Third-party Call Control in VoIP Networks for Call Center Applications

A. Miloslavski, V. Antonov, L. Yegoshin, S. Shkrabov,
J. Boyle, G. Pogosyants, N. Anisimov

Genesys Telecommunication Labs (an Alcatel company)
1155 Market St., San Francisco, CA 94103
http://www.genesyslab.com

Abstract -- **In this paper we discuss the problem of building third-party call control in a VoIP network based on ITU-T Recommendation H.323. We will consider a specific case that could be used in the call center environment. We will suggest and advocate a solution with H.323 gatekeepers connected to CTI server via special protocol. The CTI Server contains all the logic for service execution while a gatekeeper implements only elementary switching functions. Some features of the protocol are considered.**

I. Introduction

One of the most important applications of Computer Telephony Integration (CTI) is the call center (CC) [1]. The key element on which all the software of the call center relies is a CTI-link containing a communication protocol between a telephony switch and a computer that is often referred to as a CTI server. CTI protocol allows the CTI Server to monitor and control processing of telephony calls, i.e., it provides third-party call control. Modern call centers are comprised of complex and expensive software called call center applications that provide the processing of inbound and outbound calls from customers. For instance, a CC application may include preliminary call processing by Interactive Voice Response (IVR), routing the call to the most appropriate operator (agent), assisting the agent in call processing, managing customer relations, reporting, etc.

Using emerging Voice over IP (VoIP) technologies promises to substantially improve the performance of the call center. At the same time, in order to painlessly transition from PBX-based to VoIP networks in call centers one needs to solve certain technical problems, such as how to substitute these networks without changing other call center software. In other words, we need to build third-party call control in VoIP networks in order to provide PBX-based services, such as call transfer, conferencing, and automatic call distribution.

In this paper we will consider one of the solutions for the problem using the example of call center software developed at Genesys Telecommunication Labs, a wholly owned subsidiary of Alcatel. We will consider the case of when VoIP net is based on ITU-T Recommendation H.323 [2].

II. The Problem

The typical structure of a call center employing CTI-link is depicted in Fig.1. The key element of the call center is a CTI-link that connects PBX and CTI Server to provide visibility and control of the switching domain by the computing domain. The CTI link makes it possible to organize call processing in a flexible way. For example, the processing of an inbound call may involve the following steps. The CTI Server learns about the incoming call due to an event received via the CTI-link. Then it routes the call to an IVR that may collect additional information about the call and the customer. Then the Router routes the call to the most appropriate agent to be processed. The telephony call is transferred along with related data. During the call processing the agent may use information from the database (about the call and the customer), may consult with another agent, organize a conference or transfer the call to another agent. To initiate all these PBX-based services the agent usually uses a special desktop application that, in its turn, exploits the third-party call control functions. It should be mentioned that the Genesys CTI Server uses a call model that has much in common with the SCTA call model [3].

Now we want to substitute the PBX network with the VoIP network based on the H.323 Recommendation. In other words we want to build a third-party call control in the H.323 network which is exactly the same as that in the PBX network. We should stress the complexity of the problem that stems from the nature of the networks. Indeed, the CTI-link assumes centralization, while H.323 is built on a distributive principle.

In relation to this we should mention the work [4] devoted to building third-party call control in the H.323 network providing CSTA-based interface for applications. The most general case of H.323 is considered to assume a high degree of distribution. In particular, the H.323 network can be configured without gatekeepers, and intelligence is distributed over all endpoints. For "centralization," to monitor and control calls a new element called the "Marshal" is introduced. It plays the

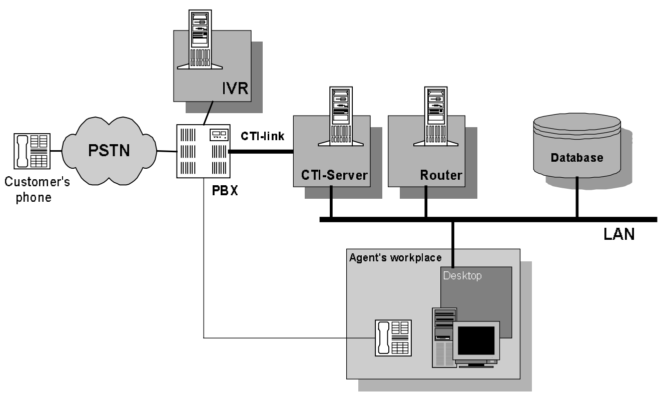

Fig. 1. Call Center Environment

role of the CTI Server. Each endpoint is equipped with an additional element called "Deputy" which connects with the Marshal. Deputies inform Marshal about a call's progress and operate on its behalf (e.g., initiate a connection). This solution requires an additional protocol "Marshal-Deputy", which makes an H.323 configuration more complicated. It seems that this approach faces some severe difficulties since some important scenarios (e.g., conferencing) are still not developed.

Fortunately, in less general cases where we can choose the configuration of the net (e.g., in the case of a call center) we can find less complex solutions. The natural candidate for the central element is a gatekeeper that in some configurations (the gatekeeper routed mode) possesses all information about call signaling, call control, and supplementary service processing. However endowing a gatekeeper with CTI Server functionality is not a good solution because it will result in a complex and inflexible gatekeeper implementation. Moreover, it is not clear what to do when there are several gatekeepers in the network.

A more reasonable approach is to move CTI Server functions into a special server connected to the gatekeeper(s). The one example of this approach is the concept of "thin gatekeeper" developed within Dialogic [5]. According to this approach a gatekeeper is connected to a special server via CSTA protocol. This solution assumes that the gatekeeper understands and operates in accordance with CSTA protocol.

At the same time, notice that CSTA has been developed to connect switching and computing domains [3]. In fact, CSTA provides a copy of the switching call model in the CTI Server. Moreover, it keeps these call models consistent using a request/event mechanism. In our point of view this solution is still complicated because in the VoIP network there is no longer any boundary between the switching and computing domains. Everything is in the computing domain and it is reasonable to have only one call model rather than two.

III. Solution

In this paper we suggest a configuration where all service logic is placed in the CTI Server while gatekeepers perform more elementary operations.

A. Architectural environment

Traditionally, CTI protocols were largely vendor specific, thus forcing CTI software vendors to develop separate software modules for each switch model. A number of protocols were proposed as potential standards for CTI links, such as CSTA [3], MGCP/Megaco [7] and newer revisions of TAPI [8]. Unfortunately, such attempts at such CTI protocol standardization are not likely to produce compatible implementations. It is easy to see that any non-trivial CTI software suite has a need to maintain an accurate replica of the switch state, which in practice means that the CTI software has to replicate the call model of the particular switch, as shown in Fig.2(i). Any discrepancy between the actual call model implemented by the switch vendor and its reverse-engineered replica in the CTI software causes loss of coherency between the actual state and its image in the control software. Worse yet, practically all switch vendors introduce subtle changes to their call model in successive versions of switch software (this is unavoidable when new features are added and programming errors are corrected). Packet-switched telephony makes call models even more complicated by replacing centralized switches with a heterogeneous, distributed switching environment.

One approach to solving the problem of call model incompatibility would be to standardize the call model itself, providing a rigid specification for switch behavior. This, in practice, is a very challenging task because of the richness of possible call behavior and call control features. The current generation of telephone switches provides hundreds of these features to accommodate diverse customer requirements. Any single standard defining the switch behavior is bound to be overly restrictive; therefore, vendors reacting to customer demand will be compelled to expand it, thus defeating the very purpose of standardization.

As a result, none of the proposed CTI standards even attempt to specify a standard call model (beyond some elementary features), thus leaving the actual behavior of switches to the discretion of the manufacturer. While helping to solve the easy problem of CTI message format encoding and decoding, they leave the complicated task of switch behavior modeling completely untouched.

In this paper we advocate another approach based on Simple Media Control Protocol (SMCP) [6] developed within Genesys Labs. The SMCP solves the behavioral compatibility problem by providing CTI software access to the basic switching functions, with no proprietary call model overlay. In SMCP architecture, a switch vendor

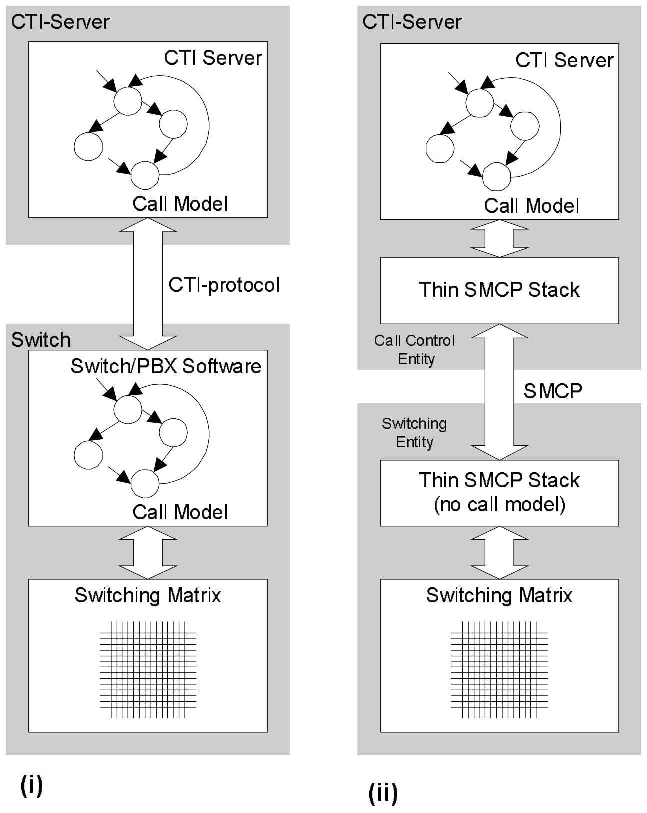

Fig. 2. Old and new architectures

not have to implement an elaborate call model, a thin SMCP protocol stack is sufficient. The PBX call control functionality can be external and developed separately; see Fig.2(ii). A switch vendor supporting SMCP could use third-party SMCP-based PBX call control software instead of doing costly in-house PBX design or integration. Note that in SMCP architecture, only one call model is used in CTI- controlled operation, thus eliminating the problem of CTI software and switch call model incoherence.

Since SMCP provides support for fault tolerance, a fallback to the bundled PBX software provides the same degree of protection from CTI server failures as the historical CTI architecture.

The simplicity of the SMCP model allows it to incorporate functions not generally found in conventional voice-only call models. Indeed, the call commutation model defined in this document is in no way specific to voice, and can be used to control a very wide variety of call-like interactions, such as analog voice, synchronous or packetized digital voice, interactive video, chat, whiteboard or desktop sharing sessions.

A. Summary of SMCP

A.1. Basic notions

SMCP is a protocol that regulates a communication between two objects – Call Control Entity (CCE) and Switching Entity (SWE), see **Fig. 2**(ii). The Switching Entity is a module that fulfills only switching functions and does not aware about services. The *Call Control Entity* contains service logic and implements it with the aid of SMCP instructing SWE what to do in order to provide the service. Thus SMCP is a protocol of a low level that provides only basic switching functions and does not contain service logic.

Note that CCE and SWE are abstract modules in the sense that their physical embodiment depends on the architecture of VoIP network. For example, if SMCP is used in an H.323 network, SWE may be implemented as a Gatekeeper. In an MGCP based network the role of SWE can be played by a Media Gateway Controller (MGC).

The SMCP assumes that the SWE implements an abstraction of a commutating device allowing separate control of call legs (call leg is a half of a connection). In other words, SWE must make sure that externally visible calls are kept established even when the commutating device tears down and reestablishes internal connections between them. Fig. 3. illustrates the voice switch-based commutator, and Fig.4. shows how the SMCP commutator model maps to a typical enterprise VoIP configuration.

The call legs are terminated at endpoints and are associated with physical or virtual ports (there are situations when endpoints do not have associated ports). Some ports can be used for support of multiple simultaneous connections. The parts of call legs outside of a switch or a gateway are called exterior, connections between them inside switches are interior. More than two call legs can be connected together to form a multi party conversation.

Typically, endpoints can terminate only one call leg at a time. However, in some cases (for example, call treatment

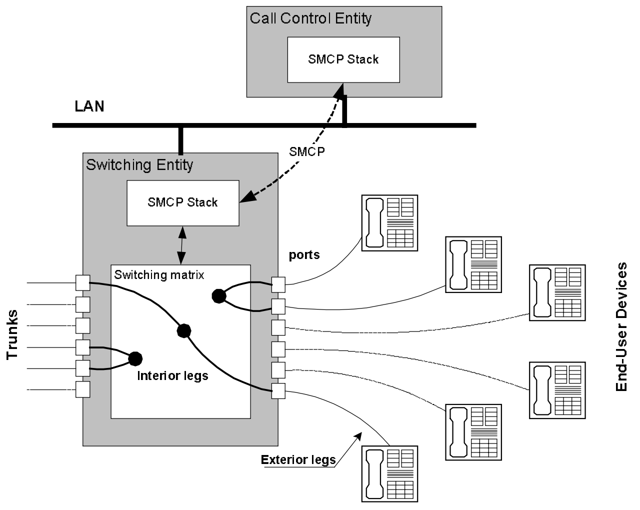

Fig. 3. Voice Switch as a SMCP Commutator

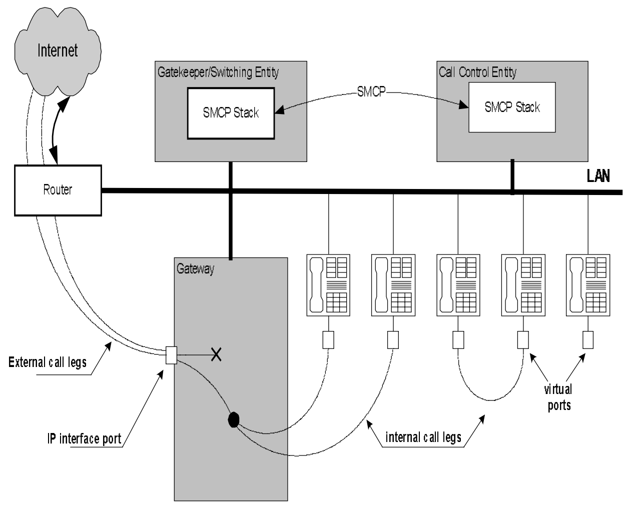

Fig.4. VoIP System as a SMCP Commutator

sources, such as music-on-hold) a single endpoint can be connected to multiple call legs. All endpoints have unique endpoint addresses represented as ASCII character strings; such strings can be telephone numbers, URLs, E-mails, etc; the exact naming scheme is application specific. Ports have unique vendor-specific names.

A.2. Message syntax

As in many other Internet protocols (e.g., HTTP, SIP, MGCP), SMCP messages are text based, i.e., all messages may contain only ASCII printable characters, spaces and tabs. Each line is terminated with the **CR-LF** sequence (0D 0A hex.) Messages have the following structure:

```
  Message-verb reference-number CR LF
   attribute-name: attribute-value CR LF
       attribute-value-continuation CR LF
   attribute-name: attribute-value CR LF
 CR LF
```

Message verbs and attribute names are strings composed of letters, digits and minus. Attribute values are arbitrary strings of printable characters, spaces and tabs. An attribute value string can span several lines, with continuation lines starting from space or tab. An empty line (line containing only the **CR LF** sequence) terminates a message. The messages sent by CCE will be referred to as requests, and messages sent by SWE will be called events or notifications. Both sides must also send replies in response to messages. Reference numbers are unique decimal integer numbers generated by CCE and used to match replies to requests; all requests and notifications must have reference numbers.

A.3. Exterior Call Control

The protocol consists of several procedures. However, in this paper we will consider only the main procedures related to call control. The other procedures such as configuration management, fault tolerance management, encryption, etc. are out of the scope of this paper and can be found in [6].

The exterior call control commands and events provide a mechanism for manipulating the endpoint sides of call legs.

Outbound Calls

For managing outbound calls SMCP uses the following protocol data units:

CALL	Make an outbound call;
BUSY	Port is busy;
CALLING	Outbound call request acknowledgment;
GOTCA	Got call alerting from endpoint;
ANSWERED	Call answered;
CONNECTED	Connection established.

Outbound calls generally progress through several stages:

- CCE commands SWE to initiate the connection with a **CALL** request.
- If the request is not valid, SWE responds with an **ERROR** reply; if the selected port is busy, SWE responds with a **BUSY** reply; otherwise the request is considered valid and SWE allocates a call leg identifier and returns it to CCE with a **CALLING** reply.
- SWE initiates a transport connection to the endpoint, and sends a Call Setup message to the endpoint.

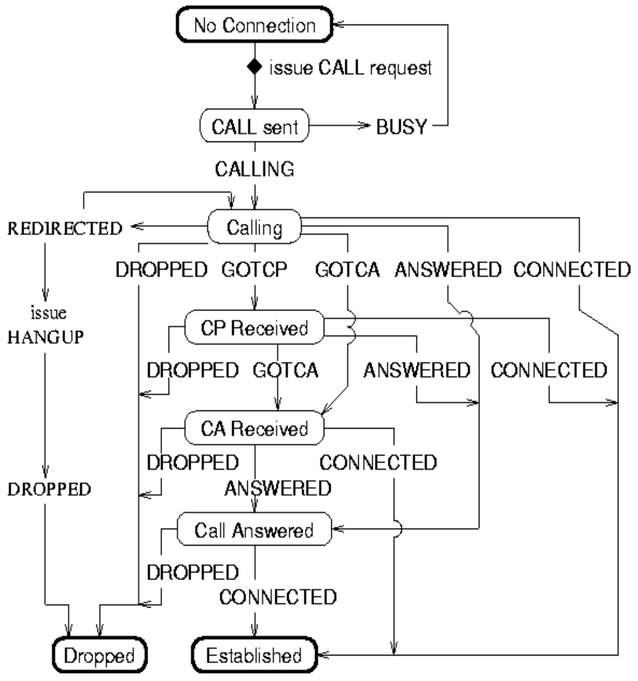

Fig. 5. CCE State Machine for Outbound Calls

- The endpoint may reply with a Call Proceeding message to indicate that it accepted further responsibility for call delivery.
- The endpoint may then reply with a Call Alerting message to indicate that the call has been delivered to the end-user device which started alerting the user.
- If the callee answers the call, the endpoint device replies with a Setup message, thus initiating the process of establishing a transport connection. SWE informs CCE about this event with an **ANSWERED** notification.
- When a transport connection is established the endpoint device replies with a Connected message. After that the call is completely established, and SWE sends a **CONNECTED** notification to the client.
- At any time after the **CALLING** notification, SWE may inform CCE that the connection cannot be completed with a **DROPPED** notification. If CCE did not send a **HANGUP** request for this call already, this request must be sent to release the connection after receipt of a **DROPPED** notification.

A diagram of CCE state transitions while performing an outbound call is shown in Fig. . Note that the connection from a switch to a directly attached phone set is also an outbound call in the SMCP call model. Making outbound calls generally requires specification of a switch or gateway port, and the additional target address (which can be for directly attached devices, can contain an E.164 phone number for PSTN calls, or an E-mail address or URL for VoIP calls). If SWE implements a numbering plan, the port specification may be omitted and selected automatically by the target address using the plan's target lookup table.

Incoming Calls

For managing inbound calls SMCP uses the following protocol data units:

RING	Notify about incoming call;
SENDCP	Send Call Proceeding message;
SENDCA	Send Call Alerting message;
ANSWER	Answer the call.

A typical incoming call scenario is:

- SWE detects the incoming attempt to establish a transport connection, accepts the transport connection, and reads the Setup message.
- SWE alerts CCE about the incoming call with a **RING** notification.
- CCE may instruct SWE to send a Call Proceeding message to the calling party with a **SENDCP** request, and wait for an **OK** reply.
- CCE may then instruct SWE to send a Call Alerting message to the calling party with a **SENDCA** request, and wait for an **OK** reply.
- When CCE wishes to answer the call, it asks SWE to perform the transport protocol negotiation with the **ANSWER** request, and wait for an **OK** reply.
- SWE informs CCE about success or failure of the negotiation; if the negotiation was successful the connection is considered established and SWE sends a **CONNECTED** notification to the CCE.
- At any time after the original **RING** notification, SWE may inform CCE that the connection was abandoned with a **DROPPED** notification. If CCE did not send a **HANGUP** request for this call, this request must be sent to release the connection after receipt of a **DROPPED** notification.
- After receiving any reply or notification pertaining to the current call leg CCE may command SWE to drop the connection with a **HANGUP** request. After issuing the **HANGUP** request CCE should wait for the **DROPPED** notification from SWE before assuming that the port is available.

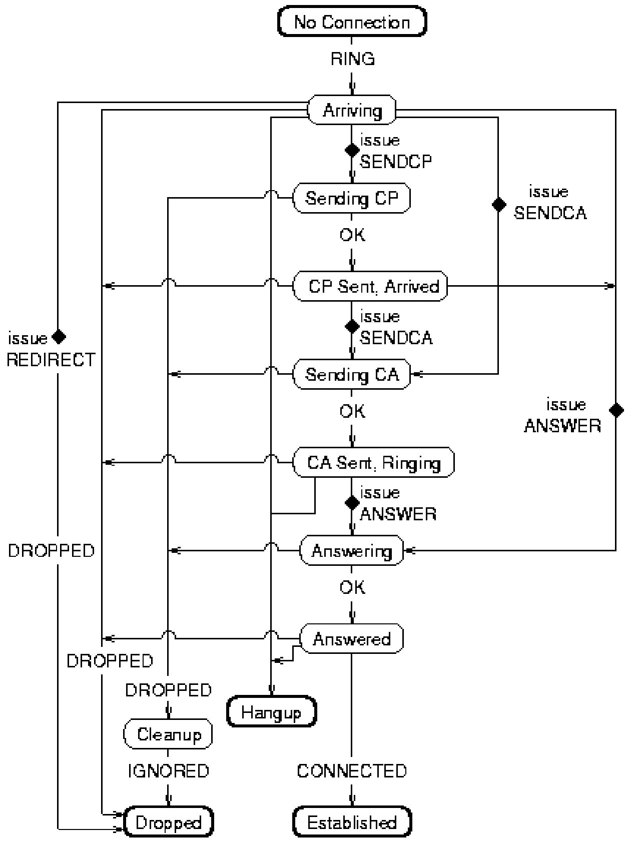

Fig. 6. CCE State Machine for Incoming Calls

A diagram of CCE state transitions while answering an incoming call is shown in Fig. 6.

Call Release

An established connection may be released by either the local or remote party. CCE may request releasing the call leg with a **HANGUP** request. When the call is dropped due to an endpoint's action or a transport disconnection, SWE sends a **DROPPED** notification to CCE. CCE then must release the call leg with a **HANGUP** request. SWE may reuse the call leg identifier only after it has sent a **DROPPED** notification and received a valid **HANGUP** request.

A.4. Interior Media Stream Commutation

The interior media stream commutation allows interconnecting existing exterior call legs in an arbitrary manner. The connections are established with a **MAKE** request and are torn down with **BREAK** or **HANGUP** requests. It is important to note that interior connections for different media streams related to the same call leg are different connections, and can be independently established or torn down.

MAKE - Make Connection Between Call Legs

The **MAKE** request instructs SWE to connect media stream(s) from an existing call leg to another (target) call leg, or to a conference in which the target call leg is participating. The interior connection can be made even if exterior connections are not fully established (this may affect subsequent codec type negotiations). A **MAKE** request can only be issued for a call leg that is not yet a party to an interior connection.

If a target call leg is a party to an existing interior connection, SWE must perform appropriate actions to allocate and configure an MCU for the newly created or expanded conference. GK is also responsible for establishing the appropriate translation between different codec types, if applicable.

If the **MAKE** request is successful, SWE replies with **OK**. If a **DROPPED** notification was generated for the current call leg, **MAKE** request is ignored (resulting in an **IGNORED** reply). If the target call leg was dropped, but CCE did not yet issue the corresponding **HANGUP** request, **MAKE** should be executed nonetheless (this is important in case the other call leg participated in a conference) and no error should be reported.

BREAK - Break Interior Connection Between Call Legs The **BREAK** request disconnects a call leg from a two-party or multi party interior connection(s). Unlike the **HANGUP** request, **BREAK** does not affect the state of the exterior connections, and merely severs the interior connection between the specified call leg and the other parties. If the **BREAK** request is successful, SWE replies with **OK**. If a **DROPPED** notification was generated for the current call leg, the **BREAK** request is ignored, resulting in an **IGNORED** reply.

B. Third-party service creation

Exterior and interior leg control procedures allow one to develop different services whose logic is placed into the CTI Server as in Figure 2 (ii). In particular there may be standard PBX-based services like Plain Old Telephone Service (POTS) and complementary services like Call Transfer, Call Forwarding, Conferencing, Call Hold, etc. Moreover, we can design specific services that are not supported by standard PBXs.

In this section we consider the widely used service of making a call on behalf of a third-party application. The service is initiated by receiving a request *MakeCall(x,y)* and consists of establishing a connection between phone *x* and phone *y*. First, the service process establishes a connection with the device *x* using an outbound scheme (see Section III.A.3). If phone *x* answers, then the process tries to establish a connection with the device *y* also using the outbound scheme. If this phone also answers, the process makes a stream connection using a **MAKE** request. The scenario of successful connection establishment is depicted in Fig.7. The service is performed by a special process that creates separate sub-processes for legs *x* and *y*. The progress of the service execution is reported to the Application by means of several events.

IV. Conclusion

In this paper we suggested an approach to the design of third-party control in H.323 networks primarily intended for use in a call center environment. This approach is based on a new protocol called SMCP that connects a CTI Server with elements of a H.323 network. The main distinction of this approach is that it contains only one call model that is maintained by the CTI Server. On the one hand this approach makes it possible to eliminate unnecessary complexity from the elements of H.323 networks. On the other hand, it gives more flexibility in creating third-party services for call center operation. For example we can develop more specific services for call centers than are allowed by approaches based on conventional CTI protocols like CSTA.

We should also point out one important issue that may arise during service design based on SMCP usually known as a feature interaction problem [9]. In this case, indeed, there may be undesirable interaction when protocol messages can be differently interpreted depending on the services they implement. However, this issue is left for further study.

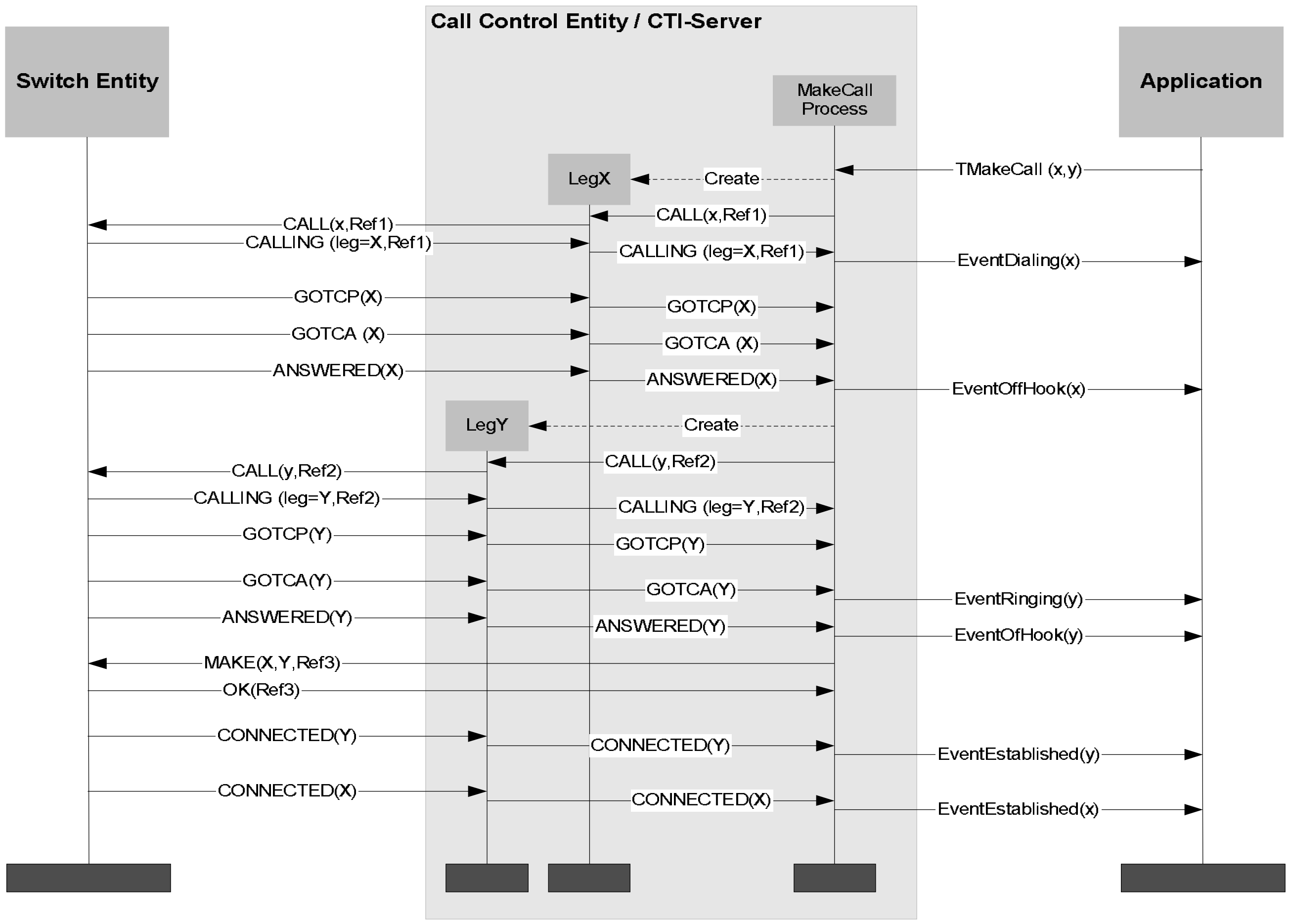

Fig.7. Scenario of Make Call service

ACKNOWLEDGMENTS

The authors are grateful to David Solnit and Steven Hamilton for technical help.

REFERENCES

[1] R.Walters. CTI in Action, Wiley, 1997.
[2] ITU-T Recommendation H323. “Packet-Based Multimedia Communication Systems”, Geneva, Switzerland, September 1999.
[3] ECMA-269. Services for Computer Supported Telecommunications Applications (CSTA) Phase III, 3rd edition, December 1998. See http://www.ecma.ch
[4] N.Oliver, T.Miller, J.D. Smith (Eds) Notes on CSTA in IP Telecommunications, Working paper by ECMA TC32-TG11, 15 May, 2000.
[5] M. Robins. Pain-Free Internet Telephony-Powered CT-Apps, Internet Telephony Magazine, August 1999.
[6] V.Antonov, L.Yegoshin. Simple Media Control Protocol, Genesys Telecommunication Labs. October 2000. See http://www.genesyslab.com
[7] T.Taylor. Magaco/H.248: A New Standard for Media Gateway Control. IEEE Communication Magazine. October 2000, Vol.38, No.10, pp.124-132.
[8] IP Telephony with TAPI 3.0. White Paper. Microsoft.
[9] E.J. Cameron, N. Griffeth, Y.-J. Linand, M.E. Nilson, W.K. Schnure, and H. Velthuijsen. A Feature Interaction Benchmark for IN and Beyond. In: L. G. Bouma, H.Velthuijsen, (Eds.) Feature Interactions in Telecommunications Systems, IOS Press, Amsterdam, 1994, pp. 1-23