IDENTITY-BASED DISTRIBUTED COMPUTING FOR DEVICE RESOURCES

FIELD OF THE INVENTION

The invention relates generally to computer systems, and more particularly to computer resources.

BACKGROUND OF THE INVENTION

In early personal computing systems, application program had to be programmed with knowledge of various computing devices in order for the application to use such devices. For example, in order to print to a particular printer, a word processing program had to know how to print to the printer, and generally supplied a driver for each supported printer. Alternatively, a printer would supply drivers with it for the most popular programs. The user had to configure each program to inform that program as to which printer was connected to the personal computer.

Contemporary operating systems such as Microsoft Windows® provided developers with a way to interface applications with such devices regardless of the specific device driver needs and requirements. For example, with the Microsoft Windows® operating system a user or the device tells the operating system which printer (or printers) is connected, after which any application is able to communicate with the printer driver via the operating system. As a result, to accomplish

159159

G00002

printing, application programs generally only need to deal with a set of standardized printing functions provided by the operating system. The enormous benefits of such an abstraction cannot be understated, e.g., this abstraction enables any one of thousands of software programs to interact with any one of thousands of printers and/or other devices, without the program having to know (or otherwise supply) the details about the printer in advance.

Microsoft Corporation's .NET technology provides a somewhat similar type of abstraction regarding a user's or company's data. In general, with .NET technology, any .NET-capable program on any device can interact with a user's data via centralized schema-based services. Because of the schematization, the programs do not need specific knowledge about the details of that data, e.g., which used to be stored in proprietary file formats, but rather only needs to be able to deal with (read and write) data formatted according to one or more of the schemas.

With .NET technology, a user's (or other entity's) identity controls access to the centrally-maintained data, e.g., a user may use a .NET Inbox service keep his or her electronic mail data centrally stored (e.g., on the Internet or an intranet), and then access that data via the .NET inbox service, at any time, from any device capable of connecting to

the Internet or an intranet and having a program capable of interacting with the .NET Inbox service. The user's identity is used to securely match the user with the appropriate data. Users can also controllably allow access to data by other users, e.g., a user can selectively make certain business contact information available to business associates, make personal contact information available to friends, and so on, based on the others' identities.

While .NET thus provides tremendous data-based benefits to users, companies and other entities, the centralized service and centralized data are not particularly concerned with how a user uses various devices and those devices' resources. For example, devices such as a personal computer and pocket personal computer have, for quite some time, been able to directly synchronize with one another. To an extent, .NET technology is another way to keep such data synchronized, although indirectly, without needing specialized and proprietary synchronization programs.

However, even with .NET, a user cannot easily use one device to access the resources of another device, such as to use the computing power, storage, bandwidth, input and output mechanisms, programs and/or other resources of one device on behalf of another device, e.g., to perform a complex or better-suited task on the more powerful device. What is

needed is a common, consistent way to remotely discover and expose computing resources of various computing devices to one another for cross-resource usage.

SUMMARY OF THE INVENTION

Briefly, the present invention provides a system and method for discovering and describing computing resources available at various computing devices associated with an identity, and for exposing those resources as services that are addressable by software programs, such that programs on the devices can leverage another device's resources. The device services are identity-based, in that a device owner's devices discover each other and/or other devices accessible by a user based on the owner's or other user's identity. Further, based on identity, a device owner (or administrator) can control what other users, and/or other devices can leverage the resources of each owned device, as well as to what extent.

Computing resources may include, for example, the storage capacity of the devices, bandwidth, processing power (e.g., the CPU and RAM capabilities), input mechanisms and methods, rendering and other output mechanisms and methods (e.g., screens, printers and audio), and programs. A device is thus abstracted into its resources, which are then programmatically

addressable from other devices in a composible fashion (e.g., the remote resources may be defined in such a way that they are, in turn, comprised of smaller resources contained within the larger definition of the resource) . In other words, for devices to participate in an intelligent distributed computing model, various aspects of each participating device are abstracted and made programmatically addressable from other devices. A unified identity-based mechanism discovers the list of remote resources on the device. A set of one or more schemas consistently describes the remote resources on the device, and a unified security model allows the owner of the device to control who has access to which resources, and to what extent.

With the present invention, desirable computing solutions for a great number of example scenarios are enabled, including a set of real-time scenarios in which one user device interacts with one or more remote computing devices in real-time, and a set of scheduled or predicted scenarios that operate transparently and automatically on a user's behalf.

Other advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1A is a block diagram generally representing a computer system into which the present invention may be incorporated;

FIG. 1B is a block diagram generally representing another computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram representing various components in a distributed computing environment in accordance with an aspect of the present invention;

FIG. 3 is a block diagram generally representing an architecture for discovering and exchanging resource information in accordance with an aspect of the present invention;

FIGS. 4 and 5 are block diagrams representing examples of how the present invention can improve a user computing experience by distributing computing operations to resources, in accordance with an aspect of the present invention;

FIG. 6 is a flow diagram representing exemplary logic and related operations performed to facilitate the sharing of resources in a distributed computing environment in accordance with an aspect of the present invention; and

FIG. 7 is a block diagram generally representing an example implementation including resources distributed in accordance with an aspect of the present invention.

DETAILED DESCRIPTION

EXEMPLARY OPERATING ENVIRONMENTS

FIGURE 1A illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to: personal computers, server computers, hand-held or laptop devices, tablet devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe

computers, distributed computing environments that include any of the above systems or devices, and the like.

The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, and so forth, that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in local and/or remote computer storage media including memory storage devices.

With reference to FIG. 1A, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of the computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and

not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus. Note that any and all of the elements shown in computer 110 may be connected to each other by standard high-speed computer networks, including those distributed across a wide geographic area. For example, the system memory 130 may be located in a physically different location than processing unit 120, but yet defined in a composible fashion as logical computer 110.

The computer 110 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer 110 and includes both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM,

digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by the computer 110. Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or

program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1A illustrates operating system 134, application programs 135, other program modules 136 and program data 137.

The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1A illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

The drives and their associated computer storage media, discussed above and illustrated in FIG. 1A, provide storage of computer-readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1A, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146 and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers herein to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a tablet, or electronic digitizer, 164, a microphone 163, a keyboard 162 and pointing device 161, commonly referred to as mouse, trackball or touch pad. Other input devices not shown in FIG. 1A may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected

to the system bus 121 via an interface, such as a video interface 190. The monitor 191 may also be integrated with a touch-screen panel or the like. Note that the monitor and/or touch screen panel can be physically coupled to a housing in which the computing device 110 is incorporated, such as in a tablet-type personal computer. In addition, computers such as the computing device 110 may also include other peripheral output devices such as speakers 195 and printer 196, which may be connected through an output peripheral interface 194 or the like.

The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1A. The logical connections depicted in FIG. 1A include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet. For example, in the present invention, the computer system 110 may comprise source machine from which data is being migrated,

and the remote computer 180 may comprise the destination machine. Note however that source and destination machines need not be connected by a network or any other means, but instead, data may be migrated via any media capable of being written by the source platform and read by the destination platform or platforms.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160 or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1A illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

FIGURE 1B illustrates an example of another suitable operating environment (computer device 188) in which at least

part of the present invention may be implemented. As with FIG. 1A, the operating environment of FIG. 1B is only one other example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. The other computer device 188 in FIG. 1B typically includes at least some form of computer readable media. Computer-readable media can be any available media (as generally defined above) that can be accessed by the computing device 188.

FIG. 1B shows functional components of a computing device 188 such as a handheld (pocket-sized or tablet) personal digital assistant, including a processor 189, a memory 190, a display 192, and a keyboard 193 (which may be a physical or virtual keyboard). The memory 190 generally includes both volatile memory (e.g., RAM) and non-volatile memory (e.g., ROM, PCMCIA cards, and so forth). An operating system 194 is resident in the memory 190 and executes on the processor 189, such as the Windows® CE operating system from Microsoft® Corporation, or another operating system.

One or more application programs 195 are loaded into memory 190 and run on the operating system 194. Examples of applications include email programs, scheduling programs, PIM (personal information management) programs, word processing programs, spreadsheet programs, Internet browser programs, and

so forth. The handheld personal computer 188 may also include a notification manager 196 loaded in the memory 190, which executes on the processor 189. The notification manager 196 handles notification requests, e.g., from the application programs 195.

The handheld personal computer 188 has a power supply 197, which is implemented as one or more batteries. The power supply 197 may further include an external power source that overrides or recharges the built-in batteries, such as an AC adapter or a powered docking cradle.

The exemplary handheld personal computer 188 represented in FIG. 1B is shown with three types of external notification mechanisms: one or more light emitting diodes (LEDs) 198 and an audio generator 199. These devices may be directly coupled to the power supply 197 so that when activated, they remain on for a duration dictated by a notification mechanism even though the handheld personal computer processor 189 and other components might shut down to conserve battery power. The LED 198 preferably remains on indefinitely until the user takes action. Note that contemporary versions of the audio generator 199 use too much power for today's handheld personal computer batteries, and so it is configured to turn off when the rest of the system does or at some finite duration after activation.

IDENTITY-BASED DISTRIBUTED COMPUTING FOR DEVICE RESOURCES

The present invention is generally directed to a system and method that facilitate the distribution of resources of devices to other devices. The present invention is ordinarily not involved with what the devices do with each other's resources, what is communicated at that time, or how the communication between devices takes place. Rather, the present invention can be generally considered more as a structured way for devices to find out about each other's resources, capabilities and other details, and then allow the devices to share resources on their own. Although not necessary to the present invention, as part of facilitating the resource distribution, security is provided for practical purposes, at least to ensure that valid credentials are required to use another device's resources. Thus, the present invention is preferably identity-based, in that distributed resources have relationships with one another according to identity data (e.g., credentials) provided to the devices.

In general, the present invention is directed to distributing computing among a user's devices, wherein a user (or owner / administrator) is anyone authorized to control and/or use the devices. However, because in general devices are increasingly becoming disaggregated, the methods and

systems described herein may be more properly described as distributed computing of resources. For example, a single distributed device may be made up of many resources, but may share as little as one its resources (e.g., only its non-volatile storage) to participate as a resource provide. Notwithstanding, as used herein, the terms "device" and "resource" are generally interchangeable except where otherwise apparent, e.g., using another device or using a resource of the other device to accomplish some computing task are generally considered the same thing. Also, as used herein, bandwidth is a device resource, even though bandwidth may not ordinarily be considered as being associated with any particular device.

As generally represented in FIG. 2, a distributed computing environment 200 constructed in accordance with various aspects of the present invention includes a first device 202 (Device A, such as the computer system 110 of FIG. 1A) and a second device 212 (Device B, such as the computer system 188 of FIG. 1B), along with possibly other devices (not shown). Each device that participates in the distributed computing environment 200 includes a distributed computing manager, 204 and 214. In general, and as described below, the distributed computing managers 204 and 214 may comprise some amount of code that communicate with one another over some

medium, e.g., wireless FM, line-of-sight such as infrared, satellite, a direct connection, the Internet, or an intranet. Indeed, more than one such medium may be available, and the distributed computing managers 204 and 214 may choose which one or ones are best for a given circumstance.

An initial purpose of the distributed computing managers 204 and 214 are to locate each other. Such location may be automatic, such as by having each device occasionally broadcast its availability to others and/or by polling for the availability of other devices, or alternatively may operate in response to some specific request from a program (or user) to connect to another device. As represented in FIG. 2, the distributed computing managers 204 and 214 each include a security mechanism 206 and 216, respectively, that exchange credentials 207, 217 or the like (e.g., obtained from an authentication service 220) to ensure that the user who is currently operating the device which is contacting it (and/or the device itself), is valid in the distributed computing environment, and similarly, if necessary, that the device which is being contacted is valid. Note that a program being used may also factor into the security / access control, e.g., a first program running on one device may have authority to use another device's resource or resources, but not another program, independent of or in addition to the user identity

and/or the device identity. Further note that the security mechanism may operate per identity and per resource, e.g., one entity may be authorized to use a device's display, but not its storage, while another entity may be able to use any of that same device's available resources, including its storage. This is similar to the .NET security model in which services provide methods that control access to a users' data based on the requesting user's identity, defined role and scope for that role. In this way, data can be accessed by its owner, and shared to an extent determined by the owner. In the present invention, instead of data it is a device resource being accessed, whereby a user's identification, defined role and scope for that role determine the extent to which a resource may be accessed.

Indeed, the security mechanisms of the present invention are based on those essentially those employed for security / access control in .NET services, as generally described in United States patent applications, serial nos. 10/003,754 entitled "IDENTITY-CENTRIC DATA ACCESS" and serial number 10/003,767, entitled "AUTHORIZING A REQUESTING ENTITY TO OPERATE UPON DATA STRUCTURES" filed on October 22, 2001, assigned to the assignee of the present application, and herein incorporated by reference. To this end, at least a thin subset of the .NET service runs on each device which

expects to advertise and share computing resources in the environment 200. In addition, a localized internal consistent security model may be executed on these devices such that local security is provided even when two local peer-to-peer devices may be disconnected from the centralized (e.g., Passport) authentication service 220 (FIG. 2).

The present invention is not limited to any particular transmission medium, but provides that the distributed resource services may be directly addressable to the target device, or may be queried via the Internet or an intranet, such as via the centralized set of .NET services on the web. When connected via the Internet or intranet, a program may query the last known computing resources available on any device associated with the user's identity, even when the device is offline or otherwise inaccessible. Bandwidth is also a resource, and may impact how a device's resources are shared, e.g., it may be more efficient to leverage a personal computer's resources when the personal computer has a high-speed connection to the Internet, but not if the personal computer connects only at low speed, such as if the other device has a high-speed connection. In fact, the personal computer may want to leverage the high-speed connection of the other device, in which event that other device may then leverage a resource of the personal computer, e.g., the

devices exchange resources to provide a more-optimal user experience.

In accordance with one aspect of the present invention, the distributed computing managers 204 and 214 also each include a respective discovery mechanism, 208 and 218, that allows the devices to discover another participating device's shared resources / capabilities, that is, when that other device is configured to share its resources. The respective resource data 209 and 219 is formatted (e.g., in XML) and exchanged in accordance with a defined schema, as described below. As represented in FIG. 2 by the exchange of the schematized data 209 and 219, both devices 202 and 204 are configured to share their resources (R_{A1}-R_{Am} and R_{B1}-R_{Bn}) with other devices in the distributed computing environment. However as can be appreciated, in a given environment, a particular device (e.g., a mobile telephone) may be only a consumer of resources rather a provider of any resource, or conversely may be only a provider, and not a consumer.

A device that is sharing its resources need not directly communicate the resource data with the other device, but instead, for example, can advertise its resources somewhere for other devices to discover. For example, a .NET service (.NET Devices) is provided that centrally stores device resource capabilities such that data may be appropriately

formatted before sending it to a device, e.g., so as to not send high-resolution graphics to a device incapable of displaying them. A similar .NET Resources service, or an extended .NET Devices service, may be provided to advertise this information to other devices.

The following tables set forth some example schemas that may be used (e.g., with XML-formatted data) to exchange resource (storage and processing power) information regarding a user's devices, which in this example include work and home personal computers, and a cellular telephone:

```
<myStorage>
      <device name="Work PC" uuid="7a733d84-b394-4ba1-56b5c0d">
            <driveSize/>
            <freeSpace/>
            <transferSpeed/>
            <seekLatency/>
      </device>
      <device name="Home PC" uuid="6d733d84-b3f4-4ba1-56b5c0f">
            <driveSize/>
            <freeSpace/>
            <transferSpeed/>
            <seekLatency/>
      </device>
      <device name="Cell Phone" uuid="5f733d84-b3f4-4ba1-56b5c0f">
                  <driveSize/>
                  <freeSpace/>
                  <transferSpeed/>
                  <seekLatency/>
            </device>
</myStorage>
```

```
<myProcessingPower>
      <device name="Work PC" uuid="7a733d84-b394-4ba1-56b5c0d">
            <cpuSpeed/>
            <cpuUsage/>
            <memorySize/>
            <memoryUsage/>
      </device>
      <device name="Home PC" uuid="6d733d84-b3f4-4ba1-56b5c0f">
            <cpuSpeed/>
            <cpuUsage/>
            <memorySize/>
            <memoryUsage/>
      </device>
      <device name="Cell Phone" uuid="5f733d84-b3f4-4ba1-56b5c0f">
            <cpuSpeed/>
            <cpuUsage/>
            <memorySize/>
            <memoryUsage/>
      </device>
</myProcessingPower>
```

As is understood, the schema is preferably resource based, i.e., multiple devices may be listed for each resource type, to match the general concept of an identity having distributed resources available for use, independent of the actual device on which the resource exists. Notwithstanding, the present invention is not limited to any particular schema and/or formats, e.g., the schemas may be device-based, such that each device may instead provide its list of available resources to other devices such that the same information is known.

As generally represented in FIG. 3, an architecture 300 is provided in order for devices to participate in an intelligent distributed computing model, in which various

aspects of the participating devices are abstracted and made programmatically addressable from a different device. A universal plug-and-play (UPnP 2.0) infrastructure or the like may be used as the basic, low-level mechanism 302 for discovery of these resources. Alternatively, the list of resources for various computing devices may be cached in some well-known location, such as an Active Directory store, to enable querying of remote resources even if the device is only accessible through some intermediary such as a firewall or wireless hub. Atop the discovery mechanism are the schemas 304, e.g., one schema associated with each specific type of the computing resource, with each of those schemas exposed, for example, as in the .NET architecture.

Note that while UPnP provides a suitable discovery mechanism 304, it is only part of the process of actually using the remote computing resources. Thus, UPnP and .NET technologies are used to facilitate resource sharing, but are not used to actually employ the resources available on a device. Rather, once the resources are known, appropriately focused protocols for that specific resource domain are used. By way of example, UPnP, with the addition of the .NET identityHeader, endpointHeader, and requestHeader SOAP semantics, is well-suited to obtain the video screen characteristics of the remote device, however a protocol such

as RDP will be used to transfer video data after the initial discovery and handshaking occurs between the devices. Note that this is similar to the way .NET approaches data services today, e.g., .NET provides a mechanism to connect to an electronic endpoint representing a proprietary (e.g., Messenger) protocol, but once that endpoint is connected, .NET allows the client to use an existing real-time protocol. Other protocols appropriate for other types of resources are used, e.g., CIFS+ for storage, which scales well to both corporate and home storage scenarios and enables the user's file system to continuously expand to other storage devices, without forcing the user to deal with drive letters, namespaces, and so forth. In summary, the present invention provides an identity-based approach to discovering these additional resources that are available for use, but does not thereafter directly deal with their use.

FIG. 6 is an example flowchart that represents some general aspects of the present invention, beginning with step 600 which represents receiving a request to use a shared resource. The request may be general, e.g., requesting the list of resources that a user has available on devices configured to participate, or may be a request for a specific resource, e.g., in response to an advertisement. For purposes

of this example, the request is treated as a request for one specific resource.

Step 602 represents the evaluation of the credentials, such as provided by the Microsoft .NET Passport authentication service. The request may include the credentials, or the credentials may be separately exchanged, e.g., before, as part of, or after step 600. If not valid at step 602, step 602 branches to step 604 which represents rejecting the request. Note that depending on a given environment, the request without proper credentials may be ignored rather than specifically rejected.

If valid, step 606 is executed, and represents determining the amount of the requested resource that is available. For example, a resource such as storage may be limited because it has only limited free space, and further, a device may be configured to make only part of its storage space available for sharing, reserving the rest for itself. A resource such as processor cycles may be made available only to the extent the processor is not being otherwise used, and even then may be further limited so that another device can only take up to a maximum amount of the unused cycles. Any resource may have a time-out value associated with it, which may be further contingent upon its use, e.g., storage can be shared for no more than one hour per request, and the request

will be terminated if no file system requests are detected at least every two minutes. In this manner, a device may be configured to share its resources to an extent controlled by its owner / administrator.

Note that the resource access may be dynamic, e.g., unused processor cycles may be shared, however CPU usage is a number that fluctuates, and the requesting guest device may be prioritized along with other devices and/or processes on the device that is sharing the CPU. For example, guest users may only consume a controllable percentage of resources when the owner of the device is not otherwise using them, such as bandwidth or processing power. Both CPU and bandwidth are examples of resources which are scheduled according to time constraints (e.g., cycles or bits per second) rather than non-time-based resources such as disk or memory usage. As a result, the scheduling mechanisms for time-based resources in general will be fundamentally different than for non-time-based resources. Note that kiosk locations would enable many scenarios that would be useful to such guests. A resource such as a display monitor may be either entirely available for sharing or not at all, depending on whether or not it is presently being used. Thus, although not represented in FIG. 6, step 606 may also reject (or ignore, if appropriate) a request when there is no amount of a requested resource

available for sharing. Step 608 represents returning a response to the requesting entity, preferably the schema-based response that provides the resource capability data to the requesting entity.

Step 610 represents allowing the requested resource to be shared, and includes, for example, the exchange of data between the requesting device and the sharing device so that the sharing performs some corresponding task for the requesting device. Step 612 tests whether the resource was exhausted, e.g., timed out or is no longer available. Note that this is shown as a loop in FIG. 6 for purposes of simplicity, however it is readily understood that the actions may be event driven. Further, a step (such as step 606) may be included in the loop / used to fire the event, e.g., the resource usage is dynamically monitored to determine when it is exhausted for the particular request. Note that a resource may be directly exhausted when the requesting entity informs the device that the resource is no longer needed.

Step 614 represents notifying the requesting device that the resource is no longer available for its use. Step 616 represents disabling the resource usage, so that, for example, once exhausted the requesting device will need to make a new request if it wants to share resources.

FIGS. 4 and 5 provide examples of some of the many possible uses of the present invention. As will be understood from the examples of FIGS. 4 and 5, integrating computational proxy support into a network edge device or appliance may result in a better user experience for users of an internal network.

In a first example represented in FIG. 4, a small mobile device 400 (such as a mobile telephone) wants to print to a printer 402, however the small device 400 does not have an appropriate printer driver. This may be because the mobile device 400 does not have sufficient memory to install and maintain a printer driver needed for that printer 402, and/or does not have an easy way for the needed printer driver to be installed, and/or for other reasons. Note that this is a real scenario, and one proposed solution is to emit XHTML-PRINT as a simplified PDL (Page Description Language) from the mobile device 400 to the printer 402, e.g., over some network connection (shown via the dashed line in FIG. 4). Although XHTML-PRINT is adequate to an extent, this solution has a number of drawbacks, e.g., XHTML-PRINT is not WYSIWYG, cannot control line breaks, and generally provides a poor user experience.

However, if via the present invention, the mobile device 400 first dynamically discovers a personal computer 404

equipped with a proper printer driver 406, (and which the mobile device user is authorized to use), the mobile device 400 instead provides printing data via the personal computer 404 to the printer driver 406, which in turn will create a WYSIWYG output using PCL, Postscript or rasterization. As mentioned above, the present invention provides the discovery mechanism, and thereafter the mobile device 400 may communicate in any way with the personal computer 404, such as by a lightweight PDL that has much higher-fidelity (relative to XHTML-PRINT). The result is high-quality output rendered very quickly, by leveraging the faster processor and proper printer driver 406 in the personal computer 404.

FIG. 5 demonstrates a second example, generally directed to providing an improved user experience when operating a PDA (personal digital assistant) 500 that is to be executing managed code remotely on a server 502 (e.g., an ASP.NET server). One way to accomplish this is to have the PDA connect to the server to run the managed code, (e.g., by leveraging Mobile Controls support in ASP.NET), with the results transmitted as HTML, as indicated by the dashed line in FIG. 5. The server 502 dynamically scales the HTML content to the appropriate screen size of the device 500, without the ISV creating special-purpose code for each device 500. However, this requires many round trips with the server 500,

with the screen flashing for updates, and generally provides a degraded user experience.

An improved solution provided by the present invention is accomplished by having the PDA dynamically discover that a personal computer 504 associated with the user's identity can be a proxy for a number of computational resources, as generally represented in FIG. 5. In such an event, the PDA issues an HTML request for a remote page via the proxy personal computer 504. The personal computer 504 intercepts the request and starts a session with the remote server 502 to offload the UI rendering managed code, e.g., using WSDL as the protocol to the server 502. Typically, the bandwidth from the proxy personal computer 504 to the server 502 will be less than the bandwidth from the PDA device 500 to the proxy personal computer 504, whereby the smaller amount of data transferred over the slower link to and from the server 502 results in a more responsive user interface when displayed on the PDA 500, providing an improved user experience. Of course, the two bandwidths can be first evaluated to ensure that the proxy solution is the more-optimal one.

Further, note that if the PDA 500 was equipped with locally-running controls that the server was able to leverage, an even better experience may be provided. For example, ASP.NET automatically leverages any local controls (e.g., such

as a date picker control) running locally on a PDA running a Windows®-based operating system. This results in even less round trips to the server 502 and/or proxy 504, with less resultant screen updating / flashing.

As can be appreciated, there are numerous other examples that are enabled by having computing resources revolve around an identity, instead of around the particular devices those resources ship with. The resource schemas and service fabric are not limited to simply describing the resources, but also include elements to assist those resources working together. This can be appreciated from the following two types of scenarios, namely a set of real-time scenarios in which the user is interacting with a remote computing device in real-time, and a set of scheduled or predicted scenarios. An example of a predicted scenario may include a storage load balancing mechanism that occasionally runs on a home network, to make sure that one computing device's free storage space is not going to run out while another drive has plenty of room.

Thus, a rich schema may provide data that goes beyond describing the resources themselves, such as methods to schedule, at a particular time or at regular intervals, prioritization machinery, and data indicating how to deal with possible contention between real-time and predicted tasks, or multiple such tasks. The schema may provide for the concept

of cost, e.g., if two connections are available to the Internet, it is valuable to recognize that the user may be paying one price for unlimited access but another price for an amount of data transferred over a different mechanism, and select the better option.

Replication/sharing by a logging system may be used to understand how users are interacting with their devices, and to achieve a consolidated view of how a user operates, rather than have only disparate views separated by device. This data is appropriately known to many devices, and will provide an automatic way for such data to be consolidated. Prediction optimizations may be saved and exchanged based on monitored usage, e.g., if the system knows that a user manually downloads particular information to a hand-held computer every Friday before leaving work, the system can start the download during device idle time so the "synchronization" appears instantaneous. End user options/settings/preferences is also valuable information for the shared environment, e.g., what is a desirable or even necessary feature for some users is undesirable or even annoying to others. By maintaining this data, the end user has the power to be in control of their experience. Also, in a typical environment, there will be at least one rich client that uses the protocol/schema to perform system management, where "system" refers to the overall set of

resources associated with an identity. Algorithms and methods can use the schema for predicting such operations.

As is understood, there are many valuable scenarios enabled by through abstracting device resources and aggregating them around an identity. A number of examples are provided, however the list should not be considered exhaustive or in any way limiting, but rather serves to illustrate some of the many benefits and advantages of the present invention. As examples mentioned above, a device with a small computer screen may leverage an unused computer monitor connected to a personal computer for better display, use resources of a proxy to facilitate a better display on its own display (FIG. 5) or print to a printer even without printer driver, by connecting to a printer via a personal computer that does have the correct driver (FIG. 4).

Other examples can be seen in a very low-powered device, such as a mobile telephone, performing complex computations by using the memory and CPU of a personal computer. Stored data can be automatically and transparently moved from a first hard drive to a second when the first hard drive is getting full, including, for example, from personal digital video recorders that record video programming and need to delete old shows to make room for more recent recordings. The present invention instead moves those older shows to another storage, rather

than delete them. A portable audio player need not be able to play any type of physical media, but can instead receive and buffer audio streamed to it from a personal computer and play the audio from memory.

Other examples include allowing users to easily access their files from any device regardless of where it is stored; for example, while on the road with only a small personal computer, being able to easily open files stored on a home personal computer or work personal computer, (independent of centralized .NET data storage). A user may see all of the free space in various devices, including in various public sharepoints on a corporate network. A large-screen TV connected to an appropriate device may be used to view photographs saved on a personal computer folder, run a slideshow, watch home movies or movie trailers saved on a personal computer, and so on.

FIG. 7 provides a demonstration of an implementation 700 run using four interconnected computing systems, 702-705. The implementation 700 includes a command shell 710 and application program 712 (or similar function code) on the computing system 704, a keyboard resource 714 on the computing system 702, a storage (e.g., disk) 716 on the computing system 703, and a display 718 on the computing system 705. Thus, in this example the resources are distributed such that there is

an input resource the on computer 702, storage resource on the computer 703, and an output resource on the computer 704. The computer 704 acts as a coordinating authority for the implementation 700. Although the resources are on different computer systems, the resources are accessed in accordance with the present invention to essentially form a single functioning system.

The shell resource 710 calls the application 712, and also connects to the other devices' resources through ports 724, 726 and 728, conceptually represented in FIG. 7 as one port per resource, although a separate input and output port may be provided. Ports essentially comprise structured queues that can be written (output) to or read (input) from via a handle to that port. Ports are mobile, data is not lost, and security can be associated with a port such that no one except the intended entity will see the data intended for it.

The shell 710 used in the implementation 700 is based on a programming language referred to as xSpresso, as described in United States provisional patent application serial number 60/379,864, filed on May 10, 2002 and herein incorporated by reference in its entirety. In general, xSpresso is directed to a programming language and runtime that brokers control and data flows among asynchronous system components. Among other things, xSpresso expresses the cooperative dimensions of

programs or services, including ordering and timing, that can be programmatically verified. The language may programmatically express the passing of an organizational scheme formed from a customizable tag-based language such as XML.

In the example of FIG. 7, the numbered arrows labeled 1-4 and 6-10 represent an example of the various communications between the ports 724, 726 and 728 and the shell 710, to display the text data of a file in the storage 716 to the display 718, e.g., similar to the well-known "type" function. In the example of FIG. 7, a user of the keyboard resource 714 initiates the operation by typing the word "type" on the keyboard 714 (arrow labeled one (1)). The shell 710 receives this via the port 724 and communicates data to the port 728 for display on the screen 718, (arrow labeled two (2)), e.g., to echo what the user has input and/or to prompt for a filename of a file that stores the text that the user wants to have displayed. In FIG. 7, the arrow labeled three (3) represents the user typing in the filename, such as corresponding to an XML file, which may also be echoed (not separately labeled).

With the filename, the shell 710 then writes data (represented by arrow labeled four (4)) to the port 726 that corresponds to the storage (e.g., disk) resource 716, which

results in a read request to the storage's file system. In parallel, the shell 710 calls (the wide arrow labeled five (5)) the application program 712 to receive and output the read data. In response, the storage resource 716 begins outputting data via port 726 to the application program 712, which in turn outputs it to the port 728 of the display resource 718, as represented by the arrows labeled six (6) and seven (7). The arrows labeled eight (8) and nine (9) represent the rest of the data being transferred in a similar manner. The arrow labeled ten (10) represents the status (e.g., success) code being returned to the shell 710.

In keeping with the present invention, storage, input and output can be placed on different systems, with no change to the implementation. Legacy interoperability may be provided via common language runtime (CLR) classes that represent common legacy components (e.g., the keyboard, console and storage). System dispersion is provided in a local and transparently distributed implementation, as is explicit handling of concurrency and message-passing.

As can be seen from the foregoing detailed description, there is provided a method and system for distributed computing of device resources based around the concept of identity. The method and system abstracts the resources of a computer from the individual devices in a manner that provides

numerous benefits to a user and generally improves the user's computing experience. The method and system are schema-based, and are extensible and highly flexible.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

WHAT IS CLAIMED IS:

1. In a computing environment, a method comprising:

at a first computing device, discovering a second computing device;

at the first device, obtaining data related to the second device, the data including resource information of the second device formatted according to a schema understood by the first device; and

the first device interpreting the data to access a resource of the second device.

2. The method of claim 1 further comprising, communicating a request to use the resource from the first device to the second device.

3. The method of claim 1 further comprising, communicating identity information from the first device to the second device.

4. The method of claim 3 wherein the identity information corresponds to credentials.

5. The method of claim 4 wherein the credentials are provided by an authentication service.

6. A computer-readable medium having computer-executable instructions to perform a method, the method comprising:

at a first computing device:

locating information about a resource of at least one other device that is available for use, the information located based on an identity associated with the first computing device;

selecting a resource of a second device from the information; and

accessing the resource of the second device.

7. In a computing environment, a system comprising:

a first computing device associated with at least one identity, the first device including a first resource and configured to provide access to the first resource by at least one other device associated with the identity, the first computing device providing information about the first resource according to a resource schema associated with the at least one identity;

a second device, the second device configured to request access to the first resource; and

the second device performing a computing task including using the first resource on the first device to perform at least part of the task.

8. The system of claim 7 wherein the first resource comprises storage.

9. The system of claim 7 wherein the first resource comprises bandwidth of a connection from the first device to at least one other computing device.

10. The system of claim 7 wherein the first resource comprises processing cycles of a processor.

11. The system of claim 7 wherein the first resource comprises an input method.

12. The system of claim 7 wherein the first resource comprises an output method.

13. The system of claim 7 further comprising a third computing device connected to the first computing device, wherein the second device accesses the first resource to perform an operation with the third computing device.

14. The system of claim 7 wherein the third computing device comprises a printer, and wherein the first resource corresponds to a printer driver on the first computing device.

15. The system of claim 7 wherein the third computing device comprises a server, and wherein the first computing device acts as a proxy to the server for the second computing device.

16. The system of claim 15 wherein the bandwidth from the second computing device and the first computing device is greater than the bandwidth from the first computing device to the third computing device.

17. A computer-readable medium having stored thereon a data structure, the data structure comprising:

a first field for identifying a type of resource;

a second field for identifying a device that includes a resource corresponding to the type of resource identified in the first field;

a set of at least one third field, each third field in the set corresponding to the second field for identifying

information about the device resource identified by the first field; and

wherein the data structure is used to provide resource information to another device that is different from the one identified in the second field such that the other device can use a resource identified in the third field set.

18. The data structure of claim 17 wherein the first field includes information corresponding to a storage resource type.

19. The data structure of claim 18 wherein the third field set includes information corresponding to total size data of the storage resource.

20. The data structure of claim 18 wherein the third field set includes information corresponding to free space data of the storage resource.

21. The data structure of claim 17 wherein the first field includes information corresponding to bandwidth data.

22. The data structure of claim 17 wherein the wherein the first field corresponds to processing power.

23. The data structure of claim 22 wherein the third field set includes information corresponding to speed data of a processor.

24. The data structure of claim 22 wherein the third field set includes information corresponding to usage data of a processor.

25. The data structure of claim 22 wherein the third field set includes information corresponding to size data of a memory.

26. The data structure of claim 25 wherein the third field set includes information corresponding to usage data of a memory.

27. The data structure of claim 17 wherein the second field includes information identifying the device as a personal computer.

28. The data structure of claim 17 wherein the second field includes information identifying the device as a disk drive.

29. The data structure of claim 17 wherein the second field includes information identifying the device as a telephone.