Infrastructure for Peer-to-Peer Applications in Ad-Hoc Networks

Nitin Desai, Varun Verma and Sumi Helal
Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611, USA
{nsdesai, vverma, helal}@cise.ufl.edu

***Abstract*—The proliferation of mobile devices and the pervasiveness of wireless technology have provided a new environment to deploy peer-to-peer systems. Existing peer-to-peer systems target very specific application space like sharing music files or instant messaging. They were mainly developed for wired connections and higher computing stationary devices like PCs. Thus they fall short of accommodating the complexities of ad-hoc environments. We introduce Konark, a service discovery and delivery protocol for ad-hoc, peer-to-peer networks. We also describe an implementation of Konark that provides an infrastructure to deploy generic peer-to-peer systems. Konark takes the help of an underlying network for peer naming and message routing. For resource discovery, Konark uses a completely distributed, peer-to-peer mechanism that provides each peer the ability to advertise and discover resources in the ad-hoc network. It has an XML based description template that allows resources to be described in a human and software understandable forms. A micro HTTP server present on each peer handles service delivery, which is based on SOAP.

Keywords: *peer-to-peer computing, Ad-hoc service discovery protocols..*

I. Introduction

The rapid increase in the usage of mobile devices and the accompanied maturity in wireless technologies such as 802.11 have made wireless networks almost as ubiquitous as traditional wired networks [HEL99]. An important consequence of such networks is ad-hoc networks. These networks are characterized by their lack of required infrastructure, decentralized operations and ease of formation. As the "deployment" and usage of such networks increases, new paradigms begin to emerge that incorporate and utilize these new communication and computing capabilities in ways that were previously not thought of or were impossible to implement. Some of the possible applications and scenarios for ad-hoc networks are as follows:

- Task oriented collaborative computing in emergency relief stations or in battlefield.
- An application on your handheld device that packages your personal expertise of being a web-security consultant as a service and advertises this service in the ad-hoc network formed in the relevant places like technical conferences. It also responds to searches for such services with more details such as your current location, your schedule, and how much you charge for a consulting session.
- The ability to share with others entertainment sources such as music and games available on your personal handheld device. Such a capability would be extremely useful while waiting for a flight in an airport lounge, or other similar situations of killing time in a public place.

All the above-mentioned scenarios are examples of mobile peer-to-peer applications over spontaneously formed ad-hoc networks. The resources being shared or searched are packaged as services. Services can be those offered by devices (e.g. printers, fax machines), or simply software services that are device independent. Even software services offer a myriad range of possibilities with opportunities related to an individual's career, entertainment or daily chores like ordering food.

The feasibility of above scenarios requires not only the formation of networks and packaging of resources as services, but also a resource (service) discovery and delivery mechanism suited to the needs of such networks. Low-level technologies necessary to form peer-to-peer, ad-hoc networks are available. The primary missing link is a higher-level framework and protocol, which will enable peers to discover, advertise and use the services over ad-hoc networks. We use the term resource and service interchangeably.

To address the opportunities, issues, and requirements discussed, we present Konark [DES02, VER02], a peer-to-peer based middleware designed specifically for discovery and delivery of device independent services in ad-hoc networks. Konark handles different P2P computing issues like peer naming, peer and resource discovery, resource-metadata handling, message routing and resource delivery. It thus provides an infrastructure to deploy mobile peer-to-peer applications over ad-hoc networks.

Napster [NAP01], Gnutella [GNU01] or JXTA [JXT01] are for stable environments like Internet. Unlike Napster or Gnutella that provide infrastructure for specific applications like sharing files, Konark supports a large number of possible services. Konark is platform independent, and provides an application development platform.

We start this paper by discussing some issues related to ad-hoc networks, service discovery protocols and peer-to-peer computing. These issues were considered while designing Konark. We then present the design and implementation, followed by suggested future work.

II. MOBILE P2P SYSTEMS AND KONARK ARCHITECTURE

The computing environment defined by small handheld mobile devices and ad-hoc networks will be vastly different from the traditional infrastructure based environment comprising of PCs with wired connections [BUS01]. While designing an infrastructure for peer-to-peer systems in an ad-hoc network, it is imperative to keep in consideration inherent features of ad-hoc networks. A whole another set of issues is also raised by the type and range of services targeted.

One of the primary differences between ad-hoc networks and infrastructure-based networks is the formation of the network itself. In ad-hoc networks, device participation is dynamic. It is not possible to assume that a controlling entity, whether an administrator or another device in the network, will be present to assign addresses to the nodes. Konark assumes an IP level connectivity among devices in the ad-hoc networks. It is a valid assumption considering the fact that the IP networks are almost ubiquitous. Most of the modern operating systems provide zero configuration techniques like Auto-IP [HAT01] that assign IP address to the device in the absence of administrative services. Since each node gets a unique identity, the problem of naming gets addressed. Such an assumption also makes Konark independent of the link layers, which could either be 802.11, IRDA, Bluetooth or any other protocol on which IP can be implemented.

Existing peer-to-peer systems like Napster provide infrastructure to share a limited kind of services. With the increasing popularity of handheld devices, and the support for mobile commerce, we envision an entirely new set of possible services. These services could vary from resources such as games or music, personal information such as professional expertise, information-oriented services such as maps or weather, or commerce based services like tickets to a movie. Users would be able to package their own resources or information as services and offer them to others. This requires services to have simple and rich description capabilities along with support for cross-platform lightweight usability. Konark defines an XML-based simple and rich description language to describe these kinds of services. This language is similar to WSDL and enables description of a wide range of services.

An important factor while designing resource providing systems is the storage of resources' metadata, that is, information about available resources. One possible approach is to have a centralized repository. Any node offering a service would register its service with this repository, and all peers seeking any service would query it for available services. For highly dynamic networks, it is fallible to assume presence of such a centralized repository. It also might not be possible to have algorithms such as leader election to choose a new repository each time the previous one moves out. Also, with devices being small and resource-less, it would be difficult for any device to maintain information about all services in the network. Konark uses a completely distributed peer-to-peer approach to solve this problem. It specifies that each peer will have a local repository that will be responsible for maintaining information about services being offered by that device.

Peer discovery enables nodes to locate one another and the resources (services) they offer. Nodes also need to make their resource information available to other peers in the network. Konark supports both advertisement and discovery of the resources. Discovery and advertisement use IP multicast to locate peers and discover/publish service-information. As the peers will be in physical proximity and thus in the same subnet, IP multicast is a good choice. If the ad-hoc network spans across multiple smaller networks, we assume network support for routing multicast traffic. By taking the help of the network itself, Konark eliminates the need for complicated infrastructure needed to perform application level message routing. This keeps the resource poor devices light-weight.

To make service information available to other devices in the network, peers can push their service-information into the network on a fixed multicast group. The rate of advertising depends on a lot of factors. It can be on a periodic basis or can be stimulated by events such as a new device joining the network. Advertising can be also based on other factors such as geographical or temporal information. These service advertisements contain time-to-live (TTL) information and help in self-healing of the systems. The client peers can cache this service information to use it later so that they do not have to locate the services again, thus reducing network traffic. This caching can be based on user preferences in the form of filtering techniques, or device capabilities like memory and available battery. If nodes offering services want to continue offering the service beyond the TTL, they have to refresh the advertisements. Each cached service entry requires minimal amount of memory as it stores only the name, TTL and URL of the service. When the peers need to locate services, they can use a distributed pull method to retrieve desired service information. This helps them to get real time services available in the network.

Handling the service metadata is a very important issue. Metadata may be organized as the different service categories. With the increase in the number and variety of services, locating a particular service (such as chess), or services of particular types (such as games), becomes increasingly difficult. Depending upon the network size and other factors, the information obtained about network services - either via actively discovering them or by passively caching the advertisements - can be in large quantities. Also, the kind of m-commerce oriented services being targeted requires a high level of user interaction. This makes it necessary to have a registry(repository) that stores and arranges the service information in an extremely manageable and user friendly fashion. Another advantage of this feature would be very effective advertising and discovery of service information based on categories.

To handle the above requirement, Konark presents a service registry based on a tree-structure. We use a basic tree skeleton with service classification levels that are generic at top and become more specific as we move down the tree levels. The nodes in the tree are not services themselves but act as placeholders for services in that particular category. We use XML to internally represent this basic tree, making it scalable so that new class of services can be easily added. We present the same tree to the users as an interface to help them easily manage and use the services interactively. We strongly believe that it is possible to come up with a basic service-tree structure that can accommodate most of the services needed by a mobile user. Standardization of this tree will help to realize this goal. The tree structure described above is not only used for maintaining local services and information about network services, but also for advertising and discovery. Discovery can be done at any level of the tree – the broader the range of services desired, the higher the level of the tree- using the tree-paths. Keywords can be used for discovering specific services. Advertising can be done at various levels of the tree – from generic advertising at the higher levels of the tree (such as all games) to very specific at the leaf node (e.g. chess). The use of the tree-paths during advertising and discovery helps to solve the service-matching problem considerably. Since the internal storage of the services can be mapped to the user interface, the quantity of information in the tree of the device can be controlled.

Yet another important design issue is the actual delivery of services. As more and more manufacturers push out handheld devices into the market, the heterogeneity of ad-hoc networks vis-à-vis platforms increases. To support this trend, it is critical to support cross-platform service delivery. Konark uses the widely accepted standards such as HTTP and SOAP to handle service delivery. It provides for a micro HTTP server on each device that can handle service requests from clients. The service requests and responses are based on SOAP.

III. DESIGN

Konark architecture consists of mainly two parts: Service Discovery and Service Delivery. Service Discovery deals with peer naming, service discovery/advertisement and metadata handling in the form service tree (service registry). Service Delivery deals with describing a resource using our XML-based service description language and delivering it using a micro-HTTP server resident on each peer, using standardized internet protocols like HTTP and SOAP. Each peer acts as a server and a client at the same time. We use the terms server-peer or client-peer whenever a particular peer acts like a server or client respectively.

A. Service Discovery

Each device has a Konark SDP Manager that is central to the service discovery mechanism. This acts like a broker to discover the required services, and also registers and advertises device's local services. Figure-1 shows the Konark service discovery protocol stack. Konark SDP Manager interacts with the messaging layer to send and receive the discovery and advertisement messages. The messaging layer is built above the transport layer. The different issues handled in this part are:

1) Peer naming and Message routing

As described earlier, each peer obtains an IP-address using zero-configuration technique like Auto-IP. These devices will then join a locally scoped multicast group (239.255.255.251) so that all peers in the network will receive any message sent to the group.

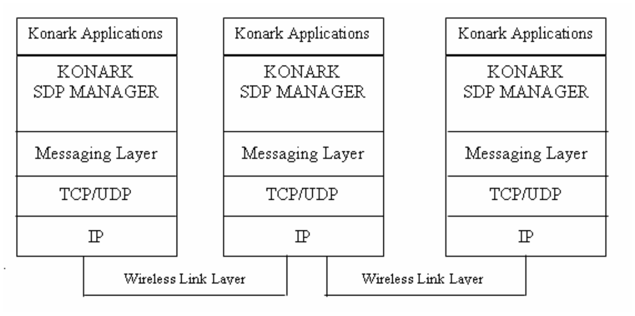

Figure 1. Konark Service Discovery Stack

2) Metadata Handling(Service Registry)

Metadata deals with possible categories of resources. Service registry is a structure that enables peers to store local services and cached network services. Each peer will have generic service categories arranged as a hierarchical tree. This service tree represents the classification levels where services become more specific from root towards the leaves. Figure-1 shows an example of such a service tree. The oval shaped nodes represent generic service types that form a basic service tree. Such a tree will be uniform and present across all devices implementing Konark architecture. The services shown in rectangles are the actual services registered locally or available network service(cached or discovered).

Actual services get added to the tree based on their category. Each service has a unique path from the root. This path helps in service matching during advertisment and discovery. The network services will be associated with TTL and will be deleted after that unless re-advertised by the corresponding peer. This tree-structure enables easy management of the services and very effective service discovery and advertisement as described in the next section.

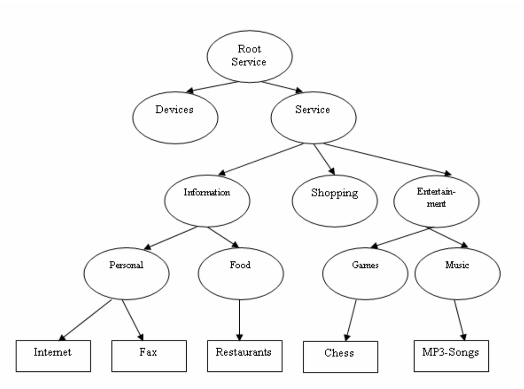

Figure 2. Example Service Tree

3) Service Discovery and Advertising

Resource discovery is done using *active pull* mechanism. The peers send out a service discovery message on a multicast group. (Figure 3) For generic service categories (e.g. games), the search message consists of registry tree-path, and for a specific service (e.g. chess) it consists of a keyword. The message also contains the port on which it listens for replies. The peers who receive this message do service matching using path or keyword. For generic service discovery based on path, the peers can have more than one registered service in their service tree. Peers then send a unicast response with a service advertisement message (Figure 4) for each matched service, to the port as specified in the discovery message.

Path or Keyword	PORT

Figure 3. Service Discovery Message

Peers use an advertisement process to periodically announce their registered services using *passive push.* The peer can advertise “all” its registered services, “generic” services identified by generic path in the service tree, or a “specific” service defined by an registered service leaf node (rectangular) in the tree. Each service is sent using the advertisement message that contains the actual service name, the path of the service from the root, the type of the service, the URL where the service description will be available and the time-to-live of the service specified in minutes.

Service Name	Path	Type	URL	TTL

Figure 4. Service Advertisement Message

On receiving a service advertisement message, other peers in the network add the information to their registry tree based on path, after considering user preferences in the form of filtering and device memory.

B. Service Delivery

Service delivery is a two-step process – service description where the client learns about the properties and capabilities of the service, and service usage where the client avails the capabilities.

1) Service Description Language

Under Konark architecture, each service is a bundle of two components – a service description file that describes a service, and a service object. The registry of any server-peer contains these two components for each service being offered by that device. While the service object can be in the form of a class file or a DLL, the description file is a plain text file containing complete information about the characteristics and functions of the service. Konark defines an XML based service description language to enable services to explain their characteristics. The language is loosely based on WSDL (Web Services Description Language) [W3C01], the emerging standard for describing web services.

2) *Delivery Architecture*

Service delivery primarily involves communication between a Client Application and the micro-HTTP server of the service provider. The client device invokes the URL obtained during the discovery phase to get more information about the services. A typical URL would be similar to http://169.254.30.42/music.xml where music.xml is a service description file. The service description file contains complete information about the properties of the service and the methods provided by the service. The user can interact with the service by invoking any of the available functions with proper parameters. The user function invocation is packaged as a SOAP request and sent to the HTTP-server.

IV. IMPLEMENTATION

Our protocol is independent of both operating systems and programming languages. We have implemented our protocol in two versions of Java, namely: Personal Java 1.2 and J2ME CLDC/MIDP. For this implementation, we used Pocket-PC 3.0 based iPAQs from Compaq and the Jeode VM from Insignia Inc. We used Lucent's wireless card for the 802.11 wireless interface to form an ad-hoc network among iPAQs. Our second implementation was for J2ME CLDC/MIDP platforms on devices like Motorola iDEN Phones.

Konark provides a platform in the form of APIs to build Konark mobile peer-to-peer applications. We also provide intuitive user interface in the form of a service tree which simplifies human-device interaction.

V. CONCLUSION AND FUTURE WORK

Konark provides an infrastructure and development platform to deploy P2P systems for ad-hoc networks. We did not consider security while designing the protocol. Security has to be considered at all levels of the protocol. Standardization of the basic generic service tree is also an important aspect to accommodate most of the services needed by a mobile user. Standardization of the Konark service description language is also necessary along with extending it to describe new variety of services. Finally, we invite the reader to tune to Konark's web site [KON] for updates of ongoing work.

VI. REFERENCES

[BUS01] D. Buszko, W. Lee, and A. Helal, Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration," Proceedings of the ACM International Conference on Supporting Group Work (Group'01), September 2001.

[DES02] Nitin Desai, *Konark – An Ad-Hoc Service Discovery Protocol*, Master's Thesis, University of Florida, Gainesville, August 2002.

[GNU01] Gnutella, www.gnutella.com

[HAT01] M. Hattig, *Zeroconf IP Host Requirements*, IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-reqts-09.txt, 2001.

[HEL99] A. Helal, B. Haskell, J. Carter, R. Brice, D. Woelk, and M. Rusinkiewicz, *"Any Time Anywhere Computing: Mobile Computing Concepts and Technology*," Kluwer Academic Publishers. ISBN 0-7923-8610-8, Published October 1999.

[JXT01] *The JXTA Project*, Sun Microsystems, http://www.jxta.org, 2001.

[KON] http://www.harris.cise.ufl.edu/projects/adhoc-srv-discovery

[KOR01] Gerd Kortuem, Jay Schneider, Dustin Preuitt, Thaddeus G. C. Thompson, Stephan Fickas, Zary Segall, *When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad Hoc Networks,* 1st International Conference on Peer-to-Peer Computing, August 2001.

[NAP01] Napster, www.napster.com

[NID00] Michael Nidd, *Service Discovery in DEAPspace*, Technical Report IBM Zurich Research Labs, Zurich, January 2000.

[ORA01] Peer-to-Peer: Harnessing the power of Disruptive Technologies, March 2001.

[SCH01] Rudiger Schollmeier, *A definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications,* 1st International Conference on Peer-to-Peer Computing, August 2001

[SUN99] Sun Microsystems, *JINI technology*: http://www.sun.com/jini, January 1999.

[UPN00] *UPnP Forum*, www.upnp.org, 2000.

[VER02] Varun Verma, *Konark – A Service Delivery Protocol*, Master's Thesis, University of Florida, Gainesville, August 2002.

[W3C01] W3C Consortium, *WSDL*, http://www.w3.org/TR/wsdl, 2001.