Bridging Communities with pMOOSBridge

Paul Newman

March 17, 2009

Abstract

This document will give you a description of how to use pMOOSBridge to link multiple MOOS communities together.

1 Introduction

pMOOSBridge is a powerful tool in building MOOS-derived systems. It allows messages to pass between communities and is able to rename the messages as they are shuffled between communities. Many of the sections in this document rely on pMOOSBridge to set up different communications topologies. There is no correct topology — choose one that works for your own needs. One in-

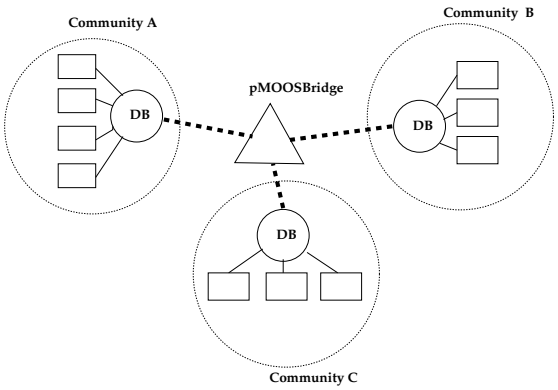

Figure 1: A possible MOOSBridge configuration. One instance of pMOOSBridge can "talk" to a limitless number of communities. The configuration block specifies what should be mapped or "shared" between communities and how it should be done. The SHARE command specifies precisely what variables should be shared between which communities.

stance of pMOOSBridge can "talk" to a limitless number of communities. The

configuration block specifies what should be mapped or "shared" between communities and how it should be done. The `SHARE` command specifies precisely what variables should be shared between which communities and the syntax is intuitive:

`SHARE=` *Comm@Host:Port[V1[,V2...]] -> Comm@Host:port [V1,V2...]*

The triplet Comm@Host:Port is a description of a community — name and hostname/port pair. The community description can be omitted on the LHS of the arrow, in which case the mission-file-scope defaults are assumed (see the example below). Each variable ("V") on the LHS (source) community will be inserted into the community on the RHS If no variable names are specified on the RHS (destination) community the original names are used, otherwise there is a one-to-one mapping between variable names on the LHS and new variable names (aliases) on the RHS. If however there are more named shared variables than aliases, the variables for which an alias is not specified retain their original names.

For example

```
SHARE= VehA@nym.robots.ox.ac.uk:9000 [GPS_X]->VehB@kayak.mit.edu:9000 [GPS_X]
```

Here the variable `GPS_X` is shared between a community called "VehA", running from a `MOOSDB` on the machine called "nym.robots.ox.ac.uk" listening on port 9000, is being inserted into a community called "VehB" using a `MOOSDB` running on the machine called "VehB@kayak.mit.edu" also listening on port 9000. In both communities the variable is called "GPS_X". However when viewed with something like `uMS` (see the document on Graphical Tools) it can be seen that the `m_sOrginatingCommunity` member of the `MOOSMsg` carrying this data in the "VehB" commmunity will be "VehA".

```
SHARE= VehA@nym.robots.ox.ac.uk:9000 [GPS_X]->VehB@kayak.mit.edu:9000 [GPS_X_A]
```

This is similar to the above example, only `pMOOSBridge` will rename "GPS_X" to "GPS_X_A" in the destination community. The next example shows how the source address need not be specified. When omitted the source community is taken to be the community on which `pMOOSBridge` is running at the time. This example also shows how destination variable names may be omitted, in which case the original (source community) variable name is preserved.

```
SHARE= [GPS_X]->VehB@kayak.mit.edu:9000
```

Finally, more than one mapping can be specified in one line:

```
SHARE= [GPS_X,OVEN_TEMP]->VehB@kayak.mit.edu:9000 [GPS_X_A]
```

Here GPS_X is being mapped and renamed to `GPS_X_A` in community "VehB" but the variable `OVEN_TEMP` is simply being shared without renaming. It is important to realise that sharing is not bidirectional. In this case, a process notifying change in `GPS_X_A` in community "VehB" *would not* result in pMOOSBridge notifying the `MOOSDB` in community "VehA" that "GPS_X" has changed.

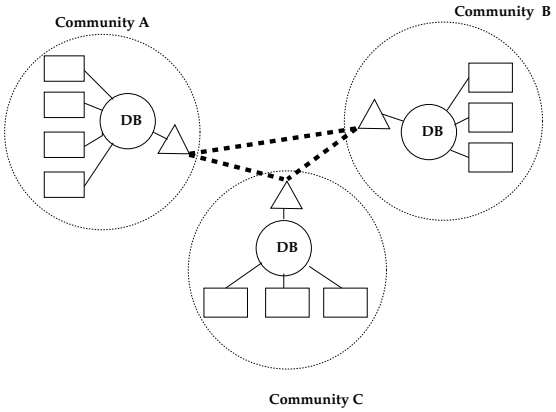

Figure 2: An alternative MOOSBridge Configuration — one bridge per community. This may be preferable; it is undesirable to have one process manage all the sharing of data. However it offers no *functional* advantage over the topology shown in Figure 1.