2015 IEEE International Conference on Services Computing

Cascading Failure Tolerance in Large-Scale Service Networks

Kemas M. Lhaksmana
Department of Social Informatics
Kyoto University
Kyoto, Japan
kemas.muslim@ai.soc.i.kyoto-u.ac.jp

Yohei Murakami
Unit of Design
Kyoto University
Kyoto, Japan
yohei.murakami@design.kyoto-u.ac.jp

Toru Ishida
Department of Social Informatics
Kyoto University
Kyoto, Japan
ishida@i.kyoto-u.ac.jp

***Abstract*—The rapid growth of services and the Internet of Things vision lead to the future of Internet in which a massive number of services are available and connected to each other. In such service network, dependency between services potentially causes cascading failure, where the failure of one service can cause the failure of dependent services. Cascading failure tolerance is determined by the topology of the network and the degree of service interdependency. As to the former, we analyze cascading failure in scale-free, exponential, and random service networks. We find that scale-free topology has generally the highest tolerance. This is contrast to cascading failure in power network, where random topology provides better tolerance. For the latter, we find that the number of cascade failed nodes increases as the inverse of the average number of alternate services, e.g. functionally equivalent services. This suggests that increasing the number of alternate services can significantly improve the network tolerance if each service only has few alternate services available.**

***Keywords*-cascading failure; service network; scale-free network**

I. Introduction

In the near future, the Internet is expected to include a massive number of smart devices that will be accessible as services from all over the web [1]. The rapid growth of services is driven by the adoption of service-oriented computing (SOC) into the Internet of Things, such as the effort to provide sensors as services [2], and to combine smart device services as in service composition [3]. These services will interact with each other to form a large-scale service network. Service composition creates dependency between composite services and their component services. In this context, cascading failure occurs when one or more component services are unexecutable, and the composite service thus becomes unexecutable.

The tolerance of service network to cascading failure is determined by the topology of the network and the degree of interdependency between the services. Despite the lack of research addressing cascading failure in service network, scale-free networks are widely accepted having a topology that is robust against random failure [4]. However, this finding does not take cascading failure into account. On the other hand, in the case of cascading failure in power networks, random networks offer better tolerance than scale-free networks [5].

A strongly interdependent service network has indeed less tolerance against cascading failure. As the services are more dependent on each other, the failure of one service is more likely to cascade to more services. Reducing the interdependency of a service network can be achieved by reducing the number of component services in each service composition. However, since the selection of component services are determined by the problem domain and defined during the design of service composition, reducing the degree of dependency may not be possible without redesign the composition. Another way to reduce the interdependency is by increasing the number of alternate services, e.g. functionally equivalent services, that can replace component services by providing the same functionality. However, increasing the number of alternate services in each service composition can be costly. Therefore, it is important to know how many alternate services should be added to the service network to significantly improve its tolerance against cascading failure.

The vision that the future Internet will be populated with large-scale service networks and the lack of research on service network cascading failure motivate this research to investigate which topology provides better tolerance against cascading failure, and analyze how the interdependency between services contribute to the tolerance. The contributions of this research are some important findings as follows: (i) scale-free topology provides better tolerance, followed by exponential and random topology; (ii) the effect of network topology on tolerance is more significant when the network has less number of alternate services (degree of alternative); (iii) the number of nodes that fail cascade-wise is inversely proportional to the degree of alternative, and (iv) somewhat linear to the average number of required component services (degree of dependency). These findings are important for designing a service network and service composition.

The rest of this paper is organized as follows. First, a definition of cascading failure and some related work are provided in Section II. Next, the properties of service networks and two publicly accessible service networks are presented in Section III. We explain how the service network is modeled and generated in Section IV. The simulation

978-1-4673-7281-7/15 $31.00 © 2015 IEEE
DOI 10.1109/SCC.2015.11

settings and the results are presented and analyzed in Section V. Finally, the contribution of our work is provided in Section VI.

II. Cascading Failure

In the study of critical infrastructure, cascading failure is a type of failure in which the disruption of one infrastructure causes disruption in another [6]. From network perspective, cascading failure is a situation where the failure of a node (or an edge) cascades to one or more nodes (or edges) in the network. Research on power grid cascading failure has been actively conducted as large-scale failures have occurred in some countries [7], [8], [9]. In the Internet industry, major companies, such as Instagram, Reddit, and Heroku, have also experienced problems due to the outage of Amazon EC2.

According to the direction of the cascade, we classify cascading failure into two types: *vertical* and *horizontal* cascading failure. In vertical cascading failure, the dependency between adjacent nodes determines the direction of the cascade. For directed networks, the failure can cascade along the direction of the edges, in the opposite direction, or in both ways as in metabolism network [10]. In horizontal cascading failure, the cascade direction is not necessarily related to the dependency between nodes. The failure can cascade to non-adjacent nodes that are connected to the same adjacent node.

Horizontal cascading failure has been widely studied in power networks, where the cascade is triggered by a node that tries to redistribute the load from an overloaded node (link) to another [5]. In service network, such failure may occur because of load redistribution among servers and routers handling user request. While research in power network cascading failure could be beneficial for understanding horizontal cascading failure in service networks, to the best of our knowledge, there is a lack of research on vertical cascading failure in service networks. Despite the rapid growth of publicly accessible services in the Internet, large-scale service networks have yet to become common, and thus cascading failure in service networks has not gained much interest. However, understanding vertical cascading failure is important since many existing real networks exhibit similar dependency among the nodes [11], [12]. Likewise, interdependent services populating the future Internet may experience the same type of cascading failure. In this paper, we address vertical cascading failure in service networks that is driven by the dependency between services.

III. Service Network

Service composition has been widely used to combine different service functionalities in service oriented environment. The Language Grid[1] and ProgrammableWeb[2] are the examples of such environment where different services and APIs can be combined to create new services. Service composition creates dependency between composite services and their required component services. The collection of services and the dependency between them in service oriented environment form service network where the nodes are services and the links are the dependency. As illustrated in Fig. 1, we define a service network as a directed network $N = (S, E)$, where $S = A \cup C$ is a set of services consisting of atomic services A and composite services C. The set $E = L \cup R$ consists of the set of dependency links L and alternate links R. These two links connect composite services to their required component services. The distinction between the two and more detailed explanation of Fig. 1 is provided in the following subsection.

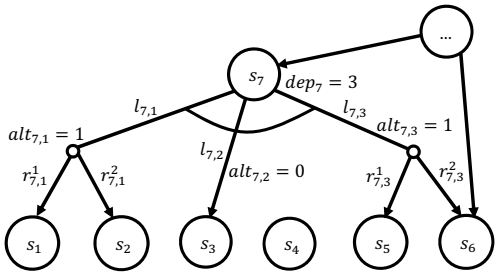

Figure 1. A sub graph of a service network. Circles represent services, and directed links represent the dependency of composite services on their required component services. Service s_7 is a composite service which requires three kinds of services. The existence of alternate services is indicated by a small circle that splits the dependency link, e.g. at the end of dependency links $l_{7,1}$ and $l_{7,3}$.

A. The Properties of Service Network

In this section, we consider the properties of service network that affect cascading failure. The number of cascade failed nodes is determined by the topology of the service network, the interdependency between the services, the depth of composite services, and the number of composite services in the network.

1) Network Topology: Topology of a network is usually determined based on their structural properties, such as degree distribution, which is one of the most widely accepted definitions [13]. Networks with different topology have different tolerance to cascading failure [5]. Service networks and many other real networks are growing networks in that the number of nodes increases over time. Based on their degree distribution, growing networks are mainly classified as scale-free or exponential networks. The degree distribution of the former follows a power-law, while the latter is exponential (Fig. 2). Formally, the power-law characteristic of a scale-free network is defined as

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

where $P(k)$ is the probability that a node in the network has k connections, and degree exponent γ is typically in

[1] langrid.org

[2] www.programmableweb.com

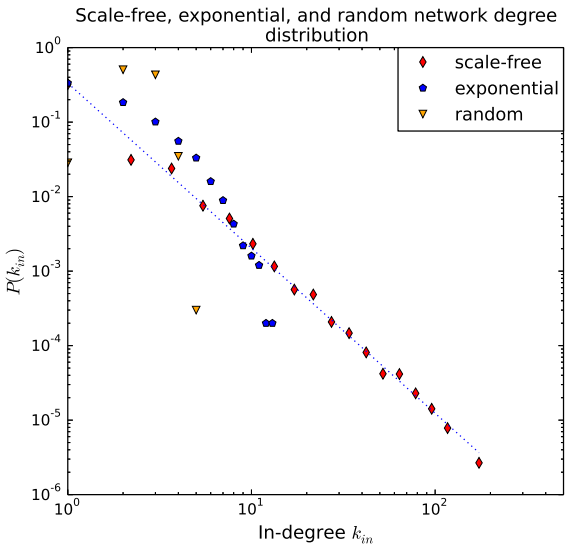

Figure 2. In-degree distribution of scale-free, exponential, and random service networks. The x-axis is the in-degree k_{in}, and the y-axis is $P(k_{in})$, i.e. the fraction of nodes in the network having in-degree k_{in}. A scale-free network is recognized with its power-law degree distribution. As it is named, the degree distribution of an exponential network is exponential. Finally, the degree distribution of a random network follows Poisson distribution.

the interval $2 < \gamma < 3$ [14]. This characteristic is the result of growth and preferential attachment, where newly added nodes prefer to connect to higher degree nodes. In the absence of preferential attachment, a growing network will become exponential [14].

Some existing work shows that real service networks exhibit scale-free characteristic [15], [16]. Scale-free networks also have been proven to have high tolerance against random failure, but high vulnerability to directed attack [4]. However, this finding does not take cascading failure into account. In contrast with this result, research in power network shows that random network has better tolerance against cascading failure [5].

In addition to the two aforementioned network types, another type of network of interest is random network. Some service networks may also have a rather static number of nodes, such as the network of embedded devices in a smart building environment. In contrast with growing networks, this kind of network assumes a fixed number of nodes that are randomly connected to each other [17]. Random networks are recognized by their Poisson degree distribution (Fig. 2).

2) Degree of Interdependency between Services: We have illustrated a service network in Fig. 1, where circles represent services and the outgoing links from the services represent the dependency of composite services on the required component services. To quantify the degree of interdependency between services in a service network, we define *degree of dependency* $\langle dep \rangle$ and *degree of alternative* $\langle alt \rangle$. The former can be obtained by taking the average of the number of dependency links dep_i for every composite service s_i in the network as follows

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

where $|C|$ is the number of composite services. Likewise, the latter is the average of the number of alternate links $alt_{i,j}$ for every dependency links $l_{i,j}$ which defined as

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

where $|L|$ is the number of dependency links.

In Fig. 1, $dep_7 = 3$ because there are three dependency links $l_{7,1}$, $l_{7,2}$, and $l_{7,3}$ belong to service s_7. These indicate that service s_7 combines three services to perform its function. Dependency links that belong to a composite service are connected by a curve to emphasize that they are mandatory. Since each functionality can be provided by one or more services, $alt_{i,j}$ quantifies the number of services that can substitute another service in case of failure. For example, the number of alternate links of dependency link $l_{7,2}$ is defined as $alt_{7,2} = 0$, because the link only connects s_7 to one service s_3. We call the dependency between s_7 and s_3 as *strong dependency*. When s_3 is unexecutable (fail), the dependent composite service s_7 will also be unexecutable because no alternate service is available. In this situation, the failure of s_3 cascades to s_7. The existence of alternate service is illustrated by splitting the dependency link into two or more alternate links as in $l_{7,1}$ and $l_{7,3}$. For instance, if service s_1 is unexecutable, the composite service still able to perform its function by invoking service s_2. In this case, the degree of alternative of both $l_{7,1}$ and $l_{7,3}$ are 1. We call this type of dependency as *weak dependency*.

3) Depth of Service Compositions: In Fig. 1, the composite service s_7 combines three functionalities provided by five atomic services. In other cases, composite services may not always combine atomic services. Service composition can be nested when a composite service combines one or more composite services. Failure of a component service that participates in nested composition may cascade all the way to the outermost composite services. Consequently, the deeper the nested composite services, the more services that potentially fail cascade-wise. To measure the dependency between composite services and their nested component services we define the depth of service s_i recursively as

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

where $S_i \subset S$ is the set of component services of s_i, $|S_i|$ is its cardinality, and $depth_j$ is the depth of a component service $s_j \in S_i$. Finally, we also need to define the average

of *depth* to reflect the depth of composite services in the whole network as

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

4) Proportion of Composite Services: The number of composite services in the network also determines the number of cascade failed nodes. The higher the number of composite services, the higher the likelihood of cascaded failure to occur. The proportion of composite services to the number of all services in the network is defined as follows

$$c = \frac{|C|}{|S|} \tag{6}$$

In this paper we only analyze the effect of network topology and degree of interdependency on cascading failure. For the depth of service composition, we assume shallow depth as in actual service networks where $\langle depth \rangle$ is close to 1. As for the proportion of composite services c, we also fix the value according to the same value of the ProgrammableWeb service network. Two publicly accessible actual service networks and their properties are addressed in the following subsection.

B. Publicly Accessible Service Networks

1) The Language Grid Service Network: The Language Grid is a service grid infrastructure that allows users to wrap their language resources as services, and to combine the services by means of service composition [18]. The dependency between composite services and their required component services in the Language Grid provides a straightforward example of a service network. The Language Grid service network model represents the typical service composition in service oriented environment. The service composition is modeled into two levels of abstraction, which are service type level and service instance level. Service types classify service instances based on their functionalities. Services with the same functionalities, e.g. translation services, belong to the same service type. A composite service type is defined as a combination of different service types. Each service type can be realized by one or more service instances, while each service instance typically belongs to a service type.

The Language Grid worldwide operation has been established such that service composition across different service grids is possible [19]. We analyze 117 atomic services and 21 composite services available in Kyoto Language Grid, which constitute the majority of the global Language Grid services. The composite services are vital since users frequently invoke these services by almost half (47%) of all invocations. The Language Grid services have low interdependency. On average, each composite service combines three component services at $\langle dep \rangle = 2.81$, and each component service can be substituted by roughly eleven services at $\langle alt \rangle = 10.92$. We also found that, on average, the composite services of the Language Grid are shallow in depth at $\langle depth \rangle = 1.29$. This implies that nested composite services are rare. The degree distribution shows that currently the network is between exponential and scale-free topology (Fig. 3). However, as the network grows larger, there is a possibility that it will morph into a scale-free network. This is because composite services are more likely to utilize widely used and popular services than less utilized and non-popular services.

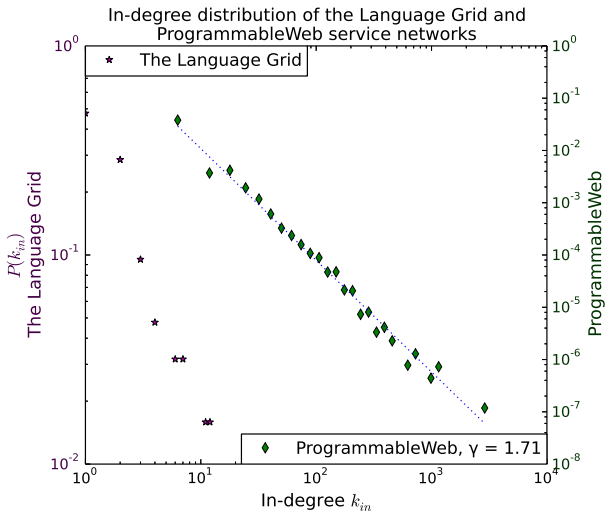

Figure 3. The Language Grid service network in-degree distribution is between scale-free and exponential due to its limited number of services, while the ProgrammableWeb service network fits with a power-law on degree exponent $\gamma = 1.71$.

2) ProgrammableWeb API Network: ProgrammableWeb is a website that provides information and directory of publicly available application programming interface (API). The API directory is open directory that allows developers to browse and retrieve information about existing APIs and mashups, and add information about their own APIs and mashups to be listed in the directory.

We retrieved 11,853 APIs and 7,513 mashups as of October 15, 2014, and we found that, on average, each mashup combines two APIs at $\langle dep \rangle = 2.11$. Unlike the Language Grid service network, the directory and retrievable data do not provide useful information to identify alternate links. Therefore, in this network we assume that all the links are dependency links, and thus $\langle alt \rangle = 0$. We also calculated the depth of the mashups and found that they are shallow in depth at $\langle depth \rangle = 1.00$, that indicates very rare nested mashups. An analysis of ProgrammableWeb network shows that both of the in-degree and out-degree distribution exhibit scale-free characteristic as they follow a power-law [15]. We also observed the in-degree of the recent ProgrammableWeb APIs and found that the distribution fits with a power-law at degree exponent $\gamma = 1.71$ (Fig. 3). The in-degree distribution confirms the scale-free characteristic shown by

Table I
THE LANGUAGE GRID (LG) AND PROGRAMMABLEWEB (PW) SERVICE NETWORK PROPERTIES

Property	LG	PW
Topology	Between exponential and scale-free	Scale-free
Degree of dependency $\langle dep \rangle$	2.81	2.11
Degree of alternative $\langle alt \rangle$	10.92	0
Depth of compositions $\langle depth \rangle$	1.29	1.00
Services $\|S\|$	138	19,366
Composite services $\|C\|$	21	7,513
$c = \|C\|/\|S\|$	0.15	0.39

the aforementioned work. Finally, the comparison of the two publicly accessible service networks in this section is summarized in Table I.

IV. GENERATING SERVICE NETWORKS

This section explains how we generate service networks on which cascading failure will be analyzed. First, we present how we generate growing service networks, i.e. scale-free and exponential service networks, and then random service networks. Next, we also address how the actual service networks, which are the Language Grid and ProgrammableWeb service networks, are generated.

A. Growing Service Network

Since both scale-free networks and exponential networks are growing network, generally we use the same algorithm to generate these kinds of service networks. The algorithm to generate scale-free service network is based on the scale-free web graph model [20], which is a scale-free directed network model. The work also formally proves that the network will eventually exhibit scale-free characteristic as the network grows.

The preferential attachment probability function to generate scale-free service network is defined as [21]

$$\Pi(k_i^{in}, \alpha) = \frac{k_i^{in} + \alpha}{\Sigma_j (k_j^{in} + \alpha)} \tag{7}$$

where k_i^{in} is the in-degree of service s_i, and j is the index for all services in the network. Function $\Pi(k_i^{in}, \alpha)$ calculates the likelihood of choose service s_i having k_i^{in} degree, which implies that services with higher in-degree have more likelihood of being chosen. The initial attractiveness parameter α yields the attractiveness value for isolated and young nodes [21].

The parameters used in growing service network algorithm are as follows:

- n_0 is the number of isolated services at time t_0
- Δt is the duration of the network generation where $\Delta t \geq 0$
- c is the expected proportion of composite services in the network in the interval $[0, 1]$
- 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 works as follows. First, initialize the network with n_0 number of isolated services. At each timestep t in the interval $[t_1, t_0 + \Delta t]$:

1) Add a service s_i to the network.
2) With probability c, generate a random number dep_i in the interval $[1, \delta \times 2 - 1]$.
3) If dep_i is generated in step (2), create dep_i dependency links from s_i. The set of dependency links belong to s_i is $L_i = \{l_{i,0}, ..., l_{i,dep_i - 1}\}$.
4) For each dependency link $l_{i,j} \in L_i$ created in step (3), generate $alt_{i,j}$, a random number in the interval $[0, \lambda \times 2]$. Perform one of these steps:
   a) If $alt_{i,j}$ is zero, choose an existing service and connect the dependency link $l_{i,j}$ from s_i to the chosen service.
   b) If otherwise, choose $alt_{i,j}+1$ number of existing services. Split dependency link $l_{i,j}$ into $alt_{i,j} + 1$ number of alternate links and connect these alternate links to the chosen services.

   The services are chosen using preferential attachment (Eq. 7) for scale-free service network, or in random manner for exponential service network.

In Fig. 2, the scale-free service network is generated with the parameters $n_0 = 10$, $\Delta t = 10,000$, $c = 0.4$, $\delta = 1$, $\lambda = 3$, and $\alpha = 1$. The in-degree distribution follows a power-law with degree exponent $\gamma = 2.22$. The exponential service network is also generated with the same parameters as the scale-free network but excluding α. As expected, the network shows exponential in-degree distribution.

B. Random Service Network

The algorithm to generate random service network is similar with the one to generate exponential service network. However, as a non-growing network, the algorithm does not include node addition and thus Δt is ignored. Random service network is initialized with n_0 isolated services, where n_0 is the expected size of the network, i.e. $n_0 = |S|$. For each service $s_i \in S$, the steps (2), (3), and (4) in the Section IV-A are performed. The in-degree distribution of a generated random service network is shown in Fig. 2, where $n_0 = 10,010$, $c = 0.4$, $\delta = 1$, $\lambda = 5$, and $\alpha = 1$. Poisson distribution is shown, and the probability $P(k_{in})$ peaks around the average in-degree at $\langle k_{in} \rangle = 2.47$.

C. Actual Service Network

The Language Grid service composition model distinguishes service types and service instances. To generate the

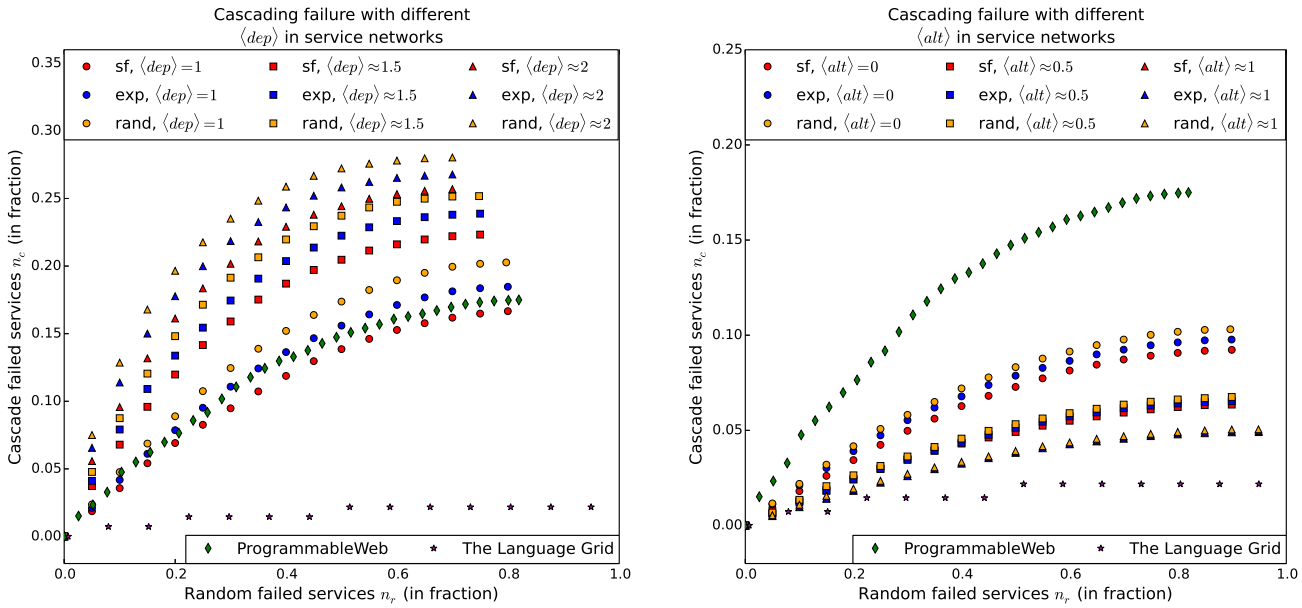

Figure 4. Cascading failure in in service networks with different degree of dependency $\langle dep \rangle$ (a) and different degree of alternative $\langle alt \rangle$ (b). In (a), $\langle alt \rangle = 0$, while in (b) $\langle dep \rangle = 1$. The x-axis is n_r, i.e. the number of services that are chosen randomly to represent random failure. The y-axis is the number of cascade failed nodes n_c, which is the number of services that have failed because of random failure. Both axes are represented in fraction against the initial size of the networks.

Language Grid service network based on this model, each service instance is created as a service node. The links from composite service nodes to their component service nodes are created based on the service instances they combined according to their invocation history. For example, a composite service that combines translation with bilingual dictionary service instances will be created as a service node that combines translation service nodes, e.g. Google Translation service, and some existing bilingual dictionary services. The generation of ProgrammableWeb service network is more straightforward. The APIs and mashups are generated as atomic service nodes and composite service nodes, respectively, while the dependency links are generated from these composite service nodes to their required components.

V. Cascading Failure Simulation and Analysis

To simulate cascading failure, first we generate service networks by implementing the algorithm presented in Section IV. For growing service networks, each is initialized with a small number of isolated service nodes ($n_0 = 10$), and then the network grows for the duration $\Delta t = 10,000$ until 10,000 nodes are added. As for random service networks, the number of services is fixed at $n_0 = 10,010$. The proportion of composite services is set at $c = 0.4$ according to the same property of ProgrammableWeb service network. For scale-free networks, initial attractiveness parameter is set at $\alpha = 1$. The service networks are generated with different values of $\langle dep \rangle$ and $\langle alt \rangle$ to investigate the effect of these properties to the number of services that have experienced cascaded failure n_c. For the generation of actual service networks, the parameters n_0, Δt, and c are given, as well as the network topology, $\langle dep \rangle$, and $\langle alt \rangle$, which are the result of their service composition.

Related work on network tolerance assumes node failure to be either random failure or directed attack [4], [5], [8], [10]. We perform random-based cascading failure after the service network is generated. At each timestep, a service is randomly chosen and deactivated to simulate a random failure. For convenience, let n_r be the cumulative number of such services. Any other services that strongly depend on this service will experience cascaded failure. This process is repeated until the network breaks down completely when $n_r + n_c = |S|$. Due to the stochastic nature of the algorithm, the simulation of cascading failure is performed multiple times for each combination of $\langle dep \rangle$ and $\langle alt \rangle$.

A. Service Network Breakdown on Cascading Failure

Figure 4a shows cascading failure in service networks where $\langle alt \rangle = 0$ with different $\langle dep \rangle$ values. Higher $\langle dep \rangle$ strengthens the effect of cascading failure because it represents higher interdependency between the services. We also assessed network breakdown with $\langle dep \rangle = 1$, which is the lowest possible value to have different $\langle alt \rangle$ (Fig. 4b). Higher $\langle alt \rangle$ provides more alternate services which consequently suppresses n_c (e.g. at $\langle alt \rangle \approx 1$). Cascading failure on real networks also gives similar result. The graph also shows that

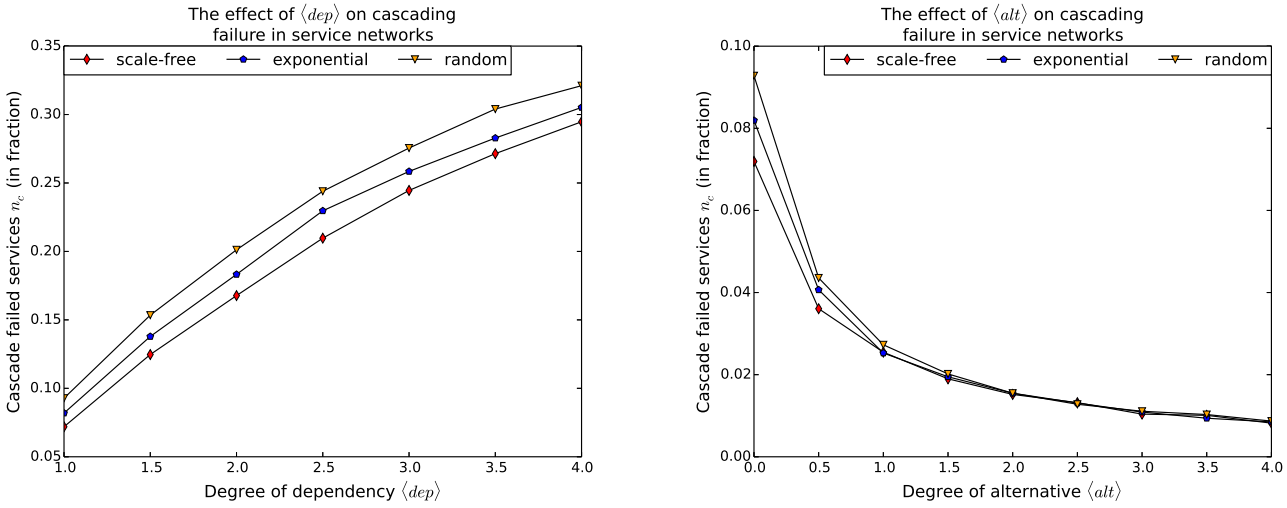

Figure 5. The effect of degree of dependency $\langle dep \rangle$ (a) and degree of alternative $\langle alt \rangle$ (b) on cascading failure in service networks. The y-axis is the number of cascade failed services n_c (in fraction) after random failure is performed on 20% of services in the network. In (a) $\langle alt \rangle = 0$, while in (b) $\langle dep \rangle = 1$ for the scale-free, exponential, and random service networks.

the existence of many alternate services on the Language Grid with $\langle alt \rangle = 10.92$ maintains the fraction of cascade failed services under 0.03. On the other hand, the absence of alternate services on ProgrammableWeb, i.e. $\langle alt \rangle = 0$, causes lower tolerance. The out-degree distribution of ProgrammableWeb also provides better tolerance than the generated scale-free network at similar $\langle dep \rangle \approx 2$.

Both Fig. 4a and Fig. 4b show that the scale-free topology generally has the lowest n_c, and then subsequently followed by the exponential and random service networks. The variation of tolerance between different network topologies is a result of their degree distribution. As illustrated in Fig. 2, both scale-free and exponential networks have very few high-degree nodes, while most of the nodes have low degree. The number of high-degree nodes (hubs) in a scale-free network is less than in an exponential network that have the same size and the same number of links. Consequently, in scale-free networks, random failure is much less likely to occur among these hubs. In random networks, the distribution of the nodes peaks at the average degree (Fig. 2). As a result, random networks have higher likelihood to experience random failure among high-degree nodes than exponential networks. Decreasing the interdependency between services improves the tolerance against cascading failure. As $\langle alt \rangle$ increases, the gap of n_c between different topologies decreases (Fig. 4b). This means that at high $\langle alt \rangle$, network topology has no significant effect on tolerance.

B. The Effect of Degree of Dependency on Cascading Failure

Figure 5a plots the relationship between n_c and $\langle dep \rangle$ in scale-free, exponential, and random networks. The graphs show the number of nodes fail in cascade-wise n_c when 20% of the services experience random failure at the corresponding $\langle dep \rangle$. In all of these networks, the degree of alternative is set at $\langle alt \rangle = 0$. The graphs show a linear relationship where n_c increases with $\langle dep \rangle$. We posit that this is due to the shallow depth of composite services in the network. In the actual service networks, shallow depth is shown in the Language Grid at $\langle depth \rangle = 1.29$, and in ProgrammableWeb at $\langle depth \rangle = 1.00$. The graphs suggest that reducing $\langle dep \rangle$ effectively lowering n_c. However, this may not always be possible in practice since the component services are usually determined by the problem domain and defined at design time.

C. The Effect of Degree of Alternative on Cascading Failure

Figure 5b shows how $\langle alt \rangle$ contributes to n_c where $\langle dep \rangle = 1$. For the scale-free, exponential, and random service networks, the graphs clearly show the inverse proportional relation between the two, where n_c decreases as the inverse of $\langle alt \rangle$. The inverse proportional relation suggests that the tolerance of the service network against cascading failure can be improved by increasing $\langle alt \rangle$. However, the improvement is potentially significant only when $\langle alt \rangle$ is currently low. In the graphs, n_c significantly drops in the interval $0 \leq \langle alt \rangle \leq 2$. On the other hand, when $\langle alt \rangle$ is already high (e.g. $\langle alt \rangle > 2$), the improvement is getting less significant. Increasing the degree of alternative by providing more alternate services for each component service significantly improves the network tolerance against cascading failure when the degree of alternative is currently low.

VI. Conclusion

This paper addresses cascading failure in service network, where the failure of one service can cascade to the other dependent services. Our objectives are to investigate which network topology provides better tolerance to cascading failure, and how the dependency between services contribute to cascading failure. We modeled a service network as a network of service composition, where the nodes are services, and the links are dependency between composite services and their required component services. The cascading failure simulations we performed reveal some important findings:

1) Scale-free topology generally shows better tolerance, followed by exponential and random topology, which is contrast with the load-based cascading failure tolerance in power networks [5].
2) The effect of network topology on tolerance is more significant on lower *degree of alternative*, which represents the average number of alternate services for each required component service.
3) The number of nodes experiencing cascading failure increases as the inverse of degree of alternative.
4) The number of nodes experiencing cascading failure is somewhat linear to the average number of required component services (degree of dependency).

These findings are valuable for designing a service network and designing the dependency between services as in service composition. Adding few alternate services for each composite service could significantly improve the tolerance if the degree of alternative is currently low, e.g. when each component service has only several alternate services; this is not so effective if the degree of alternative is already high.

Acknowledgment

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

References

[1] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based Internet of Things: Discovery, query, selection, and on-demand provisioning of web services," *Services Computing, IEEE Transactions on*, vol. 3, no. 3, pp. 223–235, 2010.

[2] S. Lee, J. Jo, Y. Kim, and H. Stephen, "A framework for environmental monitoring with Arduino-based sensors using Restful web service," in *Services Computing (SCC), 2014 IEEE International Conference on*, June 2014, pp. 275–282.

[3] J. Im, S. Kim, and D. Kim, "IoT mashup as a service: Cloud-based mashup service for the Internet of Things," in *Services Computing (SCC), 2013 IEEE International Conference on*, June 2013, pp. 462–469.

[4] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[5] P. Crucitti, V. Latora, and M. Marchiori, "Model for cascading failures in complex networks," *Physical Review E*, vol. 69, no. 4, p. 045104, 2004.

[6] S. Rinaldi, J. Peerenboom, and T. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *Control Systems, IEEE*, vol. 21, no. 6, pp. 11–25, Dec 2001.

[7] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, and R. Setola, "Modelling interdependent infrastructures using interacting dynamical models," *International Journal of Critical Infrastructures*, vol. 4, no. 1, pp. 63–79, 2008.

[8] R. Kinney, P. Crucitti, R. Albert, and V. Latora, "Modeling cascading failures in the North American power grid," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 46, no. 1, pp. 101–107, 2005.

[9] J. Ash and D. Newth, "Optimizing complex networks for resilience against cascading failure," *Physica A: Statistical Mechanics and its Applications*, vol. 380, pp. 673–683, 2007.

[10] A. G. Smart, L. A. Amaral, and J. M. Ottino, "Cascading failure and robustness in metabolic networks," *Proceedings of the National Academy of Sciences*, vol. 105, no. 36, pp. 13 223–13 228, 2008.

[11] R. G. Little, "Controlling cascading failure: Understanding the vulnerabilities of interconnected infrastructures," *Journal of Urban Technology*, vol. 9, no. 1, pp. 109–123, 2002.

[12] S. Gupta, A. Ray, S. Sarkar, and M. Yasar, "Fault detection and isolation in aircraft gas turbine engines. part 1: Underlying concept," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 222, no. 3, pp. 307–318, 2008.

[13] S. N. Dorogovtsev and J. F. Mendes, "Evolution of networks," *Advances in Physics*, vol. 51, no. 4, pp. 1079–1187, 2002.

[14] A.-L. Barabási, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statistical Mechanics and its Applications*, vol. 272, no. 1, pp. 173–187, 1999.

[15] K. Huang, Y. Fan, and W. Tan, "An empirical study of ProgrammableWeb: A network analysis on a service-mashup system," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*, June 2012, pp. 552–559.

[16] Z. Feng, B. Lan, Z. Zhang, and S. Chen, "A study of semantic web services network," *The Computer Journal*, 2014.

[17] P. Erdős and A. Rényi, "On the evolution of random graphs," *Magyar Tud. Akad. Mat. Kutató Int. Közl*, vol. 5, pp. 17–61, 1960.

[18] T. Ishida, Y. Murakami, and D. Lin, "The Language Grid: Service-oriented approach to sharing language resources," in *The Language Grid*, ser. Cognitive Technologies, T. Ishida, Ed. Springer Berlin Heidelberg, 2011, pp. 3–17.

[19] Y. Murakami, M. Tanaka, D. Lin, and T. Ishida, "Service grid federation architecture for heterogeneous domains," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, June 2012, pp. 539–546.

[20] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '03. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 132–139.

[21] Q. Chen and D. Shi, "The modeling of scale-free networks," *Physica A: Statistical Mechanics and its Applications*, vol. 335, no. 1, pp. 240–248, 2004.