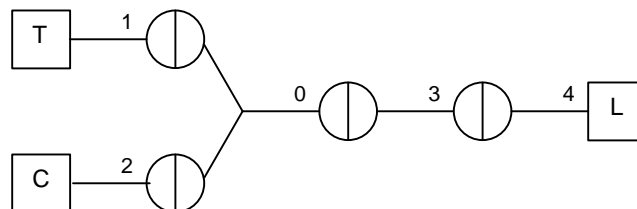


CONGRUENT SOFTWARE, INC.
98 Colorado Avenue
Berkeley, CA 94707
(510) 527-3926
(510) 527-3856 FAX

FROM: Peter Johansson
 TO: IEEE P1394.1 Working Group
 DATE: October 6, 1999
 RE: Stream connection setup

In the process of reviewing the efforts by David James to specify isochronous connection management in BR047R07 I've encountered some unresolved design issues. I believe they require the working group's attention, discussion and resolution before incorporation in the draft standard.

This document is based upon the following Serial Bus net; the drawing is simplified by the omission of most of the devices that would populate a real-world example including the cycle masters and isochronous resource manager! Neither the prime portal nor the alpha portals are illustrated, since they are not relevant to the scenarios discussed.



Bridges are shown as circles with a line to divide each portal from the other. Thus bus IDs are shown above the lines that represent different arbitration domains. The controller, talker and listeners are squares with the designation C, T and L respectively. Now for some definitions:

controller: a node that uses bridge management messages to set up or tear down routing for a stream between a talker and one or more listeners.

initial entry portal: the first bridge portal that eavesdrops on a transaction request or response and transforms the subaction's *source_ID* from a local node ID to a virtual node ID.

listener: a node (other than a bridge portal) that is the recipient of a stream.

listening portal: a bridge portal that listens to a set of channel numbers so that the packets may be retransmitted by its co-portal.

reallocation proxy: a bridge portal (only one per bus) responsible to reallocate channel numbers and bandwidth after a bus reset. The reallocation proxy also keeps track of listeners on the local bus; if there are ever no listeners, either as a result of explicit connection teardown or because of node removal, the reallocation proxy will coordinate the release of channel numbers and bandwidth.

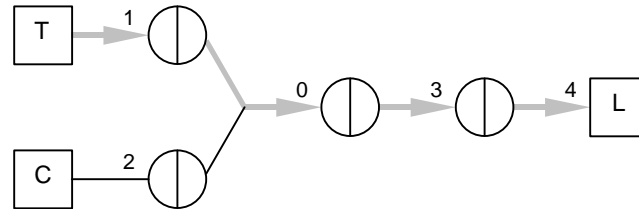
stream: Either asynchronous or isochronous data originated by a talker and received by zero or more listeners. A stream is uniquely identified by the talker's virtual node ID and the channel number used by the talker.¹ A stream's parameters include the payload, arbitration overhead and speed.

talker: a node (other than a bridge portal) that is the source of a stream within the net.

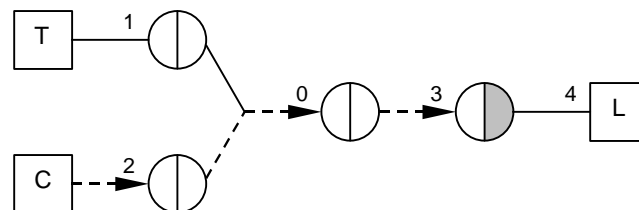
talking portal: a bridge portal that transmits stream packets for a set of channel numbers.

terminal exit portal: the last bridge portal that transmits (or could transmit) a transaction request or response en route to its destination; the portal that normally transforms the subaction's *destination_ID* from a virtual node ID to a local node ID.

Let's start with a simple connection set-up that does not overlay an existing connection. The controller on bus 2 wishes to establish a path from the talker on bus 1 to the listener on bus 4 (shown by gray arrows in the figure below). Once the controller knows the path has been created it may instruct the talker to commence broadcast.



The controller initiates the connection by transmitting a JOIN request as if it were intended for the listener. The JOIN request has its *snarf* field set to one, which causes it to be intercepted by the terminal exit portal. The path of the JOIN request is indicated by dotted lines below; the portal that intercepts the JOIN request is shown shaded.



The JOIN request contains parameters sufficient to permit incremental set-up of the entire connection and eventual provision of completion notification to the originating controller.

¹ It may be more practical to identify a stream by the talker's EUI-64 and an index to one of the talker's plug control registers, but the principal remains the same.

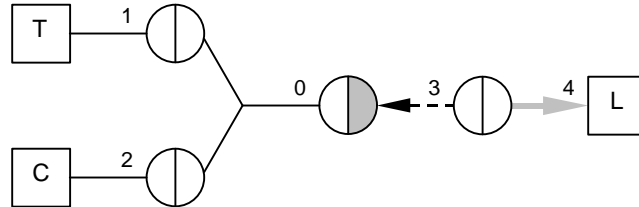
JOIN request parameters	
Parameter	Description
<i>stream_ID</i>	Uniquely identifies a stream within a net of interconnected buses. It is not yet resolved whether <i>stream_ID</i> is a composite of <i>talker_virtual_ID</i> and an index value unique at the talker, a composite of <i>talker_EUI_64</i> and an index that selects one of the talker's output plug control registers or another construct.
<i>controller_virtual_ID</i>	When the path is ultimately completed, confirmation is sent to the controller.
<i>talker_virtual_ID</i>	Permits the path to be established in a reverse direction, one bus at a time, from the listener to the talker.
<i>payload</i>	In conjunction with <i>speed</i> , permits calculation of bandwidth units to be allocated from BANDWIDTH_AVAILABLE. Zero indicates an asynchronous stream.
<i>speed</i>	The speed at which stream packets are to be transmitted.

The recipient of the JOIN request, a terminal exit portal, becomes the reallocation proxy for the stream identified by *stream_ID* on this bus. The reallocation proxy is often, but not always, the talking portal for the stream. A reallocation maintains a database for all active streams, keyed upon *stream_ID*. The database includes the following:

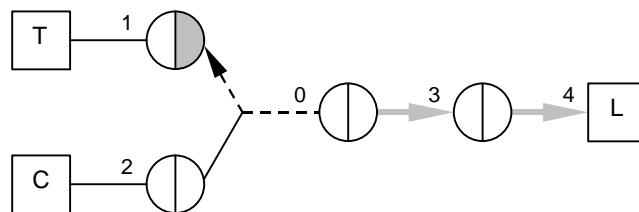
Stream context information maintained by a reallocation proxy	
Parameter	Description
<i>stream_ID</i>	Uniquely identifies a stream within a net of interconnected buses.
<i>controller_virtual_ID</i>	Provide error or other information to the controller if the reallocation proxy takes action that affects the stream.
<i>channel</i>	The channel number allocated by the reallocation proxy from CHANNELS_AVAILABLE for the stream's use on the local bus
<i>payload</i>	If nonzero, the maximum payload permitted in a stream packet transmitted on the local bus.
<i>speed</i>	The speed at which stream packets are to be transmitted.
<i>bandwidth</i>	The number of bandwidth units allocated for the stream's use on the local bus. It is calculated from <i>payload</i> and <i>speed</i> but also includes an allowance for isochronous arbitration overhead.
<i>members</i>	A bit map that represents each node on the local bus for which a JOIN request has been successfully processed. The use of a bit map in place of a count permits the reallocation proxy to monitor the removal of listeners or talkers.

Upon receipt of a JOIN request, the reallocation proxy first reserves the necessary resources, channel number and bandwidth (the latter only in the case of an isochronous stream) on its local bus. If successful in its attempted reservations, reallocation proxy's local bus is configured for the stream (indicated by the gray arrow in the next figure below) and an iterative process is started that extends a stream path backward from the listener's bus to the

talker's bus.² This process uses a JOIN request sent to either *a*) the talking portal (or what will be the talking portal) for the stream on the immediately upstream bus or *b*) a portal arbitrarily selected to be the reallocation proxy on the talker's bus. This JOIN request is essentially the same as the initial JOIN request originated by the controller except that its *destination_ID* is the talker's (not the listener's) virtual node ID and the *snarf* field has a value of three, which causes it to be intercepted by the initial entry portal. The path of the first JOIN request is shown by a dotted line below:



If the recipient of this second type of JOIN request is not already a reallocation proxy for the stream identified by *stream_ID* it becomes one, as described above, and allocates the necessary resources from the isochronous resource manager on its bus. If the recipient is already the talking portal and reallocation proxy for the stream, it simply adds the new listener (the bridge portal identified by *source_ID* in the JOIN request's packet header) to its bit map of members on the local bus. In either case, if resources are successfully allocated, the process is iterated as shown below:



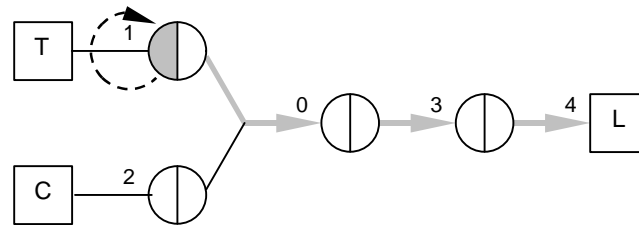
In this iteration, the JOIN request is once again addressed to the talker's *destination_ID* but the *snarf* field causes it to be intercepted by the next entry portal en route back to the talker. Resources were successfully allocated on bus 3; the new JOIN will cause resources to be allocated on bus zero and the bit map of members to be updated.

Eventually the connection's path will be established on all intermediate buses and the terminal bus that contains the listener. At this point the behavior required of a JOIN request changes. All of the previous JOIN requests have nominated a bridge portal as the reallocation proxy for a particular bus and all of these bridge portals are also the talking portal that transmits the stream on the bus. However, now that we have arrived back at the talker's bus, there is no need for a talking portal: the stream's source is the talker itself. Yet there is still a requirement for a reallocation proxy on the talker's bus, since the talker is not necessarily a controller and not necessarily responsible to reallocate stream resources after bus reset.

² Error recovery and the teardown of a partially setup connection are discussed later.

Which node should be selected as the reallocation proxy? Clearly it should be a bridge portal, since it must understand the protocol messages used to manage stream connections. The choice is somewhat arbitrary, but I suggest that the dominant portal on the talker's bus become the reallocation proxy for two reasons: a) the working group has already defined a selection process that uniquely determines the dominant portal on a bus and b) the identity of the dominant portal is unlikely to change without simultaneously invalidating the existing routing for the stream.

The last step in resource allocation is shown by the figure below. In this particular case, no Serial Bus transaction is involved even though conceptually the JOIN request was sent by the listening portal for the stream to the dominant portal which, in this example, happen to be one and the same and is indicated by the dotted line.



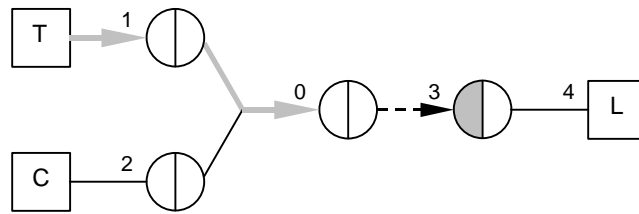
Once the final set of resources are allocated (on the talker's bus), it is necessary to communicate to the various listening portals en route from talker to listener the *channel* number they should associate with a particular stream ID. This is accomplished by a LISTEN request propagated and modified by the listening portals. The parameters included within a LISTEN request are as follows:

LISTEN request parameters	
Parameter	Description
<i>stream_ID</i>	Uniquely identifies a stream within a net of interconnected buses.
<i>channel</i>	The channel number on which the talking portal for the bus will transmit the stream.
<i>controller_virtual_ID</i>	When the path is ultimately completed and all listeners enabled, confirmation is sent to the controller.

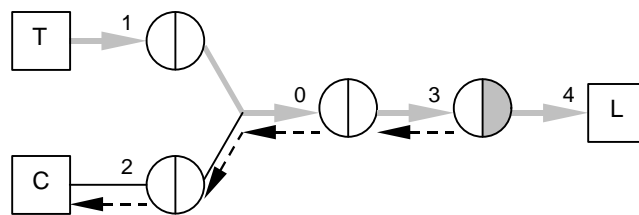
The process starts after the dominant portal on the talker's bus has allocated resources in its role as reallocation proxy for the stream. A LISTEN request is transmitted that identifies the channel number allocated and associates it with the stream; this request's *destination_ID* is set to the value of the listener's virtual ID (saved from the JOIN request originated by the controller) and the *snarf* field is set to three so the packet will be intercepted by listening portals on the way towards the listener.

When a listening portal receives a LISTEN request, it configures itself to listen to the indicated channel and forward stream packets to its co-portal. The listening portal also forwards the LISTEN request to its co-portal, which modifies the request to identify the channel number used for the stream on the talking portal's bus. The talking portal retransmits the LISTEN request, which is still addressed to the listener on the terminal bus. This process iterates for all intermediate buses until the LISTEN request is intercepted by the

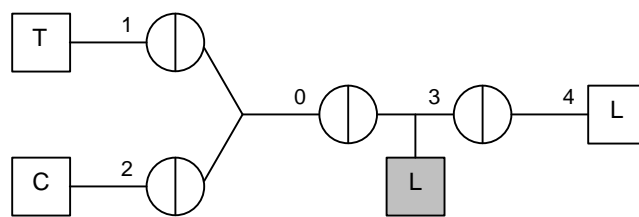
terminal bridge. The figure below shows the arrival of the request at the terminal bridge; at this point, all of the upstream buses are fully configured and able to transport the stream.



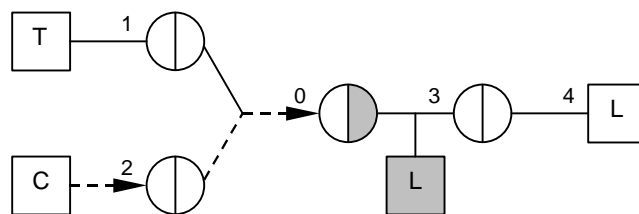
The terminal bridge has several jobs. First, it configures itself to listen on the indicated channel (as have all of the upstream listening portals). Next, it passes the request to its co-portal but instead of further propagation of the request to another listening portal, the co-portal configures the listener to receive the stream on the allocated channel. Once this step is complete, the terminal bridge transmits a CONFIRMATION message to the controller that first sent the join. This confirmation contains the *stream_ID* supplied by the controller in the original request (the path of the confirmation message, transmitted by the portal that intercepted the original JOIN request from the controller, is shown by a dotted line).



What happens in the case that a new listener is to be added to an existing multicast group? The operations are fundamentally similar, even though the route from the talker to the new listener may completely or partially overlay the path already in use. Consider the addition of a new listener (shown shaded).

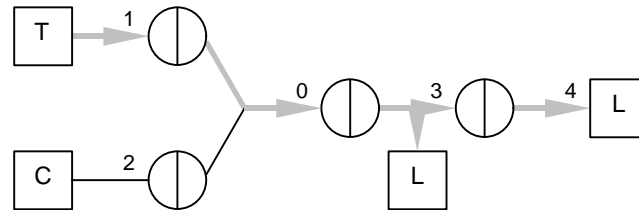


The controller transmits a JOIN request to the listener to be added; the *snarf* field is one to cause the terminal exit portal to intercept the request.

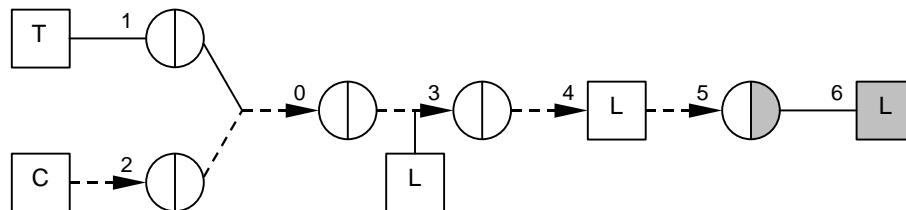


The sequence of JOIN, LISTEN and CONFIRM messages generated by the bridge portals is exactly analogous to that already described, except that no new resources are allocated. The shaded bridge portal, in its role as reallocation proxy, updates its bit map of members to indicate the presence of the new listener but otherwise has no additional work to perform.

Once the operations have completed, the resultant stream path is shown by the gray arrows below.



If the overlaid connection involved a new listener beyond the current extent of the stream's flow, the stream path is extended to accommodate the added listener. This is illustrated by the next figure.



The variations on this theme are endless (complex multicast routes with multiple branches may be created) but the basic operations of JOIN, LISTEN and CONFIRM remain the same.

In all of the preceding discussions, the identity of the new member of the stream multicast group has been implicitly derived from information in both the packet header and data payload of the join request. The working group may wish to make this information explicit within the data payload itself. In any case, the table below specifies how the identity of the new member is derived.

<i>snarf</i>	New member virtual node ID	Comment
0	<i>talker_virtual_ID</i>	When the dominant portal is selected as a reallocation proxy on the talker's bus, the JOIN request is sent directly to the dominant portal and the new member's identity is obtained from this field in the data payload.
1	<i>destination_ID</i>	The JOIN request originated by the controller is addressed and routed to the listener to be added but the packet is snarfed by the terminal exit portal.
2	Not used by JOIN requests	
3	<i>source_ID</i>	The JOIN requests propagated toward the talker's bus are snarfed by the initial entry portal en route to the talker.