An approach of the Future Narrow Band Digital Terminal protocol.

Frédéric GAUCHE

Université Paris Est, LACL
Créteil, FRANCE

e-mail: frederic.gauche@dirssic.fr

Website: http://www.dirssic.fr

ABSTRACT

In this paper, we describe a part of a new interoperability protocol designed by the US for secure voice and data communications, the *Future Narrow Band Digital Terminal* protocol. This one is intended to become a standard for NATO voice communications. Our work is a first step to formally verify and validate this protocol. We give a first simple model, written in the UPPAAL tool model checker, of the *Call Setup* process described in the *Signaling Plan* specification. We then use our model to test some properties and expose some results that led us to find some unawaited situations.

Keywords

Secure communication protocol, interoperability, verification, FNBDT, UPPAAL.

1. INTRODUCTION

Interoperability is considered very important, even critical, for information systems. To make equipment, terminals, elements used to be able to communicate and interoperate over various networks, it is necessary to define and develop specifications or sets of specifications that every system will integrate and implement for its own specific needs. These specifications define how all elements will discuss in order to design interfaces.
Telephone is a good example of interoperable systems. All interfaces are standards managed by ITU-T. Anyone is able to make a phone call without taking care of who manufactured his telephone or which equipment are used by the providers.
The Future Narrow Band Digital Terminal (FNBDT) protocol is relatively new in the world of the communication protocols and has been developed to assure this role of interoperability for secure voice and data communications. For the present reserved to the US and NATO world, there are not very much available publications on it and its specifications are not widely accessible. However, this protocol is very much interesting since it intends to become a standard for communication systems within NATO and may possibly become a standard for interoperability of secure voice in the future.

To specify and validate a new protocol is not an easy task. Implementations and tests are of course the basis for that and are particularly important in the case of interoperability. One other possibility is to model those protocols through proof or model checking tools for example in order to perform specific tests on different properties we want to validate. Most often this is much less expensive than a real implementation and lots of tools have been developed for the verification and validation of real-time systems or protocols.

Most of the related works and available papers on the FNBDT protocol can be found in IEEE conferences proceedings and address only low levels aspects of the use of FNBDT ([DTSBRBH02], [DT02], [DT01], [SDRBT02]). We don't know about any available formal model of highest levels of FNBDT.

We give here a first approach of a model of the FNBDT *Call Setup*. In a first part we explain the main features of FNBDT; we insist particularly on the *Call Setup* of the protocol which represent the core of our study. We then give a model of this *Call Setup* under the UPPAAL tool model checker. Finally we describe some results returned by our model with the use of the verifier tool.

2. FNBDT

The Future Narrow Band Digital Terminal (FNBDT) is a US project, launched in 1997, which aims at allowing communications between equipment working over multiple and various networks (RTC, GSM, CDMA, ATM, IP, radio, etc.). It has been developed for the US government to become a standard to assure interoperability of secure voice and data communication and proposed by the US for the use within NATO in 2003. The protocol developed has been renamed as SCIP[1] in 2004.
The global approach is to define a core specification allowing every NATO nation to build its own solutions using architectures and standard protocols. An international working group has been created in 2003 to work on (IICWG[2]). The participation to this group however is submitted to the signature of a governmental NDA.
The protocol specifications are not widely accessible. Only one version of the Signaling Plan specification [FNBDT-210] can be found over the Internet[3]. This paper deals only with this specification.

The FNBDT-210 document specifies all signaling necessary to the implementation of operational modes defined for the establishment of a secure end-to-end communication between two terminals.
• Certificates, key and other information exchanges between the end-to-end users previously to the secured traffic.
• Transmission of end-to-end secured traffic of voice and data.

[1]Secure Communication Interoperability Protocol
[2]International Interoperability Control Working Group
[3]www.dtic.mil

• Signaling necessary to establish, control and maintain the secure communication.
• Signaling for the electronic "over-the-air" rekey.
The FNBDT-210 can be considered as the start point for the implementation of other operational modes specifically defined for the needs of individual nations.

Other specifications are necessary to the full implementation of the protocol on devices ([WIKI] and [NC3A]). But these are not freely available over the Internet.
• The FNBDT-120 document is the key management plan.
• The FNBDT-230 document gives the cryptographic specifications.
• STANAG-4591 specifies the MELPe[4] secure voice standard for NATO.

2.1 The protocol Call Setup

FNBDT works within the highest layers of the OSI Reference Model (Application level). A data transmission channel has to be established first between the terminals in order to open the FNBDT session. Means necessary and used for the establishment of this data channel are not part of the FNBDT specification.
Over this channel, the protocol uses two modes for transmission. For the call setup and for data, it uses a *Framed traffic* mode similar to an ARQ[5] protocol with FEC[6] to ensure reliable transmission. For voice, it uses a *Full Bandwidth traffic* mode which is simply a stream of MELPe data blocks in order to maximize the use of the available bandwidth. In that case, a synchronization block is sent at regular intervals in place of a data frame.

The FNBDT Call Setup is established over the opened data channel through the exchange of four messages (Figure 1) :

• The *Capabilities* message.
• The *Parameters/Certificate* message.
• The *F(R)* message.
• The *Cryptosync* message.

In the current study for this paper, we consider only this part of the protocol.

2.2 The Capabilities Message exchange

When a data channel is opened between two terminals, the first message to be sent or received is the *Capabilities* message (CM). This one contains all information that will allow them to control and evaluate their mutual compatibility to the protocol and decide which common algorithms they will use. At this time there is no rule for the process of the exchange. Each terminal can send his or receive the far end's CM first.
When a terminal sends a CM before he has received one, he automatically starts a *First Message Timer*. This timer allows him to timeout when he doesn't receive any recognizable FNBDT CM message from the far end terminal on time. Initially this timer is set to 30 seconds. Should this timer expire, the terminal would return to the Idle state assuming the far end terminal has made no response. It then can possibly send a new CM.

FNBDT is defined to allow terminals to work with different sets of *operational modes* and *keysets*. Four operational modes are defined : secure voice, secure data, clear MELP

[4]Mixed Excitation Linear Prediction
[5]Automatic Repeat Request
[6]Forward Error Correction

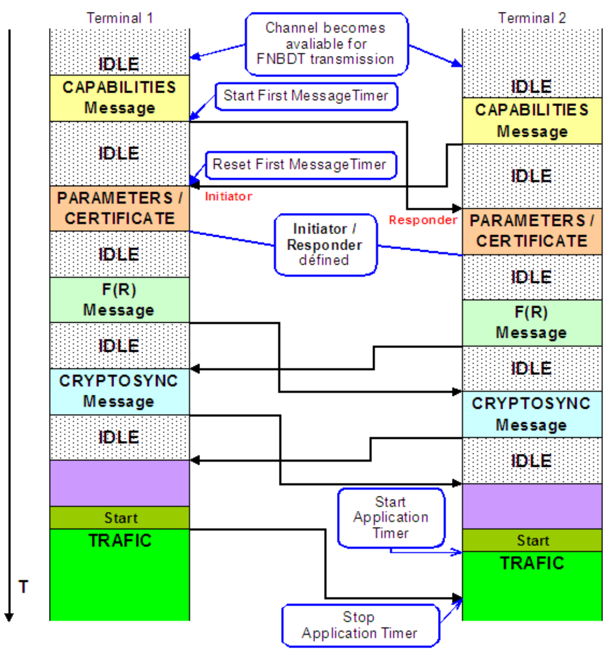


Figure 1. Secure Call Setup Signaling

voice and native clear voice. The fourth one is only offered as a possibility for the terminal to immediately leave the FNBDT call setup and use the underlying possibilities offered by the equipment. A *keyset* is a set of the definition of a key exchange protocol and of all data necessary to the use of this KEP. Different keysets can be defined for any specific use of the teminal. For example, one keyset can be defined for a *National* use, a second one for a *NATO nations* use and a third one for *Non NATO nations* use. One terminal having one of these keysets implemented should be able to discuss with all terminals having at least the same keyset implemented while a terminal not having the *National* keyset implemented should not be able to discuss with any terminal using this keyset (they still may possibly be able to use another common keyset). Each terminal gives a priority to his usable *keysets*.
Examples :

case 1

T1	T2
National NATO NonNATO	 NATO NonNATO

case 2

T1	T2
National NATO NonNATO	National

case 3

T1	T2
National NATO	 NonNATO

case 4

T1	T2
National NATO NonNATO	NATO National NonNATO

• In case 1 : the terminals are able to discuss using *NATO* or *NonNATO* keysets but T1 will not be able to establish a communication in *National* mode.
• In case 2 : the two terminals will carry their communication over *National* keyset mode.
• In case 3 : the two terminals have no common keyset. They will not be able to establish any secure communication.
• In case 4 : the two terminals are able to use all keysets but haven't the same priority of use. The choice will depend on the roles that will be allocated.

Operational modes and keysets are linked together. A terminal may associate different keysets to each of his operational modes. For example, a terminal may define :

Operational mode	associated keysets
secure voice	National keyset 1 NATO keyset 1
secure data	NATO keyset 2 Non NATO keyset 2

Where *NATO keysets* 1 and 2 may or may not be the same.

In order to assure the two terminals to choose a common operational mode and keyset for their session, the protocol defines a procedure leading the two terminals to define a *role* through a specific *Initiator/Responder* procedure. The CM includes a specific eight bits field; one bit allowing the terminal to define a preferred role is named *I/R bit*; the other seven bits represent a random number. Priority is given to the *I/R bit*, the random number is only used in the case where both terminals have defined the same preferred role into their exchanged CMs. The leader takes the *Initiator* role, the other has the *Responder* role.
Globally the process to select the terminal's role can be summed up as in the following table (where *RN(s)* the random number sent in its CM by the terminal and *RN(r)* the random number received in the distant's CM) :

Table 1 : Definition of Initiator/Responder role

	CM received : RN(r)			
	I/R bit		1	0
		RN(s) % RN(r)		
CM sent : RN(s)	1	>	I	I
		=	ND	I
		<	R	I
	0	>	R	I
		=	R	ND
		<	R	R

I = *Initiator*
R = *Responder*
ND = *Not Defined*

In the *Not Defined* case, (i.e. it isn't possible to define different roles) the terminal enters a fail call procedure and returns to the *Idle* state. he then can possibly send a new CM.
The role assigned to each terminal allows him to define which operational mode and associated keyset will be used. The first definition is the Op-mode : the first mode in the *responder*'s list that meats the first choice of the *initiator*'s list is chosen. If no common mode can be defined at this step, then the same is done with the second choice of the *initiator*'s list, etc. The same procedure is then applied to the associated keyset list. For example, in case 4 above, the *National* keyset would be chosen if T1 acts as initiator, but the *NATO* keyset would be in the case where T2 acts as initiator.

In all states, in case a terminal receives an *unrecognized message*, he may silently discard it and remain in the same signaling state as prior to receiving it or invoke a fail call procedure (paragraph 2.2.1.3 of the FNBDT-210 specification).
In the current study here, we consider the case where the terminal ignores the message he receives if this one doesn't fit the one he is waiting for within the logical progress of the protocol.

2.3 Parameters/Certificate, F(R) and Cryptosync Message exchange

After the CM exchange has been completed, three other messages are exchanged by the terminals :
• The *Parameters/Certificate* message (PC) contains all the parameters associated to the chosen operational mode and the user certificate issued for the chosen keyset.
• The *F(R)* message (FR) is used for the exchange of values necessary to the key exchange protocol (KEP). However, the exact definition of its values is not fully defined into the [FNBDT-210] specification, this is part of the FNBDT-230 document.
• The *Cryptosync* message (CS) has two main fields : The IV (Initialization Vector) used for the cipher application and an encrypted packet for the control of the good synchronization of the cipher mode of both terminals.
Those three messages must be sent in that order. It is possible for a terminal to send his F(R) before he receives the far end's PC, but he must wait for the far end's FR before he is able to send his CS.

2.4 Fail Call Notifications, Other Messages

The Fail Call procedure may be invoked in some cases :
• When the *First Message Timer* times out. This occurs when the terminal receives no CM from the far-end before the timer expires while he is waiting for the distant's answer after he has sent his own CM.
• When a message is received but not valid (ex. : CM received but no common operational mode can be defined).
In all cases, when this procedure is invoked, the terminal sends to the far-end a *Notification* message (NM) indicating the action executed : Return to *idle* state or close communication and go to *NotConnected* state. As a consequence, such a notification message can possibly be received at all moment.

3. UPPAAL

UPPAAL is an integrated tool environment developed jointly at Uppsala University in Sweden and Aalborg University in Denmark. It is used for modeling, validation and verification of real-time systems such as controllers or communication protocols in which timing aspects are critical.
"UPPAAL is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical." [UPPAAL]
The Graphical user interface (GUI) of the UPPAAL tool is used for modeling, simulation and verification.

3.1 Modeling

One configuration of a system in UPPAAL consists in a tuple $\{\mathcal{L}, v\}$ where $\mathcal{L}$ is a location vector indicating the state of each of the automatons and v is a value of the variables. Semantics of this model has three types of configuration changes :

- *Time* can progress through states with a duration d as long as each of the invariants of these states remain satisfied. Clocks values increase of d and integer variables are not modified.
- *Synchronization* can occur when two complementary actions are made possible and the guards associated to the edges are satisfied. Corresponding states are modified and clocks and integer variables values are set according to update indications.
- *Internal actions* (no synchronization) are possible to any process as soon as the guard of an edge is sat-

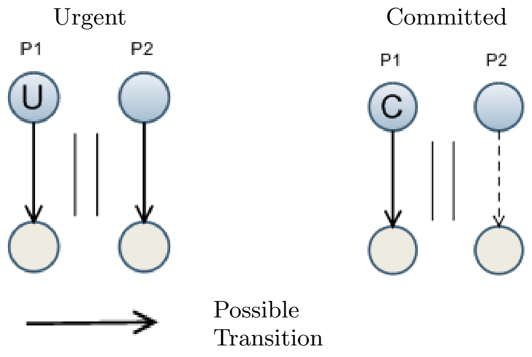
isfied. This action is made independently from other components of the system. Clocks and integer variables values are modified according to update indications.

Transitions between locations are labeled through :

- *Guards* which are conditions on values of variables. These have to be satisfied to go through transition.
- *Synchronizations* which allows processes to synchronize over channels. Generally, the absence of synchronization indicates an internal action of the process.
- *Updates* which allow the modification of variables and clocks. Updates of expressions change the state of the system.

In UPPAAL, a model consists in a set of timed automata which can be linked and synchronized through binary channels of type send/receive. These structures have two types of variables : Clocks evolving synchronously with time and discreet bounded integers. Any state of the automaton can have conditions on variables called *Invariant* which has to remain satisfied as long as the system remains in this state. Locations in UPPAAL can be of three types : *normal*, with or without invariants, *urgent* and *committed*.

- In *urgent* locations, time cannot progress but interleaving with normal states remains allowed.
- In *committed* locations, not only time cannot progress, but also the next transition must make the system to leave this location.



Each template must also have one and only one *initial* location.

3.2 Simulation and verification

Simulation and verification of systems developed under the GUI can be achieved through the *Verification server* and the *Verifier* of the UPPAAL tool.
• The *Verification server* is used in the simulator of the GUI to compute successor states. It enables to examine all possible dynamic executions of a system during design or to visualize traces of executions generated by the Verifier.
• The *Verifier* allows requests of type "Exists", "For all" or "Leads to" which are passed to the *Verification server* through a *Requirement Specification Language*. It is used to check safety and liveness properties by on-the-fly exploration of the state-space of a system in terms of symbolic states represented by constraints.

In our study we use the UPPAAL verifier tool to test some properties linked to the *First Message Timer* used into the FNBDT protocol.

4. A SIMPLE MODEL OF FNBDT CALL SETUP WITH UPPAAL

[FNBDT-210] describes an algorithm of the FNBDT Call Setup. Our model intends to remain as close as possible from the algorithm specified into this document (Figure 2).

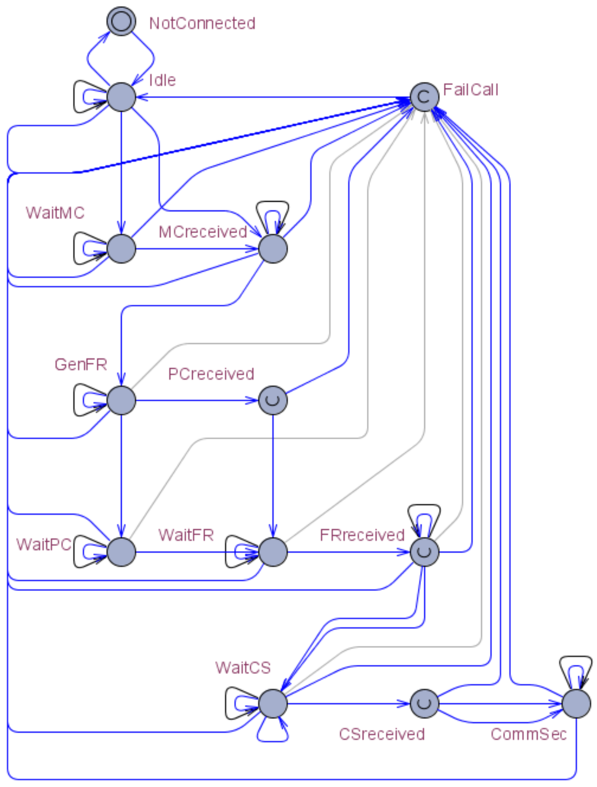


Figure 2. UPPAAL model for FNBDT Call Setup

The model has 13 locations which, for most of them, are of type *Wait for Type of message* and *Type of message received* (WaitMC, MCreceived, WaitPC, PCreceived, WaitFR, FRreceived, WaitCS, CSreceived). One special location is reserved when the *FailCall* procedure is invoked (FailCall). Three are the locations where the terminal is at the beginning (NotConnected, Idle) or at the end (NotConnected, CommSec) of the Call Setup process.
Finally, the global system consists in the use of two different templates as described below.
The model will be available at the author's web site.

4.1 Transmission of messages from one terminal to another

The transmission of messages from one terminal to another is executed through two lists accessible to both terminals. The terminal *sends* his messages by writing them at the bottom of his *sending* list -which consequently is the *reception* list of the far-end terminal- and *receives* the far-end terminal's messages by picking them up at the top of his *reception* list -which is also the *sending* list of the far-end terminal-. This is a good approach to simulate the *Message Transport* layer described in the first part of [FNBDT-210].

4.2 Fail Call Notifications, Other Messages treatment

When the *Fail Call* procedure is invoked, the terminal sends to the far-end a *Notification* message (NM) indicating the action executed : Return to *idle* state or close communication and go to *NotConnected* state. We've seen that, when a terminal gets back to the *Idle* state, as long as the data channel between the two terminals is not closed, he may possibly send a new CM to the far-end. This doesn't mean that the messages already sent through the opened data channel are destroyed. As a consequence, at all states except those where the terminal's only possible action is to send a message (*FRreceived*, *CSreceived*), he must be able to receive and process any message in his *reception* list. This is done in the model through the two loops at each state (Figure 3).

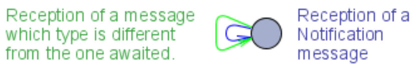

Reception of a message which type is different from the one awaited.
Reception of a Notification message

Figure 3. Notification and other messages reception

In any case, the reception and treatment of a *Notification* message is first priority over any other action.

4.3 Definition of Random Number

We've seen above that, in order to avoid the problems linked to the possibility that both terminals may send their CM at the "same" time, the terminal defines a 7 bits random number he sends in his CM. This is done to allow terminals to define their role as *Initiator* or *Responder*.
To simplify the model, we've let the terminal use only two numbers (basically 1,2). As a matter of fact, this is sufficient to resolve all cases defined at Table 1. The results can easily be extended to the use of a 7 bits number.

5. RESULTS

FNBDT signaling specification may be sometimes unclear on some points. One of these we pointed out concerns the treatment of messages when a terminal returns to the *Idle* state (for example when after its *First Message Timer* has expired) without closing the underlying data channel.
Our feeling was that some unconsidered ways may occur due to the "ignore unawaited messages" assumption.

5.1 Assumptions and restrictions

We've seen at 2.2 that, when a terminal receives an *unrecognized message*, he may perform different procedures. We place here in the case where :
• (AS1) When a message is received but is not of the type awaited, it is ignored.
When a terminal receives a CM, he may specify his *preferred role* into his response and let the choice of the role be defined by the *Random number*[7]. Here, we've chosen a procedure to minimize the possibility of glare :
• (AS2) When a terminal sends his capabilities first (i.e. without having received a CM from the far-end), he specifies his *I/R bit* as *Initiator*.
• (AS3) When a terminal sends his capabilities after he has received a CM from the far-end (i.e. he answers a first message), he specifies his *I/R bit* as the opposite of the one specified in the received CM.
It is easy to extend our results to a more general case.

[7]One may even remark that it is always possible for the terminal receiving first the CM from the distant, to decide the role he wants to play. If he wants to be *initiator*, it is sufficient for him to answer with the *I/R bit* set to initiator and with a "*Random number* larger than the one he has received.

5.2 Properties to test

We've seen at paragraph 2.2 why both terminals have each to get a different role. We've wanted to test whether it was possible for them to take the same role despite of the rules (*I/R bit*, *Random number*) defined by the protocol.
Given a system, we've used the UPPAAL verifier with the following query :
$E <> (T0.CommSec) and (T1.CommSec) and (T0.IRbit == T1.IRbit)$
Which means : *It exists one configuration of the system where both terminals are in the CommSec location and they both have defined the same role.*
Notations : On all figures above, MESxy means MES=type of message, x=terminal's number, y=session's number. For example : CM12 = *Capabilities Message* (CM) sent by terminal 1 for session number 2 where "session" represents an ordered subset of one or more messages belonging to an ordered set {CM, PC, FR, CS} which can be initialized by a terminal each time he enters the *Idle* state.

5.3 Result one

As we can see through the trace bellow (figure 4), we've got a positive result to our request.

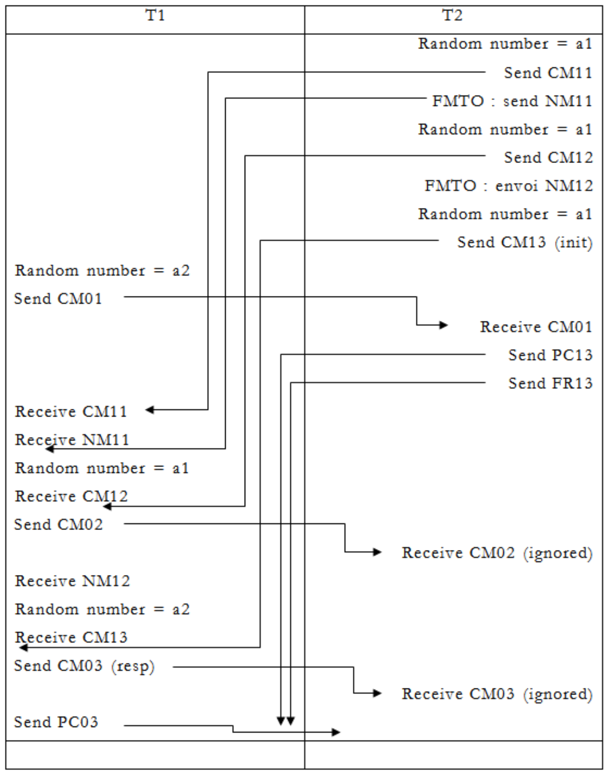


Figure 4. Identical choice of role (1)

• Terminal 0 defines his role to *Responder* (CM03) since he answers a *Capabilities* message from the far-end (CM13) defining a role of *Initiator* {AS3}.
• Before sending his (CM01) message, terminal 0 defines his random number as "a2". Before sending his (CM13) message, terminal 1 defines his random number as "a1". Terminal 1 then defines a role of *Responder* since he compares

the two random numbers included in the two messages with (a1 > a2) (see Table 1).

This result is due to the {AS1} assumption ignoring a CM message when this one has already been received. Here T1 ignores CM02 and CM03 since the terminal considers it has already received the far-end *Capabilities* message and awaits for a *Parameters/Certificate* message.

5.4 Result two

In this second case, we've forced the system to use no random number (i.e. we forced the processes to use only the same number). Our goal was to test whether there existed a way for both terminals to take the same role without any use of the random number. The UPPAAL query was the same as above.
Surprisingly, the tool also returned a positive result (Figure 5). In that case, both terminals also define the same role. However, they even don't use the random numbers comparison to define it.

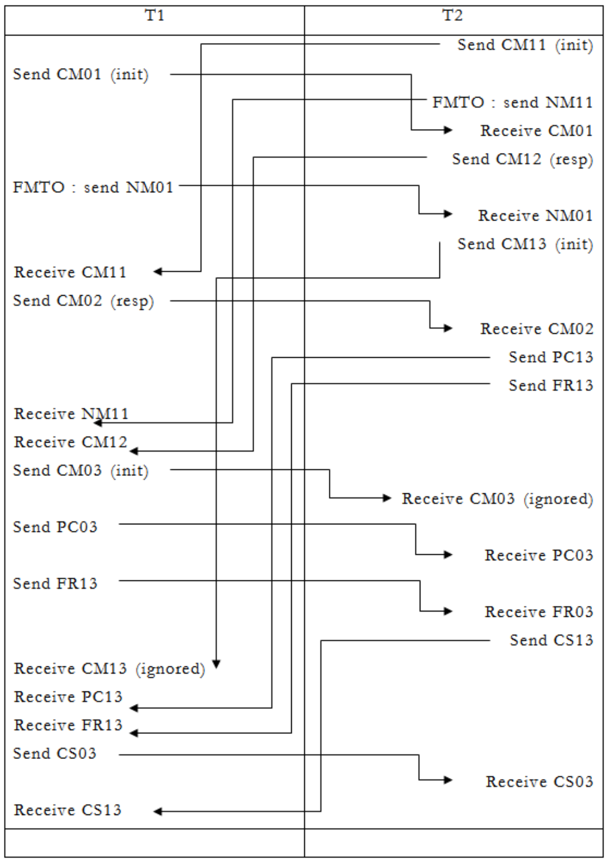


Figure 5. Identical choice of role (2)

• Terminal 0 defines his role to *Initiator* (CM03) since he answers a *Capabilities* message from the far-end (CM12) defining a role of *Responder* {AS3}.
• Terminal 1 defines his role to *Initiator* (CM13) since he sends a first *Capabilities* message (CM13) {AS3}.

It is obvious that here too, the result is due to the {AS1} assumption ignoring a CM message. Here CM13 is ignored by T0 and CM03 is ignored by T1 for the same reasons as in our first result.

However, one can consider that those situations have no consequence on the security of the system since none of them leads to place the system in a security failure. We emphasize the fact that these are not normal situations and shouldn't happen.
Our model allows to easily prove that these situations can't happen if only one Call Setup procedure is permitted on an opened data channel : i.e. on any problem, the data channel is closed. But there would be no interest in that case to specify a possible return to the *Idle* state.

6. CONCLUSION

The FNBDT protocol, which we should now name SCIP intends to become a standard for NATO interoperable voice communications. It is already implemented into a large number of equipment developed in the United States. As far as we know, no formal model of it, for which at this time very few documentation remains available, can be found over the Internet.
We've described here a first simple model under the UPPAAL tool of the FNBDT *Call Setup* protocol that leads us to some possible not awaited situations. As a matter of fact, these are mainly the consequence of some assumptions made through our reading of the process specification and could easily be avoided by a simple limitation of this process. In future work however, we want to propose a simple modification based on the inclusion of "session" numbers as we've defined them here that would also avoid these situations without limiting the process. We also want to extend our study to a larger one using our model to specify new properties to test.
By the way, since the only [FNBDT-210] specification available over the Internet is dated of September 1999, we suspect that it is highly probable that this one has already been modified on some points.

REFERENCES

[FNBDT-210] General Dynamics Communication Systems, "FNBDT Signaling Plan rev 1.1", *www.dtic.mil* 1999.

[UPPAAL] Uppsala University and Aalborg University, "UPPAAL help", *www.uppaal.com* 2008.

[BDL04] G. Behrmann, A. David, K.G. Larsen, "A tutorial on UPPAAL", *www.uppaal.com* 2004.

[WIKI] Wikipedia "SCIP", *www.wikipedia.org/wiki/SCIP.*

[NC3A] NATO NC3A "SCIP", *elayne.nc3a.nato.int/msec/scip/index.html.*

[DTSBRBH02] Daniel E.J., Teague K.A., Sleezer R., Brewer J., Raymond J., Beck W.J. Hershberger J., "The Future NarrowBand Digital Terminal", *Circuits and Systems, 2002.*

[DT02] Daniel E.J., Teague K.A., "Simulation of FNBDT over Internet and 3G wireless networks", *Circuits and Systems, 2002.*

[DT01] Daniel E.J., Teague K.A., "Performance of FNBDT and low rate voice (MELP) over packet networks", *Signals, Systems and Computers, 2001.*

[SDRBT02] Sleezer R., Daniel E., Raymond J., Brewer J., Teague, K., "Counter Mode Encryption for FNBDT/MELP", *Proceedings of the IEEE Midwest Symposium on Circuits and Systems* 2002.