Prioritization of Emergency Network Traffic using Ticket Servers: A Performance Analysis

Cory Beard, University of Missouri-Kansas City, BeardC@umkc.edu
Victor Frost, University of Kansas, frost@eecs.ku.edu

Abstract: Broadband packet networks must dynamically recognize some network flows, like those that deal with disaster response, military operations, or emergencies as having greater importance than others. An architecture of geographically distributed ticket servers has been proposed to issue importance tickets to identify the priority that a flow should be given in the current dynamic network context. Any type of user or flow can be given priority, depending on the user needs and the context. Here the potential viability of such an architecture is studied with respect to the performance requirements of ticket servers, network overhead that would be added, and the severity of longer connection setup times. A hybrid simulation approach is used. The negotiation interaction between users and servers is simulated in a prototype. Network and ticket server performance are modeled using prototype results combined with analytical techniques for networks of queues. Two scenarios are studied, a hurricane event and an office building bombing event, and ticket server performance requirements and connection setup delays were not found to be prohibitive to the successful implementation of such an architecture.

Keywords: Network management, ticket servers, performance.

I. INTRODUCTION

Modern broadband data networks are being designed to integrate all types of multimedia traffic. More importantly, however, they are designed to integrate and support the activities of all types of *users*. As networks become more and more useful to society, new types of users and user applications emerge, and people will increasingly rely upon these networks to be available and reliable.

The user type of particular interest in this paper is the National Security/Emergency Preparedness (NS/EP) user. Recent terrorist events in the United States on September 11, 2001, have shown that telecommunication networks provide

---

An earlier version of this work appeared as "Ticket Server Performance Evaluation Using a Hybrid Simulation Approach," *Proceedings of the SCS 2001 Advanced Simulation Technologies Conference*, Seattle, Washington, April 2001.

This work was partially supported by the Madison and Lila Self Graduate Fellowship at the University of Kansas

tremendous value to society in response to disasters. These events have also shown what is common with disaster response, however, that tremendous stress is placed on these networks. In New York City, most notably the stress was on wireless networks and the public switched telephone network (PSTN). The stress came both from damaged facilities and network demand up to 400% of normal [1].

Of particular interest for this paper are multimedia communications and applications over the public Internet. Aside from isolated problems with web sites of news organizations, the Internet performed admirably after the events of September 11 [2]. But it is anticipated that as provision of voice and video services increases over the Internet the same problems experienced by telephone and wireless networks will increasingly be seen on the public Internet.

NS/EP users currently use the public Internet for outreach, information sharing, and electronic mail. However, use of the public Internet for mission-critical activities is currently modest [3, 4]. Instead of using the public Internet for critical functions, the NS/EP community depends more on specialized PSTN services, like the Government Emergency Telecommunication Services (GETS) [4], and on dedicated TCP/IP networks. The reliability and security of the public Internet is considered inadequate for mission-critical functions, even though several applications have been identified that could make valuable use of the public Internet [5].

This paper specifically addresses the issue of network resource availability for NS/EP network communications. It is common in NS/EP contexts for the availability of network resources to be severely restricted in the aftermath of a major event [6] [7] [8] [9]. In some cases, demand on traditional telecommunications networks has reached five times normal levels during the first day of an event [10]. Much of the traffic demand, however, is for lower priority uses, such as people outside a disaster area calling to see the status of loved ones [11]. Also, user behavior can hamper disaster response. For example, with the Alfred P. Murrah Federal Building Bombing in Oklahoma City in 1995, out-of-town media called emergency 9-1-1 centers and demanded to stay on the line until they could get information ([9], p. 356). Overloaded circuits also made it very difficult for off-duty emergency workers to call the emergency command center when they received a page to come help with the emergency.

Serving priority users in NS/EP situations involves providing a so-called "emergency lane" for priority traffic [12] [13] [14] where important resource requests are given priority access to network resources. This can be realized in many ways, for example through using dedicated resources, policies to limit resource usage by low priority users [15], or preemption of low priority users when no free capacity is available.

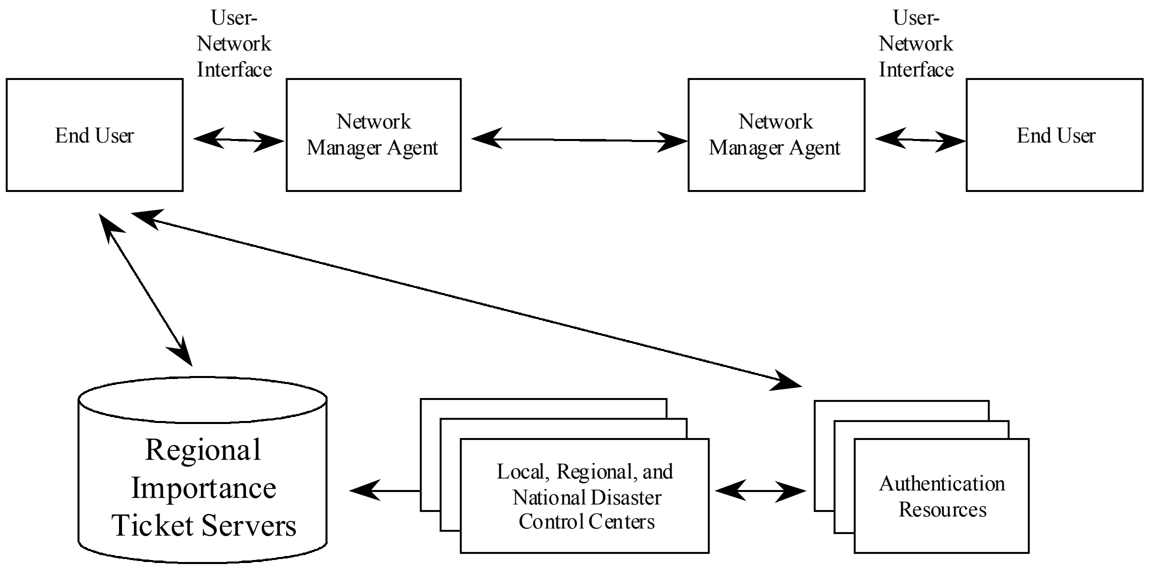

FIGURE 1 - FLOW IMPORTANCE ADMINISTRATION ARCHITECTURE

When resources are scarce, those users and user applications that are of higher value or importance should be given greater access to resources. During congestion that occurs in normal conditions, those of greater importance to network operators would be those which have greater revenue-generating capability. During exceptional conditions like emergencies or disasters, however, users and user applications of greatest importance would likely be those which relieve danger to life and property. A mechanism for prioritizing user traffic would therefore be valuable in both normal and exceptional operating contexts.

In [16], a ticket server architecture was proposed to manage priority network activities. This architecture is illustrated in Figure 1. Users interact directly with regionally distributed ticket servers to request importance tickets that can be presented to network manager agents along with priority requests for network resources. These network manager agents control access to wired Internet communications, wireless communications, etc. The ticket servers maintain a model of the dynamic crisis context of the network and issue tickets according to a user's identity, organization, and need in the current context. Ticket servers maintain this context model through coordination with local, regional, and national disaster control centers.

In addition to these dynamic models of the network context, this architecture also provides users with great flexibility in negotiating for importance tickets. Users are able to request tickets directly, find out why tickets were not granted, update information, provide authentication to verify information they have provided, and even request that the server reconsider its view of the current context to match the user's view of that context. It is of great benefit to users, especially in tense conditions caused by a crisis, to have these capabilities. Intelligent agent technology and agent communication languages are used to provide automated mechanisms that facilitate this interaction for the user.

Several important performance issues must be considered with such an architecture. Interaction with ticket servers should not cause prohibitively long connection setup times, ticket server performance requirements must be reasonable, and excessive amounts of extra network traffic must not be generated. This paper will show that these issues are not significant enough to prevent a successful implementation of the architecture, especially in light of the benefits such an architecture would provide.

A quantitative performance analysis of this architecture was conducted using an approach that combined system prototyping and analytical network queueing analysis. A system prototype was used to simulate the interaction of users with ticket servers and record the numbers and types of interactions that occurred. Then the performance of the ticket servers was assessed using this information, combined with analytical models for a network of queues. Section II discusses the simulation of the user negotiation processes. Section III then discusses the analytical performance assessment of the servers. Results from using this methodology are then presented for simulation of a hurricane event and an office building bombing event similar to the Alfred P. Murrah Federal Building Bombing in Oklahoma City in 1995 [9].

II. SYSTEM PROTOTYPE DESIGN

Since users interact and negotiate with ticket servers using intelligent agents and an agent communication language, the first objective of this research was to create a system prototype to imitate these interactions. User and ticket server processes were created using Java Agent Template, Lite (JATLite) [17] that provides extensions to the Java programming language to support agent communication using KQML [18]. To provide intelligent agent capabilities, the Java Expert System Shell (JESS) [19] was used. This provided a rule-based expert system shell for user agents to generate requests for tickets and respond to responses to those requests. JESS was also used for server processes to respond to requests for tickets based on the current dynamic context and respond to user requests to renegotiate tickets. Ticket server processes also implemented rudimentary mechanisms for guarding against harmful user behaviors. One example was a mechanism for ending a negotiation session once it was unlikely that a user's further interaction would significantly improve a ticket. If not controlled, such user behavior could generate excessive unnecessary load on ticket servers.

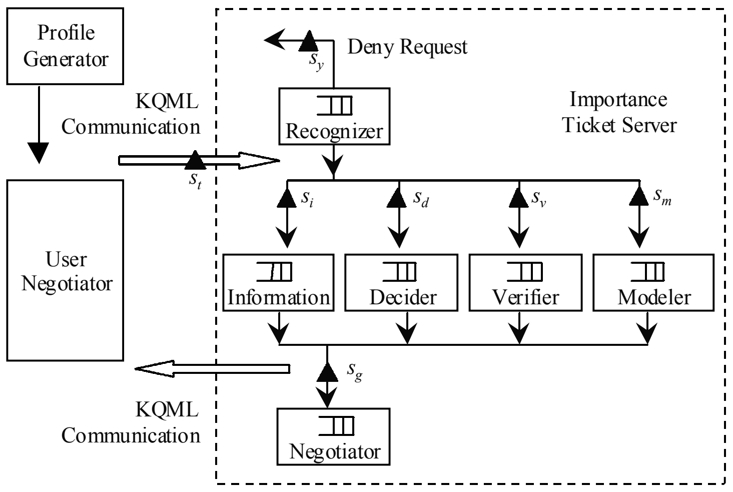

Figure 2. Prototype System Components

The purpose of the prototype was to characterize the negotiation process for a particular *profile*. A profile was defined as the interaction of a user with a ticket server in a certain context. A series of profiles could then be combined into a *scenario* to observe the negotiation process timeline as context changed.

Figure 2 shows the overall design of the user and server processes for the prototype. The user process consisted of the User Negotiator that performed the negotiation activity for the user and generated KQML communication. A profile generator was used to start a particular negotiation session for a particular profile.

The Importance Ticket Server process was modeled as a combination of six modules. Each module performed a distinct function and could process a certain number of requests per second. By explicitly identifying each of these modules, it is possible to assess the performance of each module as a queueing system. The next section shows how these modules were combined to find the performance of the server as a whole. The black triangles shown at each module were used to assess the load on each module by recording the total number of requests per negotiation session that entered each module. These total numbers of requests were stored in variables s_t, s_y, s_i, s_d, s_v, s_m, and s_g respectively.

A brief description of each module in the ticket server is as follows.

- Recognizer – Takes a user message and retrieves a session record about an ongoing ticket server session for that user.
- Information Module – Processes messages where users request information to be stored in their session record or ask about the information in their record.
- Decider – An expert system that makes the decision about the value of a ticket to issue based on the context status maintained by the Modeler.

- Verifier – Verifies information provided by the user by contacting references given by the user or checking the validity of verification information provided directly by the user.
- Modeler – Models the dynamic context in which the network is operating. This model can be updated manually by human experts, automatically in coordination with disaster control centers, or in response to information provided by a user. For the purpose of the prototype, it was assumed that the context was predefined in the profile for a particular negotiation session.
- Negotiator – Once a request has been processed, the Negotiator decides how a subsequent request for this session should be handled. It may decide to end the session or allow the session to continue.

III. PERFORMANCE ANALYSIS APPROACH

Once the total number of messages per session that enter each module for a particular profile are recorded, a queueing network model can be used determine the overall performance of the server in the presence of many requests from different types of users. When combined with the arrival process of requests for new sessions from particular profiles, the overall arrival rate to each module can be determined.

It was assumed that requests for tickets arrived to ticket servers at the Recognizer according to a Markov process of rate γ. An M/M/1 queue was then used to model each module. A more accurate characterization of each module might have been to assume deterministic service times (i.e., M/D/1), since no appreciable difference existed in the way a message was handled between different users or messages. It is more complicated, however, to characterize a network with deterministic service times. An M/M/1 system, can be analyzed, however, and M/M/1 delays are greater than or equal to M/D/1 delays for the same arrival and service rates.

Figure 3 shows the model of the queueing network that was used. When a request first arrived to the server, it was routed to the Recognizer. The Recognizer created a new session record for the current request and then routed the request to a subsequent module based on the nature of the user's request. If the server had established that this type of request should not be considered, the message was routed to the Deny Request module. If not sent to the Deny Request module, the message was processed by the appropriate module and then sent to the Negotiator. The Negotiator then decided how a future request for this session should be treated.

After having a message processed at either the Negotiator or Deny Request modules, the user was notified of the result of the current request, given a ticket if appropriate, and then was given a chance to respond. The user could choose to end

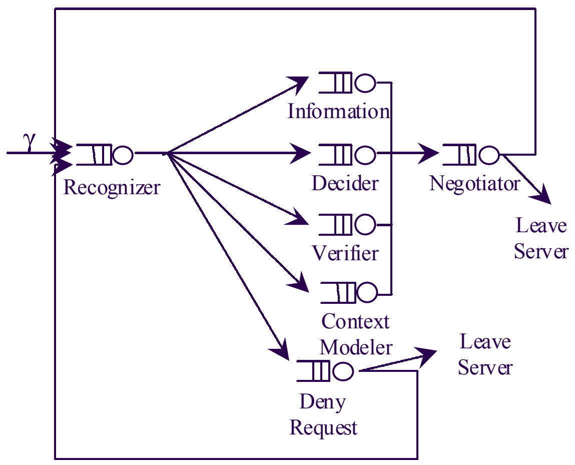

Figure 3. Queueing Network Model

the session, which is signified by the "Leave Server" routes out of the Negotiator and Deny Request modules in Figure 3. The expected total delay from an initial arrival into the Recognizer until taking one of the "Leave Server" paths, therefore, signified the total setup delay that was incurred by the user from seeking a ticket from the server.

If the user chose to respond to the server by making another request, this is signified by the feedback loops in Figure 3. The next message from the user was then handled by the Recognizer. If the Negotiator previously decided to not allow the session to continue, the request was sent to the Deny Request module. The Deny Request module then sent a message to the user indicating that the session had been terminated. Users could continue to make requests, as signified by the feedback loop from the Deny Request module, but requests would always be denied. At the point the user chose to cease making these requests, the "Leave Server" path from the Deny Request module was taken and total delay could be computed.

Total delay was computed as follows. Given an arrival rate for requests for new sessions, γ, the total number of requests per negotiation session that entered each module could be used to determine routing between queues. These total numbers of requests per session were given in variables s_t, s_y, s_i, s_d, s_v, s_m, and s_g respectively as shown in Figure 2.

For example, assume a particular profile generated five messages to the ticket server, so that s_t=5. The user ends the session after being notified by the Deny Request module that the session had been terminated. Two messages were sent to the Information module (s_i=2), and one apiece were sent to the Decider, Verifier, and Deny Request modules (s_y=s_d=s_v=1). Routing probabilities out of the Recognizer, therefore, are 2/5 to the Information module, 1/5 to the Decider, Verifier, and Deny Request modules, and 0 to the Context Modeler module. Routing out of the Negotiator flows back to the Recognizer with probability one (since the user never chooses to end the session after a message from the Negotiator). Routing out of the Deny Request module follows the "Leave Server" path with probability one (i.e., the only time a message is processed by the Deny Request module, the result is a "Leave Server" response).

Note that it is assumed that the queueing network behaves as if it were using probabilistic routing. Even though the route a particular profile will take through the network is deterministic, it can be viewed as probabilistic routing when looking at the aggregate of a class of users.

The overall expected delay through the server for the completion of a full negotiation session is D_{server}. To find the delay between an arrival and the point where the session leaves the server, the arrival rates into each queue, λ_i, λ_d, λ_v, λ_m, λ_g, λ_y, and λ_c, must first be determined. For the decider module, it can be shown that, λ_d is

$$\lambda_d = s_d \gamma \tag{1}$$

This result follows intuition that says that the arrival rate into a queue is equal to the session arrival rate to the server times the number of times the user interacts with a module during a session. The arrival rates to all other queues have correspondingly similar equations.

The output of each M/M/1 queue would be a Poisson process and standard results for delay in an open network of M/M/1 queues can be used. Queues are assumed to be independent and M/M/1 to yield the following result.

$$\begin{aligned} K &= 7 = \text{number of queues} \\ k &= 1,\ldots,7 = \text{index for the } k^{th} \text{ queue} \\ \mu_k &= \text{service rate for server } k \\ \rho_k &= \frac{\lambda_k}{\mu_k} \end{aligned} \tag{2}$$

$$\begin{aligned} D_{\text{total}} &= \text{Expected Delay for an Open M/M/1 Queueing Network} \\ &= \frac{1}{\gamma} \sum_{k=1}^{K} \frac{\rho_k}{1-\rho_k} \end{aligned} \tag{3}$$

Substituting results from (1) and (3),

$$D_{\text{total}} = \frac{1}{\gamma} \sum_{k=1}^{K} \left(\frac{\frac{s_k \gamma}{\mu_k}}{1 - \frac{s_k \gamma}{\mu_k}} \right) = \sum_{k=1}^{K} \frac{s_k}{\mu_k - s_k \gamma} . \tag{4}$$

Using the standard result for the expected delay through an M/M/1 queue,

$$D_{\mathrm{M/M/1}} = \frac{1}{\mu - \lambda}, \tag{5}$$

the expected delay through the k^{th} queue can then be designated as

$$D_k = \frac{1}{\mu_k - \lambda_k} = \frac{1}{\mu_k - s_k \gamma}. \tag{6}$$

Using (4) and (6), the expected delay through the entire server can then be written as

$$\begin{aligned} D_{\text{server}} &= \sum_{k=1}^{7} s_k D_k \\ &= s_i D_i + s_d D_d + s_v D_v + s_m D_m + s_g D_g + s_t D_c + s_y D_y. \end{aligned} \tag{7}$$

This result indicates that the expected delay through the server is the sum of the delays through each queue times the number of messages per session into that queue. Note that in Figure 2 the number of messages per session into the recognizer was s_t, the total number of interactions per session with the server. That is why D_c is multiplied by s_t in (7), where D_c is the delay through the Recognizer.

A user's session delay is defined as the time between a session arrival and the time the session ends through one of the two routes to leave the server shown in Figure 3. To find this total delay, considerations for security processing and message transfer delay are also included.. By denoting the delay for security processing (i.e., encryption, authorization, etc.) as D_{sec} and the network message transfer delay in each direction as D_{net}, the total delay for interaction with the server is

$$D_{\text{total}} = D_{server} + 2s_t(D_{\text{sec}} + D_{net}). \tag{8}$$

Figure 4 shows the queueing network when it supports two types of profiles simultaneously. As a message leaves the Recognizer it is first routed depending on its profile. When considering multiple profiles simultaneously present in the server, as in Figure 4, D_{server} for class r is defined as

$$D_{r,\text{server}} = s_{ir} D_i + s_{dr} D_d + s_{vr} D_v + s_{mr} D_m + s_{gr} D_g + s_{tr} D_c + s_{yr} D_y, \tag{9}$$

where, for example, s_{ir} is the total number of messages to the Information module per session for profile r. The delay for module k is computed as

$$D_k = \frac{1}{\mu_k - \lambda_k} = \frac{1}{\mu_k - \sum_{r=1}^{R} s_{kr} \gamma_r}. \tag{10}$$

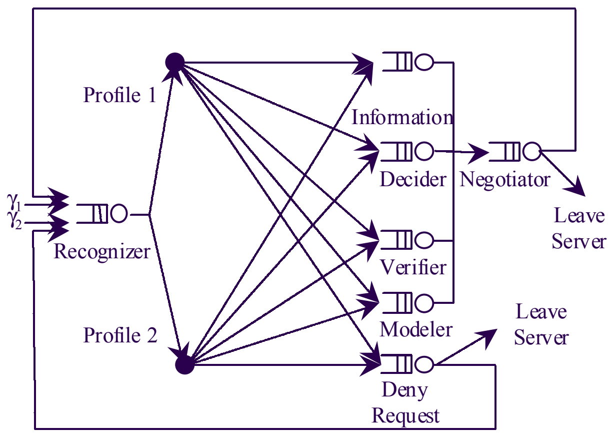

Figure 4. Queueing Network Model With Two Profiles

where γ_r is the session arrival rate for profile r. Total delay is found the same as in (8).

IV. HURRICANE SCENARIO

Two scenarios were designed to assess the performance and feasibility of the ticket server architecture for a major disaster [20]. The first was intended to model a hurricane where disaster efforts were spread over a broad geographic area and communication capacity was reduced from damage to the communications infrastructure. The second was meant to consider a bombing to an office building where damage was localized and network traffic was concentrated.

Before addressing the specifics of each scenario, it is important to consider the conditions the scenarios were intended to reproduce. The basic thesis of this research is that the public network would be a useful resource in disaster recovery if access to resources were based on context-sensitive priorities. The scenarios therefore concentrated exclusively on the use of public network resources. In this context, disaster response can involve some or all of the following considerations.

- Geographic distribution – The impact of a disaster can be spread over a wide geographic area.
- Communication system damage – Disasters may cause a reduction in available public network resources for some or all of the time of disaster response.
- Communications trunks – The focus here will be on major trunks that carry large amounts of traffic.
- Overloading – Even if capacity were not degraded, the load on a communication system typically increases dramatically in disaster. This load consists of load from disaster management personnel and from people affected by the disaster. A major increase in load is also typically generated from users outside the disaster area who are seeking to contact users in the area to find out if they were affected [21].

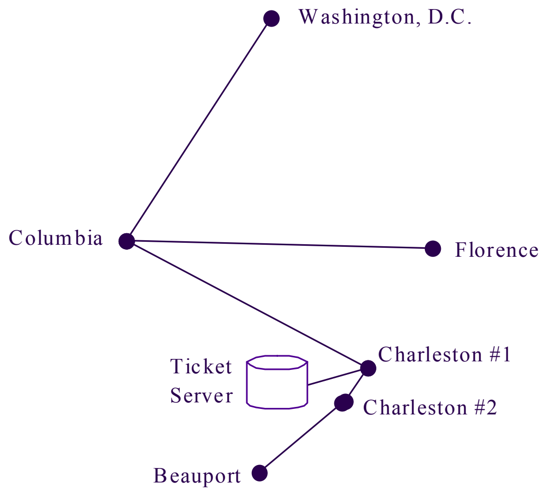

Figure 5. Network For Hurricane Scenario

Table 1. Table Of The Dynamic Context For The Hurricane Scenario

Stage	Hour	Context
1	0-24	Hurricane Preparation and Occurrence
2	25-48	Search and Rescue
3	49-72	Medical Relief
4	73-96	Basic Needs Provisioning (Food and Housing)

- New users – Many new users may be introduced into the disaster area who are search and rescue, assessment, medical, or economic relief experts. New users would have to take extra steps in the negotiation process with ticket servers to verify their identity. Existing users are assumed to have already established their identity.
- Disaster response phases – Disaster recovery can occur in several distinct phases. Ticket servers must adjust how they issue tickets depending on how disaster recovery is progressing.
- User types – For some types of users, their importance as recognized by ticket servers will not change. It is expected, however, that for many users the importance levels will change related to the dynamic context of the disaster.
- Length of disaster – The focus here was on the first 72 hours after an event. Communications load typically returns close to normal levels after this period [22]. Also, mortality rates increase dramatically after 72 hours [23], so the focus here was to make best use of communications resources before 72 hours had expired.

The factors above determined the types of users, how they interacted with ticket servers, and the loads on network links. Performance analysis was also affected by the methods by which users interacted with ticket servers.

The first scenario replicated conditions that occur during a hurricane disaster. Figure 5 illustrates the example communications network that was considered. The cities are some of those in South Carolina that were affected by Hurricane Hugo in September of 1989 [6, 7]. Each line signifies a direct high capacity broadband link. Charleston was assumed to have two major nodes with a link in between.

Table 2. Table Of User Classes For The Hurricane Scenario

Class #	Description	Ticket Value Per Stage			
		1	2	3	4
1	E-911	5	5	5	5
2	Police/Fire	4	4	4	4
3	State Disaster Managers	4	4	3	3
4	FEMA Managers	3	4	3	2
5	Search and Rescue	5	5	4	4
6	Medical	2	4	4	3
7	Needs Provisioning	2	2	3	4
8	Media	3	4	3	3
9	General public in disaster area – non-emergency communications	1	1	1	1
10	General public from outside disaster area	0	0	0	0

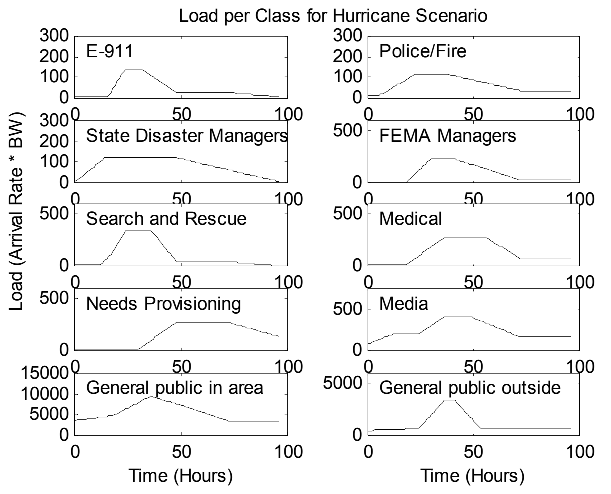

Figure 6. Class Loads For Hurricane Scenario

Servers were directly associated with links and tickets had to be obtained separately for each link on a path. This is not the best way to administer tickets, but does represent the worst case from a performance standpoint. Here it was assumed that a ticket server was located at the Charleston#1 node to issue tickets for the Columbia-Charleston#1, Charleston#1-Charleston#2, and Charleston#2-Beauport links. The main landfall of the hurricane was assumed to occur near Charleston. Columbia was the location from which state disaster management personnel were organized. Table 1 shows the time line for the scenario that defined the dynamic context at hourly increments based on the phases of disaster response. The context signifies the highest priority activity that was occurring, even though other activities would also be occurring.

The Charleston#2-Beauport link was assumed to be heavily affected by the disaster. It was assumed that a 50% loss of capacity was experienced for several hours, as actually happened from a downed microwave tower for Hurricane Hugo [6]. The Charleston#1-Charles-ton#2 link was somewhat degraded (20% loss of capacity), and the other link experienced no loss of capacity.

Table 2 shows user classes for the scenario, along with the ticket value issued to each class in each stage from Table 1. Figure 6 shows the ticket server session arrival rates generated by each class at each hourly interval. Note that the loads for general public users are much larger than those for most of the other specific user types. Also, load from the general public takes longer to grow and peaks later.

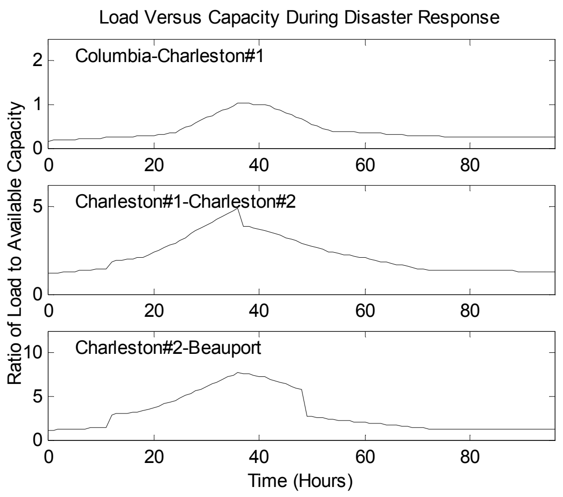

Figure 7 - Overloads Per Link For Hurricane Scenario

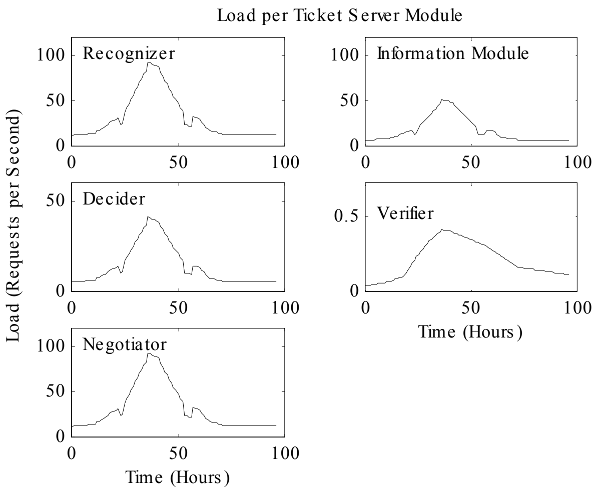

Figure 8. Loads Per Ticket Server Module

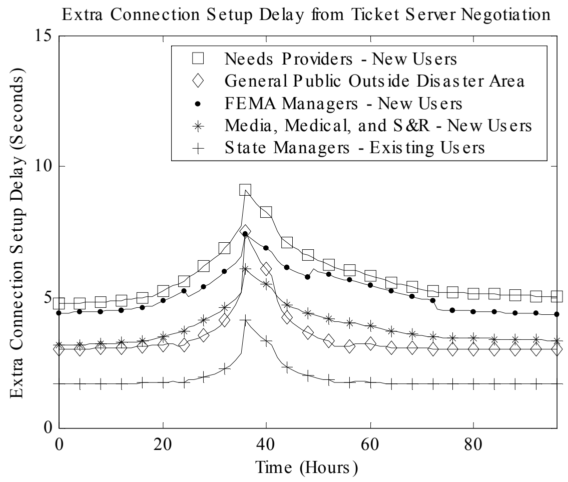

Figure 9. User Subclasses With Longest Delays For Hurricane Scenario

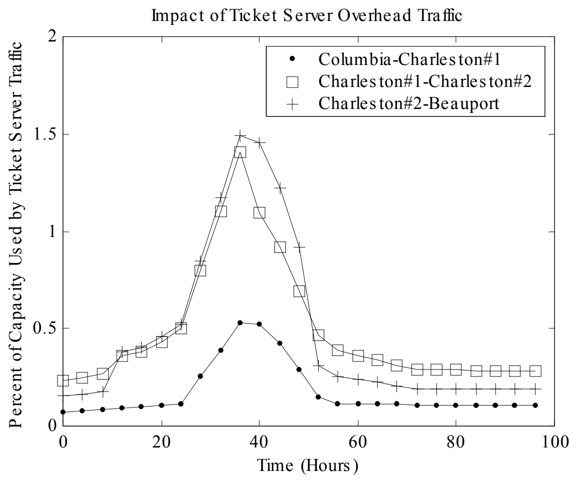

Figure 10. Network Overhead For Ticket Server Traffic In Hurricane Scenario

The loads for each of the classes reflected the time dynamics of the disaster effort for different users. Their communication traffic peaked during the stages of the disaster where their services were needed most. These load curves were not created from actual network load traces, but closely replicated typical disaster conditions as described in the literature. The traffic patterns for those in state and federal disaster management roles reflected the time dynamics of response, assessment, and coordination of people and resources to respond to the disaster.

Figures 7, 8, 9, and 10 show the results using the performance analysis approach from Sections II and III. Figure 7 shows that two of the links experienced significant load demand above available capacity, while the Columbia-

Charleston#1 link only had peak load equal to its capacity. Figure 8 shows the arrival rate in terms of the number of requests per second that were sent to each module (it was assumed for the scenario that no requests were sent to the Modeler). To have reasonable and bounded delays, all modules had to be able to process more requests per second than their peak load. The requirements on the Decider module appeared to be the most stringent. Even though it needed to process fewer messages per second than other modules, its function was also more complex. Also for the Verifier module, the requirement of up to 0.5 requests per second might also be difficult to meet, because high variability might occur because of needing to contact third parties for verification information.

After choosing a set of module processing rates that were greater than these peak loads, Figure 9 was created that shows the classes with the highest peak delays for their interaction during a session with the ticket server. It was assumed that network delays (D_{net},) and security delays (D_{sec}) were both 20 milliseconds. The classes that are shown are predominantly those which required verification as part of their ticket server session. The worst case delay was 9.1 seconds; a delay less than 10 seconds is reasonable. Although most users on telephone systems will give up before they have to wait 10 seconds, here it is assumed that users are willing to wait because the ticket server notifies them at the beginning of the session about the possible need to wait.

Finally, Figure 10 shows the percentage of the capacity of each link that was occupied by ticket server overhead traffic. This shows the maximum overhead that could occur, because it was assumed that users acquired tickets for each link they needed to use, sometimes needing 2 or more tickets. It also assumed that users were not co-located with the ticket server, so all requests had to traverse network links and create overhead traffic. Each message was assumed to be 1000 bytes long. Overhead levels never exceeded 1.5% of capacity even when network links were heavily overloaded.

V. OFFICE BUILDING BOMBING SCENARIO

A second type of scenario was designed to consider a bombing to an office building where damage was localized and network traffic was concentrated into that localized area. Recent events similar to this in the United States were the Alfred P. Murrah Federal Building Bombing [9] in Oklahoma City in 1995 and the terrorist bombings in New York City on September 11, 2001. In these events, the damage occurred in an isolated area and the destruction was man-made. This scenario was created to emulate the Oklahoma City Bombing event. Many aspects of this scenario were similar to the hurricane scenario as follows.

- Many new users to the area.

- High load from outside into the area.
- Periods of high overload. For the overall city of Oklahoma City, call volumes were as high as three times normal ([9], p. 359)..

Significant differences existed, however, as follows.

- Security – Since the disaster was man-made, there would be higher network traffic related to security. It would also be more important to issue tickets securely.
- Differences in viewpoint between users and ticket servers – Ticket servers may not be immediately aware of the occurrence of the event, so users close to the damage area could provide information for the servers to update their models. Users away from the disaster area also might not be aware of the event's occurrence and the reason for reduced availability of network resources.
- Geographic concentration – The event is localized, rather than spread over an area as with a hurricane. Traffic is therefore focused on a single location.
- No damage to the communication system – Other than potential damage to facilities within the building itself, communication systems are unharmed and capacity is not degraded.
- Different relief phases – The timing of disaster relief would still involve search & rescue and medical phases, but would not have a phase to provide for physical needs (i.e., food and shelter), since only a few people are affected. The search and rescue phase might also last longer, due to the complexity of finding victims in the midst of a weakened building structure.
- More users using a "Keep Trying" negotiation approach – Since the event was not predicted, users would be much more insistent in trying to obtain network resources, either because they are seeking information about the event or because they are not aware an event has occurred. In the Oklahoma City Bombing case, out-of-town media personnel kept calling the Oklahoma City 9-1-1 center for statements and demanded to stay on the line until they could get information ([9], p. 356).

Figure 11 shows the example communications network that was considered. The same types of users were involved as the hurricane scenario, but with a different time line and network loading pattern were assumed as shown in Figure 12. Once again, these load curves were not created from actual network load traces, but were derived from the detailed report

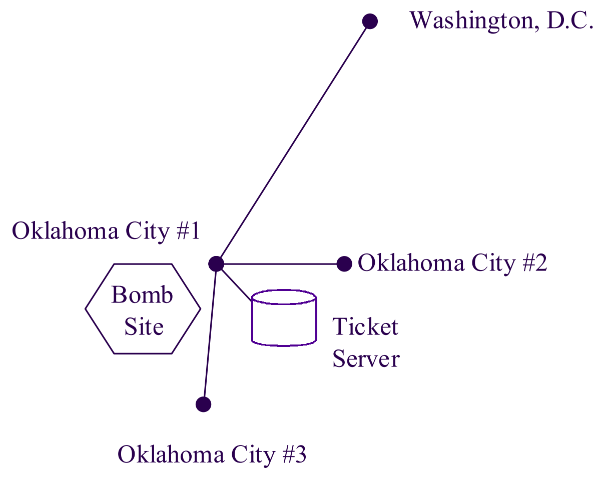

Figure 11. Network For Office Building Scenario

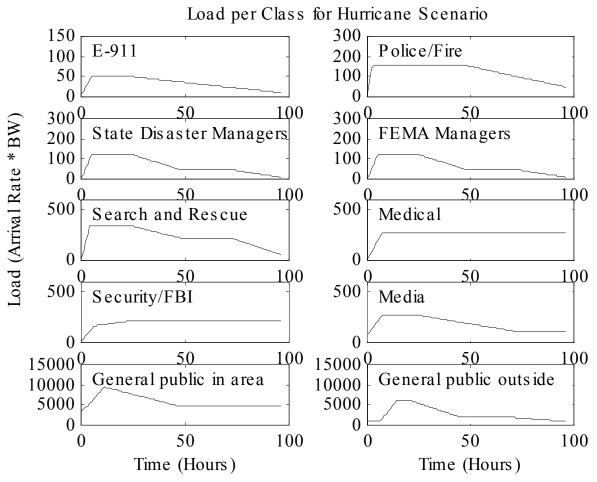

Figure 12. Class Loads For Office Building Scenario

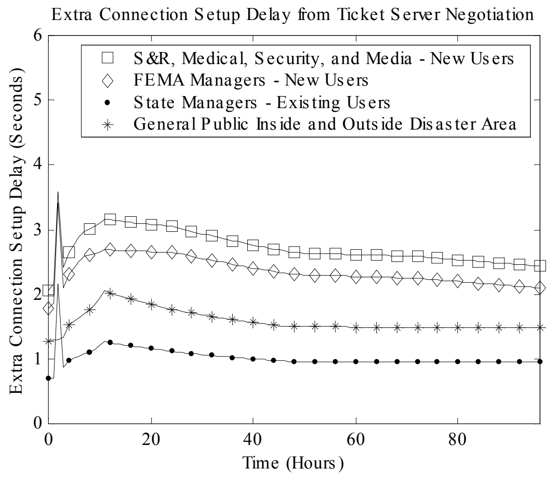

Figure 13. User Subclasses With Longest Delays For Office Building Scenario

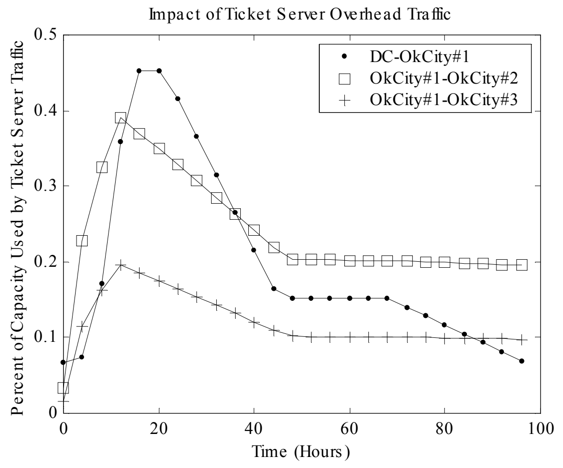

Figure 14. Network Overhead For Ticket Server Traffic For Office Building Scenario

on the Oklahoma City Bombing [9]. In the hurricane scenario, load grew somewhat gradually in anticipation of the event. In this case, load jumped suddenly.

Figures 13 and 14 are the similar to Figures 9 and 10. Figure 13 shows that in this case, the user class with the highest peak delay had a delay of 3.5 seconds. Figure 14 shows that network overhead peaked at 0.45% of capacity.

Although results for both of these scenarios were not based on actual network traces, the results can be treated as a reliable assessment of the viability of ticket servers for three reasons.

1. The results are upper bounds on the actual performance that would occur. Tickets were assumed to be issued individually for every link in a connection.
2. Link loading was consistent with typical disaster response conditions. Loading on links at full capacity can reach up to five times normal levels after an event [10]. If link capacity is degraded, load levels would be even higher.
3. The time line of each scenario closely replicated that of typical disaster response conditions.

VI. CONCLUSION

The purpose of this paper was to assess the viability of implementing an architecture of ticket servers for issuing tickets to requests for important connections. Instead of using a full discrete event simulation approach, a hybrid prototype/analysis approach was used. A prototype was developed to create user and server intelligent agent processes to carry out negotiation sessions for tickets. Simulation of the prototype consisted of observing the total number of requests per session that were sent to each ticket server module for particular profiles. To find performance results for server processing requirements, negotiation session delays, and network overhead, analytical M/M/1 queueing network results were used to combine the outputs from the prototype with a disaster network loading timeline.

Ultimately, these results are useful in two ways. First of all, a useful approach was applied to a problem that involved negotiation and intelligent agents. Discrete event simulation need not be performed to simulate message traffic. Performance metrics can be obtained by a combination of system prototyping and analysis of networks of queues, by looking at arrivals on a session basis instead of on a per message basis.

Finally and most importantly, however, this simulation study showed that ticket servers could be effectively implemented. Server processing requirements, connection setup delays, and network overhead levels were not prohibitive. By implementing such an architecture, broadband networks can be used to more effectively meet the needs of society in response to disaster conditions.

REFERENCES

[1] B. Brewin, Nation's Networks See Sharp Volume Spikes After Attacks, *Computerworld*, September 17, 2001.

[2] E. Noam, Testing the Communications Network, *The New York Times*, September 24, 2001.

[3] The President's National Security Telecommunications Advisory Committee, Network Group, *Internet Report: An Examination of the NS/EP Implications of Internet Technologies*, June 1999.

[4] National Communications System, *Fiscal Year 2000 Annual Report: Exploring Solutions for Communications Reliability*, 2000.

[5] Federal Emergency Management Agency, Technology Applications by the Federal Emergency Management Agency in Response, Recovery, and Mitigation Operations, *Paper Presented to the 27th Joint Meeting of the U.S./Japanese Panel on Wind and Seismic Effects*, Tokyo/Osaka, Japan, May 16-27, 1995.

[6] C. Wilson and A. Lindstrom, Survival of the network: The single greatest disaster in U.S. history couldn't take down telephone service, *Telephony*, October 23, 1989.

[7] J. S. Griswold, T. L. Lightle, and J. G. Lovelady, Hurricane Hugo: Effect on State Government Communications, *IEEE Communications Magazine*, June, 1990, pp. 12-17.

[8] A. Taylor, Andrew is Brutal Blow for Agency, *Congressional Quarterly Weekly Report*, Sept. 12, 1992, p. 2703.

[9] The City of Oklahoma City, *Alfred P. Murrah Federal Building Bombing: Final Report*, Fire Protection Publications: Stillwater, Oklahoma, 1996.

[10] S. Adamson and S. Gordon, Analysis of Two Trunk Congestion Relief Schemes, *Proceedings of IEEE MILCOM '93*, pp. 902-906, 1993.

[11] G. Philip and R. Hodge, Disaster Area Architecture, *Proceedings of IEEE MILCOM*, 1995, pp. 833-837.

[12] Computer Science and Telecommunications Board, National Research Council, *Computing and Communications in the Extreme: Research for Crisis Management and Other Applications*, 1996.

[13] Description of an International Emergency Preference Scheme (IEPS), International Telecommunication Union (ITU), Geneva, Switzerland, *ITU-T Recommendation E.106*, March 2000.

[14] Draft ITU-T Recommendations F.706 - International Emergency Multimedia Service (IEMS).

[15] C. Beard and V. Frost, Prioritized Resource Allocation for Stressed Networks, *IEEE/ACM Transactions on Networking*, Vol. 6, no. 5, October 2001, pp. 618-633.

[16] Cory C. Beard and Victor S. Frost, "Dynamic Agent-Based Prioritized Connection Admission for Stressed Networks," *Proceedings of the 1999 IEEE International Conference on Communications*, Vancouver, Canada, June 1999.

[17] http://java.stanford.edu.

[18] The DARPA Knowledge Sharing Initiative External Interfaces Working Group, "DRAFT Specification of the KQML Agent-Communication Language, plus example agent policies and architectures," June, 1993.

[19] http://herzberg1.ca.sandia.gov/jess.

[20] C. Beard, *Dynamic Agent Based Prioritized Resource Allocation for Stressed Networks*, Doctoral Dissertation, University of Kansas, 1999.

[21] Clinton Administration, "The Effect of the National Information Infrastructure (NII) on Local, State, and Federal Emergency Management," September 7, 1994.

[22] Network Reliability Council, Focus Group 5, "Network Reliability -- The Path Forward: Telecommuting as a Back-Up In Emergencies", February 1996, Section 5.

[23] Federal Emergency Management Agency, *Federal Response Plan, Emergency Support Function #9 – Urban Search and Rescue Annex*, April 1999.