ISO 15765-3（2004）

道路车辆——控制局域网络诊断——

第 3 部分：

一元化诊断服务实施（CAN 的 UDS）

道路车辆——控制器局域网（CAN）的诊断——
第 3 部分：
一元化诊断服务实施（CAN 的 UDS）

1 范围

这部分 ISO 15765 协议按照 ISO 14229-1，描述了在 ISO 11898 定义的控制器局域网中统一诊断服务（UDS）的实施。它给所有汽车连接至 CAN 网络服务器及外部测试设备提供诊断服务及服务器存储器编程的需求。它对汽车内部 CAN 总线架构无任何要求。

2 参考的标准

下述的参考文档对于该文档的应用是必不可少的。

3 术语，定义和缩略词

为编撰该文档目的，这些术语和定义已在 ISO 14229-1，ISO 15765-1 及 ISO 15765-2 中给出，以下缩略词术语同样适用。

DA 目标地址
ID 标识符
DLC 数据长度码
GW 网关
LSB 最低有效位
MSB 最高有效位
NA 网络地址
SA 源地址
SM 子网掩码
TOS 服务类型

4 协定

该部分 ISO 15765 协议基于 ISO 14229-1 的协定，该协议遵从使用到诊断服务的 OSI 服务协议。

5 统一诊断服务（UDS）对照 OSI 模型的应用

见图 1

6 应用层及会话层

6.1 应用层服务

该部分 ISO 15765 协议使用 ISO 14229-1 的客户机-服务器式的应用层服务。该系统具有测试、检测、监视，诊断及汽车服务器在线编程的功能。

6.2 应用层协议

该部分 ISO 15765 协议使用 ISO 14229-1 应用层协议。

6.3 应用层诊断会话管理定时

重要——任何一个服务器端产生的<N_Result>不等于 N_OK 的 N_USData.indication 的指示服务，服务器应用层都不应该有一个应答信息。

6.3.1 概况

下述的是应用层及会话层的定时参数及它们如何在客户机-服务器模式中如何处理的。

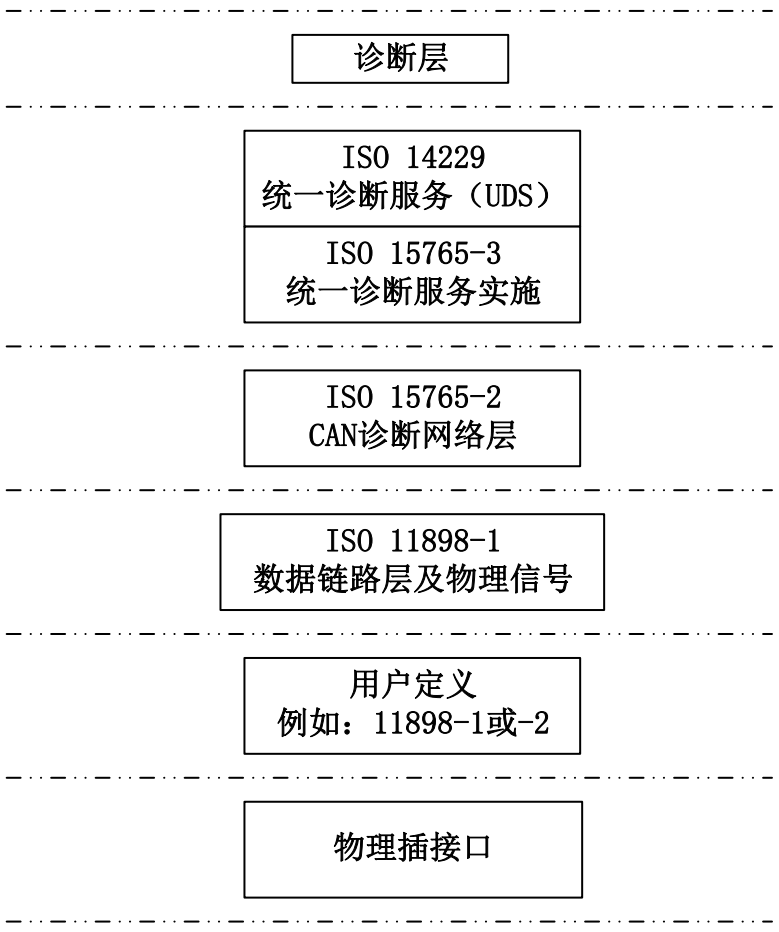


图 1　OSI 模型中，基于 CAN 的 UDS 实施

下述的几种通信会话方式需区别开：

a) 物理的通信在如下期间

1）默认会话方式

2）非默认的会话方式——需进行会话处理

b) 功能的通信在如下期间

1）默认的会话方式

2）非默认的会话方式——需进行会话处理

所有的情况下，请求服务器否定应答信息的扩展的定时应答，包括应答码 78hex 应当予以考虑。

定义在 ISO 15765-2 的网络层主要是处理客户机-服务器的应用层及诊断会话管理的定时。

6.3.2 应用层定时参数定义

用于默认的诊断会话的应用层定时参数值应按照如下表 2 设置

表 2——默认会话的应用层定时参数定义

定时参数	描述	类型	最小值	最大值
$P2_{CAN\_Client}$	成功发送请求信息（通过 N_USData.con 应答指示）到接收答复信息开始（多帧信息的 N_USDataFirstFrame.ind 和单帧信息的 N_USData.ind）的超时设置	定时器重载值	$P2_{CAN\_Server\_max} + \Delta P2_{CAN}$	N / A^a
$P2^*_{CAN\_Client}$	接收到应答码为 0x78 的否定应答（通过 N_USData.con 指示）到接收答复信息开始（多帧信息的 N_USDataFirstFrame.ind 和单帧信息的 N_USData.ind）的扩展的超时设置	定时器重载值	$P2^*_{CAN\_Server\_max} + \Delta P2_{CAN\_rsp}$	N / A^b
$P2_{CAN\_Server}$	在接收到请求信息（通过 N_USData.ind 指示），服务器开始答复信息的运行要求	运行要求	0	50ms
$P2^*_{CAN\_Server}$	在传递了 0x78（扩展的超时设置）的否定应答码（通过 N_USData.con 指示），服务器开始答复信息的运行要求	运行要求	0^c	5000ms
$P2_{CAN\_Client\_Phys}$	客户机成功发送不需应答的物理地址请求信息（通过 N_USData.con 指示），到它能发送下一个物理地址请求信息等待的最小时间（见图 6.3.5.3）	定时器重载值	$P2_{CAN\_Server\_max}$	N / A^d
$P2_{CAN\_Client\_Func}$	客户机成功发送功能地址请求信息（通过 N_USData.con 指示），到它能发送下一个功能地址请求信息等待的最小时间，有可能不需应答也有可能该请求数据只被某个子网功能地址服务器支持（见图 6.3.5.3）	定时器重载值	$P2_{CAN\_Server\_max}$	N / A^d

a 客户机等待一个应答信息发送的最长时间由客户机决定，但必须满足 $P2_{CAN\_Client}$ 必须比指定的 $P2_{CAN\_Client}$ 最小值要大；

b $P2^*_{CAN\_Client}$ 值由客户机决定，但必须满足该值必须比指定的 $P2^*_{CAN\_Client}$ 最小值要大；

c 扩展的应答定时，在连续的应答码为 0x78 的否定应答信息之间最小值为 $1/2\ P2^*_{CAN\_Server}$，最大容差为±20%的 $P2^*_{CAN\_Server}$；

d 客户机发送下一个请求的最长等待时间由客户机决定，但必须满足非默认会话的 $S3_{Server}$ 定时在服务器一直保持运行。

$\Delta P2_{CAN}$ 参数被认为是所有系统网络设计参考延时，该延时通过网关及总线带宽加上安全系数（例如最坏情况的 50%）。最坏情况（客户机-服务器-客户机信息传输一个来回的必须得传送时间），基于系统的设计，并受以下因素的影响：

a）包含网关的数量

b）CAN 帧发送的时间（波特率）

c）CAN 总线的使用情况

d）CAN 设备驱动使用方法（轮询方式还是中断方式）及网络层的处理时间

$\Delta P2_{CAN}$ 分为两个时间，一是客户机发送请求至服务器的时间，一是服务器发送应答至客户机的时间。

$$\Delta P2_{CAN} = \Delta P2_{CAN\_Req} + \Delta P2_{CAN\_Rsp}$$

图 2 展示的是 $\Delta P2_{CAN}$ 组成的一个例子。

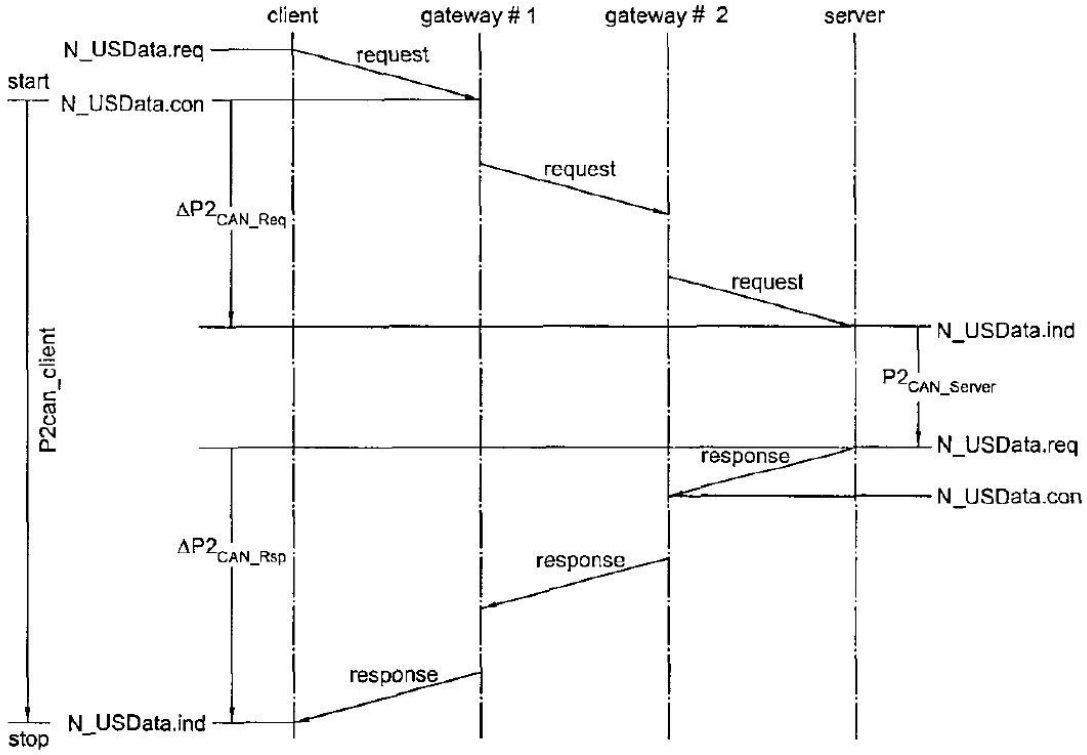


图 2 ——$\Delta P2_{CAN}$ 组成的一个例子——单帧请求和应答信息

注意：为了简单描述定时参数，在以下所有的图中，假定客户机到服务器在同一个网络中。所有的说明及附图按照时间顺序表述。

6.3.3 会话层定时参数定义

当诊断会话而不是默认的会话启动的时，需要按如下表 3 的会话层定时参数进行会话的操作。

表 3——会话层定时参数定义

定时参数	说明	类型	推荐超时 ms	超时 ms
$S3_{CIient}$	在功能地址（0x3E）由客户机发送的用于保持诊断会话的信息请求之间的时间，而不是多服务器的默认会话时间（功能的通信），或者对某一具体服务器发送请求最大时间间隔。（物理的通信）。	时间重置值	2000ms	4000ms
$S3_{Server}$	在没有接收到任何请求信息时，服务器保持诊断会话的时间，不是默认会话活动时间。	时间重置值	N/A	5000ms

而且，服务器转变到非默认会话时，应当改变它的应用层定时参数 $P2_{CAN\_Server}$ 和 $P2^{*}_{CAN\_CIient}$，以完成适用于诊断会话的操作。非默认的诊断会话适用的定时参数在诊断会话控制应答信息中报告，当一个应答需要传递（见图 9.2.1 服务说明）或需要提前通知客户不传递任何应答信息时。当客户机启动功能的非默认会话时，它应当调整响应的服务器的定时参数。

表 4 定义了客户机和服务器开启/重启的 $S3_{CIient}$ / $S3_{Server}$ 定时条件。对于客户机，周期性发送功能地址（0x3E）请求信息，应当与连续地发送物理地址（0x3E）请求信息区别开，后者仅仅在没有其它任何诊断请求时发送。对于服务器，不需要这两种（0x3E）的操作方式。

表 4 说明 $S3_{Server}$ 定时器操作是基于网络层服务的，也就是说，$S3_{Server}$ 定时器在接收到不支持的诊断请求信息时，重启。

6.3.4 客户机和服务器定时器资源要求

对于客户机及服务器在默认会话及任何非默认会话完成上述时间定时的定时器资源要求应按照表 5 及 6 所示。在非默认会话期间，表 6 所示附加的定时器资源要求适用于客户机及服务器。

表 4 ——客户机及服务器的会话层定时启动/停止条件

定时参数	动作	物理和功能通信， 使用功能地址， 周期性发送请求信息	物理通信， 使用功能地址， 连续发送请求信息
$S3_{Client}$	初始化开始	N_USData.con 用于指示诊断会话控制（10hex）请求信息的完成。只适用于非默认会话的会话类型。	若不需应答，N_USData.con 指示诊断会话控制（10 hex）请求信息的完成。
			若需一个应答，N_USData.ind 指示诊断会话控制（10 hex）请求信息的完成。
	随后的开始	N_USData.con 指示功能地址（0x3E）请求信息的完成，它是在 $S3_{Client}$ 定时每次到时时发送。	若不需应答，N_USData.con 指示诊断会话控制任何请求信息的完成。
			若需一个应答，N_USData.ind 指示诊断会话控制任何请求信息的完成。
			N_USData.ind 在接收到多帧应答信息时，指示出错。
$S3_{Server}$	初始化开始	如果需要一条应答信息被传送的话，N_USData.con 指示诊断会话控制应答信息的完成，表示从默认会话转变为非默认会话。	
		如果不需应答。成功地完成请求的服务，该请求为诊断会话控制（10 hex）请求信息要求从默认会话转变至非默认会话，	
	随后的结束	N_USDataFirstFrame.ind 指示多帧请求信息开始，N_USData.ind 表示任何一个单帧请求信息的接收。如果使用默认会话，$S3_{Server}$ 被禁用。	
	随后的开始	如果需要一条应答信息被传送的话（包括肯定及否定应答），N_USData.con 指示任何应答信息的完成，确定一条服务的执行（最后回复信息）。否定应答应答码 0x78 不会重启 $S3_{Server}$ 。	
		如果不需要任何应答信息（肯定或否定），请求动作的完成（服务结束）	
		N_USData.ind 指示接收多帧请求信息时的出错。	
		当请求发送未被请求的信息，如基于某一事件的周期性数据及应答，见 6.3.5.4 服务器关于 $S3_{Server}$ 更多的处理。	

表 5——默认会话下定时器资源要求

定时参数	客户机	服务器
$P2_{CAN\_Client}$	为每一个逻辑通信通道（物理和功能通信）设置一个单独的定时器是需要的，例如，点对点通信需要一个独立的通信通道。	N/A
$P2_{CAN\_Server}$	N/A	为扩展的应答定时一个可选择的定时器保证随后的否定应答的发送比 $P2^{*}_{CAN\_Server}$ 早一些。
$P2_{CAN\_Physical}$	需为每一个物理通信口提供单独的定时器	N/A
$P2_{CAN\_Functional}$	需为每一个功能通信口提供单独的定时器	N/A

表 6——非默认会话下另外的定时资源需求

定时参数	客户机	服务器
$S3_{CIient}$	当使用周期性发送，功能地址（0x3E）请求信息保持服务器在非默认状态，需提供单独的定时器，不需为每一个激活的诊断会话提供额外的定时器。	N/A
	当在无其它诊断请求时，使用连续的发送物理地址（0x3E）请求信息保持单个服务器在非默认状态，为每一个点对点通信通道设置单独的定时器	
$S3_{Server}$	N/A	服务器需一个单独的定时器，因为只有单诊断会话能在一个服务器中激活。

6.3.5 具体的定时参数描述

6.3.5.1 物理通信

6.3.5.1.1 默认会话下物理通信

图 3 描述了客户机和服务器在默认会话下物理地址请求信息定时的操作。

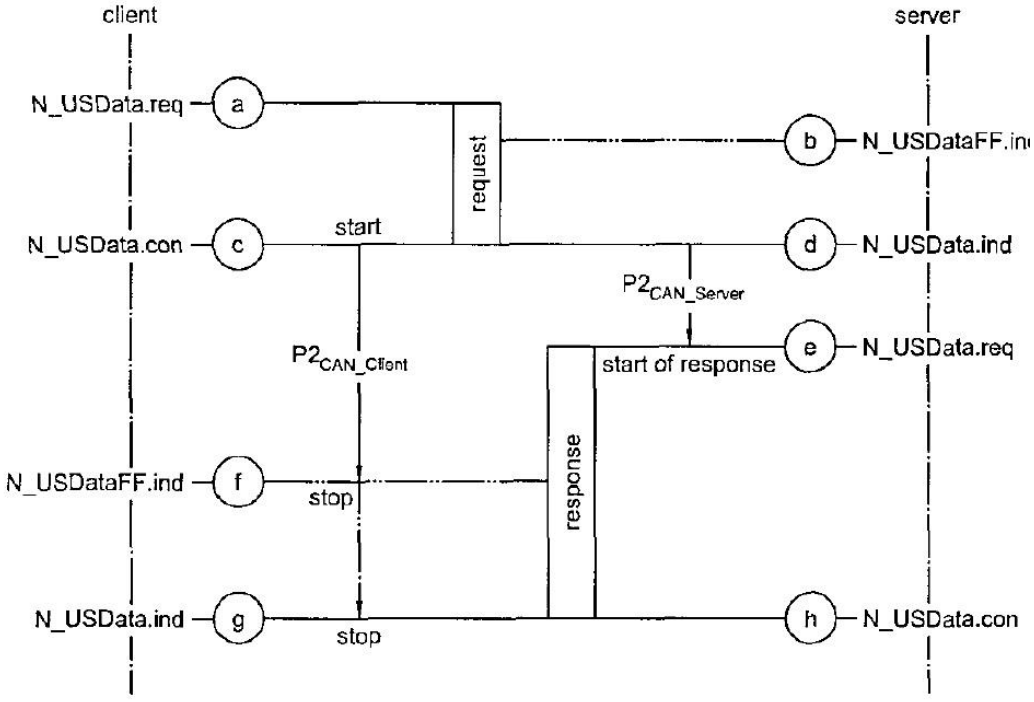


图 3——默认会话下物理通信

a) 客户端诊断应用层通过发送 N_USData.req 到网络层开始发送请求信息。网络层传递该请求信息至服务器。该请求信息要么以单诊的形式或多帧的形式。

b) 在多帧信息情况下，请求开始于网络层发送的 N_USDataFF.ind 通知服务器。

c) 请求信息的完成通过客户机 N_USData.con 指示。当接收到 N_USData.con 时，客户端使用默认重载值为 $P2_{CAN\_Client}$，启动 $P2_{CAN\_Client}$ 定时器，该定时器的值应当考虑到车载网络设计上（通信网关，总线带宽，等）所有的延时。为了简单化，该图假定客户机和服务器在一条总线上。

d) 服务器通过 N_USData.ind 指示请求信息的完成。

e) 服务器在接收到 N_USData.ind 指示时，要求在 $P2_{CAN\_Server}$ 时间内开始回复信息。也就是说，在多帧回复信息条件下，首帧必须在 $P2_{CAN\_Server}$ 时间内发送，对于单帧回复信息，该单帧必须在 $P2_{CAN\_Server}$ 时间内回复。

f) 在多帧应答信息情况下，客户机通过网络层 N_USDataFF.ind 指示首帧的接收。当接收到首帧时，客户机停止 $P2_{CAN\_Client}$ 定时器。

g) 如果完整的信息接收到，或者在接收过程中出现了错误，网络层最后都产生一个 N_USData.ind。在单帧响应信息，通过单个的 N_USData.ind 指示单帧的接收。当接收该单帧指示时，客户端停止 $P2_{CAN\_Client}$ 定时器。

h) 服务器通过 N_USData.con 指示响应信息的完成。

6.3.5.1.2 默认会话期间扩展了应答定时的物理通信

图 4 描述了默认会话期间客户机和服务器物理地址请求信息定时操作，及服务器请求扩展的响应定时（否定应答码 0x78 的处理）。

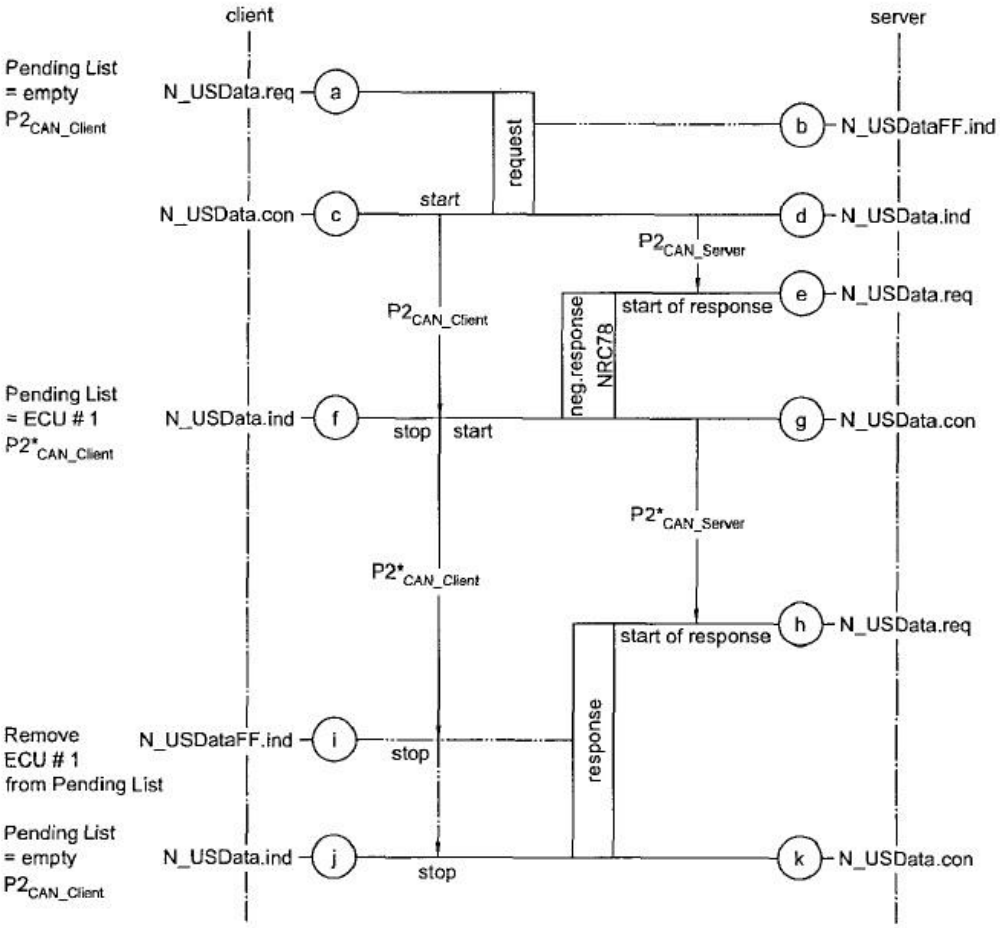


图 4 ——默认会话期间的物理通信——扩展了应答定时

a) 客户端诊断应用层通过发送 N_USData.req 到网络层开始发送请求信息。网络层传递该请求信息至服务器。该请求信息要么以单诊的形式或多帧的形式。

b) 在多帧信息情况下，请求开始于网络层发送的 N_USDataFF.ind 通知服务器。

c) 请求信息的完成通过客户机 N_USData.con 指示。当接收到 N_USData.con 时，客户端使用默认重载值为$P2_{CAN_Client}$，启动$P2_{CAN_Client}$定时器，该定时器的值应当考虑到车载网络设计上（通信网关，总线带宽，等）所有的岩石。为了简单化，该图假定客户机和服务器在一条总线上。

d) 服务器通过 N_USData.ind 指示请求信息的完成。

e) 服务器在接收到 N_USData.ind 指示时，要求在$P2_{CAN_Server}$时间内开始回复信息。也就是说，在多帧回复信息条件下，首帧必须在$P2_{CAN_Server}$时间内发送，对于单帧回复信息，该单帧必须在$P2_{CAN_Server}$时间内回复。

f) 服务器在给定的$P2_{CAN_Server}$时间内无法提供请求的信息时，它可以通过发送应答码为 0x78 的否定应答信息请求扩展的定时窗。客户端接收到否定应答信息时，客户端网络层产生一个 N_USData.ind。接收到应答码为 0x78 的否定应答信息，客户端重置它的$P2_{CAN_Client}$定时器，但使用的是扩展的重载的$P2^{*}_{CAN_Client}$定时值。

g) 服务器在发送否定应答信息 N_USData.con 之后，要求在给定的扩展的$P2_{CAN_Server}$（$P2^{*}_{CAN\_Server}$）时间内应答信息。如果在给定的扩展的$P2^{*}_{CAN_Server}$时间内仍无法提供请求的信息，服务器则继续发送应答码为 0x78 的否定应答。客户端使用的是扩展的重载的$P2^{*}_{CAN_Client}$定时值重置它的$P2_{CAN_Client}$定时器。为了简单起见，图中只显示了一个应答码为 0x78 的否定应答信息。

h) 一旦服务器可以提供请求的信息（肯定的否定的应答，而不是应答码 0x78 的应答），它就启动最后结果的应答信息。

i) 在多帧应答信息情况下，客户机通过网络层 N_USDataFF.ind 指示首帧的接收。当接收到首帧时，客户机停止$P2_{CAN_Client}$定时器。

j) 如果完整的信息接收到，或者在接收过程中出现了错误，网络层最后都产生一个 N_USData.ind。在单帧响应信息，通过单个的 N_USData.ind 指示单帧的接收。当接收该单帧指示时，客户端停止$P2_{CAN_Client}$定时器。

k) 服务器通过 N_USData.con 指示响应信息的完成。

6.3.5.1.3 非默认会话期间的物理通信

6.3.5.1.3.1 功能地址（0x3E）信息

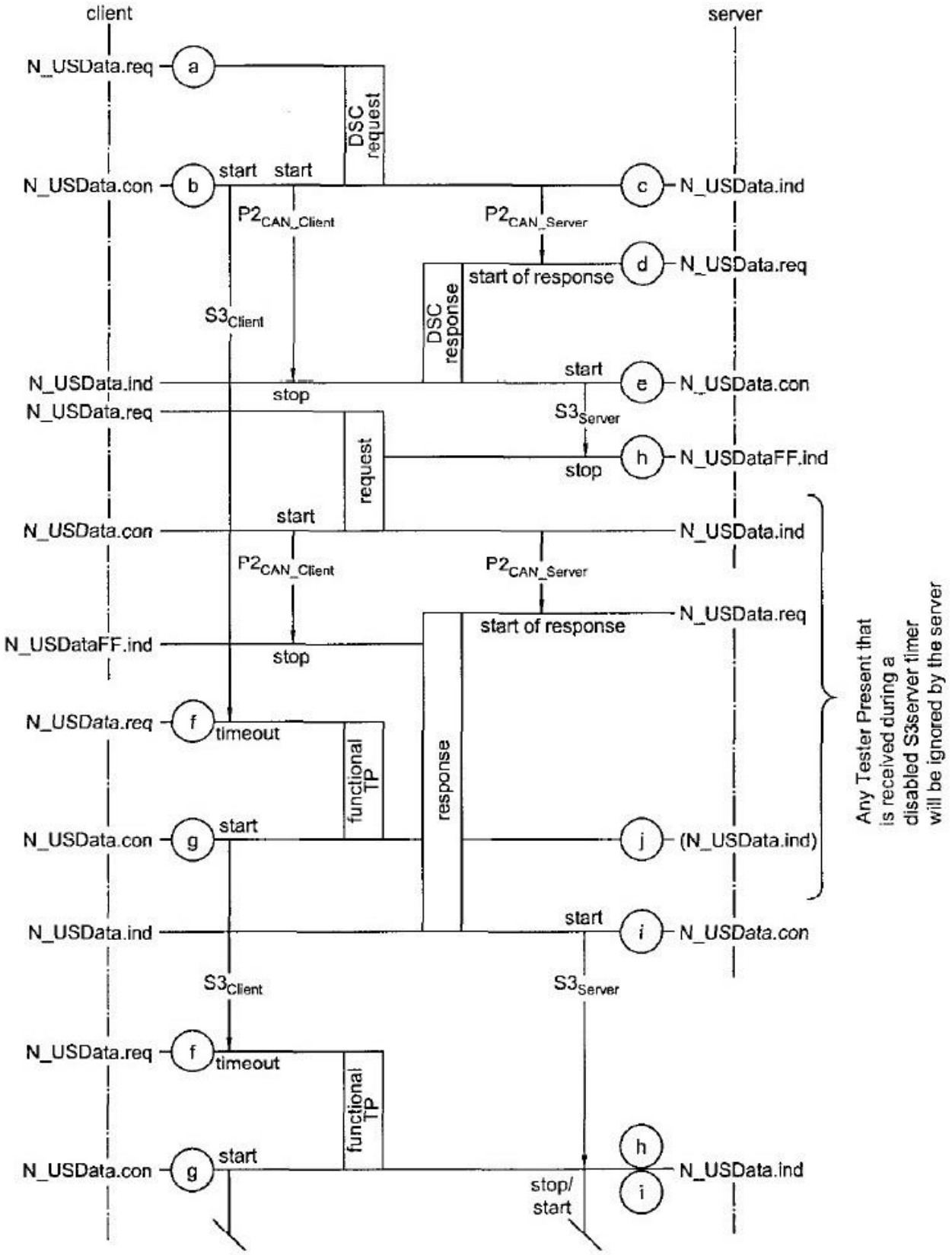


图 5 ——非默认会话期间的物理通信——功能地址

图 5 描述了客户机和服务器非默认会话期间物理通信及使用功能地址的定时处理。客户机周期性发送(0x3E)请求信息，不需要服务器的应答信息。

$P2_{CAN_Client}$与$P2_{CAN_Server}$定时处理与6.3.5.1.1和6.3.5.1.2小节中描述的处理方法相同。唯一的区别是客户端重置的值及服务器端发送结果应答时间会有不同。这是由于转变到另一会话层而不是使用默认会话层，因此使用的是不同的$P2_{CAN_Client}$的值。（见9.2.1节诊断会话控制（0x10）服务对定时参数更详细的描述。）

a) 客户端诊断应用层通过发送N_USData.req至网络层，传递诊断会话控制（0x10）请求信息。网络层传递该请求信息至服务器。

b) 请求信息是单帧信息。它的完成通过客户端N_USData.con指示。6.3.5.1.1和6.3.5.1.2描述的应答定时适用于此。客户端产生的N_USData.con促使$S3_{Client}$定时器开启（会话定时器）。

c) 服务器通过N_USData.ind的发送器一个应答。服务器应当发送诊断会话控制（0x10）的肯定应答信息。

d) 服务器通过N_USData.con指示应答信息发送的完成。然后服务器开启$S3_{Server}$定时器，只要它不超时，它就一直处于非默认状态。客户机负责保证$S3_{Server}$定时器在它超时之前复位，以保证服务器处于非默认会话状态。

e) 一旦客户机开启了$S3_{Client}$定时器，这会促使不需应答信息的功能地址（0x3E）请求信息的发送。每一次发送的时机都是在$S3_{Client}$超时时发送。

f) 在网络层通过N_USData.con指示（0x3E）请求信息传递完成之后，客户机再次启动$S3_{Client}$定时器。这就是说，功能地址请求信息是在每一次$S3_{Client}$定时超时之后，周期性发送的。

g) 服务器在处理诊断服务的任何时间内，它都停止$S3_{Server}$定时器。

h) 当诊断服务处理完之后，服务器重启$S3_{Server}$定时器。这就是说，诊断服务，包括（0x3E），都重置$S3_{Server}$定时器。诊断服务是在接收到请求信息（N_USDataFF.ind或者N_USData.ind服务）与完成最后结果应答这个期间内处理的。这里是需要一条应答信息的。或者请求然后诊断服务动作的完成不需要任何应答信息。（及时到达一个点会促使一个应答信息的发送）

i) 所有（0x3E）请求信息，在服务器处理另外一条请求信息期间接收的话，都会被服务器忽略。因为它已经停止了$S3_{Server}$定时器，并且在服务处理完之后重启。

6.53.5.1.3.2 物理地址（0x3E）信息

图6描述了非默认会话期间客户机与服务器物理通信的定时处理。以及使用物理地址（0x3E）请求信息需要服务器返回应答信息以保持在没有其它诊断服务的时候诊断会话的持续。

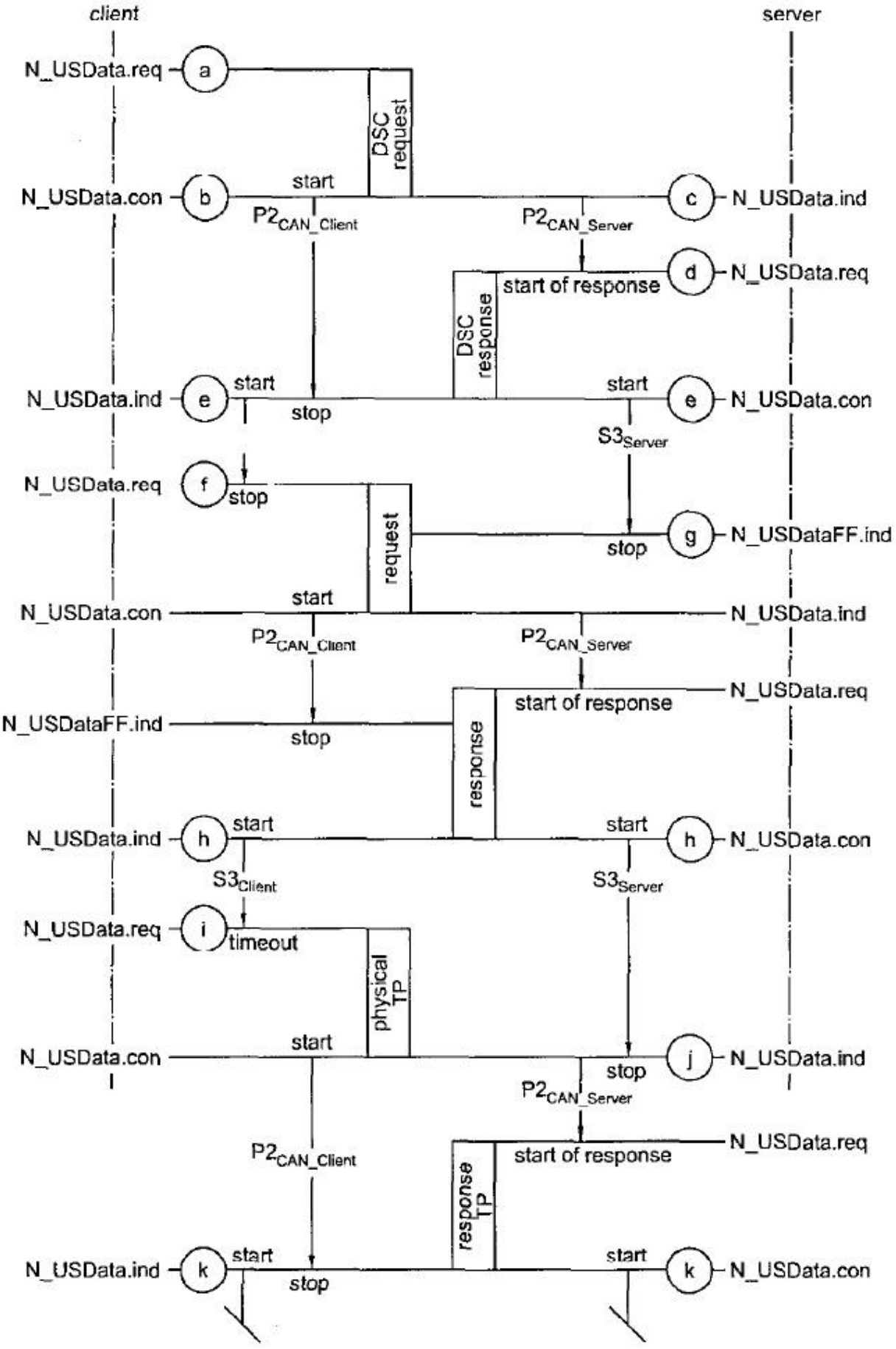


图 6 ——非默认会话期间的物理通信——物理地址

a) 客户端诊断应用层通过发送 N_USData.req 至网络层，传递诊断会话控制（0x10）请求信息。网络层传递该请求信息至服务器。

b) 请求信息是单帧信息。它的完成通过客户端 N_USData.con 指示。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。客户端产生的 N_USData.con 不会促使 $S3_{CIient}$ 定时器开启（会话定时器）。这与使用功能地址不同，使用功能地址会周期性发送(0x3E)信息保持诊断会话一直处于激活状态（见 6.3.5.3.1）。

c) 服务器通过 N_USData.ind 指示请求信息的完成。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。

d) 图上给出，假定客户机需要服务器一个应答。服务器应当发送诊断会话控制（0x10）的肯定应答信息。

e) 服务器通过 N_USData.con 指示应答信息发送的完成。然后服务器开启 $S3_{Server}$ 定时器，只要它不超时，它就一直处于非默认状态。客户机通过 N_USData.ind 指示诊断会话控制（0x10）的接收。这将促使 $S3_{CIient}$ 的开启。客户机负责保证 $S3_{Server}$ 定时器在它超时之前复位，以保证服务器处于非默认会话状态。

f) 客户机任何时候发送一条请求信息至服务器（包括（0x3E）信息），它都会停止 $S3_{CIient}$ 。

g) 接收到请求信息的单帧或首帧，服务器都停止 $S3_{Server}$ 定时器。服务器通过 N_USData.ind 标识请求信息的完成。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。

h) 客户机通过 N_USData.ind 指示应答信息的完成，这促使客户机开启 $S3_{CIient}$ ，服务器通过 N_USData.con 指示应答信息的完成，这促使服务器开启 $S3_{Server}$ 。还有一种客户机不需要应答的情况，客户机接收到网络层 N_USData.con 确认标识请求信息发送完时，开启 $S3_{CIient}$ ，服务器完成请求的动作时，开启 $S3_{Server}$ ，为简单起见，图中显示的是需要应答的情况。

i) 如果客户机在 $S3_{CIient}$ 超时之前，没有发送任何诊断请求信息，这促使客户机在 $S3_{CIient}$ 超时时，发送一条物理地址（0x3E）请求信息。

j) 服务器通过 N_USData.ind 指示（0x3E）请求信息的接收。这促使服务器停止 $S3_{Server}$ 定时器。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。

k) 客户机通过 N_USData.ind 指示（0x3E）应答信息的完成，这促使客户机开启 $S3_{CIient}$ ，服务器通过 N_USData.con 指示（0x3E）应答信息的完成，这促使服务器开启 $S3_{Server}$ 。还有一种客户机不需要应答的情况，客户机接收到网络层 N_USData.con（0x3E）标识请求信息发送完时，开启 $S3_{CIient}$ ，服务器完成请求的动作时，开启 $S3_{Server}$ ，为简单起见，图中显示的是需要应答的情况。

6.3.5.2 功能通信

6.3.5.2.1 默认会话期间的功能通信

图 7 描述了默认会话期间，一个客户机与 2 个服务器功能地址请求信息的定时处理。从服务器角度看，这与物理地址请求信息的定时处理没什么区别。但是客户机对定时的处理就与物理通信不同。

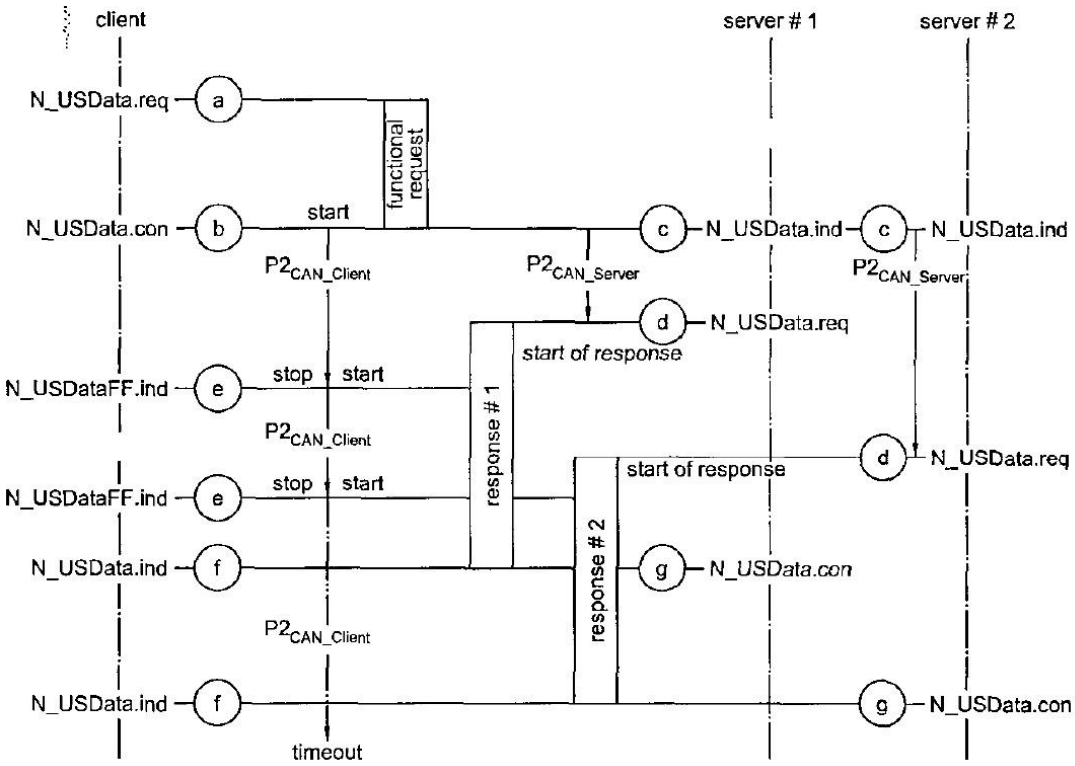


图 7——默认会话期间的功能通信

a) 客户端诊断应用层通过发送 N_USData.req 至网络层开始发送功能地址请求信息。网络层传递该请求信息至服务器。功能地址请求信息只能是单帧信息。

b) 客户机通过 N_USData.con 指示请求信息的完成。当接到 N_USData.con 时，客户机启动 $P2_{CAN\_Client}$ 定时器，使用默认的重置值 $P2_{CAN\_Client}$。该定时器的值应当考虑到车载网络设计上（通信网关，总线带宽，等）所有的延时。为了简单化，该图假定客户机和服务器在一条总线上。

c) 服务器通过 N_USData.ind 指示请求信息的完成。

d) 功能地址服务器在接收到 N_USData.ind 后，要求在 $P2_{CAN\_Server}$ 时间内发送应答信息。也就是说，在多帧回复信息条件下，首帧必须在 $P2_{CAN\_Server}$ 时间内发送，对于单帧回复信息，该单帧必须在 $P2_{CAN\_Server}$ 时间内回复。

e) 在多帧应答信息情况下，客户机通过网络层 N_USDataFF.ind 指示首帧的接收。当接收到首帧时，客户机停止 $P2_{CAN\_Client}$ 定时器。

f) 当接收到首帧/单帧指示接下来的应答信息，客户端要么知道服务器即将应答或已经应答过了，则停止 $P2_{CAN\_Client}$，要么不是所有服务器应答或它不知道服务器即将应答（客户机等待进一步的应答信息）时，重启 $P2_{CAN\_Client}$。如果完整信息接收到或者在接收过程中产生了一个错误，网络层产生最

后结果 N_USData.ind。对多帧信息的最后一个 N_USData.ind 不对 $P2_{CAN\_Client}$ 定时器产生影响。

g) 服务器通过 N_USData.con 指示应答信息发送的完成。、

6.5.3.2.2 默认会话期间扩展应答定时的功能通信

图 8 描述了默认会话期间客户机与 2 个服务器功能地址请求信息的定时操作。这里一个服务器通过应答码为 0x78 的否定应答请求一个扩展的应答定时。从服务器角度看，这与物理地址请求信息的定时处理没什么区别。但是客户机对定时的处理就与物理通信不同。

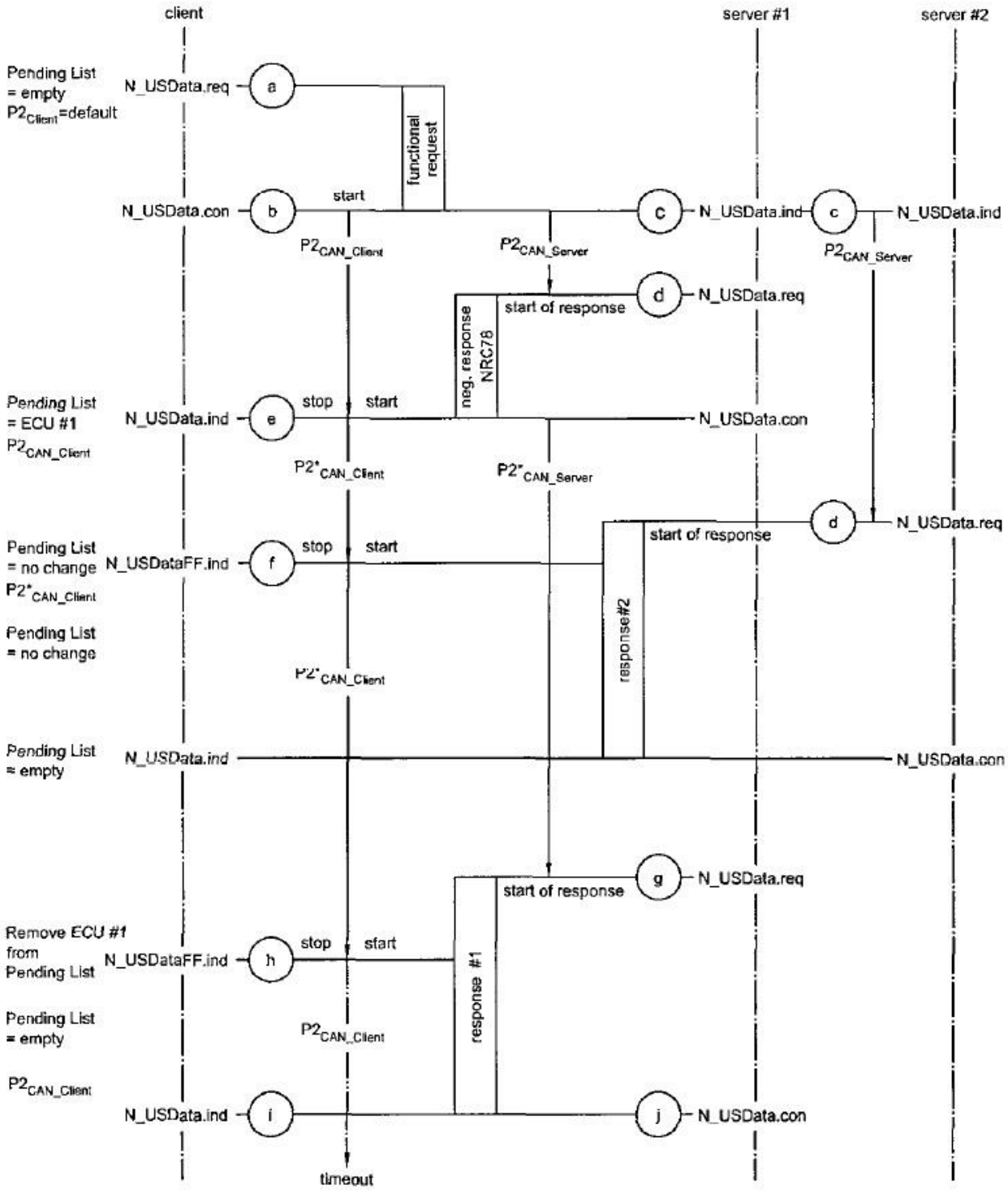


图 8——默认会话期间功能通信——扩展的应答定时

a) 客户端诊断应用层通过发送 N_USData.req 至网络层开始发送功能地址请求信息。网络层传递该请求信息至服务器。功能地址请求信息只能是单帧信息。

b) 客户机通过 N_USData.con 指示请求信息的完成。当接到 N_USData.con 时，客户机启动 $P2_{CAN\_Client}$ 定时器，使用默认的重置值 $P2_{CAN\_Client}$。该定时器的值应当考虑到车载网络设计上（通信网关，总线带宽，等）所有的延时。为了简单化，该图假定客户机和服务器在一条总线上。

c) 服务器通过 N_USData.ind 指示请求信息的完成。

d) 功能地址服务器在接收到 N_USData.ind 后，要求在 $P2_{CAN\_Server}$ 时间内发送应答信息。也就是说，在多帧回复信息条件下，首帧必须在 $P2_{CAN\_Server}$ 时间内发送，对于单帧回复信息，该单帧必须在 $P2_{CAN\_Server}$ 时间内回复。服务器在给定的 $P2_{CAN\_Server}$ 时间内无法提供请求的信息时，它可以通过发送应答码为 0x78 的否定应答信息请求扩展的定时窗。

e) 客户端接收到否定应答信息时，客户端网络层产生一个 N_USData.ind。接收到应答码为 0x78 的否定应答信息，客户端重置它的 $P2_{CAN\_Client}$ 定时器，但使用的是扩展的重载的 $P2^{*}_{CAN\_Client}$ 定时值。并且，客户端应当在挂起应答信息列表存储一个服务器标识。一旦在存储在客户端挂起的服务器开始它最后结果应答信息（肯定或否定应答信息包括应答码为 0x78 的应答），它将从挂起应答信息列表中删除。当无任何应答信息挂起时，客户端重新为 $P2_{CAN\_Client}$ 使用默认的重载值。为简单化，图中，显示了从服务器#1 的仅一个应答码为 0x78 的否定应答。

f) 只要至少有一个服务器在客户机端挂起时，从任一服务器端任何进一步的应答信息，都会促使 $P2_{CAN\_Client}$ 定时器使用扩展的值 $P2^{*}_{CAN\_Client}$ 重启（见图 9，该图显示了当客户机接收到第二个服务器应答信息开始的情况）。

g) 至于物理的通信，服务器请求扩展的应答定时要求在扩展的时间 $P2_{CAN\_Client}$（$P2^{*}_{CAN\_Client}$）内，应答信息。一旦服务器能提供请求的信息，它就通过发送 N_USData.req 至网络层开启最后结果应答信息。如果服务器仍然不能在扩展的 $P2^{*}_{CAN\_Client}$ 时间内提供请求的信息，它将继续发送应答码为 0x78 的否定应答信息。这会促使客户机再次重启 $P2_{CAN\_Client}$ 定时器，使用扩展的重载值 $P2^{*}_{CAN\_Client}$。已经存储在客户端挂起应答信息列表中，服务器端包含应答码为 0x78 的否定应答信息不影响客户端该信息列表。

h) 如 6.3.5.2.1，在多帧应答信息情况下，从任一服务器端接收的首帧，客户机都是通过网络层 N_USDataFF.ind 指示的。单帧应答信息通过 N_USData.ind 指示。当接收到首帧/单帧指示接下来的应答信息，客户端要么知道服务器即将应答或已经应答过了，则停止 $P2_{CAN\_Client}$，要么不是所有服务器应答或它不知道服务器即将应答（客户机等待进一步的应答信息）时，重启 $P2_{CAN\_Client}$。

i) 如果完整信息接收到或者在接收过程中产生了一个错误，网络层产生最后结果 N_USData.ind。这对 $P2_{CAN\_Client}$ 定时器不影响。而且适用挂起应答信息列表的处理。

j) 服务器通过 N_USData.con 指示完成发送。

6.3.5.2.3 非默认会话期间的功能通信

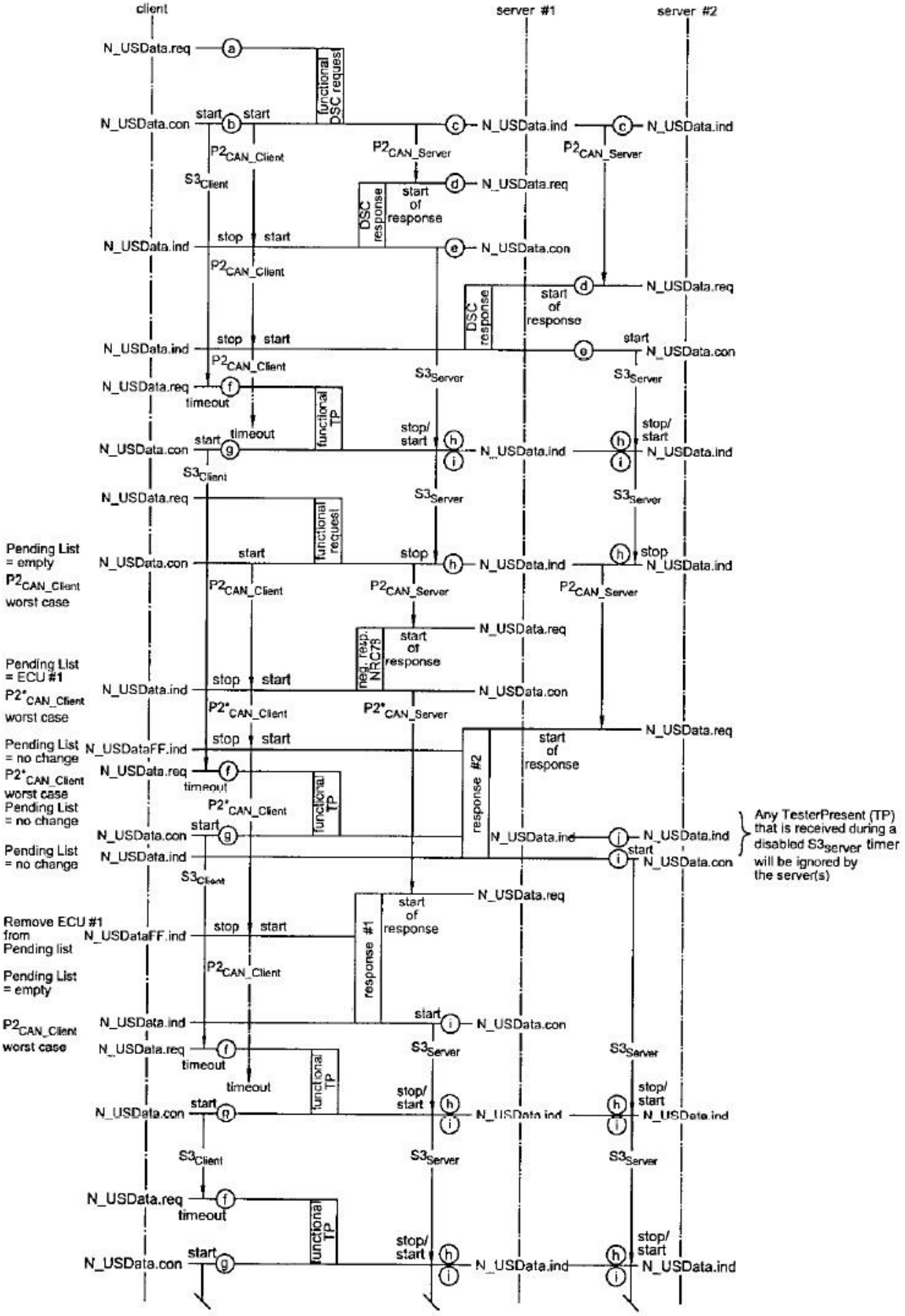


图 9 非默认会话期间的功能通信

图 9 描述了非默认会话期间客户机与 2 个服务器功能地址请求信息的定时操作。这里一个服务器通过应答码为 0x78 的否定应答请求一个扩展的应答定时。从服务器角度看，

a) 客户端诊断应用层通过发送 N_USData.req 至网络层开始功能地址诊断会话控制（0x10）的发送。网络层传递该请求信息至服务器。请求信息是单帧。
b) 客户端通过 N_USData.con 指示请求信息的完成。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。除此之外，客户端产生的 N_USData.con 促使 $S3_{CIient}$ 定时器开启（会话定时器）。
c) 服务器通过 N_USData.ind 指示请求信息的完成。6.3.5.1.1 和 6.3.5.1.2 描述的应答定时适用于此。
d) 图上给出，假定客户机需要服务器一个应答。服务器应当发送诊断会话控制（0x10）的肯定应答信息。
e) 服务器通过 N_USData.con 指示肯定应答信息发送的完成。然后服务器开启 $S3_{Server}$ 定时器，只要它不超时，它就一直处于非默认状态。客户机负责保证 $S3_{Server}$ 定时器在它超时之前复位，以保证服务器处于非默认会话状态。
f) 一旦客户机开启了 $S3_{Client}$ 定时器，这会促使不需应答信息的功能地址（0x3E）请求信息的发送。每一次发送的时机都是在 $S3_{Client}$ 超时时发送。
g) 在网络层通过 N_USData.con 指示（0x3E）请求信息传递完成之后，客户机再次启动 $S3_{Client}$ 定时器。这就是说，功能地址请求信息是在每一次 $S3_{Client}$ 定时超时之后，周期性发送的。
h) 服务器在处理诊断服务的任何时间内，它都停止 $S3_{Server}$ 定时器。
i) 当诊断服务处理完之后，服务器重启 $S3_{Server}$ 定时器。这就是说，诊断服务，包括（0x3E），都重置 $S3_{Server}$ 定时器。诊断服务是在接收到请求信息（N_USDataFF.ind 或者 N_USData.ind 服务）与完成最后结果应答这个期间内处理的。这里是需要一条应答信息的。或者请求然后诊断服务动作的完成不需要任何应答信息。（及时到达一个点会促使一个应答信息的发送）
j) 所有（0x3E）请求信息，在服务器处理另外一条请求信息期间接收的话，都会被服务器忽略。因为它已经停止了 $S3_{Server}$ 定时器，并且在服务处理完之后重启。

$P2_{CAN\_Client}$ 与 $P2_{CAN\_Server}$ 定时处理与 6.3.5.1.1 和 6.3.5.1.2 小节中描述的处理方法相同。唯一的区别是客户端重置的值及服务器端发送结果应答时间会有不同。这是由于转变到另一会话层而不是使用默认会话层，因此使用的是不同的 $P2_{CAN\_Client}$ 的值。（见 9.2.1 节诊断会话控制（0x10）服务对定时参数更详细的描述。）

6.3.5.3 客户机请求信息最小时间

为服务器轮询的服务数据的解读，这对客户机请求信息发送的最小间隔时间有要求的。例如，基于标准的功能，服务器可能处理诊断请求信息以预定的速率（例如 10ms）。诊断

服务数据解读预定时间应当比运行要求时间$P2_{CAN_Server}$短，以满足 6.3.5 和 6.3.5.1.2 对服务器要求。

请求信息间隔时间的最小定时参数分为如下两个定时参数。

——$P3_{CAN\_Functional}$：该定时参数适用于所有功能地址请求信息，因为它在不支持应答数据的情况下，服务器不要求响应功能地址请求信息。

——$P3_{CAN\_Physi}$：该定时参数适用于不需服务器应答的物理地址请求信息。（suppressPosRspMsgIndicationBit = TRUE）。

物理通信在需要服务器应答的情况下，客户端可以在接收到最后一条应答信息的时候立即发送下一个请求，因为服务器在完成最后结果应答时——意味着该请求已被服务器完全处理完了。

图 10 描述了功能通信期间出现一个问题的例子。当客户机在它确认所有期望的服务器都对先前做了应答时，立即发送下一个请求信息。

该情景不仅适用于功能地址请求也适用于物理地址请求，这里客户机不需接受任何应答信息（suppressPosRspMsgIndicationBit = TRUE）。

为了处理上述情况，在一条物理或功能地址请求信息与新的物理或功能地址请求信息之间，最小时间$P3_{CAN_Physical}$和$P3_{CAN_Functional}$需要为客户机定义。

a) $P3_{CAN_Physical}$的值与物理地址的服务器$P2_{CAN_Server_max}$的值相同。该定时适用于所有诊断会话（默认的或非默认的）的所有物理地址请求信息而且所有情况下，都不需要服务器应答。

客户机每次启动$P3_{CAN_Physical}$定时，都发送一条不需应答的物理地址请求信息到总线上，并且，网络层通过 N_USData.con 指示。当客户机在先前请求信息完全处理完之后，想要发送新的物理地址请求信息时，这只有在$P3_{CAN_Physical}$定时器不处于活动的情况下。

客户端在发送一条新的物理地址请求信息的时刻，启动$P3_{CAN_Physical}$。然后信息的发送要等到$P3_{CAN_Physical}$超时。

b) $P3_{CAN_Functional}$的值是所有功能地址服务器$P2_{CAN_Server_max}$，所有诊断会话（默认的或非默认的），所有功能地址请求信息的最大值（最坏情况）。

客户端每次开启$P3_{CAN_Functional}$定时器，都发送不需应答的功能地址请求信息到总线上，并且客户端网络层通过 N_USData.con 指示。当客户机在先前请求信息完全处理完之后，想要发送新的物理地址请求信息时，这只有在$P3_{CAN_Functional}$定时器不处于活动的情况下。客户端在发送一条新的物理地址请求信息的时刻，启动$P3_{CAN_Functional}$。然后信息的发送要等到$P3_{CAN_Functional}$超时。

注意：“完全处理完”就是说要么不需应答时没有接收到任何应答，要么所有期待的应答都

接受到了。应答的服务器知道并且要求应答，或者服务器不知道并且要求应答时出现 $P2_{CAN_Clien}$超时。

对服务器的要求是它应当在 $P2_{CAN\_Server}$（见图 7.3）时间内应答信息，这就是说，诊断信息的解读时间应当短于 $P2_{CAN\_Server}$。

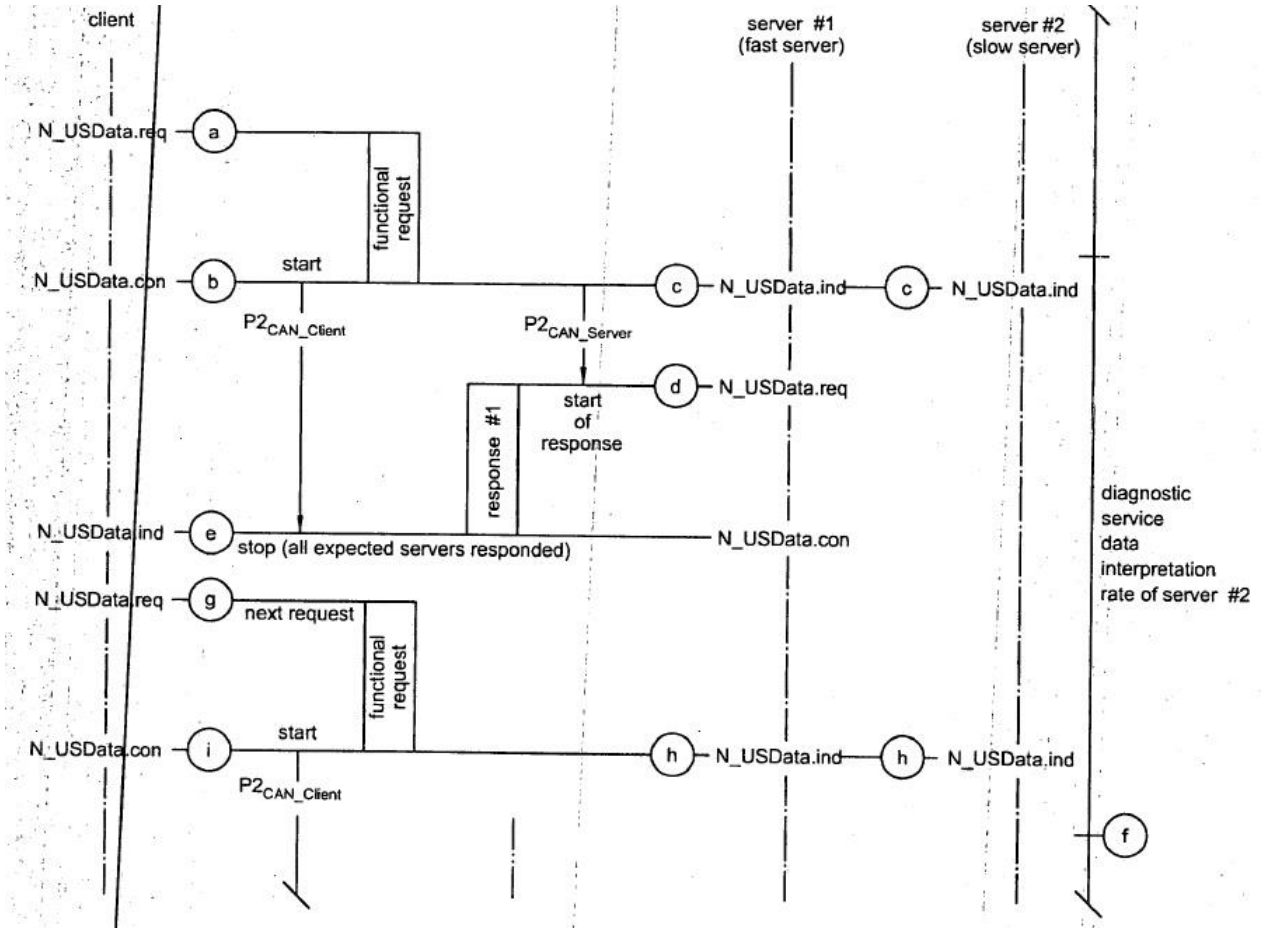


图 10 ——发送下一条请求太早的例子

a) 客户端诊断应用层通过发送 N_USData.req 功能地址请求信息到网络层。网络层传递信息到服务器。

b) 客户端通过 NUSData.con 只是请求信息的完成。客户机使用默认的 $P2_{CAN_Client}$值开启 $P2_{CAN\_Client}$ 定时器。

c) 服务器通过 N_USData.ind 指示请求信息的完成。服务器使用默认的 $P2_{CAN_Server}$值开启 $P2_{CAN\_Server}$ 定时器。

d) 对于请求的信息，假定只有服务器#1 支持请求信息，也就是说服务器#2 不会应答信息。服务器#1 是快速服务器，能很快处理完请求的信息并在 $P2_{CAN_Server}$时间内发送应答信息。

e) 客户机接收到应答信息。这通过 N_USData.ind 指示。客户机仅仅期待服务器#1 的应答信息，因此它停止 $P2_{CAN_Client}$定时器。

f) 服务器#2 是慢速服务器，并且在一段时间内（诊断服务数据解读时间）解读请求信息，最坏的情况下，在网络层接收到请求信息之前进行了最后一次请求的信息检查。这就是说，请求会存储在一个

缓冲区并且在检查请求信息的例程时执行。当服务器#2 处理该条请求时，它确定了它不需要应答，因为它不支持该条请求信息。如图所示，这有可能在服务器#1 完成应答信息之后或是在客户机下一条请求信息之后发生。

g) 客户机在所有期待的应答信息完成之后，会立即发送下一条请求。

h) 服务器通过 N_USData.ind 指示请求信息的完成。但仅仅在快速服务器#1 中进行，因为在服务器#2 不处理最近一条信息。

i) 客户机新的请求的完成通过 N_USData.con 指示。

图 11 描述了客户机（基于图 10 说明的通信情况）$P3_{CAN_Functional}$定时处理。除此之外图 11 显示了客户机功能地址（0x3E）的请求。在$S3_{Client}$ 超时且$P3_{CAN_Functional}$活动时（请求将等待$P3_{CAN_Functional}$超时）。

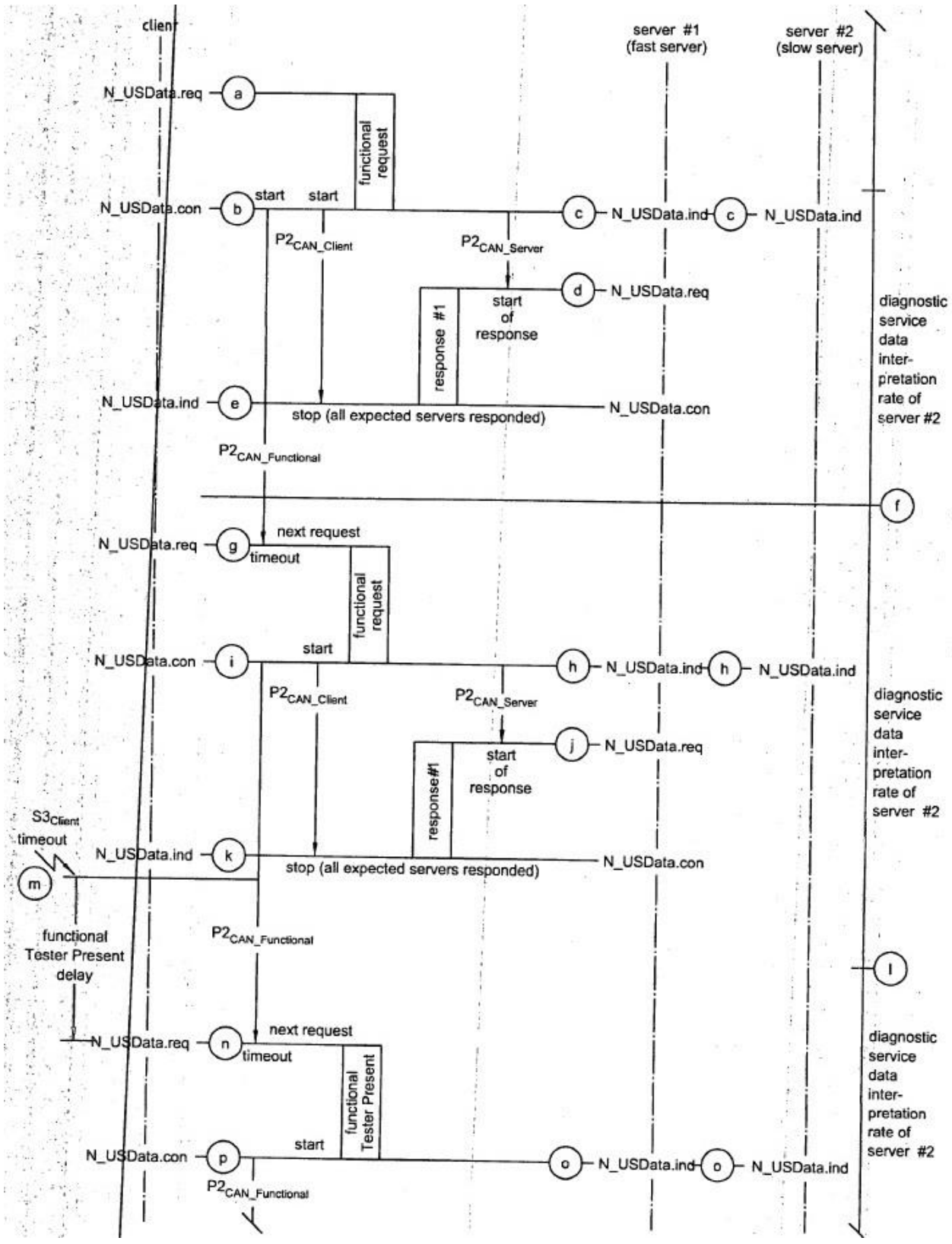


图 11——功能地址请求信息间隔时间最小值（$P3_{CAN\_Functional}$）

a) 客户端诊断应用层通过发送 N_USData.req 至网络层开始发送功能地址请求信息。网络层传递请求只服务器。

b) 客户端通过 N_USData.con 指示请求信息的完成。客户机开启 $P2_{CAN_Clien}$定时器并且开启

$P3_{CAN_Functional}$定时器。

c) 服务器通过 N_USData.ind 指示请求信息的完成。

d) 对于请求的信息，假定只有服务器#1 支持请求信息，也就是说服务器#2 不会应答信息。服务器#1 是快速服务器，能很快处理完请求的信息并在$P2_{CAN_Server}$时间内发送应答信息。

e) 客户机接收到应答信息。这通过 N_USData.ind 指示。客户机仅仅期待服务器#1 的应答信息，因此它停止$P2_{CAN_Client}$定时器。

f) 服务器#2 是慢速服务器，并且在一段时间内（诊断服务数据解读时间）解读请求信息，最坏的情况下，在网络层接收到请求信息之前进行了最后一次请求的信息检查。这就是说，请求会存储在一个缓冲区并且在检查请求信息的例程时执行。当服务器#2 处理该条请求时，它确定了它不需要应答，因为它不支持该条请求信息。

g) 尽管客户机接收到了功能地址请求信息所有期待的应答信息,它仍要等待$P3_{CAN_Client}$超时之后才允许发送下一条请求信息。在$P3_{CAN_Client}$超时的时刻，客户机发送下一条请求信息。

h) 新的请求信息服务器中通过 N_USData.ind 指示。并服务器#1 立即处理，而服务器#2 下一次检查请求信息例程中处理该请求。

i) 客户机通过 N_USData.con 指示新的请求的完成，并且开启$P3_{CAN_Functional}$定时器。

j) 对于请求的信息，假定只有服务器#1 支持请求信息，也就是说服务器#2 不会应答信息。服务器#1 是快速服务器，能很快处理完请求的信息并在$P2_{CAN_Server}$时间内发送应答信息。

k) 客户机接收到应答信息。这通过 N_USData.ind 指示。客户机仅仅期待服务器#1 的应答信息，因此它停止$P2_{CAN_Client}$定时器。

l) 服务器#2 是慢速服务器，并且在一段时间内（诊断服务数据解读时间）解读请求信息，最坏的情况下，在网络层接收到请求信息之前进行了最后一次请求的信息检查。这就是说，请求会存储在一个缓冲区并且在检查请求信息的例程时执行。当服务器#2 处理该条请求时，它确定了它不需要应答，因为它不支持该条请求信息。

m) 客户机$S3_{Client}$ 定时器超时，促使客户机发送不需服务器应答的功能地址（0x3E）请求信息。在这种情况下，$P3_{CAN_Functional}$此时仍然活动着，（0x3E）的发送应当到$P3_{CAN_Functional}$超时时发送。

n) 当$P3_{CAN_Functional}$定时器超时的时候，客户机可以通过 N_USData.req 发送功能地址（0x3E）请求。

o) 服务器通过 N_USData.ind 指示（0x3E）请求信息的接收。

p) 客户机通过 N_USData.con 指示（0x3E）请求的完成，并启动$P3_{CAN_Functional}$定时器。

图 12 描述了客户机$P3_{CAN_Physical}$定时器的操作。该图显示了不需应答的物理地址请求的发送操作及$S3_{Client}$ 超时时功能地址（0x3E）请求信息。

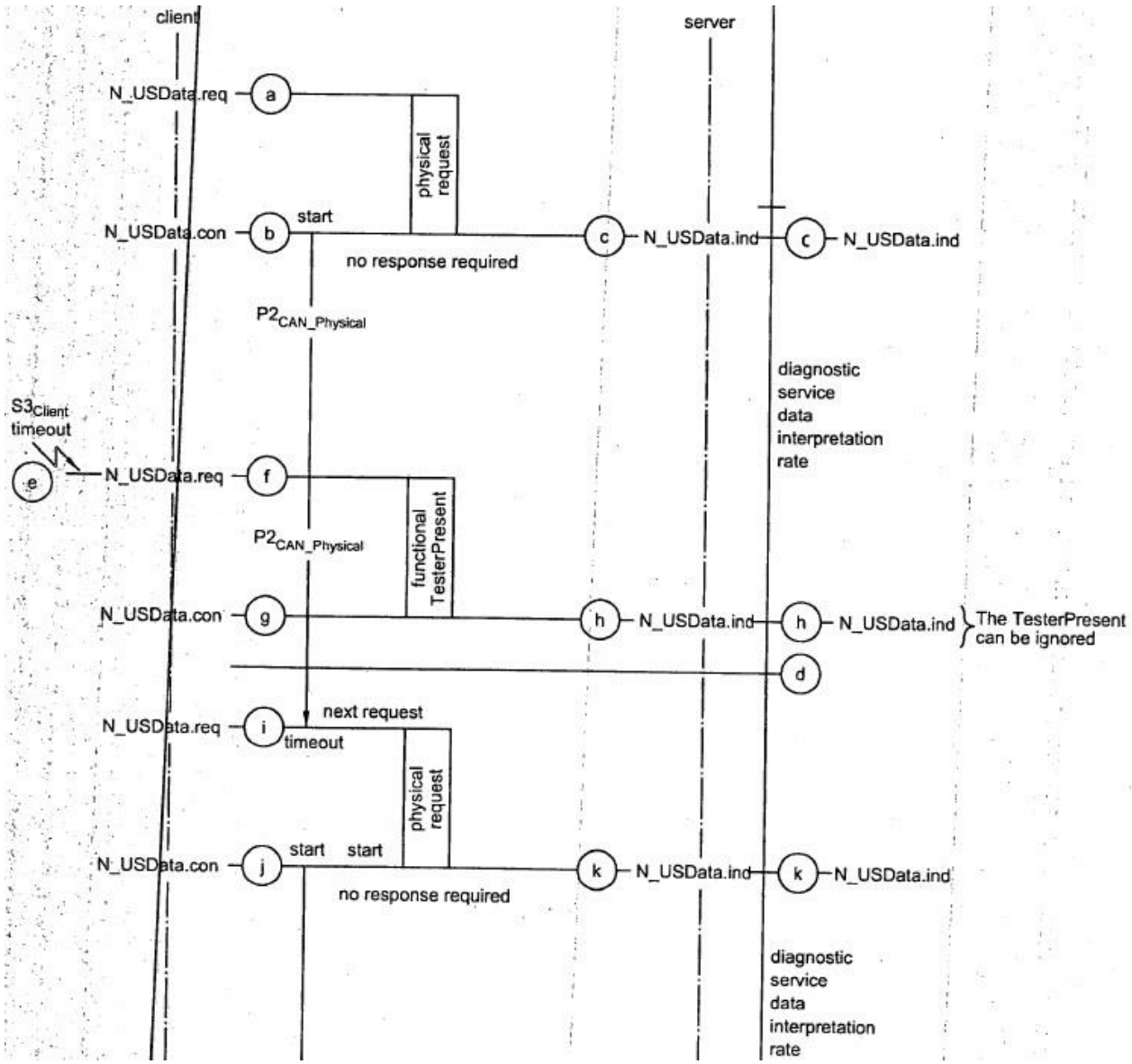


图 12——物理地址通信间隔最短时间 $P3_{CAN\_Physical}$

a) 客户端诊断应用层通过发送 N_USData.req 至网络层开始发送物理地址请求信息。网络层传递请求只服务器。

b) 客户端通过 N_USData.con 指示请求信息的完成。客户机开启 $P3_{CAN_Physical}$定时器。由于不需要应答信息，因此，客户机不需要开启 $P2_{CAN_CIient}$定时器。

c) 服务器通过 N_USData.ind 指示请求信息的完成。在任何非默认会话期间，$S3_{Server}$ 定时器此刻是停止的。

d) 服务器在一定时期内（诊断服务数据解读时间）解读请求。在下一次检查请求例程中请求被处理。在非默认会话期间，服务的完全执行会重置 $S3_{Server}$ 定时器。

e) 客户机 $S3_{CIient}$ 定时器超时，促使客户机发送功能地址（0x3E）请求信息，不需服务器的应答。

f) 假定$P3_{CAN_Functional}$定时器此时没有活动，也就是说请求被立即发送。

g) 客户机通过 N_USData.con 指示（0x3E）请求信息的完成。

h) 服务器通过 N_USData.ind 指示（0x3E）请求信息得接收。此刻，先前接收到的物理请求仍然在服务器端挂起（还没有处理）并且$S3_{Server}$ 定时器停止。因此，接收到的（0x3E）请求信息会被服务器忽略。

i) 当$P3_{CAN_Physical}$定时器在客户机超时，客户机会通过发送 N_USData.req 发送下一条物理地址请求信息至网络层。

j) 客户机通过 N_USData.con 指示物理地址请求信息的完成。客户机现在重新开启$P3_{CAN_Physical}$定时器。由于不需应答信息，因此客户端不启动$P2_{CAN_Client}$定时器。

k) 服务器通过 N_USData.ind 指示请求信息的完成。在任何非默认会话情况下，$S3_{Server}$ 定时器此刻停止。

6.3.5.4 主动提供的应答信息

服务器主动提供的应答信息要么是周期性例程（见服务 ReadDataByPeriodicIdentifier in 9.3.4）或者配置引发的，例如 DTC 状态的变化或者一个日期标识的改变（见服务 ResponseOnEvent in 9.2.8）。

所有主动提供的应答信息服务器都不应当重启$S3_{Server}$ 定时器。这在周期性信息传输或者时间触发的事件中时间的时间间隔比$S3_{Server}$ 短的情况下，有效避免了诊断会话的锁死。$S3_{Server}$ 定时器只应当在处理一条请求信息并发送最后结果应答信息（例如，初始肯定应答指示一个请求成功执行）的时候被重置。

6.3.6 出错的处理

应用层以及客户机和服务器在物理通信、功能通信期间的会话管理出错的处理应当按照表 7、表 8。假定客户机和服务器都按照该部分 15765 协议进行应用层及会话层的定时处理。

表 7——客户机错误处理

通信阶段	客户端错误类型	客户机处理	
		物理通信	功能通信
请求发送	网络层的 N_USData.con 指示否定结果值	客户机在 $P3_{CAN\_Physical}$ 时间之后，有出错指示，应当重发最后的请求 重启 $S3_{Client}$ （由于 $S3_{Client}$ 在请求发送时停止了）	客户机在 $P3_{CAN\_Physical}$ 时间之后，有出错指示，应当重发最后的请求
$P2_{CAN\_Client}$ $P2^{*}_{CAN\_Client}$	超时	客户机重新发送最近的请求信息。重启 $S3_{Client}$ （由于 $S3_{Client}$ 在请求发送时停止了）	这里客户机不知道多少服务器应答，这就是指示客户机不再有应答信息了。不用再重复请求信息了。 客户机在进一步请求之前，应当完全接受到所有的应答信息。
			这里客户机知道有多少服务器应答，这就是指示客户机不是所有的服务器都应答。 客户机在完全接收到所有应答信息之时发生了超时，应当重新请求信息。
应答接收	N_USData.ind 网络层否定结果值	客户机重新发送最近的请求信息。重启 $S3_{Client}$ （由于 $S3_{Client}$ 在请求发送时停止了）	客户机在完全接收到所有应答信息之时，出错，应当重新请求发送信息。
客户机出错处理运行最多 2 次，也就是说，最坏情况下，请求服务的发送只能是 3 次。			

表 8——服务器出错处理

通信阶段	服务器错误类型	处理
请求接收	网络层 N_USData.ind 指示否定结果值	重启 $S3_{Server}$ 定时器（由于它在接收到先前首帧指示时停止了），服务器应当忽略该请求。
$P2_{CAN\_Server}$ $P2_{CAN\_Client}$ $P2^{*}_{CAN\_Client}$	超时	N/A
应答发送	网络层 N_USData.ind 指示否定结果值	重启 $S3_{Server}$ 定时器（由于它在接收到先前的请求信息时停止了）。服务器不应当重新发送该应答信息。

7 网络层接口

7.1 概述

该部分的 ISO 15765 协议使用 ISO 1576502 定义的网络层服务进行诊断信息的收发。本节定义应用层协议数据单元（A_PDU）到网络层协议数据单元（N_PDU）的映射。
注意：网络层的服务用语应用层及诊断会话管理的定时。（见 6.3）

7.2 流控 N_PCI 参数定义

客户机 Stmin 参数不应该使用 0xF1-0xF9 的值。这些 Stmin 参数值应汽车制造商要求服务器应当支持。

7.3 信息发送的 A_PDU 到 N_PDU 的映射

应用层协议数据单元的参数按照下表 9 所示映射到网络层协议数据单元。它用于定义客户机/服务器诊断服务信息的请求/应答。
网络层向应用层的(N_USData.con)成功发送确认服务。应用层是需要这项服务，因为它需要在请求/应答完成时立即进行另外的动作（例如 ECU 重启，波特率调整等）。
表 9——ServiceName.request/ServiceName.response A_PDU 到 N_USData.request N_PDU

A_PDU 参数（应用层协议数据单元）	说明	N_PDU 参数（应用层协议数据单元）	说明
A_SA	应用层源地址	N_SA	网络层源址
A_TA	应用层目标地址	N_TA	网络层目标地址
A_Tatype	应用层目标地址类型	N_Tatype	网络层目标地址类型
A_RA	应用层远程地址	N_AE	网络层地址扩展
A_PCI.SI	应用层协议控制信息服务代码	N_Data[0]	网络层数据
A_Data[0]-A_Data[n	应用层数据	N_Data[1]N_Data[n+1	网络层数据

7.4 信息接收的 N_PDU 到 A_PDU 的映射

网络层协议数据单元的参数按照下表 9 所示映射到应用层协议数据单元。用于定义接收到的诊断请求/应答的确认/指示。
网络层对接收到首帧 N_PDU (N_USDataFirstFrame.ind)时指示不直接到应用层，因为它仅仅用于应用层定时（见 6.3）。因此没有 N_USDataFirstFrame.in N_PDU 到 A_PDU 的映射的定义。
表 10————N_USData.ind N_PDU 到 ServiceName.conf/ServiceName.ind A_PDU 的映射

N_PDU 参数（应用层协议数据单元）	说明	A_PDU 参数（应用层协议数据单元）	说明
N_SA	网络层源址	A_SA	应用层源地址
N_TA	网络层目标地址	A_TA	应用层目标地址
N_Tatype	网络层目标地址类型	A_Tatype	应用层目标地址类型
N_AE	网络层地址扩展	A_RA	应用层远程地址
N_Data[0]	网络层数据	A_PCI.SI	应用层协议控制信息服务代码
N_Data[1]N_Data[n+1	网络层数据	A_Data[0]-A_Data[n	应用层数据

8 标准的诊断 CAN 标识

8.1 法规 OBD 的 11 位 CAN 标识

法规 OBD 的 11 位 CAN 标识也用于扩展的 CAN 诊断（例如功能请求 CAN 标识能用于功能地址（0x3E）请求信息保持非默认会话处于激活状态。）

如果 ISO 15765-4 说明的 11 位的 CAN 标识在扩展的诊断中重新使用，适用如下要求：

a) ISO 15765-4 协议的网络层定时参数同样适用于扩展的诊断；

b) DLC（CAN 数据长度码）应当设置为 8 并且 CAN 帧应当包含 8 字节（未使用的字节也应当填充）；

注意：ISO 15765-4 允许最大 8OBD 相关服务器，为 8 个服务器定义了 11 位 CAN 标识。

8.2 法规 29 位 OBD 的 CAN 标识

法规的 29 位 CAN 标识应按照 ISO 15765-2 说明的标准固定的地址格式，同样能用于扩展的诊断。

如果 ISO 15765-4 说明的 29 位的 CAN 标识在扩展的诊断中重新使用，适用如下要求：

a) ISO 15765-4 协议的网络层定时参数同样适用于扩展的诊断；

b) DLC（CAN 数据长度码）应当设置为 8 并且 CAN 帧应当包含 8 字节（未使用的字节也应当填充）；

注意：表中给出的 CAN 标识符按照 ISO 15765-2 协议优先级信息使用默认的值。

8.3 扩展的诊断 29 位 CAN 标识

8.3.1 概述

本部分说明使用 29 位 CAN 标识的标准地址及路由的概念。主要使用了最流行的网络协议（IP）的握手机制。因此地址及路由的算法可用于不同子网位置的节点的通信及路由。

准地址及路由的概念遵循如下的特征：

——网络结构最灵活的设计操作

——完全定制的网络及节点地址

——CAN 控制器硬件过滤特征通过分配合适的网络及节点地址优化。

——网关需要知道与它连接的子网的网络地址，而不需要所有子网成员的地址。

下面描述了 CAN 标识符结构的技术细节，包括地址，子网掩码。也包括了对路由及广播的算法的详细描述。

8.3.2 29 位 CAN 标识符结构

本文档描述的 29 位 CAN 标识符结构与如下协议是兼容的。有 ISO 15765-2, ISO 15765-3, ISO 15765-4 及 SAE J1939-21.因此 SAE J1939-21 定义的 29 位 CAN 标识结构中 25 位的编码（保留/扩展数据页）和 24 位编码（数据页）应当确定该 CAN 标识或 CAN 帧是 J1939 的还是 ISO 15765 的。这对汽车网络设计者根据他的需求及对 SAE J1939 和 ISO 15765 协议的使用，定制非诊断的信息及相关 CAN 标识是重要的。

8.3.2.1 SAE J1939 的 29 位 CAN 标识符结构

关于 SAE J193929 位 CAN 标识符格式见如下表 11

表 11——SAE J1939 的 CAN 标识符结构

29 位 CAN 标识符					
28、27、26	25	24	23-16	15-18	7-0
优先级	保留/扩展数据页	数据页	PDU 格式	PDU-特定域（目标地址或 PDU 格式扩展）	源地址(独有的源地址）

8.3.2.2 ISO 15765 的 29 位 CAN 标识符结构

表 12 显示了 ISO 15765 的 CAN 标识符结构与 SAE J1939 格式的区别。
25 位——SAE J1939 保留/扩展数据页，ISO 15765 使用扩展数据页
24 位——SAE J1939 数据页 ，ISO 15765 数据页
因此，ISO 15765 格式与 SAE J1939 格式的 29 位 CAN 标识能在同一个 CAN 总线上互不影响的共存。

表 12——ISO 15765 的 CAN 标识符结构

29 位 CAN 标识					
28-26	25	24	23，22	21-11	10-0
优先级	扩展数据页	数据页	服务类型（TOS）	源地址	目标地址
	编码见 8.3.2.4		编码见 8.3.2.5	源地址（独有的源地址）	目标地址（独有的目标地址）

8.3.2.3 优先级域

SAE J1939 定义的优先级域用于 CAN 总线的仲裁机制。由于 CAN 标识符不再能自由分配（源地址和目的地址包含在 CAN 标识符中),CAN 信息优先级由发送者分配并间接由接收者分配。存在 8 种不同的优先级。
优先级 6 分配至诊断请求信息 / 帧。

8.3.2.４ 扩展的数据页及数据页域

扩展的数据页及数据页位决定了使用哪一种 2 9 位的 CAN 标识。见表 1 3 编码的说明

表 1 3 ——扩展数据页及数据页域

扩展的数据页位 2 5	数据页位 2 4	说明
0	0	SAE J1939 定义或厂家定义的“标准通信信息”
0	1	SAE J1939 定义或厂家定义的“标准通信信息”
1	0	SAE J1939 定义或厂家定义的“标准通信信息”
1	1	ISO 15765 定义的

8.3.2.5 服务类型（TOS）域

服务类型域用于表述一个节点不需要分配不同地址的情况下，分配不同项服务。因此，8 种不同的服务类型能同时分配给单个的目标地址。不同服务类型的定义见表 14

表 14——服务类型的定义（TOS）

位 23	位 22	服务类型（TOS）	说明
0	0	ISO 保留	该位组合为 ISO 为将来保留
0	1	OEM-定义的信息	该位组合指示信息为 OEM 特定的，ISO 15765-3 及以前的协议信息能通过相同的网络但不同的协议信息混合使用在一个服务器上。
1	0	网络控制信息 协议/网络管理	该位组合指示帧包含的网关收发数据用于支持当前子网状态的信息（例如，网络无法到达/网络超载）和节点信息（例如，主机无法到达）
1	1	ISO 15765-3 定义的 信息	该位组合包含了节点 ISO 15765 定义的诊断服务。CAN 帧用户数据字节包括诊断请求（ISO 15765-3）使用网络层服务及 ISO 15765-2 定义的传输层

8.3.2.6 源地址

源地址包含发送实体地址。该信息保证了正确仲裁以及被接收者用于回复信息。源地址结构见 8.3.3 描述。

8.3.2.7 目标地址

目标地址包含接收实体的地址信息。这应是一单独节点，广播地址或通用广播。网关使用目标地址决定 CAN 帧是否应当路由到另外一条 CAN 总线上。该目标地址结构见 8.3.3 所述。

8.3.3 地址结构

8.3.3.1 概述

目标地址及源地址都编码在 29 位 CAN 标识符中，并且每个长度为 11 位。如下所示，字母“X”和“Y”代表可变参数。

8.3.3.2 地址的定义

一个地址包含两个部分

a) **网络地址**

网络地址部分包含第一个连续的位“X”地址并且决定了一个节点所在的网络。同一物理总线上的节点应当分配同一个网络地址。网络地址部分不应当将所有的位置为 1.因此，最小的网络地址长度应为 2 个位。最大长度应为 9 个位因为因为至少需要 2 个位提供固定节点地址。最大的子网数量可根据如下计算：

$2^X - 1$（X 代表使用到网络地址的位的个数）

b) **节点地址**

节点地址部分包含了地址中剩下的连续的位“Y”（Y=11-X），并决定了子网中具体的节点。在子网中应当是独有的。所有的位都置位 0 或 1 是不允许的。所以最小节点地址长度为 2 个位，最大为 9 个位。子网中最多节点个数根据如下公式计算：

$2^Y - 2$（Y 代表使用到节点地址的位的个数）

分配给节点独有的地址应当存储在节点的内部存储器中。一个节点接收目标地址域为该节点地址的的信息。

表 15 展示了源地址和目标地址的一个例子。发送及接收节点不在同一个子网中。

表 15——源地址和目标地址的一个例子

29 位 CAN 标识																												
28	27	26	25	24	23	22	21										11	10										0
优先级 0x6			ISO 15765 格式		服务类型 ISO 15765 信息		源地址 0x2ED											目的地址 0x32F											
1	1	0	1	1	1	1	0	1	0	1	1	1	0	1	1	0	1	0	1	1	0	0	1	0	1	1	1	1

8.3.3.3 子网掩码

子网掩码为网络地址及节点地址分配。

子网掩码长度为 11 位（与地址长度一致）。子网掩码的值通过设置开始连续的位“X”为 1 分配。将网络地址部分设置为 1，将节点地址的部分设置为 0.（见表 16 和表 17 发送与接收者的子网掩码的例子）

由于固定的子网掩码长度及一开始的连续的位“X”设置为 1，只有这些位置位 1 而不是所有位。因此需要一个短记号定义子网掩码。

表 16——发送端子网掩码例子

子网掩码										
10	9	8	7	6	5	4	3	2	1	0
0X7C0(短的记号/5)										
网络地址部分					节点地址部分					
1	1	1	1	1	0	0	0	0	0	0

表 17——接收端子网掩码例子

子网掩码										
10	9	8	7	6	5	4	3	2	1	0
0X7C0(短的记号/5)										
网络地址部分						节点地址部分				
1	1	1	1	1	1	0	0	0	0	0

每一个分配子网掩码的节点都应当存储在它内部存储器内。相同子网的节点分配相同的子网掩码。

8.3.3.4 网络地址

节点的网络地址现在可以通过分配地址及子网掩码计算出来。见表 18 和 19 发送者和接收者的例子决定了网络地址。

表 18——发送者网络地址

源地址										
位	10	9	8	7	6	5	4	3	2	1	0
地址：0x2ED	0	1	0	1	1	1	0	1	1	0	1
子网掩码：/5	1	1	1	1	1	0	0	0	0	0	0
网络地址：0x2C0	0	1	0	1	1	0	0	0	0	0	0

表 19——接收者网络地址

源地址											
位	10	9	8	7	6	5	4	3	2	1	0
地址：0x32F	0	1	1	0	0	1	0	1	1	1	1
子网掩码：/6	1	1	1	1	1	1	0	0	0	0	0
网络地址：0x320	0	1	1	0	0	1	0	0	0	0	0

为了描述子网掩码，网络地址及子网掩码按如下形式记录：
<网络层地址>/<短的子网掩码记录>
实例：
发送端子网：0x2C0/5
接收端子网：0x320/6
该信息被网关用来路由。

8.3.3.5 广播地址

8.3.3.5.1 通用广播地址（0x7FF）

通用广播地址允许在网络上所有节点广播信息。为了发送一个广播信息到整个网络，目标地址必须为 0x7FF(所有的位都设置为 1)。包含该目标地址的信息将会被所有网关路由。所有的网络节点都应当接收并处理地址为 0x7FF 的信息。

8.3.3.5.2 子网广播地址

子网的广播用于广播信息到特定子网上的节点。为了发送一条广播信息到某一特定子网上，该子网广播地址应当计算出来。通过将目标子网信息（网络地址及子网掩码）可实现。即将所有节点地址的部分设置为 1。见表 20 对于接收子网的子网广播的例子

表 20——接收子网的子网广播的例子

目标地址											
位	10	9	8	7	6	5	4	3	2	1	0
地址：0x32F	0	1	1	0	0	1	0	0	0	0	0
子网掩码：/6	1	1	1	1	1	1	0	0	0	0	0
网络地址：0x320	0	1	1	0	0	1	1	1	1	1	1

子网广播信息网关正常路由
所有的节点都必须接收网络地址与他们自身网络地址相同的信息，并且在目标地址域节点地址的部分所有的位都应设置为“1”。

8.3.4 信息接收

每一个子网的节点都将 CAN 帧中目标地址与它自己的地址相比较。如果匹配的话，包含的信息就传递至 OSI 模型相邻的上层进一步处理。

8.3.5 路由

8.3.5.1 概述

路由适用于当一个节点与另外一个节点不再一个子网上，因而 CAN 帧就需要从一个子网传递至另一个子网。者通过另外的节点，物理上连接到 CAN 帧接收子网及发送子网。因此，一个 CAN 帧从源子网到目的子网，可能通过几个网关

8.3.5.2 网络及子网结构

大体上，网络可按需求设计，需考虑如下几个条件：
——地址应当是唯一的。
——所有同一子网的节点都必须使用相同的子网掩码。
——所有同一子网的节点都必须使用相同的网络地址。
——当一个网络地址分配个一个子网时，在那个地址范围的网络地址都不应当分配个其它的网络，因为这会导致路由问题。
图 13 显示了连接到网关的四个子网的配置。3 个子网通过一个网关连接的，第 4 个子网通过另外的一个网关连接。

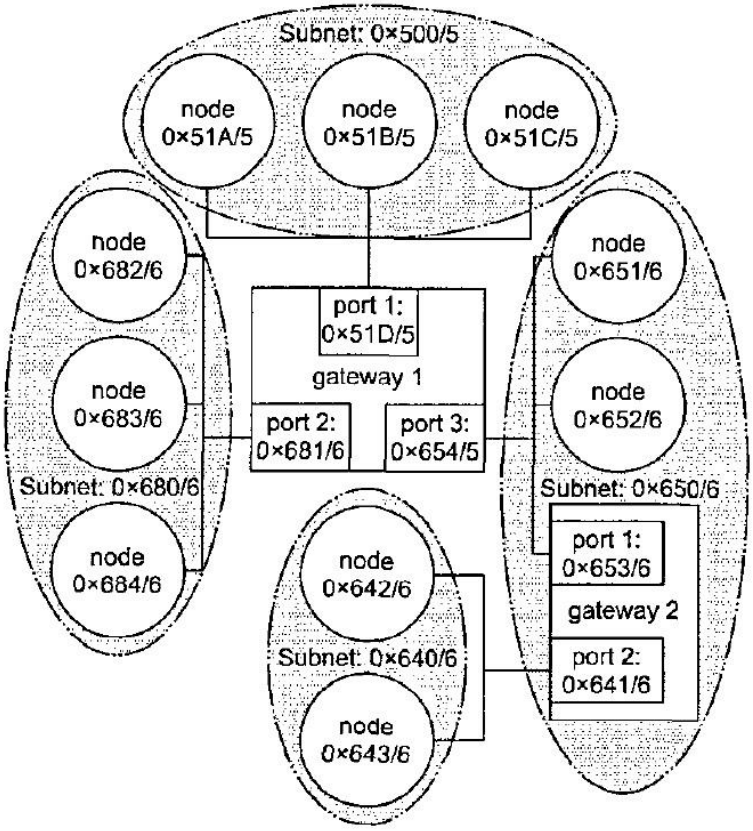


图 13——网络配置例子

8.3.5.3 网关及路由

8.3.5.3.1 说明

网关是连接多于一个子网的节点，由此能够将一个子网的 CAN 帧传递到另一个子网上。

8.3.5.3.2 端口

一个端口是网关连接到物理子网的接口。网关至少有两个端口，每一个端口都分配所在子网的网络地址及子网掩码。
见图 13 的配置有 2 个网关，网关 1 有 3 个接口，网关 2 有 2 个接口。

8.3.5.3.3 路由表

为了确定一个 CAN 帧是否需要被路由，需要生成一张路由表并存储在网关的存储器中。路由条目包含网络地址，子网掩码及能到达的子网的端口。该条目应当存有通过该网关每一个连接的子网（直接的或间接的）。
见表 21 所示的是图 13 的网络。通过对网络 640/6 和 650/6 的分级设计。路由表条目缩减

到一个条目 640/5。

表 21——路由表例子

子网（网络地址/子网掩码）	端口
网关 1	
500/5	1
680/5	2
640/5	3
网关 2	
500/5	1
680/6	1
650/6	1
640/6	2

8.3.5.3.4 路由算法

连接到不同的子网的网关从端口接收所有的信息。如果网关是一有地址的节点，那么直接连接到该网关端口，在所有地址范围中只有一个地址应当被分配。在合适的路由算法之前，有另外的对信息接收的检查。如果目标地址为 0x7FF，信息除了信息接收端口，被复制到所有端口，忽略正常的路由算法。

8.3.5.3.5 路由例子

见图 15 从地址 0x51A 的客户机到地址 0x642 服务器 CAN 帧传输的路由的例子，该例子使用表 21 的路由信息。

在接收到该信息时，下一步如下进行处理。

a) 网关 1

1） CAN-ID 分析：DA=0x642，见表 22 和表 23

表 22——网关 1 路由判定

路由描述	网络	端口
（0x642 逻辑且 0x7C0）= 0x640 != 500→非本地地址→路由	500/5	1

表 23——网关 1 路由分析

路由描述	网络	端口
（0x642 逻辑且 0x7E0）= 0x640 ！= 680→下一入口	680/6	2
（0x642 逻辑且 0x7C0）= 0x640 = 640→正确路径	640/5	3

2） 查该信息是否是发送到网关：0x642 != 0x654

3） 将信息送至端口 3

b) 网关 2

1）CAN-ID 分析：DA=0x642.见表 24 和表 25

表 24——网关 2 路由判定

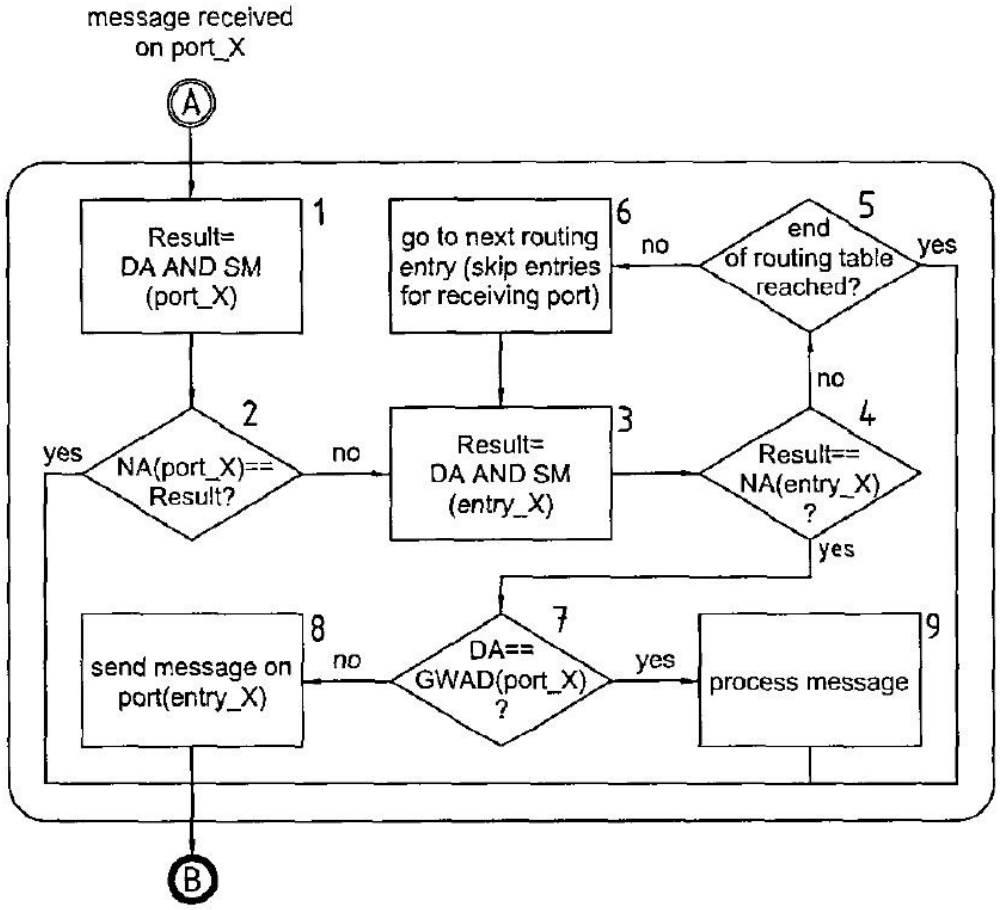
路由描述	网络	端口
（0x642 和 0x7C0）= 0x640 != 650→非本地地址→路由	650/6	1

表 25——网关 2 路由分析

路由描述	网络	端口
（0x642 逻辑且 0x7E0）= 0x640 != 640→正确路径	640/6	2

2）检查该信息是否发送到网关：0x642 != 0x641

3）将信息传递至端口 2



步骤：

A 从端口“X”接收到信息

1 将接收信息的目标地址与接收信息端口的子网掩码一位一位进行逻辑“且”操作。

2 将结果与接收信息的端口的网络地址进行比较。端口的网络地址要么存储在节点存储器内，要么通过端口的地址及子网掩码计算出来。如果结果与网络地址相等，接收到的信息是该端口子网的本地信息，并且不再做任何路由（B）。如果结果与端口网络地址不等，则需要进行路由分析。进行第 3 步。

3 将接收信息的目标地址与当前路由表入口的子网掩码一位一位进行逻辑“且”操作。

4 将结果与当前路由表入口的网络地址进行比较。如果匹配的话，算法从步骤 8 继续，否则算法从步骤 5 继续。

5 如果有另外的路由表入口，算法从步骤 6 继续，否则，不再做任何路由（B）。

6 选择了下一个路由表入口，算法跳到步骤 3 继续。

7 信息的目标地址与网关当前端口的地址进行比较。该步骤只有在网关是一个有地址的节点时才需要，否则算法直接跳到步骤 8.如果是网关当前端口的地址，算法从步骤 9 继续，如果目标地址与网关地址不同，算法从步骤 8 继续。

8 信息发送到路由表网络地址与目标地址匹配的入口端口上。

9 信息是发送到网关节点上的，并由网关应用层处理。

B 路由结束

Key

DA 目标地址

GWAD port_X 的网关地址

NA 网络地址

SM 子网掩码

entry_X　网关路由表入口#X
prot_X　网关的#X 端口

图 14 路由算法顺序图

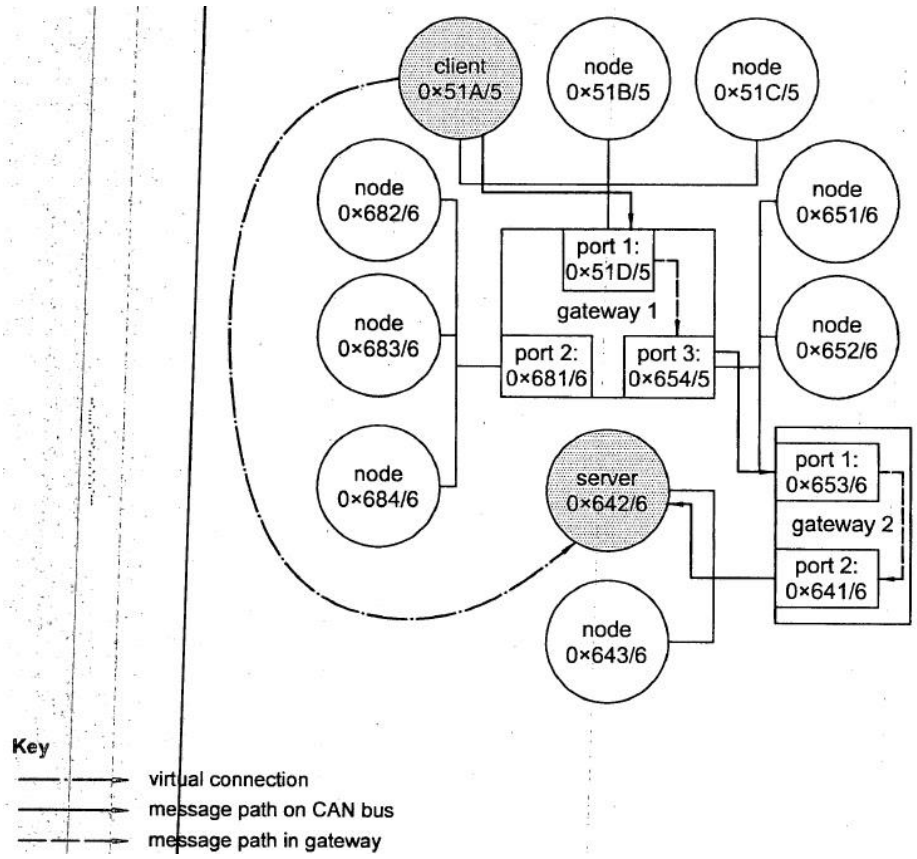


图 15——从客户机 0x51A 到服务器 0x642 的路由的例子

9 诊断服务实施

9.1 统一诊断服务总览

该部分定义了 ISO 14229-1 定义的诊断服务是如何适用于 CAN 的。对于每一个应用服务，都定义了可用的子功能及数据参数。

注意：子功能参数的定义考虑了 suppressPosRspMsgIndicatonBit 参数的最高有效位。该参数在 ISO 14229-1 中定义。

表 26 用于提供所有统一诊断服务的总览，它们适用于 CAN 诊断实施，表包含了可用服务总数。使用该部分 ISO 15765 协议实施 CAN 诊断的某些应用上可能限制了可使用服务的数量，并可将它们按应用范围/诊断会话（默认会话，编程会话等）进行归类。

表 26——CAN 诊断——统一诊断服务总览

诊断服务名称 （ISO 14229-1）	服务 ID 值 （16 进制）	子功能 支持	suppressPosRspMsgIndicatonBit =TRUE（1）;（无应答）支持(a)	子目录
诊断与通信管理功能单元				
DiagnosticSessionControl	10	是	是	9.2.1
ECUReset	11	是	是	9.2.2
SecurityAccess	27	是	是	9.2.3
CommnicationControl	28	是	是	9.2.4
TesterPresent	3E	是	是	9.2.5
SecuredDataTransmission	84	—	N/A	9.2.6
ControlDTCSetting	85	是	是	9.2.7
ResponseOnEvent	86	是	是	9.2.8
LinkControl	87	是	是	9.2.9
数据发送功能单元				
ReadDataByIdentifier	22	—	N/A	9.3.1
ReadMemoryByAddress	23	—	N/A	9.3.2
ReadScalingDataByIdentifier	24	—	N/A	9.3.3
ReadDataByPeriodicIdentifier	2A	—	N/A	9.3.4
DynamicallyDefineDataIdentifier	2C	是	是	9.3.5
WriteDataByIdentifier	2E	—	N/A	9.3.6
WriteMemoryByAddress	3D	—	N/A	9.3.7
存储数据发送功能单元				
ReadDTCInformation	19	是	是	9.4.1
ClearDignositicInformation	14	—		9.4.2
输入/输出控制功能单元				
InputOutPutControlByIdentifier	2F	—	N/A	9.5.1
例行的远程激活功能单元				
RoutineControl	31	是	是	9.6.1
上载/下载功能单元				
RequestDownload	34	—	N/A	9.7.1
RequestUpload	35	—	N/A	9.7.2
TransferData	36	—	N/A	9.7.3
RequestTransferExit	37	—	N/A	9.7.4

a 这是指 suppressPosRspMsgIndicatonBit = FALSE(0) 表示服务支持使用该子功能参数。系统设计者需保证如果客户机不需要一个应答信息[suppressPosRspMsgIndicatonBit = TRUE（1）]并且服务器需要超过 $P2_{CAN_Server}$的时间去处理请求信息时，客户机应当在连续请求之间插入足够的时间。有可能的一个情况是，当服务器不执行请求的动作，也不指示任何原因至客户机。

9.2 诊断与通信控制功能单元

9.2.1 诊断会话控制（DiagnosticSessionControl）（10hex）服务

表 27 定义了适用于 CAN 诊断服务的子功能参数

表 27——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
01	defaultSession	U	DS
02	ECUProgrammingSession	U	ECUPS
03	ECUExtendedDiagnosticSession	U	ECUEDS

表 28 和 29 定义了应答信息数据参数结构,sessionParameterRecord 适用于 CAN 诊断实施。

表 28——会话参数记录定义

记录的位位置	说明	Cvt	16 进制值	助记忆法
	SessionParameterRecord[]#1={			SPREC_
#1	$P2_{CAN\_Server\_max}$(高字节)	M	00-FF	P2CSMH
#2		M	00-FF	P2CSML
#3	$P2_{CAN\_Server\_max}$（低字节）	M	00-FF	P2ECSMH
#4		M	00-FF	P2ECSML
	$P2*_{CAN\_Server\_max}$（高字节）			
	$P2*_{CAN\_Server\_max}$（低字节）}			

表 29——会话参数记录内容定义

参数	说明	#占用字节	解决	最小值	最大值
$P2_{CAN\_Server\_max}$	服务器支持默认的定时用于激活诊断会话 $P2_{CAN\_Server\_max}$	2	1ms	0ms	65535ms
$P2*_{CAN\_Server\_max}$	服务器支持扩展的（NRC 78hex）用于激活诊断会话 $P2*_{CAN\_Server\_max}$	2	10ms	0ms	655350ms

9.2.2 ECU 复位（ECUReset）(11hex）服务

表 30 定义了该项 CAN 服务的子功能参数

表 30——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
01	hardReset	U	HR
02	keyOffOnReset	U	KOPPONR
03	softReset	U	SR
04	enableRapidPowerShutDown	U	ERPSD
05	disableRapidPowerShutDown	U	DRPSD

9.2.3 安全访问（SecurityAccess）(27hex)服务

表 31 定义了该项 CAN 服务的子功能参数

表 31——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
01	requestSeed	U	RSD
02	sendKey	U	SK
03,05 07-5F	requestSeed	U	RSD
04,06 08-60	sendKey	U	SK

9.2.4 通信控制服务（CommunicationControl）(28hex)

表 32 定义了该项 CAN 服务的子功能参数

表 32——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
00	enableRxAndTx	U	ERXTX
01	enableRxAndDisableTx	U	ERXDTX
02	disableRxAndEnableTx	U	DRXETX
03	disableRxAndTx	U	DRXTX

表 33 定义了 CAN 服务实施可用的数据参数

表 33——数据参数定义——通信类型

Hex (位 1-0)	描述	Cvt	助记忆法
01b	application	U	APPL
10b	networkManagement	U	NWM
位 1 - 0 适用任何组合。每一个位代表一种通信类型，可能每次都不止初始化一种通信类型。			

9.2.5 测试仪现场服务（TesterPresent）(3E hex)

表 33 定义了该项 CAN 服务的子功能参数

表 33——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
00	zeroSubFunction	M	ZSUBF

9.2.6 安全数据传输服务（SecuredDataTransmission）(84 hex)

对于该 CAN 服务实施，既没有额外的需求也没有限制。

9.2.7 控制故障吗信息设置服务（ControlDTCSetting）(85 hex)

表 35 定义了该项 CAN 服务的子功能参数

表 33——子功能参数定义

Hex (位 6-0)	描述	Cvt	助记忆法
01	on	M	ON
02	off	M	OFF

9.2.8 基于事件应答服务（ResponseOnEvent）(86 hex)

CAN 实施须满足以下相关条件

a) 大量的 ResponseOnEvent 服务都伴随着不同的启动和停止诊断服务的要求（不同的时间类型 EventTypes,应答记录服务 serviceToRespondTo-Records）。

b) 当 ResponseOnEvent 服务激活时，服务器能够同时处理诊断请求及相应的应答信息。这应当完成一组请求/应答 CAN 标识。见图 16.如果相同的 CAN 请求/应答标识用于诊断通信中以及 serviceToRespondToresponses 服务。如下限制需适用：
   1）在一个事件发生之后，服务器应当忽略即将到来的诊断请求。并且开始运行 serviceToRespondTo-response 服务，直到该服务完成。
   2）在客户机发送一个诊断请求后，接收到任何应答信息，应答信息应当按照可能的应答服务 serviceToRespondTo-responses 及期望的诊断应答分类。
   3）如果应答是一个 serviceToRespondTo-response（由基于事件应答服务的应答）。客户机应当在 serviceToRespondTo-response 完全接收到之后，重复该请求。
   4）当应答不确定时(例如,应答可能产生于一个事件的应答或者一个诊断请求的应答)，客户机应当将该应答同时作为一个 serviceToRespondTo-response 及诊断请求应答。客户机不应当重复该请求除非 NegativeResponseCode busyRepeatRequest(21hex)（见否定应答码，ISO 14229-1 的定义。）
c) ResponseOnEvent 服务只有在激活的诊断会话中可用的诊断服务中使用。
d) 当 ResponseOnEvent 服务处于激活状态时，诊断会话中任何的改变都回中止当前的 ResponseOnEvent。例如，如果一个 ResponseOnEvent 服务在阔真的诊断会话建立起来了，它在服务器转换到默认会话时应当中止。
e) 如果 ResponseOnEvent（0x86）服务在默认会话时建立了，如下应当适用：
   1）事件类型子功能参数位 6 设置为 0（不存储事件），那么该事件应当在服务器关电时中止，并且服务器不应当在重启或开电时继续一个 ResponseOnEvent 诊断服务。(例如，ResponseOnEvent 中止)。
   2）事件类型子功能参数位 6 设置为 1（存储事件），它会重新发送 serviceToRespondTo-responses 按照 ResponseOnEvent 在重启电后。

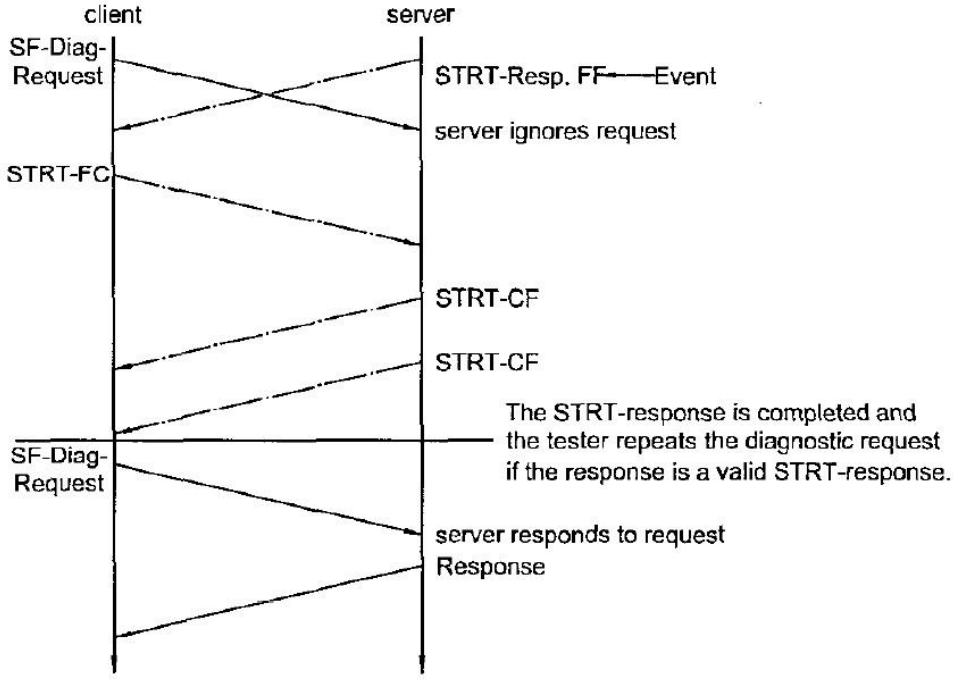


图 16——发生时同时存在的请求

f) 子功能参数值 responseRequired = “no”应当只用于事件类型 eventType = stopResponseOnEvent, startResponseOnEvent 或 clearResponseOnEvent。当检测到指定事件时，服务器应当总是返回一个事件触发的应答。
g) 服务器应当最后发送肯定应答用于指示 ResponseOnEvent(0x86)服务到达了限定事件窗，除非出现以下某种情况：

1）如果事件类型没有建立 ResponseOnEvent，例如，stopResponseOnEvent，
startResponseOnEvent， clearResponseOnEvent 或 reportActivatedEvents；
2）如果事件窗建立起来了
——如果在事件窗关闭之前，服务已经处于非激活状态。
——子功能参数事件类型位 6 设置为 0（不存储），并且系统关电或重启

h) 当指定的事件监测到时，服务器应当以合适的 serviceToRespondTo-response 立即应答。立即应答信息应当不会破坏其它任何诊断请求及已经处于发送状态的应答（例如，该 serviceToRespondTo-response 应当延迟直到当前信息发送完之后。见图 17）。

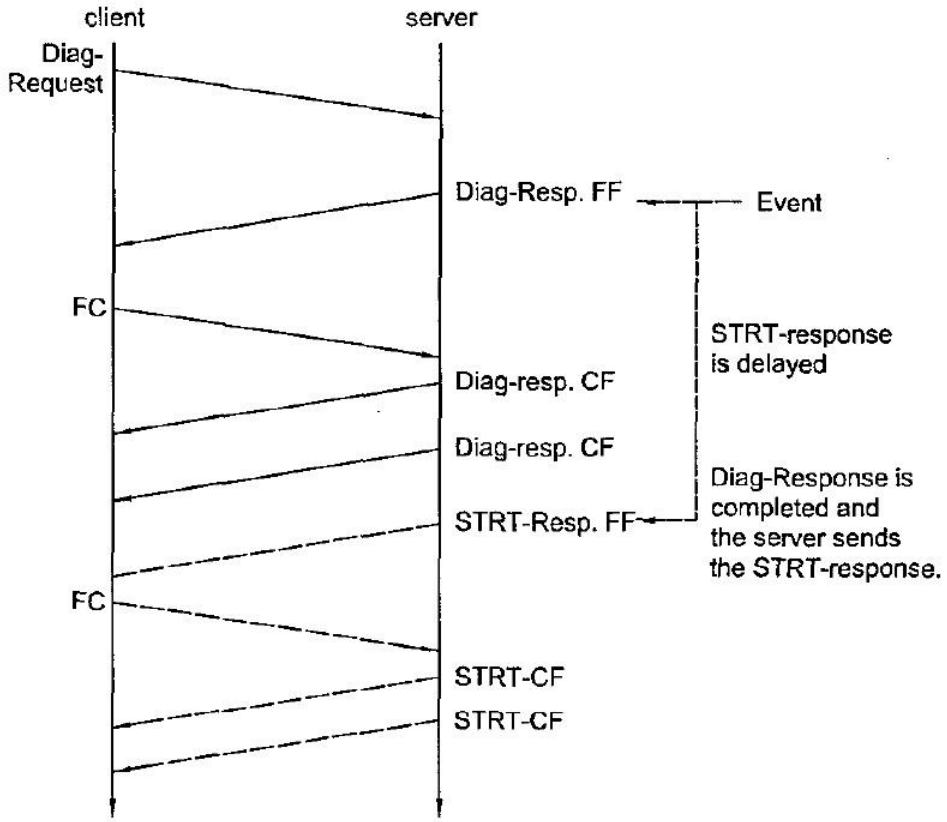


图 17——在一条信息处理过程中事件发生

i) ResponseOnEvent 服务仅适用于短暂的事件及条件。服务器应当在每一事件发生时，返回一个应答。在一个时期内持续的情况，应答服务应当在启动发生的时候执行。该事件类型的定义主要是为了避免 serviceToRespondTo-responses 较高频率发生，因而需采取合适的手段进行处理。在 serviceToRespondTo-responses 的最小时间间隔属于事件类型（eventTypeRecord）的一部分（汽车厂家指定）。

表 36 和 37 定义了 CAN 服务实施的子功能参数

表 38 定义了 CAN 服务实施的数据参数

表 36——事件类型子功能位 6 定义——存储状态

位 6 值	说明	Cvt	助记符
0	doNotStoreEvent	M	DNSE
1	storeEvent	U	SE

表 37——子功能参数定义

（位 5-1）hex	说明	Cvt	助记符
00	stopResponseOnEvent	U	STPROE
01	onDTCStatusChange	U	ONDTCS
02	onTimerInterrupt	U	OTI
03	onChangeOfDataIdentifier	U	OTI
04	reportActiveEvents	U	OCOCID
05	startResponseOnEvent	U	STRTROE
06	clearResponseOnEvent	U	CLRROE
07	onComparisonOfValues	M	OCOV

表 38——数据参数定义——serviceToRespondToRecord.serviceId

推荐的服务（ServiceToRespondTo）	请求服务标识（SId）
ReadDataByIdentifier	22 hex
ReadDTCInformation	19 hex
RoutineControl	31 hex
InputOutputControlByIdentifier	2F hex

9.2.9 连接控制（LinkControl）(87 hex)服务

表 39 定义了 CAN 服务实施的子功能参数

表 39——子功能参数定义

（位 6-1）hex	说明	Cvt	助记符
01	verifyBaudrateTransitionWithFixedBaudrate	U	VBTWFBR
02	verifyBaudrateTransitionWithSpecificedBaudrate	U	VBTWSBR
03	transitionBaudrate	U	TB

9.3 数据发送功能单元

9.3.1 通过标识符读数据服务（ReadDataByIdentifier）(22hex)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.3.2 通过地址读内存（ReadMemoryByAddress）(23 hex)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.3.3 通过标识符读刻度数据（ReadScalingDataByIdentifier）(24hex)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.3.4 通过周期的标识读数据（ReadDataByPeriodicIdentifier）(25hex)

在 ISO 14229-1 定义了两种类型的应答信息用于该服务，如下所示：

——应答信息类型#1（包括服务标识，periodicDataIdentifier 和 periodicDataIdentifier 的回应）：该类型的应答信息映射到 USDT 信息，使用与其它的 USDT 相同的 CAN 标识应答。单个 periodicDataIdentifier 的 USDT 信息不应超过一个 CAN 帧，也就是说，完整的 USDT 应答信息应当在一个单帧的 N_PDU 内。

——应答信息类型#2 包括（periodicDataIdentifier 和 periodicDataIdentifier 的数据）：该类型应答信息映射到 UUDT 信息，使用不同的 CAN 标识作为 USDT 的应答信息。单个的 periodicDataIdentifier 的 UUDT 信息应当在单个 CAN 帧范围内。

两个应答类型对客户机服务器的要求如下表 40 和 41.

表 40——周期发送——对于应答类型#1 信息的要求

信息类型	客户请求要求	服务器应答要求	服务器约束
USDT 使用相同的 CAN 标识用于诊断通信及周期发送	无约束	对周期发送只有单帧应答 对于可能的新请求（非周期发送）有多帧应答	对于新过来的请求应当有较高优先级，周期发送应当延迟
			周期应答应当作为普通的 USDT 信息处理（使用协议控制信息（PCI），服务标识（SId）和周期数据标识（periodicDataIdentifier)），并由服务器网络层处理。也就是说，当使用标准地址，最大有 5 字节 periodicDataIdentifier 的数据，当使用扩展地址，最大有 4 字节 periodicDataIdentifier 的数据。
			对于到来的多帧请求信息，所有周期发送机制在多帧请求的首帧或单帧请求的 N_USData.ind 被应用层出力时时，应当被延迟。一旦服务完成（包括发送最后结果应答信息），周期信息发送应当继续。

USDT： 未答复的拆分数据传输， ISO 15765-2 网络层，包括拆分数据传输的协议控制信息

UUDT ：未答复的未拆分数据传输，CAN 单帧，不包括协议控制信息，导致 7/8 数据字节为标准/扩展地址。

表 41——周期发送——对于应答类型#2 的应答要求

信息类型	客户请求要求	服务器应答要求	服务器约束
UUDT 使用不同的 CAN 标识用于周期发送	无约束	对周期发送只有单帧应答 对于可能的新请求（非周期发送）有多帧应答	周期发送的请求作为一个普通的诊断请求并且通过服务器网络层实现该应答（作为服务标识为 0x6A 的 USDT 信息）
			在接收到 N_USData.con 只是肯定应答完成时，应用层开启独立的机制，处理周期发送的信息。
			服务器处理周期发送是作为一个单帧 UUDT 以信息支流的形式。（例如，直接通过 CAN 控制器数据链路层而不需网络层写 UUDT 信息）
			对于 UUDT 信息，不需要包含协议控制信息（PCI）和服务标识（SId）,只包含周期标识。所以，对于标准地址最大 7 字节 periodicDataIdentifier 数据，对于扩展地址最大 6 字节 periodicDataIdentifier 数据。

图 18 和图 19 描述了两个类型的周期应答信息，服务器应当对其处理。而且，图中显示，对于周期发送应答信息不应当受服务器定时器 $S3_{Server}$ 的影响。对于这两张图，假定非默认会话在周期机制设置之前就激活了。（ReadDataByPeriodicIdentifier 服务需要非默认会话条件下执行）。

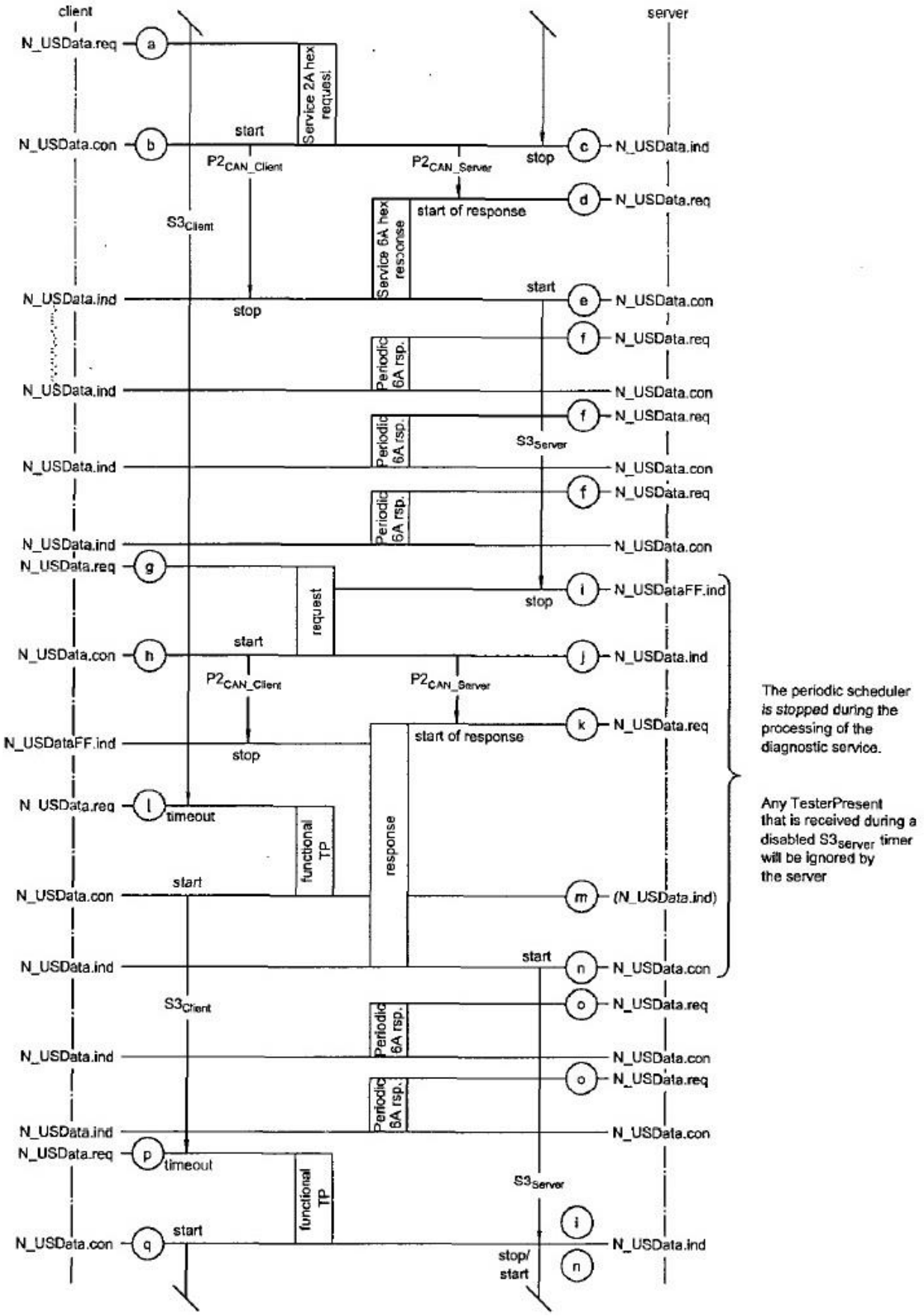


图 18——应答信息类型#1 的处理

a) 客户机诊断应用通过发送 N_USData.req 到网络层开始发送 ReadDataByPeriodIdentifier(0x2A)请求信息。网络层发送该信息之服务器。请求信息要么做为单帧信息要么是多帧信息（依赖于请求信

息中 periodicDataIdentifier 包含的数量）。该例中，假定请求信息是单帧信息。

b) 客户机通过 N_USData.con 指示请求信息的完成。随后应答定时参考 6.3.5.1.1 和 6.3.5.1.3。

c) 服务器通过 N_USData.ind 指示请求信息的完成。随后应答定时参考 6.3.5.1.1 和 6.3.5.1.2，而且，服务器停止 $S3_{Server}$ 定时器。

d) 图中，假定客户机需要服务器的一个应答。服务器应发送 ReadDataPeriodicIdentifier 肯定应答信息指示请求已被处理并且周期信息的发送将要随后开启。

e) 服务器通过 N_USData.con 指示 ReadDataPeriodicIdentifier 应答信息发送的完成。随后服务器重启 $S3_{Server}$ 定时器，这用于保持处于非默认会话状态，而不超时。

f) 服务器开始发送周期应答信息（单帧信息）。每个周期信息使用网络层协议并且使用用于其它应答信息的 CAN 应答标识。因此，服务器每次发送 N_USData.req 到网络层时，一个周期信息都被发送，并且当前没有其它服务在被服务器处理。例子中，假定服务器在下一个请求信息到来之前，能发送客户机发起的 3 周期信息。周期应答信息的发送对定时器 $S3_{Server}$ 无任何影响（见 6.3.5.4）。

g) 客户机诊断应用层通过发送 N_USData.req 到网络层开始发送 ReadDataByPeriodIdentifier(0x2A) 请求信息。网络层发送该信息之服务器。请求信息要么做为单帧信息要么是多帧信息（依赖于请求信息中 periodicDataIdentifier 包含的数量）。该例中，假定请求信息是单帧信息。

h) 客户机通过 N_USData.con 指示请求信息的完成。随后应答定时参考 6.3.5.1.1 和 6.3.5.1.3。

i) 在周期机制激活期间，服务器通过 N_USData.ind(或者单帧的 N_USData.ind)指示请求信息的开始。服务器应当立即停止周期机制，处理接收到的请求信息。而且，服务器在处理任何诊断服务的时候，它都停止 $S3_{Server}$ 定时器。

j) 服务器通过 N_USData.ind 指示多帧请求信息的完成，随后应答定时参考 6.3.5.1.1 和 6.3.5.1.2 。周期信息发送机制处于非激活状态。

k) 图中，假定客户机需要服务器的一个应答。服务器应当通过发送 N_USData.req 至网络层发送一个肯定（或否定）应答。该例中，假定应答时多帧信息。

l) $S3_{CIient}$ 定时器超时，客户机发送功能地址 TesterPresent(0x3E)请求信息重启服务器中的 $S3_{Server}$ 定时器。

m) 服务器在处理先前请求，发送多帧信息应答的过程中。因此，服务器在接收到 TesterPresent(0x3E) 请求信息时，不应动作。因为 $S3_{Server}$ 定时器还没有重新激活。

n) 当诊断服务完全处理完了，服务器重启 $S3_{Server}$ 定时器。这是说，所有诊断服务包括 TesterPresent(0x3E)都能重启 $S3_{Server}$ 定时器。诊断服务在接收到请求信息（N_USDataFF.ind 和 N_USData.ind）到完成最后结果应答的时间内都会执行。当需要应答信息时，或者当有请求导致的动作的完成但不需应答信息时（指定时间的到达导致的应答信息）。这包括否定应答信息（包括应答码 0x78）。服务器在完成处理之后（最后结果信息完全发送）重启周期机制。

o) 服务器重启周期应答信息的发送（单帧信息）。每一周期信息使用网络层协议，及用于其它应答信息的 CAN 应答标识。因此，服务器每周期发送一个 N_USData.req 至网络层并被传输，并且服务器没有其它服务在当前处理。周期应答信息的发送对 $S3_{Server}$ 不影响。（见 6.3.5.4）

p) 一旦 $S3_{CIient}$ 开启（非默认会话激活），导致功能地址 TesterPresent(0x3E)的发送，不需要应答信息。每次在 $S3_{CIient}$ 超时时。

q) 网络层通过 N_USData.con 指示 TesterPresent(0x3E)请求信息发送完之时，客户机再次重启 $S3_{CIient}$ 定时器。也就是说，功能地址 TesterPresent（0x3E）请求信息在 $S3_{CIient}$ 定时器每次超时的周期都发送。

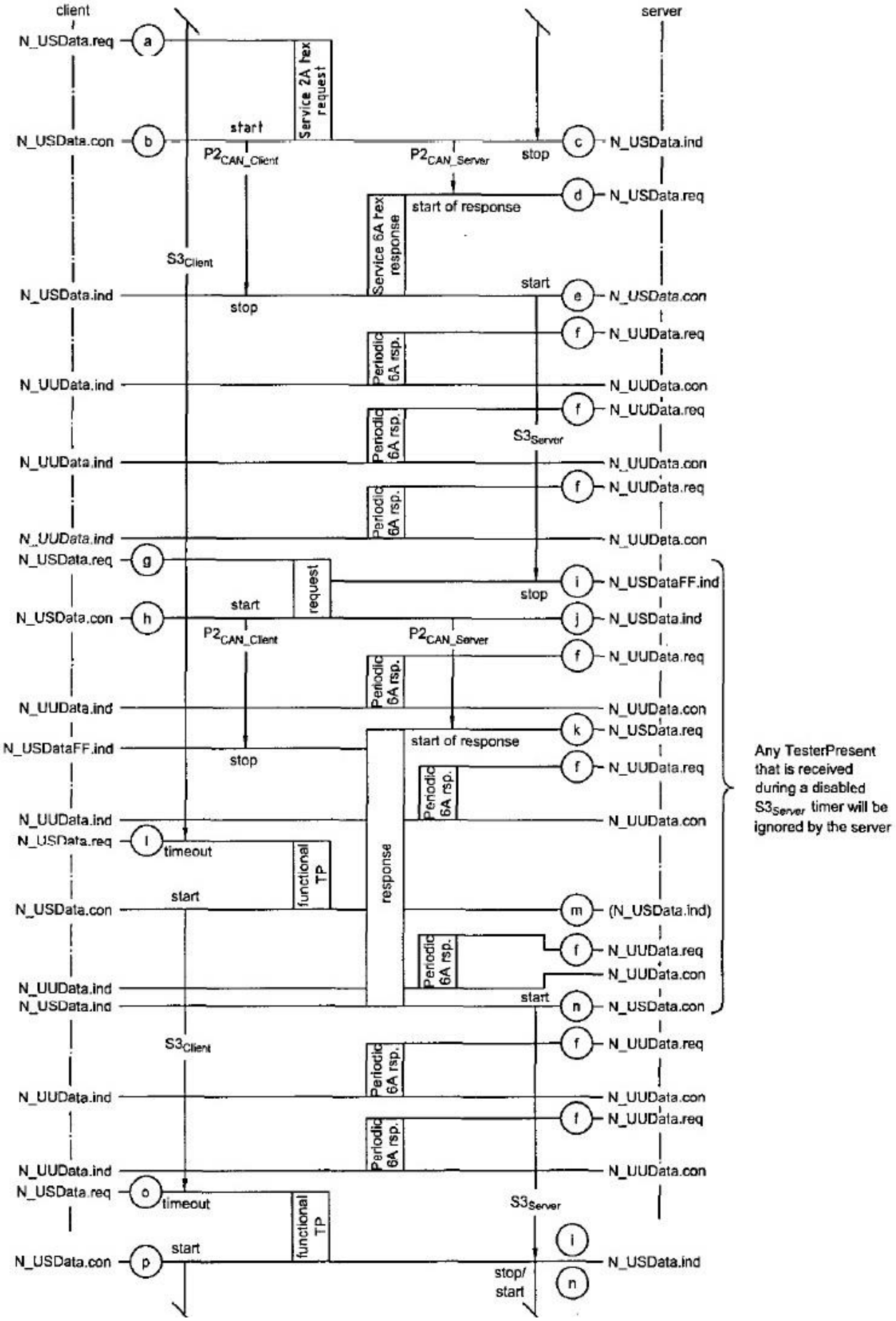


图 19——应答信息类型#2 处理

a) 客户机诊断应用通过发送N_USData.req到网络层开始发送ReadDataByPeriodIdentifier(0x2A)请求信息。网络层发送该信息之服务器。请求信息要么做为单帧信息要么是多帧信息（依赖于请求信息中periodicDataIdentifier包含的数量）。该例中，假定请求信息是单帧信息。

b) 客户机通过N_USData.con指示请求信息的完成。随后应答定时参考6.3.5.1.1和6.3.5.1.3。

c) 服务器通过N_USData.ind指示请求信息的完成。随后应答定时参考6.3.5.1.1和6.3.5.1.2，而且，服务器停止$S3_{Server}$定时器。

d) 图中，假定客户机需要服务器的一个应答。服务器应发送ReadDataPeriodicIdentifier肯定应答信息指示请求已被处理并且周期信息的发送将要随后开启。

e) 服务器通过N_USData.con指示ReadDataPeriodicIdentifier应答信息发送的完成。随后服务器重启$S3_{Server}$定时器，这用于保持处于非默认会话状态，而不超时。

f) 服务器开始发送周期应答信息（单帧信息）。每个周期信息使用网络层协议并且使用用于其它应答信息的CAN应答标识。因此，服务器每次发送N_USData.req到网络层时，一个周期信息都被发送，并且当前没有其它服务在被服务器处理。例子中，假定服务器在下一个请求信息到来之前，能发送客户机发起的3周期信息。周期应答信息的发送对定时器$S3_{Server}$无任何影响（见6.3.5.4）。

g) 客户机诊断应用层通过发送N_USData.req到网络层开始发送ReadDataByPeriodIdentifier(0x2A)请求信息。网络层发送该信息之服务器。请求信息要么做为单帧信息要么是多帧信息（依赖于请求信息中periodicDataIdentifier包含的数量）。该例中，假定请求信息是单帧信息。

h) 客户机通过N_USData.con指示请求信息的完成。随后应答定时参考6.3.5.1.1和6.3.5.1.3。

i) 在周期机制激活期间，服务器通过N_USData.ind（或者单帧的N_USData.ind）指示请求信息的开始。服务器应当立即停止周期机制，处理接收到的请求信息。而且，服务器在处理任何诊断服务的时候，它都停止$S3_{Server}$定时器。

j) 服务器通过N_USData.ind指示多帧请求信息的完成，随后应答定时参考6.3.5.1.1和6.3.5.1.2 。周期信息发送机制处于非激活状态。

k) 图中，假定客户机需要服务器的一个应答。服务器应当通过发送N_USData.req至网络层发送一个肯定（或否定）应答。该例中，假定应答是多帧信息。当网络层发送完多帧应答信息时，周期机制继续发送周期应答信息。

l) $S3_{Client}$定时器超时，客户机发送功能地址TesterPresent(0x3E)请求信息重启服务器中的$S3_{Server}$定时器。

m) 服务器在处理先前请求，发送多帧信息应答的过程中。因此，服务器在接收到TesterPresent(0x3E)请求信息时，不应动作。因为$S3_{Server}$定时器还没有重新激活。

n) 当诊断服务完全处理完了，服务器重启$S3_{Server}$定时器。这是说，所有诊断服务包括TesterPresent(0x3E)都能重启$S3_{Server}$定时器。诊断服务在接收到请求信息（N_USDataFF.ind和N_USData.ind）到完成最后结果应答的时间内都会执行。当需要应答信息时，或者当有请求导致的动作的完成但不需应答信息时（指定时间的到达导致的应答信息）。这包括否定应答信息（包括应答码0x78）。服务器在完成处理之后（最后结果信息完全发送）重启周期机制。

o) 一旦$S3_{Client}$开启(非默认会话激活),导致功能地址TesterPresent(0x3E)的发送,不需要应答信息。每次在$S3_{Client}$超时时。

p) 网络层通过N_USData.con指示TesterPresent(0x3E)请求信息发送完之时,客户机再次重启$S3_{Client}$定时器。也就是说，功能地址TesterPresent（0x3E）请求信息在$S3_{Client}$定时器每次超时的周期都发送。

表 42 描述了 CAN 服务实施可用数据参数

表 42——数据参数定义——发送模式

16 进制数	描述	Cvt	助记符
01	sendAtSlowRate	U	SASR
02	sendAtMediumRate	U	SAMR
03	sendAtFastRate	U	SAFR
04	stopSending	U	SS

9.3.5 动态定义数据标识（DynamicallyDefineDataIdentifier）（0x2C）服务

当客户机动态定义一个 periodicDataIdentifier 并且动态定义的 periodicDataIdentifier 总长超过了单帧周期应答信息最大长度时，该请求应通过应答码 0x31(请求超出范围， requestOutOfRange) 否定应答拒绝，见 ReadDataByPeriodicIdentifier(9.3.4)参考周期应答信息格式。

当多个 DynamicallyDefineDataIdentifier 请求信息用于设置单个的 periodicDataIdentifier 并且服务器检测到连续请求期间超出了最大字节数时，服务器应将该定义置位请求之前，以免超出。

表 43 描述了 CAN 服务实施可用数据参数

表 43——子功能参数定义

16 进制数	描述	Cvt	助记符
01	defineByIdentifier	U	DBID
02	defineByMemoryAddress	U	DBMA
03	clearDynamicallyDefinedDataIdentifier	U	CDDDI

9.3.6 通过标识写数据服务（WriteDataByIdentifier）(0x2E)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.3.7 通过地址写内存服务（WriteMemoryByAddress）(0x3D)

9.4 存储发送数据功能单元

9.4.1 读故障码信息服务（ReadDTCInformation）（19 hex）

表 44 描述了 CAN 服务实施可用数据参数

表 44——子功能参数定义

进制（位 6-0）	描述	Cvt	助记符
01	通过状态掩码报告故障码个数	U	RNODTCBSM
02	通过状态掩码报告故障码	M	RDTCBSM
03	短标识报告故障码信息	U	RDTCSSI
04	记录号报告故障码信息	U	RDTCSSBDTC
05	故障码号报告故障码信息	U	RDTCSSBRN
06	故障码号报告故障码扩展数据记录	U	RDTCEDRBDN
07	安全码记录报告故障码号	U	RNODTCBSMR
08	安全码记录报告故障码	U	RDTCBSMR
09	报告故障码安全信息	U	RSIODTC
0A	报告支持的故障码	U	RSUPDTC
0B	报告第一个测试失败的故障码	U	RFTFDTC
0C	报告第一个确认的故障码	U	RFCDTC
0D	报告近期测试失败最多的故障码	U	RMRVDTC

0E	报告近期确认最多的故障码	U	RMRCDTC
0F	状态掩码报告镜像内存故障码	U	RMMDTCBSM
10	通过故障码号报告镜像内存故障码扩展数据记录	U	RMMDEDRBDN
11	通过状态掩码报告镜像内存故障码个数	U	RNOMMDTCBSM
12	状态掩码报告排放相关 OBD 故障码个数	C	RNOOBDDTCBSM
13	状态掩码报告排放相关 OBD 故障码	C	ROBDDTCBSM

表 45 描述了 CAN 服务实施可用故障码状态位

故障码故障类型字节用于 CAN 服务中，则 DTCFailureTypeByte 定义应按照 ISO 15031-6.

表 45——DTC 状态位定义

位	说明	Cvt		助记符
		排放的	非排放	
0	测试故障	U	U	TF
1	在该监测循环检测故障	M	C_1	TFTMC
2	挂起的故障码	M	U	PDTC
3	确认的故障码	M	M	CDTC
4	从最近一次清除时测试未完成	C_2	C_2	TNCSLC
5	从最近一次清除时测试故障	C_2	C_2	TFSLC
6	该监测循环测试未完成	M	M	TNCTMC
7	警告指示请求	M	U	WIR

C_1：当位 2（pendingDTC）支持的时候，位 1（testFailedThisMonitoringCycle）是强制的，位 2（pendingDTC）不支持的时候，位 1（testFailedThisMonitoringCycle）为用户可选的。

C_2：位 4（testNotPassedSinceLastClear）和位 5(testNotFailedSinceLastClear)应当总是同时支持。

9.4.2 清故障码信息服务（ClearDiagnosticInformation）(0x14)

表 46 描述了 CAN 服务实施可用数据参数

表 46——子功能参数定义——一组故障码

16 进制数	描述	Cvt	助记符
000000-FFFFFE	单独/单个故障码	U	SDTC
FFFFFF	所有组（所有故障码）	M	AGDTC

9.5 输入输出控制功能单元

9.5.1 输入输出控制标识服务（InputOutputControlByIdentifier）(0x2F)

控制选项记录的第一个字节用作输入输出控制参数时，表 47 描述了 CAN 服务实施可用数据参数

表 46——子功能参数定义——输入输出控制参数

16 进制数	描述	Cvt	助记符
00	返回控制到 ECU	U	RCTECU
01	重启到默认	U	RTD
02	冻结当前状态	U	FCS

03	短期调整	U	STA

9.6 例程远程激活功能单元

9.6.1 例程控制服务（RoutineControl）(0x31)

表 48 描述了 CAN 服务实施可用数据参数

表 48——子功能参数定义

16 进制数	描述	Cvt	助记符
01	启动例程	U	STR
02	停止例程	U	STPR
03	请求例程结果	U	RRR

9.7 上传/下载功能单元

9.7.1 请求下载服务（RequestDownload）(0x34)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.7.2 请求上传服务（RequestUpload）(0x35)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.7.3 传输数据服务（TransferData）(0x36)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。

9.7.4 请求传输退出服务（RequestTransferExit）(0x37)

对于该项服务的 CAN 实施既没有定义另外的要求，也没有限制。