2011 14th International IEEE Conference on
Intelligent Transportation Systems
Washington, DC, USA. October 5-7, 2011

Peer-based Communication for a distributed, virtual Traffic Management Center

Basel H. Kilani, Patrick A. Campbell, Joseph P. Havlicek, Monte P. Tull,
Alan R. Stevenson, and Ronald D. Barnes

Abstract— **In an effort to monitor and alleviate roadway traffic conditions, the Oklahoma Department of Transportation (ODOT) has deployed a statewide Intelligent Transportation Systems (ITS) architecture consisting of a large number of devices, including cameras, dynamic message signs, and speed sensors along Oklahoma highways. These devices are connected throughout a private ITS fiber-optic network to controlling stations located at stakeholder agencies statewide, forming a virtual Traffic Management Center (TMC). This decentralized approach allows individual consoles on the virtual TMC to display and control reachable devices even if portions of the network become disconnected. Enabling this fault-tolerant design is a novel peer-based communications protocol. The communication system is dynamically configured and automatically resolves communications regardless of network configuration. This paper introduces this robust peer-based approach and describes its implementation within the Oklahoma virtual TMC. Results of this implementation of the system are also presented.**

I. Introduction

To avoid the prohibitive cost of the construction and operation of a centralized Traffic Management Center (TMC) [1]–[5], the state of Oklahoma has instead adopted a novel statewide ITS architecture. This has taken the form of a highly cost-effective, decentralized *virtual TMC* that is used by transportation agents distributed across the state to display information from and control ITS devices deployed on Oklahoma roadways [6]. The virtual TMC resides on a private, dual-ring fiber-optic network connecting ITS devices such as cameras, Dynamic Message Signs (DMS), and Remote Traffic Microwave Sensors (RTMS) to controlling stations called *ITS Consoles*. This virtual TMC further integrates data from devices not directly connected to the network (including third-party devices) through the internet. ITS Consoles making up the virtual TMC can be deployed to any location on the ITS network and are in use at almost a hundred stakeholder agencies around the state, including Department of Transportation field offices, municipal governments, the State Emergency Operations Center, and the Oklahoma City 911 Center. This decentralized structure of Oklahoma's ITS architecture results in lower costs, more fault-tolerance, better security, and cross-state accessibility [7].

This work was supported in part by grants from the Oklahoma Department of Transportation and the Oklahoma Highway Safety Office.

B. Kilani, P. Campbell, J. Havlicek, M. Tull and R. Barnes are with the University of Oklahoma, School of Electrical and Computer Engineering, Norman, OK 73019, USA. {basel,pacampbell,joebob,tull,ron}@ou.edu

A. Stevenson is with the the Oklahoma Department of Transportation, Technology Services Division, Oklahoma City, OK 73105, USA. astevenson@odot.org

A peer-based communication system was developed for the distributed Oklahoma ITS network to increase fault-tolerance of the system by decentralizing the flow of data within the network and reducing unnecessary data transmissions. Prior to the development of the peer-based system described in this paper, data from devices connected to the network via the public internet was distributed to all ITS Consoles via a single centralized gateway server that bridged the public internet and the private ITS network. In this arrangement, each ITS console was responsible for both obtaining data from the server and updating the server with any information gathered by the console. Each console was also responsible for actively "pinging" all machines and devices on the network in order to determine network availability.

The new peer based communication system provides a framework for coordinating the flow of information in the virtual TMC by dynamically creating and maintaining master nodes, called *superpeers* within the network. These superpeers serve all network-reachable ITS Consoles, taking on all communications responsibilities for those consoles and greatly reducing the data communications within the network. The system is designed so that any subset of the statewide network of ITS Consoles automatically designates a single machine to be a superpeer. When network disconnections or reconnections occur, the peer-based communication protocol resolves any existing superpeers (or the lack thereof) into a single superpeer. This communication strategy reinforces the decentralized nature of the ITS network by continuing to provide all available functionality regardless of network configuration.

This paper describes the design and development of this peer based communication system. An overview of the Oklahoma ITS architecture is first presented in Section II. In addition, the justification for the new communication system within that architecture is given. The details of the new communication system are presented in Section III, including a description of the system design components and system behavior. Finally, initial results of testing the new communication system are summarized in Section V.

II. Oklahoma ITS

Even though having a TMC is highly desirable for centralized coordination between transportation stakeholders, the costs of construction and operation are substantial. For states with relatively lower budgets, the annual operational cost alone can be prohibitive. According to the Research and Innovative Technology Administration (RITA) the equipment cost for building and maintaining a TMC in a medium city (between 250,000 and 750,000 people) was estimated to be USD 4.3M in 2008 [8]. Furthermore, the estimated TMC operational cost for a regional/statewide TMC with continuous 24/7 operations is approximately USD 1.8M per year [9]. These cost considerations are one of the major factors in the development of the Oklahoma ITS. Due to tight budgetary constraints, Oklahoma's ITS architecture has to be inexpensive and highly cost-efficient.

Robust fault-tolerance is also critical to preventing network and technical failures across the expansive state from causing a system-wide failure. Scalability is another consideration as the extended deployment of the system has spanned more than a decade. The ITS network needs to be easily expanded to new devices and additional roadway and traffic areas. Finally, as transportation agents are spread between two major metropolitan areas and across the diffuse highways throughout the state, access to the system has to be location independent, with stakeholder agencies able to receive data from and control ITS devices from anywhere near the private ITS fiber-optic network.

A. A Decentralized Approach to ITS

In [10], after giving his broad analysis of centralization versus distribution, Amin delves into specific kinds of systems where he describes the need for the complex systems that make up a transportation network to have self-healing features that prevent them from being crippled when a failure occurs in a specific area. Due to its nature, a centralized system unfortunately does not have this ability to self-heal and continue to operate efficiently in the case of a centralized point of failure.

Incorporating these cost and architecture issues, the distributed ITS architecture operating as a virtual TMC was developed in contrast to the centralized TMCs that most other states and large municipalities have implemented. Cost ruled out the construction of a permanent, centralized TMC facility. A TMC would also not be able to fulfill all the requirements mentioned above due to the potential vulnerabilities of a centralized architecture. Oklahoma's distributed ITS operates without the building, maintenance, and personnel costs required by a traditional TMC. The system was implemented through a private communication network connecting transportation and emergency management personnel and locations to roadside ITS devices. Low-cost ITS consoles were deployed to numerous existing agency locations.

B. Robust and Fault-Tolerant ITS Network

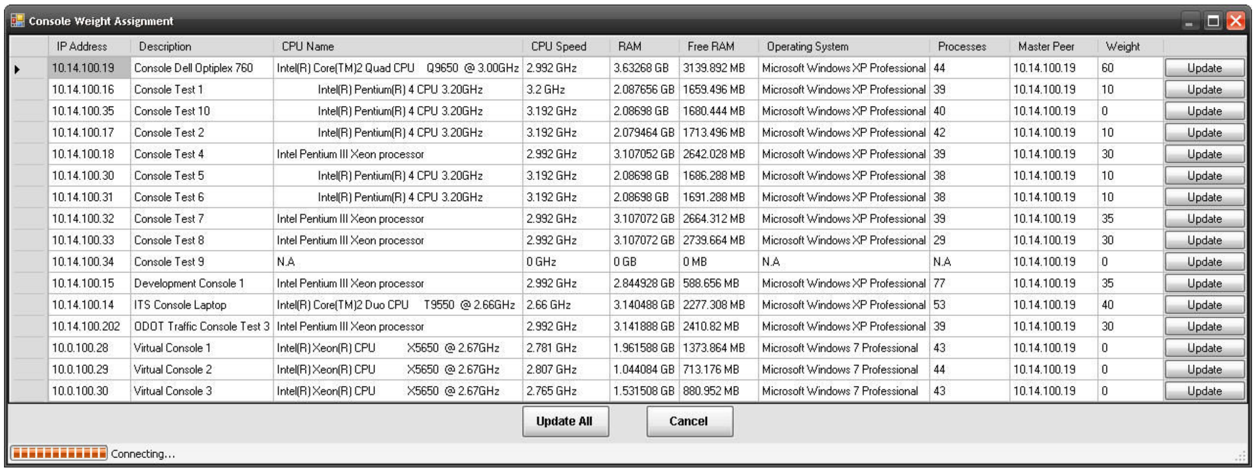
The Oklahoma ITS is built on the premise that an ITS console, disconnected from any part of the ITS network can still continue to function with the other consoles and devices that remain reachable. In this way, an operator using any console can operate independently, controlling those devices that are currently reachable by that console. Thus, when the network is segmented into smaller networks due to hardware failures or interruptions in communications, each of the smaller networks can function independently. Even in the event of an accidental cut of a fiber-optic trunk on the network, any consoles on a smaller network are able to obtain information about and communicate with all other consoles and devices on that network. To achieve this level of robustness, individual consoles that make up the virtual TMC must each be able to recognize and respond to network disconnections.

This fault-tolerance can be approached in a variety of different ways, and several approaches have been applied to the Oklahoma ITS model. Originally, inter-console communication was handled by custom peer-to-peer software. Consoles sent socket messages to all other reachable consoles. Due to the difficulties involved in handling network failures, however, this software was eventually retired and the consoles were modified to use the Microsoft Message Queuing (MSMQ) [11] service to send messages to other network-available consoles. In both of these cases, upon network failure every console is responsible for handling attempts to communicate with devices that are no longer available. If some consoles are no longer available, messages for those consoles are stored until they can eventually be delivered. If other devices are not reachable, the console software times out when attempting to communicate with them.

An improved approach was developed to increase the fault tolerance in the ITS network by augmenting the MSMQ message communication with a superpeer-based communication system. The MSMQ messaging system is still used for a great deal of inter-console communication. Consoles can still control devices directly, but most of the processes of gathering information from devices as well as determining the condition of the network are now delegated to a *superpeer* console that is uniquely and automatically chosen within the network. This superpeer-based communication reduces the load on individual consoles and provides a structure for the organized flow of information within the Oklahoma ITS network.

III. Superpeer Based Communication

Peer to Peer (P2P) network architectures have become increasingly popular with many successful P2P applications including file sharing, content delivery, cloud computing, and communications. Its use in ITS has also been proposed for the exchange of geo-located data [12], for maintaing distributed heterogeneous databases [13], and in vehicle-to-vehicle and vehicle-to-infrastructure communication [14]. Although there are many benefits to these types of architectures, there are also potentially serious drawbacks. In [15], Sacha and Dowling describe how many P2P systems are designed with the assumption that all peers are similar, with equal preference given to each peer. Unfortunately, this arrangement can potentially create bottleneck situations where data is not processed at equal rates because the



Console Weight Assignment

IP Address	Description	CPU Name	CPU Speed	RAM	Free RAM	Operating System	Processes	Master Peer	Weight	
10.14.100.19	Console Dell Optiplex 760	Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00GHz	2.992 GHz	3.63268 GB	3139.892 MB	Microsoft Windows XP Professional	44	10.14.100.19	60	Update
10.14.100.16	Console Test 1	Intel(R) Pentium(R) 4 CPU 3.20GHz	3.2 GHz	2.087656 GB	1659.496 MB	Microsoft Windows XP Professional	39	10.14.100.19	10	Update
10.14.100.35	Console Test 10	Intel(R) Pentium(R) 4 CPU 3.20GHz	3.192 GHz	2.08698 GB	1680.444 MB	Microsoft Windows XP Professional	40	10.14.100.19	0	Update
10.14.100.17	Console Test 2	Intel(R) Pentium(R) 4 CPU 3.20GHz	3.192 GHz	2.079464 GB	1713.496 MB	Microsoft Windows XP Professional	42	10.14.100.19	10	Update
10.14.100.18	Console Test 4	Intel Pentium III Xeon processor	2.992 GHz	3.107052 GB	2642.028 MB	Microsoft Windows XP Professional	39	10.14.100.19	30	Update
10.14.100.30	Console Test 5	Intel(R) Pentium(R) 4 CPU 3.20GHz	3.192 GHz	2.08698 GB	1686.288 MB	Microsoft Windows XP Professional	38	10.14.100.19	10	Update
10.14.100.31	Console Test 6	Intel(R) Pentium(R) 4 CPU 3.20GHz	3.192 GHz	2.08698 GB	1691.288 MB	Microsoft Windows XP Professional	38	10.14.100.19	10	Update
10.14.100.32	Console Test 7	Intel Pentium III Xeon processor	2.992 GHz	3.107072 GB	2664.312 MB	Microsoft Windows XP Professional	39	10.14.100.19	35	Update
10.14.100.33	Console Test 8	Intel Pentium III Xeon processor	2.992 GHz	3.107072 GB	2739.664 MB	Microsoft Windows XP Professional	29	10.14.100.19	30	Update
10.14.100.34	Console Test 9	N.A	0 GHz	0 GB	0 MB	N.A	N.A	10.14.100.19	0	Update
10.14.100.15	Development Console 1	Intel Pentium III Xeon processor	2.992 GHz	2.844928 GB	588.656 MB	Microsoft Windows XP Professional	77	10.14.100.19	35	Update
10.14.100.14	ITS Console Laptop	Intel(R) Core(TM)2 Duo CPU T9550 @ 2.66GHz	2.66 GHz	3.140488 GB	2277.308 MB	Microsoft Windows XP Professional	53	10.14.100.19	40	Update
10.14.100.202	ODOT Traffic Console Test 3	Intel Pentium III Xeon processor	2.992 GHz	3.141888 GB	2410.82 MB	Microsoft Windows XP Professional	39	10.14.100.19	30	Update
10.0.100.28	Virtual Console 1	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	2.781 GHz	1.961588 GB	1373.864 MB	Microsoft Windows 7 Professional	43	10.14.100.19	0	Update
10.0.100.29	Virtual Console 2	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	2.807 GHz	1.044084 GB	713.176 MB	Microsoft Windows 7 Professional	44	10.14.100.19	0	Update
10.0.100.30	Virtual Console 3	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	2.765 GHz	1.531508 GB	880.952 MB	Microsoft Windows 7 Professional	43	10.14.100.19	0	Update

Fig. 1. Peer Weight Assignment Form

different characteristics of the peers within the system. They postulate that a global knowledge of the system is needed to efficiently assign weights to peers based on characteristics such as numbers of connections, bandwidth, uptime, etc. In [16], the idea of "superpeers" is introduced, in which data is stored on the fastest, most reliable peers. However, again because of the lack of global knowledge of the system, there is no clear method of identifying which peers would make the most appropriate superpeers.

In our superpeer-based-communication approach within the ITS network, a single machine is chosen to act as a superpeer, gathering information from services and devices, redistributing that information to all nodes on the network, and actively determining the status of the network. This machine can be a server or an ITS console. Other ITS consoles within this framework operate as normal peers.

A. Peer Weights

The selection of the superpeer within the network is based on *Peer Weights*. Each console on the network is assigned a Peer Weight by network managers, and the Peer Weights of all machines are propagated to and stored on each console. The Peer Weight is a single number, and machines that are better suited to performing superpeer duties are assigned higher Peer Weights. Since superpeer duties can consume system resources, machines that possess greater processing power, more RAM, or are configured as servers are generally assigned higher Peer Weights.

Peer Weights are assigned to machines using a custom Visual Basic form available within each ITS console. This form is shown in Fig. 1. The form displays a list of potential superpeer machines on the network, as well as resource information for each form. The Win32 Classes from Windows Management Instrumentation (WMI) were used to collect the hardware specifications and configurations of the peers, including RAM, processor speed, and operating system. WMI is the infrastructure for management data and operations on Windows-based operating systems [17]. ITS Console users can use the Weight Assignment Form to manually enter Peer Weights for potential superpeer machines. Once the Peer Weights have been entered and the user has clicked 'Update All', the form distributes the new Peer Weights to all available ITS consoles on the network via MSMQ. Upon receiving messages with updated Peer Weights, each ITS console stores the weights in a local database.

B. Superpeer Selection

With replicated copies of the Peer Weights of potential superpeer machines in each local database, every ITS console is capable of determining which machine should act as the superpeer. Pseudo-code for the superpeer selection process is shown in Table I.

The superpeer selection process is completed independently on each ITS console during initial setup. This process consists of determining which potential superpeer machines are available, and then iterating through these machines and selecting the machine with the highest Peer Weight as that console's new superpeer. In the event that multiple potential superpeer machines have been assigned identical Peer Weights, then superpeer responsibilities are arbitrarily given to the machine with the lowest IP address. Since Peer Weights for all machines are stored on each ITS console, all consoles on a connected portion of the network reliably and independently select the same superpeer.

After an ITS console has run the superpeer selection process, it attempts to obtain device and network information directly from the new superpeer. In the case where the ITS console has selected itself as the new superpeer, that console will commence performing superpeer duties, providing device and network information to all other network-reachable consoles. In the case where the selection process is run independently on all consoles within two separate networks, it will successfully result in a single new superpeer selected in each network. Each superpeer will then be responsible for information gathering duties within its own network.

C. Superpeer Duties

The machine selected as the superpeer within a network is responsible for gathering information from services and

TABLE I
PSUEDO-CODE FOR SUPERPEER SELECTION AND DUTIES

```
Determine the Superpeer
  Get a List of ITS Consoles IP address from local Database

  FOR EACH Console IP in the list of Consoles IP
    Get the identified Superpeer's IP and its
      weight from local Database
    PING Console IP
    Update Network Status of Console IP in local Database

    IF Console is Online
      Get Console Weight by connecting remotely to its Database
      PING the Superpeer IP to Verify that it is Online

      IF Superpeer is Online
        Compare Superpeer and Console weight

        IF Superpeer weight > Console weight
          New Superpeer = Superpeer IP

        ELSE IF Superpeer weight < Console weight
          New Superpeer = Console IP
        ELSE
          IF Superpeer IP < Console IP
            New Superpeer = Superpeer IP
          ELSE
            New Superpeer = Console IP

        UPDATE Superpeer field with New Superpeer
          in Local Database
      ELSE
        UPDATE Superpeer field with Console IP in
          Local Database
  NEXT

Determine Superpeer Duties

  IF My IP == Superpeer IP
    Run Who is the Superpeer
    Start Superpeer Services
  ELSE
    Stop Any Superpeer Services (if running)
    Perform Normal Peer Duties
```

TABLE II
NORMAL PEER DUTIES PSUEDO-CODE

```
Get the identified Superpeer's IP from the local Database
PING Superpeer IP
Update Network Status of Superpeer IP in local Database

IF Superpeer is Online
  Connect remotely to Superpeer Database and Verify that
    it is still the Superpeer

  IF Superpeer == Superpeer field on the remote Database
    Get Consoles Network Status
    Get All the ITS Devices Network Status ex: Camera,
      DMS, Alarm, etc.
    Update local Database

  ELSE

    IF My IP == Superpeer field on the remote Database
      Perform Superpeer Duties
    ELSE
      // A new machine has become the Superpeer
      UPDATE Superpeer with Superpeer on the
        remote Database
      Perform Normal Peer Duties
```

devices and actively determining the status of the network. This greatly reduces the work load for normal peer consoles as they can simply obtain information from the superpeer. Dynamic superpeer selection ensures that superpeer duties are always carried out even if some network communications are broken, as isolated sub-networks will select a single superpeer for that portion on the network.

The duties of the superpeer machine primarily consist of actively querying all devices in the local database, determining their availability, and storing their status information in the database. The superpeer is also responsible for actively checking for the existence of a new superpeer. In the event that two sub-networks become reconnected, the new single network should only have a single superpeer. The superpeers of the networks check for new superpeers by running the superpeer selection process at set intervals. In this way, machines that have been newly connected to the network are queried and their Peer Weights are obtained. If any new machines are better qualified to be the superpeer for the network, they are assigned as the superpeer of the current superpeer machine. The current superpeer machine can then revert to a normal peer, and cease to run superpeer duties.

D. Normal Peer Duties

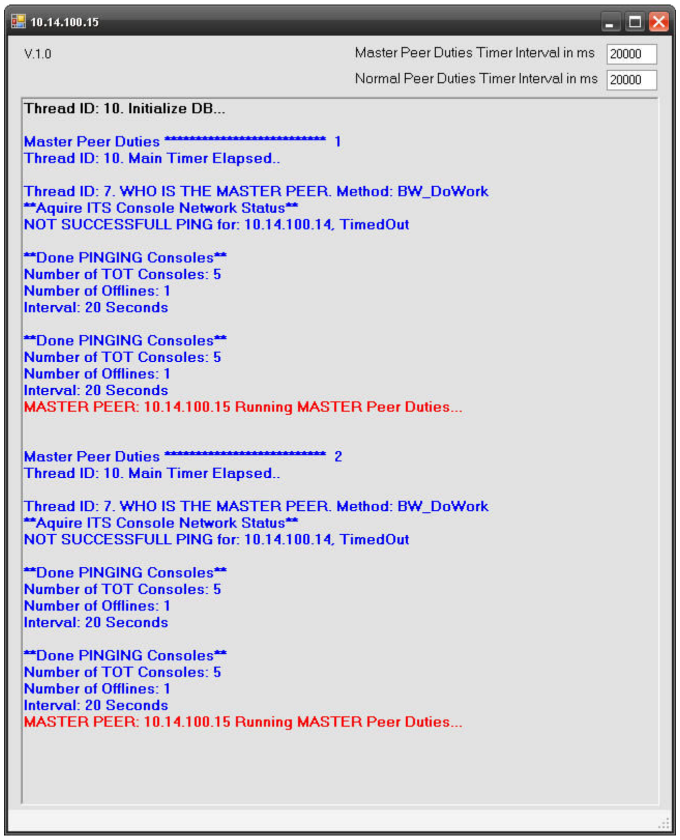
Consoles that are not determined to be superpeers are instead considered to be normal peers. They perform normal peer duties that consist primarily of obtaining information from the current superpeer. Pseudo-code for the normal peer duties is shown in Table II.

The normal peer responsibilities are composed tracking the current superpeer and obtaining information from that superpeer as long as it remains available. The first step in these duties is to check if the normal peer had previously been a superpeer. In this case, superpeer duties are immediately stopped. Most remaining normal peer duties are performed by copying information from the superpeer's database. If at any point the superpeer becomes unavailable, each normal peer runs the selection process to determine which machine should become the new superpeer in the network.

Recall that the superpeer also continuously checks to see if there is a new superpeer for the network. If one has been located, then that machine is assigned as the superpeer of the previous superpeer. Thus, all normal peers become aware of these changes simply by checking the superpeer of their superpeer. If a normal peer finds that a new superpeer has been designated, then the normal peer updates its own database with the new superpeer. From that point on, the normal peer attempts to obtain all information from the new superpeer rather than the previous one.

IV. ADAPTABLE PEER CONFIGURATIONS

The dynamic nature of the superpeer-based system results in very high tolerance of network communication and device failures. When portions of the network are no longer reachable, the superpeer is able to quickly determine which devices or services are unavailable. Normal peers quickly know the network status by simply communicating with the



```
10.14.100.15
V.1.0                              Master Peer Duties Timer Interval in ms  20000
                                   Normal Peer Duties Timer Interval in ms  20000

Thread ID: 10. Initialize DB...

Master Peer Duties ************************** 1
Thread ID: 10. Main Timer Elapsed..

Thread ID: 7. WHO IS THE MASTER PEER. Method: BW_DoWork
**Aquire ITS Console Network Status**
NOT SUCCESSFULL PING for: 10.14.100.14, TimedOut

**Done PINGING Consoles**
Number of TOT Consoles: 5
Number of Offlines: 1
Interval: 20 Seconds

**Done PINGING Consoles**
Number of TOT Consoles: 5
Number of Offlines: 1
Interval: 20 Seconds
MASTER PEER: 10.14.100.15 Running MASTER Peer Duties...


Master Peer Duties ************************** 2
Thread ID: 10. Main Timer Elapsed..

Thread ID: 7. WHO IS THE MASTER PEER. Method: BW_DoWork
**Aquire ITS Console Network Status**
NOT SUCCESSFULL PING for: 10.14.100.14, TimedOut

**Done PINGING Consoles**
Number of TOT Consoles: 5
Number of Offlines: 1
Interval: 20 Seconds

**Done PINGING Consoles**
Number of TOT Consoles: 5
Number of Offlines: 1
Interval: 20 Seconds
MASTER PEER: 10.14.100.15 Running MASTER Peer Duties...
```

Fig. 2. Peer Information/Feedback Window

superpeer instead of having to separately determine device and service availability. Likewise, when devices and services are restored, the superpeer (and thus all normal peers) are soon aware of their status.

In the event that the network becomes separated into isolated networks, the network containing the original superpeer will continue to function as normal, while the nodes of the other isolated network will find that they can no longer connect to that superpeer and will immediately select a new machine within that network to carry out the superpeer's duties. When two isolated networks are reconnected together to form a single network, one of the superpeers immediately relinquishes its superpeer status, and the combined network again has a single superpeer.

In fact, the network can be further subdivided with no loss of stability. The network could be increasingly cut until the only device left on a network is a single ITS console. Even in that case, the console will assign itself superpeer duties and will attempt to communicate with any available ITS devices or services. As isolated networks become reconnected, these duties continue to be carried out by a single superpeer within the network. Even reconnected networks containing three or more superpeers will quickly select a new single superpeer, with any extra superpeers relinquishing their status and assuming normal peer roles. All of this happens automatically, enabling the ITS console network to continue to provide ITS console users with all information available to them in the event of hardware or network failure.

V. Results and System Implementation

The superpeer-based communication system described in the previous sections has been implemented as Windows services using the Visual Basic programming language. These services output information to a feedback window, shown in Fig. 2. This feedback window shows the superpeer selection process in real-time as well as which machine within the network is currently acting as the superpeer. Results of the superpeer duties and normal peer duties are shown so that communication between peers can be monitored along with any changes in peer configuration.

The superpeer duties and normal peer duties have been exhaustively tested and demonstrated in a highly dynamic test environment. The test environment consisted of several networked ITS consoles, with each console independently running peer services. Peer Weights were assigned to each of the consoles using the Weight Assignment Form. Upon initialization, consoles successfully selected the machine with the highest Peer Weight as the superpeer for the network.

Connections between the consoles in the test environment were manipulated by connecting and disconnecting network cables. When the test network was partitioned, the network containing the superpeer continued to function as normal, with the superpeer noting the absence of the disconnected consoles. Meanwhile, consoles in the network without the superpeer, upon determining that the superpeer was no longer available, successfully selected a new superpeer which then began running superpeer duties. Upon reconnecting the test networks, the superpeers successfully selected the original superpeer as the single superpeer within the network and the temporary one reverted to a normal peer. Similar tests were run for different combinations of network structures, including cases where a single console was separated in a network by itself. In each case, a single superpeer performed the appropriate duties within each isolated network.

Further tests were performed in which new Peer Weights were dynamically assigned to consoles using the Weight Assignment Form while peer services were running. In these tests, a machine other than the current superpeer was given a new highest Peer Weight for the network. This implies that this second machine should begin running superpeer duties. In each test, the current superpeer successfully detected that a different machine should be the superpeer for the network. This machine was successfully marked as the new superpeer, with the current one stopping its duties and starting normal peer duties. The new superpeer would then read from the original superpeer's database that it was the new superpeer. It immediately ceased normal peer duties and began running superpeer duties.

VI. Conclusions and Future Work

This paper describes the design and implementation of a new peer based communication system within the Oklahoma ITS network. This system makes use of individual "superpeers" that are dynamically selected in parallel by distributed nodes of the system. This selection is automatic and highly responsive to changes in network structure. The superpeer selection process is based on the use of Peer Weights that are tailored to the qualifications of each node, and superpeers assume communications responsibilities for the entire network. The new peer-based communication system has proven to be robust, resulting in additional fault-tolerance and significantly improving the efficiency of data communications within the large distributed ITS network.

There are potential system features that could be developed to improve the functionality of this system. Currently, superpeers within the network gather data from the single centralized gateway between the private ITS network and the public internet mentioned in Section I. While the use of superpeer-based communication has eliminated the need for each console to communicate directly with this gateway, it still represents a single point of failure. This could be relieved with the installation of additional secure gateways at different locations within the statewide network. In this case, if the superpeer responsibilities are updated to include location of and communication with available gateways, the Oklahoma ITS network would be able to retain communications with the devices on the public internet even in the event of server failure or network disconnection.

The peer-based communication system could further be used to relieve additional communication strain on the network. In addition to the ITS devices on the network, there are also multiple monitoring stations that provide the status of ODOT property and devices. This status information includes temperature, connectivity, and security data and are currently gathered by the central server on the network. The collection of this status data could also be added to the superpeer responsibilities, which would both reduce the load on the central server and ensure that all monitoring continues in the event of hardware or network failure.

References

[1] R. Bradley, "*Walking in L.A.: The Data Driven City* ," *GOOD*, June 2010, available: http://www.good.is/post/walking-in-l-a-the-data-driven-city/ (visited November, 2010).

[2] C. Krueger, C. Hedden, and S. Langlois, "The Chicago Traffic Management Center preliminary design study planning effort," in *Proceedings of the 13th ITS America Annual Meeting*, May 2003.

[3] "2006 annual report SMART SunGuide Transportation Management Center (TMC," Florida DOT District IV, Tech. Rep., Jan 2007, available: http://www.smartsunguide.com/PDF/Annual%20Report%20-%20screen%20V2.pdf (visited May, 2011).

[4] "2005 annual report SMART SunGuide transportation management center (TMC)," Florida DOT District IV, Tech. Rep., Jan 2006, available: http://www.smartsunguide.com/PDF/Annual%20Report%2006_JAN_31%20FINAL.pdf (visited May, 2011).

[5] J. Perrin, R. Disegni, and B. Rama, "Advanced transportation management system elemental cost benefit assessment," The Utah DOT and U.S. DOT, Tech. Rep., March 2004.

[6] R. Huck, J. Havlicek, J. Sluss, Jr., and A. Stevenson, "A low-cost distributed control architecture for intelligent transportation systems deployment in the state of oklahoma," in *Proc. IEEE Int'l. Conf. Intel. Transportation Syst.*, Vienna, Austria, Sep. 2005, pp. 919–924.

[7] B. Kilani, E. Vorakitolan, J. Havlicek, M. Tull, and A. Stevenson, "Distributed ITS control and the Oklahoma virtual TMC," in *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, Oct. 2009, pp. 1–6.

[8] *RITA ITS Cost Database*. Available: http://www.itscosts.its.dot.gov/its/benecost.nsf/ByLink/CostDocs (visited April, 2011).

[9] U.S. DOT Federal Highway Administration, "TMC pooled fund study," in *Transportation Management Center: Business Planning and Plans Handbook*, December 2005.

[10] M. Amin, "National infrastructure as complex interactive networks," in *Automation, Control, and Complexity: An Integrated Approach*, T. Samad and J. Weyrauch, Eds. New York, NY: John Wiley and Sons LTD, 2000, pp. 263–286.

[11] *Message Queuing Overview* . Available: http://msdn.microsoft.com/en-us/library/ms703216(v=VS.85).aspx (visited October, 2010).

[12] E. Kuhn, R. Mordinyi, H.-D. Goiss, S. Bessler, and S. Tomic, "A P2P network of space containers for efficient management of spatial-temporal data in intelligent transportation scenarios," in *Proceedings of the Eighth International Symposium on Parallel and Distributed Computing*, July 2009, pp. 218–225.

[13] X. Hong, Z. Yi, H. Jianming, and S. Jingyan, "A dynamic data information management algorithm for multi-database system under WAN," in *Proceedings of the 2005 Intelligent Vehicles Symposium*, June 2005, pp. 545–549.

[14] T. Ernst, V. Nebehaj, and R. Srasen, "CVIS: CALM proof of concept preliminary results," in *Proceedings of the 9th International Conference on Intelligent Transport Systems Telecommunications*, Oct. 2009, pp. 80–85.

[15] J. Sacha and J. Dowling, "A gradient topology for master-slave replication in peer-to-peer environments," in *Databases, Information Systems, and Peer-to-Peer Computing 2005*, 2005, pp. 86–97.

[16] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings of the 19th International Conference on Data Engineering*, March 2003, pp. 49–60.

[17] *Windows Management Instrumentation* . Available: http://msdn.microsoft.com/en-us/library/aa394582(v=VS.85).aspx (visited April, 2011).