Faculty Publications

2003-03-17

Transport Discovery in Wireless Multi-Transport Environments

Shannon B. Barnes
barnes@byu.edu

Charles D. Knutson
knutson@cs.byu.edu

Ryan W. Woodings

Follow this and additional works at: <https://scholarsarchive.byu.edu/facpub>

Part of the [Computer Sciences Commons](#)

Original Publication Citation

Shannon B. Barnes, Ryan W. Woodings, and Charles D. Knutson. "Transport Discovery in Wireless Multi-Transport Environments." Proceedings of the IEEE Wireless Communications and Networking Conference 23 (WCNC 23), New Orleans, Louisiana, March 17-19, 23.

BYU ScholarsArchive Citation

Barnes, Shannon B.; Knutson, Charles D.; and Woodings, Ryan W., "Transport Discovery in Wireless Multi-Transport Environments" (2003). *Faculty Publications*. 507.
<https://scholarsarchive.byu.edu/facpub/507>

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

Transport Discovery in Wireless Multi-Transport Environments

Shannon B. Barnes
Intel Research and Development
2111 N.E. 25th Avenue
Hillsboro, Oregon 97124
Email: shannon.b.barnes@intel.com

Ryan W. Woodings, Charles D. Knutson
Mobile Computing Laboratory
Brigham Young University
Provo, Utah 84602
Email: {woodings, knutson}@cs.byu.edu

Abstract—In order to utilize multiple transports, devices must discover common mechanisms for communication, a procedure we call Multi-Transport Discovery. The Multi-Transport Discovery algorithm presented in this paper is a four-phase procedure (Transport Probing, Transport Querying, Address-to-Device Mapping, and Transport Accessibility) that can discover common transports within a multi-transport environment. Transport Probing uses a transport-dependent device discovery mechanism to discover an initial link. Transport Querying communicates over the probed link to query additional transports. Address-to-Device Mapping correctly correlates each transport to a remote device. Finally, Transport Accessibility periodically ascertains link availability during an application session.

I. INTRODUCTION

Wireless multi-transport environments are becoming increasingly common. A multi-transport environment consists of two or more devices each possessing at least two common transports, suggesting that communicating devices have alternate means of exchanging information.

As an example, personal digital assistants (PDAs) commonly include an IrDA port for inter-PDA data transfers and a USB connector for communication between a PDA and a personal computer. They may also integrate Bluetooth and 802.11b transceivers. Several PDAs support add-on cards that offer additional transports (such as wired Ethernet or analog modem). In addition, some PDAs are converging with mobile phones, adding cellular packet-based networks to their cadre of supported transports.

Despite the proliferation of multi-transport PDAs, integrating multiple transports within a single task or session is not common (and normally not possible). This paper presents an algorithm that enables mobile devices to discover common transports. In ad hoc mobile environments, this discovery process must be done without the assistance of well-known third party agents. Awareness of mutual transports is a first step in enabling network applications to exploit the capabilities of multi-transport mobile devices.

This paper introduces Multi-Transport Discovery within the context of the Quality of Transport (QoT) architecture developed in the Mobile Computing Laboratory of Brigham Young University. While this paper demonstrates the algorithm within the framework of QoT, Multi-Transport Discovery is relevant to all systems that may benefit by incorporating cooperative

multi-transport capabilities. The algorithm we present here incorporates four phases: Transport Probing, Transport Querying, Address-to-Device Mapping, and Transport Accessibility.

The remainder of this paper is structured as follows. Section II discusses the benefits of utilizing multiple transports and the role of Multi-Transport Discovery. Section III presents overviews of the architecture required for Multi-Transport Discovery and each of its phases. Section IV provides a description of how these phases fit within the Multi-Transport Discovery algorithm. Section V describes an implementation of the algorithm on a Linux-based PDA. In section VI we present conclusions.

II. BENEFITS FOR MULTI-TRANSPORT ENVIRONMENTS

As new communication technologies emerge, it is common for devices to adopt new transports while continuing to support legacy technologies. As a result, over time devices increasingly support multiple transports. For example, many PDAs and cell phones that have traditionally supported infrared communication using IrDA have now added Bluetooth technology without eliminating IrDA. As a result, Bluetooth and IrDA transports generally co-exist on these devices.

Device support for multiple transports generally takes one of two forms. 1) Some new technologies emerge as potential replacements for currently inadequate communication mechanisms. In situations where a new technology promises to replace an existing one, legacy support is often used to temporarily accommodate existing applications with which users are comfortable. Over time, the new transport replaces the old one. In such situations, the multi-transport environment is temporary. 2) Some technologies emerge to meet needs previously unfulfilled by existing communication mechanisms. In such situations the technologies do not compete per se, but represent complementary communication mechanisms that fulfill different usage models. For example, cellular telephony fills a very different need than point-to-point infrared beaming, even though they both involve wireless communication. When technologies are complementary, the resultant multi-transport environment is much more enduring as a user platform.

New technologies continue to emerge, some competitive and some complementary. As examples, 802.11 continues to evolve and compete with wireless networking standards

(including its own prior versions) [1], while Ultra-Wideband shows promise for interference-resistant short-distance high bandwidth data communication [2], [3]. As stated in [4], “no single standard protocol or protocol family will become the universal protocol supported by all networked systems.”

While the architecture of most PDAs allows for coexistence of transports, heterogeneous protocols normally do not intelligently work together and as a result, applications, session layer protocols, and services are typically tied to a single specific transport. As new transports are introduced, existing applications need to be altered and recompiled to utilize these new technologies. Some research has attempted to utilize the potential multi-transport capabilities of mobile devices. For example, it has been proposed that IrDA could be used to speed the device discovery time of Bluetooth [5]. This study demonstrates the use of IrDA to exchange Bluetooth connection parameters, allowing IrDA’s efficient device discovery mechanism to offset the relatively slow inquiry process of Bluetooth. Seamless connectivity, whether at home, at the office, or traveling in the car is the goal presented in studies such as [6], [7]. These studies introduce the concept of Wireless Overlay Networks, in which mobile devices may switch from fast short-range wireless networks to slower high-range networks in order to preserve the best possible connection.

Quality of Transport (QoT) introduces intelligence between session layers (such as HTTP, FTP, and OBEX) and multiple transports (such as IrDA, Bluetooth, and TCP/IP), facilitating dynamic transparent switching of the underlying protocol stack to achieve the best connection possible [8], [9]. A device equipped with QoT might initiate a fast data transfer with 4 Mbps IrDA but switch to a slower RF-based transport (such as Bluetooth) if the line-of-sight requirement became disrupted. QoT could continue to preserve the connection if the device moved out of Bluetooth’s 10-meter range by switching to a longer range RF transport (such as 802.11b) preserving the connection at the cost of higher power consumption. QoT transparently enables applications to take advantage of multi-transport capabilities rather than limiting an application to one transport.

III. MULTI-TRANSPORT DISCOVERY

Most network-based applications are designed to operate over a specific transport (for example, web browsers are typically designed to use HTTP over TCP/IP). However, in cooperative multi-transport environments, there can be no initial assumption that a particular transport exists, but rather that communicating devices are capable of creating a link with at least one transport. Multi-Transport Discovery is the mechanism that allows devices to discover common transports.

By understanding the nature of available communication channels, QoT can make dynamic decisions that optimize the quality of the connection. Connection quality can be assessed in various ways, including: 1) preserving battery life; 2) maximizing range; and 3) maximizing throughput. Hence, the ultimate goal of QoT is to provide the user with the best

Fig. 1. Flowgraph of Multi-Transport Discovery algorithm

connection possible, where “best” is defined in terms that are most meaningful to the user.

In order for this process to be convenient and transparent, the two communicating devices must be able to dynamically discover common transports with as little user intervention as possible.

Figure 1 describes the mechanism for Multi-Transport Discovery. This algorithm is divided into four phases: Transport Probing, Transport Querying, Transport-to-Device Mapping, and Transport Accessibility. The first two phases (Transport Probing and Transport Querying) identifies all existing transports in a multi-transport environment. The Transport-to-Device Mapping phase works in unison with Transport Querying to properly match transports with each remote device. The last phase, Transport Accessibility, is an ongoing process that continually checks discovered transports to ascertain their availability. These are described in the sections that follow.

A. Remote Device Table

The main function of the Multi-Transport Discovery algorithm is to populate and update a Remote Device Table (RDT). This table identifies all connectable remote transports found via Multi-Transport Discovery. The RDT provides the raw information required by cooperative multi-transport systems to utilize available transports. Information found in the RDT includes transport connection parameters (such as IP addresses for TCP/IP), availability status, transport-to-device mappings, and various link descriptors (including bandwidth,

Fig. 2. Structural representation of a Remote Device Table

range, cost, and power constraints). These descriptor values can be statically configured, queried between devices, or dynamically discovered. The RDT provides applications the information necessary to connect to available transports and perform decision making functions.

Figure 2 shows a simplified representation of the internal structure of an RDT. Each remote device supports one or more transports, each of which is associated with attributes describing various link descriptor values.

B. Transport Probing

The main objective of Transport Probing is to find at least one transport through which communication can be established with each remote device. After an initial connection is made, full transport capabilities can then be exchanged over this link.

Transports have varying device discovery capabilities. Short-range wireless transports such as IrDA and Bluetooth were designed to dynamically discover devices within range, determining connection parameters in an ad hoc fashion without *a priori* information. LAN transports such as TCP/IP typically require a user to know either an IP address or a domain name that can be converted to an IP address in order to establish a remote connection at a network level. An application using TCP/IP can overcome this limitation by implementing a broadcast/multicast packet scheme to find remote devices, but this function is not inherent in the transport level. These methods can be limited when nodes are separated by routers that do not forward multicast packets or are not assisted by third party servers.

While the goal of Transport Probing is not to find all available transports, in practice a local device normally must probe all of its transports in order to be certain it has found all connectable devices. Transports supported by discovered devices, but which were not discovered during Transport Probing will be subsequently revealed during Transport Querying of each remote device.

Preliminary implementations of QoT incorporate IrDA, Bluetooth, and TCP/IP. A brief synopsis of Transport Probing using each of these three transports is described below:

1) *Transport Probing with IrDA*: To initiate Transport Probing with IrDA, the primary device broadcasts a fixed number of sequenced discovery XID (Exchange Identification) frames with increasing slot numbers (starting with 0). All IrDA remote devices that receive the initial XID frame generate a random slot number. The secondary device responds with its own XID when it receives a discovery XID from the local device that corresponds to its chosen slot number. The responding XID packet from the remote device contains the device address needed to initiate subsequent link negotiations and link establishment [10].

2) *Transport Probing with Bluetooth*: To initiate Transport Probing with Bluetooth, the local device enters an Inquiry state, during which it transmits ID packets containing an Inquiry Access Code (IAC). Discoverable devices respond with a Frequency Hop Synchronization (FHS) packet, which contains information such as the remote device's BT_ADDR, that is needed to perform a connection [11].

3) *Transport Probing with TCP/IP*: TCP/IP is an example of a transport that does not have a mechanism whose job is strictly to discover all available TCP/IP connections on a network or subnetwork. The usage model for TCP/IP is to connect to a computer whose address or name is already known. However there are protocols within TCP/IP and Domain Name System (DNS) that can be used to find closely connected hosts. These techniques originate from algorithms used in creating dynamic network topologies.

For QoT, we incorporated a multicast system in which each QoT-enabled device registers with the same multicast group. We designed a simple protocol in which Transport Probing devices send out a multicast request for QoT devices, signaling each remote device to return its IP address.

QoT also incorporates an ICMP (Internet Control Message Protocol) Echo Request packet mechanism in order to discover non-QoT devices. This action is equivalent to sending a PING to all the computers on a broadcast domain. All TCP/IP hosts that accept a broadcast message respond with an Echo Reply Packet identifying their IP address. The local host then receives the Echo Reply Packets from all responding nodes. This is an effective method for situations in which nodes are connected in a point-to-point fashion, rather than separated by routers. However, some operating systems may reject broadcast ICMP Echo Request packets because they can be used for denial-of-service attacks.

C. Transport Querying

As previously discussed, Transport Probing only attempts to find an initial link to remote devices. Transport Querying completes the discovery process by identifying all of the transport capabilities of each remote device. Transport Querying works because each device is aware of all the transports it supports. Once a local device has established an initial link

with a remote device, it can query to discover the additional supported transports.

Transport Querying complements Transport Probing by discovering supported transports that were not visible during the probing phase. Transports that require line-of-sight (such as IrDA) or those that have limited range (such as Ultra-Wideband and Bluetooth) might remain undiscovered after Transport Probing. However, if at least one link can be established, querying for transports guarantees that a local device can be made aware of all transports supported by accessible remote devices. To facilitate Transport Querying within QoT, we designed the Transport Exchange Protocol.

The Transport Exchange Protocol offers two querying function calls, `Transport_Query` and `Transport_Info_Query`. The first function, `Transport_Query`, issues a QoT Transport Query Packet to the recipient over a link discovered during Transport Probing. The recipient device returns a Transport Query Response Packet, which contains a list of all available transport types supported by the remote device. After the local device receives the list, it can obtain additional information about specific transports by calling the `Transport_Info_Query` function for a particular transport. Remote devices may then respond by sending a QoT Transport Info Response Packet, containing the parameters necessary to connect via the particular transport, and a structure that contains name/value pairs providing various link descriptors (such as cost, range, speed, etc.).

It should be noted that certain transports support native service discovery protocols that can be used to store and exchange transport information in an ad hoc point-to-point fashion. For example, the Information Access Service (IAS) of IrDA and the Service Discovery Protocol (SDP) of Bluetooth could each be used to perform functions similar to the Transport Exchange Protocol. However, not all transports provide standard ad hoc point-to-point service discovery mechanisms, so the creation of an independent Transport Exchange Protocol was deemed necessary within the QoT architecture.

D. Address-to-Device Mapping

While all connectable transports might be discovered, they provide little value unless transports can be mapped to corresponding devices. For example, suppose that Transport Probing, reveals an IrDA link, a Bluetooth link, and a TCP/IP link over 802.11b. This may represent one device with three transports or a combination of multiple devices each possessing one or two transports.

After the Transport Probing phase is complete, each probed transport is represented in the Remote Device Table (RDT) as a unique remote device, and is subjected to a Transport Query. After a query is performed on a probed transport, the transport query results are compared to the other transports discovered during probing for duplicate addresses. If there is a network address match, then the probed transport entry is deleted from the RDT. This process is iteratively applied

Session Transport	Time interval between accessibility checks		
	IrDA	Bluetooth	802.11
	IrDA	—	1 min.
	Bluetooth	3 sec.	—
	802.11	1 min.	1 min.

TABLE I
TIME INTERVALS FOR CHECKING TRANSPORT ACCESSIBILITY

to all remaining transports, yielding an accurate mapping of transports to devices in the RDT.

E. Transport Accessibility

Transport Accessibility refers to the periodic verification of the availability of a discovered transport. The main task for this phase is to ensure that a known wireless transport is in range.

After the first three phases of Multi-Transport Discovery, the RDT is populated with information concerning remote devices and their respective transports. An application session (such as ftp download) may be initiated after the completion of the first three phases of Multi-Transport Discovery. The Transport Accessibility phase operates in the background, ascertaining link availability for various transports during the application session. This phase provides information that enables a multi-transport management system (such as QoT) to determine the feasibility or desirability of dynamically switching transports during a session.

Each transport in the RDT is periodically tested to determine link availability. The interval between link checks varies between transports, and is dependent on (1) the nature of the transport being checked and (2) the nature of the link being used in the active session. These two factors can be illustrated in the following example.

Suppose two QoT-enabled PDAs are engaged in a data transfer session and each PDA is equipped with IrDA, Bluetooth, and 802.11b. The assumed goal in this scenario is to give the user a consistent data connection while minimizing battery utilization. Hence, QoT would prioritize first to using IrDA (available within 1 meter), then Bluetooth (available within 10 meters) and finally 802.11b. As a result, Transport Accessibility for higher priority transports would occur more frequently than for lower priority transports. Table I provides an example of guidelines that might be used to prioritize interval times in such a scenario. Note that the values in this table are merely examples. An empirical study of appropriate interval values is currently being conducted but preliminary results are beyond the scope of this paper.

IV. MULTI-TRANSPORT DISCOVERY ALGORITHM

Figure 3 presents the algorithm for Multi-Transport Discovery. A brief description of the algorithm follows.

Let a be a transport address
 Let t be a transport type
 Let T be a list of transport types
 Let d be a remote device containing $\{a_0 \dots a_{n-1}\}$ where n is the number of transport addresses on a device
 Let d_i be a local device containing $\{t_0 \dots t_{m-1}\}$ where m is the number of supported transport types
 Let D be a Remote Device Table containing $\{d_0 \dots d_{r-1}\}$ where r is the number of remote devices
 Let s represent a link's availability status

Algorithm Multi-Transport Discovery

Phase 1: Transport Probing

- (1) **for each** t_i **in** d_i
- (2) $a \leftarrow \text{Probe}(t_i)$
- (3) **if** a **not in** D
- (4) $D \leftarrow \text{InsertNewDevice}(a)$

Phase 2 & 3: Transport Querying and Transport-To-Device Mapping

- (5) **for each** d_i **in** D
- (6) $T \leftarrow \text{Transport_Query}(d_i, a_0)$
- (7) **for each** t_j **in** T **where** t_j **is in** d_i
- (8) $a \leftarrow \text{Transport_Info_Query}(d_i, a_0, t_j)$
- (9) $D \leftarrow \text{UpdateTable}(d_i, a)$
- (10) **for each** d_k **in** D **where** $k > i$
- (11) **for each** t_m **in** D_i
- (12) **if** t_m **equals** $d_k.t_0$
- (13) $D \leftarrow \text{DeleteDevice}(d_k)$

Phase 4: Transport Accessibility

- (14) **repeat**
- (15) **for each** d_i **in** D
- (16) **for each** a_j **in** d_i **such that** a_j **is not** the current link
- (17) $s \leftarrow \text{TestAddress}(a_j)$
- (18) $D \leftarrow \text{UpdateTable}(a_j, s)$

Fig. 3. Multi-Transport Discovery Algorithm

The multi-transport discovery algorithm begins with the Transport Probing phase. Each of the local device's transports is subjected to the `Probe` function (2), which performs the transport dependent method for discovery of remote devices. Each remote transport discovered from probing is inserted as a new device in the Remote Device Table (4).

The Transport Querying and Transport-to-Device Mapping phases work in unison to help achieve better efficiency. Each transport discovered through probing is represented as a unique device within the RDT. At this stage each remote device in the RTD is associated with only one transport. Each of these transports is subjected to a `Transport_Query` (6), which returns a list of transport types found on the remote device. Transport types supported by both local and remote devices are then subjected to a `Transport_Info_Query` (8) which returns information needed to perform link establishment as well as attributes that enable the system to make intelligent

decisions for dynamic transport switching. The RDT entry for the remote device is updated with new transport information (9). Each of the returned transports is then checked against the remaining probed transports for duplicates (12). By deleting any duplicate probed transports, we eliminate unnecessary querying calls and properly associate groups of transports to a particular device.

After all transports are discovered, the Multi-Transport Discovery algorithm then engages in the Transport Accessibility phase. In this phase each of the common transports (16) are tested to see if the link is available. The `TestAddress` function (17) takes into account the type of transport being tested in determining the proper delay to be incorporated. Once the status is returned, the RDT is updated.

Fig. 4. Screenshot of a Multi-Transport Algorithm implemented on a Linux based PDA

V. IMPLEMENTATION

Figure 4 shows a screenshot of an implementation of the Multi-Transport Discovery Algorithm running on a Linux-based PDA. This device includes a USB port, a Bluetooth Compact Flash (CF) card, and an integrated IrDA transceiver. When the PDA is in its cradle, the USB port simulates Ethernet, allowing a connection to the LAN through a PC serving as a gateway. The screenshot shows the device after Transport Probing and Querying and indicates that the PDA has discovered three devices (Traer, Keokuk, and Swisher) and their respective transports (Bluetooth, IrDA and TCP). Traer and Swisher are Linux based PC's that both have TCP/IP network cards as well as serial IrDA dongles. Traer is also equipped with a Bluetooth radio connected by USB. Keokuk is a nearby PDA with an integrated Bluetooth transceiver.

During the discovery process, the PDA found Traer and Keokuk by probing Bluetooth, and Swisher from probing IrDA. Traer and Swisher both implement the Multi-Transport Discovery Algorithm, so during the Transport Querying phase the PDA was able to determine the transports of both PC's. When the screenshot was taken, the PDA was not in its cradle, and as a result could not access any TCP/IP connections. The Transport Accessibility phase was not able to create links through TCP/IP and therefore their statuses are shown as unavailable.

A user (or a multi-transport capable application) can now see the transports supported by each device and can therefore make appropriate link establishment decisions. In the screenshot, the user chose to make a connection to Traer using Bluetooth knowing that she wanted the ability to walk around the room and not be limited by either a cradle (TCP/IP) or a line-of-sight requirement (IrDA). In this implementation, she is free to dynamically change her transport selection as needed to optimize the data link.

VI. CONCLUSIONS

Research efforts involving multi-transport environments have commonly relied on *a priori* knowledge of the presence of particular transports in order to provide connectivity. The Multi-Transport Discovery algorithm described in this paper provides a mechanism for mobile devices to discover all transports within an ad hoc multi-transport environment. This algorithm was designed in conjunction with the Quality of Transport (QoT) project in the Mobile Computing Laboratory at Brigham Young University. Multi-Transport Discovery is an important element in a variety of intelligent ad hoc multi-transport systems when common transports are not statically known.

REFERENCES

- [1] IEEE 802.11, The working group for wireless LANs. [Online]. Available: <http://www.ieee802.org/11>
- [2] K. Siwiak, "Ultra-wide band radio: Introducing a new technology," in *Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC)*, vol. 2, Rhodes, Greece, May 2001, pp. 1088–1093.
- [3] J. Foerster, E. Green, S. Somayazulu, and D. Leeper, "Ultra-wideband technology for short- or medium-range wireless communications," *Intel Technology Journal*, no. Q2, p. 11, May 2001.
- [4] R. J. Clark, M. H. Ammar, and K. L. Calvert, "Protocol discovery in multiprotocol networks," *Mobile Networks and Applications*, vol. 2, no. 3, pp. 271–284, 1997.
- [5] R. W. Woodings, D. Joos, T. Clifton, and C. D. Knutson, "Rapid heterogeneous ad hoc connection establishment: Accelerating Bluetooth inquiry using IrDA," in *Proceedings of the Third Annual IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 1, Orlando, Florida, Mar. 2002, pp. 342–349.
- [6] E. A. Brewer, R. H. Katz, et al., "A network architecture for heterogeneous mobile computing," *IEEE Personal Communications*, vol. 5, pp. 8–24, Oct. 1998.
- [7] M. Baker, X. Zhao, and J. Stone, "Supporting mobility in MosquitoNet," in *Proceedings of the 1996 USENIX Technical Conference*, San Diego, California, Jan. 1996.
- [8] R. W. Woodings, S. B. Barnes, and C. D. Knutson, "Transport independence and multi-transport issues in protocol design," *Mobile Computing Laboratory, Brigham Young University, Tech. Rep. BYU-MCL-02-02*, Sept. 2002.
- [9] "Quality of Transport (QoT) protocol specification," Brigham Young University, Provo, Utah, Aug. 2002.
- [10] "Infrared Data Association serial infrared link access protocol (IrLAP)," Infrared Data Association, Walnut Creek, California, June 1996.
- [11] "Specification of the Bluetooth system," Bluetooth SIG, Feb. 2001.