Organization of Gateways and the Detection of Partitions

November 1, 2000

1 Organization of gateways

The organization of gateways reflects the connectivities which exist between various units within the system. Using this information, a node should be able to communicate and take actions to for any other node within the system. Any given node within the system is connected to one or more other nodes within the system. We refer to these direct links from a given node to any other node as *hops*. The routing information associated with an event, specifies the units which should receive an event. At each $g^{\ell}(C_i^{\ell})$ finer grained disseminations targetted for units u^{ℓ} within C_i^{ℓ} are computed. When presented with such a list of destinations, based on the gateway information the best hops to take to reach a certain destination needs to be computed. A node is required to route the event in such a way that it can service both the coarser grained disseminations and the finer grained ones. Thus a node should be able to compute the hops that need to be taken to reach units at different levels. A node is a level-0 unit, however it computes the hops to take to reach level-ℓ units within its context $C^{\ell+1}$ where $\ell = 0, 1, \cdots, N$ where $+1$ is the system level.

What is required is thus an abstract notion of the connectivities that exist between various units/super-units within the system. This constitutes the *connectivity graph* of the system. At each node the connectivity graph is different while providing a consistent overall view of the system. The view that is provided by the connectivity graph at a node should be of connectivities that are relevant to the node in question. Figure 1 depicts the connections that exist between various units of the 4 level system which we would use as an example in further discussions.

2 Constructing the connectivity graph

The organization of gateways should be on which provides an abstract notion of the connectivity between units u^{ℓ} within the context $C^{\ell+1}$ of the node. This interconnection can span multiple levels where if the gateway level is ℓ, a unit u_i^x $(x < \ell)$ within the context C^{x+1} is connected to u_j^{ℓ} within $C^{\ell+1}$. Units u_i^x and u_j^{ℓ} share the same $C^{\ell+1}$ context. For any given node within the system, the connectivity graph captures the connections that exist between units u^{ℓ}'s within the context C_i^{ℓ} that it is a part of. Thus every node is aware of all the connections that exist between the nodes within a cluster, and also of the connections that exist between clusters within a super cluster and so on. The connectivity graph is constructed based on the information routed by the system in response to the addition or removal of gateways within the system. This information is contained within the *connection*.

Not all gateway additions or removals/failures affect the connectivity graph at a given node. This is dictated by the restrictions imposed on the dissemination of connection information to specific sub-systems within the system, The connectivity graph should also provide us with information regarding the best hop to take to reach any unit within the system. The link cost matrix maintains the cost associated with traversal over any edge of the connectivity graph. The connectivity graph depicts the connections that exist between units at different levels. Depending on the node that serves as a level-ℓ gatekeeper, the cluster that the node is a part of is depicted as a level-1 unit having a level-ℓ connection to a level-ℓ unit, by all the clusters within the super cluster that the gatekeeper node is a part of.

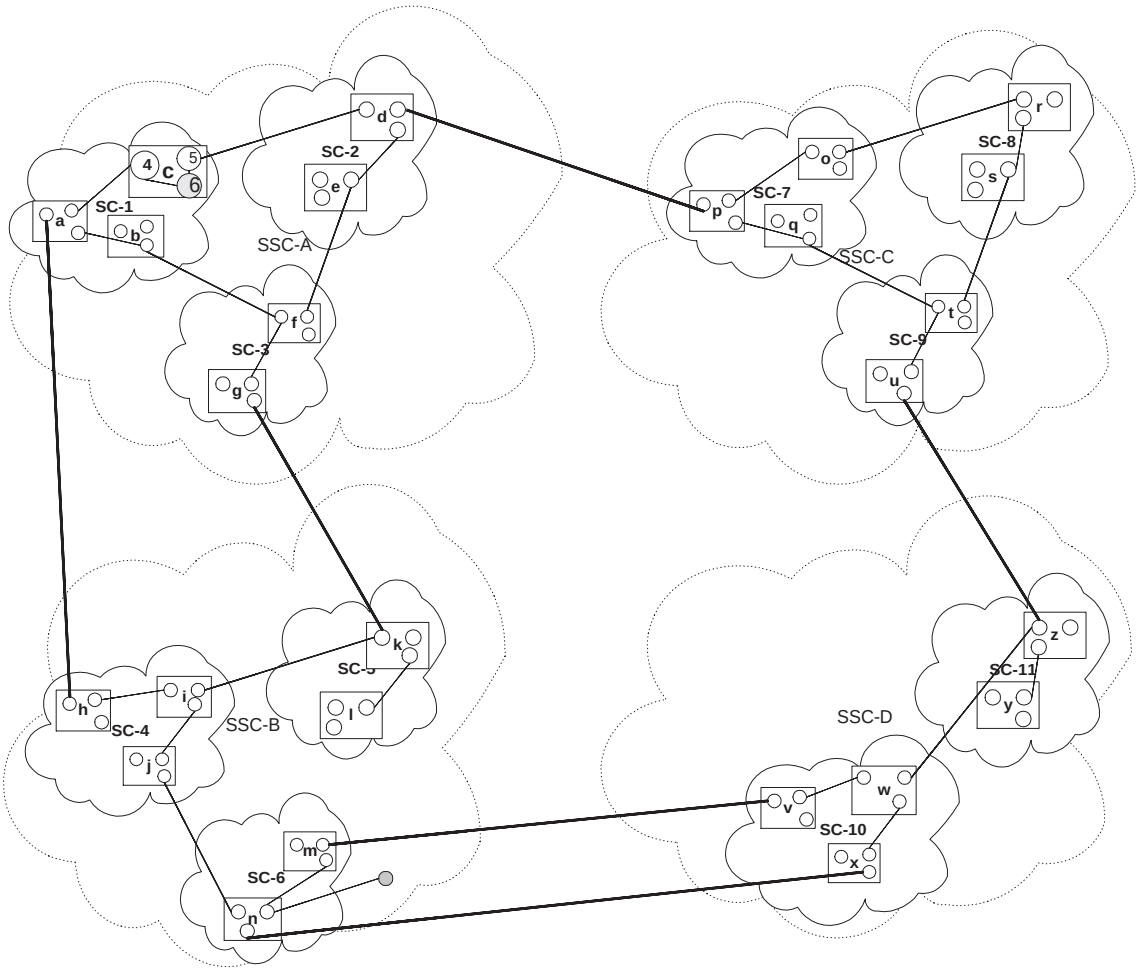

Figure 1: Connectivities between units

2.1 The connection

A connection depicts the interconnection between units of the system, and defines an edge in the connectivity graph. Interconnections between the units snapshots the kind of gatekeepers that exist within that unit. A connection exists between two gatekeepers. A level-ℓ *node* denoted n_i^ℓ in the connectivity graph, is the level-ℓ context of the gatekeeper in question and is the tuple $< u_i^\ell, \ell >$.

A level$-\ell$ connection is the tuple $< n_i^x, n_j^y, \ell >$ *where* $x \mid y = \ell$ *and* $x, y \leq \ell$. Units u_i^x and u_j^y share the same level-ℓ context $C_k^{\ell+1}$. For any given node n_i^ℓ in the connectivity graph we are interested only in the level $\ell, \ell+1, \cdots, N$ gatekeepers that exist within the unit and not the $\ell-1, \ell-2, \cdots, 0$ gatekeepers that exist within that unit. Thus, if a level-ℓ connection is established, the connection information is disseminated only within the higher level context $C_i^{\ell+1}$ of the sub-system that the gatekeepers are a part of. This is ensured by never sending a level-ℓ gateway addition information across any gateway $g^{\ell+1}$. Thus, in Figure 1 for a super-cluster gateway established within **SSC-A**, the connection information is disseminated only within the units, and subsequently the nodes in SSC-A.

When a level-ℓ connection is established between between two units, the gatekeepers at each end create the connection information in the following manner.

(a) For the gatekeeper at the far end of the connection, the node information in the connection is constructed using its level-ℓ context

(b) The other node of the connection is constructed as level-0 node.

(c) The last element of the connection tuple, is the connection level ℓ_c.

When the connection information is being disseminated through the context $C_i^{\ell+1}$, it arrives at gatekeepers at various levels. Every gatekeeper $g^p \ni p \leq \ell_c$, at which it is received, checks to see if any of the node information depicts a node n^x where $x < \ell_c$ if this is the case the next check is if $p > x$. If $p > x$ the node information is updated to reflect the node as level-p node by including the level-p contextual information of g^p. If $p \not> x$ the connection information is disseminated *as is*. Thus, in Figure 1 the connection between **SC-2** and **SC-1** in **SSC-A**, is disseminated as one between node **5** and **SC-2**. When this information is received at **4**, it is sent over as a connection between the cluster **c** and **SC-2**. When the connection between cluster **c** and **SC-2** is sent over the cluster gateway to cluster **b**, the information is not updated. As was previously mentioned, the super cluster connection **(SC-1,SC-2)** information is disseminated only within the super-super-cluster **SSC-A** and is not sent over the super-super-cluster gateway available within the cluster **a** in **SC-1** and cluster **g** in **SC-3**.

2.2 The link cost matrix

The link cost matrix specifies the cost associated with traversing a link. The cost associated with traversing a level-ℓ link from a unit u^x increases with increasing values of both x and ℓ. Thus the cost of communication between nodes within a cluster is the cheapest, and progressively increases as the level of the unit that it is connected to increases. The cost associated with communication between units at different levels increases as the levels of the units increases. One of the reasons we have this cost scheme is that the dissemination scheme employed by the system is selective about the links employed for finer grained dissemination. In general a higher level gateway is more overloaded than a lower level gateway. Table 1 depicts the cost associated with communication between units at different levels.

level	**0**	**1**	**2**	**3**	$\ell_{\mathbf{i}}$	$\ell_{\mathbf{j}}$
0	0	1	2	3	ℓ_i	ℓ_j
1	1	2	3	4	$\ell_i + 1$	$\ell_j + 1$
2	2	3	4	5	$\ell_i + 2$	$\ell_j + 2$
3	3	4	5	6	$\ell_i + 3$	$\ell_j + 3$
$\ell_{\mathbf{i}}$	ℓ_i	$\ell_i + 1$	$\ell_i + 2$	$\ell_i + 3$	$2 \times \ell_i$	$\ell_i + \ell_j$
$\ell_{\mathbf{j}}$	ℓ_j	$\ell_j + 1$	$\ell_j + 2$	$\ell_j + 3$	$\ell_j + \ell_i$	$2 \times \ell_j$

Table 1: The Link Cost Matrix

2.3 Organizing the nodes

The connectivity graph is different at every node, while providing a consistent view of the connections that exist within the system. This section describes the organization of the information contained in connections (section 2.1) and super-imposing costs as specified by the link cost matrix (section 2.2) resulting in the creation of a weighted graph. The connectivity graph constructed at the node imposes directional constraints on *certain* edges in the graph.

The first node in the connectivity graph is the *vertex*, which is the level-0 server node hosting the connectivity graph. The nodes within the connectivity graph are organized as nodes at various levels. Associated with every level-ℓ node in the graph are two sets of links, the set L_{UL} which comprises connections to nodes $n_i^a \ni a \leq \ell$ and L_D with connections to nodes $n_i^b \ni b > \ell$. When a connection is received at a node, the node checks to see if either of the nodes is present in the connectivity graph. If any of the nodes within the connection is not present in the graph, they are added to the graph. For every connection, $< n_i^x, n_j^y, \ell >$ *where* $x \mid y = \ell$ *and* $x, y \leq \ell$, that is received if $y \leq x$ node

- n_j^y is added to the set L_{UL} associated with node n_i^x
- n_i^x is added to the set L_D associated with the node n_j^y.

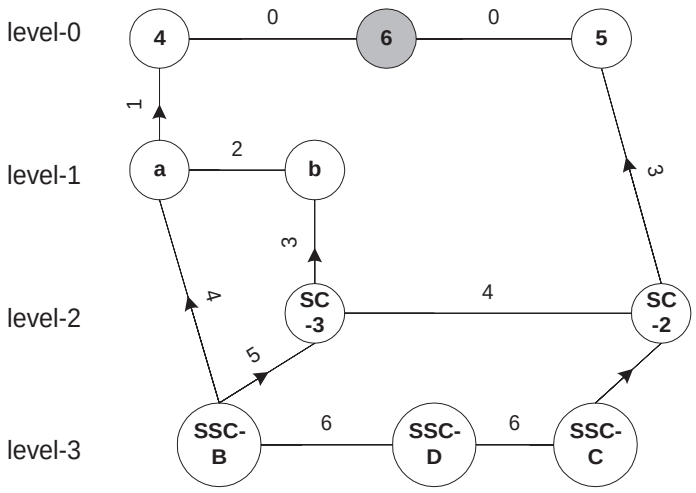

Figure 2: The connectivity graph at node 6.

The process is reversed if $x \leq y$. For the edge created between nodes n_i^x and n_j^y, the weight is given by the element (x, y) in the link cost matrix.

Figure 2 depicts the connectivity graph that is constructed at the node SSC-A.SC-1.c.6 in Figure 1. The set L_{UL} at the node **SC-3** in the figure comprises of node **SC-2** at level-2 and node **b** at level-1. The set L_D at **SC-3** comprises of the node **SSC-B** at level-3. The cost associated with traversal over a level-3 gateway between a level-2 unit **b** and a level-3 unit **SC-3** as computed from the linkcost matrix is 3, and is the weight of the connection edge. The directional issues associated with certain edges are imposed by the algorithm for computing the shortest path to reach a node.

2.4 Computing the shortest path

To reach the vertex from any given node, a set of links need to traversed. This set of links consitutes a *path* to the vertex node. In the connectivity graph, the best hop to take reach a certain unit is computed based on shortest path that exists between the unit and the vertex. This process of calculating the shortest path starts at the node in question. The directional arrows indicate the links which comprise a valid path from the node in question to the vertex node. Edges with no imposed directions are bi-directional. For any given node, the only links that come into the picture for computing the shortest path are those that are in the set L_{UL} associated with every node.

The algorithm proceeds by recursively computing the shortest paths to reach the vertex node, along every valid link (L_{UL}) originating at every node which falls within the valid path. Each fork of the recursion keeps track of the nodes that were visited and the total cost associated with the path traversed. This has two useful features -

(a) It allows us determine if a recursive fork needs to be sent along a certain edge. If this feature were missing, we could end up in an infinite recursion in some cases.

(b) It helps us decide on the best edge that could have been taken at the end of every recursive fork.

Thus say in the connectivity graph of Figure 2 we are interested in computing the shortest path to SSC-B from the vertex. This process would start at the node SSC-B. The set of valid links from SSC-B include edges to reach nodes **a**, **SC-3** and **SSC-D**. At each of these three recursions the paths are reflected to indicate the node traversed (SSC-B) and the cost so far i.e 4,5 and 6 to reach a, SC-3 and SSC-B respectively. Each recursion at every node returns with the shortest path to the vertex. Thus the recursions from a, SC-3 and SSC-D return with the shortest paths to the vertex. This added with the shortest path to reach those nodes, provides us with the means to decide on the shortest path to reach the vertex, which in this case happens to 5.

2.5 Building and updating the routing cache

The best hop to take to reach a certain unit is contained in the last node that was reached prior to reaching the vertex. This information is collected within the routing cache, so that messages can be disseminated faster throughout the system. The routing cache should be used in tandem with the routing information contained within a routed message to decide on the next best hop to take. Certain portions of the cache can be invalidated in response to the addition or failures of certain edges in the connectivity graph.

In general when a $level - l$ node is added to the connectivity graph, connectivities pertaining to units at level $\ell, \ell+1, \cdots, N$ are effected. For a $level - N$ system if a gateway g^{ℓ} within $u_i^{\ell+1}$ is established, the routing cache to reach units at level $\ell, \ell+1, \cdots N$ needs to be updated for all units within $u_i^{\ell+1}$. The case of gateway failures, detection of partitions and the updating of the cache is dealt with in a later section.

3 Exchanging information between super-units

When a subsystem u_i^{ℓ} is added to an existing system $u^{\ell+j+1}$ information regarding $g^{\ell+j}, g^{\ell+j-1}, \cdots, g^{\ell}$ is exchanged between the system and the sub system. The way the set of connections, comprising the connectivity graph, are sent over the newly established link is consistent with the rules we had set up for sending a connection information over a gateway as discussed in section 2.1. Thus if a new super cluster **SC-4** is added to the SSC-A sub-system and a super cluster gateway is established between SC-4 and node SC-1.6 the connectivity graphs at node 6 would be as depicted in Figure 3.(a) while the connectivity graph at the gatekeeper in SC-4 would comprise of the following connections sent over the newly established gateway.

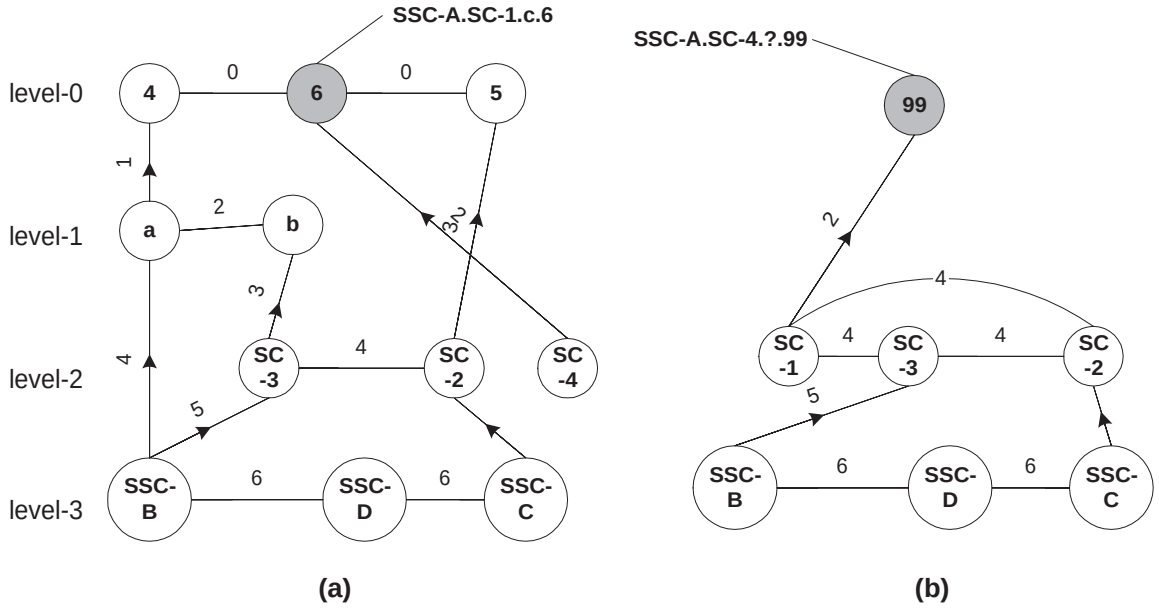

Figure 3: Connectivity graphs after the addition of a new super cluster SC-4.

Figure 3.(b) depicts only the connections which describe the connections involving level-2 gateways and upwards at node **99** in SC-4.

4 Detection of partitions

Partitions arise due to node failures or link failures. There are two different kinds of partitions that can arise in our system due to a connection failure - unit partitions and system partitions. The way

the system deals with each case is different. Dealing with partitions is through *delegation* where, each super-unit of the system deals with partitions that could arise within its units. Detection of partitions is an extremely desirable feature since in our system a client can roam in response to the partition. Thus clients hosted within the units of a primary partition can roam to nodes which are in the primary partition. Healing of the partitions could result in the affected units being able to deal with clients in a consistent manner and share the client load of the system.

The information contained in the loss of connections is identical to that contained in the addition of a new connection. Also the dissemination of this loss of connection is dealt with in exactly the same way as additions are as described in section 2.1. Loss of connections result in the removal of link information from the sets L_{UL} and L_D associated with the node. If the connection that was lost is the connection $< n_i^x, n_j^y, \ell >$ *where* $x \mid y = \ell$ *and* $x, y \leq \ell$, if $y \leq x$ node n_j^y is removed from the set L_{UL} associated with node n_i^x and n_i^x is removed from the set L_D associated with the node n_j^y. The process is reversed if $x \leq y$. The detection of partitions is very simple. At the node whose L_{UL} is updated to reflect the connection loss. If $\#L_{UL} = 0$ the unit corresponding to the node is unit partitioned. If $\#L_{UL} = 0 \bigcap \#L_D = 0$ the unit corresponding to the node is system partitioned.

Referring to the connectivity graphs in Figure 3 in the case of node 6 in 3.(a) the loss of the connection between clusters **a** and **b** results in **b** being unit partitioned within **SC-1** though it connected to SC-3. If however the only link that failed is the one connecting SC-1 and SC-3, no units are partitioned. In the last case, if the link connecting SC-1 and SC-4 hosted at node 6 fails, both node 6 in Figure 3.(a) and node 99 in Figure 3.(b) conclude that SC-4 has been system partitioned.

For nodes with $\#L_{UL} = 0$ the cost associated with reaching the vertex node $\rightarrow \infty$. All units which have their shortest path to the vertex resulting in a cost that $\rightarrow \infty$ are unit partitioned.