AwareWare: An Adaptation Middleware for Heterogeneous Environments

Qiang Wang, Liang Cheng
Laboratory Of Networking Group (LONGLAB, http://long.cse.lehigh.edu)
Department of Computer Science and Engineering, Lehigh University
Bethlehem, PA, USA, 18015
qiw3@lehigh.edu, cheng@cse.lehigh.edu

Abstract-A heterogeneous environment consists of a number of dissimilar networks, computing devices, end users, applications, and environmental conditions. Traditional distributed applications are generally not aware of the heterogeneities of the environment. In this paper we present "AwareWare", which is a middleware that facilitates applications to be more adaptive in such a heterogeneous environment. AwareWare addresses five types of heterogeneities and advocates dynamic component reconfiguration as a unified approach for both architectural and application level adaptation. AwareWare includes environment measurement tools, an adaptation decision module that is separated from other constructions of the application, and reconfiguration mechanisms for component based distributed applications. Prototype examples are presented that demonstrate the potential use of the middleware.

I. INTRODUCTION

The terminology "heterogeneous" in Merriam-Webster dictionary is defined as "*consisting of dissimilar or diverse ingredients or constituents*". A heterogeneous environment is a distributed computing environment which consists of a number of *dissimilar* computing elements, e.g. networks, devices, and etc. Heterogeneity is an important characteristic of the today's computing environment: heterogeneous communication paths across both wired and wireless domains in data networks [8]; various applications have been developed on portable devices to communicate with PCs [7]; Web content delivery starts to take into account of different display sizes of clients' devices [9]… With the advances in data communication and hardware development, the computing environment will be more heterogeneous. However, most traditional distributed systems (e.g. groupware) assume that the environment is homogeneous, i.e. computers with similar capacity are interconnected through same type of network, and applications communicate with each other without the awareness of the conditions of their peers. With the presence of many different types of networks, devices, and applications in use, a more sophisticated distributed system must be able to handle heterogeneous environments as well.

Challenging problems [10] arise when system developers address the heterogeneity. The effort to develop a sophisticated distributed system in a heterogeneous environment is considerably extensive, which motivates us to develop a middleware called AwareWare as a general framework to facilitate adaptive application construction.

The rest of the paper is organized as follows. Section II discusses five types of heterogeneities AwareWare addresses. The analysis of the characteristics of heterogeneous environments is the foundation to identify the most commonly used functions as middleware APIs for application developers. Section III summaries the related work. Section IV introduces the middleware framework and advocates dynamic component reconfiguration. The middleware consists of awareness management tools, an adaptation decision module, and reconfiguration mechanisms as a general approach for adaptation to component based distributed applications. Section V presents a traditional groupware system and an adaptive sensor network as example applications that use the middleware. Section IV discusses and concludes the paper.

II. AWARENESS FOR ADAPTATION

In this section, we discuss five major sources of heterogeneities considered in our middleware, as shown in Figure 1, i.e., the network, device, end user, application, and environment. Based on awareness of these heterogeneities, applications may adapt their behaviors accordingly.

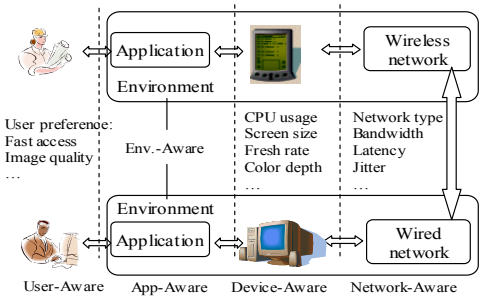

Figure 1: A Heterogeneous Environment

Network-awareness: In heterogeneous environments, multiple devices may connect to each other via different network links, varying from high-speed LAN, dial-up, to wireless connection, where the characteristics of the network are different from each other. Even with a known network connection, dramatic network performance changes are often experienced during a particular communication session, especially in wireless networks. An individual wireless channel is also subject to path loss, fading, and environmental interference, which presents the need for special treatment for

wireless communication in many applications. The capability to detect the existence and characteristics of wireless links is included in our middleware. In addition to the application's own awareness, the application may also need to know the information known to its peers (the peer's awareness). Bandwidth, latency, jitter, and etc. are commonly used to specify the network characteristics.

Device-awareness: The need for device-awareness lies in the fact that computing devices (e.g. PC, Palm) that host applications in heterogeneous environments may vary significantly in their device capacity parameters, namely CPU power, display size, memory size, display refresh rate, and etc. Moore's law leads the computing performances to grow exponentially, making the mixed use of powerful computers and older computers a common place in the real world. Devices are more diversified in pervasive computing, where different types of devices are interconnected. The inequity of the devices in a communication session can makes the devices with lesser capacity vulnerable to large amounts of information generated by high end devices, therefore results in degraded user experience (and even worse, the machine may stop responding to the local user's input). It is necessary to protect the less capable machines from being overwhelmed. Further, different devices may require different presentation of the data. For example, Internet Content Providers may need to change their content presentations in order to fit different display size of many hand held devices, e.g. Palm, cell phone, and etc.

User-awareness: End user is one of the most dynamic factors in a computing environment, since they may have very different preferences for a single application. For example, some Web users may favor a faster download over the quality of the content, while others vice versa. Users' preferences may also change dynamically. User awareness can be collected in an explicit or implicit way. In an explicit approach, users can specify their preferences through Human Computer Interfaces, i.e., by selecting menus and dialog boxes. In an implicit approach, software can identify the users' preferences intelligently by using an intelligent agent and/or machine learning algorithms.

Application-awareness: Internal states of local and remote applications can also be useful for adaptation. For example, in a distributed Virtual Reality (VR) game, each application broadcasts current avatar position/states to their peers. The time for a graphic engine to render a scene varies significantly with different scene complexities [11]. Knowing the rendering time of a receiver may let an application adapt its data sending rate. Therefore, adaptation middleware needs to exchange the shared internal states across the network.

Environment-awareness: Physical environment can be measured by sensor networks. A sensor network consists of many sensor devices that measure the environment, and communicate with each other through wireless link. Adaptation can be triggered by external environmental events. One example application is discussed in Section V.B.

After identifying five most significant parts of heterogeneities, the goal of the AwareWare middleware is to address them in a generic way to facilitate adaptation.

III. Related Work

Heterogeneous environments attract more and more attention in research community. Fig. 2 shows the related work in an organized way. Most existing adaptation solutions either focus only on network awareness (fixed networks, or wireless networks) or device awareness (end hosts).

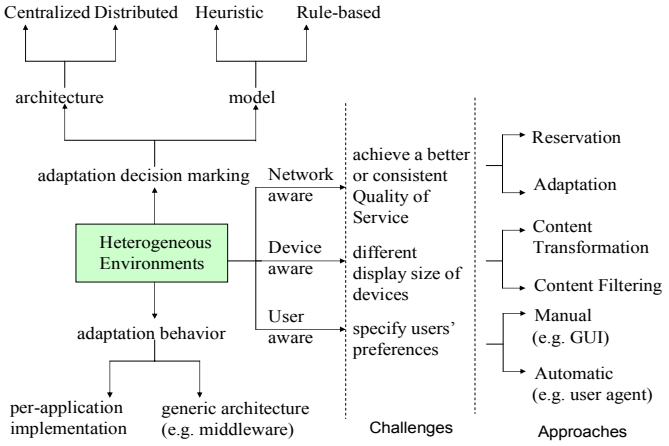

Figure 2: Classification of Related Work

The research in network aware generally aims to achieve a better or consistent QoS (Quality of Service) over heterogeneous network connections. Reservation and adaptation are two basic approaches to achieve a better or consistent QoS in heterogeneous networks. In a reservation-based system, the system will dedicate the resources necessary to an application to provide a certain level of quality. There exists a significant amount of research in this area. AwareWare is an adaptation approach. The advantage of implementing adaptation based systems in middleware level is that it does not require tight integration or modification of the best-effort services in Operating System and network protocol stack.

In adaptation based systems, two approaches exist, and each with distinct focuses. One approach is to dynamically reconfigure the middleware itself, thus legacy applications do not require any modifications. The other approach is that reconfiguration occurs in the application level, which offers more appropriate application specific adaptation choices. AwareWare uses a hybrid approach, where message filtering and transformation for heterogeneous devices are handled by middleware reconfiguration, and application behavior adaptations are realized through application reconfiguration.

Examples of adaptation based systems include Conductor [6], Agile [4], and etc. Darwin project at UC-Berkeley [1] provides a scalable and highly available proxy to assist mobility and dynamic adaptation. Odyssey [5] is a platform for adaptive mobile data access. Odyssey's approach is to adjust the quality of the data which the mobile user tries to access to

match available resources when there is less bandwidth. Bolliger and Gross [12] presents a framework-based approach to construct network aware applications. The framework provides solutions to two fundamental challenges in network-aware applications: how to find out the dynamic changes in network service quality; and how to map high level application centric quality measures to low level network-centric quality measures.

Device aware research usually focuses on the services that could support the difference type of devices, through content transformation and filtering. By tuning the content for particular devices, Darwin supports dynamic adaptation of content for thin clients [1]. Correa and Marsic [13] describes the content transformation for heterogeneous devices to collaborate in Virtual Reality. The mappings (i.e. the semantic transformation) transform an object's 3D coordinates generated from a 3D VR computer to 2D coordinates to be used in a 2D hand held device. Furthermore, by specifying the different levels of data consistency policy (varies from continuous update, threshold update, action change update, to none update) to different types of devices, network traffic can be reduced for devices with poor network connections. However, the central server approach they used in the research limits the scalability of the system.

For adaptation decision making, from architecture point of view, systems are either centralized or distributed, where a centralized architecture uses a central server to manage awareness information and each application can get an indication of the resources available to it. A distributed architecture needs to exchange the awareness among distributed applications and the decision making probably depends on partial information available. Trefftz, Marsic, and Zyda [14] demonstrate a switchboard architecture as the decision making algorithm to handle the heterogeneity in virtual environments, where computing power, network speed, and users' preferences are different for different participant. By solving the linear equations that defines policies and users' preferences, the solutions are mapped back to control each applications' modality, e.g. to control the frequency of frame updating. The main objective of their scheme is to allow slower nodes to participate in the session by preventing fast nodes from flooding slow nodes with too many messages. Their approach also uses a central server as the resource manager, therefore the adaptation decision making is centralized. Our system differs from this architecture by providing a distributed adaptation decision making as another choice, therefore would be better suited for many other situations, where the scalability is major consideration, or the centralized server architecture is not available.

IV. Middleware Architecture

Adaptation middleware is a software system situated between the operating system and applications with flexible adaptation support. The AwareWare middleware consists of five parts, as shown in Fig. 3: Awareness measurement tools for network, device, end-user, application, and environment; adaptation decision running time system; adaptation policy language; message filtering and transformation; and dynamic reconfiguration interface to applications.

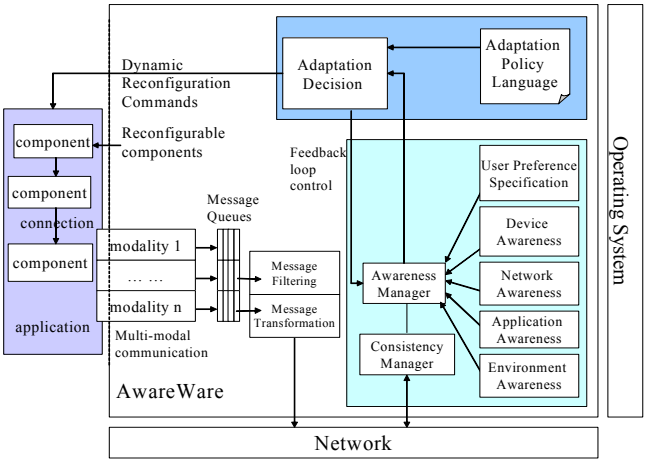

Figure 3. AwareWare Middleware Architecture

A. *Awareness Measurement Tools*

There are many existing network awareness measurement tools that collect network performance [2] (e.g. bandwidth, latency, jitter, and etc.). Device characteristics (e.g. CPU usage, memory, display size, frame refresh rate, battery consumption, and etc.) are usually obtained through system APIs. The user preference, in terms of high level expectation of the service, is generally specified by graphic user interfaces. Intelligent agents and advanced interaction interfaces are interesting approach to identify users' preference automatically or semi-automatically.

B. *Awareness Organization and Synchronization*

Awareness is organized in a tree structure and can be easily added, retrieved, and modified according to the name convention. This approach provides a flexible way to integrate more measurement tools, if needed in the future.

In distributed architecture, a number of synchronization mechanisms can be used to maintain different levels of consistency for awareness across the network. The consistency management in middleware can be categorized into centralized architecture and distributed architecture. We plan to support both architectures. In the central architecture, there is a central server to manage all awareness information from clients. Therefore decision making is a centralized algorithm, based on all information available at the server. In the distributed architecture, each client maintains replicas of awareness from other clients, and a consistency policy is assigned to each individual replica in the time when the replica is initially cloned. Since different applications may require different levels of consistency, several consistency policies are included in the middleware:

1. Automatic synchronization: An awareness producer synchronizes its shared object whenever any changes occur.
2. Lazy synchronization: Synchronize the shared object only when a receiver needs its value.

3. Conditioned synchronization: Only need to synchronize the shared object when application-defined conditions are met.

C. *Dynamic Reconfiguration*

Adaptation tactics are specified in an adaptation policy language. The language is interpreted and executed by run-time support system.

From an architectural point of view, adaptation can be seen as the reconfiguration of the system architecture with respect to environment changes. Dynamic reconfiguration [3] is used in AwareWare as the basis of the adaptation. AwareWare controls applications which are configurations of components where the configuration adapts at runtime. Therefore an application consists of multiple *reconfigurable components* interacting with each other through *connections*. A connection is a link between one interface of a component and one interface of another.

Adaptation functions usually are mixed with other codes, making checking and verification a difficult job. However our approach treats adaptation and configuration behaviors explicitly and consistently, and separates the *adaptation tactic* from the application. We implement a CORBA packaging template for each dynamic reconfigurable component with several standard *reconfiguration primitives*, (e.g. blocking/unblocking a connection, serializing/restoring internal states of a component, binding/unbinding a connection etc.). Packaging codes for each component also handles connections between components, tracks the component's state, and responds appropriately to reconfiguration primitives. Thus complex reconfiguration mechanisms are handled transparently to developers.

AwareWare uses an adaptation policy language to describe adaptation tactics. The adaptation tactics are specific to the application. Thus the developer decides what awareness should be used, when reconfiguration should be invoked, and how an adaptation tactic should be expressed in terms of reconfiguration primitives. The language also contains the specification of required and provided interfaces for each component, and the interaction protocol between components. Several tools are also proposed to analyze the specification for correctness and automatically generate the configuration infrastructure. The configuration infrastructure manages instantiation and connection of components and enforces the specified configuration behavior at runtime.

The decision module is a virtual machine that interprets the adaptation tactic at run-time. The reconfiguration primitives are used by the decision module to interact with components when making a configuration change; and these are standard primitives, not defined by system developers.

V. EXAMPLE APPLICATIONS

A. *Adaptive Remote Application Sharing*

Remote application sharing (i.e. remote screen sharing) is a groupware system that captures the screen image and sends it to the remote computers frame by frame frequently. Traditional application sharing systems (e.g. Microsoft NetMeeting) send the whole screen to a remote user, generally without awareness of networks, devices and different user preferences. We propose an adaptive application sharing system using AwareWare.

AwareWare supports multi-modal communication. A program has been implemented to identify all windows components on the screen. In this way, each window can be viewed as a modality (different communication channels) such that each can have different adaptation behaviors. A user interface can relate each window to different user preferences. For example, the top-most window can be specified as "the most important" one and the related user's preference is "good response time". Other windows are "less important". When the network connection is poor (slow), the frame update rate of the "less important" windows could be less frequent and the image quality (in term of color depth of the screen capture image) of these "less important" windows can be sacrificed. In this way, the quality of "the most important" window is well preserved.

In each frame, each captured screen image is relatively large. For example, a true color (4 bytes per pixel) image of a 1024 * 768 pixel screen is of approximately 3.14 M bytes. An interactive application needs to send 10 update frames per second to the remote computer, making compression compelling in order to communicate to the remote computer through low bandwidth connection, for example, dial-up connection. However, compression demands CPU power and increases overall delay. When CPU is busy and available network bandwidth is sufficient, it is more interactive for the application to send the data without compression. The decision of whether the data needs to be compressed and which compression algorithm to use is the key decision making.

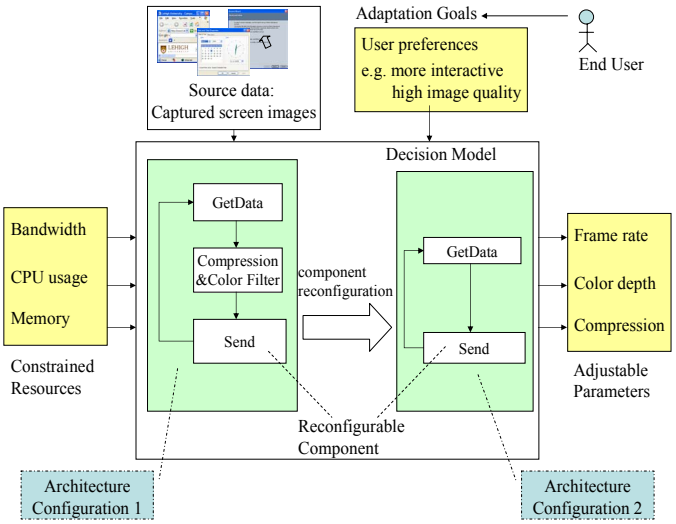

Figure 4. Adaptive Remote Screen Sharing

The concept of dynamic reconfiguration is further illustrated in this application, as shown in Fig. 4. It shows that the decision making is a process of satisfying the adaptation goals under the constraint of some resources, and use dynamic

reconfiguration as the basic mechanism to adjust application behaviors. The application has two different configurations. One configuration consists of the interconnection of three components, as shown in the left part of Fig. 4, to get the screen capture data, compress the data, and then send the data. Another configuration is the interconnection of two components, as shown in the right box of Fig. 4, to get the screen capture data, and then send out without compression. AwareWare monitors bandwidth, CPU and memory usage, and decides which configuration to use, in order to fulfill user perceived quality of service. By dynamic reconfiguration, AwareWare changes application's component configuration and behavior dynamically.

B. Adaptive Sensor Networks

Using AwareWare, a sensor network can also work adaptively. The sensor network is integrated with the Internet. It consists of many distributed sensor nodes which are deployed to monitor the environment such as temperature, light, object movement, and etc.

Adaptation middleware serves not only as a gateway for the sensor network to communicate with applications across the Internet, but also contains the adaptation policies which control the adaptation behavior of both the applications and the sensor network. Thus the AwareWare provides a feedback loop between applications and awareness modules. Consider a sensor network deployed in a forest for wild fire monitoring. In normal situations, sensors send environment data periodically. When the monitored temperature becomes higher than a threshold, which means that the forest might be on a "red alert", sensor nodes will be triggered to collect the temperature data more frequently and other new information, e.g. humidity. In one word, the awareness module (e.g. data collection frequency, source, and fidelity) by the sensor network can be adaptive through the feed-back loop.

VI. DISCUSSION AND CONCLUSION

In this paper, a novel architecture for adaptive middleware is presented, and examples are illustrated to substantiate its applications. The framework provides a promising general solution of adaptation for heterogonous environments. Our system differs from previous research in four major ways.

First, our system targets five sources of heterogeneity, as discussed in Section II. Heterogeneous environment is an active research area; however, there are few researches that consider five sources of heterogeneity in an integrated platform. Existing adaptation solutions either focus only on network awareness (fixed networks, or wireless networks) or device awareness. The lack of the systematic approach to all heterogeneities motivates us to develop AwareWare.

Secondly, component reconfiguration is used as a unified approach to handle architecture level and application level adaptation. Component reconfiguration mechanism not only can handle planned changes, e.g. the change of application architectures corresponding to different adaptation tactics, but also can handle unplanned changes. With the emergence of different awareness measurement tools, different devices, and different decision modules, the ability to handle unplanned changes makes the system architecture for functionality addition and changes. Furthermore, adaptation tactics usually are mixed with other function codes, making checking and verification a difficult job. In our approach, the adaptation tactics are separated from other constructs of the application.

Thirdly, AwareWare provides a feedback loop for awareness modules inside the middleware, as described in Section V.B. In other words, the behavior of the awareness modules will also be adaptive to the environment, in addition to the applications' adaptive behavior. Current research in awareness measurement simply treats measurement tools as an input module. The feedback loop and interaction of different awareness information further control the behavior of awareness measurement tools. We believe that it needs to be addressed in the middleware level since the measurement is an inherent part of the middleware and the adaptation of measurement tools itself is required by many applications.

ACKNOWLEDGMENTS

We thank Dr. Christine Hofmeister, Li Yu for discussions on dynamic reconfiguration. Valuable comments from Scott Frees, and members at LONGLAB are greatly appreciated.

REFERENCES

[1] Brewer, E.A., Katz, R.H., Chawathe, Y., Gribble, S.D, and et al., A network architecture for heterogeneous mobile computing. *IEEE Personal Communications,* 5(5), Oct. 1998.

[2] Cheng, L. and Marsic. I., Accurate bandwidth measurement in xDSL networks, *Computer Communications,* Vol. 25(18), pp. 1899-1710. Nov. 2002.

[3] Agnew, B., Hofmeister, C., and Purtilo, J., Planning for change: a reconfiguration language for distributed systems, *IEE Distributed Systems Engineering,* 1(5): 313-322, Sept. 1994.

[4] Li, B., *Agilos: A Middleware Control Architecture for Application-Aware Quality of Service Adaptations*, Ph.D. dissertation, UIUC, 2000.

[5] Noble, B., System support for mobile, adaptive applications, *IEEE Personal Communication,* Feb. 2000.

[6] Yarvis, M.D., Reiher, P., Gerald, J.P., *Conductor: Distributed Adaptation for Heterogeneous Networks*, Kluwer Academic, 2002.

[7] Isaacs, E., Walendowski, A, Ranganathan, D., Mobile instant messaging through Hubbub, *Communications of the ACM,* 45(9), Sept. 2002.

[8] S. Biaz, *Heterogeneous Data Networks: Congestion or Corruption?*, Ph.D. dissertation, Texas A&M University, Aug. 1999.

[9] Mérida, D., Fabregat, R., Urra, A., and Bueno, A., Analysis and regeneration of hypermedia contents through Java and XML tools**.** *Proceedings of the International Conference on Information Technology: Computers and Communications,* April 2003.

[10] Daoud, F. and Mohan S., Challenges of personal environments mobility in heterogeneous networks, *Mobile Networks and Applications*, Vol. 8, Issue 1, Feb. 2003.

[11] Funkhouser, T., and C. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments, *Computer Graphics*, Vol. 27, pp. 247-254, 1993.

[12] Bolliger, J. and Gross, T., A framework-based approach to the development of network-aware applications, *IEEE Transaction on software Engineering*, Vol. 24, No. 5, pp. 376-390, 1998.

[13] Correa, C.D. and Marsic. I., A flexible architecture to support awareness in heterogeneous collaborative environments, in *Proceedings of the Fourth International Symposium on Collaborative Technologies and Systems,* pp. 69-77, Orlando, FL, Jan. 2003.

[14] Trefftz. H., Marsic. I., and Zyda. M., Handling heterogeneity in networked virtual environments, *Presence: Teleoperators and Virtual Environments*, Vol. 12, No. 1, pp.37-51, Feb. 2003.