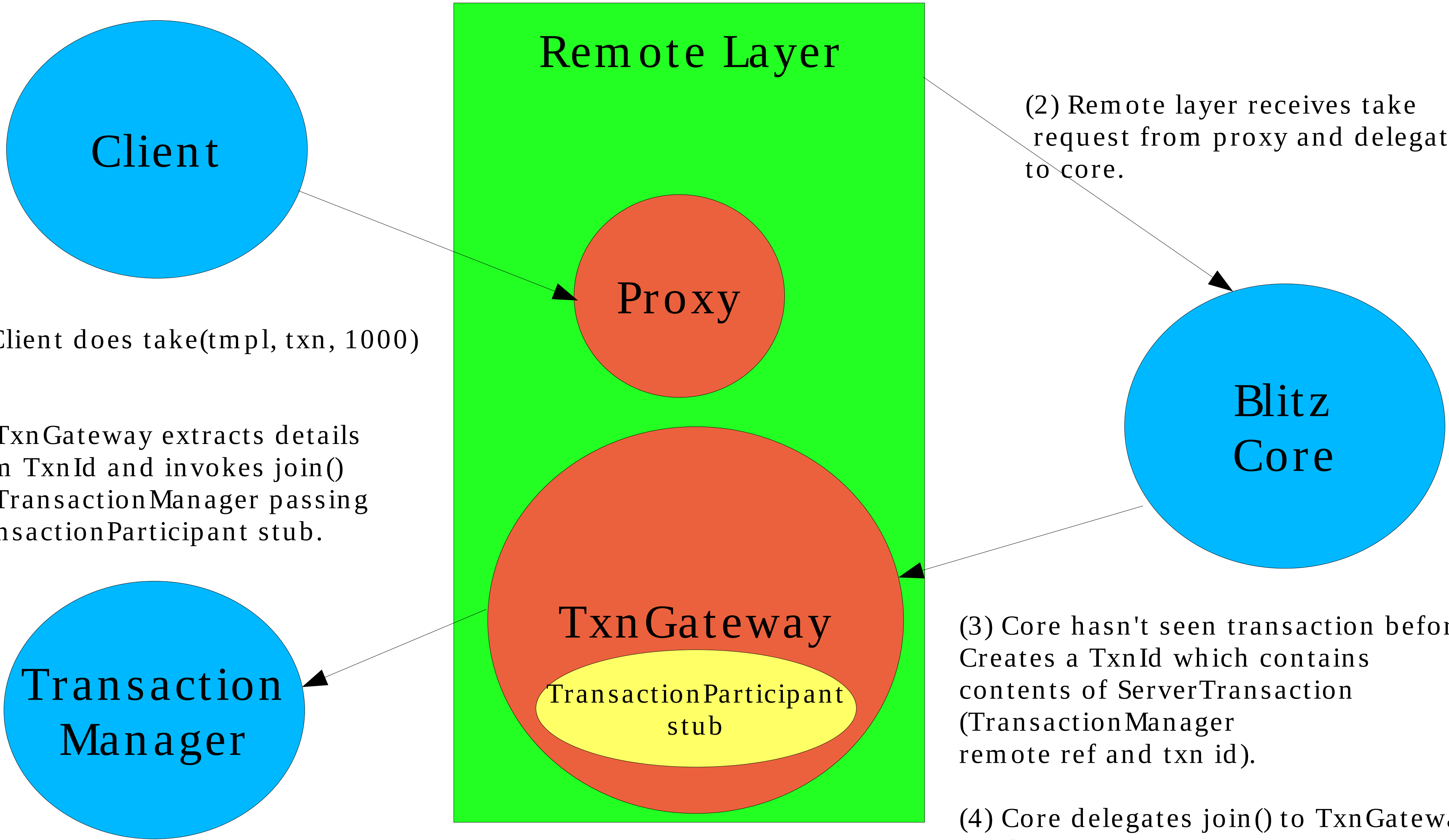
# Using Blitz in Embedded Mode

# Basic Blitz Architecture



(1) Client does take(tmpl, txn, 1000)

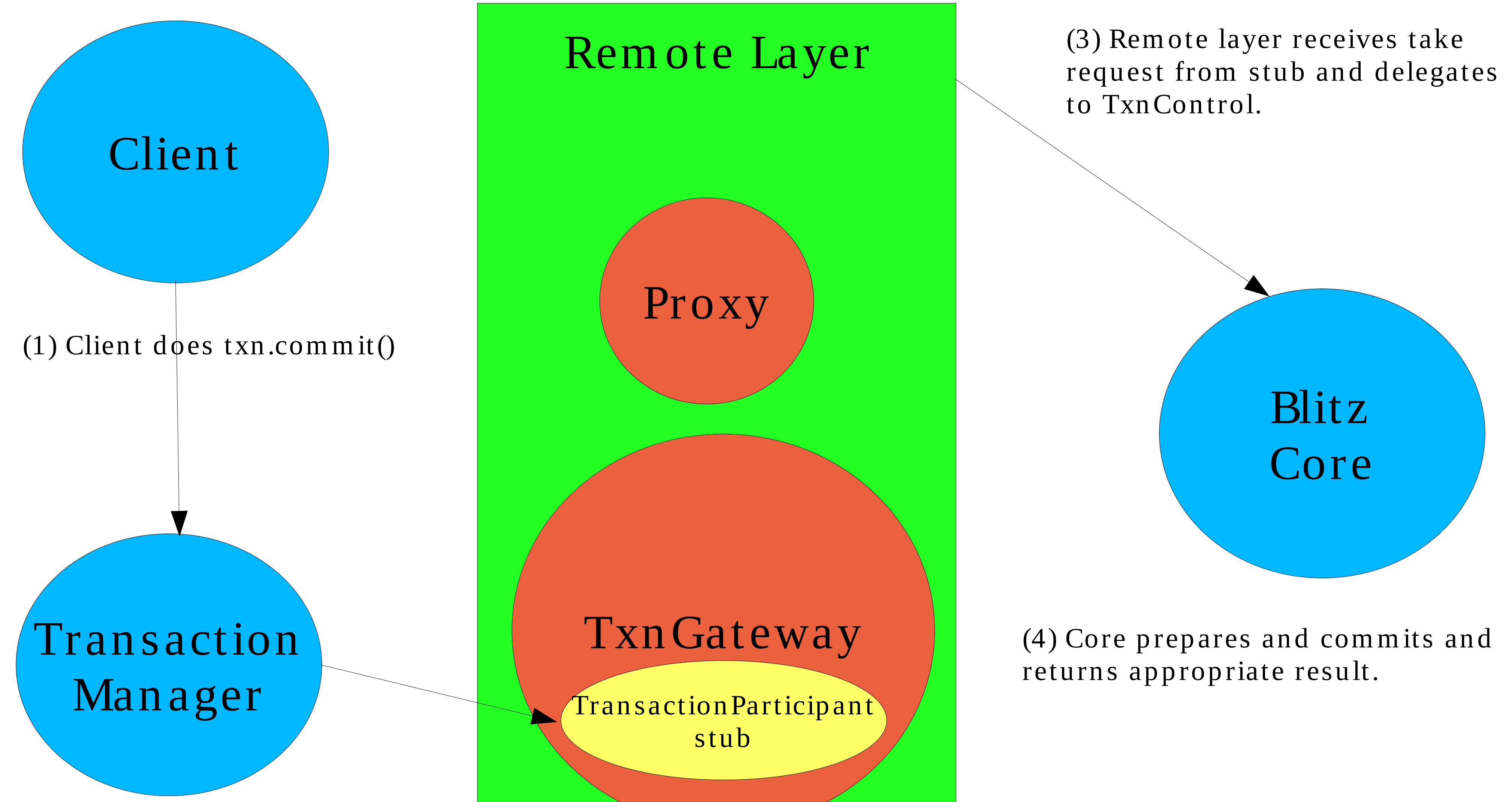
(2) Remote layer receives take request from proxy and delegates to core.

(3) Core hasn't seen transaction before. Creates a TxnId which contains contents of ServerTransaction (TransactionManager remote ref and txn id).

(4) Core delegates join() to TxnGateway implementation.

(5) TxnGateway extracts details from TxnId and invokes join() on TransactionManager passing TransactionParticipant stub.

# Basic Blitz Architecture



(1) Client does txn.commit()

(2) TransactionManager invokes prepareAndCommit() on TransactionParticipant stub

(3) Remote layer receives take request from stub and delegates to TxnControl.

(4) Core prepares and commits and returns appropriate result.

# Embedded Usage

Blitz Core knows nothing of remoteness and delegates it out to implementations of interfaces. These implementations are passed in at construction time. Typically, the remote layer provides implementations of these interfaces.

Who provides these implementations when using LocalSpace which has no remote layer?

Answer: It's the responsibility of the application developer.

If you do not plan to use transactions at all, simply pass null when constructing a LocalSpace:

```
LocalSpace myLocalSpace = new LocalSpace(null);
JavaSpace mySpace = myLocalSpace.getProxy();
```

If you do want to use transactions, you must create a TxnGateway implementation which will act as “glue” between the Blitz core and whatever transaction manager you propose to use. Your TxnGateway implementation must provide a suitable stub to the transaction manager which it can invoke on. When the transaction manager does invoke on the stub, you should delegate to LocalSpace.getTxnControl().

A totally local implementation would consist of a null TxnGateway and usage of TxnControl as follows:

```
public class TxnGatewayImpl implements TxnGateway {
    public int getState(TxnId anId) {
        return TransactionConstants.COMMITTED;
    }

    public void join(TxnId anId) {
        System.out.println("Join: " + anId);
    }
}
```

**continued to next page......**

# Embedded Usage

```
public class TxnMgr implements TransactionManager, Serializable {
    private long theId;

    public TxnMgr(long anId) {
        theId = anId;
    }

    public boolean equals(Object anObject) {
        if (anObject instanceof TxnMgr) {
            TxnMgr myOther = (TxnMgr) anObject;

            return (myOther.theId == theId);
        }

        return false;
    }

    public Created create(long lease) throws LeaseDeniedException,
                                 RemoteException {
        throw new org.dancres.util.NotImplementedException();
    }

    public void join(long id, TransactionParticipant part,
                         long crashCount)
        throws UnknownTransactionException, CannotJoinException,
               CrashCountException, RemoteException {
    }


    public int getState(long id)
          throws UnknownTransactionException,
                              RemoteException {
        return TransactionConstants.COMMITTED;
    }

    public void commit(long id)
        throws UnknownTransactionException,
               CannotCommitException,
               RemoteException {
        throw new org.dancres.util.NotImplementedException();
    }

public void commit(long id, long waitFor)
        throws UnknownTransactionException, CannotCommitException,
               TimeoutExpiredException, RemoteException {
        throw new org.dancres.util.NotImplementedException();
    }


    public void abort(long id)
        throws UnknownTransactionException, CannotAbortException,
               RemoteException {
        throw new org.dancres.util.NotImplementedException();
    }

    public void abort(long id, long waitFor)
        throws UnknownTransactionException, CannotAbortException,
               TimeoutExpiredException, RemoteException {
        throw new org.dancres.util.NotImplementedException();
    }
} // End Txn Mgr
```

**Client does:**

```
// Initialization
TxnMgr myMgr = new TxnMgr(0);
long myNextTxnId = 0;

LocalSpace myLocalSpace = new LocalSpace(new TxnGatewayImpl());
JavaSpace mySpace = myLocalSpace.getProxy();

// Create a transaction
ServerTransaction myTxn = new ServerTransaction(myMgr, myNextTxnId++);

mySpace.write(........, myTxn, Lease.FOREVER);
myLocalSpace.getTxnControl().prepareAndCommit(myMgr, myTxn.id);

// Create next transaction
myTxn = new ServerTransaction(myMgr, myNextTxnId++);
mySpace.take(........, myTxn, Lease.FOREVER);
myLocalSpace.getTxnControl().prepareAndCommit(myMgr, myTxn.id);
```