Networks and Protocols
International University Bremen
Dr. Jürgen Schönwälder

Course: 320301
Date: 2004-12-13
Deadline: 2004-12-13

Final Examination

Problem F.1: *internetworking* (2+2+2+2+2=10 points)

Indicate which of the following statements are correct or incorrect by marking the appropriate boxes. For every correctly marked box, you will earn two points. For every incorrectly marked box, you will loose one point. Statements which are not marked or which are marked as true and false will be ignored. The minimum number of points you can achieve is zero.

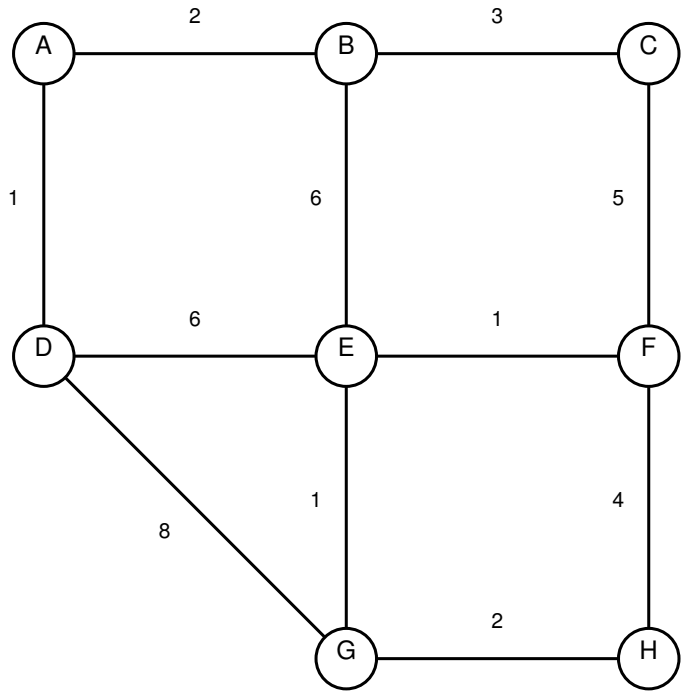
true	false	
☐	☐	The IPv6 interface identifier of IPv6 addresses assigned to IEEE 802 interfaces must be constructed from MAC addresses using the EUI-64 algorithm.
☐	☐	IPv6 uses Neighbor Discovery (ND) to map IPv6 addresses to IEEE 802 MAC addresses.
☐	☐	IPv6 auto-configuration provides the same services as DHCP for IPv4.
☐	☐	MTU path discovery is only needed in IPv4 networks.
☐	☐	Transport protocols such as TCP and UDP compute the header checksum over a pseudo header to exclude fields that are modified by routers.

Solution:

true	false	
☐	☒	The IPv6 interface identifier of IPv6 addresses assigned to IEEE 802 interfaces must be constructed from MAC addresses using the EUI-64 algorithm.
☒	☐	IPv6 uses Neighbor Discovery (ND) to map IPv6 addresses to IEEE 802 MAC addresses.
☐	☒	IPv6 auto-configuration provides the same services as DHCP for IPv4.
☐	☒	MTU path discovery is only needed in IPv4 networks and not in IPv6 networks.
☒	☐	Transport protocols such as TCP and UDP compute the header checksum over a pseudo header to exclude fields that are modified by routers.

Problem F.2: *shortest path routing* (15+2+3=20 points)

Consider the following network topology:



a) Determine the shortest-paths from node A to all destinations using Dijkstra's algorithm. For each step of the algorithm, fill in a row in a table structured as follows:

Step	A	B	C	D	E	F	G	H	Current	Permanent
0	0	∞	∞	∞	∞	∞	∞	∞		
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
...										

b) Which Internet routing protocol is based on Dijkstra's algorithm?

c) Explain the difference between interior and exterior routing protocols. Name interior and exterior routing protocols you are aware of.

Solution:

a) Computation of shortest paths using Dijkstra's algorithm:

Step	A	B	C	D	E	F	G	H	Current	Permanent
0	0	∞	∞	∞	∞	∞	∞	∞		
1	0	2(A)	∞	1(A)	∞	∞	∞	∞	A	A
2	0	2(A)	∞	1(A)	7(D)	∞	9(D)	∞	D	A,D
3	0	2(A)	5(B)	1(A)	7(D)	∞	9(D)	∞	B	A,B,D
4	0	2(A)	5(B)	1(A)	7(D)	10(C)	9(D)	∞	C	A,B,C,D
5	0	2(A)	5(B)	1(A)	7(D)	8(E)	8(E)	∞	E	A,B,C,D,E
6	0	2(A)	5(B)	1(A)	7(D)	8(E)	8(E)	12(F)	F	A,B,C,D,E,F
7	0	2(A)	5(B)	1(A)	7(D)	8(E)	8(E)	10(G)	G	A,B,C,D,E,F,G
8	0	2(A)	5(B)	1(A)	7(D)	8(E)	8(E)	10(G)	H	A,B,C,D,E,F,G,H

The resulting shortest path routes are:

Destination	Cost	Path from A
A	0	A
B	2	A → B
C	5	A → B → C
D	1	A → D
E	7	A → D → E
F	8	A → D → E → F
G	8	A → D → E → G
H	10	A → D → E → G → H

b) The Open Shortest Path First (OSPF) routing protocol uses Dijkstra's algorithm.

c) An interior routing protocol is used within an administrative domain. RIP and OSPF are examples of interior routing protocols. Organizations administrating a network domain are free to choose any interior routing protocol that suits their needs.

An exterior routing protocol is used to find paths between administrative domains. BGP is the exterior routing protocol used in the current Internet. Organizations who want to connect their networks to the Internet must speak BGP in order to become reachable.

Problem F.3: *domain name system* (5+2+3+5=15 points)

a) Explain briefly the purpose of `A`, `AAAA`, `SOA`, `CNAME` and `PTR` resource records.

b) What are recursive queries? Which information must be known by a DNS server to support recursive queries?

c) The DNS protocol can run over UDP and TCP. Explain why both transport protocols are useful. How does a DNS implementation select the transport protocol to use?

d) The `MX` resource record is used to locate email servers providing email services for a given DNS domain. What would be required to generalize the `MX` record so that arbitrary servers providing services in a DNS domain can be found?

Assume that a service is identified by a service name and a transport protocol name (to allow for services utilizing multiple transport protocols) and that you can introduce new resource records but not change the DNS protocol itself.

Solution:

a) Short description of the resource record types:

`A`	Maps a DNS name to an IPv4 address
`AAAA`	Maps a DNS name to an IPv6 address
`CNAME`	Maps a DNS name to name (canonical name)
`PTR`	Pointer to another part of the name space (reverse mappings)
`SOA`	Start and parameters of a DNS zone (start of zone of authority)

b) DNS has two modes of operation: A DNS client can either directly search through a chain of DNS servers to find the answer to a question or the client can ask its DNS server to recursively search for the answer. In later case, it is required that the server has knowledge about the IP addresses of the root name servers in order to resolve arbitrary queries.

c) UDP is very lightweight transport protocol since it does not require a connection setup phase. UDP has however a packet size constraint, especially if fragmentation is to be avoided. For larger DNS transfers, such as zone transfers, it is thus desirable to use TCP as a transport protocol. The DNS protocol automatically switches to TCP if a the TC (truncated) bit is set in a response packet.

d) A generalization of the `MX` record actually exists in the form of so called `SRV` records (RFC 2782). To lookup a service identified by a service name and a transport protocol name, it is necessary to encode this information into the DNS name. The resource record itself should contain at least a priority and a name (or IP address). The `SRV` record actually contains in addition a weigth parameter and a port number:

```
; Name                          Priority Weight Port   Target
_sip._tcp.example.com.    IN SRV  0        1      5060   server1.example.com
_sip._udp.example.com.    IN SRV  0        2      5060   server2.example.com
```

Problem F.4: *formal notations* (10+10+5=25 points)

Imagine an application protocol for managing rooms in a building. Every room is represented by a room record. Room records must contain a number used to identify the room and the type of the room (office, storage, lab, or other). A room can have an optional name (e.g. *Network Lab*) and an optional contact person (e.g. *J. Schoenwaelder*). Finally, every room record may list zero or more associated user names. All names are ASCII character strings which may include white space characters. Sample textual records might look as follows:

```
room {
  number = 87,
  type = office,
  user = "Juergen Schoenwaelder"
}
```

```
room {
  number = 86,
  type = lab,
  name = "Network Lab",
  contact = "Juergen Schoenwaelder",
}
```

You are welcome to derive from this specific format as long as the requirements spelled out above are satisfied.

a) Write suitable ABNF rules for the representation of room records in a textual format.

b) Write an ASN.1 module for the representation of room records.

c) What are the advantages/disadvantages of the two formalisms? Which one would you prefer for the project outlined above?

Solution:

a)

```
room = "room" WSP "{" LWSP body LWSP "}" CRLF

body = room-number sep room-type [sep room-contact] *room-user

room-number = "number" equal number

room-type = "type" equal type

room-contact = "contact" equal string

room-user = sep "user" equal string

equal = WSP "=" WSP

sep = LWSP "," LWSP

type = "office" / "storage" / "lab" / "other"

string = DQUOTE *VCHAR DQUOTE

number = 1*DIGIT
```

```
; the definitions below are taken from RFC 2234

LWSP           =  *(WSP / CRLF WSP)

WSP            =  SP / HTAB

SP             =  %x20

CRLF           =  CR LF

CR             =  %x0D

LF             =  %x0A

HTAB           =  %x09

DIGIT          =  %x30-39

DQUOTE         =  %x22

VCHAR          =  %x21-7E
```

b)
```
ROOMS DEFINITION IMPLICIT TAGS ::= BEGIN

Room ::= SEQUENCE {
    number  INTEGER,
    type    RoomType,
    contact String,
    users   UserList
}

RoomType ::= INTEGER { other(0), office(1), storage(2), lab(3) }

UserList ::= SEQUENCE OF User

User ::= SEQUENCE {
    name String
}

String ::= OCTET STRING         -- must define some character set here --

END
```

c) ABNF is well suited for describing textual formats which can be read and edited by utilizing many existing tools. ABNF definitions usually end up being human readable, which simplifies debugging and deployment. The downside of ABNF is that textual representations are often less compact.

ASN.1 is closer to programming languages and allows to generate implementation data structures from ASN.1 definitions. The default ASN.1 encodings are binary and hence often more space efficient than textual formats. The downside is that binary representations are much harder to read and debug.

For the given project, a textual representation clearly makes more sense since there is little to gain using binary representation due to the strings and since it is necessary to frequently edit room descriptions and adapt software to it. (The example actually stems from a real project which however used XML and Relax NG.)

Problem F.5: *mail transfer and access protocols* (5+5+5+5=20 points)

Answer the following questions related to the Internet electronic mail (email) system. Provide concise answers and write clearly.

a) The Internet email system consists of mail delivery agents (MDAs), mail user agents (MUAs), and mail transfer agents (MTAs). Suppose an IUB student is sending an email to another IUB student using the official IUB email addresses. In which sequence will MDAs, MTAs and MUAs be involved in the process? Which protocols play a role in the process?

b) Explain the difference between an envelop address and a header address.

c) Explain the main principle of base64 and quoted-printable encoding. Why are these encodings needed at all?

d) SMTP uses 3-digit structured reply codes and this concept has been carried over to many other application layer protocols. What is the benefit of using SMTP-like 3-digit structured reply codes?

Solution:

a) The email message starts its journey by a MUA which will deliver the message to the first MTA. The MTA stores and forwards the email message to other MTAs using SMTP until the message reaches the destination MTA which will transfer the message via SMTP or LMTP to the MDA for delivery to the receivers mail box. The receiver MUA will fetch the message from the MDA's mail box using POP or IMAP.

b) The envelop address is the address used by the SMTP protocol to forward messages. Like in ordinary paper mail systems, there is not necessarily a relationship between the envelop address and the header address which is part of the message itself.

c) Base64 encodes three bytes (that is 24 bits) by means of four characters taken from a 6-bit character set. Quoted printable encoding is based on the idea to encode only bytes that are not in the 7-bit US-ASCII alphabet by means of escape sequences which carry hexadecimal representations of the original byte values. Encodings like base64 and quoted-printable are needed since the original SMTP definition is restricted to the 7-bit US-ASCII alphabet

d) 3-digit structured reply codes have the benefit that implementations can handle unknown error codes as long as they belong to the same class of error code. In addition, 3-digit structured reply codes are typically used with ABNF specified protocol where clear-text error messages are shipped with the reply code.

Problem F.6: *remote procedure calls* (2+2+2+2+2=10 points)

Indicate which of the following statements are correct or incorrect by marking the appropriate boxes. For every correctly marked box, you will earn two points. For every incorrectly marked box, you will loose one point. Statements which are not marked or which are marked as true and false will be ignored. The minimum number of points you can achieve is zero.

true	false	
☐	☐	RPC stub procedures are used to achieve transparency by hiding all communication details.
☐	☐	At-most-once RPC semantics require that servers maintain state about recently executed RPC calls.
☐	☐	The ONC RPC system uses a triple (P,V,F) to identify a procedure where P is a program name, V is a version number and F is the procedure name.
☐	☐	RPC binding can be performed by registering a name identifying a server in the DNS name server.
☐	☐	Marshalling is the process of transferring data structures used in RPCs from one address space to another and includes data serialization and data representation conversion.

Solution:

true	false	
☒	☐	RPC stub procedures are used to achieve transparency by hiding all communication details.
☒	☐	At-most-once RPC semantics require that servers maintain state about recently executed RPC calls.
☐	☒	The ONC RPC system uses a triple (P,V,F) to identify a procedure where P is a program name, V is a version number and F is the procedure name.
☐	☒	RPC binding can be performed by registering a name identifying a server in the DNS name server.
☒	☐	Marshalling is the process of transferring data structures used in RPCs from one address space to another and includes data serialization and data representation conversion.

Good luck and have a nice and relaxing christmas break!

/js

ABNF Core Definitions (RFC 2234)

```
ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z

BIT            =  "0" / "1"

CHAR           =  %x01-7F
                       ; any 7-bit US-ASCII character,
                          excluding NUL

CR             =  %x0D
                       ; carriage return

CRLF           =  CR LF
                       ; Internet standard newline

CTL            =  %x00-1F / %x7F
                       ; controls

DIGIT          =  %x30-39
                       ; 0-9

DQUOTE         =  %x22
                       ; " (Double Quote)

HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

HTAB           =  %x09
                       ; horizontal tab

LF             =  %x0A
                       ; linefeed

LWSP           =  *(WSP / CRLF WSP)
                       ; linear white space (past newline)

OCTET          =  %x00-FF
                       ; 8 bits of data

SP             =  %x20
                       ; space

VCHAR          =  %x21-7E
                       ; visible (printing) characters

WSP            =  SP / HTAB
                       ; white space
```