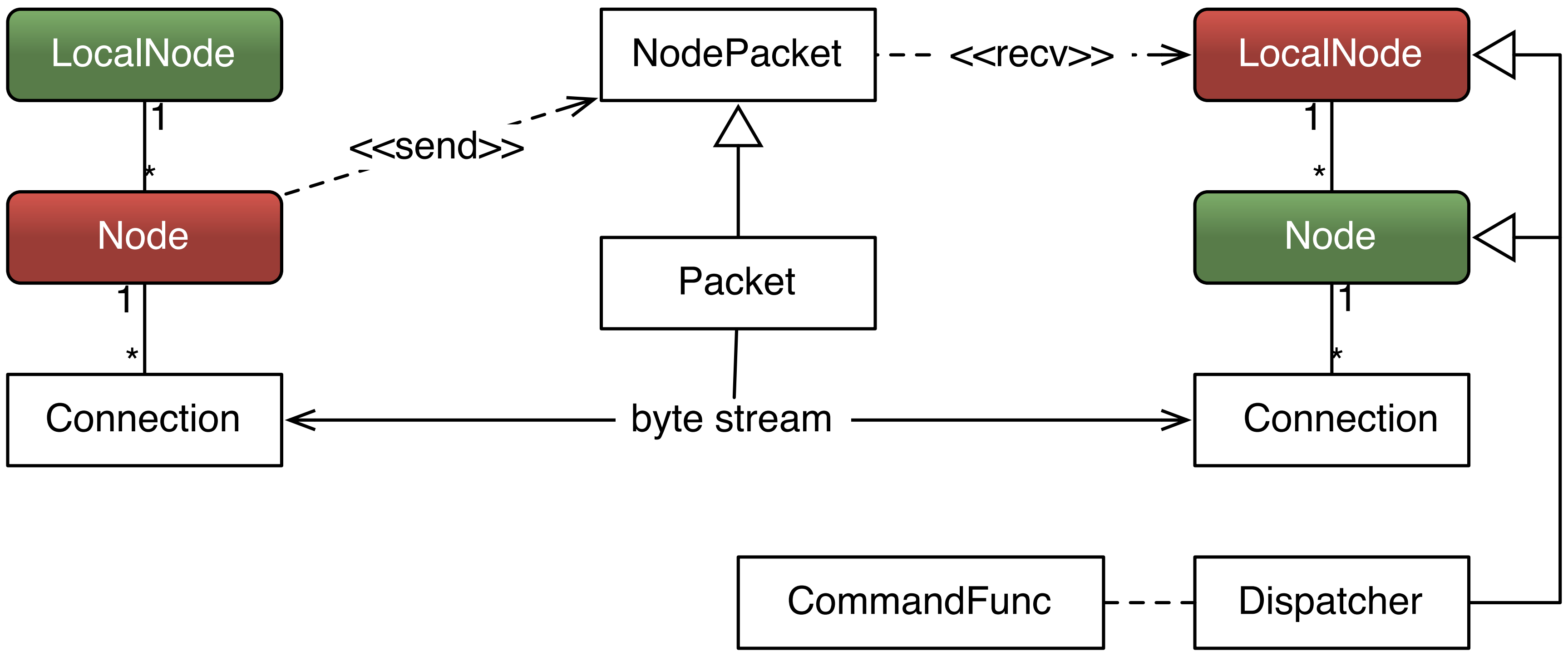
- Reliable Stream Protocol over UDP multicast
- Full reliability: receiver may throttle sender
- Sliding send window (~4MB)
- Early nacks, scattered early acks, lazy ack req
- Tuned for LAN performance
  - Lock-free read and write queues
  - Merging of small writes and nacks
  - Aggressive congestion control

Nodes

- Nodes use Packets
- Fast packet-to-method dispatch
- LocalNode: local listen, receive and dispatch
- Node: proxy of remote LocalNode



Nodes


LocalNode
Node
Connection
NodePacket
Packet
<<send>>
<<recv>>
byte stream
LocalNode
Node
Connection
CommandFunc
Dispatcher
1
*
1
*
1
*
1
*

Nodes

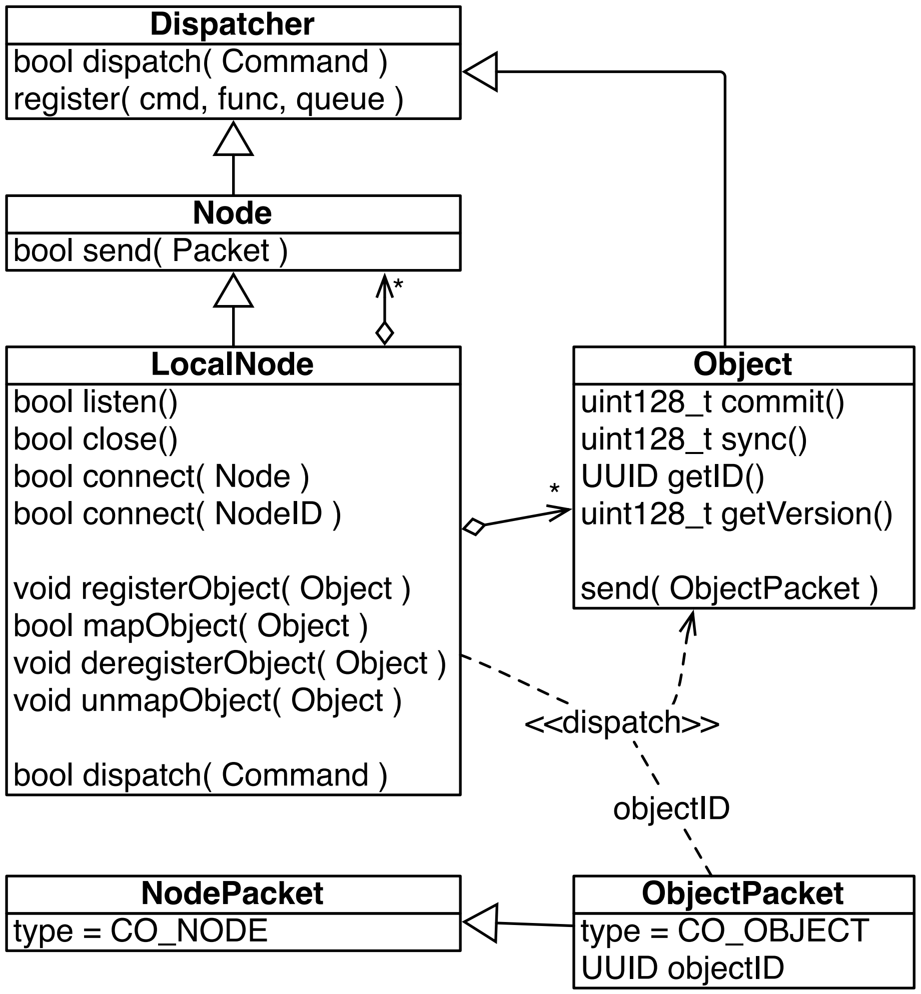
- Connect using:
  - Explicit Node with connection descriptions
  - Node identifier: queries peers, then Zeroconf
- Zeroconf “_collage._tcp”:
  - LocalNode announces Node ID and connections
  - Can be augmented with key/value pairs
  - Can be used to browse announced Collage services
- LocalNode manages objects and data cache

Objects

- Distributed, versioned objects
- Application manages lifetime of instances
- UUID (uint128) to address across processes
- Version (uint128) for synchronization
- Generic co::Object, simplified by co::Serializable
- ObjectMap facilitates management

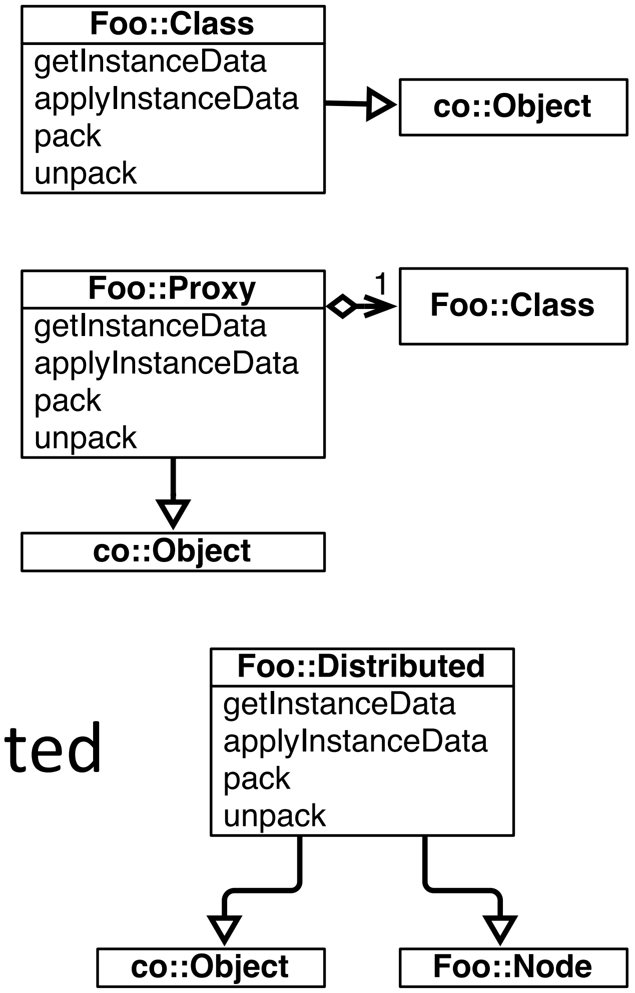
Objects

- Register master instance
- Map slave instances to master identifier
  - Pulls instance data
  - Push-based possible
- v = master.commit()
- slave.sync(v)



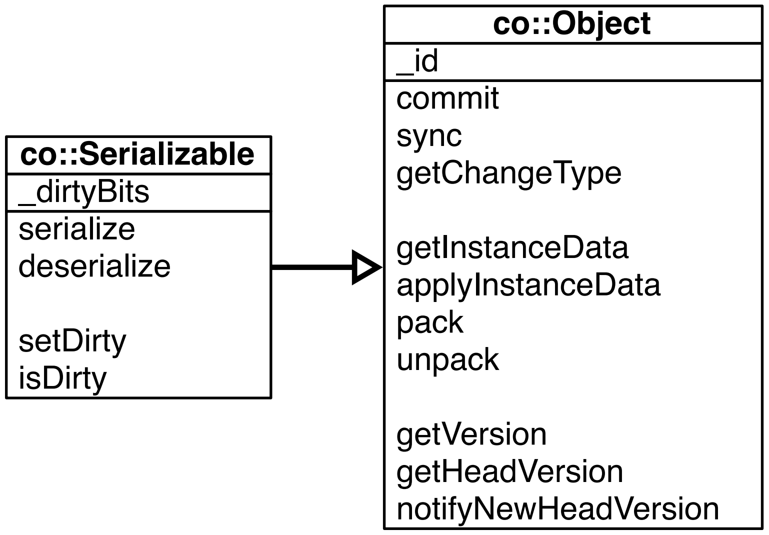
Object Usage

- Approach: subclass, proxy or multiple inheritance
- Type: static, unbuffered, instance or delta
- Implement serializers
  - DataO/IStream interface
  - Common data types implemented
  - Buffering, compression

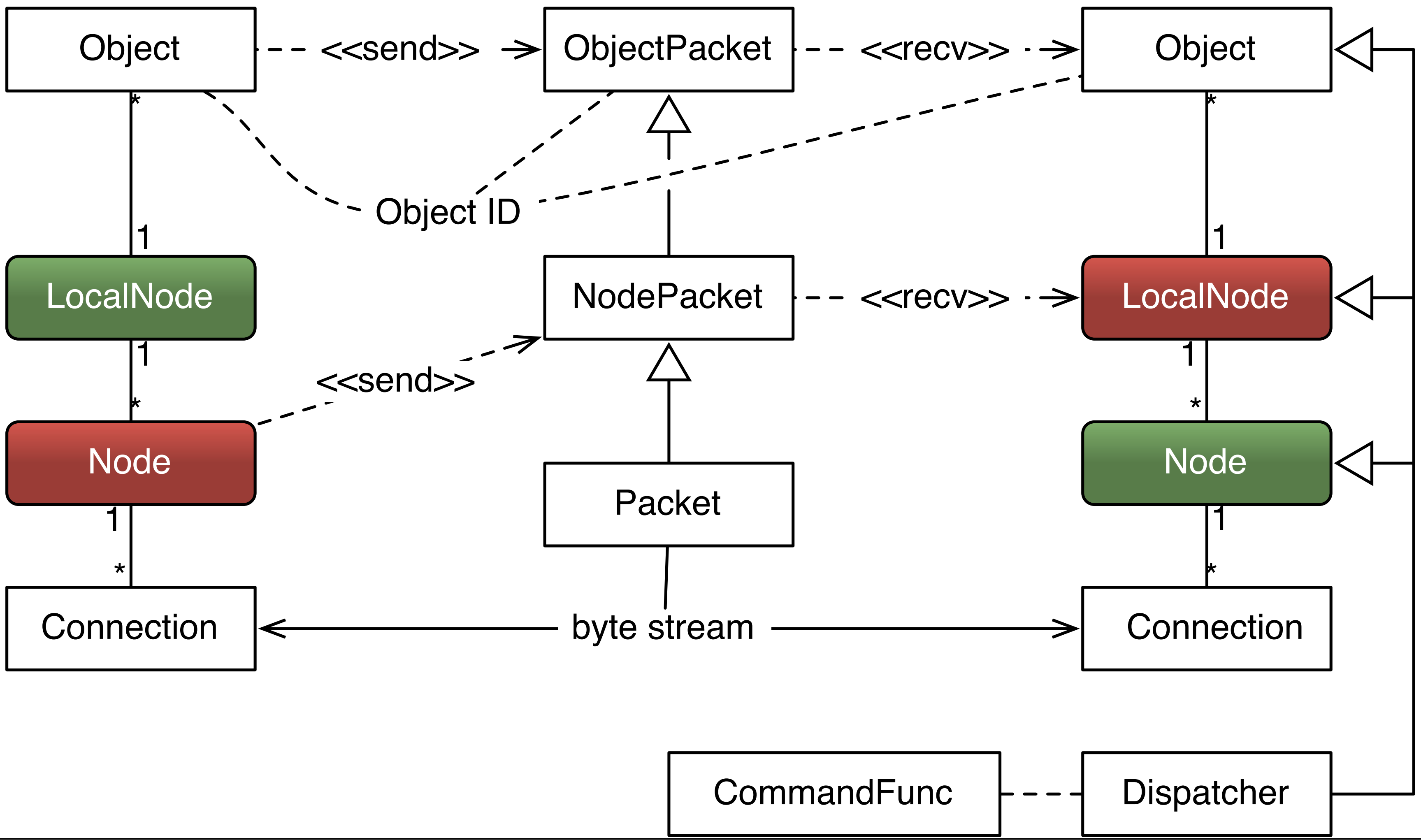


Object vs Serializable

- Object:
  - Two serializer pairs
  - External dirty state
- Serializable:
  - One serializer pair
  - 64 bit dirty mask
  - Single inheritance contract



Object Communication



Objects

- Barrier
  - Per-version height
  - Master-Slave protocol
- QueueMaster - QueueSlave
  - Single-producer, multiple-consumer FIFO
  - Configurable prefetching
- ObjectMap
  - Maps, commits and syncs objects

Users

- Equalizer
  - All internal shared objects
  - Large-scale scalable rendering (>100 nodes)
  - Large low-latency VR installation (up to 100 nodes)
- CoDASH distribution library for dash graphs
  - Monsteer monitoring and steering library

Future

- Separate project on github
- Stable API definition
- Endian handling
- Better compression plugins
- Message-based connections
- ‘Multicast’ over unicast trees