Evaluation of Architectures for Reliable Server Pooling in Wired and Wireless Environments

M. Ümit Uyar, *Senior Member, IEEE*, Jianliang Zheng, Mariusz A. Fecko, *Member, IEEE*, Sunil Samtani, and Phillip T. Conrad, *Member, IEEE*

***Abstract*—The reliable server pooling (RSP) allows a pool of redundant information sources to be viewed as a single transport endpoint and, therefore, is able to provide persistent connections and balanced traffic for different applications. The Internet Engineering Task Force RSerPool Working Group has proposed an architecture to implement the RSP, which defines an overlay network providing an upper layer protocol or an application with a range of reliability services, from simple server selection to a fully automatic session-failover capability. The simulation experiments conducted in both wired and wireless environments show that the current version of the RSerPool works well in fixed and relatively reliable environments, but its performance worsens rapidly as the networks become more unreliable or mobile. The issues we identified in wireless mobile ad hoc networks include network partition, high signaling overhead, difficulty in synchronization among name servers, and excessive aggressiveness in handling failures. Alternative design options for the RSP in wireless and mobile environments are introduced and evaluated.**

***Index Terms*—Ad hoc networks, battlefield networks, failover, reliable server pooling (RSP), RSerPool, service overlay networks.**

I. Introduction

THE *reliable server pooling* (RSP) is an overlay network designed to provide an upper-layer protocol or an application with a range of reliability services, from simple server selection to a fully automatic session-failover capability. It increases the system's reliability and availability by handling session failures and pooling redundant information sources into a single transport-layer endpoint. These features imply that, in case of session failures, most applications can be transparently switched to another server without restart.

The span of overlay networks includes improving network-layer connectivity [11], choosing the optimum number and location of servers [19], and enhancing the performance of delivering HTTP data and streaming video via content delivery networks [15]. Resilient overlay networks (RON) [2] are among the few overlay networks designed for failure detection and recovery. Similar to the RSP, they seek to alleviate disruptions in end-to-end communications in the midst of failures. However, RONs focus is limited to reacting to path outages and periods of degraded performance to route packets either directly over the Internet or through RON nodes.

Due to a need for reliable and persistent connections, IETF has developed several related approaches including stream control transmission protocol (SCTP) (RFC2960) [22] and RSerPool (RFC3237) [26]. The latter is a particular approach to the RSP currently being defined by the RSerPool Working Group (WG), and consists of two complementary protocols: endpoint name resolution protocol (ENRP) [31] and aggregate server access protocol (ASAP) [23].

To evaluate possible designs of the RSP in different environments, this paper investigates if the RSerPool can provide sufficient performance in wired and wireless networks, especially for mission-critical applications such as disaster recovery and battlefield communications. As part of this research [28], [29], an NS-2 [27] simulation testbed for the RSerPool has been implemented. A set of simulation experiments were run to obtain metrics for different aspects of the framework for wired and wireless environments. Our simulation results show that the current version of the RSerPool performs well in wired environments, but its performance worsens rapidly as the networks become highly unreliable or mobile. The results identify problems in wireless mobile ad hoc networks as network partition, high overhead, difficulty in synchronization among name servers, and too aggressive fault handling.

Based on the experiments, two of the main deficiencies of the RSerPool in wireless mobile networks are found to be the inaccuracy of the failure-detection mechanism for servers within a pool, and the high overhead of finding the home name server. To overcome these problems, we introduce alternative mechanisms and present their numerical evaluation.

Section II provides a background for the RSP and related technologies. Section III gives an overview of the Internet Engineering Task Force (IETF) RSerPool WG architecture. Experiment metrics are defined in Section IV. Section V describes the NS-2 simulation testbed. Simulation results are presented in Section VI. Guidelines for designing the RSP in wireless and mobile environments are introduced in Section VII.

II. Motivation and Background

Persistent connections to servers can become broken due to severe network stress, server or link failures, constant mobility in wireless mobile networks, or, for military applications, loss of assets in a battle. The traditional abort-and-restart approach

Manuscript received November 15, 2002; revised June 2, 2003. This work was prepared through collaborative participation in the Communications and Networks Consortium supported by the U.S. Army Research Lab under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011.

M. U. Uyar and J. Zheng are with the Electrical Engineering Department, City College, City University of New York, New York, NY 10031 USA (e-mail: uyar@ccny.cuny.edu; umit@eesls0.engr.ccny.cuny.edu).

M. A. Fecko and S. Samtani are with Applied Research, Telcordia Technologies, Inc., Piscataway, NJ 08854 USA.

P. T. Conrad is with the Computer and Information Science Department, University of Delaware, Newark, DE 19716 USA.

Digital Object Identifier 10.1109/JSAC.2003.818806

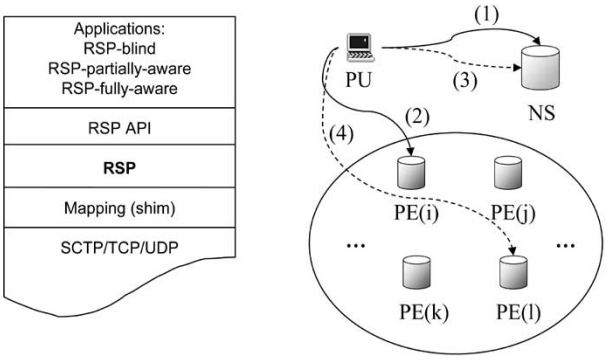


Fig. 1. Reliable server pooling: protocol stack and basic operations.

often results in long delay and is insufficient for applications such as military communications, real-time transactions and videoconferencing, distance diagnosis, and disaster recovery.

Different approaches have been reported in the literature to improve the server reliability and enhance the system availability [1], [2], [11], [14], [24], [26], [32]. Since most focus on the Internet [2], [11], [32] or the telephony applications [26], it is unclear how they perform in wireless mobile networks. Among these approaches, the RSP is the most promising to meet the timeliness requirements of persistent connections.

A. Reliable Server Pooling

An important characteristic of RSP is its name-based addressing model that isolates a logical communication endpoint from its Internet Protocol (IP) address(es), thus making it possible to switch an end-to-end session from a failed or overloaded server to an alternative one, without reestablishing the session. Servers with the same application functionality are grouped into server pools. A client can access a server pool by consulting a name server (NS). A server in a pool is called a pool element (PE); a client being served by a PE is called a pool user (PU).

An example RSP operation is shown in Fig. 1. In step (1), the PU contacts its home NS to query for transport layer addresses of PEs in a pool it wants to access. Then, in step (2), the PU selects a PE using a server-selection algorithm based on predefined selection policies. First, PE_i is selected and an end-to-end session established between the PU and PE_i. If PE_i fails or is overloaded during the session, the PU can switch the session to another PE. To switch over, the PU contacts its home NS to obtain a fresh PE-list for the pool, and selects one of them [steps (3) and (4)].

B. Scope of RSerPool

The IETF RSerPool WG architecture is a lightweight, best-effort approach to the RSP since it lacks quality-of-service (QoS) features and connection-admission control. The aim of the RSerPool is to provide an open standard, with signaling protocols that do not entail layering violations, i.e., they preserve the end-to-end model of IP. The RSerPool, with its loosely coupled architecture, does not set a complete fault-tolerant computing as its goal. As a result, server pools may span several network domains, and the pool services are likely to survive localized disasters; a similar statement may not be true in the case of server farms or clusters that share a local network.

The above advantages and limitations determine the current scope of the RSerPool. It should be viewed as a distributed, open platform that supports various reliability services to the applications. For example, the RSerPool allows plugging in various server-selection policies, which may be a simple round-robin or least-recently used algorithm (used in our study), as well as more sophisticated ones based on load balancing. (Advanced server-selection techniques are reported in several studies [9], [13], [32].)

The transparent session migration (and the related issue of keeping data integrity) is an open research problem beyond the scope of this paper. General techniques requiring special modifications to a transport protocol or software wrappers around it, are described in [1] and [24]. For simple data access transparency among PEs, one can consider data-replication techniques for partitionable ad hoc networks [6], [14], or a file synchronization protocol (e.g., rsync [25]). Instead, we assume a failover capability for simple applications (e.g., file transfer), and focus on the core RSerPool architecture and signaling protocols. The ongoing work [8] on the transparent failover within the IETF is outlined in Section III-A.

C. Simple Server Replication

Simple server replication is a basic paradigm to increase the network reliability and efficiency. It is typically provided only as a means of backup such that a content-equivalent server (e.g., a mirrored web server) takes place of a failed server. In most approaches, sessions originally served by the failed server are lost. Also, server replication only focuses on server failures and cannot handle other failures such as those of routers or links. As a result, simple server replication by itself cannot guarantee highly available services [24]. Table I details the significant differences between the concepts of simple server replication and the RSP.

D. Application-Layer Anycasting

Application-layer anycasting [32] has been applied to server selection in a widely replicated Web service. Similar to the RSP, it offers flexibility in the server-selection criteria, whereas in another anycasting paradigm—network-layer anycasting—server selection is typically determined by a routing protocol (e.g., through hop-count). Nevertheless, there are a number of differences between the RSP and application-layer anycasting:

- The hierarchy of resolvers (and the caching of resolution results) imply that the anycasting offers good network scalability, but at the expense of load balancing. The RSP is better suited to load balancing at the time of server selection.
- Unlike in the RSP, no end-to-end communication occurs between the host and its name server due to recursive resolvers. This feature makes it difficult for anycasting to meet the timeliness requirements of the RSerPool.
- In the RSP, interoperability between different namespaces has to be provided by a mechanism different from that of recursive resolvers.

TABLE I
COMPARISON OF SERVER REPLICATION AND RELIABLE SERVER POOLING

	Server Replication	**Reliable Server Pooling**
Server Group	Set up statically.	Set up and managed dynamically.
Server Access	Uses IP address or (DNS) domain name.	Uses pool handle.
Single Point of Failure	Yes.	No. Multi-path, multi-home, and multi-server (server pool) can be exploited to eliminate single points of failure.
Failover	No. A failed session needs to be restarted with a backup server.	Yes. Session state can be shared or recreated among servers to support failovers.
Load Balancing	No. It cannot switch a session to another server even if the current server is overloaded. End users are required to switch to a new server by themselves.	Yes. It can support various server selection and switchover policies, and trigger a switchover due to connection failures or system performance degradations.
Name Server	DNSs need to be set up manually, and only provide static name resolution service.	NSs can be automatically configured. They provide dynamic and accurate name resolution service.

- An anycasting resolver returns a list of IP addresses, whereas a PU receives a list of transport addresses (i.e., IP addresses plus port numbers) from an RSerPool NS. As a result, the RSP can support multiple PEs on one IP address.

E. Other Related Technologies

The domain name service (DNS) (RFC1034, RFC1035) provides a means of resolving a name to an IP address or set of IP addresses, similar to one of the key functions of the NS in the RSerPool. However, the caching effects of the current DNS architecture are a key stumbling block in using DNS as a solution for the RSP. For the DNS to meet the timeliness requirements of RSerPool, the time-to-live (TTL) values in the DNS records would have to be reduced by four orders of magnitude or more (from hours to seconds), dramatically increasing the load on DNS servers. Caching also interferes with the ability to do load balancing. DNS can implement a form of crude load balancing by cycling through multiple IP addresses for a single domain name. However, the DNS server cannot predict how many subsequent requests will be directed to a particular server while that name-to-address mapping is cached at a particular local DNS server. Finally, DNS servers do not currently monitor the status of the hosts for which they provide name-to-address mapping, and hence in the event of a server failure, the DNS server may continue to respond with the IP addresses of failed servers [20].

The service location protocol (SLP) (RFC2608) [12] provides a means of mapping a request for a service to the location of relevant servers (through URLs). The main difference is that SLP is service-oriented while RSerPool is communication-oriented [20]. Unlike the RSerPool, SLP does not provide for the transfer of data, or for monitoring the reachability and availability of the service providers. SLP is essentially a service-discovery protocol operating at an application level.

Hull [15] surveys proprietary systems for web-server farms. Such solutions typically provide access to a number of back-end servers through a single virtual IP address. A front-end device acts as an intermediary. All communications between clients and the server farm usually pass through this device, which inspects or modifies headers at OSI layers 2, 3, 4, and 7. Some vendors provide solutions for failover of the front-end device, as well as for load balancing and fault localization between sites [3]. These solutions typically are targeted for Web applications, and associated media-streaming protocols, which requires vendor-specific hardware. For example, to utilize Cisco Distributed Director [7], a vendor router has to be deployed in the proximity of each replicated server. In contrast to the RSerPool, the layering violations imply that the end-to-end model of IP no longer holds.

III. RSERPOOL ARCHITECTURE

Recall from Section II-A that there are three classes of entities in the RSP (Fig. 1): PEs, NSs, and PUs. In the IETF RSerPool, NSs play a critical role because they manage and maintain the entire server-pool namespace. Each NS communicates with its peers through a special channel and shares the same view of the namespace. All NSs have the same functionality, while taking care of a different group of PEs and PUs. An NS is said to be the home NS of those PEs and PUs that are under its supervision. Any entity, except the NSs, must find a home NS to whom all its requests for the namespace access must be sent. A server can register into a pool to become a PE of the pool by consulting its home NS. Similarly, a PE can leave a pool through deregistration. A server may join multiple pools simultaneously. To access a pool, a PU must contact its home NS to query for transport layer addresses of PEs in that pool. This procedure is transparent to upper layer applications.

The IETF RSerPool WG architecture consists of two complementary protocols, namely, endpoint name resolution protocol (ENRP) [31] and aggregate server access protocol (ASAP) [23], as outlined below.

- *ENRP:* It defines a registry service for distributing a pool's operation and membership information. An ENRP server uses PEER_PRESENCE message to locate other ENRP servers during initialization. (The terms ENRP server and NS are used interchangeably.) The PEER_PRESENCE is also used as a heartbeat message that is periodically sent by an NS to inform its peers of its active status. When a pool membership changes, an NS sends out a PEER_NAME_UPDATE message to inform its peers.
- *ASAP:* It monitors the reachability of the PEs in a server pool and has the ability to automatically switch an association from one PE to another. A server can join/leave a pool by re-/de-registering through its home NS. ENDPOINT_KEEP_ALIVE is used to determine a PE's health status; ENDPOINT_UNREACHABLE is sent by a PU to its home NS to indicate that it has problems to reach a

certain PE. A PE or PU has to find a home NS by sending SERVER_HUNT to the ENRP client channel before it can access any RSP services.

A. Transparent Session Migration

A desirable feature of any RSP framework is transparent session migration (or transparent switchover). In general, transparent switchover means that once a PU establishes a session with a PE from the pool and if that PE becomes unavailable, the session will be transferred to another PE without a PU's intervention. Typically, transparent switchover requires that the consistent state be shared among PEs, as in the context of distributed database systems [21], [30].

Application requirements for the consistency of shared state vary. For example, a server pool for a financial application may require strict consistency with three-phase commit operations. On the other hand, a pool that provides read-only access to rarely updated web pages may have looser requirements.

While the RSerPool cannot guarantee transport-layer switchover, it provides some support for application-layer switchover [8]. This support is described below and illustrated for a server PE_1 that fails during a session, and server PE_2 to which that client is redirected.

1) Failover Callback: This service allows a PU's client to register an application-specific callback function that is invoked automatically by the PU whenever a failover occurs. The callback function can be used by the application to send messages to PE_2 to reestablish the session on the new PE. The application is, thus, relieved of having to *detect* failures.

As an example, consider a one-way file-transfer application (all transfers are from servers to clients).

- PEs contain the same files, with changes occurring every several minutes. A file synchronization protocol (e.g., rsync [25]) keeps the inconsistencies between PEs transient.
- The client initially sends PE_1 a request packet consisting of a filename and an offset of zero.
- The server initially responds with the file size, last modification time, and an MD5 hash, followed by packets containing bytes from the file, each with an offset.
- Upon failover, the client sends PE_2 a request with the filename and offset of the least byte in the file that has not yet been received, and last modification time, and MD5 hash. If the file on the server is a match, the server picks up where the file transfer left off. If not, the session is aborted.

In this scenario, the client can simply request the file, and then read packets from the session, updating a variable with the total bytes read, and the least byte not yet read, until the entire file is transferred. Except in the rare case of an inconsistency between PE_1 and PE_2, any failover is handled gracefully by the failover callback function. The RSerPool takes care of the details of detecting that PE_1 has failed, and establishing a new session with a suitable PE_2.

2) Application-Level Ack/Retransmission: This service provides, upon failover from PE_1 to PE_2, the retransmission of messages that were not both received *and processed* by the receiving server application on PE_1. Each time a message is sent from the client to the server, that message is buffered by the RSerPool layer on the PU until a so-called *application layer ack* (ALA) is returned by PE_1. If this service is used, the server application has the responsibility to process each message read from the session (including updating the shared server state by some application-specific means), and then signal the RSerPool layer on PE_1 to send the ALA for the message back to the client. Upon failover from PE_1 to PE_2, the failover callback function is first called. Then, all messages for which an ALA is outstanding are transmitted to PE_2.

As an example, consider a situation-awareness application, where the servers aggregate reports from multiple clients to update a shared distributed database. When the server receives a report, it will signal the RSerPool on PE_1 to return an ALA for that message only after the distributed database has been successfully updated with the information in that report, and the changes have been committed. If the client does not receive the ALA, it cannot be assured that the report was committed to the shared database; hence, the report will be resent after the PU fails over to PE_2.

IV. Quantifying Reliable Server Pooling

A. Performance Metrics

We define several performance metrics to evaluate basic RSerPool operations.

- *Number of home-hunt attempts per PE/PU per time unit:* This metric reflects the home hunt efficiency.
- *Percentage of successful transmissions among NSs:* It measures the synchronization ability among NSs.
- *Number of switchovers per PU per time unit during an application session:* It shows the RSerPool effectiveness in increasing the reliability of an application session.
- *Ratio of RSerPool messages to data messages:* This is the ratio of total the RSerPool control messages sent to the data messages successfully received.
- *Latency of home hunt:* It is the period from the time a PE/PU begins to hunt for a home until it receives the first response from one of the NSs.
- *Ratio of unnecessary PE deregistrations to total PEs in the RSerPool namespace:* It measures the unnecessary deregistrations of operational PEs from a pool.

B. Session Switchover Metrics

A *successful switchover* is defined as the establishment of a session at the desired QoS to a PE other than the original PE of the same pool, which may be preceded by several unsuccessful attempts to contact different PEs within a pool.

Failed sessions, which are the ones that experience at least one failure, can be divided into several categories:

- *sustained session:* one that was successfully switched over each time it failed; it runs until completion through (multiple) switchovers within a pool;
- *lost session:* one that experienced an unsuccessful switchover; this category also covers sessions that were able to switchover a number of times before some failure resulted in an unsuccessful switchover;

The metrics for switchover evaluation are:

- *session sustainability throughput (SST):* the ratio of the number of sustained sessions to the failed sessions;
- *switchover efficiency (SE):* the average number of switchover attempts per successful switchover.
- *session sustainability gain (SSG):* the increase in the system's ability to admit and run a session until completion.

SSG quantifies the extra overhead that switching over existing sessions will put on both the network and the RSP infrastructure. As a result, the number of new sessions that can be successfully opened may be smaller than in the absence of the RSP. Suppose that there are E currently open sessions and N_o new session attempts. (Note that a session undergoing a series of switchovers is considered one session.) Of N_o, R_o sessions are unable to be established. The total number of open sessions in the experiment is, thus, $T_d = E + N_o - R_o$.

T_d represents the number of sessions opened due to the total external stimulus, $D = E+N_o$. Let F be the number of sessions of T_d that subsequently fail, out of which S are sustained, I are inconclusive, and L are lost: $F = S + I + L$. The values of E and N_o will be the same for networks with and without the RSP since they represent the external stimuli.

The values of other parameters are shown in Table II, for the cases with $(^p)$ and without $(^i)$ the RSP. Let w_l and w_r be the weights for a lost and a rejected session, respectively. Typically $w_l > w_r$, meaning that sustaining an existing session is more desirable than admitting a new one.

Of D sessions, let Y be the weighted percentage of those rejected or lost. Then, Y measures the degree of a system's inability to admit and run a session until completion

$$Y^p = \frac{w_l * F_d^p + w_r * R_o^p - w_l * S_d^p}{D} \tag{1}$$

$$Y^i = \frac{w_l * F_d^i + w_r * R_o^i}{D} \tag{2}$$

$$\mathrm{SSG} = \frac{Y^i - Y^p}{Y^i} = \frac{\overbrace{w_l * \left(F_d^i - F_d^p\right)}^{\text{loss}} + \overbrace{w_r * \left(R_o^i - R_o^p\right)}^{\text{loss}} + \overbrace{w_l * S_d^p}^{\text{gain}}}{w_l * F_d^i + w_r * R_o^i} \tag{3}$$

$$\mathrm{SSG}_{\max} = \frac{F_d^i}{F_d^i + R_o^i}. \tag{4}$$

In a simplified case, where all weights are the same, SSG is maximized for $S_d^p = F_d^p = F_d^i$ and $R_o^i = R_o^p$ (4). Even when we sustain all failed sessions, the comprehensive benefit decreases when the system is overloaded and rejects new session requests. As the ratio of the failed sessions to the rejected ones increases, so does the gain, with $\mathrm{SSG}_{\max} \sim 1$ when $R_o^i \ll F_d^i$.

SSG^* is a variation of SSG based on a session's partial success. It can be calculated analogous to SSG. For each set of sessions, instead of using the number of sessions in the set, the metric should use the cumulative time from a failure to the completion or loss. The values of S_d, F_d, and R_o in (1) and (2) are replaced with S_d^*, F_d^*, and R_o^*

TABLE II
SWITCHOVER PARAMETERS

number of:	RSP	no RSP
open sessions at time zero	$E^p = E$	$E^i = E$
original session attempts	$N_o^p = N_o$	$N_o^i = N_o$
attempts rejected out of N_o	R_o^p	R_o^i
failed sessions out of T_d	F_d^p	F_d^i
inconclusive sessions out of F_d	I_d^p	$I_d^i = 0$
sustained sessions out of F_d	S_d^p	$S_d^i = 0$

$$S_d^* = \sum_{j=1}^{D} \left\{ \begin{array}{ll} t_c(j) - t_f(j) & \text{if } j \text{ is sustained} \\ t_l(j) - t_f(j) & \text{if } j \text{ is lost} \end{array} \right\} \tag{5}$$

$$F_d^* = \sum_{j=1}^{D} t_c(j) - t_f(j) \quad \text{and} \quad R_d^* = \sum_{j=1}^{D} t_c(j) \tag{6}$$

where session j is supposed to complete in $t_c(j)$ time, and $t_f(j)$ and $t_l(j)$ are the amounts of time to the first failure and the loss, respectively. SSG^* aggregates the relative time by which failed sessions are prolonged. For simplicity, we consider the case where $F_d^{p*} = F_d^{i*}, R_o^{i*} = R_o^{p*}, R_o^{i*} \ll F_d^{i*}$, and $w_l = w_r$. When all sessions are lost at the first failure, SSG^* is 0; it is 1 when all sessions are sustained.

C. Server Resource Availability

A PE may not be capable of accepting additional sessions due to limited resources, for example, a battery life in wireless devices or a central processing unit (CPU) processing power in wired environments. Since PEs are RSP-aware, they can advertise their current session capacity rather than have network monitors measure the load on the PEs' ports. This capacity is represented as a *degree of readiness* (DR), defined as the number of associations that a PE is willing to sustain at a certain level of quality. A PE may belong to different pools at varying DRs.

The PE capacity to accept different sessions is expressed by its capability set, which is a sequence of three-tuples $\langle \mathrm{ST}, \mathrm{SP}, \mathrm{DR} \rangle$, where ST is the server type, SP is the server priority, and DR is the degree of readiness. For example, advertisement $\{\langle \mathrm{SIP}, 1, 1000 \rangle \langle \mathrm{HTTP}, 2, 500 \rangle\}$ implies that the PE can serve as a SIP server with $\mathrm{DR}_{\mathrm{SIP}} = 1\,000$ associations, and an HTTP server with a $\mathrm{DR}_{\mathrm{HTTP}} = 500$ associations, with the SIP service given priority over the HTTP service. DR is, thus, a dynamic parameter that can change depending on varying server characteristics.

In our simulation experiments, the DR is statistically varied between zero and four to model the impact of limited server resources on the overall reliability metrics.

V. NS-2 SIMULATION TESTBED

The NS-2 simulation testbed developed at the City College of New York comprises seven functional modules, as shown in Fig. 2. The wired/wireless scenarios definition module is a Tcl script interface; the other six modules are written in C++. Each module is briefly described below.

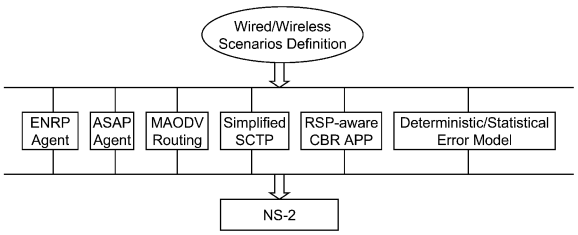


Fig. 2. NS-2 simulation testbed.

- *Wired/Wireless Scenarios Definition:* It selects the unicast/multicast routing protocols; defines the (initial) network topology; and schedules events such as initializations of NSs and PEs/PUs, registration of PEs, and starting (stopping) applications: 1) for wired networks, it defines link bandwidth, propagation delay, link queue, and traffic pattern and 2) for wireless mobile ad hoc networks, it defines mobility pattern, wireless channel, radio-propagation model, antenna model, interface queue, traffic pattern, and radio transmission range.
- *ENRP Agent:* It handles both ENRP and ASAP messages. An ENRP agent can unicast or multicast to its peers, but only unicast to ASAP agents. The module enables an ENRP agent to initialize itself and begin to serve as an NS in the server-pool namespace, acquire a peer list from a peer, download server-pool namespace data from a peer, handle PE registration, provide name resolution for a pool handle, update server-pool namespace, detect and remove unreachable PE from pool(s), and help PEs/PUs to discover home NSs.
- *ASAP Agent:* An ASAP agent gets name-resolution service from ENRP agents and provides an interface to upper-layer applications. It supports server hunt for PEs/PUs, PE registration, pool-handle resolution, endpoint keep-alive, reporting unreachable PEs, PE selection, and switchover.
- *MAODV Routing:* Multicast ad hoc on-demand distance vector protocol [18] is integrated into the testbed to support multicast and unicast in ad hoc networks.
- *Simplified SCTP:* It contains a subset of SCTP [22] closely related with the RSP. It supports connection-oriented sessions such as acknowledgment, retransmission, failure detection and notification, as well as a configurable receiving window for congestion control.
- *RSP-aware CBR APP:* It defines an RSP-aware constant bit rate (CBR) application interface, where user defines the packet size and rate.
- *Deterministic/Statistical Error Model:* A statistical error model is used to simulate link-level noise-featured errors and losses. Rare but severe failures such as cuts of links and breakdowns of servers are simulated deterministically.

A. Experimental Setup

One set of simulation experiments are devoted to wired networks, which are the main focus of the IETF RSerPool WG. The RSP plays even more important role in wireless mobile ad hoc networks, where the connections are more unreliable; yet many important applications with high-reliability requirements must operate in this environment. Another set of simulation experiments are, thus, designed to evaluate the performance of the RSerPool in such networks.

For all simulations, two server pools are constructed, each serving all PEs and half PUs. For wired networks, mesh topology is used, since other topologies (e.g., star, bus, and ring) are likely to include one or more single points of failure. All nodes are evenly distributed within a two-dimensional square plane for simplicity. All links are full-duplex with bandwidth of 2 Mb/s, a delay of 10 ms, and a random link-error model. An RSP-aware CBR model is currently used for all end-to-end application sessions.

The performance of the RSerPool in wired networks is evaluated with respect to the following parameters: 1) statistical link-error rate (0.01%) and 2) application traffic-density (10 pkts/s, 20 pkts/s, 40 pkts/s). To address the scalability problem in wired networks, four sets of scenarios are defined:

- 16 nodes (1 NS, 2 PEs, 13 PUs);
- 36 nodes (2 NSs, 5 PEs, 29 PUs);
- 49 nodes (3 NSs, 7 PEs, 39 PUs);
- 100 nodes (6 NSs, 14 PEs, 80 PUs).

For wireless mobile networks, all nodes move based on a random waypoint mobility pattern [5]. The performance of the RSerPool is evaluated for:

- transmission range [m] (100, 300, 500);
- node speed range [m/s] (0–10, 0–20, 0–30);
- pause between node movements [s] (2);
- link-error rate [%] (0.2, 0.4, 0.8);
- application traffic density [pkts/s] (10, 20, 40).

To address the scalability problem, four scenarios for each set of parameters (e.g., 300 m, 0–10 m/s, 2 s, 0.2%, 20 pkts/s), based on different number of nodes with their roaming planes, are defined as follows:

- 16 nodes (2 NSs, 2 PEs, 12 PUs in 800×800 m^2);
- 36 nodes (5 NSs, 5 PEs, 26 PUs in 1200×1200 m^2);
- 49 nodes (7 NSs, 7 PEs, 35 PUs in 1400×1400 m^2);
- 100 nodes (14 NSs, 14 PEs, 72 PUs in 2000×2000 m^2).

It has been reported [5] that, for some cases, the first 15 min of the NS-2 experiments may be unstable for the waypoint mobility model. Therefore, in our simulation experiments the results are recorded after 15 min.

VI. Simulation Results

A. Experiment Results for Wired Networks

A set of experiments for wired networks are run to evaluate the effects of the number of nodes (Table III) and traffic density (Table IV). The overall performance of the RSerPool for wired networks is satisfactory.

It can be seen from Tables III and IV that the wired networks have relatively high reliability. For example, the percentage of successful transmission among NSs is near 90% and more than 1300 data messages are delivered for each RSerPool message sent. In general, the performance of the RSerPool is quite reliable and stable when the number of nodes varies. The same is true for application traffic density, which affects the network performance mainly through use of buffers. A packet can be sent out quickly in wired networks thanks to the static routing and the

TABLE III
SCALABILITY EXPERIMENT RESULTS FOR WIRED NETWORKS

Parameters: Link Bandwidth (BW)=2Mb/s, Link Delay (D)=10ms, Link Error Rate (ErrR)=0.01%, Application Traffic Density (AppD)=20pkts/sec				
Number of Nodes (NumN)	16 nodes	36 nodes	49 nodes	100 nodes
Number of home-hunt attempts per PE/PU per second	0.002	0.002	0.002	0.002
Percentage of successful transmissions among NSs	NA	90.77%	94.15%	95.09%
Latency of home hunt (in seconds)	0.0427	0.0482	0.0523	0.0478
Ratio of unnecessary PE de-registrations to total PEs	0:4	0:10	0:14	0:28

TABLE IV
APPLICATION TRAFFIC DENSITY EXPERIMENT RESULTS FOR WIRED NETWORKS

Parameters: NumN=49, BW=2Mb/s, D=10ms, ErrR=0.01%			
AppD	10pkts/sec	20pkts/sec	40pkts/sec
Number of home-hunt attempts per PE/PU per second	0.002	0.002	0.002
Percentage of successful transmissions among NSs	94.15%	94.15%	94.21%
Ratio of RSerPool messages to data messages	1:480	1:735	1:1039
Latency of home hunt (in seconds)	0.0523	0.0523	0.0529
Ratio of unnecessary PE de-registrations to total PEs	0:14	0:14	1:14

TABLE V
SCALABILITY EXPERIMENT RESULTS FOR WIRELESS MOBILE NETWORKS

Parameters: Transmission Range (TranR)=300m, Movement Speed Range (SpdR)=0-10m/sec, Pause between Movements (PauM)=2sec, Link Error Rate (ErrR)=0.2%, Application-Traffic Density (AppD)=20pkts/sec				
Number of Nodes (NumN)	16 nodes	36 nodes	49 nodes	100 nodes
Number of home-hunt attempts per PE/PU per second	0.070	0.267	0.228	0.139
Percentage of successful transmissions among NSs	3.04%	9.83%	17.47%	6.92%
Latency of home hunt (in seconds)	116.62	192.39	114.76	131.59
Ratio of unnecessary PE deregistrations to total PEs	0:4	2:10	4:14	9:28
SE	1.00	2.00	3.22	5.11
SSG [%]	20	23	19	29
SSG^* [%]	31	38	26	31

effects of traffic densities in our simulations are rather negligible (Table IV). When the traffic density is increased to 40 pkts/s, though, there appears one PE unnecessarily deregistered by its home NS.

B. Experiment Results for Wireless Mobile Networks

Our simulation experiments indicate that the overall performance of the RSerPool in wireless mobile networks is not satisfactory. The performance of wireless mobile networks is more sensitive to network parameters such as the number of nodes, movement speed, link-error rate, and application traffic density. Of the switchover metrics defined in Section IV-B, the values of SSG and SST are practically the same due to the fact that the increase in the number of failed and rejected sessions caused by the RSerPool infrastructure is minor. Therefore, only SSG, SSG^*, and SE are provided.

Table V shows that the percentage of successful transmissions among the NSs is less than 18%, which implies that the network partition problem is serious and the NSs can hardly synchronize with one another. This is an important shortcoming since the reliability and efficiency of the RSerPool depend on the synchronization ability of the NSs.

As shown in Table VI, percentage of successful transmission among NSs increases slightly when the movement speed range is increased from 0–10 to 0–20 m/s. However, it drops about 25% when the speed range is increased from 0–20 to 0–30 m/s. A slightly higher speed may, thus, improve the performance, but if it is too high, the performance degrades due to quick expiration of the routing information (results in the peaks for some metrics at 0–20 m/s).

From Table VII, we can see the percentage of successful transmissions among NSs drops and home-hunt latency grows when link-error rate increases from 0.2% to 0.8%. However, the other metrics do not change much as the link-error rate increases because the constant mobility is the main contributor for failures in wireless mobile networks. The effect of application traffic density is not very significant, as shown in Table VIII. Nevertheless, this effect needs to be further examined, since the application traffic densities used in our simulations are relatively moderate.

As shown in Tables V–VIII, the session sustainability benefits of the RSerPool are significant, even when we assign the same weight to sustaining the existing sessions and admitting new ones. The switchovers "reclaim" up to 19%–38% of the lifetime of the failed sessions (otherwise wasted); the percentages of the sessions that run until completion thanks to the RSerPool are up to 19%–29%. The most severe drop in these metrics comes from the low transmission range and high link-error rate. Only modest percentage gains are achieved when the link-error rate grows to 0.8% (Table VII). The switchover efficiency is up to 2.67–5.11, with the number of nodes having the greatest impact on this metric. The efficiency metric does not scale well (Table V), since the drop in the efficiency is linear in the number of nodes. However, the switchover efficiency is relatively stable for the other parameters (Tables VI–VIII).

TABLE VI
MOVEMENT SPEED RANGE EXPERIMENT RESULTS FOR WIRELESS NETWORKS

Parameters: NumN=49, TranR=300m, PauM=2sec, ErrR=0.2%, AppD=20pkts/sec			
SpdR	0-10m/sec	0-20m/sec	0-30m/sec
Number of home-hunt attempts per PE/PU per second	0.228	0.171	0.227
Percentage of successful transmissions among NSs	17.47%	21.64%	16.33%
Ratio of RSerPool messages to data messages	1:5.60	1:5.61	1:6.25
Latency of home hunt (in seconds)	114.76	127.92	124.75
Ratio of unnecessary PE deregistrations to total PEs	4:14	7:14	5:14
SE	3.22	2.25	2.31
SSG [%]	19	14	21
SSG^* [%]	26	23	29

TABLE VII
LINK ERROR RATE EXPERIMENT RESULTS FOR WIRELESS NETWORKS

Parameters: NumN=49, TranR=300m, SpdR=0-10m/sec, PauM=2 sec, AppD=20pkts/sec			
ErrR	0.2%	0.4%	0.8%
Number of home-hunt attempts per PE/PU per second	0.228	0.217	0.209
Percentage of successful transmissions among NSs	17.47%	10.83%	8.12%
Ratio of RSerPool messages to data messages	1:6.22	1:4.22	1:4.37
Latency of home hunt (in seconds)	114.76	122.80	155.97
Ratio of unnecessary PE de-registrations to total PEs	4:14	4:14	4:14
SE	3.22	2.37	2.50
SSG [%]	19	12	11
SSG^* [%]	26	21	19

TABLE VIII
APPLICATION TRAFFIC DENSITY EXPERIMENT RESULTS FOR WIRELESS NETWORKS

Parameters: NumN=49, TranR=300m, SpdR=0-10m/sec, PauM=2 sec, ErrR=0.2%			
AppD	10 pkts/sec	20 pkts/sec	40 pkts/sec
Number of home-hunt attempts per PE/PU per second	0.201	0.228	0.234
Percentage of successful transmissions among NSs	18.13%	17.47%	15.48%
Ratio of RSerPool messages to data messages	1:2.74	1:6.22	1:6.99
Latency of home hunt (in seconds)	126.27	114.76	123.28
Ratio of unnecessary PE de-registrations to total PEs	2:14	4:14	5:14
SE	2.99	3.22	2.96
SSG [%]	17	19	16
SSG^* [%]	30	26	22

We also collected data for the impact of the transmission range on the RSerPool (not shown due to page limitations). The performance suffers if the range is too small or too large. These observations are confirmed by a study [4] of transmission-range effects on MAODV, where too large a range limits the effective bandwidth of neighbors (due to more congestion and interference, and higher energy consumption).

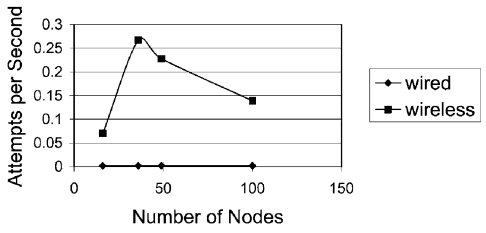
C. Performance Comparison of Wired and Wireless Networks

Our simulation results show that the performance of the RSerPool is quite different for the wired and wireless mobile networks. Fig. 3(a) illustrates that the number of home-hunt attempts per PE/PU per second is much higher in wireless mobile networks than in wired networks. The percentage of successful transmissions among NSs is about 90% for wired networks, but only between 3% and 18% for wireless mobile networks [Fig. 3(b)].

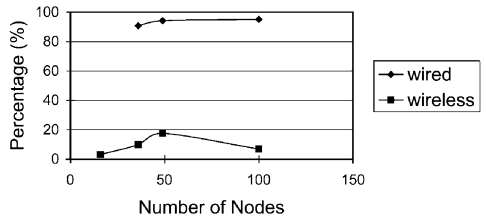
As shown by Fig. 3(c), the overhead of the RSerPool in wireless mobile networks is much higher than in wired networks. In the former, the number of data packets successfully delivered is less than six for each RSerPool packet sent. Among the RSerPool packets, about 50% are SERVER_HUNT packets and about 25% are ENDPOINT_KEEP_ALIVE and ENDPOINT_UNREACHABLE packets.

Note that ASAP has a mechanism that doubles the timer length each time a SERVER_HUNT message is not acknowledged, which aims at limiting the number of SERVER_HUNT messages sent. One may, thus, expect only modest overhead caused by home-hunt procedures. However, these procedures are initiated each time a PU/PE needs to access its home NS. Due to highly dynamic nature of mobile ad hoc networks, a PU/PE hunts for home with a high frequency, offsetting the benefit of the ASAP mechanism.

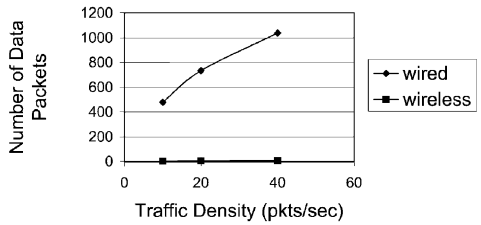
As shown in Tables V–VIII, another problem in wireless mobile networks is that a high percentage of PEs are unnecessarily deregistered by their home NSs. A PE will be deregistered by its home NS if it is found unreachable, or if it is reported unreachable by PUs more times than a given threshold. A PE or its home NS can move out of the transmission range of other nodes for a period that is long enough to trigger the unnecessary deregistration. In wired networks, on the other hand, the communications failures mainly come from the link errors, whose rate is generally very small and rarely causes unnecessary PE deregistrations.



(a) Number of home-hunt attempts per PE/PU (Tables III and V).



(b) Percentage of successful transmissions among NSs (Tables III and V).



(c) Data packets delivered per RSerPool packet sent (Tables IV and VIII).

Fig. 3. Performance comparison of wired and wireless networks.

While the poor performance of the RSerPool in wireless networks may come as a surprise, it should be noted that the experiments were run for a highly mobile infrastructure, with all nodes constantly moving in an ad hoc environment. The results confirm that a special design is needed if the RSP is to operate satisfactorily in highly stressed networks (used for disaster and military applications).

VII. Design for Wireless Mobile Environments

Two of the main deficiencies of the RSerPool in wireless mobile networks are the inaccuracy of the PE failure-detection mechanism and the high frequency of the name-server-hunt procedure. As concluded in Section VI-C, the first deficiency results in unnecessary PE deregistrations, which makes these PEs unavailable as servers and causes the NSs to exchange false PE information. The second one increases the signaling overhead and keeps a large number of PEs/PUs without an operational home NS. To overcome these problems, we introduce alternative mechanisms and present their evaluation.

A. Efficient Switchover Mechanism

To reduce the unnecessary PE deregistrations, we introduce a new switchover mechanism where each PE periodically sends a *heartbeat* message (ENDPOINT_HEARTBEAT) to its home NS. If an NS does not receive more than a predetermined number of heartbeats, it sets the PE status to UNAVAILABLE. (It resets the PE status to AVAILABLE after receiving a heartbeat.) When an NS receives an ENDPOINT_UNREACHABLE report for a PE, it

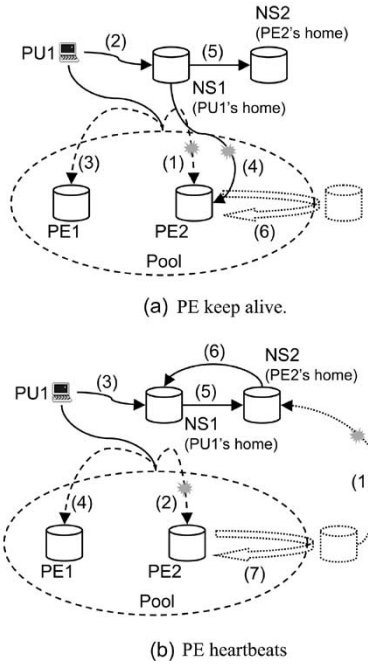


Fig. 4. Reducing false PE deregistrations.

multicasts a query to other NSs about this PE's status. Only if the PE is UNAVAILABLE, is it deregistered from the pool. In either case, PU switches over to a new PE. As opposed to the original RSerPool mechanism (Section III), no ENDPOINT_KEEP_ALIVE and ENDPOINT_KEEP_ALIVE_ACK are exchanged between an NS and a PE that is reported as unreachable.

To illustrate this new switchover mechanism, consider Fig. 4(a), where the original RSerPool mechanism is shown:

1) PU_1 detects PE_2 unreachable;
2) PU_1 reports the problem to its home NS_1 through ENDPOINT_UNREACHABLE message;
3) PU_1 switches over to PE_1;
4) NS_1 sends ENDPOINT_KEEP_ALIVE message to PE_2, to which it receives no response;
5) NS_1 sends PEER_NAME_UPDATE to all other NSs;
6) PE_2 moves back in range, but it is falsely deregistered from the pool.

This false deregistration of PE_2 can be avoided by the new switchover mechanism shown in Fig. 4(b):

1) NS_2 misses less than three consecutive ENDPOINT_HEARTBEAT messages from PE_2 and keeps PE_2 status as AVAILABLE;
2) PU_1 detects PE_2 unreachable;
3) PU_1 reports the problem to its home NS_1 through ENDPOINT_UNREACHABLE message;
4) PU_1 switches over to PE_1;
5) NS_1 multicast a query to other NSs, including NS_2, for PE_2 status;
6) NS_1 receives PE_2 status as AVAILABLE, after which it does not deregister PE_2;
7) PE_2 moves back in range, and it continues to be registered in the pool.

To evaluate the effectiveness of the new approach, a subset of experiments from Tables V and VI are run. It is found

TABLE IX
PE KEEP ALIVE VERSUS PE HEARTBEATS. *FD* IS THE NUMBER OF FALSE PE DEREGISTRATIONS

Parameters: one PE heartbeat per 30 seconds	PE reg's	PE keep alive scheme ENDPOINT_KEEP_ALIVE sent	ack'ed	*FD*	PE heartbeat scheme ENDPOINT_HEARTBEAT sent	*FD*	Overhead increase [%]
Table V, 36 nodes	10	236	165	2	300	0	-25
Table V, 49 nodes	14	142	91	4	420	1	82
Table V, 100 nodes	28	291	104	9	834	3	111
Table VI, 20 m/sec	14	204	116	7	420	0	31

TABLE X
ADVERTISEMENTS VERSUS HOME HUNT. *AI* IS THE NS ADVERTISEMENT INTERVAL [s]

Parameters: as in Table V	*AI*	SERVER_HUNT sent	ack'ed	hops	SERVER_ANNOUNCE sent	ack'ed	hops	Overhead reduction [%]
original scheme		40,890	260	42,755	-	-	-	-
new scheme	30	21,493	71	21,559	463	-	11,793	21.99
new scheme	20	20,238	56	20,421	692	-	13,893	19.74
new scheme	10	19,362	40	19,436	1375	-	18,157	12.07

that, thanks to the new switchover mechanism, the number of unnecessary PE deregistrations is reduced by 66%–100% (Table IX). The new mechanism introduces extra overhead as the number of new ENDPOINT_HEARTBEAT messages (at the rate of one heartbeat per 30 s) is typically higher than the number of ENDPOINT_KEEP_ALIVE messages and their acknowledgments in the original mechanism. However, this additional overhead does not significantly increase the network load because very small messages can be used for beaconing.

B. Finding Home NS: Advertisements Versus Home Hunt

As described in Section III, a PE or PU uses a name server to resolve names to transport addresses. At its startup time, or whenever its current home NS is not providing services, a PE or PU will attempt to find a new home server by either multicasting or sending a point-to-point SERVER_HUNT message to one or more NSs in the operation scope. However, in a dynamic mobile network, a PE/PU can move in and out of a range of an NS dynamically. This would require significant number of home-server hunts; depending on the number of nodes that are moving, the number of server-hunt multicast messages will increase. This conclusion is supported by our simulation results, which show that SERVER_ HUNT messages account for 50% of all RSerPool signaling overhead in highly mobile wireless networks (Section VI-C).

Let us consider an alternative mechanism that could reduce overhead and increase the efficiency of home hunt by making the NSs *advertise* their presence within an operation scope by a SERVER_ ANNOUNCE message. A PE/PU moving into a new operation scope could then attach to a particular NS and does not have to go through home-hunt procedures when it loses connectivity with its previous home server. This method is more efficient since SERVER_ANNOUNCE messages can be intended for use by multiple nodes to select their home server. For each periodic SERVER_ ANNOUNCE message, n nodes moving into a new operation scope during that time interval would select their home servers. This would avoid n SERVER_HUNT messages that are multicast to a number of potential home servers.

To evaluate the NS advertisement scheme, the experiment shown in Table V is run for the case of 100 nodes. The PE heartbeat mechanism described in Section VII-A is turned off. As shown in Table X, despite the fact that the cost of a SERVER_ANNOUNCE message (multicast to 86 PEs/PUs) is higher than that of a SERVER_HUNT message (multicast to 14 NSs), the overhead reduction ranges from 22% to 28%.

Without the NS advertisement scheme, only 0.65% of SERVER_ HUNT messages result in finding a new home (40 890 sent, 260 acknowledged). This very low efficiency of home hunting is mainly due to the constant movement of all nodes, which makes it difficult to efficiently set up connections between the endpoints. Consider the case where the NS advertisement interval is 30 s. The $\sim$22% reduction in the number of HOME_HUNT messages implies that the probability of finding a home NS at each attempt is increased in the NS advertisement scheme, as explained below.

Let R be the reduction in the number of HOME_HUNT messages in the NS advertisement scheme. Then, for each PE/PU, for each attempt at finding a home NS, consider the probabilities of finding this NS through the following message:

- SERVER_HUNT: $P_{\mathrm{HH}} = N_{\mathrm{HHR}}/N_{\mathrm{NN}}$, where N_{HHR} and N_{HH} are the numbers of HOME_HUNT_RESPONSE and HOME_HUNT messages, respectively;
- SERVER_ANNOUNCE in the NS advertisement scheme: $P_{\mathrm{A}} = R/N_{\mathrm{HH}}$;
- SERVER_HUNT or SERVER_ANNOUNCE: $P_{\mathrm{AHH}} = P_{\mathrm{A}} + (1 - P_{\mathrm{A}}) * P_{\mathrm{HH}}$.

From Table X, we obtain $P_{\mathrm{HH}}[\%] = 260/40\ 890 = 0.64$ in the original scheme, $P_{\mathrm{HH}}[\%] = 71/21\ 493 = 0.33$ in the new scheme, $R = 40\ 890 - 21\ 493 = 19\ 397$, and $P_{\mathrm{A}}[\%] = 19\ 397/40\ 890 = 47.44$. Finally, $P_{\mathrm{AHH}}[\%] = 47.44 + (100 - 47.44) * 0.0033 = 47.61$. In other words, in the original scheme, the probability of finding a home NS for a given PE/PU is 0.64% for a single home-hunt attempt (the overall probability increases as the PE/PU attempts several home hunts). In the new NS advertisement scheme, the probability that a PE/PU finds home at first attempt is increased to 47.61%, which makes the entire procedure significantly more efficient.

VIII. Conclusion and Future Work

The reliable server pooling (RSP) is a service-overlay framework to provide persistent connections and balanced traffic for different applications. The IETF RSerPool WG has proposed a light-weight architecture to implement the RSP.

In this paper, the performance of the RSerPool is evaluated for wired and wireless mobile ad hoc networks. The simulation results show that the current version of the RSerPool works well in fixed and relatively reliable environments, but its performance worsens rapidly as the networks become more unreliable or mobile. The issues we identified in wireless mobile ad hoc networks include network partition, high signaling overhead, difficulty in synchronization among name servers, and excessive aggressiveness in handling failures.

To overcome the inaccuracy of the PE failure detection mechanism and the high frequency of the name-server-hunt procedure, we introduce alternative mechanisms and present their numerical evaluation. In particular, a new switchover mechanism based on PE-heartbeat and a new scheme for finding a home NS, where NSs advertise their presence, are studied. We observe significant improvements in overhead reduction and the efficiency of home-hunt procedures.

Further research will investigate different server-selection algorithms [10] and the mobility models [5], which are likely to impact the performance metrics. The RSP will be studied for specific applications such as disaster recovery and combat field communications, where the mobility patterns follow a group mobility model rather than the random waypoint one.

Perfect synchronization among NSs in a dynamic mobile network cannot be achieved as nodes move in and out of range dynamically. In addition, the overhead of synchronization would be a detriment in such networks. We, thus, plan to study the RSP that uses local server-namespaces.

Acknowledgment

The authors would like to thank Dr. T. Kunz of Carleton University for providing parts of the MAODV C++ source codes.

Disclaimers

- All results in this paper apply to the RSerPool architecture and protocols from the 2001–2002 Internet drafts [23], [26], [31].
- The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

References

[1] L. Alvisi, T. C. Bressoud, A. El-Khashab, K. Marzullo, and D. Zagorodnov, "Wrapping server-side TCP to mask connection failures," *INFOCOM'01*, pp. 329–337, Apr. 2001.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM SOSP: Symp. Operation System Principles*, Banff, AB, 2001, pp. 131–145.

[3] K. Appleby, G. Goldszmidt, and M. Steinder, "Yemanja—A layered fault localization system for multi-domain computing utilities," *J. Network Syst. Manage. (Topics in Integrated Management)*, vol. 10, pp. 171–194, 2002.

[4] E. M. Belding-Royer and C. E. Perkins, "Transmission range effects on AODV multicast communication," *J. Mob. Networks Appl.*, vol. 7, no. 6, pp. 455–470, 2002.

[5] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad-hoc network research," *J. Wireless Commun. Mob. Comput. (Mobile Ad-Hoc Networking—Research, Trends and Applications)*, vol. 2, pp. 483–502, 2002.

[6] K. Chen, S. H. Shah, and K. Nahrstedt, "Cross-layer design for data accessibility in mobile ad-hoc networks," *J. Wireless Pers. Commun.*, vol. 21, no. 1, pp. 49–76, 2002.

[7] Cisco Distributed Director [Online]. Available: http://www.cisco.com

[8] P. T. Conrad and P. Lei, Services provided by reliable server pooling, in Internet Draft, IETF, 2002. draft-conrad-rserpool-service, work in progress.

[9] S. G. Dykes, K. A. Robbins, and C. L. Jeffrey, "An empirical evaluation of client-side server selection algorithms," in *Proc. IEEE INFOCOM*, Tel Aviv, Israel, 2000, pp. 1361–1370.

[10] M. A. Fecko, S. Samtani, M. U. Uyar, and P. T. Conrad, "Designing reliable server pools for battlefield ad-hoc networks," in *Proc. IIIS SCI: World Multi-Conf. System. Cybern. Inf.*, vol. X, Orlando, FL, 2002, pp. 357–362.

[11] I. Guardini, P. Fasano, and G. Girardi, "IPv6 operational experience within 6bone," presented at the INET: Internet Soc. Conf., Yokohama, Japan, 2000.

[12] E. Guttman, "Service location protocol: Discovery of IP network services," *IEEE Internet Comput. M.*, vol. 3, no. 4, pp. 71–80, 1999.

[13] F. Hao, E. W. Zegura, and M. H. Ammar, "Supporting server selection in differentiated service networks," *INFOCOM'01* , pp. 659–668, Apr. 2001.

[14] T. Hara, "Effective replica allocation in ad-hoc networks for improving data accessibility," *INFOCOM'01*, pp. 1568–1576.

[15] S. Hull, *Content Delivery Networks: Web Switching for Security, Availability and Speed*. Berkeley, CA: McGraw-Hill, 2002.

[16] A. M. Tjoa, M. Takizawa, M. Raynal, and W.-S. E. Chen, presented at the Proc. IEEE ICDCS: Int.Conf. Distrib. Comput. Syst., Vienna, Austria, 2002.

[17] B. Sengupta, R. Cruz, and G. Pacifici, presented at the Proc. IEEE INFOCOM, Anchorage, Alaska, 2001.

[18] T. Kunz and E. Cheng, "On-demand multicasting in ad-hoc networks: Comparing AODV and ODMRP," *ICDCS'02 [16]*.

[19] S. Liu and J. Turner, "Placing servers in overlay networks," presented at the SCS SPECTS: Int. Symp. Perf. Eval. Comput. Telecommun. Syst., San Diego, CA, 2002.

[20] J. Loughney, M. Stillman, Q. Xie, and R. Stewart, Comparison of protocols for reliable server pooling, in Internet Draft, IETF, 2002. draft-ietf-rserpool-comp, work in progress.

[21] F. Pedone, M. Wiesmann, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in *Proc. IEEE ICDCS: Int. Conf. Distrib. Computer System*, 2000, pp. 464–474.

[22] R. Stewart and C. Metz, "SCTP: New transport protocol for TCP/IP," *IEEE Internet Comput. M.*, vol. 5, no. 6, pp. 64–69, 2001.

[23] R. Stewart and Q. Xie, Aggregate server access protocol (ASAP), in Internet Draft, IETF, 2002. draft-ietf-rserpool-asap, work in progress.

[24] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode, "Migratory TCP: Highly available internet services using connection migration," presented at the ICDCS'02, Vienna, Austria, 2002.

[25] A. Tridgell, "Efficient Algorithms for Sorting and Synchronization," Ph.D. dissertation, Australian Nat. Univ., Canberra, Australia, 2000.

[26] M. Tuexen, Q. Xie, R. Stewart, M. Shore, L. Ong, J. Loughney, and M. Stillman, Architecture for reliable server pooling, in Internet Draft, IETF, 2001. draft-ietf-rserpool-arch, work in progress.

[27] Network Simulator—NS-2. Univ. Southern California, Infor. Sci. Inst., Marina del Rey, CA. [Online]. Available: http://www.isi.edu/ nsnam/ns

[28] M. U. Uyar, J. Zheng, M. A. Fecko, and S. Samtani, "Performance study of reliable server pooling," in *Proc. IEEE NCA: Int. Symp. Network Computer Appl.*, Cambridge, MA, 2003, pp. 205–212.

[29] ——, "Reliable server pooling in highly mobile wireless networks," in *Proc. IEEE ISCC: Int. Symp. Comput. Commun.*, Kemer-Antalya, Turkey, 2003, pp. 627–632.

[30] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Database replication techniques: A three parameter classification," presented at the IEEE SRDS: Symp. Reliab. Distrib. Syst., Nürenberg, Germany, 2000.

[31] Q. Xie and R. Stewart, Endpoint name resolution protocol (ENRP), in Internet Draft, IETF, 2002. draft-ietf-rserpool-enrp, work in progress.
[32] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee, "Application-layer anycasting: A server selection architecture and use in a replicated web service," *IEEE/ACM Trans. Network*, vol. 8, no. 4, pp. 455–466, 2000.

M. Ümit Uyar (SM'91) received the B.S. degree from Istanbul Teknik Üniversitesi, Istanbul, Turkey, and the M.S. and Ph.D. degrees from Cornell University, Ithaca, NY, all in electrical engineering.

Currently, he is with the Electrical Engineering Department, City College, City University of New York. He was a Distinguished Member of Technical Staff at AT&T Bell Laboratories, until 1993. He coedited *Conformance Testing Methodologies and Architectures for OSI Protocols* (Piscataway, NJ: IEEE). He holds two U.S. patents.

Dr. Uyar received a Vice Presidential Quality Award from Bell Laboratories for codesigning software tools, three AT&T Bell Laboratories Vice Presidential Research Appreciation Awards, and a Best Paper Award in AT&T Electronic Testing Conference. He was granted "Doçent" title by the National University Council of Turkey in 1992. He cochaired the 12th International Symposium on Protocol Specification, Testing, and Verification (1992), and the 6th Internation Conference on Formal Description Techniques (1993).

Jianliang Zheng received the M.S. degree in physics from Shanghai Jiao Tong University, Shanghai, China. He is currently working toward the Ph.D. degree in electrical engineering at City College, City University of New York, where he is working on the reliability and security of wireless mobile communications.

Mariusz A. Fecko (M'00) received the M.S. degree in electronics and computer science from Stanislaw Staszic University (AGH), Krakow, Poland, and the M.S. and Ph.D. degrees in computer and information science from the University of Delaware, Newark.

He is a Postdoctoral Fellow at the University of Delaware, where he jointly developed a formal specification and conformance-testing methodologies for radio-network protocols of the U.S. Army CECOM. In 2000, he joined Telcordia Technologies, Inc., Piscataway, NJ, as a Research Scientist in applied research area. In his work on process and test automation for telecom software systems and services, he contributed to improving the quality of exchange link through a novel use of XML technologies. He also serves as Principal Investigator in the ARL-funded research alliance in survivable wireless networks.

Dr. Fecko received the Telcordia CEO Team Award (2001) and the Executive Director Appreciation Award (2002).

Sunil Samtani received the B.E. degree in computer engineering from the University of Bombay, Maharashtra, India, in 1996, the M.S. degree in computer science from the University of Missouri, Kansas City, in 1998, and the M.B.A. in finance from the Stern School of Business, New York, NY, in 2002.

He has been with Telcordia Technologies, Piscataway, NJ, since 1999 and is currently Director of the Wireless Information Assurance Research Group. He serves as the Network Communications PI for DARPA's Software for Distributed Robotics, and leads a team of researchers to bring CECOM MOSAIC technologies to TRL-6 readiness. Before joining Telcordia, he was with SPRINT, Local Telecommunications Division, in 1998 as National Systems Product Manager, where he was responsible for Integrated Data Architecture for SPRINT's operation support systems. His current research interests include automatic configuration, QoS for IP networks, mobility management, transport protocols, and differentiated services.

Phillip T. Conrad (M'94) received the B.S. degree in computer science from West Virginia Wesleyan College, Buckhannon, in 1985, the M.S. degree in computer science from West Virginia University, Morgantown, in 1988, and the Ph.D. degree in computer science from the University of Delaware, Newark, in 2000.

He is an Assistant Professor at the University of Delaware. From 1998 to 2000, he held an Instructor position at Temple University, Philadelphia, PA, and from 2001 to 2003, he was an Assistant Professor at the same university. His research interests are in transport protocols, multimedia communication, and computer science education.

Dr. Conrad was the Advisor for the student ACM Chapter, which was a runner up for the ACM Student Chapter Excellence Awards in 2000, and won the Chapter Excellence Awards in 2001 and 2002, while at Temple University. He is a Member of the Association for Computing Machinery (ACM).