Cascading-Failure Tolerance for Language Service Networks

Kemas M. Lhaksmana, Toru Ishida and Yohei Murakami

Abstract One of the main features of The Language Grid is its support for service composition, i.e. creating new language services that meet user requirements by combining the existing ones. Despite the potential of service composition, such a service-oriented computing (SOC) application may experience cascading failure when a disruption on one or more component services is propagated to the composite services that combine them. As the number of language services grows, composite language services will become more common, and thus understanding cascading failure among language services becomes more important. This chapter investigates how failure may propagate among language services and how to improve language service tolerance to cascading failure. To this end, the dependency between language services is modeled as service network on which cascading failure is simulated and analyzed. We also generated service networks in scale-free, exponential, and random topology to analyze how cascading failure occurs in different topology. The simulation reveals that service networks with scale-free topology have better cascading-failure tolerance compares to that of other topology.

Keywords Cascading failure · Service network · Scale-free network

K. M. Lhaksmana (✉)
School of Computing, Telkom University, Jl. Telekomunikasi No. 1,
Bandung 40257, Indonesia
e-mail: kemasmuslim@telkomuniversity.ac.id

T. Ishida
Department of Social Informatics, Kyoto University,
Kyoto 606-8501, Japan
e-mail: ishida@i.kyoto-u.ac.jp

Y. Murakami
Unit of Design, Kyoto University, 91 Chudoji Awata-cho,
Kyoto 600-8815, Japan
e-mail: yohei@i.kyoto-u.ac.jp

Y. Murakami et al. (eds.), *Services Computing for Language Resources*,
Cognitive Technologies, https://doi.org/10.1007/978-981-10-7793-7_6

1 Introduction

The Language Grid was designed as service-oriented infrastructure for sharing language resources as services and combining the language services to create new services by using service composition [10]. Despite the advantages of service composition, bindings between composite services and their component services introduce dependency that may cause cascading failure issue.

To address cascading failure in The Language Grid, Fig. 1 illustrates an example of service composition. Suppose The Language Grid provides a composite service type "Translation Combined with Bilingual Dictionary" that combines "Morphological Analyzer", "Bilingual Dictionary", and "Translation" service types. Each service type can be realized by one or more service instances (hereafter "services" for brevity). For example, Mecab and TreeTagger are services of "Morphological Analyzer" service type. We call these two services as substitutable services since each of them can perform generally the same functions, and thus can substitute each other.

In the composite service "Translation Combined with Bilingual Dictionary", cascading failure may occur when one of the component services fails and no substitutable service is available. Therefore, language service infrastructure, such as The Language Grid, should maintain sufficient number of substitutable services such that cascading failure can be minimized. Since The Language Grid currently has sufficient substitutable services for each service type, i.e. roughly 11 substitutable services for each component service in average, cascading failure is less likely to

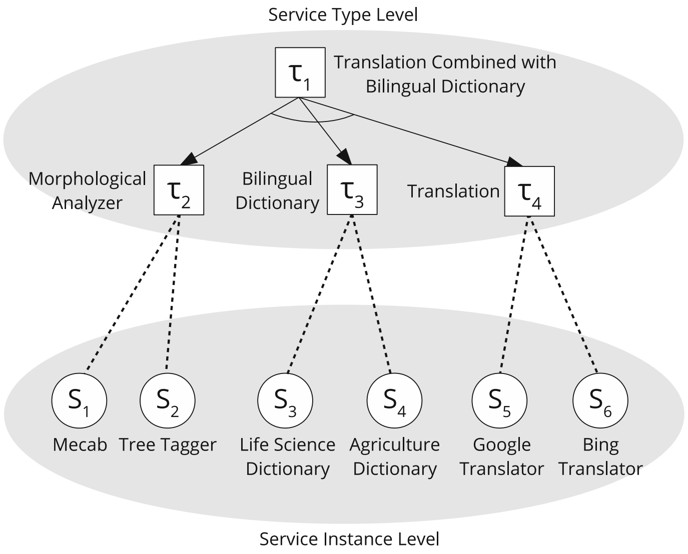

Fig. 1 Service composition in The Language Grid is modeled into two abstraction level

occur. However, in the near future The Language Grid is expected to be populated with more services and service types due to the Open Language Grid initiative that was designed to allow user groups to establish their own Language Grid servers. In addition, users may create composite services out of their own services or combining the services from other servers [11]. Therefore, investigating cascading failure among language services is still of importance.

To investigate cascading failure among language services, the dependency between the services is modeled as service network whose nodes represent services and links represent the dependency between them. On these service networks, cascading failure is simulated by randomly selecting a node to experience failure which then may trigger subsequent failures on the other dependent nodes. We observe the failure propagation in service networks of different topology. The simulation reveals that the number of failed nodes decay exponentially over the number of substitutable services. This result suggests that the existence of several substitutable services significantly improves the tolerance to cascading failure. As for the effect of network topology to cascading-failure tolerance, the simulation result shows service networks with scale-free topology have better tolerance compared to the other topologies. This is contrast to cascading failure in power networks, where scale-free topology shows poor tolerance [5].

2 Cascading Failure

Cascading failure possibly occurs in interdependent systems where failure of one system component disrupts the other dependent components. One of the well-known cascading failure in the Internet industry is Amazon EC2 outage that caused disorder to some major Internet companies which used their services [1]. In the study of critical infrastructure, cascading failure is defined as a disruption in one infrastructure that causes disruption in another [19]. Cascading failure has gained more interest since major cascading failure events [16] and power grid failures affecting large areas occurred in the past, such as those in Italy [20], North America [13], Australia, and New Zealand [2]. Major cascading failure events have also occurred in the past, such as fires in the aftermath of earthquake in San Francisco (1906) and Kobe (1995), and the disruption on some public services after a communication satellite orbital shift (May 1998) [16].

Even though much cascading failure research has been done in power network and critical infrastructure domains, it has not been much analyzed in SOC. In service networks, cascading failure happens when there is a failure in a node (link) and it is spreading into one or more than one nodes (links). Cascading failure is classified into two types: vertical and horizontal cascading failure [14]. The former happens when the failure follows a path along the links to either direction or to both direction, which is determined by the dependency among the neighboring cascades. As for the latter, the failure spreads to the other nodes (links) which are not necessary adjacent, such as that of in power network where the failure cascades due to load redistribution [5].

3 Language Service Network Model

We present the model of language service network (hereafter "service network" for brevity) as a directed network where a node represents a language service and a link represents dependency of one language service on the other. Next, we address the properties of a service network that are relevant to its tolerance against cascading failure. Finally, we address The Language Grid service network to support our analysis on cascading failure propagation and tolerance. The formal definition of a service network is given as follows:

- A service network is defined as $N = (S, E)$.
- $S = A \cup C$ is a set of services that consists of atomic services A and composite services C.
- $E = L \cup R$ is a set of links.
- L is the set of dependency links in the service network. L_i is the set of dependency links of a composite service s_i.
- R is the set of alternate links in the service network. $R_{i,j}$ is the set of alternate links of a dependency link $l_{i,j} \in L_i$ (see Fig. 2).

In addition to the definition above, to quantify the degree of interdependency between the services, the following measures are provided:

- Let service degree of dependency $dep_i = |L_i|$ be the number of services that may be executed at runtime by a composite service s_i.
- Let dependency link degree of alternative $alt_{i,j} = |R_{i,j}| - 1$ be the number of substitutable services that can substitute a failed service that associated to the same dependency link $l_{i,j}$.

An example of a service network is illustrated in Fig. 2, where a service is represented as a circle, and the dependency of a composite service on a component service

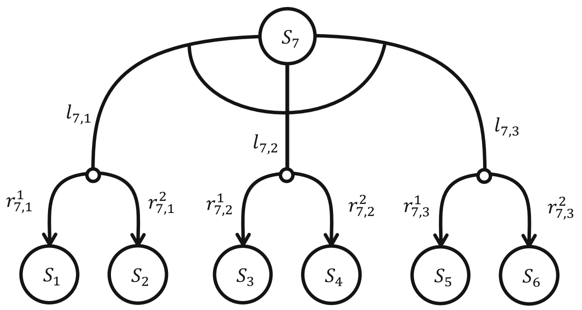

Fig. 2 The service network representation of the composite service "Translation Combined with Bilingual Dictionary" (Fig. 1). Each service is represented as a circle, while the dependency between two services is illustrated by an arrow. An arc connecting two or more links indicates that the service depends on all of the services pointed by the links, whereas a small circle indicates that the service requires either service connected by the links

is represented as an outgoing link originated from the composite service. The figure represents the dependency of the composite service “Translation Combined with Bilingual Dictionary” on its component services. In the service network, the composite service “Translation Combined with Bilingual Dictionary” is represented by the node s_7. The set of dependency links of s_7 is $L_7 = \{l_{7,1}, l_{7,2}, l_{7,3}\}$. Therefore, the degree of dependency of s_7 is $dep_7 = |L_7| = 3$.

The existence of a substitutable service is illustrated by splitting a dependency link into two or more alternate links. For example, the set of alternate links on $l_{7,1}$ is $R_{7,1} = \{r^1_{7,1}, r^2_{7,1}\}$. The degree of alternative of $l_{7,1}$ is $alt_{7,1} = |R_{7,1}| - 1 = 1$. Likewise, $l_{7,2}$ and $l_{7,3}$ degree of alternative is also 1. The existence of substitutable services, as well as alternate links, is useful to prevent cascading failure. Suppose service s_3 (Life Science Dictionary) is unexecutable, then the composite service s_7 is still able to perform its functions because there is another service, which is s_4 (Agriculture Dictionary), that can be invoked to perform the same functions.

3.1 Network Topology

We present some relevant properties that contribute to cascading-failure tolerance, which are network topology, the degree of interdependency between services, the depth of composite services, and the proportion of composite services in the network.

Some related work has proved that network topology influences tolerance to cascading failure [5]. For complex and large-scale networks, their topology is classified to their degree distribution [6]. Directed networks have in-degree distribution and out-degree distribution, which can be acquired by calculating the distribution of incoming and outgoing links. In service network, the out-degree of a service represents the number of component services that the service depends to, while the in-degree is the number of composite services that require this service as one of its components.

In this work, cascading failure is observed in service networks of three kinds of network topology: scale-free, exponential, and random network topology. The first two are the topology of growing networks, i.e. the type of networks on which the number of nodes and links increases over time. Scale-free network gains interests from researchers because of its high tolerance against random failure [3]. This network grows due to preferential attachment. Every time a node is added to the network, new connections are made from the newly added node to the existing ones. However, existing nodes with higher in-degree is always preferable to be chosen than those with lower in-degree. Scale-free networks are indicated with degree distribution that follows power-law as follows

$$P(k) \sim k^{-\gamma} \tag{1}$$

In Eq. 1, $P(k)$ is the probability to find a node having k connections, while γ is a constant within $2 < \gamma < 3$. Unlike scale-free networks on which the nodes are

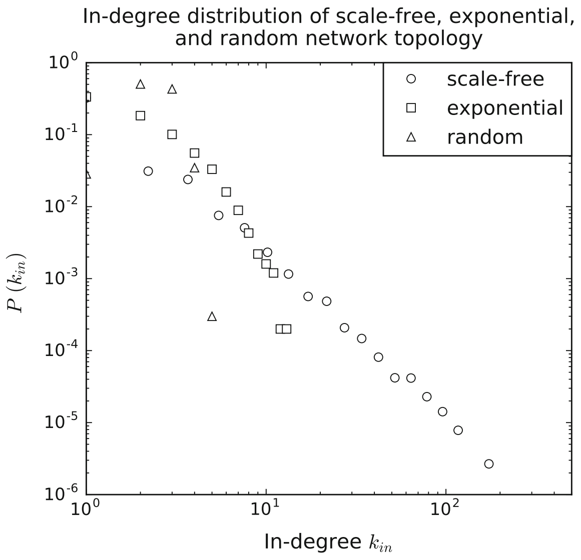

Fig. 3 In-degree distribution of scale-free, exponential, and random service network topology. The in-degree distribution of these topologies follow power-law, exponential, and Poisson distribution, respectively [15]

connected by preferential attachment, new nodes in exponential networks randomly choose existing nodes to connect. Due to this random connections, the network will exhibit exponential degree distribution [3].

Another network topology that has been long studied is random network. The number of nodes in a random network is considered to be fixed, while the connections are made in random manner between these existing nodes [7]. In this way, the network will exhibit Poisson degree distribution.

Scale-free and exponential networks are discussed since most real networks are growing networks where the number of nodes increases over time. Random network is also observed in this chapter since this type of network has been widely studied and represents the kind of service networks with fixed number of nodes. The in-degree distribution of scale-free, exponential, and random service network is illustrated in Fig. 3.

3.2 Degree of Interdependency Between Services

To measure the level of service interdependency in a service network, we provide several metrics. First, the *network degree of dependency* $\langle dep \rangle$ measures the number of required services for all composite services in the network. This is a global variable, whereas the *composite service degree of dependency* dep_i, which has been addressed earlier, applies only to a particular service $s_i \in C$. The formal definition of this measure is as follows [14].

$$\langle dep \rangle = \frac{1}{|C|} \sum\nolimits_i dep_i \tag{2}$$

Second, we also define the network degree of alternative $\langle alt \rangle$ to quantify the number of substitutable services in the network. Likewise, $\langle alt \rangle$ is also a global variable, whereas $alt_{i,j}$ represents the degree of alternative for the particular dependency link $l_{i,j}$. This measure is defined as [14]

$$\langle alt \rangle = \frac{1}{|L|} \sum\nolimits_{i,j} alt_{i,j} \tag{3}$$

In addition to the aforementioned measures, another metric is also considered since it affects a service network tolerance to cascading failure [15]. The measure is the depth of service composition, which is considered because service composition can be nested. The deeper the composition, the more possible a composite service to experience cascading failure after random failure of one of its component services. The depth of a service s_i is defined as [14]

$$depth_i = \begin{cases} \frac{1}{|S_i|} \sum\limits_{j=0}^{|S_i|-1} [depth_j + 1], & s_i \text{ is a composite service} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

while the depth of service composition for the whole network is defined as [15]

$$\langle depth \rangle = \frac{1}{|C|} \sum\nolimits_i depth_i \tag{5}$$

In this chapter, we only consider the first two measures: $\langle dep \rangle$ and $\langle alt \rangle$. Even though the other measure also affects service network tolerance to cascading failure, it can be ignored in the current service network of The Language Grid. We found that The Language Grid service composition is shallow at $\langle depth \rangle = 1.29$ [15]. This shows that most composite services in The Language Grid are not nested.

4 The Language Grid Service Network

The Language Grid is a typical example of a service network, where users can add new services and combine existing services by means of service composition [10]. In The Language Grid, service types classify services based on their functionalities. Services belong to the same service type if they share the same functionalities A composite service type is a combination of different service types. An example of a composite service type has been illustrated in Fig. 1.

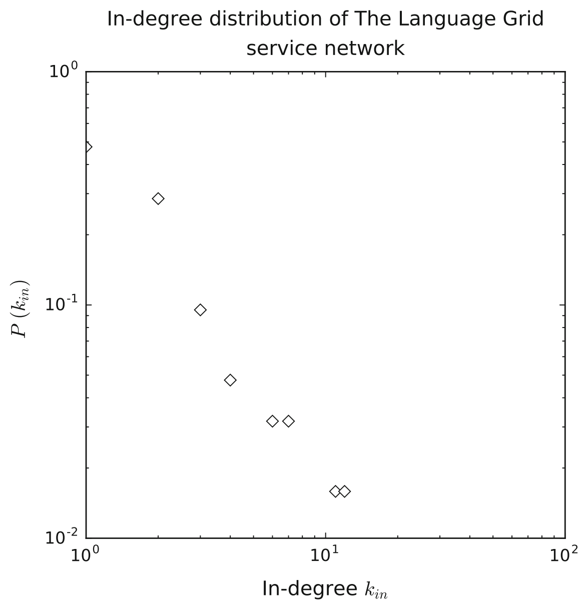

Fig. 4 In-degree distribution of The Language Grid service networks

The Language Grid operation centers have been established in several countries, including Japan (Kyoto), Thailand (Bangkok), Indonesia (Jakarta), and China (Urumqi), which in total provide 188 services as of October 2014 when the experiment was conducted. The architecture of The Language Grid allows one to combine services across different operation centers [17]. Here, we limit our analysis on 117 atomic services and 21 composite services provided by Kyoto Language Grid operation center. Even though the proportion of composite services in The Language Grid (15%) is much less than the proportion of atomic service, they are frequently invoked, which is 47% of all invocations. This indicates that the composite services are vital for The Language Grid users. These composite services mostly consist of 2–3 atomic services at $\langle dep \rangle = 2.81$. The Language Grid has many substitutable services at $\langle alt \rangle = 10.92$ [14]. The high $\langle alt \rangle$ gives The Language Grid high tolerance to cascading failure, as it will be explained in Sect. 5.2.1.

Even though the number of services in The Language Grid is expected to grow, the current size of the network is rather small. Therefore, the network topology of The Language Grid still cannot be classified by its in-degree distribution as it shows neither power-law nor exponential distribution (Fig. 4). However, larger service networks generally exhibit scale-free topology [8, 9, 12, 18]. As the other scale-free networks on which the connections are made in preferential manner [3], composite services in the Language Grid also tend to combine widely used and popular services, and thus would become scale-free [15].

5 Cascading Failure Simulation

To perform cascading failure simulation, some service networks are generated with different topology and different degree of interdependency. The first part provides the algorithms to generate these networks, while the second part addresses the way to simulate cascading failure and the analysis of the simulation result.

5.1 Generating Service Networks

In this Section, we generate artificial service networks, i.e. the networks that are computationally created from scratch instead of generated from existing service networks. We generate some service networks with different degree of interdependency and different topologies, which are scale-free, exponential, and random networks. Analyzing these three topologies help us to understand which one has better tolerance towards cascading failure.

5.1.1 Scale-Free, Exponential, and Random Service Network

Both scale-free network and exponential network are a part of growing network, which is why the same algorithm is used to generate them. What make these two are different is the way they choose existing nodes for making connections. Scale-free networks use preferential attachment to choose the service to connect to, while exponential networks choose randomly.

Our algorithm uses the preferential attachment probability function as follows [4]

$$\Pi(k_i^{in}, \alpha) = \frac{k_i^{in} + \alpha}{\Sigma_j (k_j^{in} + \alpha)} \tag{6}$$

where k_i^{in} is the service s_i in-degree, j is the index for all services, and α is the initial attractiveness parameter. According to this function, the higher the in-degree of a service, the more possible it is to be chosen. The initial attractiveness α gives value to the services with zero in-degree, such as those which are newly added to the network.

The algorithm to generate scale-free and exponential service networks uses the following parameters [15]:

- n_0 is the number of isolated services at the beginning of the simulation t_0
- $\Delta t \geq 0$ is the duration of the network generation
- $c \in [0, 1]$ is the expected proportion of composite services in the network
- δ is the expected degree of dependency $\langle dep \rangle$ such that $\delta \approx \langle dep \rangle$
- λ is the expected degree of alternative $\langle alt \rangle$ such that $\lambda \approx \langle alt \rangle$
- α is the initial attractiveness parameter

The algorithm begins at t_0 when n_0 service nodes are added to the network without making connections between them. The next steps are to add some more nodes and to make connections from the newly added nodes to the existing ones. The following steps will be performed iteratively at each timestep from $t = t_1$ until $t = t_0 + \Delta t$ [15]:

1. Create a service s_i.
2. Under probability c, generate dep_i, a random number within $[1, \delta \times 2 - 1]$.
3. If dep_i is generated in step (2)
   a. Create the set of dependency links $L_i = \{l_{i,0}, ..., l_{i,dep_i - 1}\}$.
   b. For each dependency link $l_{i,j} \in L_i$, generate a random number $alt_{i,j}$ within $[0, \lambda \times 2]$.
   c. If $alt_{i,j} = 0$
      i. Choose a service $s_p \in S \mid s_p \neq s_i$.
      ii. Connect the dependency link $l_{i,j}$ from s_i to s_p.
   d. Else
      i. Choose $alt_{i,j} + 1$ services.
      ii. Split the dependency link $l_{i,j}$ into $alt_{i,j} + 1$ alternate links.
      iii. Connect these alternate links to the chosen services.

The connections will only be made between services that are currently not connected and will not be connected to themselves, i.e. the network restricts multiple links and self links. In the algorithm, the first step is to initialize a service network with some number of atomic services, and then add one service at a time. The number of composite services is determined by the probability c (step 2). A newly added service becomes an atomic service if no dependency link created for the service, but it will become composite service if one or more dependency links are added to the service. In making connections (step 3.c and 3.d) the way to choose the existing nodes depends on the network topology to be created. If the network is expected to be scale-free, then preferential attachment (Eq. 6) is applied. Otherwise, the nodes are chosen randomly such that the network will grow as exponential network.

The same algorithm to generate exponential service network is used when we want to generate random service network, but it does not need node addition. Thus, random service network algorithm uses all the parameters except the number of duration of the network generation Δt and the number of nodes is fixed so that $n_0 = |S|$. The algorithm to generate random service network is as follows [15]:

1. Populate the network with n_0, which is also the expected size of the network.
2. For each $s_i \in S$, do the steps 2 and 3 in the algorithm as generating exponential service networks.

5.1.2 The Language Grid Service Network

Apart from generating service networks with different topology, we also generate The Language Grid service network to observe cascading failure in a real language service network. As illustrated in Fig. 1, The Language Grid services are classified into service types according to the functionalities they provide. The Language Grid service network is generated by representing service instances as atomic service nodes, and service types as composite service nodes. The links from composite service nodes to their component service nodes are created based on the service types they combined. If a combined service type is realized by only one service instance, a dependency link is created connecting the composite service node to the atomic service node. If there are more than one service instance that can provide the functionality for the service type, alternate links are created to connect the composite service node to these atomic service nodes. Nested composition is created when a composite service types combines one or more composite service types. The service network representation of a composite service in The Language Grid, including its component services, has been illustrated in Fig. 2.

5.2 *Simulation Result and Analysis*

In this simulation, we use different types of topology to create service networks, each of which has different $\langle dep \rangle$ and $\langle alt \rangle$, using the algorithm explained in Sect. 5.1. Therefore, according to their topology, the simulation generate three typical types of service network (scale-free, exponential, and random) and The Language Grid service network that is simulated from real world service network.

Scale-free and exponential service networks are created with a few initial services, i.e. $n_0 = 10$. Then, for the duration of $\Delta t = 10{,}000$, adding one service at a time to make the network grows until it reaches the size of 10,010 nodes. The scale-free networks are generated with the initial attractiveness parameter α is set at 1. For random service networks, it is initialized with fixed number of services $n_0 = 10{,}010$. For all these three kinds of topology, the composite service proportion parameter is $c = 0.4$ so that 40% of the nodes are composite services, whereas 60% are atomic services. Different $\langle dep \rangle$ and $\langle alt \rangle$ are also implemented to these networks by setting different values for δ and γ parameters. As for The Language Grid service network, as we have explained in Sect. 5.1.2, it is generated based on its existing composite services.

Briefly, the simulation works as follow. First, a service network is generated for the expected network topology, $\langle dep \rangle$, and $\langle alt \rangle$ according to the algorithm in Sect. 5.1. After the service network is created, the following algorithm is executed.

While there is an active service in the network, do the following:

1. An active service is chosen randomly to be deactivated to simulate random failure. This service is called random failed service.
2. If the random failed service is a component service, i.e. it participates in one or more service compositions, deactivate the composite services when there is no substitutable service. The deactivation is performed recursively all the way to the outermost composite services.

To analyze how cascading failure occurs in service networks, we observe the number of cascade failed services n_c over the number of random failed services n_r. The value n_c is acquired by counting the failed composite services in step 2. For each iteration above, one active service is chosen randomly to experience random failure (step 1). Therefore, n_r is actually the number of iteration. The last iteration is performed when the network is collapse, which is when $n_r + n_c = |S|$. The analysis of the simulation results are addressed next.

5.2.1 Network Topology and Cascading-Failure Tolerance

To analyze the effect of network topology on the cascading failure, we observe the number of cascade failed services over the number of random failed services (Fig. 5). The value $\langle dep \rangle$ is set at its lowest possible value 1, while the value $\langle alt \rangle$ varies at $0, 0.5$, and 1 on Fig. 5a, b, c, respectively. From the figures, networks with scale-free topology have a better tolerance compared to exponential networks because

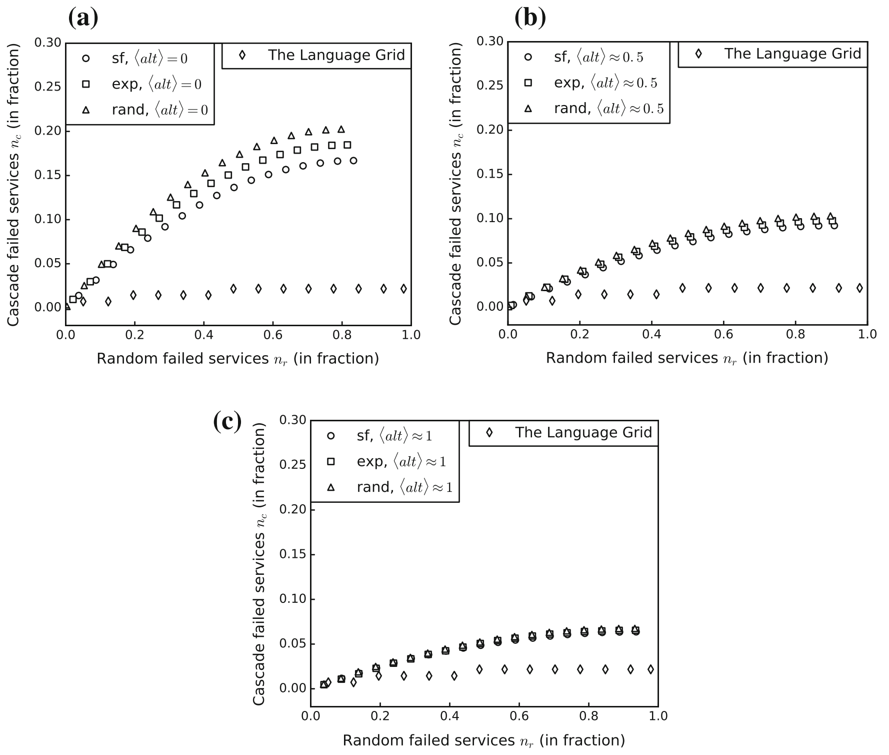

Fig. 5 The number of cascade failed services over the number of random failed services on different topology and different degree of interdependency

the fraction of high in-degree nodes in scale-free network is much less than that in an exponential network. This is similar to when we compare exponential network to random network, in that exponential network has better tolerance than random network because the fraction of high-degree nodes in exponential network is much less than the ones in random network. In addition, the graphs also show that the effect of network topology on the tolerance to cascading failure getting less significant as $\langle alt \rangle$ increases, because the more substitutable services in the network, the higher tolerance to cascading failure. The existence of substitutable services also explains the high tolerance of The Language Grid on cascading failure. With $\langle alt \rangle = 10.92$, the number of cascade failed services in The Language Grid is very low.

5.2.2 Degree of Dependency and Cascading-Failure Tolerance

To analyze the effect of $\langle dep \rangle$ on cascading failure, Fig. 6a provides the number of cascade failed services over $\langle dep \rangle$ for service networks at $\langle alt \rangle = 0$ and $n_r = 0.2$. Cascade failed services are frequent in service networks with higher $\langle dep \rangle$. This is because the interdependency between the services is stronger.

The graph also shows linear relationship between the number of cascade failed services and $\langle dep \rangle$. The shallow depth of service composition in the network is the one which is responsible for this linear relationship. So from the result above, we can decrease the number of required component services from each composite service in order to increase the cascading-failure tolerance. Unfortunately, $\langle dep \rangle$ cannot always be predefined since the number of component services in service composition is usually determined by the problem domain. Therefore, it is important to consider improving $\langle alt \rangle$ to achieve an expected degree of tolerance instead of restricting the number component services.

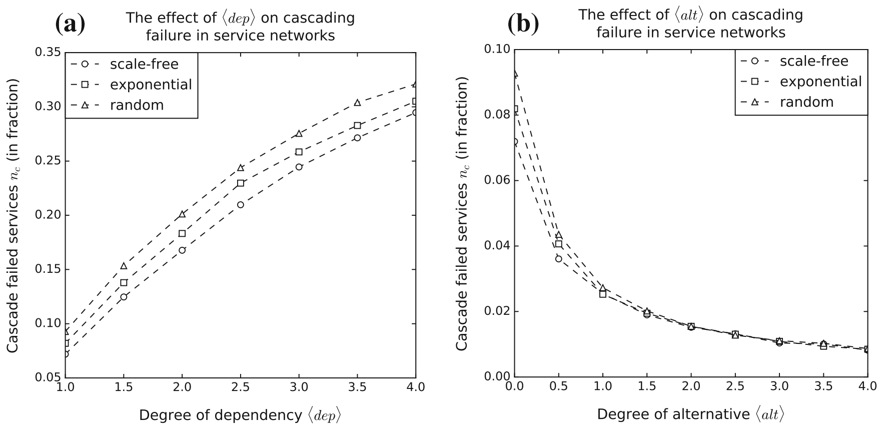

Fig. 6 The number of cascade failed services over the degree of dependency (**a**) and the degree of alternative (**b**) [15]

5.2.3 Degree of Alternative and Cascading-Failure Tolerance

To observe the effect of $\langle alt \rangle$ to service network tolerance on cascading failure, Fig. 6b illustrates the graphs of the number of cascade failed services over $\langle alt \rangle$. In these service networks, $\langle dep \rangle$ is set to 1, which is the lowest possible value to vary $\langle alt \rangle$. From the result, we can increase the network tolerance towards cascading failure by increasing $\langle alt \rangle$. However, the improvement of the tolerance to cascading failure will be significant only when the $\langle alt \rangle$ is currently low at $\langle alt \rangle < 2$. On higher $\langle alt \rangle$, the improvement of the tolerance becomes less significant. When having many substitutable services is considered costly, the simulation result suggests that having several substitutable services (e.g. 2 or 3) would be sufficient to have a good tolerance.

6 Conclusion

The purpose of this study is to analyze the effect of interdependency between services and network topology to cascading failure tolerance in language service networks. To this end, we executed a random failure simulation on scale-free, exponential, and random service networks with different degree of interdependency to observe cascading failure tolerance. In addition, the simulation was also performed on The Language Grid service networks.

We conclude some important findings in this chapter as follows [15]:

1. The effect of network topology on tolerance is more significant on lower degree of alternative, i.e. where the average number of substitutable services for each required component service is low.
2. The number of cascade failed services, i.e. the nodes experiencing cascading failure, is inversely proportional to the degree of alternative.
3. The number of cascade failed services is somewhat linear to the degree of dependency, i.e. the average number of component services.
4. Scale-free topology has better tolerance to cascading failure, followed by exponential and random topology. This statement is in contradiction with the load-based cascading failure tolerance in power networks which stated that random topology has better tolerance than scale-free [5].

Acknowledgements This research was supported by the Grant-in-Aid for Scientific Research (S) (24220002, 2012–2016) from Japan Society for the Promotion of Science (JSPS).

References

1. Armbrust, M., Fox, O., Griffith, R., Joseph, A.D., Katz, Y., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Technical Report, University of California at Berkeley (2009)

2. Ash, J., Newth, D.: Optimizing complex networks for resilience against cascading failure. Phys. A Stat. Mech. Appl. **380**, 673–683 (2007)
3. Barabási, A.L., Albert, R., Jeong, H.: Mean-field theory for scale-free random networks. Phys. A Stat. Mech. Appl. **272**(1), 173–187 (1999)
4. Chen, Q., Shi, D.: The modeling of scale-free networks. Phys. A Stat. Mech. Appl. **335**(1), 240–248 (2004)
5. Crucitti, P., Latora, V., Marchiori, M.: Model for cascading failures in complex networks. Phys. Rev. E **69**(4), 045104 (2004)
6. Dorogovtsev, S.N., Mendes, J.F.: Evolution of networks. Adv. Phys. **51**(4), 1079–1187 (2002)
7. Erdős, P., Rényi, A.: On the evolution of random graphs. Magyar Tud. Akad. Mat. Kutató Int. Közl **5**, 17–61 (1960)
8. Feng, Z., Lan, B., Zhang, Z., Chen, S.: A study of semantic web services network. Comput. J. **58**, 1293–1305 (2014)
9. Huang, K., Fan, Y., Tan, W.: An empirical study of programmable web: a network analysis on a service-mashup system. In: 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 552–559 (2012)
10. Ishida, T. (ed.): The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability. Springer Science & Business Media (2011)
11. Ishida, T., Murakami, Y., Lin, D., Nakaguchi, T., Otani, M.: Open Language Grid: towards a global language service infrastructure. In: The Third ASE International Conference on Social Informatics (SocialInformatics 2014), Cambridge, Massachusetts, USA (2014)
12. Kil, H., Oh, S.C., Elmacioglu, E., Nam, W., Lee, D.: Graph theoretic topological analysis of web service networks. World Wide Web **12**(3), 321–343 (2009)
13. Kinney, R., Crucitti, P., Albert, R., Latora, V.: Modeling cascading failures in the North American power grid. Eur. Phys. J. B Condens. Matter Complex Syst. **46**(1), 101–107 (2005)
14. Lhaksmana, K.M., Murakami, Y., Ishida, T.: Cascading failure tolerance in large-scale service networks. In: 2015 IEEE International Conference on Services Computing (SCC), pp. 1–8 (2015)
15. Lhaksmana, K.M., Murakami, Y., Ishida, T.: Analysis of large-scale service network tolerance to cascading failure. IEEE Internet Things J. **3**(6), 1159–1170 (2016)
16. Little, R.G.: Controlling cascading failure: understanding the vulnerabilities of interconnected infrastructures. J. Urban Technol. **9**(1), 109–123 (2002)
17. Murakami, Y., Tanaka, M., Lin, D., Ishida, T.: Service grid federation architecture for heterogeneous domains. In: 2012 IEEE Ninth International Conference on Services Computing (SCC), pp. 539–546 (2012)
18. Oh, S.C., Lee, D., Kumara, S.R.: Effective web service composition in diverse and large-scale service networks. IEEE Trans. Serv. Comput. **1**(1), 15–32 (2008)
19. Rinaldi, S., Peerenboom, J., Kelly, T.: Identifying, understanding, and analyzing critical infrastructure interdependencies. IEEE Control Syst. **21**(6), 11–25 (2001)
20. Rosato, V., Issacharoff, L., Tiriticco, F., Meloni, S., Porcellinis, S., Setola, R.: Modelling interdependent infrastructures using interacting dynamical models. Int. J. Crit. Infrastruct. **4**(1), 63–79 (2008)