Journal of Universal Computer Science, vol. 17, no. 6 (2011), 961-980
submitted: 15/5/10, accepted: 30/11/10, appeared: 28/3/11 © *J.UCS*

Service Oriented Multimedia Delivery System in Pervasive Environments

Zhuzhong Qian
(State Key Laboratory for Novel Software Technology, Nanjing, P.R. China
qzz@nju.edu.cn)

Sheng Zhang
(State Key Laboratory for Novel Software Technology, Nanjing, P.R. China
zhangsheng@dislab.nju.edu.cn)

Kangbin Yim
(Department of Information Security Engineering, Soonchunhyang University, Asan, Korea
yim@sch.ac.kr)

Sanglu Lu
(State Key Laboratory for Novel Software Technology, Nanjing, P.R. China
sanglu@nju.edu.cn)

Abstract: Service composition is an effective approach for large-scale multimedia delivery. In previous works, user requirement is represented as one fixed functional path which is composed of several functional components in a certain order. Actually, there may be several functional paths (deliver different quality level multimedia data, e.g., image pixel, frame rate) that can meet one request. And due to the diversity of devices and connections in pervasive environment, system should choose a suitable media quality delivery path in accordance with context, instead of one fixed functional path. This paper presents a deep study of multimedia delivery problem and proposes an on-line algorithm *LDPath* and an off-line centralized algorithm *LD/RPath* respectively. *LDPath* aims at delivering multimedia data to end user with lowest delay by choosing services to build delivery paths hop-by-hop, which is adapted to the unstable open environment. And *LD/RPath* is developed for a relatively stable environment, which generates delivery paths according to the trade-off between delay and reliability metrics, because the service reliability is also an important fact in such scenario. Experimental results show that both algorithms have good performance with low overhead to the system.

Keywords: pervasive environment, service composition, multimedia delivery
Categories: H.3.2

1 Introduction

Due to the development of IT technology, intelligent pervasive computing environment which is based on pervasive network infrastructure and related services attracts more and more attention. Effective multimedia delivery is an important service in pervasive environment, which is used to entertainment, real-time monitor, video conference, etc. Service composition [Benatallah, 02], the core technology for service oriented architecture (SOA) [Nahrstedt, 05] provides a promising solution to

dynamically building multimedia delivery system. SOA based multimedia delivery has been widely discussed in recent years.

In SOA, the basic multimedia processing unit (e.g., transcoding, image compression, subtitle embedding, logo adding) are published as multimedia services. The services in a multimedia system can be classified into two categories. Non-functional services (e.g. compression, sampling) only changes the quality of media, while functional services (e.g., subtitle embedding, logo adding) provides certain functions. There are also some services which have both of the two effects, such as transcoding. Then, the new multimedia applications can be easily constructed through compositing the basic services, which is represented as a functional path with a set of non-functional components. Since the non-functional services can reduce the data size, so the transmission delay is reduced as well. However, the execution of the non-functional services is also time consuming [Qian, 09], so when and where to use these non-functional services is a challenge. The functional path of a composite service represents the abstract execution flow. In run time, the system chooses a set of service instances to construct a real multimedia delivery path based on the functional path.
In a pervasive environment, each service components may have several replicas which are deployed in different nodes. And because of the heterogeneity of the devices and connections, these replicas have different processing and transmission delay for different service requesters. Consequently, choosing suitable service replicas to deliver and process the multimedia data will have a great effect on the end-to-end QoS.

To terminal users, delay and media quality (e.g., image pixel, frame rate) mainly affect their experience [Carlos, 10]. Intuitively, low delay and high quality media is perfect. However, high quality media means the large amount of data transmission which brings a higher delay. On the other hand, high quality media is not always necessary, because the quality of image is also limited by the display device. To a high pixel video, a low pixel display device has to discard redundant data which costs computing resources. In a pervasive computing environment, most mobile devices (e.g., cell phone, PDA, mp4 player) have limited computing resources, low bandwidth connection and a relatively small screen. Generally, to delivery media data to these mobile devices [Kafle, 10] in a pervasive environment by instable connection, delay is major factor to be considered.

Based on the above discussion, this paper studies the problem of building a multimedia delivery path with low-delay from source to destination by choosing a set of suitable service replicas. Furthermore, for different multimedia applications, the basic environments are different, which results in different delivery strategies. For a large-scale pervasive environment, there are large amount of service replicas deployed in lots of nodes. However, in a small environment such as home network, the number of services is limited and the resources are relatively controllable and stable. It is difficult to get all the information of the environment in the first scenario, thus, we propose an on-line distributed algorithm *LDPath* [Qian, 09] to deliver the multimedia data hop-by-hop. In the latter scenario, an off-line centralized algorithm *LD/RPath* [Zhang, 09] is proposed to generate the entire multimedia delivery path in advance which is used to deliver the multimedia data for a certain multimedia request. In the meantime, *LD/RPath* also considers the reliability of the services, because if one of the services was down, the cost of rebuilt service path is high.

The rest of paper is organized as follows. Section II presents some related works, and then we describe the multimedia delivery system in pervasive environments in section III. Section IV and V is the detailed description of *LDPath* and *LD/RPath* respectively, and section VI presents a prototype based on these algorithms and gives the simulation results. We draw some conclusions in section VII at the end.

2 Related Works

SOA based multimedia delivery is closely related to the service composition. Several research works have addressed this problem.

Automatic service synthesis considers how to coordinate the service components to meet the functional requirement. Project Colombo [Berardi, 05] proposed a formal model of service which employs automation to describe service process. Based on the description model several PDL rules are defined to automatic service composition. Bultan et al. [Bultan, 03] defined a session model to present the communication among single services and investigated the integrated behavior of the composite service which is superposition of related services. OWL-S (web ontology language for web service) [OWL-S] is a semantic service model which defines three top ontologies to describe the semantic of service. Basically OWL-S based service composition is depended on the IOPE (Input, Output, Precondition, and Effect) of services and utilizes workflow, Pi-calculation and intelligent planning to achieve the goal.

Since service instances running on different nodes may offer same function, system should select a most suitable instance to achieve the task in runtime. QoS-constrained service composition just focuses on finding such a service path (composed of service instances) that satisfies the QoS constraints. Zeng et al. [Zeng, 03] considered quality driven service composition. They advocated that the selection of underlying component services should be done at runtime rather than design-time, because there are many quality criteria of elementary service and many of them could not be calculated before runtime. They used linear programming to solve the service composition with multiple constraints problem. Raman et al. [Raman, 03] considered load balancing across service instances. They proposed a metric called least-inverse-available-capacity (LIAC) to choose a set of services instances to complete service composition, and they used piggybacking mechanism to make their method more efficient. They also found that if the service path is too long, it may cause overload or long end-to-end delay to the system. To discourage long paths, they added a α-factor to PathCost metric. Kalasapur et al. [Kalasapur, 07] proposed a framework for dynamic service composition and the concept of hierarchical service overlay for handling the mobility and dynamicity of a pervasive environment. They took advantage of directed graph to represent all the services in system directories and their relationships.

Choi et al. [Choi, 03] presented some general approaches to some path selection problems by transforming them to conventional shortest path problems, which could be solved by Dijkstra shortest path algorithm or Bellman-Ford algorithm. Gu et al. [Gu, 04, Gu, 06] proposed SpiderNet, an integrated peer-to-peer service composition framework to find a service path ensuring the multi-constraints by heuristic algorithm

which selects service instances hop-by-hop. SpiderNet also provides statistical multi-constrained QoS assurances and load balancing for service composition. Raman et al. [Raman, 02] proposed architecture based on an overlay network of service clusters to provide failure-resilient composition of service across the wide-area Internet.

Previous works widely investigated service composition and SOA based multimedia provision, however for pervasive multimedia delivery, they have limitations. Firstly, several functional paths may meet the user requirements while previous works only considered one functional path. Secondly, large amount of data is the critical factor for QoS guarantee, some components on service path significantly change the data size which will have a deep effect on the transmission delay. Thirdly, due to the diversity and mobility of pervasive environments, reliability of service is needed to be considered. Finally, though the context-aware technology is already used in service composition, they mainly focused on functional switching among different scenarios, and did not consider the relationship between function and QoS.

3 Multimedia Delivery System in A Pervasive Environment

3.1 Pervasive Multimedia Delivery Environment

Our research project Pervasive Service Provision Infrastructure (PSPI) has already achieved a service overlay with context of services and self-adaptive transfer mode switching. PSPI has four major parts: (1) *networking module* provides the connections among all the different kinds of devices such as workstation, laptop and PDA; (2) *resource agent*s acquire the status of devices and connections and update data in the database periodically; (3) *context manager* collects context information from resource agent and provides a uniform interface for other parts of PSPI to use information data; (4) *task parser* accepts user requests and generates functional graphs. In this paper, we focus on the multimedia delivery technology, based on PSPI, to provide optimum service in accordance with the context of services and user. Through providing pervasive services, PSPI can assist human life and make the environment more intelligent.

To clearly identify the problem, we introduce the following 4 notions for a pervasive multimedia delivery environment:

- Bandwidth: each physical link has a bandwidth, denoted as b.
- Unit Processing Time: the computing ability of two service replicas nodes may be different, so the processing delay is different when they process the multimedia data. And the multimedia application is data intensive, so the processing time is highly related to the data quantity. Thus, we imported Unit Processing Time denoted by o, to represent the average processing time of unit multimedia data.
- *IO-ratio*: *IO-ratio* is defined as the ratio of output data to input data size on a service replica node, denoted by r. This parameter determines the data transmission volume and significantly impacts on the delay of multimedia delivery, especially in a pervasive environment.
- Reliability: the reliability of a service replica is the probability that a request is correctly responded within a maximum expected time frame, denoted by y. It can be computed from historical data [Zeng, 03], e.g., $y=C/K$, where C is the

number of successful invocations during a period of time while *K* is the total number of invocations during the same period.

3.2 Functional Graph and Service Graph

To compose a new complex service, system should generate a specification of composite service to meet the user's requirement. Our previous work [Qian, 07] has proposed the service composition mechanism to automatically generate the specification on a composite service according to the user's input, desired actions and output. As we mentioned before, there are non-functional services that only changes the data transmission volume and affect the media quality. Thus, for one multimedia delivery request, there may several functional paths providing multi-quality media, each of which includes different non-functional services. That is, there may be several ways to deliver multimedia data from the source to user with different media quality. If the media quality is acceptable for the media requester, the corresponding functional path is considered as a candidate delivery path. Hence, we define a function graph to represent all these possible delivery approaches.

Definition 1 (Functional Graph): A functional graph is defined as FG = <F, FL, f_0, F_t>, where F is the set of functional components represented by *function name* (denoted by *f*) that indicates its service function. FL is the set of functional links (denoted by *fl*) between these components, f_0 is the initial functional component and F_t is the terminal functional component.

We introduce functional graph to represent all the possible functional paths from source to destination. Each functional link *fl* gives the dependences between two functional components. $fl_{i,j}$ means component *i* should be executed before component *j*. In functional graph, the functional component whose in-degree is zero is called initial component f_0 and the one whose out-degree is zero is called terminal component F_t. A path from the initial component to the terminal component is called a *functional path*, which is a possible media delivery path to meet a user's functional requirement.

Suppose there are 5 functional components in a pervasive environment, as Fig. 1 shows. There are four possible paths ($f_0f_1f_4$, $f_0f_1f_3f_4$, $f_0f_1f_2f_4$, $f_0f_1f_2f_3f_4$) from media source to f_4.

Functional graph indicates an abstract level of the multimedia delivery order in a pervasive environment. Each component may have several service replicas deployed on different nodes. And these functional components will finally be appointed to the related services to achieve the data delivery. So we introduce the concept of service graph to represents the actual delivery path in a pervasive environment.

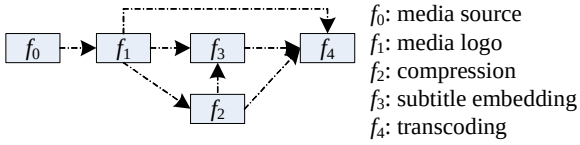


Figure 1: Functional graph

Definition 2 (Service Graph): A service graph is defined as SG = <S, E, S_0, S_t>, where S is a set of related services and E is a set of overlay links, S_0 is initial source service which has the desired media data, and S_t is terminal services.

Fig. 2 shows the related service graph of Fig. 1, where service node S_k^i provides service f_k. The outgoing links of a service means, if data is delivered to this service, the next possible service that data may be transmitted to. Typically, we add S_t as the successor of all the terminal services to represent user receiving data. As we see, one functional path may correspond to several service paths due to the existence of service replicas.

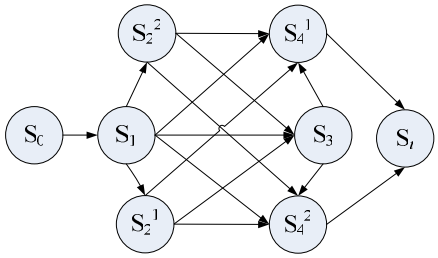


Figure 2: Service graph

4 Low delay multimedia delivery: *LDPath*

In a large-scale pervasive environment, it is difficult to get the entire context information of the whole system, thus, we propose an on-line heuristic algorithm to deliver the multimedia data hop-by-hop. For multimedia delivery, *delay* is the major factor to qualify the QoS. We define a delay metric of a service path, denoted by δ_P, which is the sum of data transmission between services and processing time on services. The delay of a service path *p* can be calculated by the following equation

$$\delta_p = m \cdot o_0 + \sum_{e_{i,j} \in p} \left(\frac{m_i \cdot r_i}{b_{i,j}} + m_i \cdot r_i \cdot o_j\right) \qquad (1)$$

m is the original data size and mo_0 represents the delay of processing on s_0. m_i is the amount of the incoming media data of service s_i, $m_i r_i$ is the outgoing data after processing, and $b_{i,j}$ represents bandwidth of $e_{i,j}$. $\left(\frac{m_i \cdot r_i}{b_{i,j}} + m_i \cdot r_i \cdot o_j\right)$represents the transmission delay of service link $e_{i,j}$ and processing delay of service s_j. We formulate the multimedia delivery problem as selecting a service path in SG, and minimums the delay δ_P.

Although the delay is the only metric to the end user, for the relay service, we should both consider delay and *IO-ratio*. That is to say, there are actually two constraints and we need to seek a tradeoff between delay and *IO-ratio*. From this point of view, LDMD is equivalence to Multi-Constrained Path (MCP) problem which is known to NP-hard [Cormen, 96].

Due to the NP-completeness of the problem, we need to resort to reasonable simplifications to make the problem can be solved within polynomial time complexity. The distributed algorithm *LDPath* which is short for Lowest Delay Path is designed to choose service replicas hop-by-hop and generates path matches one of the functional path, which achieves the context aware multimedia delivery.

Different from conventional single source shortest path problem, the transmission delay of a service link varied in accordance with the change of transmission data. Let us take Fig. 1 as an example, suppose there is only one service replica for each service component in Fig. 1, then it is also the service graph. Assume that m is the initial data size at media source f_0, then we consider the data size on f_3. If the path is $f_0f_1f_3$, then f_3 receives mr_0r_1 data; and if the path is $f_0f_1f_2f_3$, then f_3 receives $mr_0r_1r_2$ data. We can see the unpredictable data size from this simple example.

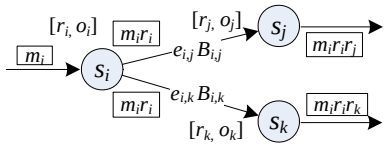


Figure 3: One part of a service graph

4.1 *ω,r*-Labeled Graph

Suppose service s_i in SG has two successor services s_j and s_k. Fig. 3 shows the context of these services, vector [r, o] is functional property and b is bandwidth of service link. Rectangles indicate the data transmission in different service links. After the multimedia data (size: m_i) is processed in s_i, if s_j is selected as the following service, then the transmission delay is $mr_i/b_{i,j}$, and the processing time on s_j is $mr_i \cdot o_j$, thus the total delay is

$$\delta_{i,j} = \frac{m_i r_i}{b_{i,j}} + m_i r_i o_j = m_i r_i (\frac{1}{b_{i,j}} + o_j) \qquad (2)$$

Suppose the initial data size is m, and the media data is delivered to s_i through service path p, then

$$\delta_{i,j} = m(\prod_{s_i \in p_{s_0,s_t}} r_i)(\frac{1}{b_{i,j}} + o_j) \qquad (3)$$

Let $\eta_{iP} = (\prod_{s_i \in P_{s_o,s_i}} r_i)$, and $\omega_{i,j} = (\frac{1}{b_{i,j}} + o_j)$. η_{iP} represents the ratio of data volume after process of s_i through service path P and the original data volume, ω denotes the delay (sum of transmission rate and processing rate) rate of service link e. Thus, $\eta_{iP} \cdot \omega_{i,j}$ indicates the potential delay of selecting service link $e_{i,j}$ as the next delivery link and service s_j as the successor of service s_i. Furthermore, the r_i indicates the following

data volume, thus we use vector [ω, r] as the label of a service link which means the delay of choosing this service link (including transmission and processing, in the rest paper, we use service link to represent the link and the related service), and the change of data volume. The service graph labeled with [ω, r] is called ω,r-labeled graph. Fig. 4 is an example of ω,r- labeled graph based on the service graph shown in Fig. 2. It is worth notice that, η is the product of all r values on a service path, thus, for different paths, η may be different. However, for a given service path P, η is constant, and the total delay δ_P can be represented as the following equation

$$\delta_P = m \cdot o_{s_0} + \sum_{e_{i,j} \in P} (m \cdot \eta_{iP} \cdot \omega_{i,j}) = m \cdot (o_{s_0} + \sum_{e_{i,j} \in P} (\eta_{iP} \cdot \omega_{i,j})) \qquad (4)$$

m and o_{s0} are constants, so the problem can be converted into choosing a service path that minimizes $\sum_{e_{i,j} \in P} (\eta_{iP} \cdot \omega_{i,j})$.

4.2 LDPath

In this subsection, we present *LDPath* to choose an optimum path for multimedia delivery, which is a heuristics algorithm and constructs the service path hop-by-hop. The basic idea of *LDPath* is choosing a service link with low ω and small *IO-ratio*. Low ω means the link has a large bandwidth and the related service has low processing time, while small *IO-ratio* indicates the data volume in the following transmission path will be reduced. Furthermore, closer the service link is to the source, bigger the "contribution" of its *IO-ratio* is to reduce or increase the delay of the whole service path. That is because more posterior service links on the path will be affected by the *IO-ratio* of those services which are close to the source.

After multimedia data is delivered to a service node and has been processed, it should choose a successor service. Suppose the current service is s_i, following we give an aggregated evaluation function *ldf* to assess successor service s_j.

$$ldf(s_j) = \omega_{i,j} + \omega^{Exp}(s_j) \cdot r_j \cdot (1 + \frac{1}{p}) \qquad (5)$$

ω^{Exp} indicates if the service link $e_{i,j}$ is chosen as the next data delivery link, i.e. the s_j is selected as the successor service, in the next step, the expected ω value of s_j's successor. Variable p denotes the position of the service link in the entire service path. Function *ldf* considers the potential effect of *IO-ratio* as well as the current delay. The product of $\omega^{Exp}(s_j) \cdot r_j$ indicates the possible delay of s_j's successor. $(1 + \frac{1}{p})$ is the weight, which represents that as p increasing, the effect of *IO-ratio* is smaller, i.e. if the service link is closer to user, its effect to the delay of whole path is smaller than that of close to source.

The calculation of ω^{Exp} is based on the successor service links and related services of s_j. Intuitively, we will choose a low delay one as the successor. If there is only one successor service link e_u and related service s_u, $\omega^{Exp} = \omega_{j,u}$. Otherwise, for all the

successors, we first remove the maximum-delayed one and ω^{Exp} is the average of these ω values.

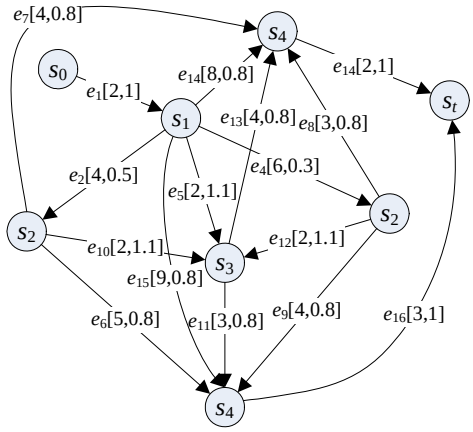


Figure 4: An example of ω,r-Labeled Graph

LDPath generates the service path from data source hop-by-hop. The successor that has a lowest *ldf* will be chosen. When the service s_t is selected, the algorithm terminates. The maximum number of hop is just the maximum length of all the service paths, which is constant for a given service graph. For each hop, it calculates ω^{Exp} and *ldf* of all the successor links and related services. Let N_e denotes the number of service links in SG. Because one service link is used at most once to calculate ω^{Exp} for the whole algorithms, totally selecting maximum-delayed successor is less than N_e, and calculating average is less than $2N_e$. Thus, the total calculation of ω^{Exp} is $O(N_e)$. The complexity of calculating all the *ldf*, i.e. complexity of *LDPath*, is $O(N_e)$.

We use Fig. 4 as an example to show how *LDPath* works. Starting from source service node s_0, *LDPath* assesses the adjacent services with evaluation function *ldf*. Because there is only one successor service link of s_0, e_1 is selected as the second hop. For service s_1, there are 5 successors, and the *ldf* values of e_4, e_5, e_2, e_{14}, e_{15} are 6.8, 6.4, 5.1, 10.4 and 12.7 respectively. Thus e_2 is selected as the third hop. In the same way, e_{10}, e_{11} and e_{15} are selected. Thus, we get a service path $p = e_1e_2e_{10}e_{11}e_{15}$. The total delay of this path δ_p is 9.176.

5 Lowest Delay/Reliability Delivery: LD/RPath

LDPath is designed for large-scale network with requests from different nodes. However, in some small-scale environments such as home network, the number of services is limited and the resources are relatively controllable and stable, where the entire context information could be collected easily. In this case, a centralized off-line algorithm is possible to generate a service path. And because the context is relatively stable in a small-scale environment, the system may use the generated path for the

whole delivery session unless the failure occurs. Because the centralized algorithm utilizes all the context information, it could generate a service path that is very closed to an optimized path.

But for a generated service path, if one of the service replicas fails in runtime, we have to reconstruct a service path, which may greatly increase the delay. Thus, the reliability of a service is introduced to represent the probability that a request is correctly responded within a maximum expected time frame of a service replica [Zeng, 03]. For the simplicity, all the physical links are assumed to be reliable all the time. As the reliability metric is multiplicative [Gu, 04], the reliability of a path is the product of all the reliability of service replicas along the path. The reliability of a path *p* is

$$y_p = \prod_{S_i \in p} y_i \qquad (6)$$

So the target is to make δ_p as smaller as possible and to make y_p as bigger as possible, there is a trade-off between these two targets. We proposed a centralized algorithm *LD/RPath* (Lowest Delay/Reliability Delivery Path) for a small network environment. In this section, we firstly give the basic idea of *LD/RPath*, then we present descriptions of the algorithm with details, finally we briefly give its time complexity analysis.

5.1 The Basic Idea of the Algorighm

Since *delay* and *reliability* are two independent QoS metrics and in order to find a low delay with high reliability, we define an integrated metric *dy* to evaluate the overall QoS of the multimedia delivery path.

$$dy_p = \frac{\delta_p}{y_p} \qquad (7)$$

δ_p and y_p is the delay and the reliability of path *p*, respectively. Our algorithm heuristically solves the problem to make dy_p as smaller as possible. *LD/RPath* is named for this purpose.

Due to the exponential complexity of the problem and the real-time requirement of the multimedia delivery application, a polynomial and heuristic algorithm is needed. We are motivated by several previous work [OWL-S, Qian, 07, Cormen, 96] to think in terms of transforming the existing problem to a classical shortest path problem, which can be easily solved by Dijkstra shortest path algorithm, Bellman-Ford algorithm or SPFA [Cormen, 96]. However, it is quite different from the shortest path problem. Firstly, both nodes and edges have weights in this problem while only edges have weight in conventional shortest path problems; secondly, we need to consider two QoS metrics while classical shortest path problem just has only one QoS metric; thirdly, the change of multimedia data size makes the delay on each physical link or each service replica node unpredictable before the path is generated.

To convert the problem, we divide *LD/RPath* into four phases according to the above analysis. *i) Splitting* is to convert the weights that on the nodes as the weights on the links; *ii) Approximate Estimating* is to approximately estimate the data size on each link and each service replica node, making delay predictable; *iii) Combining* is to combine reliability and delay into one parameter. After that, it is converted into a single source shortest path problem, we can use Dijkstra algorithm to solve it.

5.2 Descriptions of LD/RPath

In previous works [Raman, 03, Choi, 03], there are many examples about how to transform a complex network problem into a conventional shortest path problem. For example, if there are several relay nodes and a set of data processing nodes in the system. We need to deliver the data from the source to the destination through the relay nodes and must via one processing node. In order to find a shortest path from the source to the destination, instead of trying all the possible paths through the processing site, we can achieve it in the following procedure. We first copy the original network graph, and we add edges from one processing node in the original graph to the corresponding node in the copied graph. These edges represent the processing procedures (may be evaluated as processing delay) on the node. And then, we run the shortest path algorithm to find a shortest path from the source node in the original graph to the destination node in the copied graph. And the result is the desired shortest path.

i) Splitting

How can we unify all the weights to be labeled on links? Motivated by the above technology, we split every service replica node into two nodes and add a link between them to represent the service itself. Let us take S_3 in Fig. 2 for example, as Fig. 5 illustrates. S_3 is divided into two nodes and the link between the two new nodes (we call it an inner-link) represents the service, so we can label all the weights of the original service replica node on the new inner-link.

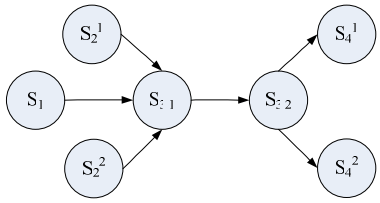


Figure 5: Splitting service node

ii) Approximate Estimating

After splitting, all the weights on the nodes are converted into links. In order to transform the problem into a conventional shortest path problem, we need to find a way to calculate all the data size on each link before the path is generated. We use an approximate estimating to calculate the data size. The algorithm is as Algorithm 1 shows. Its main idea is that for each service replica node, we use the average size of the multimedia data on each of its ingoing links to represent the data size it receives.

Algorithm 1 Approximate Estimating

```
1:  Initialization: index[i] ← 0
2:  flag ← true
3:  while flag == true do
4:     { flag ← false
5:       for each link(i,j) ∈ E do
6:          { if index(j) ≤ index(i) then
7:               { index(j) ← index(i) + 1
8:                 flag ← true }}}
9:  for each index(i) do
10:    { c ← 0, sum ← 0
11:      for each node v ∈ V do
12:         { if v has a link to index(i) then
13:              { c ← c + 1, sum ← sum + m[v] }}
14:     m[index(i)] = sum/c }
15: return m[0, N_n - 1]
```

We divide the algorithm into two parts. Part 1 (line 1 to 8) is to sort all service replica nodes. Part 2 (line 9 to 14) is an approximate calculating. When we calculate the data size on a node, we need to know all the data size on each of its ingoing links, that is, we also need to know the data size of the head nodes of these ingoing links. It is a recursive procedure, so we sort them, giving each node an index. Then we can do the approximate estimating according the sorted order.

iii) Combining

For each service replica node, we use the reciprocal of its reliability as a coefficient of the processing delay weight, which is intuited from work [Raman, 03]. It is because the cost of choosing a service replica node is inversely proportional to its reliability. That is, the larger reliability, the more likely that it will be chosen. By this method, we heuristically achieve a trade-off between delay and reliability.

The above three steps transformed the original service graph into a graph with fixed nonnegative weights on edges, which is suitable for Dijkstra shortest path algorithm. If a link (edge) is an inner link, its weight is of the form

$$\frac{1}{y_i} \cdot m_i \cdot o_i \qquad (8)$$

y_i and o_i are the reliability and unit processing time of the service replica this inner link represents, respectively. m_i is a approximate data size. If it is not an inner link, its weight is of the form

$$\frac{m_i}{b_i} \qquad (9)$$

b_i is the bandwidth of this link.

We introduced the metric *dy* as the optimized target, and all the four steps are designed to heuristically make *dy* as smaller as possible, which may indicate that delay is small and reliability is high. Here we give the time complexity analysis. Let N_n, N_e denotes the number of nodes and links in the service graph respectively. Step 1 needs to process each node and search all the ingoing and outgoing links of the node, so it takes $O(N_n N_e)$ time to process. Step 2 is similar to Step 1, it also needs $O(N_n N_e)$ time. Step 3 only needs to process each inner link, so it only needs $O(N_n)$ time. The time complexity of the well known Dijkstra shortest path algorithm is

$O(n^2)$, where *n* is the number of nodes in the transformed graph. We get $n < 2N_n$ according to the splitting property. From the graph theory, we also know that $N_e \leq N_n^2$. So the total time complexity of *LD/RPath* is

$$O(N_n N_e + N_n N_e + N_n + N_e + n^2) = O(N_n^3) \qquad (10)$$

6 Experiments and Results

The performance of two proposed algorithms is presented in this section. Firstly we briefly introduce a context-aware video surveillance system which is an SOA based multimedia application on PSPI, and then we introduce our simulation system model and show the numerical results of the two algorithms respectively.

6.1 Video Surveillance System

We implemented a set of multimedia services based on PSPI, which provides certain basic functions, e.g., video capture, compression, sampling, format conversion (BMP format to JPEG format, including five sub-services).

The context aware video surveillance system monitors certain devices through cameras (video capture service) and transmits the multimedia data to user. There are several kinds of functional paths to deliver surveillance video data: (1) direct delivery; (2) compression and delivery; (3) sampling as BMP format pictures and delivery; (4) sampling, format conversion and delivery. These four ways delivers different quality of data to satisfy different quality requirements.

According to the terminal device and context of pervasive space, a certain path is chosen to deliver the multimedia data to a terminal user. For example, when a terminal user who uses a work station with high speed wire connection, we directly deliver the video to the terminal; while a laptop with wireless connection to the network, surveillance data is compressed or converted into images through sampling service, then is delivered to user; if a user uses a PDA to receive monitor video data, due to its unstable connections and limited computing resources, the sampled images may be converted into JPG format to decrease the data size for smooth delivery.

From the PSPI experimental results, the above scenarios are not fixed. According to the context, *LDPath* selects one optimal path (lowest delay) to deliver the surveillance data to user while *LD/RPath* selects an optimal path to deliver the surveillance data to a user with low delay and high reliability. Especially when a user is roaming in the environment, the delivery path will change when he is moving from one point to another if system detects a better one with lower delay or higher reliability.

6.2 Simulation System Model

We implemented the simulation system using Java and all simulations were performed on a computer: Intel (R) Core (TM) 2 Duo CPU E8200@2.66GHz with 2 G bytes of memory.

The distributed *LDPath* algorithm which aims at delivering multimedia data to end user with lowest delay is designed for large network topology size, because that (1) the real-time context of large network is hard to collect or update; (2) *LDPath*

heuristically chooses service replica hop-by-hop, which makes it time-efficient for large network; (3) hop-by-hop choosing makes *LDPath* do not need to take reliability into consideration. The centralized *LD/RPath* which heuristically generates a delivery path with lower delay and higher reliability is designed for small network size, because that it is centralized and effective when the network topology size is small. As they have different application fields, we didn't design the comparison experiments between *LDPath* and *LD/RPath* in our simulation experiments.

In the simulation system, all parameters are designed to be tunable. We assume that there are 2000 physical nodes to deploy service components (if the node is not deployed with services, it is a relay node). We first generate *MaxService* kinds of service components and randomly construct functional graph. Based on the functional graphs, system generates the related service graphs. For each kind of service component, the number of its service replicas is between *MinInstance* and *MaxInstance*. *IO-ratio*, unit processing time, bandwidth and reliability parameters are all generated following some kinds of normal distributions. We would mention that the value of unit processing time *o* is related to *IO-ratio r*. In general, they are to some extent in inverse proportion, which means that larger compression ratio needs more processing time. We can use parameters *MaxService*, *MinInstance* and *MaxInstance* to control the network topology size for simulation experiments.

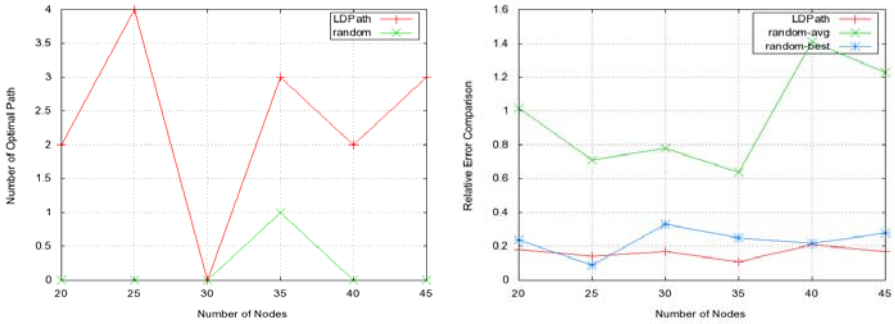
For comparison, we implemented *optimal* and *random* algorithm. The *optimal* algorithm uses network flooding, which search all the possible service paths to find the best one with respect to a certain metric. The *random* algorithm randomly selects service replicas hop-by-hop. We would mention that the flooding is high resource consumption and the *optimal* algorithm is limited in the scale of network graph topology size. In the following two subsections, we present numerical experiment results of *LDPath* and *LD/RPath* respectively.

6.3 Performance Evaluation of LDPath

We show the numerical results of comparisons between *LDPath*, *random* and *optimal* in this subsection. Algorithm *optimal* is overflow when the total number of service replicas is large than 50. Thus we firstly setup experiment (*LDPath*, *random* and *optimal*) within 50 nodes to illustrate performance of *LDPath*, then we run the comparisons between *LDPath* and *random* in large network topology sizes.

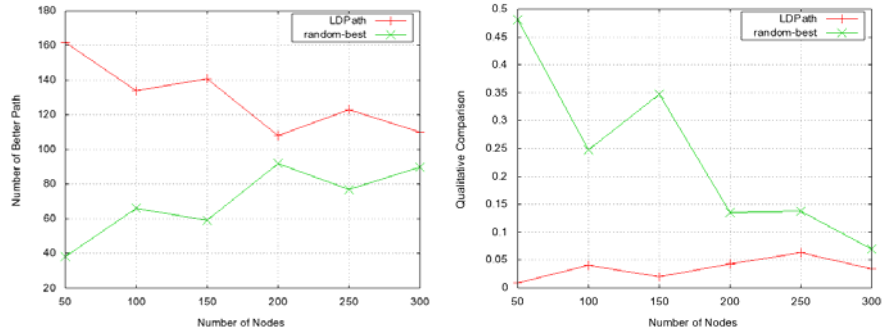
We categorize the experiment as 6 groups by the number of nodes. Each group runs 200 sets of experiment with different structures of service graph (each set runs *LDPath*, *optimal* for one time and *random* for 20 times). Since the optimal algorithm gets the lowest delay service path, we compare the results of *LDPath* and *random* with optimal. Fig. 6(a) illustrates the number of generated path by *LDPath* and *random* that hits the optimal path for each group. Only few generated paths of both two algorithms are exactly the optimal path, and it is obviously that *LDPath* is much better than random which only hit one path among all these experiments. In Fig. 6(b), we show the relative error of *LDPath* and *random*. We use the delay of the path found by optimal as the standard for normalization. For any path *p*, its *relative error* is calculated as $\frac{\delta_p - \delta_{optimal}}{\delta_{optimal}}$, where $\delta_{optimal}$ is the delay of the path found by optimal. In Fig. 6(b), we can see that the performance of *LDPath* is much better than the average

performance of random and generally its performance is better than the best result of random.



(a) Number of optimal path (b) Relative error comparison

Figure 6: Topology within 50 nodes



(a) Number of better path (b)Qualitative comparison

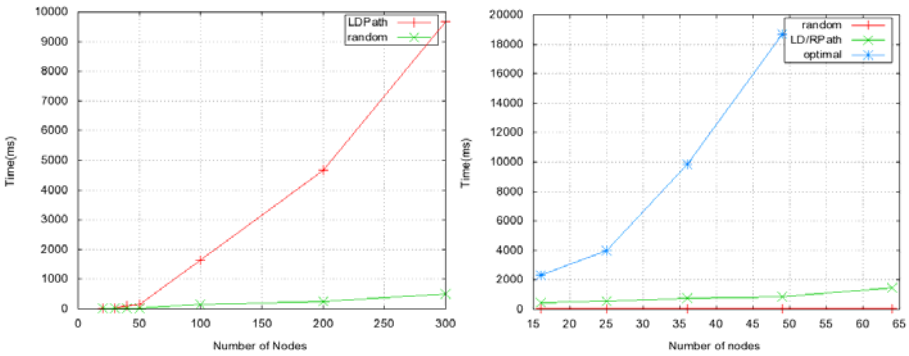
Figure 7: Large-scale topolgy

Next, we use large topologies to evaluate the algorithm. In these experiments, we also setup 6 groups of experiment with 50 to 300 nodes respectively. Each group also runs 200 sets (*LDPath* for one time and *random* for 20 times). We observe all the results and found out that the path found by *LDPath* is always better than average of random (δ_{LDPath} < AVG (δ_{random})). Fig. 7(a) shows the comparison of *LDPath* with the best results of *random*. For example, *LDPath* is 162 times better while *random-best* is 37 times better among totally 200 times when the network size is 50. We can see that in most cases, *LDPath* has a better performance. Furthermore, we show the *qualitative* comparison of the performances of *LDPath* and random. For each set, the *qualitative* value is calculated as,

$$LDpath = \frac{\sum \max(\frac{\delta_{LDpath} - \delta_{random}}{\min(\delta_{LDpath}, \delta_{random})}, 0)}{200}, \quad random = \frac{\sum \max(\frac{\delta_{random} - \delta_{LDpath}}{\min(\delta_{LDpath}, \delta_{random})}, 0)}{200} \tag{11}$$

From Fig. 7, we can see that *LDPath* has a much better performance than random-best, which means that if the path found by *LDPath* is better random, the delay is usually much smaller than *random*; while if the path generated by random is better, the delay of random is only a little better than *LDPath*.

Fig. 8(a) shows the total setup time of *LDPath* and *random* (running 200 times respectively). The running time increases rapidly as the increasing of number of service replica, because the depth of service path becomes longer. For *LDPath*, as the number of service growing, it takes more time to calculate ω^{Exp}. And we can see that the running time of *LDPath* is less than random algorithm for 20 times. Thus, *LDPath* has a better performance than random algorithm.



(a) LD/RPath (b) LD/RPath

Figure 8: Running time comparison

6.4 Performance Evaluation of LD/RPath

As centralized *LD/RPath* is designed for small network size, we mainly study the performance of *LD/RPath* in small networks in this subsection.

We firstly compared the time consumption (*optimal*, *LD/RPath* and *random*) of three algorithms at different network size. We run the three algorithms 100 times respectively, then compared the average value of running time, as Fig. 8(b) shows. We can see that the *optimal* algorithm needs much more time than *LD/RPath* or *random*. When the network topology size is 30, *LD/RPath* and *random* only consume 1s and 60ms respectively, while *optimal* consumes 20s, which is much longer than the others. When the network topology size gets larger, the running time of *LD/RPath* increases slowly and needs as much time as *random*. We also recorded the comparison results when network topology size is larger than 60 (for space limitation, we did not show it in Fig. 8(b), we found that *optimal* algorithm costs 80s when network topology size is about 70. In a real-time multimedia delivery system, 80s is too longer for a user to bear or wait, so we do not need to consider *optimal* algorithm when network topology size is larger than 70.

Then we compared the path selection effectiveness at different network size. We categorize the experiment as 3 groups by the number of nodes. Each group runs 100 sets of experiment with different structures of service graph, each set includes one

time *optimal*, one time *LD/RPath* and 100 times *random*. And we use the best result of each 100 times *random* for comparison, we term this best result *random-best*.

Firstly, we explored the detail results of the first group (*MaxService* is 7, *MinInstance* is 2 and *MaxInstance* is 7), that is, there are about 30 nodes. Fig. 9, Fig. 10 and Fig. 11 illustrate the results (for space limitation, we just lists 50 sets of comparisons). In Fig. 9, we can see that the delay of the path generated by *LD/RPath* is as smaller as *optima*l in most cases. More precise statistical data shows that the delay of path generated by *LD/RPath* is not bigger than *random-best* for 39 times among 50 times. Fig. 10 shows the reliability comparison results, the reliability of path generated by *LD/RPath* is not lower than *random-best* for 32 times among 50 times. Fig. 11 shows the comparison results of *dr* which is an integrated metric that represents the quality of the service path. Statistical data shows that *LD/RPath* is better than *random-best* for 38 times. These three figures approve that *LD/RPath* is much better than *random* which arbitrarily selects the next hop service replica.

Secondly, we compared the results of all the 3 groups (the number of nodes is 30, 60 and 100). Here we need not take into account the *optimal* algorithm, because that *optimal* algorithm cannot guarantee real-time when network topology size gets larger than 70. Table I shows the results. For each group, we recorded the number of times when the delay of path generated by *LD/RPath* is smaller than *random-best* and the number of times when the reliability of path generated by *LD/RPath* is higher than *random-best*. For example, when there are 30 nodes in the network, *LD/RPath* generated smaller delay in 82% of total comparisons and generated higher reliability in 80% of total comparisons. From the table, We can see that when the network topology size gets larger, *LD/RPath* still maintains good performance: a higher than 80 percentage cases that better than *random-best*. Table I approves that *LD/RPath* has much better performance than *random-best*.

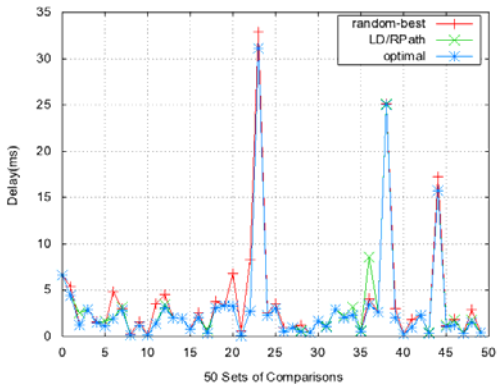


Figure 9: Comparison of Delay

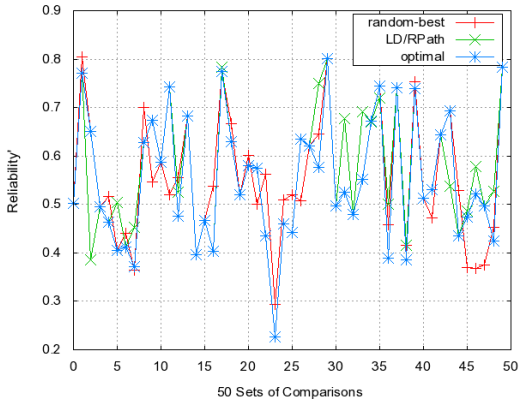


Figure 10: Comparison of Reliability

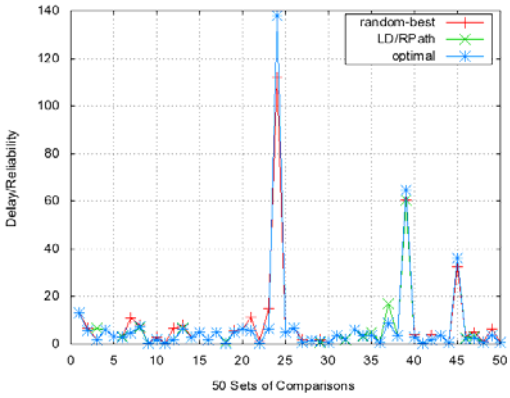


Figure 11: Comparison of Delay/Reliability

Network Size	Smaller Delay (%)	Higher Reliability (%)
30	82	80
60	74	74
100	82	70

Table 1: Comparison at Different Size

According to the simulation results from efficiency and effectiveness comparison, both *LDPath* and *LD/RPath* have a good performance. *LDPath* has a much better performance than *random-best* in large-scale network environments with low overhead to the system, while *optimal* can't satisfy real-time requirement. *LD/RPath* achieves as better performance as *optimal* when network size is small. In all, both of

them have better performance than *random* and can meet different network topology size requirements.

7 Conclusion

Effective real-time multimedia delivery in a pervasive computing environment is an important research issue. How to generate a low-delay and high-reliability path is NP-Complete. In this paper, we propose two algorithms for different environments. *LDPath* chooses service replica hop-by-hop which is adapted to a large-scale open environment. *LD/RPath* is centralized and generates a delivery path at the media source node based on the trade-off between delay and reliability, which is adapted to a small-scale environment. And to the best of our knowledge, it is the first time considering the changes of data size during transmission, which results in different delays on the same links or nodes. Numerical results show that both of *LDPath* and *LD/RPath* have good performance with relatively low complexity. We plan to focus on load balancing and stability issues of multimedia delivery in pervasive environments in future work.

Acknowledgment

This paper is partly supported by the National Natural Science Foundation of China under Grant No. 61073028, 61021062; the National Basic Research Program of China (973) under Grant No. 2009CB320705; and Jiangsu Natural Science Foundation under Grant No. BE2010179.

References

[Benatallah, 02] Benatallah, B., Dumas, M., Sheng, Q. Z., Ngu. A.: Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services, Proceedings of ICDE'02, IEEE Computer Society, pp. 297-308, San Jose, 2002.

[Berardi, 05] Berardi, D., Calvanese, D., De Giacomo, G., Hull, R., Mecella, M.: Automatic composition of transition-based semantic web services with messaging. Proceedings of the 31st international Conference on Very Large Data Bases, Trondheim, Norway, 2005.

[Bernardos, 10] Bernardos, C.J., Gramaglia, M., Contreras, L.M., Calderon, M., Soto, I.: Network-based Localized IP mobility Management: Proxy Mobile IPv6 and Current Trends in Standardization, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), Vol. 1, No. 2/3, 2010

[Bultan, 03] Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: a new approach to design and analysis of e-service composition. Proceedings of the 12th international Conference on World Wide Web Budapest, Hungary, May 2003.

[Choi, 03] Choi, S., Turner, J. Wolf, T.: Configuring sessions in programmable networks, Computer Networks, Volume 41, Issue 2, 5 February 2003.

[Cormen, 96] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, 16th ed. MIT press/McGraw-Hill, Cambridge/New York, 1996.

[Gu, 04] Gu, X., Nahrstedt, K., Yu, B.: SpiderNet: An intergrated peer-to-peer service composition framework, Proceedings of IEEE International Symposium on High-Performance Distributed Computing (HPDC-13). IEEE, June 2004.

[Gu, 06] Gu X., Nahrstedt K., Distributed multimedia service composition with statistical QoS assurances, IEEE Transactions on Multimedia, Vol. 8(1), 2006.

[Kafle, 10] Kafle, V.P., Inoue, M.: Locator ID Separation for Mobility Management in the New Generation Network, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), Vol. 1, No. 2/3, 2010.

[Kalasapur, 07] Kalasapur, S., Kumar, M. Shirazi, B.A.: Dynamic service composition in Pervasive Computing, IEEE Transactions on Parallel and Distributed Systems, vol. 18, 2007.

[Nahrstedt, 05] Nahrstedt K., Balke, W. T.: Towards building large scale multimedia systems and applications: Challenges and status, Proceedings of the first ACM international workshop on multimedia service composition, 2005.

[OWL-S]"OWL-S", http://www.w3.org/Submission/OWL-S/.

[Qian, 07] Qian, Z., Lu, S., Xie, L.: Colored Petri Net Based Automatic Service Composition, Proceedings of The 2nd IEEE Asia-Pacific Service Computing Conference (APSCC 2007), 2007.

[Qian, 09]Qian, Z., Guo, M., Zhang S. Lu, S.: Service-Oriented Multimedia Delivery in Pervasive Space, Proceedings of IEEE Wireless Communications and Networking Conference(WCNC 2009), Budapest, Hungray, April, 2009.

[Raman, 02] Raman B., Katz, R. H.: Emulation-based Evaluation of an Architecture for Wide-Area Service Composition, in SPECTS, Jul 2002.

[Raman, 03] Raman B., katz, R. H.: Load balancing and stability issues in Algorithms for Service Composition, Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03). IEEE, April 2003, pp. 1477-1487.

[Wang, 96] Wang Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications, Proceedings of IEEE Journal on Selected Area in Communications; Vol. 14, 1996.

[Xue, 07] Xue, G., Zhang, W.: Multiconstrained QoS Routing: Greedy is Good, Proceedings of Global Telecommunications Conference, GLOBECOM, Nov. 2007.

[Zeng, 03] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q. Z.: Quality Driven Web Services Composition, Proceedings of the 12th International World Wide Web Conference, May 2003.

[Zhang, 09] Zhang, S., Qian Z., Guo M., Lu, S.: An Efficient Algorithm for Multimedia Delivery in Pervasive Space, Proceedings of Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium, Aug. 2009.