An Experience in Evaluating Publish/Subscribe Services in a Wireless Network

Mauro Caporuscio
Dipartimento di Informatica
Università dell'Aquila
I-67010 L'Aquila, Italy
caporusc@univaq.it

Antonio Carzaniga and Alexander Wolf
Department of Computer Science
University of Colorado
Boulder, Colorado 80309 USA
{carzanig,alw}@cs.colorado.edu

ABSTRACT

As wireless technology becomes more available, developers of distributed applications are becoming more interested in how that technology affects the performance of their systems. We have developed a distributed publish/subscribe communication service initially hosted on the standard IP-wired network infrastructure, but would now like to rehost that service onto a GPRS wireless network. This paper reports on our experience in attempting to evaluate the performance of the service using an available emulation environment. Our conclusion from our experience to date is that current tools do not model the wireless network at an appropriate level of abstraction. In particular, they do not allow us to study the integration of individual publish/subscribe service-layer elements with GPRS network-layer elements, nor do they allow us to study multiple GPRS clients interacting over the network. Instead we were limited to results related to the interaction between an individual GPRS client and the GPRS network modeled as a monolith.

1. INTRODUCTION

Providers of network infrastructure are trying to promote the view that computer-based applications should run seamlessly over the wired and wireless portions of the network. In other words, the services that we are accustomed to having available in a wired network, such as IP, should also be available in some form over radio channels and mobile devices. In fact, such services are becoming more and more widely available. The GPRS (General Packet Radio Service) wireless network infrastructure in particular is designed to support IP-based applications [1].

Despite the appeal of this unified view of the network, in practice developers must be very careful in how they build applications intended for wireless operation. This means that they must be given an ability to study how wireless technology affects the performance of their systems.

WOSP '02, July 24-26, 2002 Rome, Italy

Two classes of systems are especially highlighting this issue. The first class consists of distributed, peer-to-peer applications, which are characterized by the *ad hoc* interaction of communicating clients. In this case, the (wireless) network is simply acting as a medium for the communication. The second class of systems consists of so-called "overlay networks" of application-level "routers". Here, the application depends upon the careful deployment across network nodes of specialized, cooperating servers. Often this is done to achieve some scalability, survivability, or administrative control properties. Notice that these two classes of systems are actually synergistic and are therefore typically used in concert.

In our work, we have developed a distributed publish/subscribe communication service architected as an overlay network [3]. This service, which we call Siena, supports a variety of applications all involving the interaction of multiple clients. The service was initially hosted on the standard IP wired network infrastructure, but now we would like to rehost that service onto a GPRS wireless network. Thus, we are encountering the problem of how to evaluate the performance of the service in this new context, and from both the perspective of client interaction and deep network deployment.

This paper reports on our experience in attempting to evaluate the performance of the service using an available emulation environment. Our conclusion from our experience to date is that current tools do not model the wireless network at an appropriate level of abstraction. In particular, they do not allow us to study the integration of individual publish/subscribe service-layer elements with GPRS network-layer elements, nor do they allow us to study multiple GPRS clients interacting over the network. Instead we were limited to results related to the interaction between an individual GPRS client and the GPRS network modeled as a monolith.

In the next section we provide some necessary background on Siena and GPRS. Following that we briefly survey available GPRS tools, explaining how we came to choose one of those tools for our experiments. We then describe the basic configuration of Siena and GPRS used in our experiments. With this foundation, we illustrate the sort of experiments made possible by the chosen GPRS tool. We conclude with some final reflections on our experience and thoughts for future work.

Note that the point of this paper is not the experimental results themselves, but rather a discussion of our experience in trying to perform the experiments. We hope that this experience will help others engaged in similar efforts, and inform those contemplating the development of new tools for wireless-network performance evaluation.

2. BACKGROUND

In this section we briefly provide some background information on Siena and GPRS. Extensive detail on both can be found in the cited references.

2.1 Siena

Siena [2, 3] is a publish/subscribe event notification service.[1] It is implemented as a distributed network of servers so as to achieve a scale suitable for large numbers of communicants and high volumes of notifications spread across a wide-area network. As depicted in Figure 1, Siena servers act as both access points, providing clients with an extended publish/subscribe interface, and as store-and-forward network routers. The clients are of two kinds: *publishers*, which are

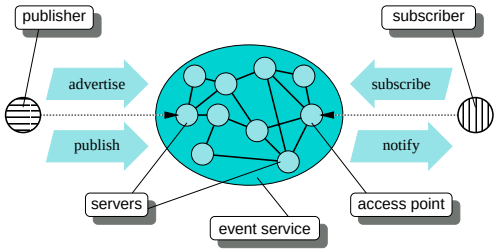

Figure 1: Siena Architecture

the generators of notifications, and *subscribers*, which are the consumers of notifications; of course, a client can act as both a publisher and a subscriber.

Publishers use the access points to *advertise* the information about notifications that they generate and to *publish* the advertised notifications. Subscribers use the access points to *subscribe* for notifications of interest by supplying a predicate, called a *filter*, to be applied to the content of notifications. Siena is responsible for selecting the notifications that are of interest to subscribers and then delivering those notifications to the subscribers via the access points. Siena uses information about the relationship among subscription predicates and advertisements to optimize the message traffic in the network.

Underlying Siena's interface is a *notification data model* that drives the semantics of the service. A notification in the model is a set of typed attributes. Each individual attribute has a *type*, a *name*, and a *value*, but the notification as a whole is purely a structural value derived from its attributes. The attribute types belong to a predefined set of primitive types commonly found in programming languages and database query languages, and for which a fixed set of operators is defined. The operators are used in the filters provided by subscribers to specify a set of constraints over the attributes. Figure 2 shows a filter on the top and a notification matching the filter on the bottom.

string	$dest = MXP$
int	$price < 500$

string	$carrier = UA$
string	$dest = MXP$
int	$price = 400$
bool	$upgradeable = true$

Figure 2: A Siena Filter and a Siena Notification

In effect, Siena is a wide-area network of pattern matches and routers overlaid atop some other wide-area communication facility, such as the Internet. Our interest here is in studying the effect of overlaying a Siena network on the GPRS infrastructure.

2.2 GPRS

The current generation of mobile radio technology is largely based on circuit-switched protocols such as GSM [11], where a traffic channel is allocated to a user for the entire duration of call. In such a protocol, the channel will be simply unused if there is no data to be transmitted in certain intervals during the call. This limits both the data rates and the number of users that can be supported.

GPRS (General Packet Radio Service) [1] is a mobile communication standard based on packet-switched radio transmission. The main advantage over circuit-switched radio technologies is its handling of the radio resources. In particular, a traffic channel is allocated only when need and is released immediately after the transmission of a packet. GPRS also allows users to be allocated multiple channels, leading to higher data rates.

GPRS is structured as a GSM overlay, although it does require some changes to a few of the basic GSM network elements. The details of this mapping are not relevant to this paper, nor are details about the internal architecture of GPRS. What is important to understand is that GPRS provides an architecture for integrating external packet-data networks (e.g., the Internet backbone wired network) and what are called *mobile stations* (i.e., cell phones, PDAs, and other such mobile devices) located within some number of physically separate *service areas*.

Figure 3 illustrates this architecture. Within a service area, GPRS provides a special IP network that, at one end, supports a gateway to the external packet-data network and, on the other end, a gateway to the GSM radio transmission network. The external packet-data network is used to tie together multiple service areas.

2.3 Deploying Siena on GPRS

A critical issue in deploying the Siena service is determining the appropriate number, location, and interconnection structure of routers. A simplistic deployment would be to position a single router somewhere within the external packet-data network. Siena clients running on the mobile stations would interact through this single router. Of course,

[1] `http://www.cs.colorado.edu/serl/siena/`

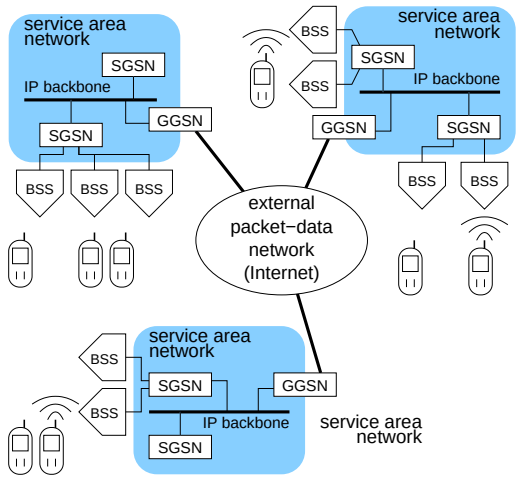

Figure 3: GPRS Architecture

such a configuration would be unlikely to support large-scale applications, and therefore we would like to study richer configurations involving multiple, cooperating Siena routers deployed in various ways throughout the network. One can consider a variety of such configurations, but given the physical and administrative structure inherent in the GPRS architecture, a reasonable allocation of routers is to place one or more routers in the external packet-data network, as well as one or more routers within each service-area network.

Deciding upon an optimal configuration requires careful performance evaluations. Therefore, we are interested in the capabilities of GPRS tools supporting such evaluations.

3. SURVEY OF GPRS TOOLS

In this section we briefly survey several GPRS tools. The tools range from those for doing network planning to those for doing performance evaluations. They also range from those developed by commercial vendors of GPRS technology to those developed by research institutions. While this survey is almost certainly not complete, the tools are all those for which we were able to obtain some amount of documentation.

3.1 Nokia NetAct™ Planner

Nokia's NetAct Planner [12] is an integrated set of tools for planning radio-based voice and data networks, including those based on GPRS technology. The tools allow one to "plan" in the sense of designing how the network will be deployed to satisfy usage and physical constraints. For example, there is a tool called the Rollout Planner that supports the process of site acquisition and project tracking. Another tool is the Transmission Planner, which supports the planning of the transmission and datacom network, including dimensioning and network architecture comparisons. A third tool supports an analysis of the placement and strength of microwave links.

3.2 Motorola GPRS Emulator

Motorola's GPRS emulator [10] is designed to help developers understand how their applications can be expected to behave over a typical GPRS connection. The emulator runs on a standalone Linux computer, with application clients and servers connected to that computer over a normal IP link. In essence, the standalone computer acts as a monolithic GPRS network. The emulator provides communication effects that reflect the performance of client/server interaction over the GPRS network under a variety of conditions, including normal loads, heavy ("busy hour") loads, and both short and long interruptions in signals.

3.3 Ericsson GATE II

Ericsson's GATE II [5] is another Linux-based emulator of a GPRS network. It emulates typical properties of a GPRS network, including varying bandwidths, loads, latencies, and radio conditions. The emulator is made available in a rather unusual way: Rather than being available for installation and use in the evaluator's environment, it is provided as a service to which one brings an application for evaluation. The evaluation itself is carried out by trained personnel at designated service centers.

3.4 University of Helsinki Seawind

In cooperation with Nokia Mobile Phone and Sonera Corporation, the University of Helsinki has developed Seawind [9], a Linux-based emulator of wireless networks. The emulator can be used to study network flow and congestion control, as well as other properties of an application communicating over a GPRS network. Like the Motorola GPRS emulator, it based on the use of a normal wireline local-area network. Link characteristics are emulated by delaying, dropping, and modifying the flow of packets according to a set of simulation parameters.

3.5 Network Simulator

NS-2 (Network Simulator) [6] is a general-purpose discrete event simulator for networks. The architecture of the simulator is designed to allow the specifics of a given network to be provided as a pluggable module. Recently, a module for simulating a GPRS network has become available [8], but we have not yet had an opportunity to fully study its capabilities. What we do understand at this point is that it is more suited to studying the internal behavior of the GPRS network than it is to studying the interaction of an application with the network.

3.6 Selecting a Tool

In order to carry out our evaluation, we needed to select from among the available GPRS tools. In a sense, our choice was easy. The Nokia NetAct Planner is targeted at network planning, not performance evaluation. The Motorola emulator, while it appears extremely well suited for our evaluation, is simply not yet available. The Ericsson GATE II emulator might also be suitable, but the fact that it is available only as a second-hand service makes it very inconvenient to iteratively develop experiments.

We selected Seawind because of its combination of reasonable functionality and immediate availability. Nevertheless, as we detail in the next section, Seawind is limiting in the kind of information that we can gather, specifically in regard to the effect of deploying and operating Siena servers in the GPRS network. NS-2 might well be an alternative worth

exploring in the future, but it too has its limitations. In fact, Seawind and NS-2 appear to be complementary, since Seawind concentrates on the interaction of an application with the (monolithic) network, while NS-2 combined with the GPRS module concentrates on the performance of the network itself.

4. EXPERIMENTATION SETUP

Seawind emulates a point-to-point communication channel extending over a GPRS network. One end of the channel represents the mobile station, while the other endpoint represents the remote host. The mobile station and the remote host act as workload generators for the GPRS network. A *network protocol adapter* binds a workload generator at each endpoint of the emulated channel. The traffic produced by one workload generator is fed into the Seawind emulation process through one adapter. It is then processed by Seawind and passed on to the workload generator at the other end through the corresponding adapter. In processing through-traffic, Seawind emulates the behavior of a GPRS network according to its configuration parameters, thereby introducing characteristic delays, errors, and packet loss.

The current version of Seawind comes with a protocol adapter for the point-to-point protocol [13] that can be used to redirect IP traffic though Seawind. In practice, running Seawind amounts to running the main Seawind emulation process connected with two PPP adapters (running as separate processes). Each adapter creates a PPP interface configured with a given IP address, and with a "peer" address corresponding to the IP address of the other adapter. A workload generator is implemented by an ordinary network application, appropriately configured to direct some of its traffic to the IP address of one of the PPP adapters bound to Seawind. Seawind produces a traffic trace in *tcpdump* format [7] that can be analyzed by a variety tools [4].

Seawind has two significant limitations for the studies that we would like to perform. First, it models the GPRS network as a simple tunnel, capable only of moving data between a mobile station and the external packet-data network. In particular, Seawind does not model workload generators deployed *within* the GPRS network, which for us means that it cannot be used to study the performance of multiple, distributed Siena routers. Second, Seawind focuses on a single pair of workload generators, not taking into account the interactions among multiple mobile stations sharing the same pool of radio links and base-station resources. While Seawind does in fact model the effect of other applications in the same cell, it does so by simulating generic, static "background" traffic. Such an approach captures some conflicts in resource allocation, but it does not reveal potential destructive dynamics resulting from the combination of interrelated applications.

Despite these two shortcomings, we can still extract some useful data using Seawind. For our experiments, we used a Siena subscriber and a Siena server as workload generators. The experiment setup is depicted in Figure 4. The subscriber plays the role of the mobile station. The server plays the role of the remote host. Notifications are produced by a publisher connected to the server directly on the remote host. Each experiment is defined by the sequence of subscriptions and notifications exchanged between subscriber and server, by the configuration of the connections between the subscriber and the server, and by the configuration of the GPRS network.

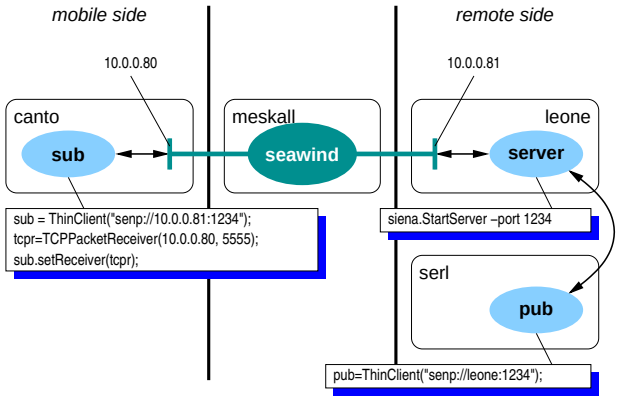

Figure 4: A Siena Mapping onto Seawind

The workload that we used in our experiments consists of one subscription posted by the subscriber, followed by a number of matching notifications sent from the server to the subscriber.

Siena uses a generic message-based communication mechanism that is realized in the current implementation by three specialized connectors. The configuration of the server/subscriber connection is obtained by selecting a specific connector. In particular, the choices include a UDP connector, a basic TCP connector, and what we refer to as a "keep-alive" TCP connector. A UDP connector sends messages through UDP packets, a basic TCP connector uses one TCP connection per message, and a keep-alive TCP connector attempts to use the same TCP connection for multiple messages.

For the configuration of the GPRS network, we experimented with a subset of the rich set of parameters offered by Seawind. In particular, in accordance with the GPRS CS-1 specification, we emulated a mobile station capable of using one uplink channel and up to three downlink channels. This setting is shown in the parameters of Table 1.

Parameter	Uplink	Downlink
ms_max_rate	3	3
available_rate	0-1	0-3
rate_base	9050 bps	9050 bps

Table 1: GPRS CS-1 Simulation Parameters

The *ms_max_rate* parameter defines the capabilities of the mobile station. A value of 3 selects the most advanced class of mobile stations, capable of handling data communications (GPRS) and normal calls (GSM) at the same time. The *rate_base* is the bandwidth of an individual channel. *available_rate* determines the range of channels available to the mobile station. The actual number of channels allotted to the mobile station at any time depends on the presence of

other GPRS or GSM users in the same cell.

In addition to the parameters of Table 1, which serve to characterize the connectivity of the mobile station to its base station, we must set other parameters that determine the quality of the communication channel. These parameters are listed in Table 2.

Parameter	Value
error_rate_type	BIT
error_probability	static 10^{-3} static 10^{-4}
error_handling	DELAY_ITERATE FORWARD DROP
error_delay_function	uniform distribution 40–50ms
delay_drop_threshold	static 10s

Table 2: GPRS Error Simulation Parameters

The effect of noise is to introduce transmission errors or delays. Errors occur with a probability determined by the *error_rate_type* and *error_probability* parameters. In our experiments, errors are set to occur at the level of individual bits with a probability of 10^{-3} and 10^{-4}.

The *error_handling* parameter determines how the GPRS network handles transmission errors. With "DELAY_ITERATE" the network provides a reliable delivery service by simply forcing retransmission, which in turn introduces a delay for end-to-end communications. Alternative modes are "FORWARD", in which errors are simply ignored and passed on to higher levels in the communication stack, and "DROP", which causes the network to drop packets that contain errors. In the case of "DELAY_ITERATE", *error_delay_function* determines the interval before retransmission and *delay_drop_thresholds* defines an upper bound for the total retransmission delay, after which a packet is simply dropped.

5. SAMPLE RESULTS

This section presents some sample results that we were able to obtain using Seawind to evaluate the configuration described in the previous section. The primary goal of these experiments was to evaluate the impact of deploying Siena onto the wireless GPRS network. We did this from two different perspectives. The first was to gather data characterizing the performance of the three different low-level connectors (UDP, TCP, and keep-alive TCP) on the wireless network. The second was to compare these results with baseline data collected on a local-area, wired network. By doing this we should get an initial indication of whether a seamless integration of wired and wireless communication is feasible for a publish/subscribe communication service.

Table 3 shows the network usage corresponding to the three low-level connectors under two different error probabilities. In essence, this table captures data on the cross product of the parameter values of Table 2. We collected counts of application-level notifications received by the subscriber and the resulting counts of IP packets. The counts shown in each cell are the average taken from five runs of the simulation. In all cases, there were 100 notifications published. The data give an indication of the circumstances that lead to different notification loss rates. For example, as we would expect, the highest loss rate occurs at an error probability of 10^{-3} under the DROP error-handling mode. We can also see that the keep-alive connector is the most sensitive to increasing error rates and decreasing quality of error-handling service.

Table 4 shows the baseline behavior obtained by running the application on a local-area, wired network. The data characterize the relative overhead of each of the low-level connector protocols. For instance, UDP encounters no overhead. (The two extra packets are used to carry the subscription and unsubscription messages.) On the other hand, approximately eight packets, on average, are required by TCP to deliver a single notification. We can compare the baseline overhead to that experienced in the wireless network. The overhead of TCP in the wireless case is approximately twelve packets per notification, considerably higher than in the local-area, wired case.

6. CONCLUSION

In this paper we have described our attempt at performance evaluations of a distributed application deployed over a wireless network. The application is characterized by the interaction of multiple clients residing at the periphery of the network, as well as by the need to deploy elements of the application deep into the network. To our disappointment, we were not able to find tools capable of supporting a full evaluation of this application. We were limited to the narrow evaluation of a single client interacting across the network with a single server. Nevertheless, our experience should not be taken as a criticism of Seawind, the tool that we decided to use for our evaluation. In fact, we found Seawind to be a reasonable and useful tool for its purpose.

Clearly, a need exists for a different kind of tool for wireless-network performance evaluation. Before embarking on the development of such a tool ourselves, we first plan to study the capabilities of NS-2 and its GPRS module, which hold some promise for modeling and evaluating services deployed deeply into a wireless network. We might in fact be able to extend them to also allow modeling and evaluation of client interaction over the network.

Acknowledgments

The work of the authors was supported in part by a Research Exchange (REX) grant from the Usenix Association. The work of the authors was also supported by the Defense Advanced Research Projects Agency, Air Force Research Laboratory, Space and Naval Warfare System Center, and Army Research Office under agreement numbers F30602-01-1-0503, F30602-00-2-0608, N66001-00-1-8945, and DAAD19-01-1-0484. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory, Space and Naval Warfare System Center, Army Research Office, or the U.S. Government.

error_probability = 10^{-3} *notifications sent* = 100						
	DELAY_ITERATE		FORWARD		DROP	
	notifications received	**IP packets**	**notifications received**	**IP packets**	**notifications received**	**IP packets**
Keep Alive	79	875	17	350	8	285
TCP	100	1173	64	1205	62	1571
UDP	79	82	73	78	66	97
error_probability = 10^{-4} *notifications sent* = 100						
	DELAY_ITERATE		FORWARD		DROP	
	notifications received	**IP packets**	**notifications received**	**IP packets**	**notifications received**	**IP packets**
Keep Alive	82	855	72	458	70	443
TCP	100	1153	100	1156	99	1152
UDP	100	106	84	95	76	91

Table 3: Siena Behavior in the Wireless GPRS Network

	notifications	**IP packets**
Keep Alive	100	437
TCP	100	828
UDP	100	102

Table 4: Siena Behavior in a Local-Area, Wired Network

7. REFERENCES

[1] G. Brasche and B. Walke. Concept, services and protocols for the new GSM Phase 2+ General Packet Radio Service. Technical report, IEEE Communications Magazine, 1997.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 219–227, Portland, OR, July 2000.

[3] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

[4] G. Combs. The ethereal network analyzer. UNIX manual. Available from http://www.ethereal.com/.

[5] Ericsson Mobility World. GATE II. http://www.ericsson.com/mobilityworld/.

[6] K. Fall and K. Varadhan. *The* ns *Manual*. The VINT Project, November 2001. A Collaboration between researchers at UC Berkeley, LBL, USC/ISI and Xerox PARC.

[7] V. Jacobson, C. Leres, and S. McCanne. tcpdump - dump traffic on a network. UNIX manual. Available from http://www.tcpdump.org/.

[8] R. Jain. *GPRS Simulations using ns-Netvork Simulator*. PhD thesis, Department of Electrical Engineering, Indian Institute of Technology - Bombay, June 2001.

[9] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, and K. Raatikainen. Seawind: a Wireless Network Emulator. University of Helsinky, Finland.

[10] Motorola Wireless Development Centre. The Motorola GPRS Emulator. http://developers.motorola.com/developers/wireless/global/uk/emulator.htm.

[11] M. Mouly and M. Pautet. Current evolution of the GSM systems. Technical report, IEEE Pers. Commun., 1995.

[12] Nokia. Nect Act Planner. http://www.nokia.com/networks/services/netact/netact_planner/.

[13] W. Simpson. The point-to-point protocol (PPP). Request for Comments, July 1994. RFC 1661.