Opportunistic Channels: Mobility-aware Event Delivery

Yuan Chen[1], Karsten Schwan[1], and Dong Zhou[2]

[1] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
{yuanchen,schwan}@cc.gatech.edu
[2] DoCoMo USA Labs, San Jose, CA 95110, USA
zhou@docomolabs-usa.com

Abstract. The delivery of data in pervasive systems has to deal with end host mobility. One problem is how to create appropriate, application-level data provisioning topologies, termed data brokers, to best match underlying network connectivity, end user locations, and the locales of their network access. Another problem is how to balance workloads in such overlay networks, in response to mobility and to changes in available processing and communication resources. This paper improves the performance of data provisioning by dynamically changing broker topologies and end users' assignments to brokers. Specifically, using publish/subscribe as a communication paradigm, a new abstraction, termed an *opportunistic* event channel, enables dynamic broker creation, deletion, and movement. Experimental and simulation results demonstrate the ability of opportunistic channels to optimize event delivery and processing when end users move across different network access points. The technique is to 'opportunistically' follow network-level handoffs across network access points with application-level handoffs of a user's broker functionality to a new, 'closer' broker. The potential load imbalances across brokers caused by such handoffs are also addressed.
Opportunistic channels are realized with the JECho event infrastructure. Performance advantages attained from their use can be substantial, with the cost of sending a message from a publisher to a mobile subscriber improved by up to 50%. Load balancing improves event delivery even for moderate numbers of event subscribers.

1 Introduction

Publish/subscribe is a widely used paradigm for interconnecting applications in distributed environments. It provides anonymous, inherently asynchronous group communication, where event providers and consumers interact via event brokers, as illustrated in Figure 1. Subscription means that an event consumer declares its interest in receiving certain events, using some predicate or more generally, in content-based subscription, a filter/conversion function defined on event contents (e.g., see the ECho and JECho infrastructures developed in our research [9, 38]). An event provider generates and publishes events, where message brokers are responsible for collecting event subscriptions and routing events from

publishers to interested consumers. Each provider/consumer connects to one of the brokers, where broker networks can be organized in multiple ways, ranging from single central broker (Elvin [31]), to hierarchical topologies (JEDI [7], Gryphon [24]), to general graphs (SIENA [3], READY [11]). Once a topology of brokers has been defined, appropriate routing paths must be established to ensure the correct and efficient delivery of events to all interested consumers.

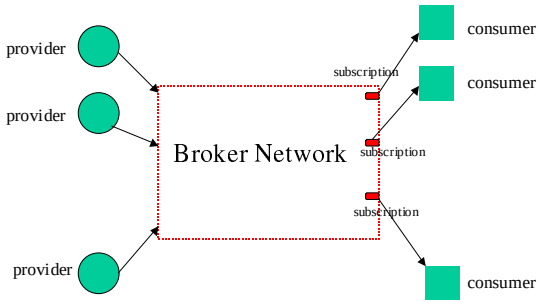

Fig. 1. Publish/Subscribe Communication Model

The issue addressed in this paper is that publish/subscribe systems' broker infrastructures or more generally, overlay networks [20] used in distributed applications do not take into account the unique needs of mobile systems. Problems include (1) which interconnection topology may currently best suit the underlying mobile system's communication paths [26] and how to dynamically construct such a topology, (2) selecting the numbers and capabilities of brokers to match the mobile platform's configuration, and (3) dynamic changes in (1) and (2) to match changes in mobile node locations [35], configurations, and resources. For publish/subscribe systems, this implies the need to use dynamically varying topologies with a changing number of brokers, and it requires runtime changes to providers' and consumers' assignments to brokers. A specific example is one in which a location change by an event provider or consumer results in a consequent change of the network access point used by the underlying Mobile IP protocol [25, 17]. This can lead to inefficiencies in broker communications, as depicted in Figure 2, where a mobile client's crossing of a network boundary results in a circuitous path from publisher to home broker to event subscriber.

Opportunistic channels is an event channel concept that addresses the mobility of event producers, consumers, and of the dynamic changes in the pervasive systems where they operate. The following attributes of opportunistic channels differentiate them from previous work:

- *Network and location awareness.* The event broker associated with an opportunistic channel is aware of the underlying network topology used for transporting events from providers to consumers. It is also aware of the respective locations of both.
- *Dynamic broker adaptation.* Brokers can be created, deleted, and moved at runtime, resulting in the ability of opportunistic channels to react to changes

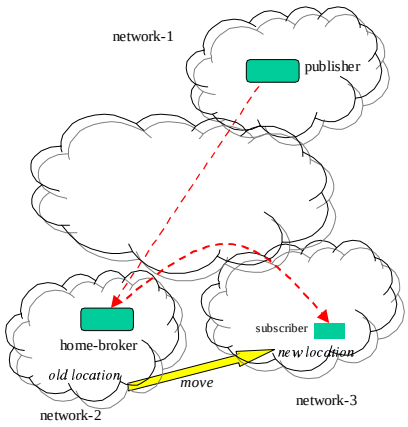

Fig. 2. Inefficient Event Delivery after Subscriber Migration

in end user locations and also to secondary effects of such changes, such as broker overloads due to changes in client/broker assignments.

The algorithm used for dynamic broker deployment and movement is one that evaluates the performance opportunities presented by current network connectivities and broker loads/node capabilities. Specifically, it attempts to optimize the event delivery path when a mobile user moves out of range of its old network access point by 'opportunistically' following the network level handoff [25, 35] with an application-level handoff of the user's brokering functionality to a broker that is 'closer' to the new client location, hence the term 'opportunistic' event channel. The potential load imbalances across brokers caused by such middleware-level handoffs are addressed by dynamic broker creation. In effect, we aim to dynamically construct portions of the event dispatch trees used in static broker architectures [24].

Opportunistic channels are realized with the JECho peer-to-peer pub/sub infrastructure [38]. A unique extension of JECho is that its current peer-based broker infrastructure (i.e., each peer runs its own broker) is enhanced with dynamically created 'third party' brokers, which can run on machines and/or in address spaces not used by event publishers or subscribers [38]. Moreover, generalizing the capabilities of other event systems, brokers perform tasks in addition to the event routing permitted by other systems [24], using subscriber-provided functions, termed *event modulators* [37]. The intent is to address the severe resource limitations existing in many mobile and embedded systems, by permitting event consumers to deploy application-specific functions that manipulate event content into event sources and/or brokers, so as to precisely meet their current needs, and to avoid needless data transfers [38]. Finally, since JECho is constructed with Java, the realization of opportunistic channels presented in this paper has some limitations, including the need for a reasonably sized Java footprints on the target mobile nodes [38]. Opportunistic channels, however, also

benefit from certain JECho functionality, such as the relative ease with which third party brokers may be created, deployed, and migrated.

The performance advantages attained from the use of opportunistic channels can be substantial. For instance, simulation results indicate that the cost of sending a message from a fixed publisher to a mobile subscriber can be improved by up to 50%. Experimental results show that the end-to-end latency can be improved by 20% with microbenchmarks, when a mobile devices uses a 802.11a-based wireless network, where access points are connected via a high performance network backbone. Furthermore, broker load balancing can deliver events at two times the speed of a non-load balancing solution when the number of subscribers reaches 16, as demonstrated with microbenchmarks. Finally, even for a moderate number of video players (i.e., 8 players), load balancing and dynamic broker creation permits video playout to be improved by a factor of 3.

In the remainder of this paper, Section 2 briefly reviews the JECho event system, clarifying the basic software architecture of opportunistic channels. In Section 3, opportunistic channels are used in a mobile environment, including experimental measurements with campus wireless network. Section 4 describes load balancing in opportunistic channels. Related work is discussed in Section 5, and conclusions and future work appear in Section 6.

2 System Architecture

2.1 Overview of JECho

JECho implements a publish/subscribe communication paradigm, providing interactive services to distributed, concurrently executing components via event channels. JECho's efficient implementation enables it to move events at rates higher than other Java-based event system implementations [38, 37]. In addition, using JECho's modulator concept [38, 37], individual event subscribers can dynamically tailor event flows to their own needs, and adapt to runtime changes in component behaviors and needs and/or changes in platform resources. JECho's implementation is in pure Java, its group-cast communication layer is based on Java Sockets, and it runs with both standard and embedded JVMs. A *modulator* is a Java object that executes in a source's or broker's address space, on behalf of some client. The basic communication model in JECho is shown in Figure 3.

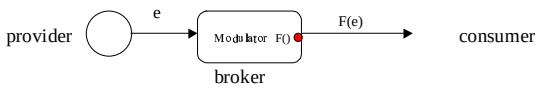

Fig. 3. JECho Communication Model

Client-defined customization in modulators, including event conversion or transformation, is performed prior to delivering the event to the consumer. Event conversion may reduce their sizes and hence reduce network traffic. Modulators

can also be used for offloading computation from constrained mobile devices. An example is a modulator that pre-converts events to the forms needed by specific client's graphical displays, thereby eliminating the costs of data conversion in the client. In fact, it is sometimes impossible to display data appropriately without performing such conversions, as with rendering OpenGL-based graphical data on a PalmTop or when applying server-resident business rules to data prior to its display on cellphones [6]. Other reasons for such conversion include the delivery of data with certain Quality of Service or to conserve power on battery-limited handheld devices [28].

2.2 Broker Adaptation Architecture

The notion of *opportunistic channel* (OC) generalizes upon JECho's modulators in three ways. First, OC modulators can run in consumers, intermediate brokers, or in providers. Second, brokers may be created dynamically. Third, for load balancing and migration, each broker is associated with a Java object called an *adaptor*. The idea is for a modulator and adaptor to adapt event dispatching to the dynamic needs of the clients, as shown in Figure 4.

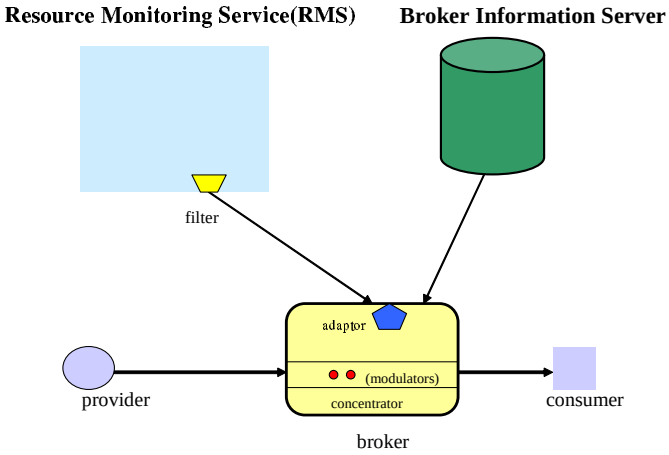

Fig. 4. Opportunistic Channel Architecture

The adaptation framework constructed with brokers and adaptors utilizes a resource monitoring service (RMS), which collects, aggregates, processes, and delivers data about the current execution environment. The RMS is able to monitor any nodes in the mobile system, using daemon processes connected via their own communication links. Monitoring data is captured from objects (including devices) at these nodes, and it is distributed to RMS clients via a customizable push-based interface. As with other monitoring systems [12], such data is contained in events, with current events containing information about CPU loads, memory availability, handoff actions occurring at the network level (to capture client mobility), and application-level communication latency and bandwidth

experienced between RMS clients. RMS clients can selectively register their interests in the data being captured, by providing a filter function installed at the source of a monitoring data stream. For example, a broker may provide a function that evaluates captured data and then provides it with a notification event only when the network topology changes and when the broker's own current CPU load exceeds 80%. Our current prototype implementation uses a central RMS server.

Another component of the architecture is a broker information server, which maintains information about current brokers, including their names, certain attributes (e.g., CPU and memory loads, IP addresses, network-related parameters,etc.) and their interconnections. A broker can get these information by sending a query to the broker information server.

A sample adaptor used by a broker is shown below. Its method 'subscribe()' registers the broker with the RMS and specifies its interest in certain monitoring data, using the object 'filter'. Adaptation code is implemented in the event handler method 'push()', which is invoked whenever an interesting event is received. Using adaptors, RMS, and filters, system developers can create potentially complex adaptation policies [34].

```
public class MyAdaptor implements BrokerAdaptor {
             //join the resource channel
              public void subscribe( ) {
                registerToRMS("CPU_Load", filter);
              }

              public void push(Object e) {
                     //adaptation code
              }
```

In summary, the adaptation model used for realization of opportunistic channels is seamlessly integrated with the JECho publish/subscribe system. Adaptation involves broker registration with a resource monitoring service and the use of potentially broker-specific adaptors. The adaptor can perform simple tasks like recording certain resource changes, to complex adaptations like changing the topology of broker interconnection.

3 Opportunistic Channels in Mobile Environments

While the concept of opportunistic channels may be applied to a wide variety of wireless networks, this paper assumes that there exists a reliable underlying network with guaranteed connectivity, where links are subject to dynamic variations in available communication bandwidth. In the presence of end user mobility, connectivity is guaranteed by Mobile IP, which performs network-level handoff actions [25, 17, 35]. The resulting transparent mobility exists in both deployed and experimental systems, including Berkeley's BARWAN and CMU's wireless Andrew systems [20, 15], which provide such support within a network (horizontal handoff) and across heterogeneous networks (vertical handoff) across

substantial geographical areas. For simplicity, in the remainder of this paper, we ignore any transient disconnection, which may happen during mobile suppliers' or consumers' migration. Furthermore, brokers are pre-created onto all nodes of the underlying network, thereby ignoring the costs of dynamic broker creation.

3.1 Mobility Adaptation

The network topologies of mobile systems change in response to mobile users' movements. When using base stations, this implies that the router (network access point) to which a mobile host connects may change when the mobile host moves across network boundaries. In Figure 5, M is a mobile host whose home network is network-2. M receives events from a source residing in network-1. There are two brokers in the system: broker-2 in network-2 and broker-3 in network-3. When M first subscribed to the event channel, it specified broker-2 as its broker and placed its modulator into broker-2. The event was delivered from the source to broker-2 through the network and finally reached M. The resulting delivery path was network-1(source)→network-2(broker-2→M). At some point in time, M moved from network-2 to network-3. If M still used broker-2, the resulting delivery path would be network-1(source)→network-2(broker-2)→network-3(M). However, if M changed its broker from broker-2 to broker-3, then the event delivery path would become network-1(source)→network-3(broker-3→M), which is much shorter.

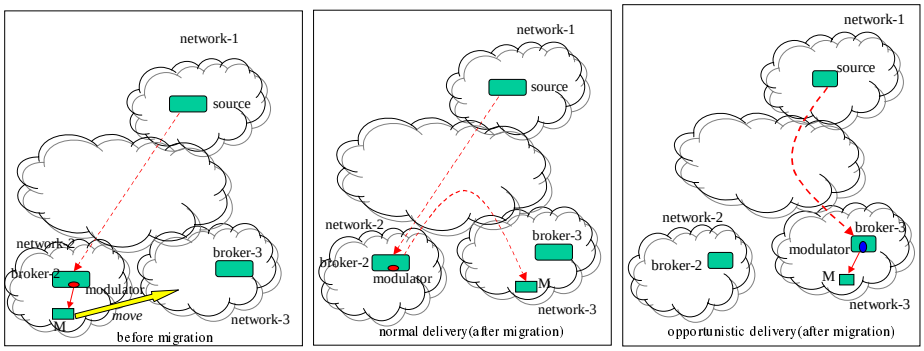

Fig. 5. Event Delivery in a Mobile Environment

In most cases, using the old, longer dispatching path (i.e., triangle delivery) results in higher event delivery latency and in the consumption of additional, often scarce network capacity. A solution is to change event delivery paths along with changes in underlying network message delivery. In particular, when a mobile consumer moves out of range of its previous network and into the range of a new one, the mobile host should connect to the broker that is 'closest' to the new network access point. This is the basic idea of opportunistic channels (OC): to 'opportunistically' move broker activities as end users move. Specifically, using the adaptation mechanism described in Section 2.2, a simple adaptor transparently relocates the mobile client's modulator to the new broker in response to a

network-level handoff, the latter detected via online monitoring. The basic steps performed by this adaptor are:

1. search for an alternative broker via the broker information server;
2. send a handoff request to the remote broker; and
3. if the remote broker agrees to accept the request, then execute the broker-level handoff protocol.

The modulator handoff protocol, described in detail in Section 3.2 and implemented by the adaptor, ensures that the dynamic handoff of a modulator is performed without losing or duplicating events. The broker information server lists the available brokers, including their names and certain attributes (e.g., CPU and memory loads, IP addresses, network-related parameters,etc.). The current implementation uses a central broker information server. More detail on scalable directory services appears in [2].

Two different adaptors are implemented and evaluated in this research. The 'simple' adaptor always performs modulator handoff, with the intent of using the broker that is closest to the current location of the mobile client. A more realistic adaptor takes into account additional factors. First, it may also consider the new broker's load. Second, it evaluates the network path to the new broker. This is important because the latency from network-1 to network-3 may actually be larger than the latency of network-1 to network-2 plus network-2 to network-3 shown in Figure 5. Specifically, the 'complex' adaptor evaluated next compares the old path with the potential new path when network handoff is detected. It schedules the modulator handoff only when the new path is shorter than the old one. In the current implementation, latency is measured by sending a ping' message between two nodes and using one-half of the round-trip time as an approximation. The source of events can be achieved from broker information server.

3.2 Modulator Handoff

Modulator handoff is illustrated in Figures 6. Our algorithm guarantees correctness properties that include (1) in order event delivery, (2) no lost or duplicate events, and (3) consistent modulator state in the presence of migration:

1. The source broker initiates a handoff by sending a HANDOFF request to the destination broker.
2. Upon receiving the handoff request, the destination broker adds the mobile client to its consumer list and sends an ACK to the source broker.
3. After receiving the ACK from the destination broker, the source broker sends a DETOUR request, which includes the name of the destination broker, to all of the event providers.
4. Upon receiving the DETOUR request, each provider atomically replaces the source broker from its consumer list with the destination broker; it then sends an ACK to the source broker and the destination broker.
5. The source broker receives events from each provider, applies the client's modulator to these event, and forwards them to the client until it receives the ACK from the provider.

6. The destination broker buffers events from each provider after it receives the ACK.
7. After ACKs from all providers are received by the source broker, it removes the client from its consumer list and sends a HANDOFF along with the current modulator to the destination broker.
8. Upon receiving the HANDOFF, the destination broker applies the modulator received from the source broker to the buffered events and starts forwarding events to the client.

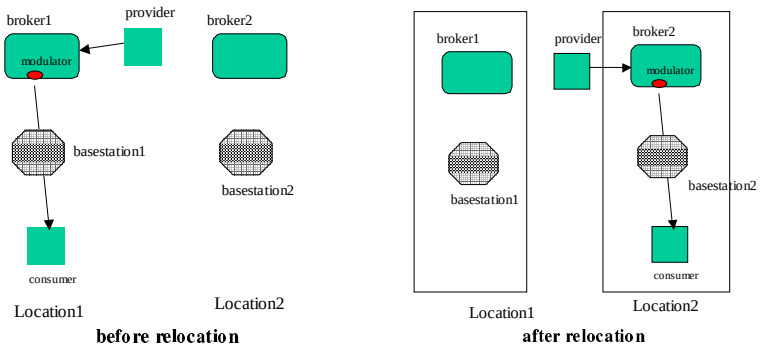

Fig. 6. Modulator Relocation

This protocol ensures that no events are lost or duplicated, since the old broker processes all events from a provider before receiving the provider's ACK mark, while the new broker processes only those events after the ACK message. The order of events from the same provider is maintained, since both the old broker and new broker process the events in the order delivered by the provider. Since modulators are stateful and since event processing may change their states, the protocol described above also ensures the consistency of modulator state. Finally, the algorithm is non-disruptive, in that the relocation procedure does not directly affect the other consumers of the providers handled by a certain broker. This is because it requires neither the providers nor the brokers to temporarily stop event delivery.

3.3 Experimental Evaluation

This section establishes the basic performance of opportunistic channels, and it demonstrates that opportunistic channels can improve the performance of event delivery in mobile environments.

Relocation Overhead When measured on a cluster of 300Mhz Pentium II Linux PCs connected with dual 100Mbps fast Ethernet, the times required to complete a relocation for a varying number of suppliers (see Figure 7) demonstrate that relocation cost increases linearly with the number of event suppliers. This is explained by the underlying peer-to-peer communications occurring in

JECho, where brokers are integrated with event suppliers unless otherwise specified. The intent, of course, is to separate the broker infrastructure from the remainder of the JECho system, so that solutions like opportunistic channels can specialize their broker behavior for the target environments being addressed. Interestingly, even for the simple peer-to-peer interconnections, total relocation cost remains small even for a moderate number of suppliers (i.e., 16 suppliers). In these measurements, each supplier is located on a different machine, and the event consumer moves across two Linux machines in the same cluster.

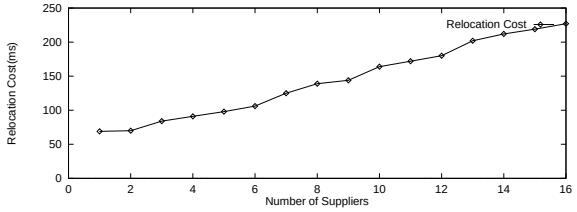

Fig. 7. Relocation Overhead

Simulation Results For the purpose of simulation, a 100 node network is generated using the BRITE (Internet topology generator developed at Boston University [21]). Nodes are assumed to be geographically distributed, and each link is assigned a cost corresponding to the distance between the nodes it connects. The cost of sending a message between any pair of nodes is the cost of the shortest path between the nodes, computed with Dijkstra's algorithm. There are multiple fixed sources and one mobile host in the network, all located on network nodes. Brokers also reside on these nodes. In the simulation, the mobile host stays on the same node for some varying time period, which is generated from an exponential distribution. When the period expires, the mobile host moves to the 'next' location, according to the mobility model used. In the simulation, four mobility models characterize different user behaviors. The random model chooses the next location randomly. The traveling salesman model moves to one of its neighbors, which is randomly chosen. The pop-up model is similar to the traveling salesman one, except that it sometimes roams to a randomly chosen remote node. The fixed model has two fixed locations which the mobile host visits more frequently than others. The mobile host moves between these two locations most of the time. Occasionally, the mobile host roams to a different node. More detail about the client behavior models used in our experimentation appears in [30].

The simulation uses a mobility-to-communication ratio to measure the mobile host's movement speed. This ratio captures the rate at which a mobile host changes its location to the rate at which it receives messages.The higher the mobility-to-communication ratio, the faster the mobile host's movement. For each relocation, adaptation cost is measured by multiplying the number of messages used by the relocation protocol by the distance between suppliers, consumer, source brokers or destination broker.

The first simulation uses one fixed source and a single mobile client. We compare the performance of using a 'normal' channel, a 'simple' opportunistic

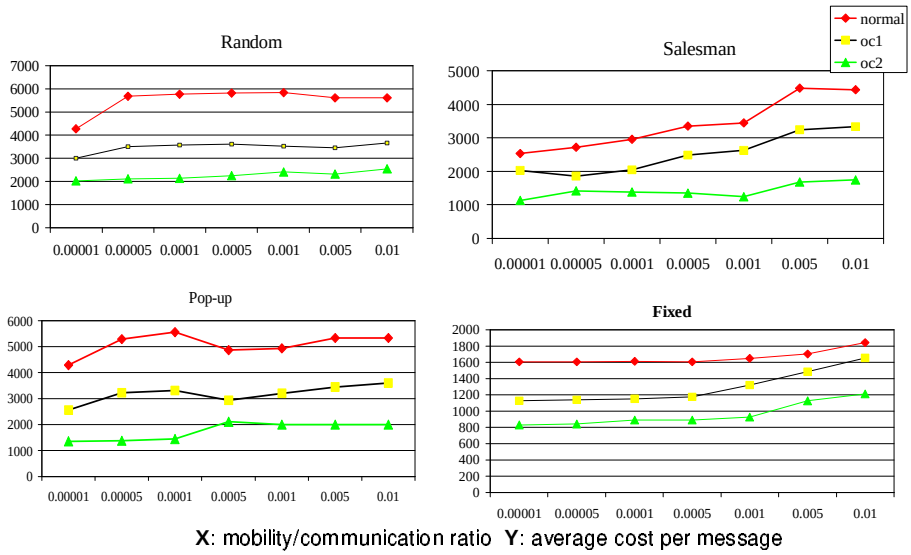

Fig. 8. Simulation Results for Single Supplier

channel (oc1) and a 'complex' opportunistic channel (oc2). oc1 uses a simple adaptor, which always performs a modulator handoff when the mobile client moves to a new location. oc2's adaptor compares the distances of old path and new path when the mobile client moves to a new location. If the new path is shorter, the adaptor relocates the modulator to the new broker. Otherwise, the old broker is used. Figure 8 shows the average cost per message of three channels, respectively, using the random model, traveling salesman and fixed model. In the figure, the X-axis indicates the mobility/communication ratio, and the Y-axis indicates the average cost of receiving a message. The results show that opportunistic channels achieve better performance than normal ones, even for random walks with very high mobility. In all cases, complex opportunistic channels (oc2) perform better than simple ones (oc1). In general, when the mobility/communication ratio increases, relocation cost and hence, average cost, increases accordingly.

The second simulation involves five fixed sources and one mobile client. The results are shown in Figure 9. They demonstrate that the opportunistic channel with the complex adaptor has the best performance, and that even the simple adaptor still outperforms non-opportunistic channels.

End-to-End Delay To substantiate our claims, experiments are performed with a microbenchmark used in three different locations on the Georgia Tech campus, as shown in Figure 10. The achieved end-to-end delays using opportunistic vs. normal channels are compared, with an event publisher running on a SUN Ultra 30 machine (H1) at subnet1. An IPAQ is the event consumer, and it accesses the network via 802.11a access points (802.11a devices offer a maximum of 56Mb bandwidth). Initially, the mobile client runs in subnet2 and connects to the event broker running on H2, with network connectivity provided via a wire-

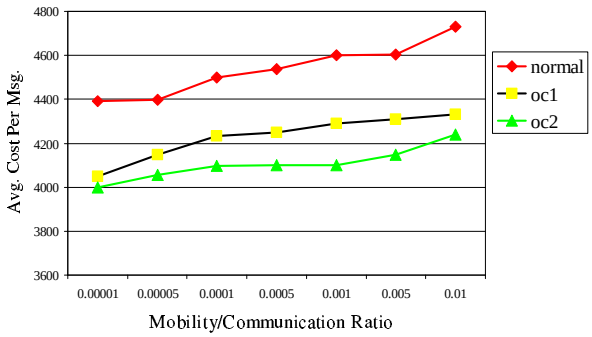

Fig. 9. Simulation Results for Multiple Suppliers

less network access point attached to subnet2. When the mobile client moves to another location (subnet3), with non-adaptive channels, the mobile client continues to connect to the old broker (H2), whereas the opportunistic channel uses a local broker (H3). With opportunistic channels, the mobile client and its broker have access to the same local network (subnet2), hence resulting in a shorter path between them. The experiment is performed at midnight in order to avoid high levels of variation in network usage. The results presented here are the average performance achieved over 50,000 events. The standard deviation for the measurements is less than 10% of the average latency.

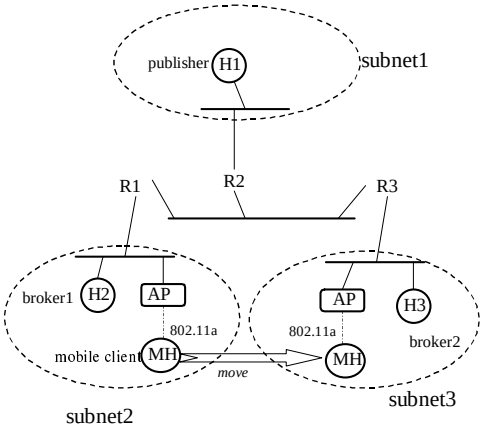

Fig. 10. Experiment Configuration for End-to-End Delay

The experiment measures the end-to-end delay with data of size 25KB in two different locations, as shown in Figure 11. Since the opportunistic channel can always connect the mobile client to a local broker, the latency is nearly identical after moving to new location. In comparison, with the non-opportunistic channel, latency increases by more than 20% after migration. This is due to the fact that event delivery for a non-opportunistic channel follows the path

subnet1→subnet2→subnet3, whereas an opportunistic channel directly delivers the event from subnet1 to subnet3. The figure also shows a breakdown of the costs involved in event latency. The portions labeled 'wired' represent the latencies from the event publisher to the wireless access points. The 'wireless' times are the delays in transmitting data from access point to mobile client. This breakdown is useful for understanding the performance differences between non-adaptive and opportunistic channels. Namely, in all of these scenarios, the wireless times are nearly identical, since all network accesses use the same 802.11a devices. The difference is the time from the publisher to the access point, that is, the 'wired' times. In this scenario, therefore, the opportunistic channel optimizes 'wired' times.

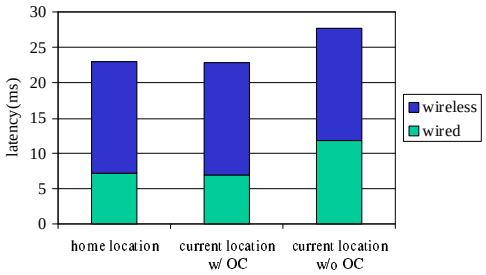

Fig. 11. Microbenchmark End-to-End Delay

Clearly, even in realistic scenarios within a campus like Georgia Tech, performance advantages can be derived from using opportunistic channels. The numeric benefits attained here are somewhat small, however, because the dominant factor in the scenario studied is the communication between broker (access point) and the mobile client. Even with 802.11a, this wireless communication is more than 2 times slower than the wired communications. This will not be the case for wide area mobility or in adhoc wireless networks, with potentially large delays between the publisher and the broker. Actually, in ad-hoc wireless networks, the delay between a publisher and a broker should be comparable to the delay between a broker to a mobile client, since both communications use wireless links. Finally, even the relatively modest 20% improvement in latency shown here can be important to applications that require real-time response. Examples include web portals [27] and the interactive virtual workbench for remote machine monitoring and control [6] evaluated in our previous research.

4 Load Balancing in Opportunistic Channels

Mobile hosts are known to be limited in computing power, storage, display, network bandwidth, etc. One approach to dealing with such limitations is per client service differentiation, that is, to customize service delivery to the needs of individual clients. JECho enables such customization by deploying modulators into event sources and/or brokers, where modulators range from simple event filters

to complex time-consuming event transcoding engines. With consumers and suppliers joining and leaving dynamically, broker loads are also subject to runtime variation, including overloads caused by modulator migration in opportunistic channels. One reason is the arrival of a large numbers of local users, as when many mobile units converge, such as during meetings. Another reason is the use of complex modulators by 'thin' clients, such as modulators that implement the flexible data transcoding required by such clients [38].

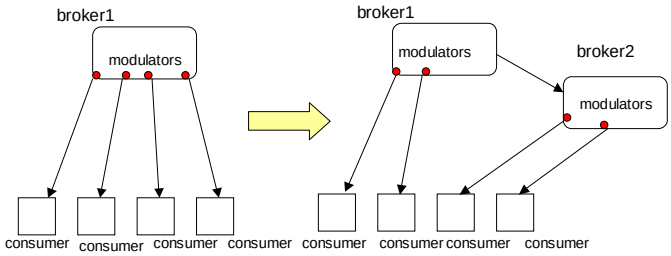

Fig. 12. Modulator Relocation for Load Balancing

4.1 Adaptations for Load Balancing

Opportunistic channels use a load balancing adaptor that permits a broker to distribute its load (i.e., modulator execution) to a less heavily loaded broker. This adaptor works as follows. The broker monitors the total execution time of all modulators in its address space, using broker-resident timers maintained by the resource monitoring service (RMS) described in Section 2. When the broker's modulator execution time exceeds some limit, which can be specified when the broker is started, it will try to find a less loaded broker and relocate some of its load (i.e., modulators) to that broker. In the current implementation, adaptors simply contact a directory service to identify a lightly loaded broker. The directory service finds a lightly loaded broker as follows. First, it has information about each broker's location, current load, status indicating whether it can receive the load balancing request and network distances between all pairs of brokers. Second, using the RMS, it collects information that includes remote brokers' CPU loads, modulator execution times, and total execution times. The frequency with which such monitoring information is exchanged is also set when the broker is started. Third, given this information, the directory executes the following, simple algorithm to identify a suitable broker:

- check all available brokers in the order of the distance from the requesting broker;
- if a less loaded broker is found, stop the search and return the broker's id;
- if no such broker exists, terminate with failure.

The broker initiating the request contacts the target broker and executes the relocation protocol described in Section 3.2. To reduce the frequency with which relocations are performed, our current implementation simply relocates half of the modulators from the overloaded broker to the target broker. The result of these actions is depicted in Figure 12.

4.2 Experimental Evaluation

Benchmarks Figure 13 compares the performance of event distribution for opportunistic vs. normal channels. The events being passed are arrays of 100 floats. A for() loop is used to emulate the behavior of complex modulator runtimes. When brokers and consumers reside on different machines, with opportunistic channels, one intermediate broker is added after 6 consumers have joined the channel, and a second broker is added after 14 consumers have joined. In comparison, with the non-opportunistic channel, all consumers use the same broker for the entire duration of the experiment. The results clearly establish the importance of dynamic load distribution across brokers.

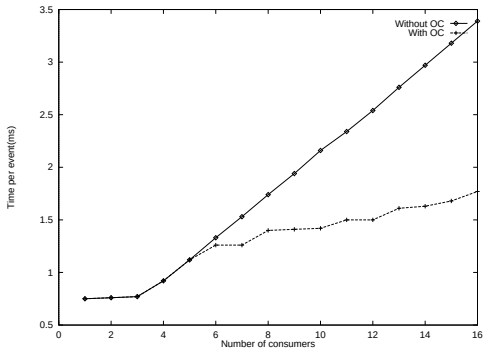

Fig. 13. Load Balancing Microbenchmark Results

Performance of a Video Player A more realistic experiment is one that measures the frame rate of a video player, varying the number of players and comparing the results with a single broker vs. dynamically 'split', multiple brokers. In the experiment, a video server runs on a Linux machine and multiple players each running on a different Linux machine receive video stream from the server. The results are shown in Figure 14. For 8 players, opportunistic channel permits the frame rate to be improved by a factor of 3.

5 Related Work

5.1 Publish/subscribe Systems

Most current publish/subscribe systems permit subscribers to specify their interests, storing these interests at providers or intermediate brokers. Typical interest expressions result in predicate-based event filtering, sometimes also enabling limited forms of event transformation. However, most such subscription

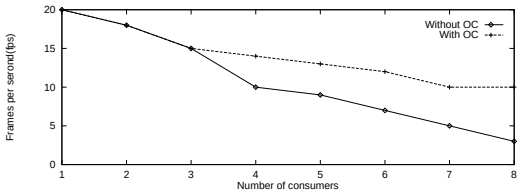

Fig. 14. Frame Rate Comparison

are stateless and therefore, do not support the general event processing needed for the complex data conversions occurring in multimedia, business, or scientific applications. In addition, a client's subscriptions cannot be changed at runtime. JECho and opportunistic channels generalize upon these systems' capabilities in our support for dynamic broker creation, load balancing across brokers and most importantly, in our ability to make broker behavior platform-aware, by dynamically monitoring and adapting to changes in platform capabilities and client behaviors.

There has been substantial research on interconnection topologies, event routing, and event matching (to eliminate redundant event transmission) in order to scale publish/subscribe systems to large numbers of publishers, subscribers, and events. Typically addressing wide area networks (e.g., Gryphon [24], SIENA [3], JEDI [7], READY [11]), The main goal in such work is to 'send' an event only towards brokers with consumers that are currently interested in that event, and to use shortest such path available. In comparison to our work, fixed topologies are used for broker interconnection, and there is no dynamic support for changing topologies or broker locations. More specifically, JEDI offers moveout and movein operations to change its dispatching system somehow, but the applications have to explicitly call those operations [8]. Elvin supports disconnection and reconnection in a mobile environment by using proxies, but the mobile client must connect to the same proxy even after it moves to a new location [33]. Neither system provides general adaptation support for dynamic reconfiguration and redeployment of the event system itself. [16] discusses general ideas about how to adapt a publish/subscribe system to mobile environments, but does not describe an implementation. Solar [5] provides a flexible and scalable data-fusion infrastructure for delivering context information in ubiquitous computing systems. Its concept of operator is similar to modulators in our systems, but modulator in JECho can perform more general operations. Furthermore, Solar focuses on how to delivery context information efficiently, whereas JECho is a general event system.

5.2 Mobile IP and its Location Management

Optimization of location management for Mobile IP [19] is similar to the way in which opportunistic channels operate. Namely, both attempt to optimize event routing paths after a mobile host changes its location, by changing the 'triangle' delivery to direct delivery. Multiple handoff algorithms have been proposed for Mobile IP [19, 13], all of which require location registration with clients'

home agent or a regional agent. In comparison, opportunistic channels use directory services in place of home or regional agent, which implies that handoff performance depends on that of the directory service. Furthermore, modulator handoff is somewhat more complex than link handoff, especially for stateful modulators. Finally, since opportunistic channels are realized on top of Mobile IP, its performance benefits and overheads depend on the underlying Mobile IP implementation.

5.3 Adaptations in Mobile Environments

Several network-level efforts address mobility issues. As already mentioned, Mobile IP (e.g., Mobile IPv6 [25, 17]) enables a mobile user to stay connected when moving across network boundaries without changing its IP address. Berkeley's BARWAN project [20] provides seamless roaming across heterogeneous networks. Furthermore, BARWAN enables data forms to be changed to suit end system or wireless network limitations, by permitting application-level, type-specific data transformation and data compression [20]. JECho offers the same functionality, but in addition, provides dynamic support for handler partitioning [37]. Zhao, Castelluccia and Baker [36] describe a general-purpose mechanism at the network level, which supports multiple packet delivery methods and multiple network interfaces, where the system adaptively selects the most appropriate method and interface. Similarly, the CMU Monarch project aims to enable adaptive mobile host communications, to make the most efficient use of the network connectivity available to the mobile host at any time [18]. The adaptors used in opportunistic channels would interact with such network-level mechanisms. Resilient Overlay Networks [1] optimize application-specific routing metrics, by monitoring the functioning and quality of network paths. Opportunistic channels could implement the same mechanisms, if appropriate. Finally, there are several TCP enhancements for wireless networks [4].

Our work focuses on middleware-level and application-level adaptations, and it can benefit from the network level research mentioned above. Since such adaptations require information about network-level changes, we can also benefit from the substantial ongoing work on real-time network monitoring, including the work performed in the Monarch [18] and net100 [22] projects.

At application-level, the Odyssey projects extends the Unix System call interface to support flexible application-aware adaptations [23], as also done in our own work addressing interactive applications [29]. The system monitors resource levels, notifies applications of relevant changes, and enforces resource allocation decision. Each application independently decides how best to adapt when notified. This is similar to JECho's adaptations where an application can be notified of resource changes and responds to such changes according to its adaptation strategy defined in a modulator.

Some adaptations based on application semantics can be provided only at application level. For example, datatype-specific data transformation and data compression must depend on the application. Similarly, HRL's Intelligent information dissemination services use bandwidth-aware filtering to adapt infor-

mation streams to resource bandwidth availability [32]. Our own previous work with JECho has addressed these problems by supporting general application-level adaptations via the dynamic deployment and partitioning of event modulators [38]. Opportunistic channels, as well as our related research on coordinating network- with application-level adaptations [14] complement these efforts, by providing models and mechanisms for linking network- or broker-level changes in resource availability with suitable application- and middleware-level adaptations.

6 Conclusions and Future Work

This paper describes how a content-based event dispatching system may be extended to cope with end user mobility and with the changing computational resources and network conditions of mobile systems. Our approach is to dynamically deploy and reconfigure the underlying event distribution infrastructure (i.e., the 'broker' overlays). The approach is realized with the novel abstraction of *opportunistic* event channels. Their implementation involves dynamically created event brokers, the runtime installation of event modulators in brokers, and the dynamic relocation of such modulators. Adaptation policies are realized by adaptors that interact with a runtime resource monitoring system. Essentially, opportunistic channels implement a middleware-level analogue of the channel handoff protocol used in wireless communications. More importantly, by coordinating network- with middleware-level handoffs, opportunistic channels can attain substantial performance improvements over non-adaptive event channels. Such improvements are due to their use of shorter network paths and the better balancing of loads across event brokers.

Future work should address some deficiencies of our current implementation, as well as generalize upon the basic concept of opportunistic channels. First, the current implementation assumes that there is only one intermediate broker between a provider and a consumer and therefore, cannot handle multi-level broker topologies. We are extending our implementation and algorithm to address this limitation. Second, our current experimentation uses wireless networks that employ base stations, with only the last hop being a wireless link. Future work will use opportunistic channels over ad-hoc wireless networks. In addition, the current implementation assumes a reliable network environment and therefore, does not consider dynamic disconnection and reconnection. Our intent is to add application-specific failure recovery to brokers, an example being the mirroring and failure recovery described for business transaction systems in [10].

References

1. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles(SOSP-18)*, October 2001.
2. F. E. Bustamante. *The Active Streams Approach to Adaptive Distributed Applications a nd Services.* PhD thesis, Georgia Institute of Technology, 2001.

3. 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(PODC 2000)*, pages 219–227, Portland, Oregon, July 2000.
4. M. C. Chan and R. Ramjee. Tcp/ip performance over 3g wireless links with rate and delay variation. In *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking(MobiCom 2002)*, Atlanta, GA, September 2002.
5. G. Chen and D. Kotz. An open platform for context-aware mobile applications. In *Proceedings of the First International Conference on Pervasive Computing (Pervasive 2002)*, August 2002.
6. Y. Chen, K. Schwan, and D. W. Rosen. Java mirrors: Building blocks for remote interaction. In *Proceedings of the 2002 International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.
7. G. Cugola, E. D. Nitto, and A. Fuggetta. The "jedi" event-based infrastructure and its application to the development of the opss wfms. In *IEEE Transactions on Software Engineering in 2001*, 2001.
8. G. Cugola, E. D. Nitto, and G. P. Picco. Content-based dispatching in a mobile environment. In *In Workshop su Sistemi Distribuiti: Algorithms, Architecture e Linguaggi (WSDAAL)*, 2000.
9. G. Eisenhauer, F. Bustamente, and K. Schwan. Event services for high performance computing. In *Proceedings of High Performance Distributed Computing-9(HPDC-9)*, August 2000.
10. A. Gavrilovska, K. Schwan, and V. Oleson. A practical approach for 'zero' downtime in operational information systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS-22)*, July 2002.
11. R. Gruber, B. Krishnamurthy, and E. Panagos. The architecture of the READY event notification service. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems Middleware Workshop*, Austin, Texas, USA, May 1999.
12. W. Gu, G. Eisenhauer, and K. Schwan. Falcon: On-line monitoring and steering of parallel programs. *Concurrency: Practice and Experience*, 10(9):699–736, August 1998.
13. E. Gustafsson and et al. *Mobile IPv4 Regional Registration.* draft-ietf-mobileip-reg-tunnel-05,IETF,September 2001.
14. Q. He, , and K. Schwan. Iq-rudp: Coordinating application adaptation with network transport. In *Proceedings of High Performance Distributed Computing-9(HPDC-9)*, July 2002.
15. A. Hills. Wireless andrew. *IEEE Spectrum*, 36(6), June 1999.
16. Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile environment. In *2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01)*, Santa Barbara, California, USA, 2001.
17. D. Johnson and C. Perkins. *Mobility Support in IPv6.* Internet Draft, IETF, draft-ietf-mobileip-ipv6-12.txt(work in progress), September 2000.
18. D. B. Johnson and D. A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Communications*, 3(1):34–42, 1995.
19. D. B. Johnson and C. Perkins. *Route Optimization in Mobile IP.* In Internet Draft(work in progress), 1998.
20. R. Katz and E. Brewer. The case for wireless overlay networks. In *SPIE Multimedia and Networking Conference*, January 1996.
21. A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: An approach to universal topology generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01*, Cincinnati, Ohio, August 2001.

22. Net100. *The Net100 Project - Development of Network-Aware Operating Systems*. www.net100.org.
23. B. D. Noble, M. Satyanarayanan, D. Narayannan, J. Tilton, J. Flinn, and K. Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles(SOSP-16)*, October 1997.
24. L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, , and D. Sturman. Exploiting ip multicast in content-based publish-subscribe systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing(Middleware 2000)*, April 2000.
25. C. Perkins. *IP Mobility Support*. IETF, Request for Comments 2002, Oct., 1996.
26. T. Phan, L. Huang, and C. Dulan. Integrating mobile wireless devices into the computational grid. In *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking(MobiCom 2002)*, Atlanta, GA, September 2002.
27. M. Pierce, C. Youn, and G. Fox. The gateway computational portal: Developing web services for high performance computing. In *Proceedings of 2002 International Conference on Computational Science(ICCS2002)*, April 2002.
28. C. Poellabauer and K. Schwan. Power-aware video decoding using real-time event handlers. In *Proceedings of the 5th International Workshop on Wireless Mobile Mult imedia (WoWMoM)*, September 2002.
29. C. Poellabauer, K. Schwan, and R. West. Coordinated cpu and event scheduling for distributed multimedia applications. In *Proceedings of the 9th ACM Multimedia Conference*, October 2001.
30. S. Rajagopalan and B. Badrinath. An adaptive location management strategy for mobile ip. In *Proceedings of the first annual international conference on Mobile computing and networking(MobiCom 1995)*, December 1995.
31. B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of A UUG97*, September 1997.
32. E. C. Shek, S. K. Dao, Y. Zhang, and etc. Intelligent information dissemination services in hybrid satellite-wireless networks. *ACM Mobile Networks and Applications(MONET) Journal*, 5(4), December 2000.
33. P. Sutton, R. Arkins, and B. Segall. Supporting disconnectedness - transparent information delivery for mobile and invisible computing. In *CCGrid 2001 IEEE International Symposium on Cluster Computing and the Grid*, May 2001.
34. R. West, K. Schwan, and C. Poellabauer. Scalable scheduling support for loss and delay constrained media streams. In *Proceedings of 5th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 1999)*, June 1999.
35. H. Yokota, A. Idoue, and T. Kat. Link layer assisted mobile ip fast handoff method over wireless lan networks. In *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking(MobiCom 2002)*, Atlanta, GA, September 2002.
36. X. Zhao, C. Castelluccia, and M. Baker. Flexible network support for mobility. In *Proceedings of The Fourth ACM International Conference on Mobile Computing and Networking(MobiCom 1998)*, October 1998.
37. D. Zhou, S. Pande, and K. Schwan. Method partitioning - runtime customization of pervasive programs without design-time application knowledge. In *Proceedings of the 23nd International Conference on Distributed Compu ting Systems (ICDCS-23)*, 2003.
38. D. Zhou, K. Schwan, G. Eisenhauer, and Y. Chen. Supporting distributed high performance application with java event channels. In *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS 2001)*, April 2001.