Tutorial: Adding new messages in Protobuf

Michael Schreier

This document intends to give an advice for adding new messages in Protobuf (with C++ in ANNarchy) and Protobuf-Net (with C# in SimpleNetwork). I will go through the process by example by adding the new message *MsgAgentGrapPos*, but neither explain the message syntax nor how to use it.

1 Adding the message in ANNarchy

1. Open *ANNarchy-2.2/protobuf/MsgObject.proto*.
2. Add the listing

```
message MsgAgentGraspPos {
  required int32 actionID = 1;
  required float targetX = 2;
  required float targetY = 3;
}
```

3. Add a new line in the existing *MsgObject*.

```
optional MsgAgentGraspPos msgAgentGraspPos = 13;
```

   Here I added it as the 13th entity, so note that each new message must have a new ID.

4. Run

```
protoc MsgObject.proto --cpp_out=.
```

5. Copy the resulting *MsgObject.pb.cc* and *MsgObject.pb.h* in ANNarchys *include* and *src* directory.
6. Optionally: Open *MsgObject.pb.cc* and customize the line

```
#include "MsgObject.pb.h"
```

   for your purpose or change the include directories in Eclipse.

7. Now the new message is ready to be used.

8. The last step is adding a new public member function to the *AnnarProtoSend* class. This function will encapsulate the construction and sending of the *MsgObject*. Don't forget to add the function in both, *AnnarProtoSend.h* and *AnnarProtoSend.cpp*.

```
int AnnarProtoSend::sendGraspPos(float targetX, float targetY)
{
  MsgObject * unit = new MsgObject();
  int id = rand();
  unit->mutable_msgagentgrasppos()->set_actionid(id);
  unit->mutable_msgagentgrasppos()->set_targetx(targetX);
  unit->mutable_msgagentgrasppos()->set_targety(targetY);

  mutex = true;
  sentQueue.push(unit);
  mutex = false;
  return id;
}
```

2 Adding the message in SimpleNetwork (with Protobuf-Net)

1. Open the *SimpleNetwork*-Project.

2. Add a new class *MsgAgentGraspPos* in the *Msg* folder. This will create the new file *MsgAgentGraspPos.cs*.

3. Edit *MsgAgentGraspPos.cs*.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ProtoBuf;

namespace SimpleNetwork
{
    [Serializable, ProtoContract]
    public class MsgAgentGraspPos
    {
        [ProtoMember(1, IsRequired = true)]
        public Int32 actionID;
        [ProtoMember(2, IsRequired = true)]
        public float targetX;
        [ProtoMember(3, IsRequired = true)]
        public float targetY;
    }
}
```

4. Edit *MsgObject.cs* and add

```
        [ProtoMember(13)]
        public MsgAgentGraspPos msgAgentGraspPos;
```

Here, same as above. Each new entity must have a new ID.

5. Build *SimpleNetwork*-Project.

6. Now the assembly *SimpleNetwork.dll* can be used with the new message *MsgAgentGraspPos*.

7. In our Project "Kinderzimmer"(Children's room), the *update* functions of *BehaviourScript* and *AgentScript* are responsible for the message handling. For each protobuf message, there also exists a protected virtual function, which will be customized in derived classes of *AgentScript*.

   Here is the *update* of *AgentScript*

```
void Update()
{
  #region handle incoming msg
  if(MySimpleNet != null){

  if (MySimpleNet.MsgAvailable()) {

    MsgObject NextMsg = MySimpleNet.Receive();

    [...]

    if(NextMsg.msgAgentGraspPos != null) {
                    processMsgAgentGrapPos( NextMsg.msgAgentGraspPos );
            }

    [...]

  }
}
```

   and the message handling function for *MsgAgentGraspPos*

```
protected virtual void processMsgAgentGrapPos( MsgAgentGraspPos msg )
    {
        Debug.Log(String.Format("AgentGraspPos recived ({0:f},{1:f})",
                        msg.targetX,
                        msg.targetY));
    }
```

   For a new protobuf message, the if-switch in the *update* function must be extented and a new protected virtual function added.

3 Appendix: Sending in C#/Receiving in C++

I explained in the first two sections the work process for adding the *MsgAgentGraspPos* message. This section explains, what to do, if you want to receive data in C++ and send data from C# with a message (e.g. the *MsgGridPosition* message).

Sending data from C#

Sending protobuf data is simple. For example *MsgGridPosition* is sent in the *AgentScript Update()* with:

```
void Update()
{
  [...]
  if (SendGridPosition) {
    MySimpleNet.Send(new MsgGridPosition(){
      targetX = gameObject.transform.position.x,
      targetY = gameObject.transform.position.y,
      targetZ = gameObject.transform.position.z
    });
  }
  [...]
}
```

Receiving data in C++

1. Add and initialize member variables for the *MsgGridPosition* message in AnnarProtoReceive.

   AnnarProtoReceive.h

   ```
   class AnnarProtoReceive{
     [...]
     bool validGridsensor_;
     float targetX_;
     float targetY_;
     float targetZ_;
     [...]
   }
   ```

   AnnarProtoReceive.cpp

   ```
   AnnarProtoReceive::AnnarProtoReceive(int socketVR, int socketAgent)
   {
     [...]

     validGridsensor_ = false;
     targetX_ = 0;
     targetY_ = 0;
     targetZ_ = 0;

     [...]
   ```

```
}
```

2. Add a new getter function for this message.

   AnnarProtoRecive.cpp

```
bool AnnarProtoReceive::getGridsensorData(float& targetX, float&
    targetY, float& targetZ)
{
  if(!validGridsensor_)
    return false;
  waitForMutexUnlock();

  targetX = targetX_;
  targetY = targetY_;
  targetZ = targetZ_;
  return true;
}
```

3. Add a new if-switch to the *AnnarProtoReceive::storeData* function.

```
void AnnarProtoReceive::storeData(long int dataLength)
{
[...]
  if(tmp->has_msggridposition())
  {
    assert(tmp->msggridposition().has_targetx());
    mutex = true;
    targetX_ = tmp->msggridposition().targetx();
    targetY_ = tmp->msggridposition().targety();
    targetZ_ = tmp->msggridposition().targetz();
    validGridsensor_ = true;
    mutex = false;
  }
[...]
}
```

4. Now the data of a *MsgGridPosition* message can be received by

```
receiver->getGridsensorData(agentX, agentY, agentZ)
```