Technical Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| FHWA/TX-04/0-1752-11 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| CENTER-TO-CENTER COMMUNICATIONS IN LOW-BANDWIDTH ENVIRONMENTS | October 2003 |
| | 6. Performing Organization Code |

| 7. Author(s) | 8. Performing Organization Report No. |
|---|---|
| Robert E. Brydia, Steve C. Liu, and Kevin N. Balke | Report 0-1752-11 |

| 9. Performing Organization Name and Address | 10. Work Unit No. (TRAIS) |
|---|---|
| Texas Transportation Institute<br>The Texas A&M University System<br>College Station, Texas 77843-3135 | 11. Contract or Grant No.<br>Project No. 0-1752 |

| 12. Sponsoring Agency Name and Address | 13. Type of Report and Period Covered |
|---|---|
| Texas Department of Transportation<br>Research and Technology Implementation Office<br>P. O. Box 5080<br>Austin, Texas 78763-5080 | Technical Report:<br>September 2001 – August 2003 |
| | 14. Sponsoring Agency Code |

15. Supplementary Notes

Project performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration.

Project Title: TransLink® Research Program

16. Abstract

Center-to-center (C2C) communications provides the capability of exchanging status and control information across centers with disparate systems. Built upon national standards, the Texas implementations provide a C2C infrastructure which allows centers to provide command and control functions to other centers. Participating centers can extract this information, forming a statewide traffic management capability.

This report summarizes the research performed to examine the impacts of using the infrastructure in a low-bandwidth environment. Multiple testing scenarios were constructed and examined to determine what, if any, negative impacts would result from center connections over modems.

The results clearly show that the amount of data involved in the exchange of most C2C information is not sufficient to stress even low-bandwidth connections. The caveat is the exchange of closed circuit TV (CCTV) snapshots, which may take 2-20 seconds, depending on a number of factors, including size of the snapshot and the modem connection speed.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Center-to-Center Communications, C2C, TMDD, Low-bandwidth, TxDOT | No restrictions. This document is available to the public through NTIS:<br>National Technical Information Service<br>Springfield, Virginia 22161<br>http://www.ntis.gov |

| 19. Security Classif.(of this report) | 20. Security Classif.(of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 46 | |

# CENTER-TO-CENTER COMMUNICATIONS IN LOW-BANDWIDTH ENVIRONMENTS

by

Robert E. Brydia
Associate Research Scientist
Texas Transportation Institute

and

Steve C. Liu, Ph.D.
Associate Research Scientist
Texas Transportation Institute

Kevin N. Balke, Ph.D., P.E.
TransLink® Center Director
Texas Transportation Institute

Report 0-1752-11
Project Number 0-1752
Project Title: TransLink® Research Program

Performed in cooperation with the
Texas Department of Transportation
and the
Federal Highway Administration

October 2003

TEXAS TRANSPORTATION INSTITUTE
The Texas A&M University System
College Station, Texas 77843-3135

# DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official view or policies of the Federal Highway Administration (FHWA) or the Texas Department of Transportation (TxDOT). This report does not constitute a standard, specification, or regulation. The engineer in charge was Kevin N. Balke, P.E. (Texas, #66529).

# ACKNOWLEDGMENTS

This project was conducted as part of the TransLink® research program and was performed in corporation with the Texas Department of Transportation and the Federal Highway Administration. The project team recognizes the following TransLink® partners for their generous support of the TransLink® research program:

- U.S. Department of Transportation, Federal Highway Administration;
- Texas Department of Transportation;
- Metropolitan Transit Authority of Harris County;
- Texas Transportation Institute; and
- Rockwell International.

The project team would also like to recognize the following individuals for their support of this specific project:

- David Gibson, Federal Highway Administration;
- Mark Olson, Federal Highway Administration;
- Al Kosik, Traffic Operations Division, Texas Department of Transportation;
- Richard Reeves, Traffic Operations Division, Texas Department of Transportation;
- Sally Wegmann, Houston District, Texas Department of Transportation;
- Terry Sams, Dallas District, Texas Department of Transportation;
- Wallace Ewell, Fort Worth District, Texas Department of Transportation; and
- Pat Irwin, San Antonio District, Texas Department of Transportation.

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

**Page**

# LIST OF TABLES

**Page**

# CHAPTER 1:
# INTRODUCTION

## OVERVIEW

Within the realm of traffic operations, traffic management centers (TMCs) have been developed as a central location for the collection of data and control of field devices. A TMC typically performs one or more of the following functions:

- operates traffic control devices,
- monitors traffic conditions,
- responds to incidents, and
- publishes traffic related information (*1*).

As part of performing these functions efficiently, TMCs need to share information with one another. One example might be an incident that affects a significant portion of the roadway network. In this situation, telling neighboring TMCs the information may allow them to manage traffic flow leading into the affected area and may allow drivers to take alternate routes to avoid the area altogether. In another situation, a TMC may want to give a neighboring agency control over a specific piece of equipment, such as a camera or a dynamic message sign (DMS).

Depending on the situation, other types of management centers may also need to share information with traffic management centers. A transit management center might want to share information on the location of the bus fleet while receiving information about roadway status. An emergency operations center might provide information on flood conditions and request control of dynamic message signs to alert drivers to dangerous conditions.

The problem with these information exchanges is that most centers do not use standardized software, hardware, or communication systems. In fact, even within the same region or state, multiple vendors may have been involved with the development of centers, leading to disparate systems that in essence cannot communicate with each other. Even agencies that have multiple TMCs in a region or area may have very different systems inside each center. All of these different systems lead to communication failures between centers—which impacts the efficiency of the roadway system to recover from incidents.

The concept of center-to-center (C2C) communications bridges this gap and provides a common method for exchanging information between centers. C2C also provides the capability to exchange control information, providing external centers with a mechanism for controlling infrastructure outside of their own devices.

The development of C2C allows each center to continue to use their proprietary systems, yet participate in a loosely connected larger structure as needs dictate. C2C removes the need to redesign and rebuild each individual system to achieve shared communications. In essence, C2C gives a center the capability of independent communications on internal systems and common communications to external systems (*2*, *3*, *4*).

## WHAT IS C2C?

Looking at the big picture, C2C is a set of functions that describe a method of communicating standard information and functions across disparate systems. At a more technical level, C2C is a set of protocols, standard message sets, and data elements that are used to send information and control between centers. Figure 1 identifies the components of the C2C specification.

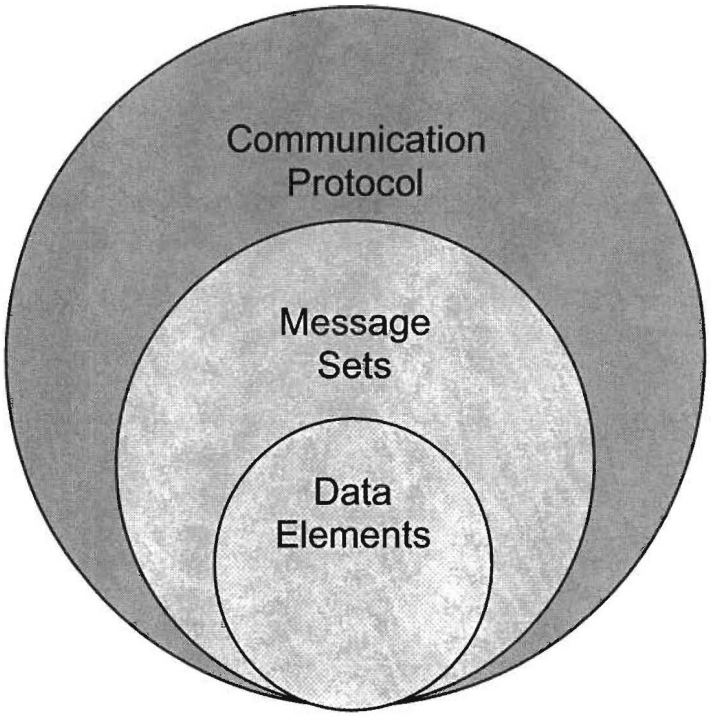


**Figure 1. Components of Center-to-Center Specification.**

At the top-most level (refer to the outermost circle in Figure 1), C2C uses a protocol to send or receive information between locations. A protocol is a "specific set of rules, procedures, or conventions related to the format and timing of data transmission between two devices" (*5*, page 680). The C2C specifications were originally published with support for the Data Exchange Abstract Syntax Notation (DATEX) and Common Object Request Broker Architecture (CORBA) protocols. Specific implementations of C2C could use either protocol to send and receive information between centers, although the same protocol would have to be used at each end.

At the bottom-most level (refer to the innermost circle in Figure 1), the C2C specifications use data elements to describe information. An individual data element might be the latitude of a field device. Another individual data element would be the corresponding longitude of the device. By describing the value of each individual data element, a complex and rich set of information can be assembled.

For example, the complete set of information regarding a traffic signal would contain dozens of individual data elements, corresponding to such items as the phasing of the signal, the timing of each phase, the status of each component of the signal, and the physical location of the signal.

The middle circle in Figure 1 represents message sets. Typically, individual data elements are assembled into functional groups. An example might be the functional group corresponding to the status of a lane control signal (LCS). The data elements in this group include:

- LCS identifier,
- LCS name,
- LCS location – latitude,
- LCS location – longitude,
- LCS status,
- LCS geometry – heads,
- LCH head capabilities, and
- LCS current delay settings (*6*).

The complete set of functional groups for the entire range of information that will be sent or received is called the message set.

In the C2C specifications, the message set was derived from the Traffic Management Data Dictionary (TMDD). The TMDD is an effort by industry and the federal government to standardize the data elements that TMCs use to communicate. The name of the specific message set used for the C2C specifications is Message Sets for External Traffic Management Center Communications (MS/ETMCC).

## HOW DOES IT WORK?

Now that the background of why C2C was created is understood, a brief explanation of how it works is warranted. Figure 2 shows a typical center, of any type. Within the center, proprietary software is often utilized to accomplish the monitoring and control functions the center performs. This software may be a computer-aided dispatch system, a traffic monitoring application, a transit routing package, or any other software that a center would use. Regardless of the software, defined data elements exist that are populated with specific pieces of information.

In order to construct a C2C interface, those data elements from the center software are mapped into the data elements within the C2C specification. For example, the center software may call the longitudinal location of a DMS something like "DMS-loc-long." The corresponding data element in the C2C specification may be "LocationLongitude." A series of mappings are created that essentially say the information in the two data elements is equal.

$$DMS - loc - long \Leftrightarrow LocationLongitude$$

Once the two elements are equated, the C2C specification groups all like objects into a message set. The values in the entire message set are then sent out of the center using the appropriate

protocol. It is important to realize that in the current specification, only the VALUE of the data elements is sent, not the IDENTITY that corresponds to the value.

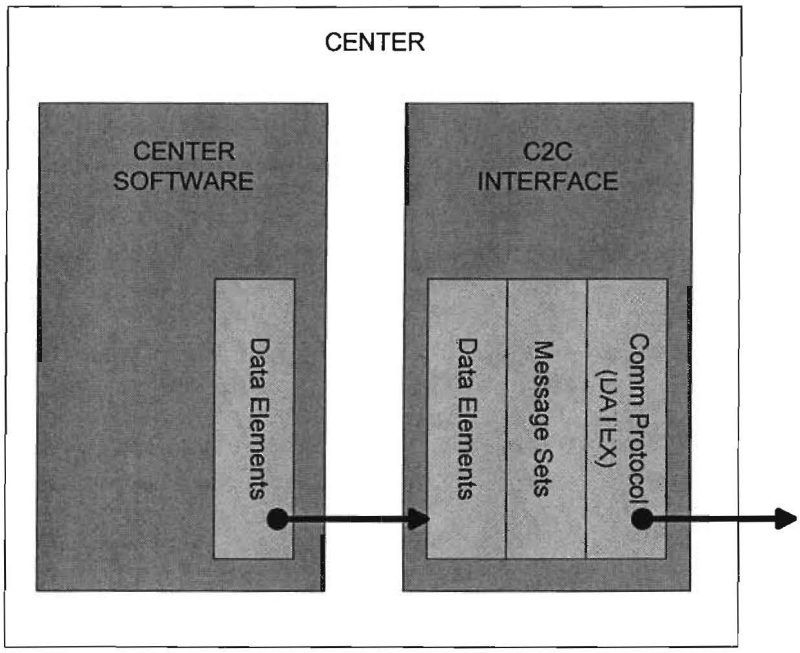


**Figure 2. How Center-to-Center Communications Works.**

In other words, for the C2C specification to work properly, both the send and receive interfaces must know exactly which message set of elements is being transmitted, because only the values will be sent. A null value will be sent for data elements that do not have a value.

At the receiving center, the process is reversed. The string of information is received by the protocol, the message set of elements is identified, and data elements are populated with values and then mapped into the specific data elements in use at that particular center. This allows each C2C interface to be customized for each center and interface to any external software platform.

## TXDOT C2C IMPLEMENTATIONS

### Current

TxDOT has pursued the implementation of C2C between TMCs within Texas. Implementations to date provide both a status and command component. Status allows a remote user to receive information about the status of the roadway network or any particular device, as long as they have the authentication credentials necessary for that task. Command components allow a user within one TMC to control equipment belonging to another TMC, if the privileges are set up to allow that interaction. The C2C interfaces at either TMC make these communications possible.

Currently, both the Dallas and Forth Worth TMCs have a C2C interface and can share status and command information.

The initial TxDOT implementation had three goals:

- to provide freeway conditions data on a graphical map between control centers and for an Intelligent Transportation Systems (ITS) Internet site,
- to extend the data server capabilities beyond freeway conditions, and
- to include additional development and integration activities (*7*).

Through the process of creating the C2C interfaces for Dallas and Forth Worth, several lessons learned have been documented by TxDOT:

- While the data elements in the national standards were mostly sufficient, the message sets (functional groups) were not. Extensive supplementation of the message sets took place to support the status and command interfaces.
- Data repositories can be created for ITS data using C2C.
- Data repositories can be hierarchical, allowing for multiple levels, e.g., local, regional, and statewide.
- User authentication information using DATEX is not secure.
- Implementation using the DATEX protocol is problematic due to lack of vendor support.
- Future implementations will likely change the communication protocol to a more widely used and supported system.
- Integration of dissimilar systems can be achieved; however, careful attention must be paid to basic system definitions and network configurations.
- Agencies without Advanced Transportation Management Systems (ATMS) can participate in the C2C infrastructure by utilizing a lightweight C2C graphical user interface (GUI) specifically designed to provide status and command information.
- Institutional issues, especially with regard to sharing control of equipment, must be addressed (*7*).

**Future**

Researchers anticipate that the future will see all TMCs in Texas utilizing C2C—in essence creating a traffic management information and control network across the entire state.

The advantage of this statewide network is that all of the information from various centers can be centralized—in effect, creating a single view of traffic into the state. This could be of enormous benefit to the tourist and trucking industries. Additionally, because the cost of building TMCs is significant, the potential to reuse the interfaces and establish a common infrastructure could accrue significant cost savings to future build-outs. Furthermore, areas without a TMC or extensive ATMS capabilities could still participate in a statewide C2C infrastructure. Finally, Texas enjoys a unique opportunity to guide future standards development as a result of the implementations to date and the work that will continue in the future.

## TRANSLINK® C2C EFFORTS

The amount of information exchanged during C2C communication can be significant. One example is when a TMC first signs on and authenticates to another TMC. During the exchange, the full database of accessible equipment is sent. In a large urban area, this listing can be extensive and take significant time and bandwidth.

Another example is the transmission of CCTV snapshots, which are used to provide feedback to the command functions of pan, tilt and zoom. Additionally, because any communications protocol incurs overhead above and beyond the information, the actual amount of C2C data sent can be quite significant.

Because the initial C2C connection (Dallas ←→ Forth Worth) was implemented over fiber, bandwidth was not a concern. However, because future C2C architecture calls for smaller centers and other areas to all connect in a statewide C2C infrastructure, bandwidth may be a concern for many areas. Additionally, there were questions as to the growth capability of these smaller areas using low-bandwidth communications, as the information being exchanged with C2C continues to be supplemented and expanded.

The TransLink® task, therefore, focused on looking at C2C communications in a low-bandwidth environment and performing an assessment of the connection capabilities of areas communicating over phone lines. Specifically, the work effort asked the following questions:

- Can C2C be implemented in a low-bandwidth environment such as a phone line?
- Can this method of information transfer be effective?
- What are the pros and cons of sending C2C in a low-bandwidth environment?
- Can areas using low-bandwidth communications "keep up" as the information being exchanged with C2C is expanded and supplemented?
- Can C2C be used to support large-scale integration efforts, such as the College Station Integration Project?

# CHAPTER 2:
# LOW-BANDWIDTH COMMUNICATIONS TESTING

To assess the capability of using C2C in a low-bandwidth environment, and answer the questions presented in Chapter 1 of this report, TransLink® obtained the most recent version of the C2C software from TxDOT. This was version 2.1.2 for the C2C Infrastructure files and 2.2.1 for the Status and Incident GUI. (*Note that prior to the completion of this task, TxDOT released version 3.x of the C2C infrastructure files).*

The distribution CD included several different C2C software components, each of which is briefly described below. Each of these software components was configured and utilized to construct a C2C infrastructure for different testing situations:

- *Data Provider* – Takes data native to a TMC format, converts it to TMDD, and makes it available in a DATEX data stream.
- *Data Collector* – Collects data from data providers and sends it to other applications using a DATEX data stream.
- *Data Extractor* – Receives information via a DATEX data stream and converts it into the format required for the receiving TMC.
- *Status Interface Test Server* – Provides test data for the C2C infrastructure in the absence of field devices sending live data.
- *Status Interface Test Client* – Receives data from the data extractor and provides confirmation of data transfer across the C2C cloud.
- *Process Status Viewer* – A program that manages the operation of the C2C software components (*3*).

## TEST CONFIGURATION 1 – ALL C2C COMPONENTS ON THE SAME MACHINE

The first test configuration utilized all the C2C components on the same machine. This configuration did not test low-bandwidth communications; rather, it was used to establish baseline knowledge of the C2C software components, their capabilities and configuration, their relation to each other, and the overall process of establishing a C2C infrastructure.

Figure 3 shows the testing configuration. Testers installed all software components on the same machine. The test utilized the status interface test server to simulate data available from field devices. The data provider takes the field test data and injects it into the C2C cloud via the data collector. The data extractor receives data from the cloud via the data collector. Finally, the status interface test client is used to retrieve the field data information on the other side of the cloud (aka C2C infrastructure).

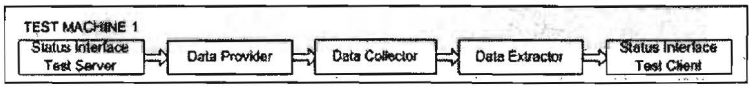


**Figure 3. All C2C Components on Same Machine.**

In order to establish a C2C cloud, many of the software components have specific parameters that must be configured before the components will talk to the next process in the line.

### The Status Interface Test Server Configuration

The status interface test sever does not have any configuration files. System parameters such as server port number and initial test data location are hard coded inside the executable file. When it is started, it loads data from the directory ".\Data\Field Tests" and listens for client connection requests on port 8152. Additional test data can be loaded by entering a directory name in the "data directory" field and pushing the "load data" button.

### The Data Provider Configuration

The data provider has two configuration files. The "SysParams.dat" file specifies the connectivity parameters to connect to the other software component. Figure 4 shows a portion of the configuration file that contains the header information. This format is common across all of the software components. Line numbers have been added to the figure to help reference the location of critical information. The actual files do not contain the line numbers.

```
1  ##############################################################################
2  # Copyright, 1999-2001, Texas Department of Transportation; (512) 416-2000
3  #
4  # The copyright to the computer program(s) and source code herein is the
5  # property of the Texas Department of Transportation. The program(s) and
6  # source code may be used and/or copied only with the written permission of
7  # the Texas Department of Transportation or in accordance with the terms and
8  # conditions stipulated in the agreement/contract under which the program(s)
9  # and source code have been supplied.
10 #==============================================================================
11 #      Source File:      SysParams.dat
12 #
13 #      Author: K.S. Honeyager
14 #
15 #      Abstract:
16 #            File includes the Data Provider system
17 #            configuration parameters.
18 #
19 ##############################################################################
```

**Figure 4. Header Format Common to C2C Software Component Configuration Files.**

Figure 5 shows the main portion of the "SysParams.dat" configuration file. Line numbers were added to the figure to illustrate the location of critical information. The important parameters for this file include:

- the IP address and port number of the *process viewer* (lines 29 through 31),
- the IP address and port number for the *status logger* (lines 36 through 37), and
- date source location (lines 42 through 44). In this configuration, the data source is the status interface test server. The port number 8152 is hard coded in the status interface test server.

```
21 # The process name displayed in the dialog box and log messages.
22 #
23 SysParams.ProcessName                    Data Provider
24
25 # The IP address, port number and heartbeat interval (in ms) for
26 # the process status viewer.
27 # Note: Use 127.0.0.1 for IP loopback.
28 #
29 SysParams.ProcessViewerAddress                          127.0.0.1
30 SysParams.ProcessViewerPort                            8001
31 SysParams.ProcessViewerHeartbeatInterval               8500
32
33 # The IP address and port number for the process status logger.
34 # Note: Use 127.0.0.1 for IP loopback.
35 #
36 SysParams.StatusLoggerAddress   127.0.0.1
37 SysParams.StatusLoggerPort      8000
38
39 # The Name, IP address, and port number for TMC Server or plug-in.
40 #
41
42 SysParams.ServerNames           Status Interface Test Server
43 SysParams.ServerIpAddresses     127.0.0.1
44 SysParams.ServerPorts           8152
45
46 SysParams.ServerConnectInterval 20000
47
48 SysParams.SubscribeRoadwayNetwork    1
49 SysParams.SubscribeIncidents          1
```

**Figure 5. The Data Provider "SysParams.dat" File.**

The second configuration file with the data provider is "DATEXASN.ini." As shown in Figure 6, this file configures the port number (line number 1) through which the data provider will send DATEX format data for other system components. The file also configures additional parameters associated with the DATEX protocol. Line numbers have been added to the figure to illustrate the location of critical information.

```
1. DATEXPort = 8201
2. PacketSize = 576
3. CircularFileSize = 4567
4. MaxHeartbeatDuration = 360
5. ResponseTimeout = 60
6. Presentation = BER
```

**Figure 6. The "DATEXASN.ini" File.**

### The Data Collector Configuration

Like the data provider, the data collector also has two configuration files. The filenames are common to both components. Overlap of the files is not an issue since the components are installed in different directories.

The critical difference in the configuration file is located on lines 44 through 46 of Figure 7. Line numbering has been added to the figure to aid in locating the critical information. For the data collector, the file points to the downstream data provider, which is identified by an IP address and port. The port number for the data source is defined by the *DATEXPort* variable in the data provider's "DATEXASN.ini" file. This configuration can be seen on line 1 of Figure 6.

Figure 7 also illustrates the configuration of the data collector when more than one data provider is utilized. In this case, multiple providers are listed, using the name, IP address, and port numbers. Lines 40 through 42 of Figure 7 show the parameters for this situation. The "#" sign at the beginning of the line indicates that these lines are comments and will be ignored when program execution begins. The use of the "#" sign allows for rapid reconfiguration of the testing sequences. As currently configured, the file in Figure 7 will communicate with one data provider.

```
20. # The process name displayed in the dialog box and log messages.
21. #
22. SysParams.ProcessName              Data Collector
23.
24. # The IP address, port number and heartbeat interval (in ms) for
25. # the process status viewer.
26. # Note: Use 127.0.0.1 for IP loopback.
27. #
28. SysParams.ProcessViewerAddress                    127.0.0.1
29. SysParams.ProcessViewerPort                      8001
30. SysParams.ProcessViewerHeartbeatInterval 8500
31.
32. # The IP address and port number for the process status logger.
33. # Note: Use 127.0.0.1 for IP loopback.
34. #
35. SysParams.StatusLoggerAddress  127.0.0.1
36. SysParams.StatusLoggerPort     8000
37.
38. # The Name, IP address, and port number for Regional Data Server(s).
39. #
40. #SysParams.ServerNames                 Data Provider 01, Data Provider 02
41. #SysParams.ServerIpAddresses          127.0.0.1,127.0.0.1
42. #SysParams.ServerPorts                8201,8202
43.
44. SysParams.ServerNames                 Data Provider
45. SysParams.ServerIpAddresses           127.0.0.1
46. SysParams.ServerPorts                 8201
```

**Figure 7. The Data Collector "SysParams.dat" File.**

### The Data Extractor Configuration

Like the data provider and data collector, the data extractor also has two configuration files. As with the previous components, the filenames are common, and overlap is avoided by installation into different directories.

Figure 8 shows the "SysParams.dat" file. The port number defined on line 42 tells the data extractor itself where to find the data source. In this case, the data source is the downstream data collector. Connections are made to that component's DATEX port that is defined by the *DATEXPort* variable in the data collector's "DATEXASN.ini" file.

```
20. # The process name displayed in the dialog box and log messages.
21. #
22. SysParams.ProcessName              Web Data Extractor
23.
24. # The IP address, port number and heartbeat interval (in ms) for
25. # the process status viewer.
26. # Note: Use 127.0.0.1 for IP loopback.
27. #
28. SysParams.ProcessViewerAddress                    127.0.0.1
29. SysParams.ProcessViewerPort                      8001
30. SysParams.ProcessViewerHeartbeatInterval         8500
31.
32. # The IP address and port number for the process status logger.
33. # Note: Use 127.0.0.1 for IP loopback.
34. #
35. SysParams.StatusLoggerAddress          127.0.0.1
36. SysParams.StatusLoggerPort            8000
37.
38. # The Name, IP address, and port number for the Data Collector or Data Provider.
39. #
40. SysParams.ServerNames                  Data Collector
41. SysParams.ServerIpAddresses           127.0.0.1
42. SysParams.ServerPorts                 8300
43. SysParams.ServerConnectInterval       10000
44.
45. # The port number for the web server.
46. #
47. SysParams.ClientPort     8400
```

**Figure 8. The Data Extractor "SysParams.dat" File.**

In the same fashion as the data collectors and providers, the data extractor provides data via a DATEX port defined in the "DATEXASN.ini" file. The data extractor also publishes data to a data port (configured on line 47 of Figure 8), for any additional applications deployed upon the data extractor.

### The Status Interface Test Client Configuration

Similar to the status interface test server, the status interface test client does not have any configuration files. System parameters are input from the GUI of the client.

As seen in Figure 9, the configuration information requires an IP address and a port number. The IP address is the IP address of the machine on which the data extractor is running. The port number is defined by the SysParams.ClientPort variable in the data extractor's configuration file "SysParams.dat" (in this case, it is 8400, which can be seen in Figure 8, line 47).

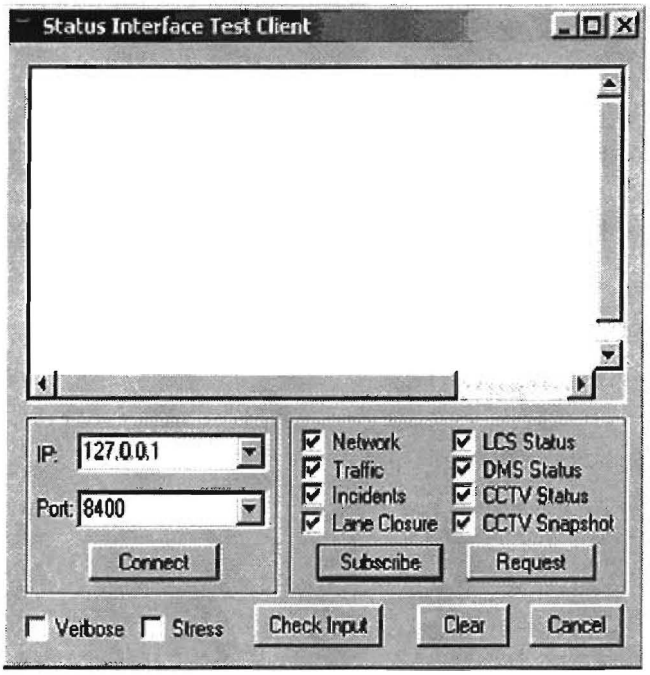


**Figure 9. The Status Interface Test Client GUI.**

### Initiate C2C Infrastructure

The process status viewer application is used to initiate the operations of the C2C software components. Each component is added to the process status viewer. All programs can be started using the "File, Start All Processes" command sequence available in the program's GUI. Figure 10 shows the process status viewer for test configuration 1. At the time of this screen capture, the status interface test client application had not been started. The process status viewer recognizes this and provides both textual and visual feedback that says the application is not responding. Starting the application removes the "error" condition, which was used to demonstrate the feedback capabilities of the process status viewer GUI.

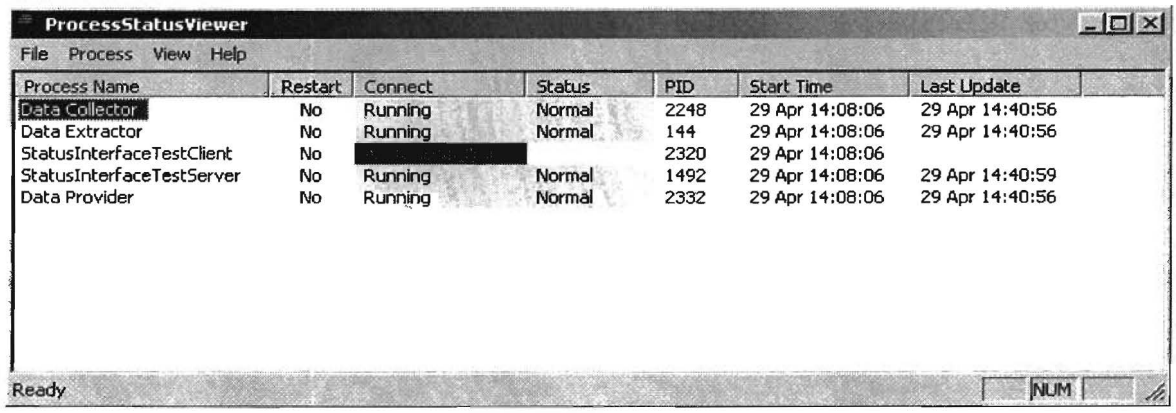


**Figure 10. Process Status Viewer.**

## C2C Software Component Response

Figure 11 shows the status interface test server GUI, which is the data source for test configuration 1. The GUI indicates that test data have been sent to the upstream client (data provider). The GUI also indicates the location of the data as well as the status of parameters that can be used to effect changes to the data stream.

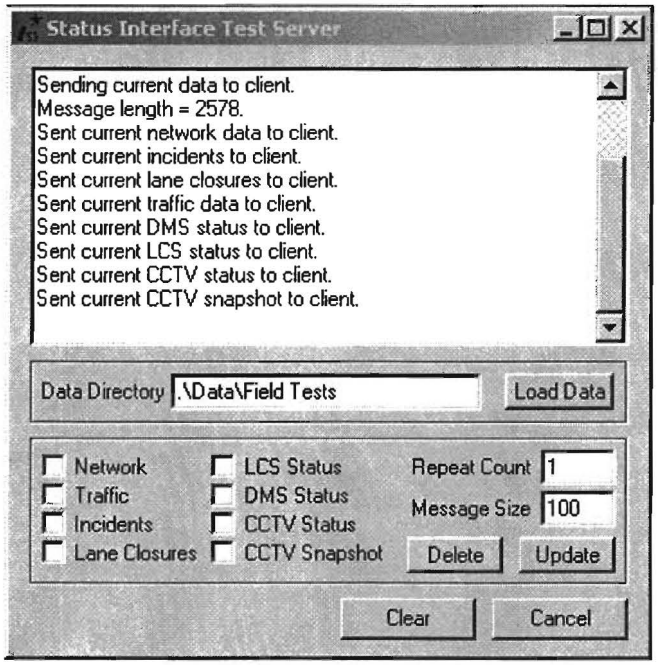


**Figure 11. Status Interface Test Server GUI.**

Figures 12 through 16 show the communications taking place as the C2C cloud is initiated. The data provider accepts a connection from the data collector in Figure 12. In Figure 13, the data collector accepts a connection from the data extractor. Figure 14 shows the data extractor connecting to the data collector, as well as opening the supplementary port for publication of the

data it receives. Figure 14 shows this port operating on port number 8400, as defined in line 47 of the configuration file in Figure 8.

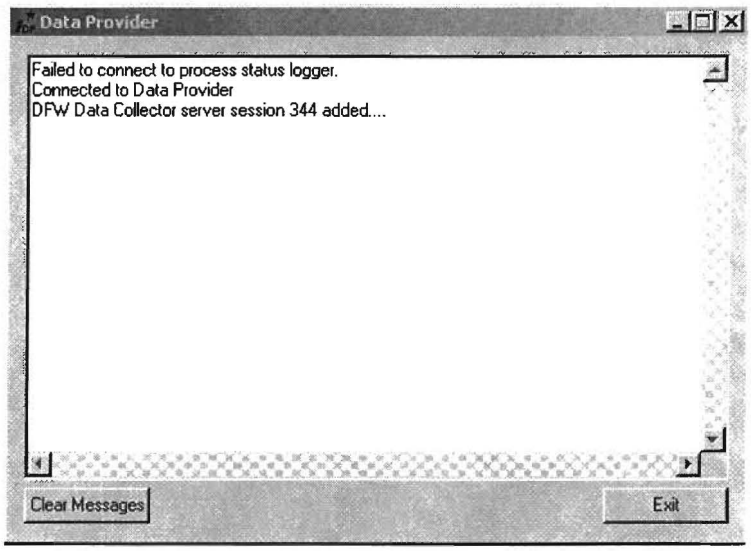


**Figure 12. Data Provider GUI.**

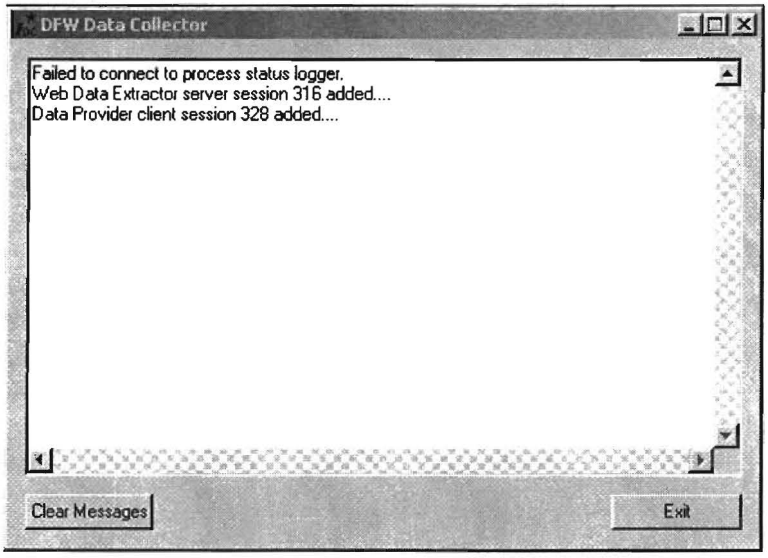


**Figure 13. Data Collector GUI.**

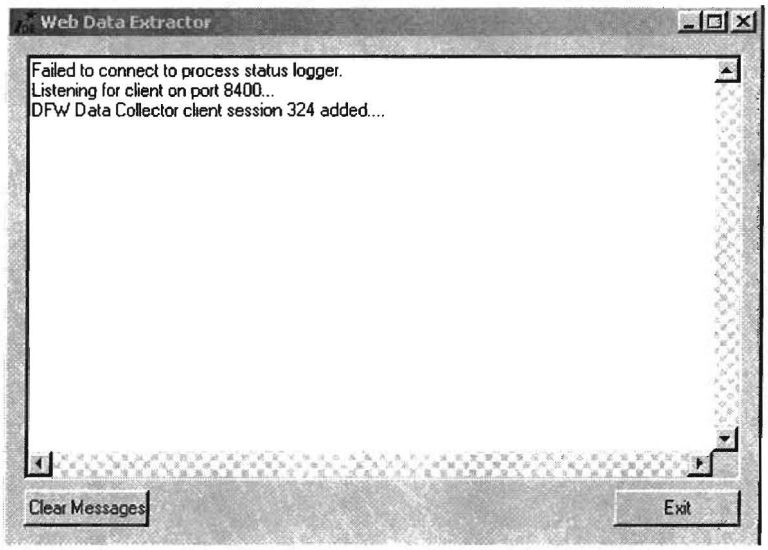


**Figure 14. Data Extractor GUI.**

Figure 15 shows the status interface test client GUI. The output flowing across the GUI indicates that the data initiated on the other side of the C2C cloud are being received. Data flow from the status interface test server to the data provider, to the data collector, to the data extractor, and to the status interface test client. The GUI has configuration options for the IP address and port number. In this test configuration, they are 127.0.0.1 and 8400, respectively. The checkboxes allow a subscriber to receive specific types of information, such as incident or DMS status. The checkboxes in Figure 15 show that the client is currently subscribed to receive all available information.

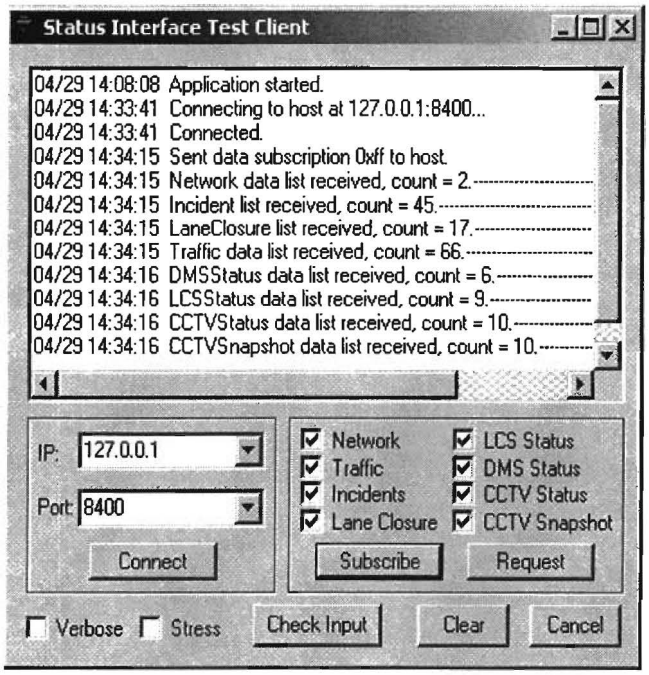


**Figure 15. Status Interface Test Client GUI.**

### Results of Test Configuration 1

The architecture of the first test configuration was not complicated. However, the goals of this configuration were not to test low-bandwidth communications, but to establish familiarity with the C2C components, understand the configuration process, and establish baseline connectivity. As a result of this configuration exercise, researchers achieved the following:

- an overall understanding of the software components in the TxDOT C2C implementation and their specific function in the C2C cloud,
- an understanding of the capabilities of each C2C component and how each component is configured,
- an understanding of how to inject data into the C2C cloud and how the data progresses through the cloud,
- an understanding of how to control the software components and provide a status check on cloud operations,
- confirmation of baseline C2C software component communication, and
- hands-on experience with all of the above.

## C2C ARCHITECTURE FOR ADDITIONAL TESTING

After successfully completing the initial test configuration, the researchers moved to a more complicated C2C architecture to utilize in additional test configurations. As shown in Figure 16, the architecture utilized multiple C2C components spread across two computers.

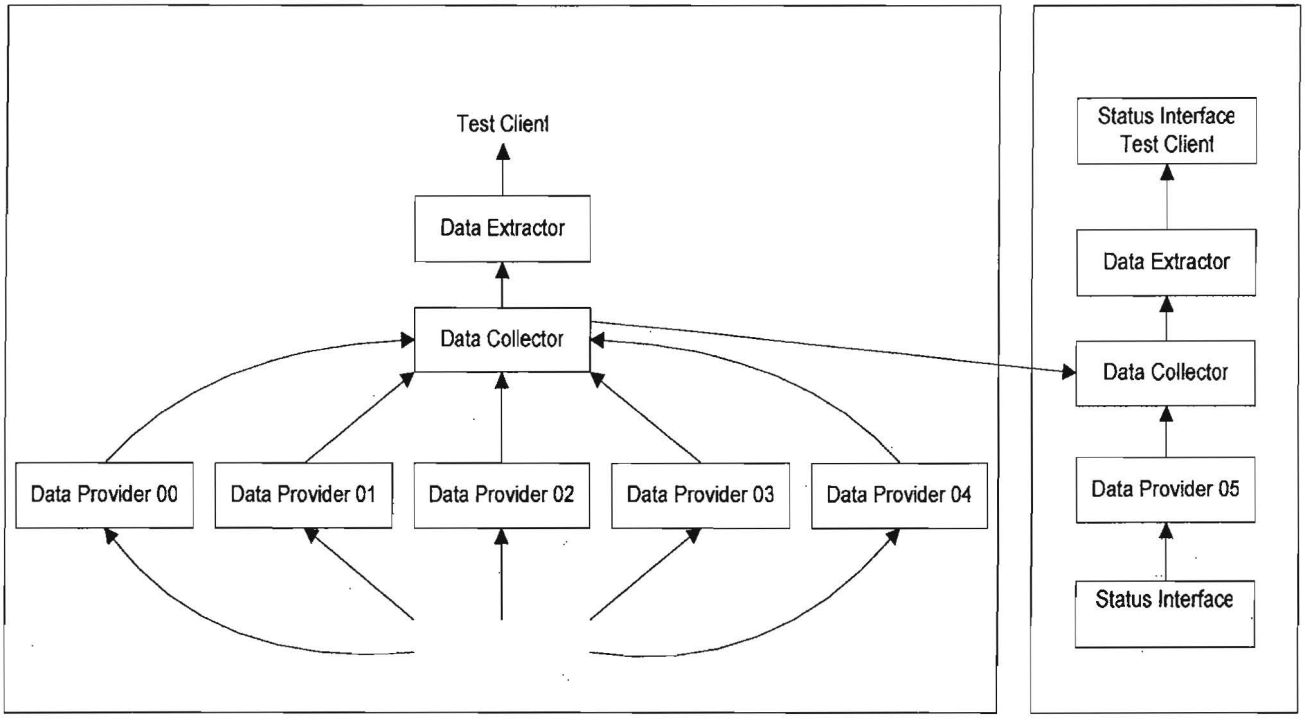


**Figure 16. C2C Software Component Architecture for Additional Testing.**

Several aspects of this architecture are important to notice. First, the use of multiple machines simulates the real-world environment of multiple TMCs injecting and extracting data into a C2C

cloud. Second, multiple data providers are utilized so that more data flow into the cloud. Third, the architecture shows that the data collectors can receive information from multiple providers and/or collectors simultaneously. Finally, the status interface test servers and clients are used to inject and extract test data from the cloud, once it is established.

### Component Configuration

Test machine 1 utilized five data providers. On an initial installation, only a single copy of a data provider can be installed. Additionally, maintenance installations only allow for repairing or removing the data provider or installing different components. Therefore, additional data provider installations were created manually. A new working directory was established for each instance of data provider. The same files were copied to each working directory. Figure 17 shows the data provider directory structure on test machine 1.

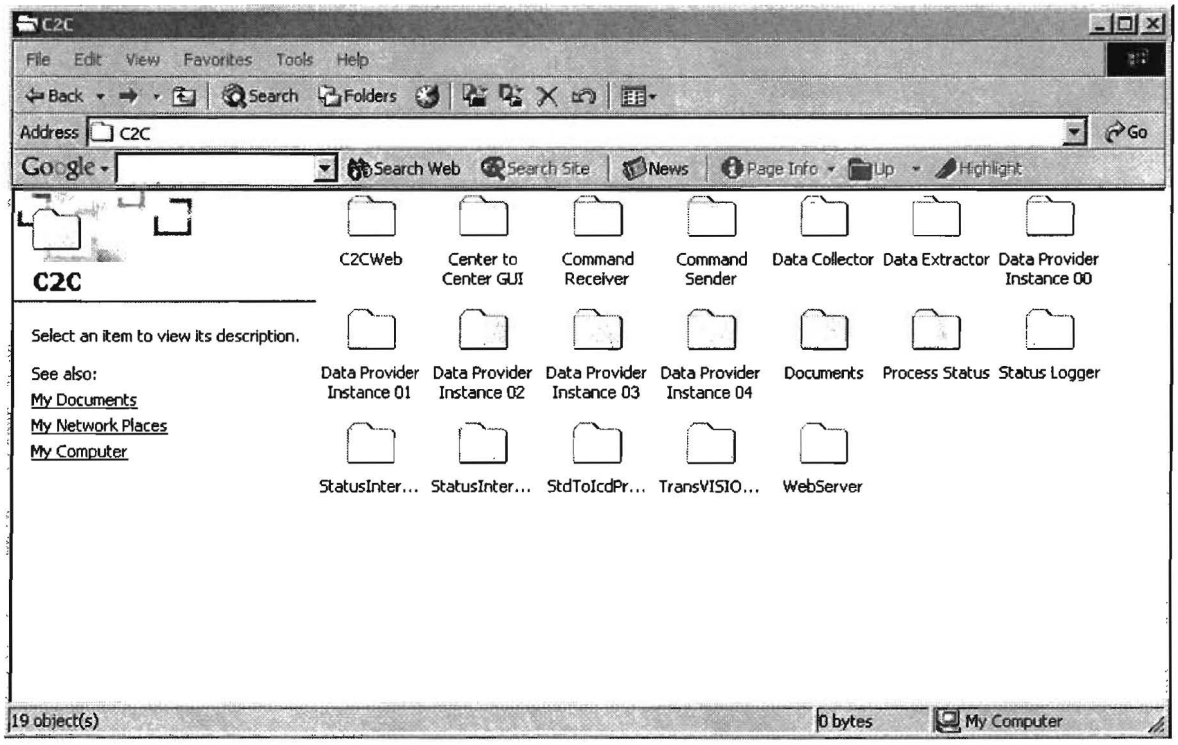


**Figure 17. Data Provider Directory Structure on Test Machine 1.**

The two configuration files for each data provider, "DatexASN.ini" and "SysParamas.dat," were configured according to the examples discussed previously. Each data provider utilized the same IP address, since they are on the same machine, but each utilized a different port number. Ports 8200, 8201, 8202, 8203, and 8204 were used for data provider 00 to data provider 04, respectively. Since each data provider is using information from the same status interface test server, that portion of the configuration file was common across all data providers.

The data collector received information from each provider and passed it along to the data collector on test machine 2, as well as to the data extractor on test machine 1. The link between test machines 1 and 2 is the target area for examining the impact of low-bandwidth

communications. The data collector was configured to listen to all data providers. This configuration was discussed earlier and referenced in Figure 7.

Each of the other components utilized in the testing was configured as previously described in this chapter.

Similar to the initial configuration, test machine 2 utilized a single data provider. That data provider fed into a data collection, which also received information from test machine 1. All of this information was then sent to the data extractor. Test machine 2 also had additional C2C software components, as illustrated in Figure 16. The configuration of these items was the same as described earlier in this chapter.

## LOW-BANDWIDTH CONNECTION TESTING

The low-bandwidth testing was accomplished using a dial-up modem and a 56 Kb connection to the Texas A&M University dial-up service. As shown in Figure 18, the initial test configuration was accomplished using test machine 1 on a dial-up connection. Test machine 2 was connected to the local area network (LAN) using a broadband connection.

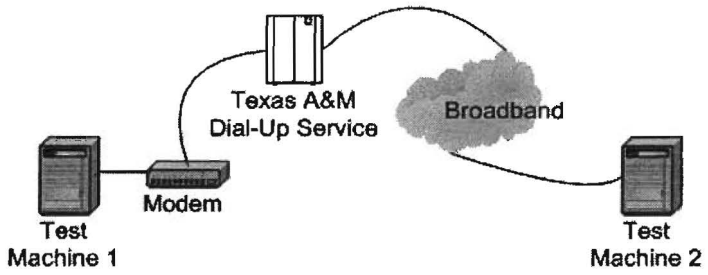


**Figure 18. Test Machine 1 Using Dial-Up Service.**

In the second test sequence, both machines were connected using a dial-up modem. Figure 19 shows this configuration. One item that should be noted is that the Texas A&M dial-up service, like most Internet Service Providers (ISPs) assigns IP addresses dynamically. Each dial-up connection is assigned a different IP connection. In order to accomplish this testing, the machine had to be dialed into the network and the IP address determined. The software configuration files for each component were then modified with the proper addressing information to achieve communications. Each software component then had to be restarted so that the new information would take effect.

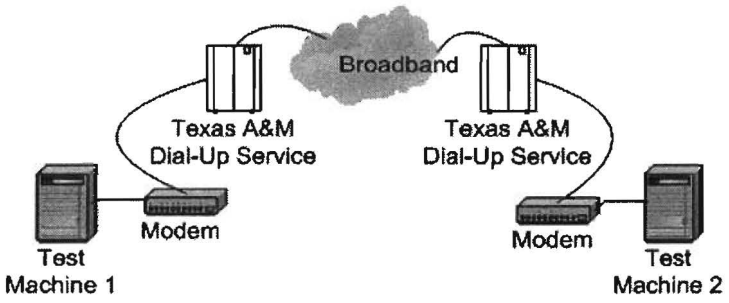


**Figure 19. Both Test Machines Using Dial-Up Service.**

Figures 20 and 21 show snapshots from test machines 1 and 2 that were taken during the second test, where both machines were connected using the dial-up modem service. As can be seen from the figures, data flows are being injected into the system and being extracted from the system. The most important result of the low-bandwidth testing was that at no point in time did the researchers run into bandwidth conditions that limited or stopped the C2C communications. Said in another manner, in testing, the TxDOT C2C implementation functioned well over a low-bandwidth connection.

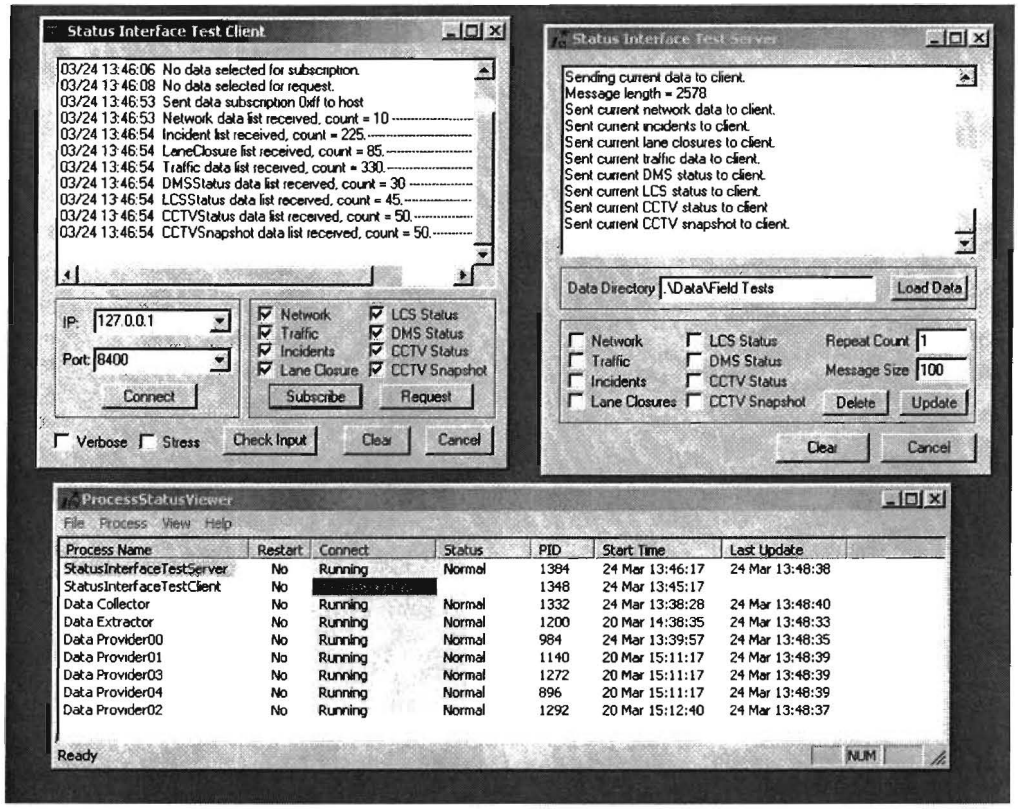


**Figure 20. Snapshot on Test Machine 1 Showing Data Flows.**

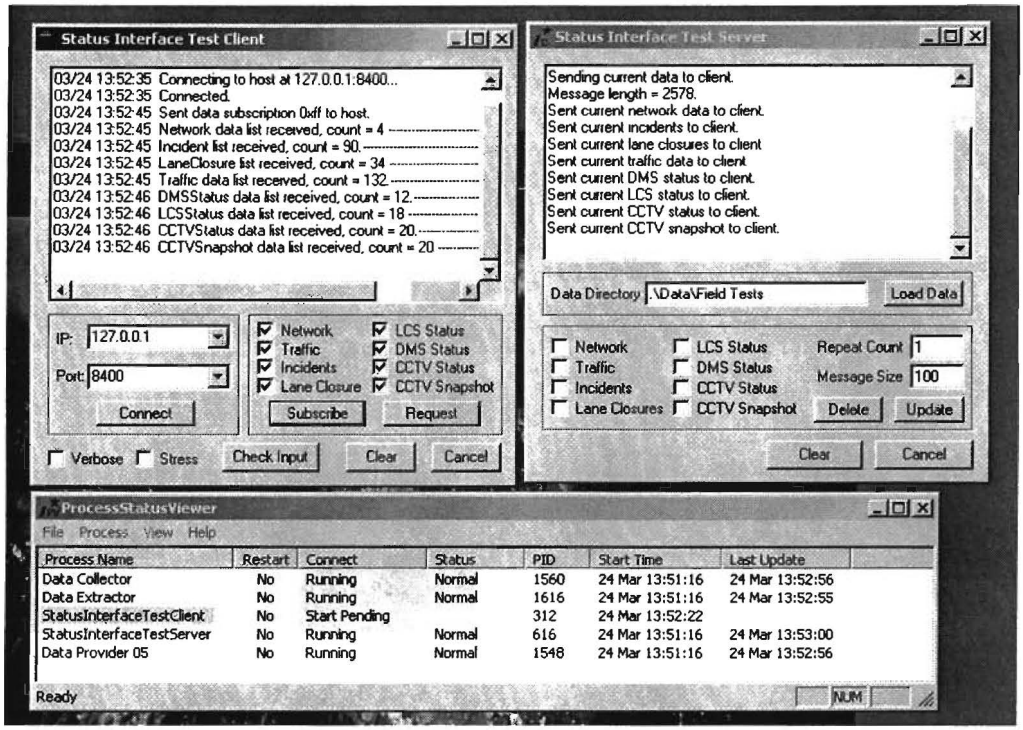


**Figure 21. Snapshot on Test Machine 2 Showing Data Flows.**

## RESULTS OF LOW-BANDWIDTH CONNECTION TESTING

Detailed analysis of transmitting data through a low-bandwidth connection depends largely on specific parameters such as the message length and the modem connection speeds. Typical speeds for modem connections are 33.6 kbps (kilobits per second) or 56 kbps. A 56 kbps connection has an ultimate connection capability of 53 kbps, due to Federal Communications Commission (FCC) regulations. As a result of noise in phone lines and other factors, most dial-up connections typically achieve about 75-85 percent of their maximum transmission capability. Therefore, the transmission speed for a 56 kbps dial-up connection is somewhere between 42 and 48 kbps. Similarly, the transmission speed for a 33.6 kbps connection is 25 to 28 kbps.

Table 1 shows the data sizes in use for message sets within the C2C environment. Excluding network status and CCTV snapshot requests, the average length of most message sets is around 100-150 bytes.

**Table 1. C2C Message Size for Command/Control Functions (in bytes). Developed from (*8*).**

| | Total Size of Data Sent (bytes) | |
|---|---|---|
| Command/Control Section | Request | Request Response |
| Connection | 68 | 38 |
| Network Device Status | 38 | 46 + Additional Data |
| DMS Status | 72 | 1148 |
| DMS Command | 1130 | 74 |
| LCS Status | 72 | 136 |
| LCS Command | 112 | 74 |
| CCTV Status | 72 | 172 |
| CCTV Snapshot | 72 | 114 + Size of Snapshot |
| CCTV "Set Direction" | 106 | 74 |
| CCTV "Set Preset" | 106 | 74 |
| CCTV "Set Absolute" | 116 | 74 |
| CCTV "Set Offset" | 122 | 74 |
| CCTV "Video Switch" | 140 | 74 |
| CCTV "Lock Camera" | 106 | 110 |

At these message sizes and typical transmission speeds, the data transfer is accomplished in well under a second. The largest message size that needs to be transmitted is a DMS message set at 1184 bytes. However, even at this length and typical modem transmission speeds, data transfers are accomplished in well under a second. This calculation is illustrated below:

**DMS Message Size:**
*1148 Bytes*

**33.6 Modem Transmission Speed:**

*(33,600 kilobits per second/8 bits per byte) = 4200 bytes per second*

**Assume 75% efficiency:**
*(4200 x 0.75) = 3150 bytes per second*

**DMS Message Transmission Time:**
*(1148 bytes/3150 bytes per second)= 0.36 seconds*

By far, the longest data transfer times in the C2C environment are seen with CCTV snapshots. The status interface test server loads CCTV snapshots in a JPEG format into the C2C environment. The data load is 114 bytes plus the size of the CCTV snapshot. Available test images were between 5 and 10 kB (kilobytes) in size. At this size, a typical modem connection speed would transfer the image in 2-3 seconds. Table 2 shows anticipated transfer times, in seconds, based on the low-end efficiency (75 percent) of a modem connection.

**Table 2. CCTV Snapshot Transfer Times for Modem Connections (in seconds).**

| | Size of CCTV Snapshot (kB) | | | |
|---|---|---|---|---|
| Modem Speed (kbps) | 10 | 20 | 40 | 60 |
| 33.6 | 3.3 sec | 6.6 sec | 13.1 sec | 19.7 sec |
| 56 | 1.95 sec | 3.9 sec | 7.8 sec | 11.7 sec |

Table 2 represents the longest transfer times, as most modem connections utilizing modern hardware should achieve a higher efficiency rate than 75 percent. In addition, the connection speed of 56 kbps is nearly universal. Therefore, a typical CCTV snapshot should transfer in 2-4 seconds within the C2C environment.

The caveat to this aspect of C2C data transfer is the situation where both the sending and receiving centers are on modem connections. In this special (and perhaps rare) situation, the data transfer times for CCTV snapshots are essentially doubled, as the center must upload to the C2C environment over a modem connection and another center must download over a modem connection. Total time from request to reception could, therefore, be on the order of 4-8 seconds for a typical snapshot. However, this situation is likely to be rare and even this response rate should be adequate for most traffic management tasks.

# CHAPTER 3:
# EXTENSIONS TO THE C2C ENVIRONMENT

With the investigation into the low-bandwidth capabilities of the existing C2C specification complete, attention turned to the next question in the research process. Specifically, can areas using low-bandwidth communications "keep up" as the information in the C2C cloud is expanded? Currently, the message sets of C2C are well defined and, as seen from the results in Chapter 2, impose a relatively light communications load, even when using modem connections. However, as C2C expands and more information is made available in the cloud, the potential for stressing low-bandwidth links increases, since the amount of information will increase and the bandwidth in which to transfer it will not.

To test this aspect of low-bandwidth C2C communications, researchers supplemented the existing infrastructure with additional data elements and message sets. Testing was then performed to examine the impacts of the additional message transfer.

This phase of the research also allowed researchers to investigate another aspect of C2C communications; namely, how easy is it to expand the existing C2C infrastructure and add additional message capability into the system?

## SUPPLEMENTING C2C USING SPECIAL EVENTS INFORMATION

In order to supplement the C2C infrastructure, the researchers decided to add a special events message capability. In order to transport the data across C2C without modifying the internal C2C software components, the special events data were tagged to look like a CCTV snapshot. In essence, the C2C environment was fooled into sending special event information, thinking it was a CCTV snapshot. This allowed the researchers to quickly supplement the C2C environment and also perform the low-bandwidth testing in the same manner as explained in the previous chapter.

The City of College Station, Texas, has an established special event form that contains specific information regarding the event, including event name, location, contact information, date and time, and information regarding provisions for waste, site cleanup, parking, traffic control, and more.

## SPECIAL EVENTS INFORMATION FLOW

The special event data were designed to be attached to the data field of the CCTV snapshot messages in the current C2C environment. This design allows the special event data to smoothly go through the data provider, data collector, and data extractor without modification of those components of the C2C components.

Figure 22 illustrates the following flow of information to transfer special event messages within the C2C environment:

1. When started, special event server listens on local port 8152 and waits for the data provider to connect.

2. Special event client connects to the data extractor.
3. Special event server sends response message that carries the special event data.
4. Special event client sends request message.
5. Special event client receives request message and extracts the special event data.

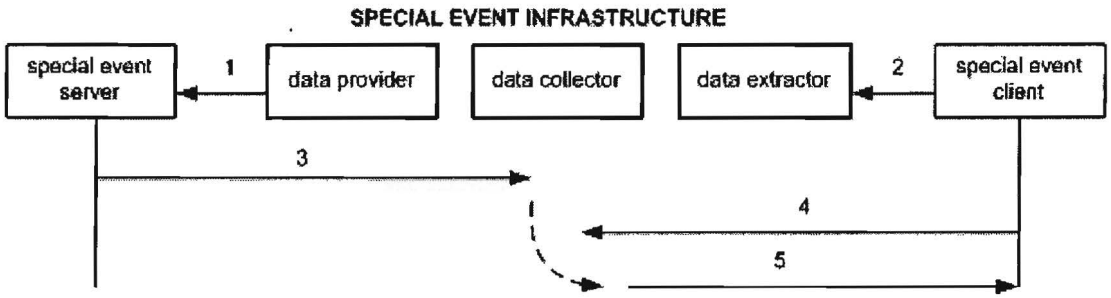


**Figure 22. Special Event Information Flows.**

In essence, the special event client and server replace the status interface test client and status interface test server that were utilized in the earlier testing.

## SPECIAL EVENTS MESSAGE SET

As stated previously, the special events information was added to C2C in such a way as to mimic the requests of CCTV snapshots. This allowed the C2C environment to accept and handle the messages as though they were CCTV snapshots and removed the need to modify the main message transfer software components.

The only difference in the message transmission is that the data field in the CCTV snapshot response message carries the camera status and JPEG snapshot information, whereas an ASCII special event information table was utilized to carry the special event information. Figure 23 depicts the message set header for the information request (data sent from the client to the server).

| Message ID | Count | Data |
|---|---|---|
| 3071h | 0 | N/A |

**Figure 23. Special Events Information Message Request Header.**

The data sent from the server to the client has a message set header formatted as per Figure 24.

| Message ID | Count | Data |
|---|---|---|
| 3071h | # Special Event Data | Table 3 |

**Figure 24. Special Events Information Message Response Header.**

Table 3 illustrates the format of the special events data sent as part of the response from the server to the client.

**Table 3. Special Events Data.**

| Data Item Description | Data Type and Size | Detailed Data Description |
|---|---|---|
| Dummy Data | Integer<br>112 bytes | Placeholder |
| Size of Special Event Data | Integer<br>2 bytes | 0-65534 bytes |
| Individual Special Event Data Elements | String-size specified above | Items are separated by "\n" |

Table 4 illustrates the individual data elements encapsulated in the message set.

**Table 4. Individual Special Events Data Elements.**

| No. | Variable Name | Description | Max-Length /Data Formula |
|---|---|---|---|
| 1 | txtApplicationName | Application Name | |
| 2 | txtDate | Date | mm/dd/yyyy |
| 3 | txtCompanyName | Company Name | |
| 4 | txtAddress | Address | |
| 5 | txtPhone | Phone | (111)-111-1111 |
| 6 | txtCity | City | |
| 7 | txtState | State | |
| 8 | txtZipCode | Zip Code | 11111-1111 |
| 9 | txtTitle | Title of event | |
| 10 | txtLocation | Location | |
| 11 | txtLocationZoning | Location Zoning | |
| 12 | txtTempSigns | Type & location of temporary signs | |
| 13 | rdbtnOutdoors<br>rdbtnTent<br>rdbtnOther | Event to be held | One of (Outdoors, Tent, Other) |
| 14 | txtFormula | Formula for flame proof solution & date solution was applied | (Valid if No.13 is "Tent") |
| 15 | txtEventOther | Description of "Other" in No. 13 | (Valid if No. 13 is "Other") |

**Table 4 (cont). Individual Special Events Data Elements.**

| | | | |
|---|---|---|---|
| 16 | txtProjectedAttendance | Projected Attendance | |
| 17 | txtDataFrom | Effective Dates (From) | mm/dd/yyyy |
| 18 | txtDateThrough | Effective Dates (Through) | mm/dd/yyyy |
| 19 | txtNumberOfDays | Effective Dates (# of Days of operation) | Integer |
| 20 | txtHours | Hours of Operation | |
| 21 | txtProvisonWast | Provisions for waste, human and other | |
| 22 | txtProvisionCleanup | What provisions are being made for site cleanup and grading if necessary | |
| 23 | txtProvisionParking | What provisions are being made for parking (including facility's name and surface composition) | |
| 24 | txtProvisionTraffic | What provisions are being made for traffic (if required) | |
| 25 | txtProvisionNoSmoking | What provisions are being made for "No Smoking" signs (if required) | |

## SPECIAL EVENTS SERVER

The special events server contained a graphical representation of the City of College Station Special Events Application. For the purposes of this research, the form was replicated exactly as it is used in paper format, including all data entry fields and wording. Figures 25 through 27 show each page of the special event application form. The form is shown with sample data entered.

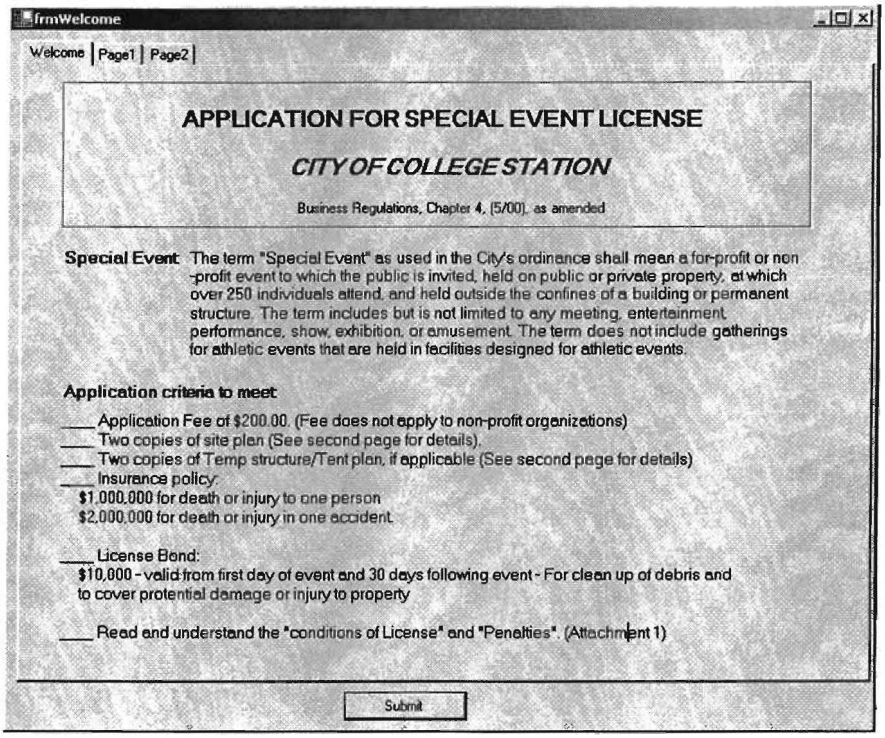


**Figure 25. Cover Page of Special Events Application Form.**

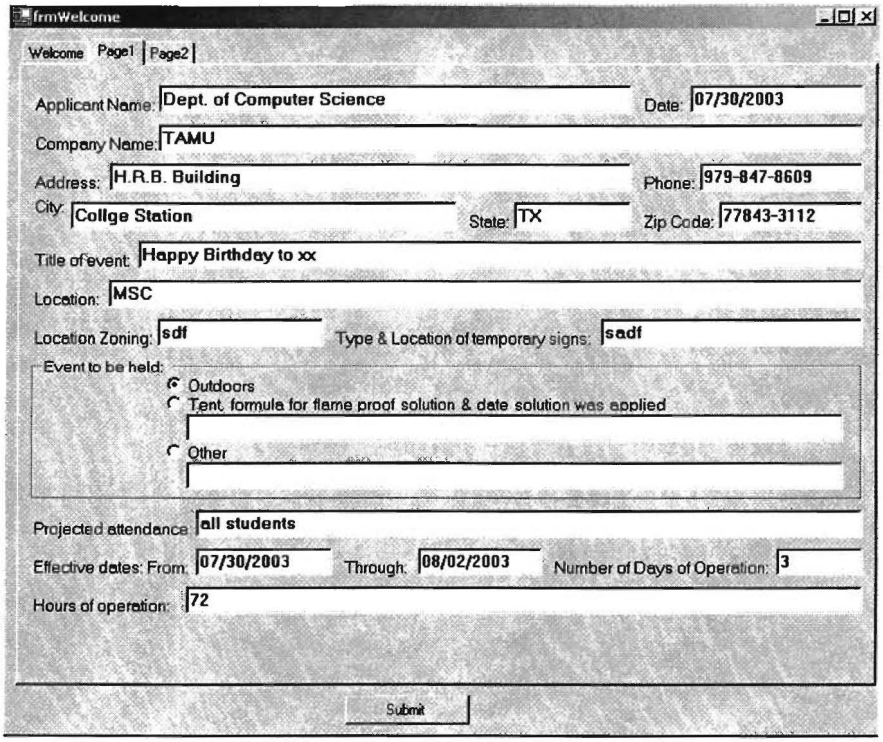


**Figure 26. Page 1 of Special Events Application Form (with sample data).**

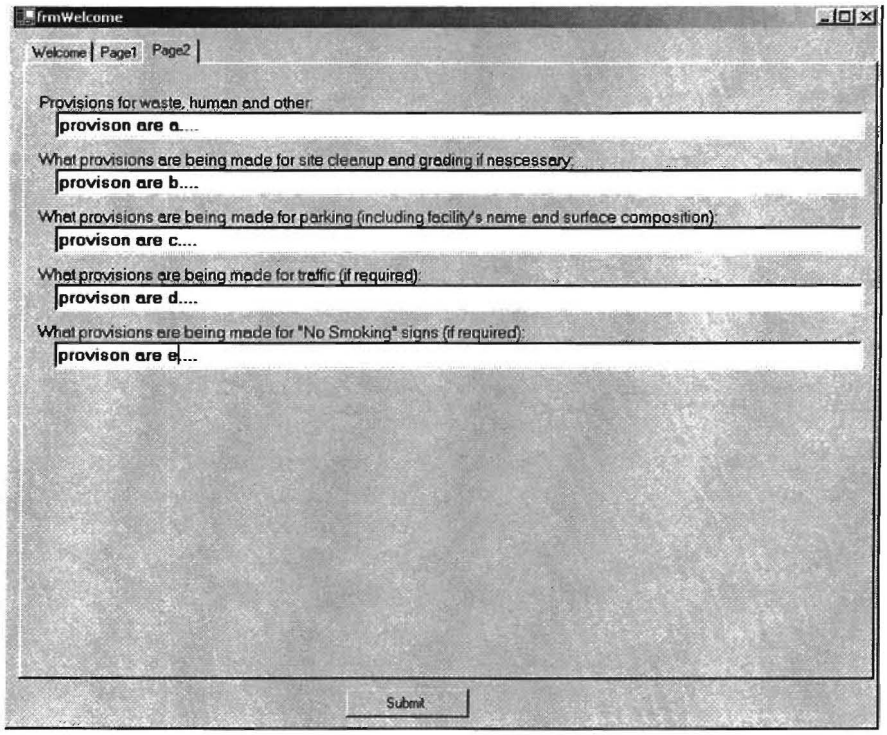


**Figure 27. Page 2 of Special Events Application Form (with sample data).**

## SPECIAL EVENTS CLIENT

The special events client performed two tasks in this application. First, it connected to the C2C infrastructure and sent an information request through the cloud to the special events server. Figure 22 shows the information flow. Figure 28 shows the special events client and the communication messages to and from the cloud. The figure also shows the standard connectivity options of IP address and port.

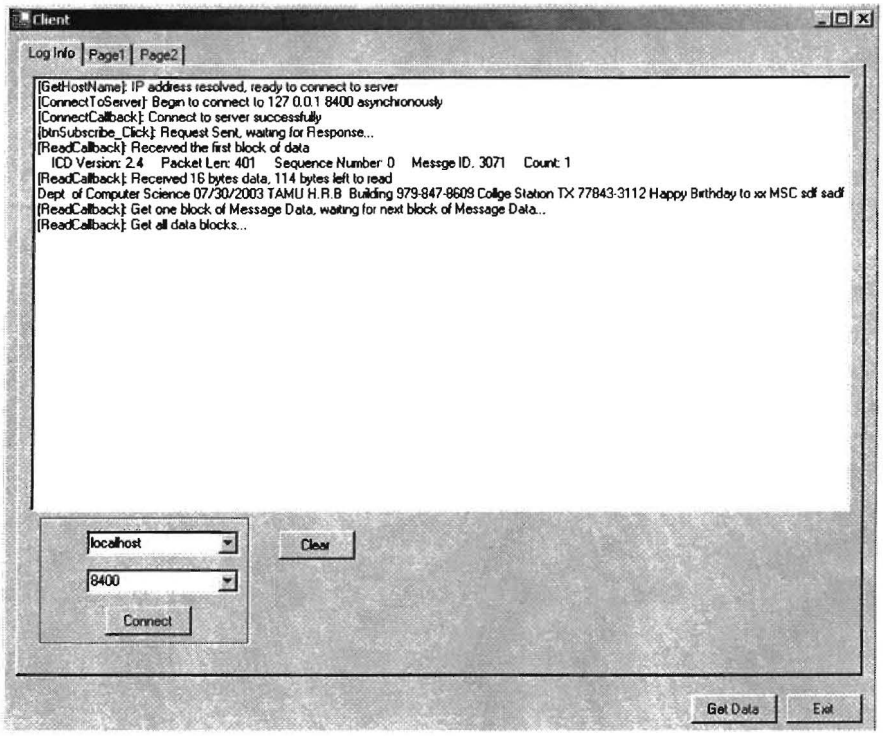


**Figure 28. Special Events Client Communication.**

Second, the special events client displayed the special event information once it was received from the cloud. The client utilizes the same screens as the data input side, so the data display screens on the client look exactly the same as Figures 26 and 27.

## LOW-BANDWIDTH CONNECTION TESTING

The C2C infrastructure established for low-bandwidth testing, and described in Chapter 2 of this report, was again utilized to test the special events addition to C2C. Analogous to previous testing, two test situations were conducted. The first (shown in Figure 29) utilized the special events server on the dial-up connection. The second (shown in Figure 30) utilized both the special events server and client on separate dial-up connections.

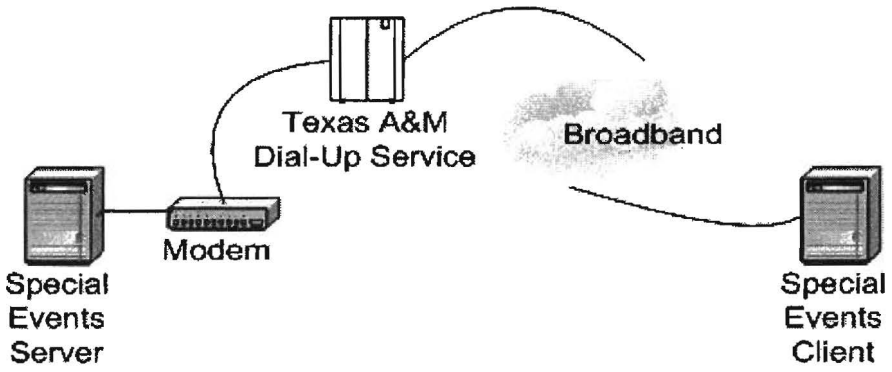


**Figure 29. Special Events Server Using Dial-Up Service.**

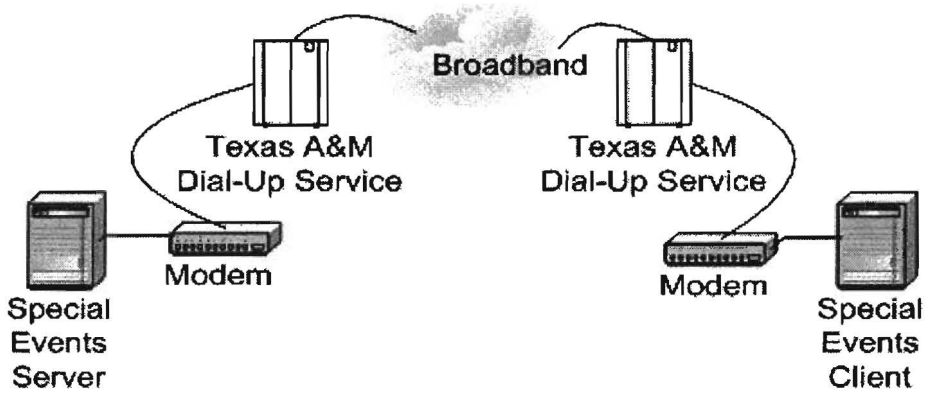


**Figure 30. Both Special Events Server and Client Using Dial-Up Service.**

## RESULTS OF LOW-BANDWIDTH CONNECTION TESTING

The results of the low-bandwidth testing for special events were exactly the same as with C2C snapshots. Specifically, no impact or response was seen that indicated that the C2C infrastructure could not be expanded and those expansions added to the message transferred in a low-bandwidth C2C infrastructure.

# CHAPTER 4:
# THE FUTURE OF C2C

## NEXT GENERATION C2C SPECIFICATION

C2C communications allow different centers to exchange data in a seamless manner. In Texas, C2C support is planned for all existing and future TMCs, building on the initial success with the Dallas and Forth Worth implementations. In effect, participation in the C2C infrastructure will build a statewide traffic management and information sharing capability.

Behind the current implementations are several years of standards development, research efforts, system integration, and software development. These efforts are not yet complete, as some of the core aspects of C2C appear likely to change in the future.

As a communications protocol, DATEX/ASN has never enjoyed a great deal of support. There is only one commercial product/vendor available and no public domain implementations. This contributes to a wholesale lack of support and installed base for DATEX and makes it costly to implement and support.

A number of efforts and trial studies have looked at replacing DATEX with a different protocol for easier and less costly implementation and support. It appears likely that a combination of XML (eXtensible Markup Language) for data encoding and Simple Object Access Protocol (SOAP) for data transfer will be used in the future. This is the direction at both the national level and within TxDOT (*9, 10*).

One of the key merits for changing the communications protocol is security. Passing DATEX messages through agency firewalls is a complicated process. Because XML and SOAP are known protocols in wide use, security considerations are widely known and less of a "threat" when sending information between disparate networks.

In addition to the security considerations, XML with SOAP forms the basis for web services. Unlike a traditional web server and browser approach, which is based on a server delivering data to a GUI, web services deliver data to applications, across a network. In essence, this is the same model in use for C2C communications, and the marriage of the two systems should allow for increased application, ease of use, and interoperability. Finally, XML and SOAP are free and in widespread use, so that support and licensing issues that come with DATEX are non-existent with the next generation of C2C.

Figure 31 illustrates the components of the next generation of the C2C specification. At the core, the individual data elements come from the Traffic Management Data Dictionary. These elements, arranged in message sets, are then encoded, or represented, in XML. This information is then sent via the SOAP communications protocol into the C2C cloud. Test implementations of this specification have been successful and development is actively under way at the national level (*9, 10*).

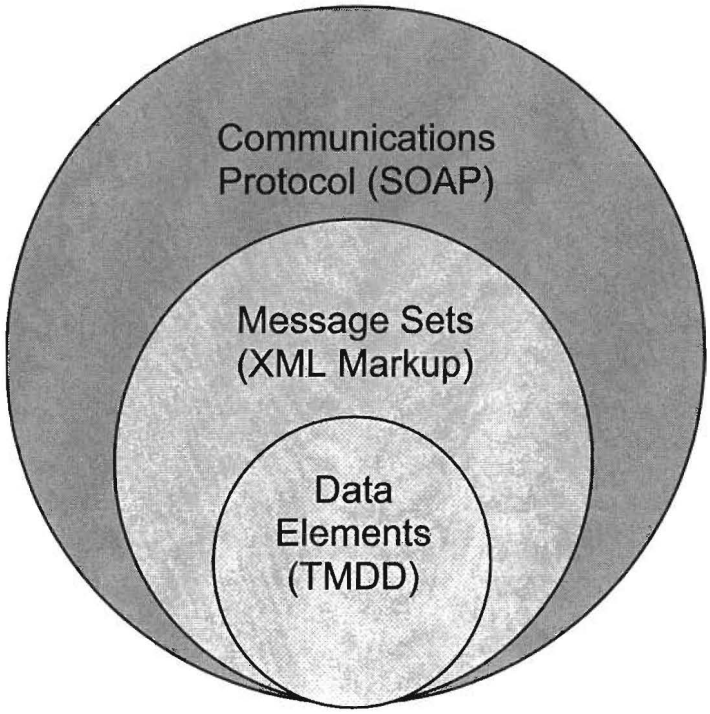


**Figure 31. Next Generation C2C Specification.**

## C2C IN LOW-BANDWIDTH ENVIRONMENTS

The investigations undertaken for this report demonstrate that low-bandwidth communication links are not a hurdle to create on-demand, ad hoc communication connections between TMCs or other types of operation centers. In fact, for the text-based message exchanges currently supported by the C2C specification, phone-line links are quite adequate to convey the information in a timely manner. Additions to the specification should perform in the same manner, based on the experiments with the special events information transfer.

As the amount of information exchanged increases, phone-line-based connectivity will be stressed. This is illustrated by the increasing message transfer times seen for larger CCTV snapshots. Information transfers that increase the data flows, such as video transfer, will stress the system even more.

In terms of control functions, phone-line-based transfers for text information will be adequate. Examples of this type of control include DMS messages, lane control signals, or similar equipment. Real-time control of video equipment, while possible over phone lines, is likely to be somewhat laborious and detrimental to efficient operations, particularly in a critical situation.

The bottom line for low-bandwidth communications is that current C2C implementations can be served over a phone line. Caution must be taken when supplementing the specification to ensure that enough spare capacity exists to accommodate the information transfers.

## POTENTIAL ENHANCEMENTS FOR C2C IN LOW-BANDWIDTH ENVIRONMENTS

The potential exists to enhance the capability of utilizing C2C in a low-bandwidth environment. The key to doing this would be to add a flag to the data types. For example, data might be flagged as critical or standard priority. Only data flagged as critical would be transferred via a low-bandwidth connection. This would preserve the capacity of the communications link and ensure that the truly important information was disseminated between centers.

For broadband connections, both priorities of data would be exchanged. When a broadband data source sends all types of data to the C2C repository, the repository would accept all types of data, but will send out the proper types of data to subscribers according to the bandwidth limit of the subscribers. This capability, while not present in the existing specification, might allow for more flexibility of C2C in the long run and might allow for critical traffic information to achieve a more widespread dissemination as the C2C cloud is expanded in an ad hoc manner during any particular event.

## TRANSLINK® SUPPORT FOR C2C

One of the critical reasons behind the investigation of C2C in a low-bandwidth environment was the College Station Integration Project (CSIP). CSIP is a federally funded deployment project, matched with local dollars, to begin the process of developing integrated operations for managing traffic during special events in the Bryan/College Station area. The project involves multiple partners, including:

- The City of College Station,
- The City of Bryan,
- TxDOT,
- Brazos County,
- College Station Urban Transportation Study Steering Committee (MPO in the Bryan District),
- Brazos Transit,
- Texas A&M University, and
- the Texas Transportation Institute (TTI).

In addition to the above partners, numerous agencies within the surrounding area are participating in the development of a regional architecture to guide future infrastructure development and projects within the ITS arena.

The project has the goal of allowing traffic data, signal timing information, train information, and video to be shared between the City of College Station, the City of Bryan, and TxDOT through the TransLink® Research Center. During special events, agencies can use the TransLink® Laboratory to monitor traffic conditions and change signal timings, in real time, from the laboratory floor. A data archiving system will also be developed and housed in the TransLink® Laboratory that will allow traffic and response information to be collected and archived for use in developing improved real-time traffic management strategies.

The project will provide the City of Bryan, the City of College Station, and TxDOT with the ability to integrate their traffic signal systems along the major travel routes, and yet maintain each agency's autonomy over their respective system. This project also looks to expand the agencies' ability to manage travelers' arrival and departure routes to these special events through motorist information systems.

While the central portion of CSIP is served by a high-speed communications network capable of transmitting data and hundreds of camera images simultaneously, the surrounding areas have no such network connection. Indeed, a low-bandwidth connection, such as a modem dial-up, may connect many partners during special events. As such, the operations and efficiency of the C2C specification in these conditions were critical to research.

Based on the results of investigating the use of C2C in a low-bandwidth environment, researchers believe the specification to be a viable and effective solution for disseminating status and control information to multiple partner agencies. In addition, the C2C communications capability provides a data transfer methodology for the data archiving effort, which is part of the project.

TransLink® will support the next generation of the C2C specification, which uses the TMDD data elements and messages sets, XML encapsulation, and SOAP as a communications protocol. In addition, researchers will supplement the existing message sets with project specific data elements to support project needs. This information and supplementary messages will be fed back into the state and national standards development process.

# REFERENCES


*1.* Gordon, R. L., R. A. Reiss, H. Haenal, E. R. Case, R. L. French, A. Mohaddes, and R. Wolcott. *Traffic Control Systems Handbook.* Report FHWA-SA-95-032. FHWA, U.S. Department of Transportation, Washington, D.C., 1996.

*2.* TMDD Steering Committee. Standards for Traffic Management Center to Center Communications. Volume 1: Concept of Operations and Requirements. Report Traffic Management Center Standard, Rev. 1.3, Draft Final. American Association of State Highway and Transportation Officials and Institute of Transportation Engineers, Washington, D.C., 2003.

*3.* Southwest Research Institute. *Center to Center Communications: Dallas/Ft. Worth Data Server Command/Control Interface Control Document.* Report DFWDS-CICD - Version 1.0. Texas Department of Transportation, Austin, TX, 2000.

*4.* Dellenback, S. W. Texas's Experience in Connecting Dissimilar Traffic Management Centers. Newsletter of the ITS Cooperative Deployment Network. http://www.nawgits.com/icdn/dellenback.html Accessed August 8, 2003.

*5.* Newton, H. *Newton's Telecom Dictionary,* 16th Edition. Telecom Books, New York, NY, 2000.

*6.* Southwest Research Institute. *Center to Center Communications: Dallas/Ft. Worth Data Server Status Interface Control Document.* Report DFWDS-SICD - Version 2.1. Texas Department of Transportation, Austin, TX, 2000.

*7.* Center-to-Center Local Self-Evaluation Report. Report C2C-LSER – Version 1.0 ITS-99 (712). Texas Department of Transportation, Austin, TX, 2003.

*8* Southwest Research Institute. *Center to Center Communications: Command/Control Interface Control Document.* Report C2C-CICD - Version 2.4. Texas Department of Transportation, Austin, TX, 2001.

*9.* Werner, J. *Texas's Center-to- Center-Communications Efforts Evolves Towards a Widely Used Internet Paradigm.* Newsletter of the ITS Cooperative Deployment Network. http://www.nawgits.com/icdn/tx_c2c.html Accessed August 8, 2003.

*10.* Dellenback, S. W. *Investigation into Alternative ITS Protocols for Center-to-Center Communications, 10-9274.* Southwest Research Institute. http://www.swri.edu/3pubs/IRD2002/10-9274.htm Accessed August 8, 2003.