CME: A Middleware Architecture for Network-Aware Adaptive Applications

Jun-Zhao Sun, Jari Tenhunen, and Jaakko Sauvola
MediaTeam, Machine Vision and Media Processing Unit, Infotech Oulu
P.O.Box 4500 4SOINFO, FIN-90570 University of Oulu, Finland
E-mail: junzhao.sun@ee.oulu.fi

***Abstract*—Network QoS parameters experience great fluctuations during the execution of network applications. Especially in mobile and pervasive computing environments the variation becomes more serious. Applications need to adapt their functions to the changing network status. Moreover, an enhanced software platform is necessary to provide adaptive network management services to upper software components. This paper proposes CME, a middleware architecture for service adaptation based on network awareness. CME is structured as the software platform both to provide network awareness to applications and to manage network resources in an adaptive fashion. The overall architecture of CME framework is descried. Core components include connection monitor, policy manager and connection controller. Adaptation mechanisms are introduced in detail.**

I. Introduction

Network applications relay on the underlying communication infrastructure to provide access to remote services and resources. Ideally these applications do not concern anything about the networks used and could only focus on the service functionalities they provide, but in practice this can never be true. Large variations in network quality of service (QoS) (e.g. bandwidth, latency, jitter, reliability) may occur during the information exchange.

The situation of variations in network QoS parameters becomes to be more serious when two concepts are taken into account, network convergence and mobile communications. All data networks are converging on IP transport and applications are also transforming towards being IP-based. However the underlying medium and link accessing techniques can be greatly diverse, mobile or fixed, wireless or wired. The converged network also mixes different types of traffic, each with very different requirements.

Mobile communications are mostly based on portable devices and wireless network connectivity. The end user's terminal can be greatly varying in terms of e.g. processing and storage capabilities, input and output capacities, energy consumption, and networking technologies. There are different wireless access technologies including e.g. WLAN, Bluetooth, GSM, GPRS, UMTS, etc. Wireless access networks greatly vary by nature, with regard to e.g. error rate, fading, interference, etc. User mobility leads to the continuous changing of location and environment, network operator and service provider, and access networks. Handoff is one main source of network variations in terms of packet delay and packet loss. As the new computing diagram for the next generation mobile computing, pervasive computing [1] introduces more variations to network performance where the communication technologies could be highly diverse and overlapping in a smart space. To specify, one user terminal may be equipped with multiple connectivity technologies ranging from wireless sensor network and short-range ad hoc connection up to local and wide range connections.

Current QoS model [2] are mainly oriented totally towards low-level network parameters such as bandwidth, latency, and jitter, targeting to provide a transparent management on network transport system to upper applications. However in many cases the mechanisms for this model such as resource reservation are neither sufficient nor even feasible. There is an increasing need for network applications to be aware of the variation in network performance and quality. Applications should be more intelligent in order to adapt to various network environments, e.g. the access technologies being used and the greatly changing network status. On the other hand despite the differences in the functionalities of the applications, the main networking services can still be abstracted to common and unified interfaces to be used by diverse applications. Adaptation functions need to be rationally distributed into both specific applications and a general system platform, and then implemented with distinct mechanisms.

The paper presents CME (Connectivity Management Engine), a middleware architecture for adaptive applications based on network awareness. CME provides flexible network supports to adaptive network applications with a set of functions and interfaces. We describe the overall architecture of CME in this paper, with the emphasis on the methodology of how to make application and network management context aware and how to provide adaptive networking services to network-aware applications. The rest of the paper is organized as follows: Section II presents the overall architecture of CME. Section III discusses the context considered in CME and the methods of how to realize context awareness. In Section IV we discuss the realization of adaptive network supports and the utilization of them in the adaptation of application. Section V concludes the paper with a discussion of future work.

II. CME Architecture

A. *Architecture and Execution Environment*

The CME architecture is illustrated in Figure 1, together with other components in the execution environment. The execution environment includes the adaptive application as the consumer of the CME's networking services, and underlying infrastructure that serves as the necessary supporting modules.

The infrastructure mainly consists of the components under the monitoring, control, and query of CME. To network monitoring, devices include various network equipments ranging from adapter card, modem, and access point, up to router and gateway. To network control, protocol entities include interfaces for the management of e.g. network device drivers, protocol stack, routing table, etc. There are also servers for the storage and query of context information. The context server can be both locally positioned in an end host and globally accessible as a centric server. Section III discusses context and context awareness in more detail.

CME middleware is the software platform above the operating system and other resource infrastructure, providing adaptive network connectivity management to upper system modules and network applications. In particular, connection monitor is responsible for the collection of transient network information both of a local host and in an end-to-end fashion. CME's caller uses policy manager to express adaptation rules. This is an easier way for the utilization of the adaptive services provided by CME, in which the application needs only to represent the requirements by policies. The detailed processing of the adaptation demands can then be left to connection controller without the concrete concerns from the applications.

Connection controller forms the management core of CME middleware. The management is realized by the creation and maintenance of connection channel. A channel is the logical link between physical application components that are located in different network devices e.g. terminal or server. Each channel uses a specific type of connection to transfer data between devices. The connection for one channel can be dynamically changed, while leaving the channel unchanged and so making applications imperceptible. The application may explicitly control each channel if necessary, but in most cases it is done implicitly by policy manager.

The specific adaptation function related to an adaptive service should be implemented as part of the application where the adaptive activities can be finally determined. The adaptation of a provided service can be realized through multiple serial stages, including adaptation triggering, approach selection and adaptation execution. Adaptation mechanisms are first triggered by some specific context according to the predefined matching criteria. Then a decision should be made on which adaptation approach will be used. Finally, service adaptation can be achieved by automatically or manually executing a command and/or changing the external behaviors (and possibly internal states) of an entity that provide the service. CME provides the adaptive network management services with a set of APIs to the upper network-aware adaptive applications. From CME's point of view and as the caller of the CME's functions, these applications are mainly end service applications but can also be other system level platforms at middleware layer e.g. file access system.

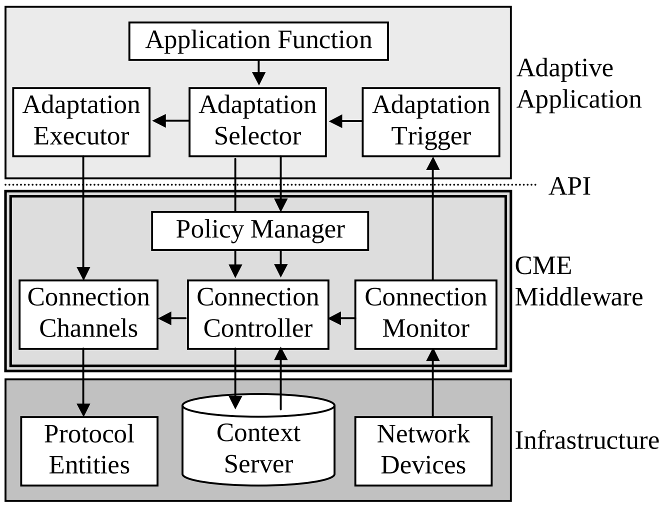

Figure 1. CME architecture and execution environment

The main feature of the CME architecture is the clear partitioning of the whole adaptive functionalities into different levels, in which each level only takes care of the functions that are most suitable to be concerned by it. The adaptation of the end application is separated into the application layer and the middleware layer. All the network adaptation mechanisms are abstracted and placed onto the CME middleware level, since the monitoring and control of network resources are mostly convenient to be implemented at this level. Semantic oriented adaptation mechanisms are then left to application level. This is due to the fact that application knows the content and the media that it consumes and processes the best.

B. *Edition and Operation Modes*

1) Edition: The architecture described above includes the necessary functions for CME, while in practice there are different editions of CME according to the different execution domains. Basically two editions of CME can be defined, i.e. client CME for mobile host and server CME for server. CME functions should be tailored to fit different editions.

Client CME is mainly used by user terminal, e.g. laptop, mobile phone, PDA, etc. Local network context for user terminal could be greatly fluctuant. One mobile host can be equipped with several network interfaces such as Bluetooth, IrDA, WLAN, and GPRS. Availability for each access technology is dynamically changing due to user's movement. Network performance may also be greatly fluctuant in terms of packet loss, latency, and bandwidth as the result of the wireless connection and handoff. Moreover, client CME should consider other context information such as energy consumption, user preference, and cost. This is mainly due to the restrictions in user device capabilities. As a result, functions in client CME focus more on the adaptive management of networking resources than application level adaptation. In particular, the adaptive management of multiple communication channels is the main task of client CME.

Server CME mainly targets servers for various network applications such as web service, information retrieval, file access, messaging, etc. The configuration and operation environments of a server are relatively stabile, i.e. usually server is permanently equipped with network devices and never mobile. So local network resources are easy to be managed. On the other hand, one server is shared by a huge number of clients that come out to be with different characteristics running in different surroundings. So for server CME functions mainly concern end-to-end network monitoring and cooperating with client CMEs to assist the connection control. Specific application adaptation mechanisms for intelligent services play more important roles than local network resource management.

2) *Operation mode:* To manage connection channels is one of the core functions of CME, which is realized by connection controller. Channel is the logical connection between two communication peer entities. According to the situation of whether CME resides in both peers or only one end, there can be two different operating modes for CME architecture, standalone operation mode and collaborative operation mode. Note that the different operation modes are defined from each channel's viewpoint. There can be plenty of channels connecting to different destinations, which is all under the control of the CME in the host. The differences of the two operation modes can only be perceived by the CME, not the upper applications.

For CME running under the standalone operation mode, the channel can only be manipulated at the local CME end. In this case the functions of the CME are heavily tailored and simplified. Only some rough information of the peer can be collected by connection monitor. To server CME most of the channel control functions are reduced to normal operations. As for client CME it is still possible to control the channel in an adaptive way to a certain extent. Examples include to open a new channel with the consideration of policies and to dynamically switch connections within an existing channel when network context change. If the peer is a mobile host moving between different access points and so keeps changing the IP address under using, there is no way to rearrange the connection with the CME.

Collaborative operation mode is one the main interests of this paper, and it is illustrated in Figure 2. Channels are now under the end-to-end control of the CMEs at both sides. This is realized by the coordination between the two connection controllers with a particular UDP signal link. Moreover, the global context server provides another indirect way for the cooperation between peers. The collaborative operation mode can also benefit connection monitor with respect to the more accurate detection of end-to-end network QoS parameters such as bandwidth. The detailed collaboration of the two connection peers is introduced in the following two sections.

III. CONTEXT AWARENESS

CME is essentially a middleware architecture for context-aware network management and adaptive applications. There are two components related to context awareness, i.e. connection monitor and context server.

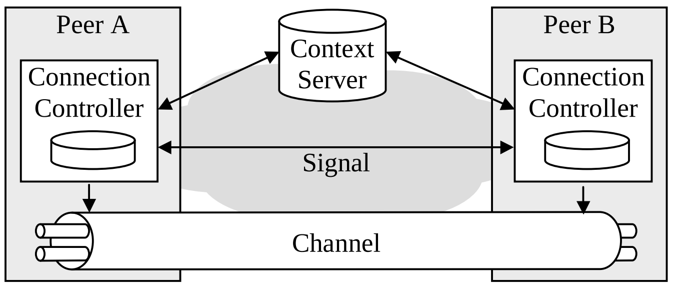

Figure 2. Collaborative operation mode of CME

A. *Connection Monitor*

Connection monitor mainly takes care of the collection and organization of network configuration and context information. This information is then used by applications and connection controllers for the purpose of both normal functions and adaptation. Both local and end-to-end network information can be monitored. Local network information includes:

- Fixed information, containing information that is the same across all the network interfaces in the host, e.g. host name, domain name, DNS servers, and node type.
- Number of network interfaces, due to the fact that a modern mobile device may be equipped with multiple network interfaces, e.g. Bluetooth, IrDA, modem, Ethernet, WLAN, GSM, GPRS, etc.
- Information of each network interface, including name, type, physical and IP addresses, gateway, DHCP server, speed, configuration parameters (e.g. dial-up number, user account, password, etc.), traffic workload at local interface and the access point being used, error rate, signal strength, SNR, power consumption, and operation status (e.g. available, operable, connecting, connected, sleeping, idle, transmitting, receiving, unconnected, unreachable, disabled, etc.).
- Packet statistics information, e.g. received, sent, and dropped packets of protocols of e.g. IP, ICMP, TCP, and UDP, etc.

End-to-end network information includes:

- Connection information, e.g. route availability, best route, and for each connection, local address and port, as well as remote address and port.
- Network QoS parameters, e.g. availability, available bandwidth, delay, response time, jitter, loss, etc.

There are three different methods of obtaining these pieces of network information from underlying infrastructure. Explicit query can be used to ask for network information when the monitor needs to know some specific information. Polling is to keep fetching context periodically from infrastructure. Event-driven method is to subscribe some special network events and then be informed when they happen. For local information when driver interfaces are not sufficient, some probes can be used to detect more precise information. For end-to-end network information, cooperation between CMEs can be very helpful in obtaining more accurate information. The similar ways as in obtaining context are adopted for the provision of the context information. That is, network information can be delivered to the applications and connection controller under the explicit query, periodical polling, and event subscription. In any case, the connection monitor plays a reactive role, i.e. calmly waiting for query or passively reporting event occurrence.

B. *Context Server*

CME provides adaptive applications with the capacity of network awareness through the information provided by the connection monitor. On the other hand, CME by itself realizes an adaptive network management by taking both network connection information and other rich context information into account. The additional context information is organized in local and global context server, maintained by other specific system software, and then accessed through interfaces by connection controller and applications. CME also contributes to the content of the two context servers with local network information, e.g. to register current network information to global context server. However in most of the cases CME just queries interesting information from context servers for adaptive management.

1) Local context server: Local host context information is organized in local context server, and can only be accessed by local CME and applications. Widely speaking the server can be a well-defined database with context data stored, or a software component that collects context information for callers in realtime. There can be various context information in the server. CME concerns only those that are related to connection management, including:

- Static network information, e.g. theoretical network capacities of each interface (type, typical bandwidth, cell size, handoff latency, power consumption, user speed, simultaneous user number of the access point), operator information, charge model and rate, etc. The static features allow a rough comparison between interfaces without measuring the runtime performance through each interface. This original information is generated through another software interface or even manually, instead of by connection monitor.
- User profile related to network management, e.g. user preference on particular network interface (default interface), priority of each interface, interface selection policies, etc. User policies are the way to balance all the factors for the interface selection by user him/herself. There are also application policies specified by each application that is willing to open a channel for its traffic flow.
- End host information, e.g. energy status, velocity, cell dwell time, distance to access point, etc. This information can be very valuable for the adaptive management of network resources. For example when the velocity of an end host is too high, the channel may maintain the GPRS connection even though a better WLAN connection is available, and then reduce the quick switch between the two accesses.

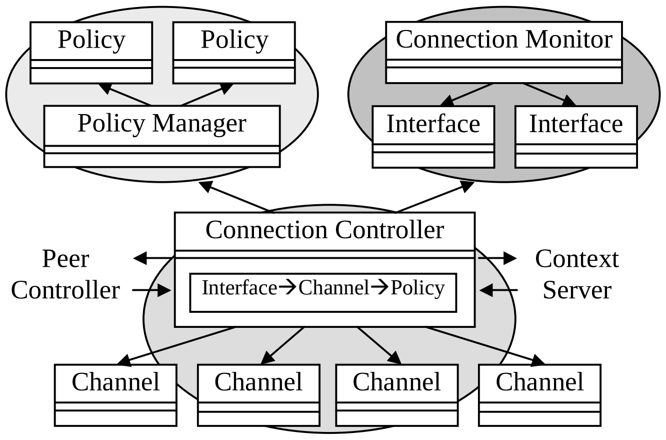

Figure 3. Object model of CME

2) Global context server: The global context server is basically a distributed or centralized storage that is accessable by any authorized terminal or server. Any context information can be shared in the server, while CME only concerns those related to adaptive connection management. Generally all the information collected by connection monitor and local context server can be stored in the global context server. However in practice the server may only contain some basic and relatively steady network information of an end host.

In particular, the network information stored in the global context server can be organized as two hierarchies. The primary element is about the host name, how many interfaces it is equipped with and the name of each interface. Each secondary element describes the type, address, and selection priority of one network interface. Each element has also a time stamp field to denote the time of the record generation. Local connection controller registers the information periodically or in case that there is any connection change event. The information is mainly used by peer hosts to select an incoming interface and to get its current IP address when initializing a channel to the host. From this viewpoint the global server serves as a dynamic DNS.

IV. ADAPTATION

The adaptation mechanisms for the network management in CME is mainly realized by policy manager and connection controller. In particular as the object model shown in Figure 3, the kernel of CME is the connection controller acting as a coordinator and executor.

A. *Policy Manager*

CME employs a policy mechanism to ease the adaptive management of network resources. Applications can just express their adaptation requirements with policies when they open new channels, and so totally disregard the detailed execution of adaptive mechanisms at the CME. A policy denotes the criteria for the selection of the best current network interface. Then the connection controller maintains each channel according to the corresponding policy. An application can also explicitly control the channel with exposed channel control interfaces.

A policy can be either static or adaptive. A static policy explicitly declares the network interface to be used. An adaptive policy is used to describe the access selection rule for one particular type of traffic flow. An adaptive policy can be represented by (traffic class, logic conditions, weighted factors). Traffic class can be any value defined in TOS [3], TC [4], DS [5] or any application specified value. Logic conditions are a series of comparison expressions connected by logic operators. Weighted factors are a set of 2-tuple (factor, weight). One sub-policy example can be

(THROUGHPUT, (bandwidth>100kbps) and (delay<5s), (cost, 0.2), (power, 0.2), (bandwidth, 0.6))

Note that not all the fields are necessary for any policy. Some policy could be very simplified, e.g. include only one field. Also note that one policy can be shared by many channels.

Policy manager is used by applications to supervise policies including policy creation and elimination. Policies are then accessed by connection controller during the channel operations. The evaluation that which interface is currently the best to a policy can be done both periodically and immediately when special events happen. Some of the application policies may conflict with user preference policies stored previously in the local context server. We assume that user preferences always have the highest priority, which is the normal case.

B. Connection Controller

Connection controller is the core component of the CME for the final realization of the network management adaptation. By acting as a coordinator and executor, it controls the activities of local CME components (policy manager and connection monitor). At the same time it is also the entity for the interaction and cooperation with other related components (e.g. context servers) and peer controllers. The functions of the connection controller fall into two categories, i.e. information maintenance and channel management.

1) Information maintenance: Connection information maintained by connection controller includes both local information and global information. Information maintenance is used to assist the maintenance of connection channel. For local information, connection controller maintains the lists of the references of all the interfaces, channels and policies, together with the mappings between them. The lists and mappings are continuously updated in case of any special event (e.g. a channel has switched the connection under using or a new channel is opened with a new policy).

Connection controller maintains global information of the end host at the global context server. The controller keeps updating the global network context data. This is accomplished by registering a new entry or updating an existing entry in the global context server. The update is performed periodically or when any related event (i.e. changes of numbers and addresses of interfaces) happens.

2) Channel maintenance: The core adaptation mechanism of the CME architecture is realized by connection controller through adaptively maintaining connection channels. There are two activities of the connection controller concerning channel maintainence, i.e. channel opening and switching. To open a new channel application may provide four parameters: target host name, traffic class, channel direction, and policy set. Connection controller then queries the global context server with the target host name to get the address of a preferred network interface of the target host. The best local interface is selected according to the policy. A channel can be either uni-directional or bi-directional.

Connection controller periodically re-evaluates the mappings between an interface and each channel according to the policy used for each channel. Moreover, the re-evaluation is also immediately done when special events (e.g. interface up or down, channel opened or closed) occur. If, according to the policy a better interface is found, then the connection controller initializes a channel switching session. The session needs the cooperation between controller peers through the signaling channel, as shown in Figure 2. One basic rule is that any incoming and outgoing connections can only be decided by the host itself instead of the peer. The decision is made indirectly through global context server for incoming channel, while directly by connection controller according to the policy for outgoing channel.

V. CONCLUSIONS

New middleware is necessary for the optimization of future mobile communications as the platform for intelligent services. The architecture of CME is proposed in this paper, which is aims at providing adaptive network management to network-aware applications. We then showed the mechanisms for the enhancement of both context awareness and adaptation. CME is currently a work in progress. We have implemented a preliminary prototype based on Java on PDA's PocketPC platform. More functionalities will be implemented gradually during the CME's evolvement. Moreover, CME will be further embedded into a broader platform to experience the investigation of real context-aware applications.

ACKNOWLEDGMENT

Financial support by the National Technology Agency of Finland is gratefully acknowledged.

REFERENCES

[1] M. Satyanarayanan, "Pervasive computing: vision and challenges," IEEE Personal Communications, vol. 8, no. 4, pp. 10-17, Aug. 2001.

[2] M. Alam, R. Prasad, and J.R. Farserotu, "Quality of service among IP-based heterogeneous networks," IEEE Personal Communications, vol. 8, no. 6, pp. 18 –24, Dec. 2001

[3] P. Almquist, "Type of service in the Internet Protocol suite," IETF RFC 1349, July 1992.

[4] S. Deering and R. Hinden, "Internet Protocol, version 6 (IPv6) specification," IETF RFC 2460, Dec. 1998

[5] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," IETF RFC 2474, Dec. 1998.