Network Working Group

Internet Draft

Expiration Date: November 1995

C. Huitema

INRIA

May 1995

Multi-homed TCP

C. Huitema

Christian.Huitema@sophia.inria.fr

draft-huitema-multi-homed-0.txt

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet- Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the

``1id-abstracts.txt'' listing contained in the Internet- Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

2. Abstract

Current TCP connections are identified by pairs of host addresses and port numbers. This introduces a "fate sharing" dependency between the connection and these values, and specially with the time to live of the host addresses. There are at least three cases where this dependency is

unduly harmful:

(1) when the host moves to a new location and is assigned a new address,

(2) when the interface used by a multi-homed host to initiate the connection is temporarily disconnected,

(3) when the host's network is renumbered, e.g. after changing provider.

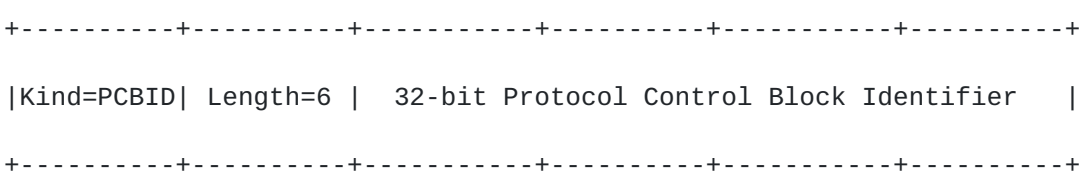
We propose to remove this fate sharing effect by allowing the set of addresses used by a TCP connection to change over time. We modify TCP defining a new type of parameter, PCB-ID, to be used during the initial synchronisation. This parameter identifies the "Protocol Control Block" associated to the TCP connection. When initiating a connection, the host attach to the SYN packet the identifier of the local PCB. If both hosts have identified their local PCB, they can now exchange "Extented TCP" packets, where the pair of 16-bit port numbers has been replaced by the 32-bit PCB-ID at the destination. This way, PCB location becomes independent of the IP addresses. The addresses in use can be stored in this PCB, so that we can change these in the course of the connection. In fact, we can use several addresses in parallel for the same connection, which is why our proposal is called "multi-homed TCP".

3. Proposed extensions to TCP

Our proposal requires two extensions to TCP: the definition of a context parameter during the synchronization exchange and the carrying of this parameter in the data packets.

3.1. The context identification parameter

When a TCP entity initiates a new connection, it can announce its willingness to receive data from alternate addresses by including a "PCB identification" parameter in the initial "synchronization" packet. This parameter has the following format:



This option is a unilateral offer, not subject to negotiation. It allows the partner to send data from an alternate source address, if indeed the "local contex id" is somehow repeated in the new packets. The context-id is normally the identification of a "protocol control block"; however, the precise format and bits carried in this parameter are entirely a local matter. The only condition is one on reuse: in order to avoid protocol errors due to old packets popping out of the network, the context-id used by one connection cannot be reused immediately. One must wait for the end of the "TIME- WAIT" state.

The parameter kind PCBID is to be allocated by the IANA.

This parameter can only be present in TCP packets where the SYN bit is set.

3.2. Incoming data

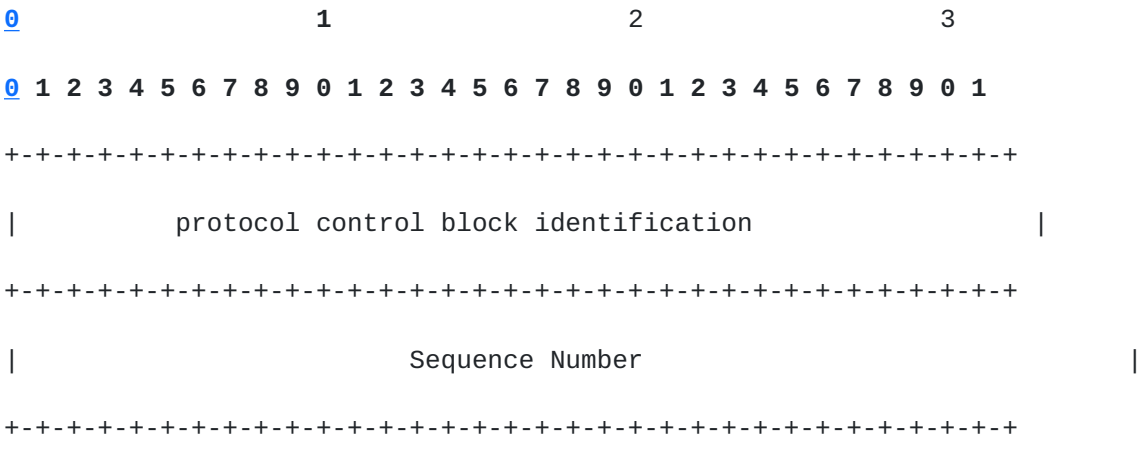
An entity that has learned the PCB identification of its partner can elect to send "Extended TCP" packets. The difference between extended TCP and regular TCP packets are that:

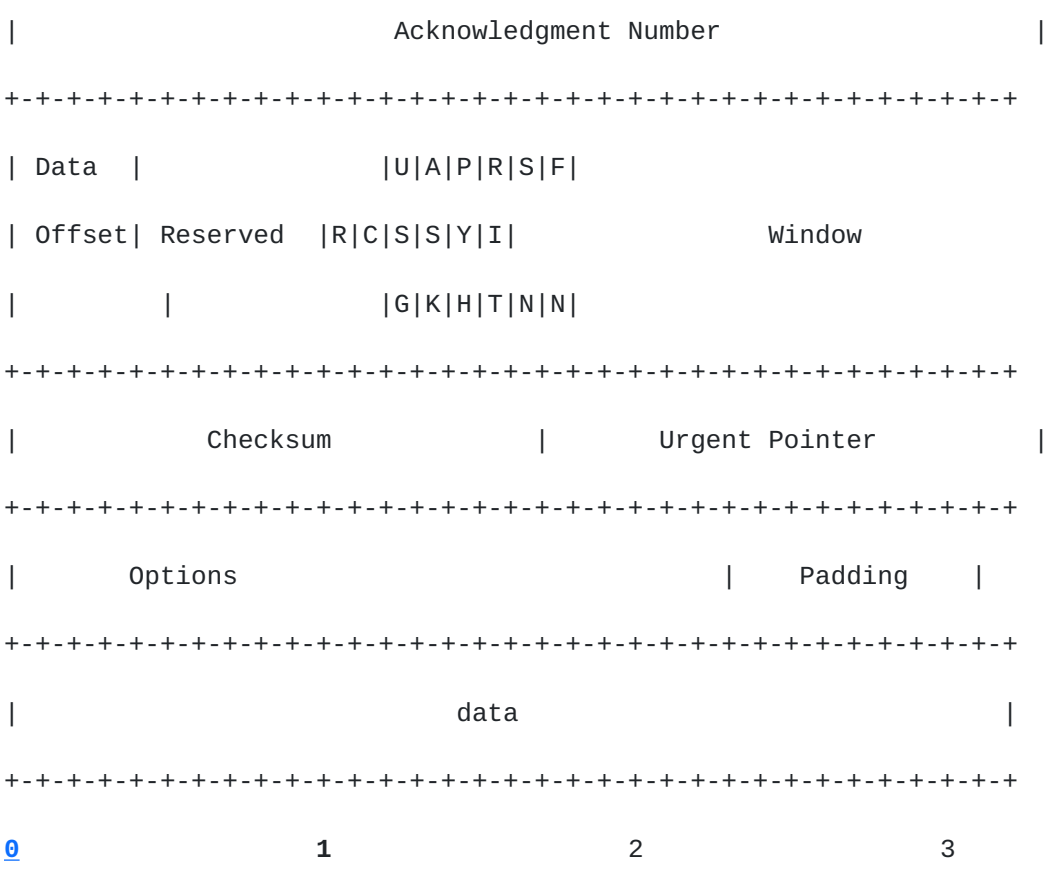
- the IP protocol type is (a value to be defined by IANA), not 7 (TCP).

- the IP source and destination addresses may be different from the ones use in the SYN packet.

- the first 32 bits of the ETCP header carry a protocol control block identification, not a pair of ports.

The extended TCP packets have a format identical to normal TCP packets, except for the replacement of the port pair by a context identification:





ETCP Header Format

ETCP packets are delivered to an ETCP incoming process, which merely looks at the context identification and finds the corresponding PCB. All packets arriving on the PCB identified by the context XB are treated exactly as if they had been sent by host A to host B, using the TCP ports "a" and "b", i.e. the ports present in the TCP packet.

3.3. Extended checksum rules

The checksum of ETCP packets is computed over the pseudo-header and the packet itself. The pseudo header is composed of the IP addresses

actually present in the packet.

3.4. PCB extension

The protocol control block of TCB is extended to include a list of partners addresses. This list is initialized with the partner address expressed in the connection request. It is populated with the source addresses from which ETCP packets bound to that PCB are received. Two dates are associated to each address, "first seen" and "last seen", i.e. the date at which a packet was received from this address the first time, and then the last time.

The list is sorted by reverse order of "last seen" packets, i.e. most recently seen first. TCP entities should normally send their packets to the most recently seen address. They may elect to spread them over all of the "currently active" addresses, i.e. all addresses whose "last seen" date is larger than the "first seen" date of the most recently seen address.

4. Finding addresses

The TCP "protocol control block" will be extended to contain a list of IP addresses of the source entity. These addresses are obtained by monitoring the IP source addresses of the ETCP packets.

In this section, we will see the address reassignent at use in four cases, deprecation, anycasting, multi-homing and mobility.

4.1. Deprecation

The address autoconfiguration and neighbor discovery procedures of IPv6 allows hosts to renumber their interface. This is supposed to occur in three phases:

1) Initially, the interface of host A is numbered with the address A1,

2) After reception of a router advertisement message, the host A decide that the preferred address of the interface is now A2. Packets sent to the deprecated address A1 can still be received for some period.

3) At the end of that period, the address A1 is invalidated. Packets sent to it are discarded by the network.

Let's discuss first the behaviour of the host A, which is engaged in a TCP connection with the host B, using ports a and b and PCB identifiers xa and xb, respectively. Initially, packets will be exchanged using the addresses A1 and B1.

```
          (A1, B1, tcp) (a, b, SYN, PCBID=xa) ===========>
                 <============ (B1, A1, etcp) (xa, SYN, ACK, PCBID=xb)
          (A1, B1, etcp) (xb, ACK, DATA) ===========>
                 <============ (B1, A1, etcp) (xa, ACK)
```

As soon as the address A1 becomes deprecated, A will stop using it. The

next data or acknowledgement packets will be sent from the preferred source address A2:

```
          (A2, B1, etcp) (xb, DATA) ===========>
                 <============ (B1, A1, etcp) (xa, DATA)
          (A2, B1, etcp) (xb, ACK, DATA) ===========>
                 <============ (B1, A2, etcp) (xa, ACK)
```

The address change is noted by B because upon reception in context "xb" of a packet from the new address A2. At this stage, B switches to the new address. Due to the propagation delay, some packets sent to the old address may still be in transit at this point. This will last for at most twice the maximum propagation delay. We may then ensure that no packets will be lost if the old address remains valid for at least twice that delay, e.g. more than four minutes.

4.2. Anycasting

In order to use IPv6's anycasting facility, we should be able to send the initial SYN packet to an anycast address. Such connection requests are doomed in today's TCP, because anycast addresses cannot be used as source addresses. ETCP solves the problem rather elegantly. Suppose that the packet sent by the host A to the anycast address X is actually routed to the nearest server for that anycast address, B, through its interface B1.

```
          (A1, X, tcp) (a, b, SYN, PCBID=xa) ===========>
```

```
                <============ (B1, A1, etcp) (xa, SYN, ACK, PCBID=xb)
          (A1, B1, etcp) (xb, ACK, DATA) ===========>
                <============ (B1, A1, etcp) (xa, ACK)
```

Because B knows the PCB identifier "xa", it can sent an ETCP packet back towards A, using the regular address B1 as source address, advertizing its own PCB identifier. A notes the source address B1 in its PCB, and the following packets are happily exchanged between A1 and B1.

The solution can in fact be extended to other variations of anycasting, e.g. when the initial packet is sent to a multicast group. The only difference here is that several response will come, e.g. from B and C, two members of the group which both decided to reply to A.

```
          (A1, G, tcp) (a, b, SYN, PCBID=xa) ===========>
                <============ (B1, A1, etcp) (xa, SYN, ACK, PCBID=xb)
          (A1, B1, etcp) (xb, ACK, DATA) ===========>
                     <============ (C1, A1, etcp) (xa, SYN, ACK,
PCBID=xc)
          (A1, C1, etcp) (xc, RST) ===========>
          (A1, B1, etcp) (xb, DATA) ===========>
```

Upon reception of the synchronisation packet from C, A realizes that this packet is a duplicate SYN, that the PCB identifier is different from the one advertized by B. It will refuse to consider the packet's

informations, and immediately send a "reset" to C, clearing the additional connection.

In the rare case where the binary values of xb and xc would be identical, A should simply discard the additional asynchronisation packet. It must not add the corresponding source address to its PCB.

4.3. Multi-homing

Suppose that the host A is equipped with two interfaces, A1 and A2. It may decide to spread the traffic over these two interfaces, sending packets alternatively from sources A1 and A2. B, in turn, will send packets alternatively towards each of these addresses:

```
          (A1, B1, tcp) (a, b, SYN, PCBID=xa) ===========>
                 <============ (B1, A1, etcp) (xa, SYN, ACK, PCBID=xb)
          (A1, B1, etcp) (xb, ACK, DATA) ===========>
                 <============ (B1, A1, etcp) (xa, ACK)
          (A2, B1, etcp) (xb, ACK, DATA) ===========>
                 <============ (B1, A2, etcp) (xa, DATA)
          (A1, B1, etcp) (xb, ACK, DATA) ===========>
                 <============ (B1, A1, etcp) (xa, ACK)
```

This is possible because B keeps in the PCB identified by "xb" a list of partners addresses. Both addresses will in turn be the "most recently seen" address.

4.4. Mobility

Mobility is treated by exactly the same mechanisms as those of "renumbering" which we expressed in 4.1. There are however two problems, fast movements and double jumps. There are few good solutions to the "fast movement" problems. If a host gets a new address each fraction of a second, there is just no way that a mechanism that exchange information with communicating hosts through the whole internet will keep pace. Proposed solutions involve home base and tunnelling. They amount to saying that the mobile host is dual homed. It has a "mobile address", which changes when the host roams, and a "home address", which remains relatively static. When the host moves very fast, it should only use its home address. When the host moves slowly, it can start using the "mobile address" for more efficiency.

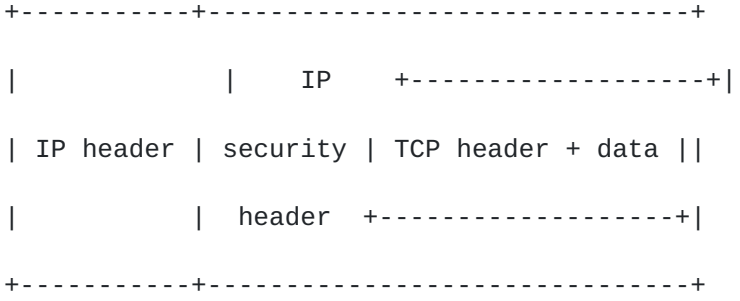
The double jump problem occurs when two mobile hosts change cells simultaneously. In this case, we are in a situation were neither of the previous addresses is usable. A cannot tell B that its address changed from A1 to A2, because the packets bound to B1 don't make it to B, and vice versa.

The only solution here is to request that mobile host maintain a stable home address, e.g. A0 for A and B0 for B. They will "refresh" the address list of their partners by regularly sending messages from these addresses. In the double jump situations, the deadlock will be broken if either A or B manages to get a packet to the home addresses A0 or B0.

5. Security considerations

The procedure that we propose introduces a significant security hole. If a third party can obtain the "context-id" used by the TCP connection, then it can send TCP packets using a bogus source address and trick the TCP entity into believing that this is the new address of its partner: it can, in short, "grab the connection". In the current Internet, such connection grabbings require that one somehow subvert the routing protocol, which is more difficult than simply listening to traffic and forging packets. This threat is in fact inherent to the mobility environment which we are willing to accomodate: we have the choice of accepting this mode of operation or implementing a secure version of TCP.

Implementing a secure version of TCP may be a good idea in any case, as TCP is currently vulnerable to several forms of attacks: as long as the origin of the packet is not securely authenticated, it is easy for intruders to send forged TCP messages and distort the state of an existing connection. We may be willing to use "IP security" to that effect, wrapping the TCP packets inside an "IP secure" payload:



The IP security header identifies a "security context", to which a key is attached, as well as the type of protection. Once a security context has been established, we obtain a proof of the source Internet address, as well as in some cases an integrity check, an confidentiality, i.e. protection against eavesdropping. How secure IP contexts are established is outside the scope of these memo.

If such protections are available, then our procedure becomes entirely acceptable. In fact, we can define three mode of operations:

(1) Secure mode, into which only those packets which were transmitted securely can be considered by TCP,

(2) Protected mode, into which packets are requested to come either from a well identified address or through a secure channel,

(3) Unprotected mode, for users who are willing to take chances.

The second mode of operation protects us against the specific threats brought in by mobility, without imposing the performance penalties of cryptography on each and every packet.

We should note that it is theoretically feasible to associate security parameters to the TCP connection, in much the same way that they are associated to a pair of IP hosts. This would solve the TCP problem entirely, making our connection very secure. But this is clearly a point of debate.

8. Acknowledgements and related works

Many of the features of this proposal were discussed privately with Lixia Zhang, John Wroclavski, Van Jacobson and Greg Minshall. I received comments from Allison Mankin on a previous version of this proposal. The support of multiple addresses bears some similarities with the "transport context names" proposal of Dave Crocker.

Various authors, notably Steve Deering, have proposed an alternative "context identification" mechanism based on "volatile port numbers": a specific port is assigned to each connexion within the SYN exchange, which is essentially equivalent to our 'context identifier". The scheme that we propose here has the advantage of using larger identifiers: 16 bits does not necessarily allow enough contexts. It also has the advantage of being entirely backward compatible (an additional TCP parameter can be ignored) and of being explicit: alternative addresses can only be used if the peer provided a context identifier.

Appendix A. Relation with other proposals.

Letting TCP connections survive a change of addresses is not a new problem.

Partial solutions have been proposed, which essentially insert a "permanent identifier" layer between the transport control and the routing process. Instead of identifying the transport contexts by a pair of source and destination addresses, one would use this "more permanent" identifiers, often called "end point identifiers" or EID. The addresses may change but the EID will remain constant. In the "classic" version of the EID proposal, these identifiers will be present in each and every packet, in addition to the addresses. Several implementations have been proposed, e.g. as innovative IPng formats. Including EIDs in the IP or IPng header itself has been ruled out during the debates, but one could design an intermediate layer between IP and TCP that would carry those headers. The proposal, however, suffers from three drawbacks:

(1) As EID are likely to have the same size than the IPng addresses, we are considering a sizeable overhead.

(2) EID don't exist yet. Creating yet another address space would implies significant administration efforts.

(3) Disconnecting identification and routing removes whatever security there is in the current IP protocol.

The early SIP and SIPP proposals included a scheme that would alleviate the first concern and eliminate the second one. Instead of creating a new numbering space for EIDs, one would simply specialize one address, e.g. the one that was used for sending the initial synchronization packet of TCP. Most connections will simply go on using that address for

their whole duration. If the host moves, or if a new provider is selected, the old address will be still be used as nominal source or destination address, but the packet would be explicitly source routed through the new address. This solution however suffers from the same security problems as the generic EID proposal, and does not entirely cure the "change of provider" problem. If the local network is renumbered, the old addresses ceases to be valid.

In fact, that argument applies to all approaches based on "permanent identifiers": we are exchanging a dependency on the IP address for a dependency on these new identifiers. Proponents of the EID approaches are ready to swear on their most sacred books, or on the head of their favorite dog, that these new numbers will remain valid, unique and stable for eternity. We may note however that there is not a single human construct that ever met this requirement. The case is even more obvious if we focus on network history, where exceptions to the unicity and stability rules have been found for all proposed "unique numbers" that were considered, be they IP addresses, which get renumbered, domain names, which change from time to time, or IEEE-802 "MAC addresses", which can easily be misconfigured.

Rather than yet another gigantic administrative effort, our solution involves only locally asserted numbers, the PCB identifiers.

3. Why TCP

Our solution focuses on TCP. One may well object that TCP is not the only protocol used over the Internet, that the fate- sharing also occurs with other transport protocols, notably over UDP. However, a quick analysis shows that "user
controlled" applications, that use the user datagram protocol, are much less sensitive than those application which relie on the services of a transport layer. This is notably the case of "client server" and "real time" application; this is also the case of "secure" applications.

The most popular building block for "client server" applications is the "stateless RPC". In this model, a client simply sends a request to the server, generally a single UDP datagram. Upon reception of this request, the server executes the required actions and sends back a reply, in most cases a single UDP datagram. No "context" is kept by the server, and the time-span of the transaction is generally quite short, which minimizes the risk of observing a move in the middle of a transaction. At worst, if such a transistion occurs, the client may repeat its request. Indeed, if the server moves, the client will have to find its new location, its new address. But that has to be solved by updating the routing process or the name service; it is not a "transport control" problem. It is true that client-server applications may have to maintain long term associations. But they generally do so by identifying "application level contexts", which are identified by the users and servers names: authentication and access controls ought to be based on user names, not on hardware addresses.

Real time applications also use UDP, generally in conjunction with

"multicast transmission". The practical experience have shown however that the multicast transmission is very often achieved through a cascade of relais, and that the source address received in the packets is not always that of the actual source. The "real time transport protocol" includes an identification mechanism by which stations are assigned numbers within multicast groups. Real time applications go to great length to reduce their packet size, using computation intensive compression techniques; an additional header in each and every packet would be entirely unwelcomed.

In fact, one may draw the general conclusion that applications which are using datagram services have to incorporate their own control functions. Identification of the partner, as well as general security measures, are an integral part of these applications. Providing them with a "one size fits all" intermediate layer would probably not help.

Then, one may also observe that even if one were to implement a "transparent" solution, e.g. using some form of stable identifiers, the TCP code would still need to be updated. The error and throughput characteristics of mobile networks are different from those of fixed networks; controlling parallel transmission over multiple paths is not quite the same as controlling a single path. Key points of the implementation, such as the retransmission strategy or the congestion control algorithm would have to be revised in any case.

Appendix B. Managing a set of addresses

We have mentioned the need to maintain a "working set" of partner addresses, and to "grade" each address in that working set. However, the procedure described in the previous section only allows for addition of addresses to the connection, not for their removal. This is because we expect that the working set of addresses will simply be a cache of the "most recently used" address of our partners. TCP entities may use whatever strategy you want to maintain the set of addresses within reasonable limits, e.g. LRU replacement. We will describe here some possible manners of grading the addresses.

The first seen and last seen monitoring will produce a first selection of selectable addresses within the working set. However, all the selected addresses should not necessarily be used. Even if several of them should be used, they need not be all used at the same rate.

Suppose that we have sent different packets to different addresses. If one of these addresses has becomed invalid, packet sent to it will be lost. Indeed, packets sent to valid addresses may also be lost, e.g. if a cosmic ray hits a photodiode at the wrong moment. It is however quite reasonable to relate the selection of addresses with the error control: if a packet sent to one particular address failed, further packets should preferably be sent another acceptable address.

Current TCP implementations try to characterize the network they are using by computing round-trip time estimates and congestion windows. When several addresses are in use, one may well believe that each address will be reached by a different path, and that each of these paths will have different characteristics.

In fact, due to the sequential nature of the acknowledgment process, it would be very difficult to maintain more than one round-trip estimate for the connection. However, it is entirely conceivable to maintain a separate congestion avoidance window for each address in the working set, and to mainatin these windows by running parallel versions of the "slow start" algorithm. The window will be initialised when a new address is added to the working set; it will be increased when a packet sent to that address is acknowledged, it will shrink when a packet sent to that address is lost.

Maintaining a congestion window per active address provides for a rationale way of spreading the traffic over the various addresses. It is clearly not the only way. In fact, this may well open the path for very fruitful researches.