Technical Report MSU-CSE-06-7, February 2006,
Submitted for publication.

Mobile Service Clouds: A Self-Managing Infrastructure for Autonomic Mobile Computing Services ⋆

Farshad A. Samimi[1], Philip K. McKinley[1], and S. Masoud Sadjadi[2]

[1] Department of Computer Science and Engineering, Michigan State University,
East Lansing, MI 48823, USA
{farshad, mckinley}@cse.msu.edu,
[2] School of Computing and Information Sciences, Florida International University,
Miami, FL 33199, USA
sadjadi@cs.fiu.edu

Abstract. We recently introduced Service Clouds, a distributed infrastructure designed to facilitate rapid prototyping and deployment of autonomic communication services. In this paper, we propose a model that extends Service Clouds to the wireless edge of the Internet. This model, called *Mobile Service Clouds*, enables dynamic instantiation, composition, configuration, and reconfiguration of services on an overlay network to support self-management in mobile computing. We have implemented a prototype of this model and applied it to the problem of dynamically instantiating and migrating proxy services for mobile hosts. We conducted a case study involving data streaming across a combination of PlanetLab nodes, local proxies, and wireless hosts. Results are presented demonstrating the effectiveness of the prototype in establishing new proxies and migrating their functionality in response to node failures.

Keywords: autonomic networking, distributed service composition, self-managing system, overlay network, mobile computing, quality of service.

1 Introduction

As the cyber-infrastructure becomes increasingly complex, need for autonomic [1] communication services is also increasing. Autonomic communication services can be used to support fault tolerance, enhance security, and improve quality of service in the presence of network dynamics. An integral part of such systems is the ability to dynamically instantiate and reconfigure services, transparently to end applications, in response to changes in the network environment. A popular approach to supporting transparent reconfiguration of software services is adaptive middleware [2]. However, supporting autonomic communication services often requires adaptation not only at the communication end points, but also at intermediate nodes "within" the network. One approach to this problem is the deployment of a service infrastructure within an *overlay network* [3], in which end hosts form a virtual network atop the physical network. The

⋆ This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, and in part by National Science Foundation grants EIA-0000433, EIA-0130724, and ITR-0313142.

presence of *hosts* along the paths between communication end points enables intermediate processing of data streams, without modifying the underlying routing protocols or router software. This paper investigates the integration of adaptive middleware and overlay networks to support autonomic communication services on behalf of mobile hosts.

We recently introduced *Service Clouds* [4], an overlay-based infrastructure intended to support rapid prototyping and deployment of autonomic communication services. In this approach, the nodes in an overlay network provide a "blank computational canvas" on which services can be instantiated on demand, and later reconfigured in response to changing conditions. When a new service is needed, the infrastructure finds a suitable host, instantiates the service, and maintains it only as long as it is needed. We implemented a prototype of Service Clouds and experimented it atop the PlanetLab Internet testbed [5]. We conducted two case studies in which we used the Service Clouds prototype to develop new autonomic communication services. The first was a TCP-Relay service, in which a node is selected and configured dynamically to serve as a relay for a data stream. Experiments demonstrate that our implementation, which is not optimized, can in many cases produce significantly better performance than using a direct IP connection. The second case study involved MCON, a service for constructing robust connections for multimedia streaming. When a new connection is being established, MCON exploits physical network topology information in order to dynamically find and establish a high-quality secondary path, which is used as shadow connection to the primary path. Details can be found in [4].

In this paper, we propose *Mobile Service Clouds,* an extension of this model that supports autonomic communication at the wireless edge of the Internet, that is, those nodes that are one, at most a few, wireless hops away from the wired infrastructure. Mobile computing environments exhibit operating conditions that differ greatly from their wired counterparts. In particular, applications must tolerate the highly dynamic channel conditions that arise as users move about an environment. Moreover, the computing devices being used by different end users may vary in terms of display characteristics, processor speed, memory size, and battery lifetimes. Given their synchronous and interactive nature, real-time applications such as video conferencing are particularly sensitive to these differences. The Mobile Service Clouds model supports dynamic composition and reconfiguration of a service path at the wireless edge. We have implemented a prototype of this model and applied it to the problem of dynamically instantiating and migrating proxy services for mobile hosts. We conducted a case study involving data streaming across a combination of PlanetLab nodes, local proxies, and wireless hosts. Results are presented demonstrating the effectiveness of the prototype in establishing new proxies and migrating their functionality in response to node failures.

The remainder of this paper is organized as follows. Section 2 provides background on Service Clouds and discusses related work. Section 3 introduces Mobile Service Clouds model. Section 4 describes the architecture and implementation of Mobile Service Clouds. Section 5 presents a case study and experimental results. Finally, Section 6 summarizes the paper and discusses future directions.

2 Background and Related Work

Figure 2 shows a conceptual view of Service Clouds. A service cloud can be viewed as a collection of hosts whose resources are available to enhance communication services (e.g., in terms of fault tolerance, quality of service, or security) transparently with respect to the communication end points. To do so requires autonomic behavior, in which individual nodes in the cloud use adaptive middleware and cross-layer collaboration to support reconfiguration. An overlay network connecting these nodes serves as a vehicle to support cross-platform cooperation. The nodes in the overlay network provide the computing resources with which services can be instantiated as needed, and later reconfigured in response to changing conditions. The Service Clouds infrastructure is designed to be extensible: a suite of low-level services for local and remote interactions can be used to construct higher-level autonomic services.

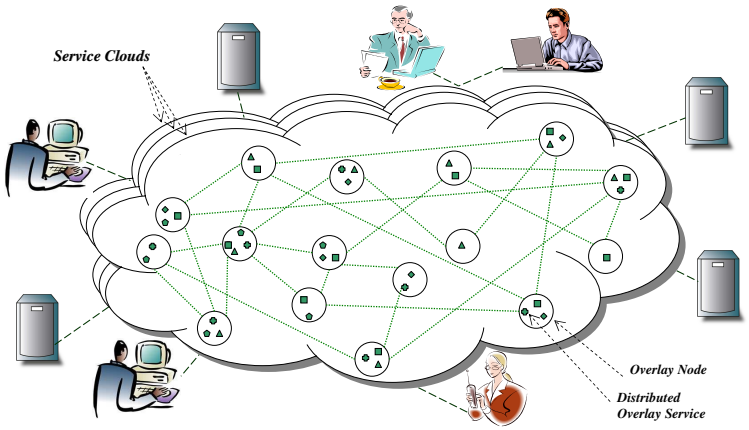


Fig. 1. Conceptual view of the Service Clouds infrastructure.

The Service Clouds infrastructure incorporates results from three areas of research in distributed systems. First, adaptive middleware and programming frameworks [2, 6–8] enable dynamic reconfiguration of software in response to changing conditions. Research in this area has been extensive [2] and has provided a better understanding of several key concepts relevant to autonomic computing, including reflection, separation of concerns, component-based design, and transparent interception of communication streams. Second, cross-layer cooperation mechanisms [9, 10] enable coordinated adaptation of the system as a whole, and in ways not possible within a single layer. The Service Clouds architecture supports cross-layer cooperation and incorporates low-level network status information in the establishment and configuration of high-level communication services. Third, overlay networks [3] provide an adaptable and responsive chassis on which to implement communication services for many distributed applications [11–14].

Recently, several groups of researchers have investigated several ways to use overlay networks to support dynamic composition and configuration of communication and streaming services (e.g., SpiderNet [15], CANS [16], GridKit [17], DSMI [18], and iOverlay [19]). Service Clouds complements to these research works by providing a

"toolkit" with which to develop and test new services that require dynamic instantiation, composition, and reconfiguration. Service Clouds provides an extensive set of components that can be used to compose complex communication services. Moreover, the developer can introduce new or customized components by simply plugging them into the Service Clouds infrastructure.

Our initial work on Service Clouds [4] focused on services in the wired network. At the wireless edge, services are often deployed on proxy nodes, which operate on behalf of mobile hosts. Extensive research has been conducted in the design of proxy services to support data transcoding, handle frequent disconnections, and enhance the quality of wireless connections through techniques such as forward error correction (FEC) [20–22]. Rather than addressing operation of specific proxy services, in this work we concentrate on the dynamic instantiation and reconfiguration of proxy services in response to changing conditions.

3 Extending Service Clouds to the Wireless Edge

Figure 2 depicts an extension of the Service Clouds concept to support mobile computing. In this model, mobile service clouds comprise collections of overlay hosts that implement services close to the wireless edge, while deep service clouds perform services using an Internet overlay network (such as the PlanetLab wired hosts used in our earlier study). In a manner reminiscent of Domain Name Service, this *federation* of clouds cooperate to meet the needs of client applications. Typically, a mobile service cloud will comprise nodes on single intranet, for example, a collection of nodes on a university campus or used by an Internet Service Provider (ISP) to enhance quality of service at wireless hotspots. A mobile user may interact with different mobile service clouds as he/she moves about the wireless edge, with services instantiated and reconfigured dynamically to meet changing needs.

Figure 2 shows an example in which a mobile user is receiving a live or interactive video stream on his mobile device. Service elements are instantiated at different locations along an overlay path according to application requirements and current conditions. For example, if the user is connected via a relatively low bandwidth wireless link, a video transcoder may be established close to the video source, to reduce the bit rate on the video stream and avoid wasted bandwidth consumption along the wired segment of the path. On the other hand, a proxy service that uses FEC and limited retransmissions to mitigate wireless packet losses, may be established on a mobile service cloud node at the wireless edge. The operation of proxy service depends on the type of data stream to be transmitted across the wireless channel. Over the past few years, our group has investigated proxy services for reliable multicasting [23], audio streaming [24, 25] and video streaming [24, 26, 27].

As the user moves within an area serviced by a single mobile service cloud, or among different service clouds, the proxy can be migrated so as to move along with the user. We refer to such an instantiation as a *transient* proxy [27]. There are several reasons to keep the proxy close (in terms of network latency) to the mobile user. First is the quality of service of the data stream delivery. For example, EPR [26] is a forward error correction method for video streaming in which the proxy encodes video

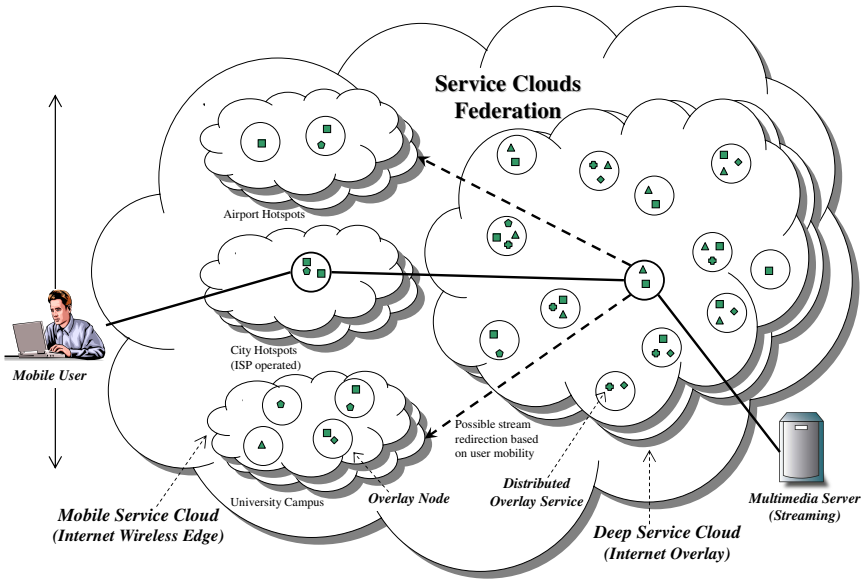


Fig. 2. Example scenario involving Mobile Service Clouds.

frames using a block-erasure code, producing a set of parity packets. The proxy sends a subset of the parity packets "pro-actively" with the stream. Additional parity packets are sent in response to feedback from the mobile device, to handle temporary spikes in loss rate. However, the effectiveness of these additional parity packets depends on the round-trip delay between the mobile device and the proxy: if the delay is too long, the parity packets arrive too late for real-time video playback. The Second reason for the proxy to be close to the mobile host is resource consumption. For example, a proxy that implements forward error correction increases the bandwidth consumption of the data stream. When the mobile device connects to a different Internet access provider, if the proxy is not relocated, then the additional traffic may traverse several network links across Internet service providers. While the effect of a single data stream may be small, the combined traffic pattern generated by a large number of mobile users may have noticeable effect on performance. Third is the policy of the service provider. While an ISP may be willing to use its computational resources to meet the needs of users connected through its own access points, this may not apply to mobile hosts using access points belonging to another provider. In the same way that the mobile device changes its access point but remains connected, the proxy services on the connection may need to move in order for the new connection to be comparable to the old one.

Mobile Service Clouds provides an infrastructure to support the deployment and migration of proxy services for mobile clients. In the experiments described in Section 5, we address another reason to migrate a proxy service, namely, fault tolerance. If a proxy suddenly crashes or becomes disconnected, another node in the service cloud should assume its duties with minimal disruption to the communication stream(s). Next, we describe the architecture and implementation of our proof-of-concept prototype.

4 Architecture and Implementation

The Service Clouds infrastructure is intended primarily to facilitate rapid prototyping and deployment of autonomic communication services. Overlay nodes provide processing and communication resources on which transient services can be created as needed to assist distributed applications. Figure 3 shows a high-level view of the Service Clouds software organization and its relationship to Schmidt's model of middleware layers [28]. Most of the Service Clouds infrastructure can be considered as *host-infrastructure* middleware, it provides a layer of abstraction on top of heterogeneous platforms. The *Application-Middleware eXchange (AMX)* provides a high-level interface for that purpose, and encapsulates the required logic to drive various overlay services. The *Kernel Middleware eXchange (KMX)* layer provides services through its interface to facilitate collaboration between middleware and the operating system. Distributed composite services are created by plugging in new algorithms and integrating them with lower-level control and data services, which in turn depend on lower-level overlay services.

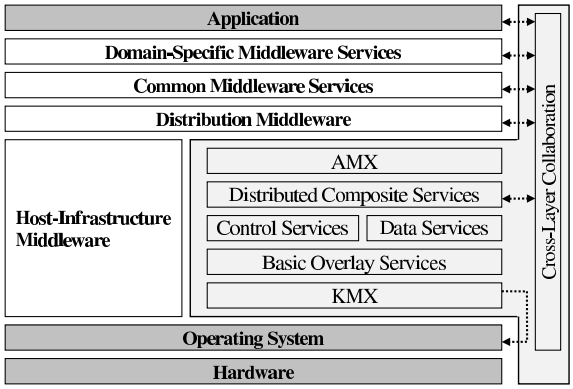


Fig. 3. Relationship of Service Clouds to other system layers [4].

Figure 4 provides a more detailed view of the Service Clouds architecture, showing those components introduced or used in this study (unshaded boxes), as well as those used in our TCP-Relay and MCON studies (shaded boxes). The architecture comprises four main groups of services, situated between the AMX and KMX layers. At the lowest layer, *Basic Overlay Services* provide generic facilities for establishing an overlay topology, exchanging status information and distributing control packets among overlay hosts. *Control Services* include both *event processors* and *DCS-specific services*. An event processor handles specific control events and messages. For example, such a service might receive certain types of inquiry packets and extract information useful to multiple higher-level services. *Data Services* are used to process data streams as they traverse a node; they include *monitors*, which carry out measurements on data streams, and *actuators*, which modify data streams. At the highest layer, *Distributed Composite Services* include *overlay engines*, which codify complex distributed algorithms, and *Co-*

ordination and Service Composition, which provides "glue" between overlay engines and lower-level services.

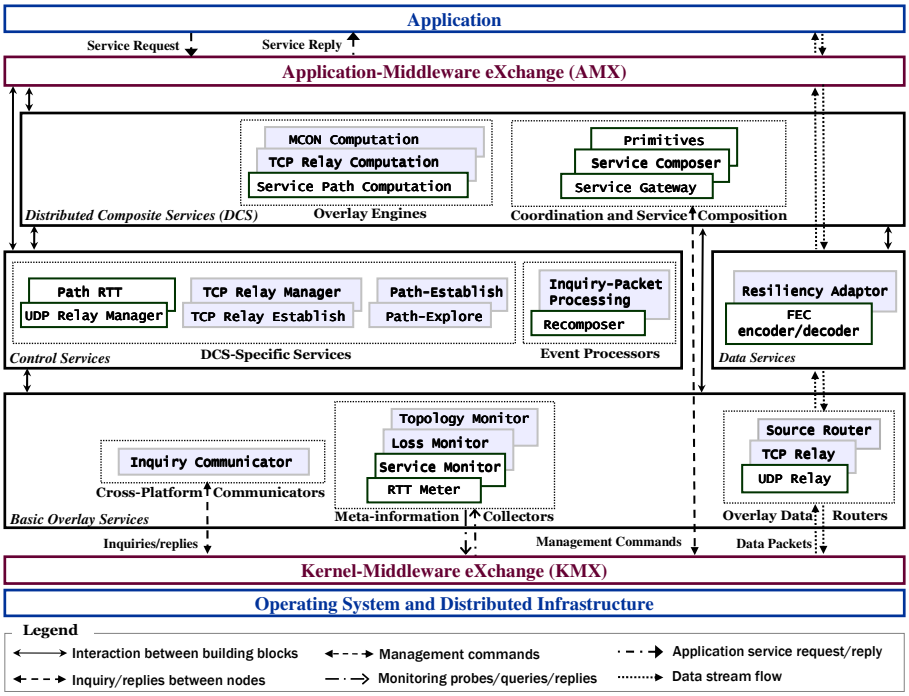


Fig. 4. Instantiation of the general Service Clouds model.

For the prototype implementation of Mobile Service Clouds, we introduced new components but also reused several others. First, the *Service Path Computation* overlay engine manages the interaction between a mobile host and a service cloud federation. Tasks include selection of a node in a deep service cloud, called the *primary proxy*, which coordinates composition and maintenance of the service path between two end nodes. The Service Path Computation also finds a suitable node in a mobile service cloud on which to deploy the transient proxy services (FEC in our study). We also required lower-level services to use in identifying potential proxies. The *RTT Meter* component uses "ping" to measure round-trip time (RTT) to an arbitrary node, and the *Path RTT* component measures end-to-end RTT between two nodes, whose communication is required to pass through an intermediate node.

The *Service Gateway* component implements a simple protocol to accept and reply to service requests. Upon receiving a request, it invokes the overlay engine to find a suitable primary proxy. The *Service Composer* component implements mechanisms for composing a service path. It uses the *Relay Manager* to instantiate and configure a UDP relay on the primary proxy and the transient proxy. The UDP relay on the transient proxy enables the infrastructure to intercept the stream and augment it with FEC encoding. Accordingly, as soon as the FEC proxy service is instantiated, the *Service Monitor* component on the transient proxy begins sending heartbeat beacons through a

TCP channel toward the service monitor on the primary proxy. The *Recomposer* component on the primary proxy tracks activity of the service monitors. Upon detecting a failure, it starts a self-healing operation that recomposes the service path and restores communication.

The prototype also has a main program that deploys the infrastructure primitives. It reads configuration files containing the IP addresses of overlay nodes and the overlay topology, instantiates a basic set of (composite) components, and configures the components according to the default or specified parameter values. Examples of these parameters include the interval between probes for monitoring purposes, service port numbers, and an assortment of debugging options. The prototype software package also includes a collection of scripts to configure nodes, update code at nodes, launch the infrastructure, run test scenarios, and collect results. To deploy management commands that control test execution on several nodes, we have used Java Message Service (JMS). Example management commands include those for gathering operation statistics on each node, changing framework parameters at run time, and shutting down the distributed infrastructure.

We emphasize that the purpose of this prototype is merely to identify different aspects of the problem and conduct a requirements analysis. Therefore, we implemented only a small set of features. Building a number of such prototype systems will help reveal the salient issues for designing a more complete Service Clouds framework.

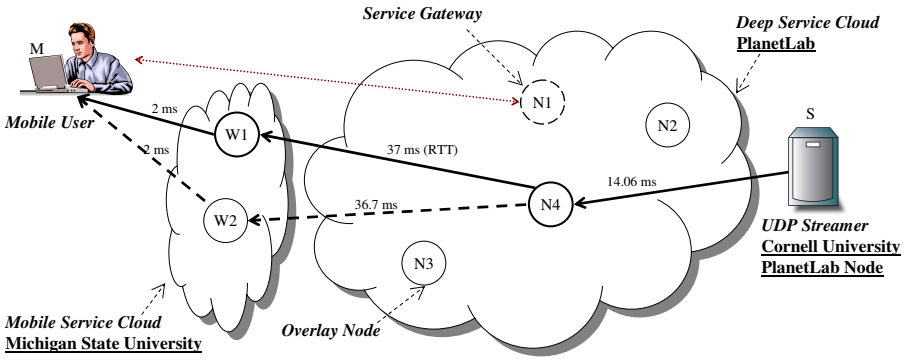
5 Case Study

We conducted a case study in which we assessed the ability of MSC (Mobile Service Clouds) to establish transient proxies for mobile devices, monitor the service path, and support dynamic reconfiguration (with minimal interruption) when the proxy node fails.

Basic Operation. In this scenario a mobile node on a wireless link wants to receive a multimedia stream (e.g., in an interactive video conference or in a live video broadcast). In this case, the MSC infrastructure needs to fulfill the following requirements. First, the quality of the received stream must remain acceptable as the wireless link experiences packet loss. Second, the video stream must be transcoded to satisfy resource restrictions such as wireless bandwidth and processing power at the mobile device. Third, stream delivery should not be interrupted as conditions on the service path change (e.g., when user movement causes a wireless network domain change, or when a proxy or other service node fails).

Figure 5 shows the configuration used in the experiments, with 4 PlanetLab nodes in a deep service cloud and two workstations on our department intranet in a mobile service cloud. These systems are all Unix/Linux-based machines. We have used a PlanetLab node to run a UDP streaming program and a Windows XP laptop to receive the stream over a wireless link. The middleware software on the mobile client connects to a Service Gateway node (N1) and requests the desired service. Gateway nodes are the entry point to the Service Clouds: they accept requests for connection to the Service Clouds and designate a service coordinator based on the requested service. In this work, we assume that gateway nodes are known in advance, as with local DNS servers. In future implementations, we plan to integrate other methods, such as directory services,

into the infrastructure to enable automatic discovery of gateway nodes. Upon receiving the request, the gateway begins a process to find a node to act as the primary proxy (N4), and when completed, informs the mobile client of the selection. The primary proxy receives details of the desired service, sets up a service path, and coordinates monitoring and automatic reconfiguration of the service path during the communication.



M	senslap10.cse.msu.edu	Mobile User
S	planetlab1.cs.cornell.edu	UDP Streamer
N1	planet-lab-1.csse.monash.edu.au	Service Gateway
N2	planetlab01.cs.washington.edu	Candidate Primary Proxy
N3	planetlab1.cs.ucsb.edu	Candidate Primary Proxy
N4	planetlab1.csail.mit.edu	Candidate Primary Proxy (chosen at run time)
W1	countbasey.cse.msu.edu	Wireless edge relay
W2	arctic.cse.msu.edu	Wireless edge backup relay

Fig. 5. The experimental testbed and example scenario.

A gateway might consider several factors in deciding on a primary proxy for a requested service: security policies of the client and service cloud components, round-trip time between the the node and the communication endpoints, and computational load at the node. In this example, N4 is chosen such that the path round-trip time between the two endpoints is minimal.

Next, the MSC infrastructure instantiates FEC functionality at the transient proxy (W1) in a mobile service cloud, and dynamically re-instantiates the service on another node when the W1 fails. "Failure" can be defined in different ways: high computational load; high RTT to the client due to change in access point used by client; software failure of the service, or hardware failure. To study and test a basic self-repair in Service Clouds, we simply inject a failure by terminating the service process on W1.

In the example depicted in Figure 5, the Service Clouds client middleware, residing on the laptop, sends a service request to the gateway node N1, which chooses N4 as the primary proxy and informs the client. Next, the client software sends a primary proxy service request to N4, which constructs a service path comprising a UDP relay on itself and a UDP relay augmented with an FEC encoder on W1. As soon as W1 starts the service, it begins sending heartbeat beacons over a TCP connection to N4 that indicate

the service node is active. N4 runs a monitoring thread that listens for beacons from W1 (sent every 5msec in our experiment). If it detects a failure, it reconfigures the overlay service path to use another node in the mobile service cloud.

Experimental Results. To test dynamic reconfiguration of a service path, the program running on W1 is terminated. We have evaluated two different strategies to realize self-healing within the framework at the time of failure detection: *(1) on demand backup* where another node, W2, is configured dynamically as soon as failure is detected; *(2) ready backup* where W2 is configured as a backup at the same time of W1 configuration, so the system only needs to configure the relay on N4 to forward the stream to W2 instead of W1.

We have measured percentage of packets received at the wireless node. Figure 6 plots the average of 12 runs for 50 millisecond epochs, indicating the situation when W1 fails and system recovers automatically. As the plot shows, the system can completely recover from the failure at W1 in less than 0.4 seconds. The "on demand backup" is slightly slower, since system has to instantiate and configure the proxy service. In the ready backup case, the service is instantiated at the time of service composition, yielding faster response.

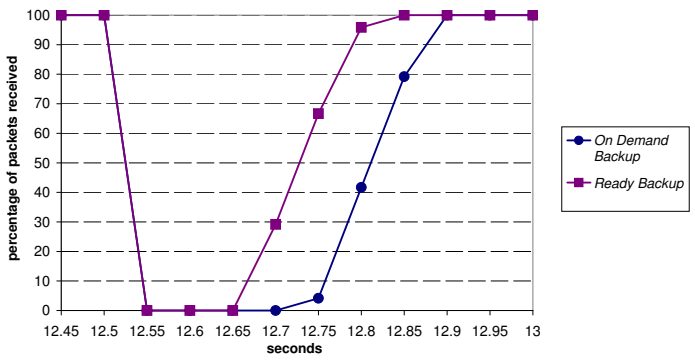


Fig. 6. Packet loss during node failure.

In future work, we will consider additional failure detection and recovery strategies. For example, adding a limited capability to the client middleware to participate in failure detection may enable the client to trigger the reconfiguration faster, since the mobile node is closer (in terms of round-trip time) to the wireless edge.

6 Conclusions and Future Work

In this paper, we addressed the issue of dynamic services at the wireless edge of the Internet. We introduced *Mobile Service Clouds*, an infrastructure for rapid prototyping and deployment of such services. We described a proof-of-concept implementation of Mobile Service Clouds which we used to support proxy instantiation and fault tolerance on a testbed comprising PlanetLab nodes and hosts on a university intranet. Preliminary

results demonstrate the usefulness of the model and the effectiveness of the prototype. Our ongoing investigations address dynamic insertion and reconfiguration of transcoding filters along a stream path, dynamic migration of proxy services to enhance quality-of-service, and integration of Mobile Service Clouds and data streaming protocols for sensor networks.

Further Information. Related publications on the RAPIDware project, as well as a download of the Service Clouds prototype, can be found at the following website: `http://www.cse.msu.edu/rapidware`.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer **36**(1) (2003) 41–50
2. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing adaptive software. IEEE Computer (2004) 56–64
3. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient Overlay Networks. In: Proceedings of 18th ACM Symposium on Operating Systems Principles (SOSP 2001). (2001)
4. McKinley, P.K., Samimi, F.A., Shapiro, J.K., Tang, C.: Service Clouds: A distributed infrastructure for composing autonomic communication services. Technical Report MSU-CSE-05-31 (Available at http://www.cse.msu.edu/rapidware/serviceclouds.pdf), Department of Computer Science, Michigan State University, East Lansing, Michigan (2005)
5. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A Blueprint for Introducing Disruptive Technology into the Internet. In: Proceedings of HotNets–I, Princeton, New Jersey (2002)
6. Zinky, J.A., Bakken, D.E., Schantz, R.E.: Architectural support for quality of service for CORBA objects. Theory and Practice of Object Systems **3**(1) (1997) 1–20
7. Redmond, B., Cahill, V.: Supporting unanticipated dynamic adaptation of application behaviour. In: Proceedings of the 16th European Conference on Object-Oriented Programming, Malaga, Spain, Springer-Verlag (2002) volume 2374 of Lecture Notes in Computer Science.
8. Liu, H., Parashar, M., Hariri, S.: A component-based programming model for autonomic applications. In: Proceedings of the 1st International Conference on Autonomic Computing, New York, NY, USA, IEEE Computer Society (2004) 10–17
9. Noble, B.D., Satyanarayanan, M., Narayanan, D., Tilton, J.E., Flinn, J., Walker, K.R.: Agile application-aware adaptation for mobility. In: Proceedings of the Sixteen ACM Symposium on Operating Systems Principles. (1997) 276–287
10. Kong, J., Schwan, K.: KStreams: kernel support for efficient data streaming in proxy servers. In: Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), ACM (2005) 159–164
11. Gribble, S.D., Welsh, M., von Behren, J.R., Brewer, E.A., Culler, D.E., Borisov, N., Czerwinski, S.E., Gummadi, R., Hill, J.R., Joseph, A.D., Katz, R.H., Mao, Z.M., Ross, S., Zhao, B.Y.: The Ninja architecture for robust Internet-scale systems and services. Computer Networks **35**(4) (2001) 473–497
12. Byers, J.W., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. IEEE/ACM Transactions on Networking (TON) **12**(5) (2004) 767–780
13. Li, B., Xu, D., Nahrstedt, K.: An integrated runtime QoS-aware middleware framework for distributed multimedia applications. Multimedia Systems **8**(5) (2002) 420–430

14. Rodriguez, A., Killian, C., Bhat, S., Kostic, D., Vahdat, A.: Macedon: Methodology for automatically creating, evaluating, and designing overlay networks. In: Proceedings of the USENIX/ACM First Symposium on Networked Systems Design and Implementation (NSDI 2004), San Francisco, California (2004) 267–280
15. Gu, X., Nahrstedt, K., Yu, B.: SpiderNet: An integrated peer-to-peer service composition framework. In: Proceedings of IEEE International Symposium on High-Performance Distributed Computing (HPDC-13), Honolulu, Hawaii (2004) 110–119
16. Fu, X., Shi, W., Akkerman, A., Karamcheti, V.: CANS: composable and adaptive network services infrastructure. In: The 3rd USENIX Symposium on Internet Technology and Systems, San Franscisco, California (2001)
17. Grace, P., Coulson, G., Blair, G., Mathy, L., Duce, D., Cooper, C., Yeung, W.K., Cai, W.: Gridkit: Pluggable overlay networks for grid computing. In: Proceedings of International Symposium on Distributed Objects and Applications(DOA), Larnaca, Cyprus (2004) 1463–1481
18. Kumar, V., Cooper, B.F., Cai, Z., Eisenhauer, G., Schwan, K.: Resource-aware distributed stream management using dynamic overlays. In: Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS 2005), Columbus, OH, USA, IEEE Computer Society (2005) 783–792
19. Li, B., Guo, J., Wang, M.: iOverlay: A lightweight middleware infrastructure for overlay application implementations. In: Proceedings of the Fifth ACM/IFIP/USENIX International Middleware Conference, also Lecture Notes in Computer Science. Volume 3231., Toronto, Canada (2004) 135–154
20. Fox, A., Gribble, S.D., Chawathe, Y., Brewer, E.A.: Adapting to network and client variation using active proxies: Lessons and perspectives. IEEE Personal Communications (1998)
21. Zenel, B.: A general purpose proxy filtering mechanism applied to the mobile environment. Wireless Networks **5** (1999) 391–409
22. Roussopoulos, M., Maniatis, P., Swierk, E., Lai, K., Appenzeller, G., Baker, M.: Person-level routing in the mobile people architecture. In: Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems, Boulder, Colorado (1999)
23. McKinley, P.K., Tang, C., Mani, A.P.: A study of adaptive forward error correction for for wireless collaborative computing. IEEE Transactions on Parallel and Distributed Systems (2002)
24. McKinley, P.K., Padmanabhan, U.I., Ancha, N., Sadjadi, S.M.: Composable proxy services to support collaboration on the mobile internet. IEEE Transactions on Computers (Special Issue on Wireless Internet) (2003) 713–726
25. Zhou, Z., McKinley, P.K., Sadjadi, S.M.: On quality-of-service and energy consumption tradeoffs in fec-enabled audio streaming. In: Proceedings of the 12th IEEE International Workshop on Quality of Service (IWQoS 2004), Montreal, Canada (2004)
26. Ge, P.: Interactive Video Multicast in Wireless LANs. PhD thesis, Michigan State University, Department of Computer Science and Engineering (2004)
27. Samimi, F.A., McKinley, P.K., Sadjadi, S.M., Ge, P.: Kernel-middleware interaction to support adaptation in pervasive computing environments. In: Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing, Toronto, Ontario, Canada, ACM Press (2004) 140–145
28. Schmidt, D.C.: Middleware for real-time and embedded systems. Communications of the ACM **45**(6) (2002) 43–48