Enabling rich content service on the edge: opportunities and challenges

Zheng Zhang
Xing Xie
Boying Lu
Shiding Lin

Jul. 1, 2002

Technical Report
MSR-TR-2002-71

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Enabling rich content service on the edge: opportunities and challenges

Zheng Zhang, Xing Xie, Boying Lu, Shiding Lin
Microsoft Research Asia
3F, Sigma Center, No. 49 Zhichun Road
Beijing 100080, P.R.China
86-10-6261-7711
{zzhang, i-xingx, i-bylu, i-slin}@microsoft.com

ABSTRACT

In this position paper, we examine the opportunities and challenges of enabling rich content service on the edge resources (CDN and smart proxies). We discuss what edge resources are engaged and the research problems involved in planning and mapping instantiated service graph on the resources. Finally, we describe the Fadel experimental system, our research vehicle to tackle the research challenges as outlined in this paper.

Keywords

Content delivery network, edge computing, adaptive content delivery, peer-to-peer overlay, distributed multimedia system

1. INTRODUCTION

Edge computing has generated lots of excitement lately, especially in the Web service field [26]. The multimedia research community, however, has had a long tradition exploring the power of distributed proxies for rich content service and processing inside the network [9][10][15][25][27].

In this paper, we examine the issue of enabling rich content service on the edge resources (CDN and smart proxies) in detail. We first make a succinct description of the service graph and, when it is instantiated, the corresponding logical graph (Section 2). We discuss the various types of edge resources in Section 3 and the problem of mapping logical graph onto edge resources in Section 4. Following that, we list the opportunities and research challenges in Section 5. The Fadel system, our experimental system to tackle these research questions is described in Section 6. Related works are covered in Section 7 and we conclude in Section 8.

2. THE RICH CONTENT SERVICE FRAMEWORK

In defining the rich content service framework, we will concern ourselves with the *logical* entities involved only. Borrowing the terminology from [10], the resulting *logical graph* is a directed graph with nodes being either the participants (producing and/or consuming content) or those processing the contents (e.g. adaptation), and edges the flow of contents.

There are only a few basic types of nodes: single-input single-output (SISO) nodes like transcoders, multiple-input single-output (MISO) nodes like video-audio mixers and single-input multiple output (SIMO) nodes like splitters in multicasting. The most general case will be multiple input multiple output (MIMO) nodes.

If we simplify the graph by including only those nodes corresponding to the participants, the resulting graph becomes what we call the *service graph*. Service graph is what the service provider offers to the end user; and logical graph is what the service developers built. Logical graph is an instantiated service graph.

In some cases, the end service graph may itself be composed by more primitive ones. For instance, a video walkie-talkie application can be considered as time-switching between multiple service graphs, each corresponding to a multicast service graph.

3. RESOURCES ON THE EDGE

There is much hype surrounding the term “edge computing” lately, especially in the Web service scene. In our opinion, the resources on the edge are really composed of two kinds. The first corresponds to the CDN installations: they are situated at strategically advantageous location in the network, and can perform simple value-added services. Vastly outnumbering the first type of edge resources are those smart proxies (the reverse proxies associated on the server side are known as *surrogates*), they are more flexible and can couple with themselves dedicated computing resources to run more complex jobs, but they certainly can not compete with the first type location-wise. These are only some of the distinctive features that set them apart. To distinguish the two, we call the former the *inner edge*, and the later the *outer edge*.

This demarcation is important, not the very least reminding us simply that one can not treat all the edge resource the same: we must be careful about what responsibilities be placed on which part of the edge resources in deploying a complex service graph.

4. MAP RICH CONTENT SERVICES ON THE EDGE

When a user subscribes to the service, a logical graph is formed (or an existing one is expanded) to accommodate her receiving context (device, bandwidth, language etc). Mostly this is through *content adaptation*. For a succinct treatment of the topic, we refer readers to [13]. Here, the constraint of the down link or the end receiver’s capacity dictates that contents be distilled with various ways (rate adjustment, frame dropping etc.), or simply because the receiver is expecting a different delivery format. Some of the procedures in this step are simple, such as distillation, some can be less trivial – for instance when the security nature of the delivery path changes [27].

The logical graph is to be mapped on to the edge resources. Following the notion of [10], we call the end graph the *physical graph*. Strictly speaking, this is not a mapping *per se*. As we will demonstrate next through a series of examples, the graph actually *changes*.

Why does the physical graph differ from the logical graph? That is due to the inevitably resource constraints associated with the graph edges in the logical graph. To accommodate those constraints, these logical edges need to be split while adding new nodes. The following is a possibly incomplete list:

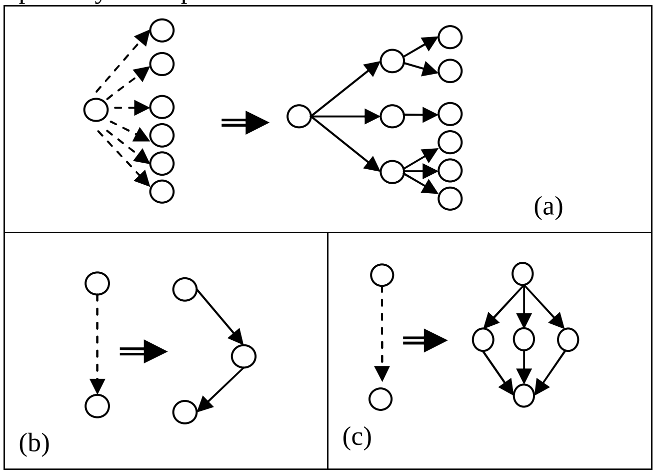

Figure 1: multicasting, detour and multi-sourcing

Multicasting. For a multicast session with lots of destinations, the scalability becomes a concern. The unavailability of IP-level multicasting across the wide area leads people to explore application multicasting tree as a way to get around [5][6][12]. This is a classical example of how logical graph is transformed (for a P2P approach please see [11]), as depicted in Figure 1(a).

De-touring. The direct route between the two nodes connected by an edge may not be the fastest or most reliable channel. There are multiple reasons to that [21]. Routing around multiple overlay hops has shown great promises, as demonstrated by the pioneer work in [2]. This situation is shown in Figure 1(b).

Multi-sourcing. Given two nodes in the graph, it is likely that the physical channel connecting the two can not carry the required load. In this case, it is possible to split the delivery path to multiple ones and then join them at the end, as illustrated in Figure 1(c). Multi-sourcing can be an effective way of "ganging" multiple delivery channels for the benefit of one single edge in the logical graph [3]. This is also useful when the pricing of the delivery channel is such that there is jump on charges beyond certain bandwidth consumption rates.

The above are examples mostly on adaptation over resource constraints. More advanced ones can be derived if the adaptation is made *content-aware*. Suppose a segment of a video is of low interest to a receiver, but is followed by one that's highly relevant. The first segment can be delivered using low-bandwidth methods such as key frames, while simultaneously the next segment is prefeteched to a nearby (new) caching node, and when the first segment is finished, the next one is drawn from the caching node instead. This is a special case of prefix-caching [22].

We engage ourselves in this somewhat lengthy discussion in order to point out that, in most of the cases, it's not sufficient to simply map a logical graph straight on to the physical resources, as are done in [9][10][15][25][27], rather the graph changes and adapts, and this happens in the initial deployment as well as subsequent periods when requests and resources vary.

5. OPPURTUNITIES AND CHALLENGES

Adding values or processing along the path where the content is being delivered has been a known thrust for many years. The debate was rather on which level this is to be played out. Active network [24] is one extreme form of such proposals at the very low (IP) level, while recently there are lots of proponents in the overlay, P2P community with the opinion that, even for the job of networking applications, it's better to do it in a virtual overlay. While these debates are still waging (or some consider this settled), the industry is going ahead by the initiatives that 1) makes the edges smart (i.e. OPES [16]) and 2) enables CDN for services other than caching and delivery [8]. The multimedia research community has already had a long and rich tradition of exploring the power of proxies. Thus, we believe enabling rich content service utilizing the power of edge computing is the right course to take.

But the challenges are many. Among those, we consider the component technologies for adaptation somewhat more mature than others. Here, interesting technologies to develop are those enabling content-aware analysis and adaptation in real time [4].

Instantiate each logical graph and deploy them on the resources has been the focus of many studies. Many existing works focus on automating the logical graph construction. This is relatively easy for simple scenarios like content adaptation but is unclear if the same can be done for more complex ones, for instance when content-aware adaptation is involved. A more pragmatic approach will be a generic platform offering automation where possible, but leave enough flexibility for 1) service developers to *program* compositions and 2) users to assert control if necessary.

On actual service deployment, many of the existing works map the logical graph straight to physical; as we have argued in the last section, this isn't enough. How to split a logical edge, with detour or multi-sourcing, and optionally with a content adaptor in between, remains as one puzzle to be solved. Adding the many types of edge resources makes the problem even more complex. This is only part of a bigger problem: if the physical graph gets big, planning it and continuously adapting it at run time, can no longer take the approach of centralized method as in [9][10][15][25][27]. On the other hand, how optimal a local, greedy algorithm performs without a global knowledge is an interesting challenge as well. Many activities in the P2P research communities have thus far focused on the aspect of self-organizing and efficient

routing [18][20][23][28], and self-adaptation in a heterogeneity environment is just starting [17], and there will be interesting synergies to be explored. Lastly, if graph changes, there must be adequate mechanism to efficiently adjust the topology and manage the handoffs.
Last but not the least, digital right enforcement and security issues need to be researched [27], this is because when a service spans globally, edge resources are likely to be contributed from different administration domain.

6. THE FADEL SYSTEM

In this section we describe Fadel (*Flexible Adaptive DELivery*), an experimental system being developed in Microsoft Research Asia to tackle the challenges as listed in Section 5.

6.1 Organizing the edge resources

All the edge servers collectively form an overlay which is itself organized in a hierarchical fashion. When a participant's (either source or destination) request comes, it connects to its nearest edge server *e*. If *e* is not a member of the overlay yet, it will join at this point. Edge servers within a geographic region are grouped into a *regional control overlay*. Clustering of the servers are using heuristics and the primary metrics is physical vicinity. A number of such heuristic algorithms have been proposed lately, using measurements to landmarks as yardsticks [19]. By some leader-election algorithm one (or several) leader is selected from a regional overlay to participate in a top-layer *global overlay*. Both overlays are self-organizing P2P networks using any of the popular proposals [18][20][23][28]. From an external point of view, these P2P networks are *distributed hash table* (DHT) that guarantees the retrieval of an item if it exists. Statistics about various resources within a region are collected periodically and stored into a set of files in the regional overlay. Regional statistics are further aggregated and reported to the global overlay. Since all the statistics are refreshed periodically (i.e. soft-state), system meta-data is responsive to dynamisms and resilient to failures.

6.2 Graph planning and deployment

There is a logical graph associated with a service session. This logical graph can be automatically formed, as proposed by [9][10][25][27]. On the other hand, more complex services that can not be automatically composed should be *programmed*. Figure 2 shows an instance of video conferencing of three clients on different devices, and X is speaking. The process of face detection (shown framed in the diagram) is to help client Z who is conferencing out of a PDA by showing X's face only. Obviously, there can be situations when this extra adaptation is counterproductive. When such services are involved, the composition needs to consult user's input and be programmed at the time of deployment, and it is neither feasible nor practical to have such plan automated beforehand.

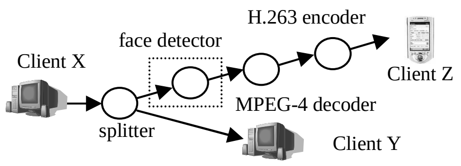

Figure 2: an instance of video conferencing (X to Y and Z)

The overlays can host multiple sessions, each has an associated logical graph. These logical graphs are stored in the global overlay as ordinary files. By default, the node has a graph in its custody is responsible for the root control of responding to new service requests to the session corresponding to the graph. When a new request arrives, the logical graph is accessed and a crude level of planning is carried out first, taking into account the nature of the request and the current system status. This planning partitions the processing into regions and then delegates to each leader its responsible part of the logical graph. Regional leaders perform the initial mapping to edge servers. The edges of this logical graph is examined next, to further expand it if necessary using detour, multi-sourcing and splitting (for multicast) as primitives. This is depicted in Figure 3.

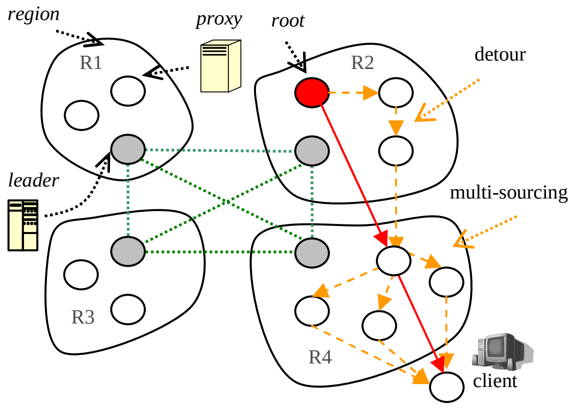

Figure 3: graph planning and deployment. Solid and dashed edges correspond to logical and physical graphs, respectively

While the above discussion deals with the bootstrapping of planning and mapping a new session, another set of algorithms are necessary to respond to changes in the resources. We are currently developing those algorithms.

6.3 Runtime environment

Our runtime environment leverages Microsoft DirectShow as the core component technology and expands upon it. Microsoft DirectShow [14] is architected for streaming media processing. Its basic building block is a software component called filter, which generally performs a single operation on a multimedia stream. For example, there are typical filters that read files, get video from a capture device, decode a particular stream format or pass data to the graphics or sound card.
The way Fadel expands DirectShow is to glue a few interfaces and components so that edge server with the DirectShow core inside can now be networked and becomes the fabric of the whole system. As shown in

Figure 4, the new interfaces we add are: *pitcher*, *catcher*, *filter manager*, *event manager* and *scripting interface*. *Pitcher* reads media data from files or the Internet and passes them to the next filter. The processed stream is sent out by *catcher* using standard protocols. *Scripting interface* provides a mechanism for the global overlay to configure the behavior of proxies. *Filter manager* is responsible for building the initial filter graph, instantiating filters, controlling data flow through the graph, as well as updating the filter graph. *Event manager* catches predefined events, processes them locally or propagates them to the global overlay or other proxies according to the script.

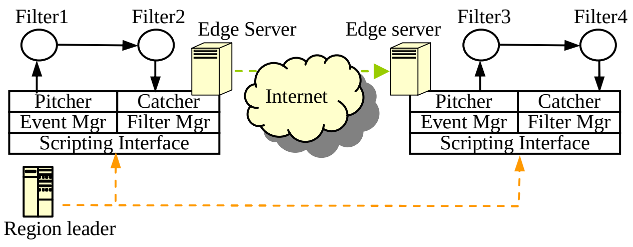

Figure 4: runtime environment

7. RELATED WORK

We draw our inspiration from many outstanding works in diverse research fields. Due to space limitation, we an only list a small portion of them in the references section. The subject of uniting the power of multiple proxy servers for streaming and on-the-fly adaptations have been treated extensively since mid 90's [1][9][10][15][25][27]. The possibility of more demanding services such as application-level multicasting and video conferencing have been proposed in [5][6][7][12]. We take the view that many if not all service instances and their adaptations, including more advanced ones such as real-time content adaptation, can be adopted under one uniform framework.

Instantiating a service graph into a logical graph and then mapping it onto a networked overlay with further runtime adaptation, is the general principle this works shares with existing systems. In fact, it can be argued that there are no other systematic ways of doing the job. We differ, however, in a few critical points in our proposals. We believe a set of resources should be self-organized into a superset of resources for a fault--tolerant and scalable infrastructure, recent progresses in the P2P research community [18][20][23][28] have shown great promises on this regard. On logical graph generation, we are conservative in the sense that automation is done wherever possible, but flexibility is left to consult user's input so that services can be composed and programmed at adequate time of deployment. Finally, as we have pointed out, mapping a logical graph straight onto a physical graph is not sufficient, and there are ready technologies [2][3][21] we plan to use to have a richer arsenal to deal with adaptation more efficiently.

8. CONCLUSIONS AND STATUS

We believe there are significant incentives to harness the power of all the edge resources, including the proxies, surrogates and CDN stations, so that a scalable and fault-tolerant infrastructure can be established to enable rich content services delivery. One of the contributions this paper makes is to enlist the opportunities as well as challenges in this direction.

It should be feasible to unite many interesting rich content services under one uniform framework, and the resulting logical graph should be generated automatically where one can, but would otherwise be possible to be programmed. We believe resources involved should be self-organizing, composing a peer-to-peer overlay in order to lower maintenance cost and be scalable.

Currently we are developing the Fadel system, using Microsoft's DirectShow as the building block. We plan to experiment with a few simple application scenarios such as video walkie-talkie over wide-area with diverse device capabilities to start out.

9. ACKNOWLEDGEMENT

We thank Wei-Ying Ma, Chun Yuan, Yu Chen, and Hong-Jiang Zhang for many insightful comments and discussions.

10. REFERENCES

[1] E. Amir. An Agent-based Approach to Real-Time Multimedia Transmission over Heterogeneous Environments, PhD thesis, UC Berkeley, May 1998

[2] D. Andersen, H. Balakrishnan, et al. Resilient Overlay Networks. The 18th ACM Symp. on Operating Systems Principles (SOSP), Banff, Canada, Oct. 2001

[3] J. G. Apostolopoulos, T. Wong, et al. On Multiple Description Streaming with Content Delivery Networks. IEEE INFOCOMM 2002, Jul. 2002

[4] S.-F. Chang, D. Zhong, et al. Real-Time Content-Based Adaptive Streaming of Sports Video. IEEE Intl. Workshop on Content-Based Access to Video and Image Libraries, Hawaii, Dec. 2001

[5] Y. Chawathe. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. PhD thesis, UC Berkeley, 2000

[6] Y. Chu, S. G. Rao, et al. A Case for End System Multicast. IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, To Appear.

[7] Y. Chu, S. G. Rao, et al. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture, ACM SIGCOMM, San Diego, Aug. 2001

[8] Edge Side Includes (ESI) Web Page. http://www.esi.org/

[9] X. Fu, W. Shi, et al. CANS: Composable, Adaptive Network Services Infrastructure. Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01), San Francisco, Mar. 2001

[10] S. D. Gribble, M. Welsh, et al. The Ninja Architecture for Robust Internet-Scale Systems and Services. Computer Networks: Special Issue on Pervasive Computing, Vol. 35, No. 4, pp473-497, Mar. 2001

[11] D. Hrishikesh, B. Mayank, et al. Streaming Live Media over a Peer-to-Peer Network. Technical Report, Stanford University, Apr. 2001

[12] J. Jannotti, D. Gi.ord, et al. Overcast: Reliable Multicasting with an Overlay Network. Proc. of the 4th Symp. on Operating System Design and Implementation (OSDI), Oct. 2000

[13] W. Y. Ma, I. Bedner, et al. A Framework for Adaptive Content Delivery in Heterogeneous Network Environments. Proc. of SPIE Multimedia Computing and Networking 2000, pp86-100, San Jose, Jan. 2000.

[14] Microsoft DirectShow Web Page. http://msdn.microsoft.com/library/en-us/wcegmm/htm/dshow.asp

[15] W. T. Ooi, R. Renesse. Distributing Media Transformation Over Multiple Media Gateways. Proc. of the 9th ACM Intl. Multimedia Conference, Ottawa, Canada, Sep. 2001

[16] Open Pluggable Edge Services (OPES) Web Page. http://www.ietf-opes.org/

[17] L. Qin, S. Ratnasamy, et al. Can Heterogeneity Make Gnutella Scalable. First Intl. Workshop on Peer-to-Peer Systems (IPTPS) 2002, Cambridge, MA, USA, Mar. 2002

[18] S. Ratnasamy, P. Francis, et al. A Scalable Content-Addressable Network. ACM SIGCOMM, 2001

[19] S. Ratnasamy, M. Handley, et al. Topologically-Aware Overlay Construction and Server Selection. IEEE INFOCOMM, 2002

[20] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pp329-350, Nov. 2001

[21] S. Savage, T. Anderson, et al., Detour: a Case for Informed Internet Routing and Transport. IEEE Micro, Vol. 19, No. 1, pp 50-59, Jan. 1999

[22] S. Sen, J. Rexford, et al. Proxy Prefix Caching for Multimedia Streams. IEEE INFOCOMM, New York, pp1310-1319, Mar. 1999

[23] I. Stoica, R. Morris,et al. Chord: A Peer-to-Peer Lookup Service for Internet Applications. HotOS VIII, 2001

[24] D. L. Tennenhouse, J. M. Smith, et al. A Survey of Active Network Research. IEEE Communications Magazine, Vol. 35, No. 1, pp80-86, Jan. 1997

[25] H. J. Wang, B. Raman, et al. ICEBERG: An Internet-core Network Architecture for Integrated Communications. IEEE Personal Communications: Special Issue on IP-based Mobile Telecommunication Networks, pp10-19, Aug. 2000

[26] WebSphere Edge Services. IBM white paper. Jan. 2002.

[27] M. Yarvis, P. Reiher, et al. Conductor: A Framework for Distributed Adaptation. Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS VII), Rio Rico, AZ, Mar. 1999

[28] B. Y. Zhao, J. D. Kubiatowicz, et al. Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing, Tech. Report, UC Berkeley, Apr. 2001