Добрый день NNN и MMM.

Для связи нашей измерительной системы с вашей АСУ нам надо договориться о протоколе обмена. Я предлагаю использовать **DIM**.

Я сделал тестовый пример для оценки технологии DIM.
Пример самый тупой в том смысле, что передаваемые данные ничего не значат, это просто иллюстрация способа обмена.
Более содержательную информацию о данных я пришлю позже.

dim - в каталоге архив DIM.
dim_dcc - в каталоге библиотека для Delphi.
dim_demo - в каталоге содержится демонстрационный пример.
Запускать надо файл demo.bat.
Тест запускает сервер, два клиента, а также утилиту dimtree для просмотра данных.
В файле screenshot.jpg показано как выглядит экран если все в порядке.
Для работы нужны dim.dll, msv*.dll, dns.exe и dimtree.exe.
При этом dim.dll, msv*.dll линкуются с серверным/клиентским приложением, а dimtree.exe - просто полезная утилита, она необязательная. Все исходники DIM есть в дистрибутиве на сайте http://dim.web.cern.ch, а msv*.dll - это просто runtime библиотеки из Visual C++, под которым компилировался dim.dll. Можно в принципе скомпилировать эту библиотеку и другим компилятором, чтобы избежать этой необходимости таскать за собой msv*.dll, но я не пробовал.

При запуске брэндмауэр Windows может ругаться, надо ему разрешить запуск dns.exe, server.exe, client.exe.

Для примера сервер публикует 2 сервиса DEMO/INFO1, DEMO/INFO2. Для примера взяты записи, в которых содержится время в разных форматах

```
Info1       : record                // 1 user published data set
  CallCount  : Integer;              // Call counter
  TickCount  : Integer;              // GetTickCount
  Now        : Double;               // Current time
 end;
 Info2       : record                 // 2 user published data set
  FileTime   : TFileTime;             // System time as file time
  DateTime   : array[byte] of char; // Date and time string
 end;
```

Клиент подписывается на эти сервисы и получает данные. Кроме того в клиенте можно посылать команды серверу, реализована только команда Reset, которая сбрасывает счетчик Info1.CallCount в 0, чисто для примера.

Технология DIM - Distributed Information Manager - средство для организации больших многомашинных систем управления в реальном времени. По идеологии она, наверное, ближе всего к OPC, но существенно проще, быстрее и понятнее. Это - лучшая сетевая технология, с которой я до сих пор встречался. OPC, конечно, круто - но уж больно тяжелая, как все мелкософтовские монстры. Разработан DIM в CERN (центрально-европейский центр ядерных исследований, Женева). Распространяется под лицензией GPL (freeware, open source). Загрузить можно с домашнего сайта http://dim.web.cern.ch. Технология много платформенная, как аппаратно, так и в смысле операционных систем и языков программирования. Работает в Win, Linux, Unix, LynxOs, Qnx etc. Работает на C, C++, Fortran, Delphi.

Хотя DIM основан на TCP/IP сокетах, он предоставляет абстракцию более высокого уровня, чем обычный client-server. Работа организуется примерно так.

Основное понятие DIM - не соединение, как в сокетах, а СЕРВИС - поименованный набор данных произвольной структуры, например, запись (в смысле Pascal). Сервер, то есть процесс, владеющий данными, ПУБЛИКУЕТ данные, а также следит за их своевременным обновлением. Клиент может в любой момент ПОДПИСАТЬСЯ на

интересующий сервис, после чего он станет получать данные по мере их обновления сервером. Клиентов может быть сколько угодно, без всяких усилий со стороны сервера. Когда пишется сервер, ничего не надо знать о клиентах - сколько их, какой у них адрес и т.д. Сервер просто предоставляет ИМЯ сервиса и Callback процедуру, которая должна вызываться при необходимости обновить данные. Все остальное делает DIM. Клиент, в свою очередь, не должен знать, где искать сервер, как к нему подключатся и т.д. Он должен знать только имя сервиса и предоставить Callback процедуру, которая должна вызываться при необходимости прочитать данные на стороне клиента. Во всей технологии существенную роль играет сервер имен DNS (Distributed Name Server), не путать с DNS в TCP/IP. На одной из машин (имя этой машины и есть имя DNS) запускается программа DNS.EXE - сервер имен (обычно в скрытом виде, без окна, чтоб случайно не убить). Задачей сервера имен является поддержание списка клиентов и серверов DIM. Сервер, когда публикует данные, отсылает их в базу данных DNS. Клиент, когда подписывается на данные, подключается к DNS и спрашивает его, как найти сервер, который предоставляет интересующий сервис. Все это скрыто в недрах DIM. С точки зрения программиста, клиенту и серверу надо знать только имя DNS и имя сервиса. Имена и IP адреса клиента и сервера вообще никого не интересуют, это забота DNS. Таким образом, например, в больших системах (десятки машин) для организации сети надо знать имя только одного компьютера (сервера DNS), вместо бесчисленного числа пар клиент-сервер. Имена сервисов являются внутренней договоренностью DIM серверов и клиентов и не зависят от сетевой конфигурации машин.
Информационные сервисы (про которые шла речь выше) асимметричны - данные идут от сервера к клиентам. Для управления сервером клиенты могут посылать серверу команды. Это второй тип сервисов. Здесь все наоборот. Сервер публикует командный сервис и Callback для обработки команд. Любой клиент может послать команду серверу. Для уведомления, что команда получена сервером, клиент предоставляет Callback процедуру.

Роль процесса (клиент или сервер) относится к сервису, а не к компьютеру. То есть один и тот же процесс может быть сервером, публикующим данные в виде набора сервисов, и в то же время клиентом, который подписывается на получение других сервисов. Это позволяет легко организовать квитирование: сервер публикует данные в сервисе A, а подтверждение получает в виде клиентской команды или другого сервиса B, на который он подписывается.

Приятным дополнением является то, что в DIM входят утилиты типа DIMTree, которые позволяют наблюдать содержимое сервисов в реальном времени, не написав ни строчки кода, что очень облегчает отладку.

Особенностью программирования под DIM является то, что там надо аккуратно вести себя в смысле потоковой безопасности, так как Callback вызов может происходить в другом потоке. А вообще система эта хорошо сделана в смысле реального времени - обмен данными идет в отдельном потоке, приоритет которого регулируется. В этом смысле используемые в Delphi сокеты намного хуже. Дело в том, что сокеты в формах VCL работают не с потоковой моделью, а с сообщениями Windows, которые обрабатываются в основном потоке программы, вперемешку с рисованием, сохранением в файлы и т.д. Представьте себе, что основной поток занялся длительным рисованием кнопочек и списочков, сохранением файлов и т.д. - очередь сообщений будет ждать, пока операция рисования или сохранения не завершится. Не спасает и то, что VCL сокеты тоже используют потоки - они ведь вызывают Synchronize для выполнения вызова событий (например, OnAccept, OnDataSend и т.д.). А это значит, управление передается опять основному потоку и если он занят - будет блокировка. Куда правильнее делать ввод-вывод в отдельном высокоприоритетном потоке, который может вытеснять второстепенные операции, если ему это надо.

Мы работаем с DIM уже несколько лет и очень довольны. Попробуйте. Мне кажется, эту технологию вполне можно взять за основу всей системы. Программировать будет не в пример проще, чем на чистых сокетах.