Design Constraints and Implementation of a PMR Smart Card Based Software

A. Gamache, P. Ardouin, P. Durant and R. Zaïane[1]
Département d'informatique, Université Laval
Conference ACT/Canada, Toronto November 91

Summary

The design of a Portable Medical Record (PMR) smart card based system introduces a new passive component in the system delivery of medical services to end users. This new carrier of medical data, the PMR, aimed at sharing health information, should also comply with every bit of medical ethics and of government regulations regarding confidentiality and access to medical information. The Québec PMR system is being designed in order to reinforce these constraints and to be independent from any major technological choice. This approach involves several technical challenges. The first is the impact of accumulating data in various controlled access zones while managing memory space to delay the unavoidable saturation. We propose certain strategies for memory management. The next problem is to immune, as much as possible, the application software from structural and technological choices. The proposed system has a three-level architecture: Application software, Interface and Driver. Each layer has its own specific functions and components; the information flows among them according to a predetermined communication protocol. This layered architecture allows for the encapsulation of every software module and allows for the implementation of a multi-user client/server approach. The smart card based system to be tested will give us better clues on behavior of health professional and patients towards a new computerized tool that some may regard as an intruder in the health care professionnal-patient relationship.

---

1. Authors' address: Département d'informatique, Faculté des Sciences et de Génie, Université Laval, Québec, Québec, Canada, G1K 7P4 fax: (418) 656-2324, e-mail: gamache@vm1.ulaval.ca
Published in the Proceedings of ACT/Canada Conference-91, Toronto, Canada

Outline:

Part 1:

- Design philosophy and constraints of PMR
- Implementation challenges
- Data structuring and mapping

Part 2:

- Memory management
- Simulation: some results on saturation

Part 3:

- Software : - General Interface application
  - Driver
- Standard set of instructions and
- Application program structure

Part 4:

- Compression techniques and
- Preliminary results with medical records
- Conclusions

Part 1

Design Philosophy of the QPMR (Québec PMR) [1]

1. Health Care Professionals (HCP) as providers of medical services and representatives of their professional body should be directly involved in the design of the QPMR:

main user groups: **medical doctors**
pharmacists
nursing personnel

2. The proposed system for managing the QPMR, should be tested and monitored extensively within a controlled pilot project prior to its implementation on a large scale basis: population of 6 million;

3. Health Care Professionals, end users and external peers should participate in assessment of this pilot;

4. The pilot project should provide the necessary data to evaluate the feasibility of generalizing the system throughout Québec;

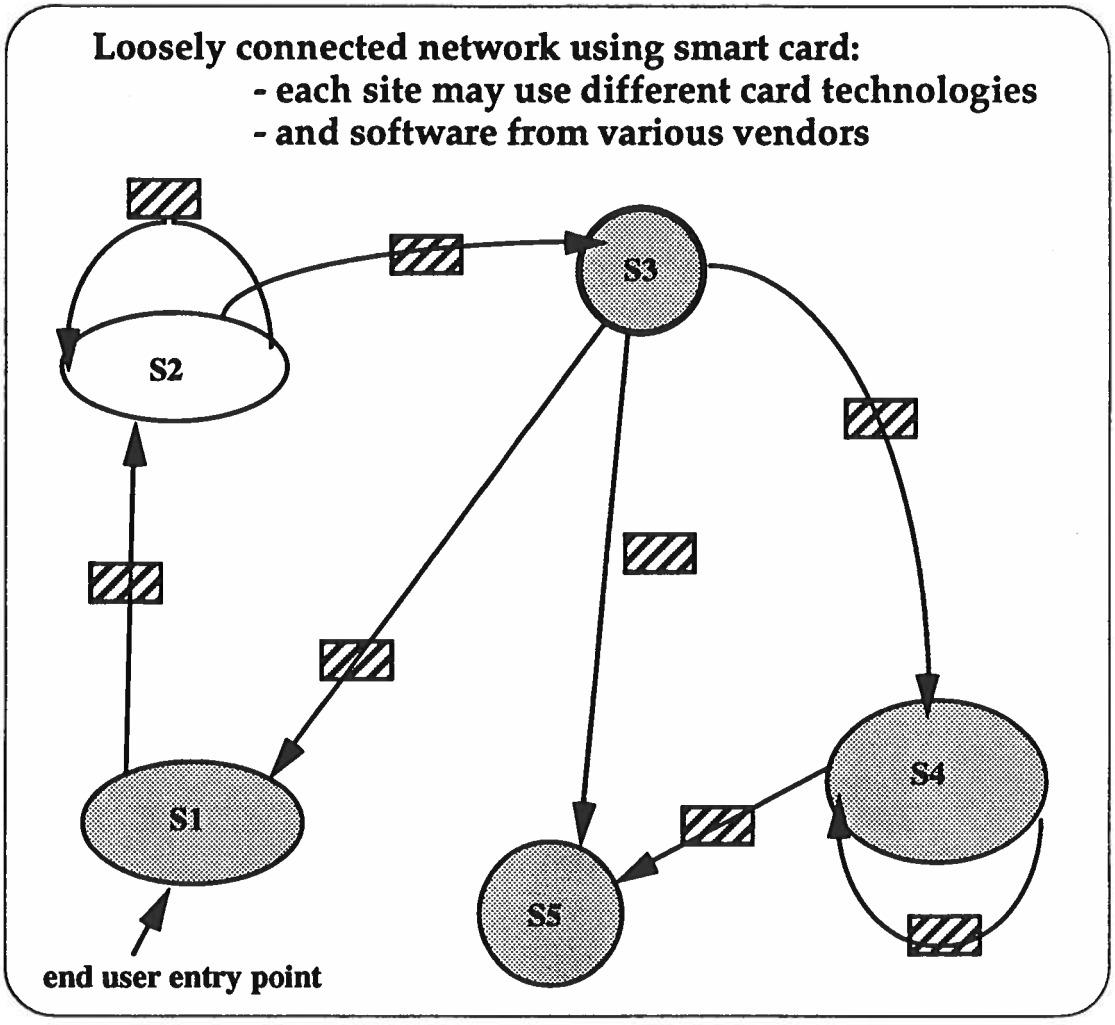
5. The software system should be highly reliable and able to cope with expected rapid technological obsolescence in this field of Integrated Circuit (IC) [2];

6. Each HCP should be accountable for every bit of information stored on the QPMR: data are stamped (with signature and date) and persistent.

Main goal (from an information system perspective)

Improving the quality of medical and pharmaceutical services based on sharing reliable and update informations in a loosely connected network of care providers.

The information bearing media is the smart card; the point of services are mostly client-clustered and links are highly secured. On a long term basis, the node is expected to be heterogeneous in terms of hardware and software.



Québec Portable Medical Record (QPMR)

System's constraints [3]

1. The QPMR should reflect the needs of HCP as required by their decision making processes;

2. The QPMR should trace every transaction that both the patient and the HCP consent to write on;

3. The QPMR should be highly secured in order to guaranty the confidentiality of data;

4. No central repository of medical data should evolve from the use of QPMR; no "easy" algorithm could be used to reconstruct the population entire database (a very slow heuristic could exist);

5. The QPMR should convey information for authorized HCP on the basis of predetermined access rights; under supervision, the data should be also accessible to the bearer;

6. Open system design: new application and new data elements may be added during the card life cycle without impairing previously defined operational applications;

Some design and implementation challenges

1. Modeling medical unstructured data:

no universal minimal data set recognized;
identification and relevance;
content coding for consistency and space saving;

2. Full and strict implementation of all user access rights to medical data;

3. Highly effective User Application Interface designed and tested for each group of HCP;

4. Highly reliable system software: fault tolerance and transparency to technological changes is looked for with respect to smart card technologies;

Data structuring [4]

Logical PMR (Application Level)

The QPMR is structured with logical zones; no reference to physical memory. Every application has a specific view of the logical PMR.

Logical zone

Every data to be entered by HCP belongs to a logical zone: a logical zone is a grouping of semantically related data elements without any reference to their access or protection rights;

---> we may have approximatively 45 logical zones in the QPMR

Logical transaction

All application view the PMR as a logical structure;

Every data to be written in a logical zone is structured within a logical transaction;

$$\text{Logical Transaction} := \{ \text{l.z.}, f_1/v_1, f_2/v_2, \ldots \textit{signature,date} \}$$

where f_1/v_1 **stands for field_name and atomic typed value,**
l.z. = logical zone number

Physical zone (System Level)

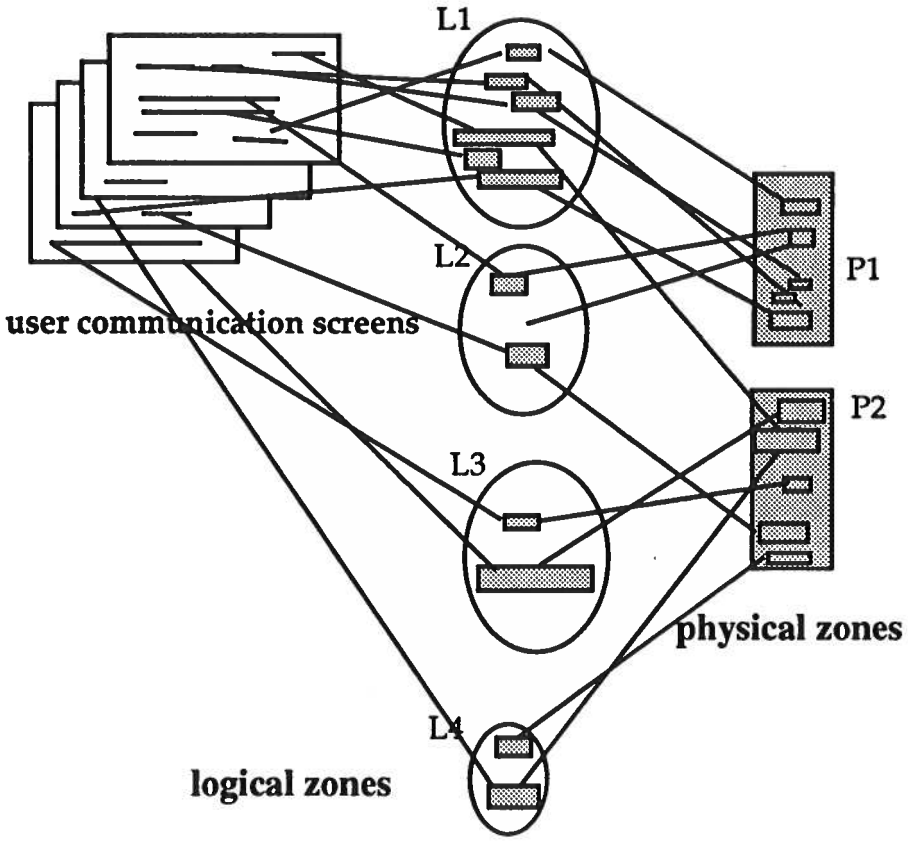
A physical zone is made of data elements (from logical zones) sharing the same access rights for all user groups;

---> we could have approximatively 15 physical zones in the QPMR

A physical zone is implemented through allocation of physical blocks, as they may be configured in a given smart card technology.

Mapping between logical and physical zones

This mapping is implemented at the system level and a user application is not concerned with physical zones.



Critical technical issues:

- The unavoidable state of memory saturation: later, the better. . . [5]

As we accumulate (strong system's constraint) medical data on the PMR without having any on-site deletion mechanism, a saturation will someday occurs.

Strategies:

- Postponing:

using memory management and effective compression techniques;

- Expanding the PMR's memory:

Resolution saturation of memory by a new card by content abstracting and reissuing of a continuation card.

what is the probability of this event?

who is going to reissue the care card?

what is the frequency of this operation on a time scale?

- Non-stop service to users in spite of technological upgrades or breakthroughs

In a large scale implementation of the QPMR, the system will have its own dynamic and evolution over time ; it should cope with new information needs and new card technologies without any service interruption;

Part 2

Memory Allocation policies [6]

- Fixed length static allocation (FLSA) :

- entire memory divided into blocks of fixed and identical size;
- all are allocated at issuing time.

2) Variable length static allocation (VLSA) :

- entire memory divided into blocks of fixed size;
- block size differs between zones;
- all are allocated at issuing time.

3) Fixed length dynamic allocation (FLDA) :

- entire memory divided into blocks of fixed and identical size;
- at issuing time, one block is allocated to each zone;
- further block dynamically allocated when needed.

4) Variable Length Dynamic Allocation (VLDA) :

- quite similar to FLDA;
- the size of each block may vary from zone to zone based on expected usage.

5) Dynamic Length Dynamic Allocation (DLDA)

- quite similar to VLDA;
- block size varies within a zone.

6) Dynamic allocation with zone increase (DAZI):

- a zone is allocated at issuing time with minimal space;
- each time a transaction is written in a zone, only the space needed will be contiguously allocated.

Saturation of a data zone

A potential saturation (PS) occurs when a zone cannot receive more data;

a final saturation (FS) occurs when a PS cannot be resolved by a given policy.

Saturation resolution policies

1) Erasing information within the same zone (ESZ)
---> may extend the card's life;

2) Erasing information in other zones (EOZ)
---> may give further extension to storage capacity;

3) Erasing information with zone shrinking (EZS)
---> may extend the card's life;

4) Disallocation of free space in any zone with reallocation (DFS)

5) Do nothing ---> final saturation

Simulation model

- An optimal design would maximize card life cycle.

- Life cycle depends mainly upon card usage.

- Card usage is characterized by number, type and size of logical transactions;

Model [5]

The model has 12 parameters and generates transactions (medical, pharmaceutical and emergency) according to a Poisson's law.

Some parameters of the model:

memory space on the card

size of a zone

size of a transaction: min and max bits

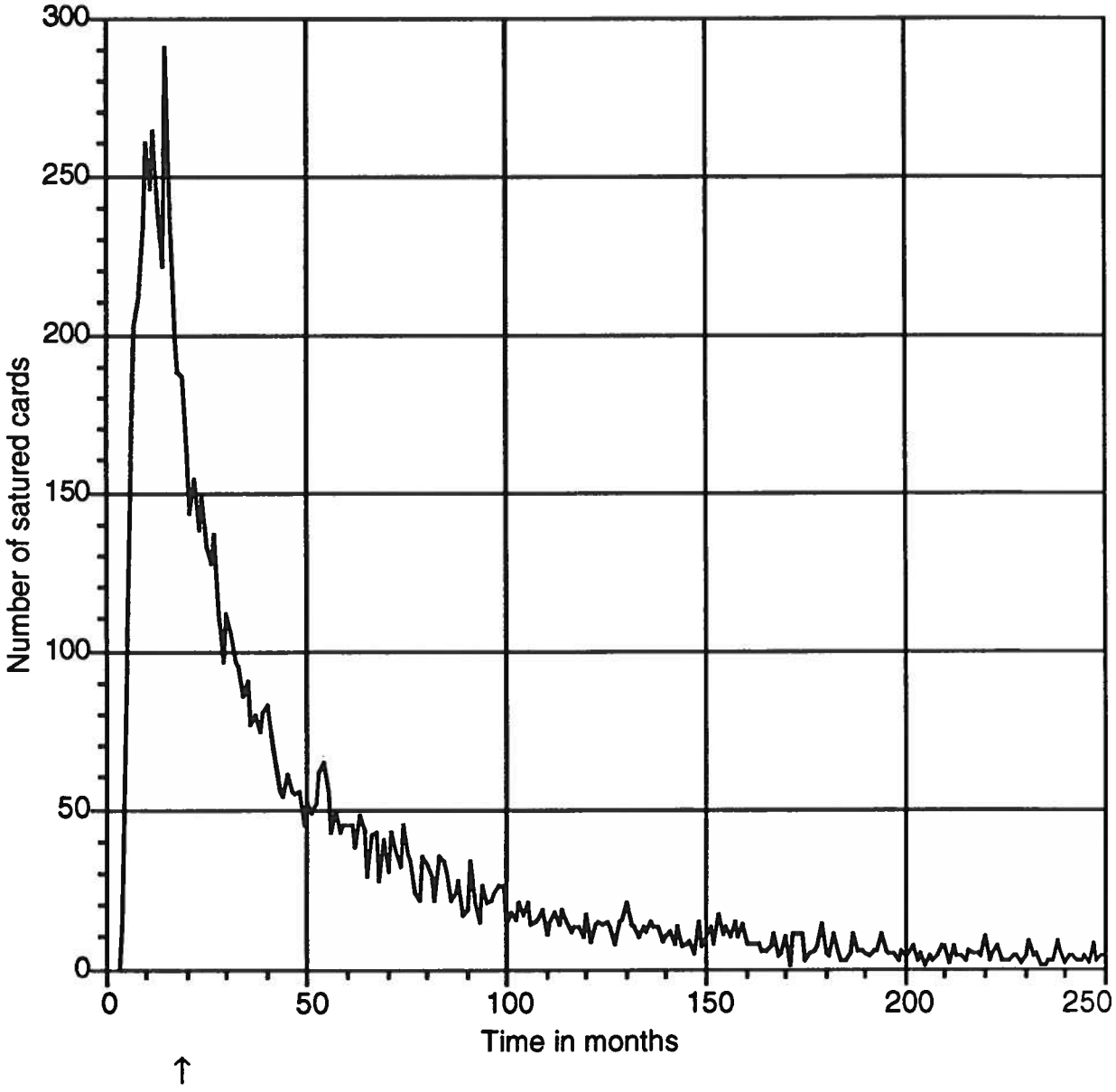
size of block descriptor

probability on the nature of a transaction: md, pharmacist, ...

memory management policy

saturation resolution policy

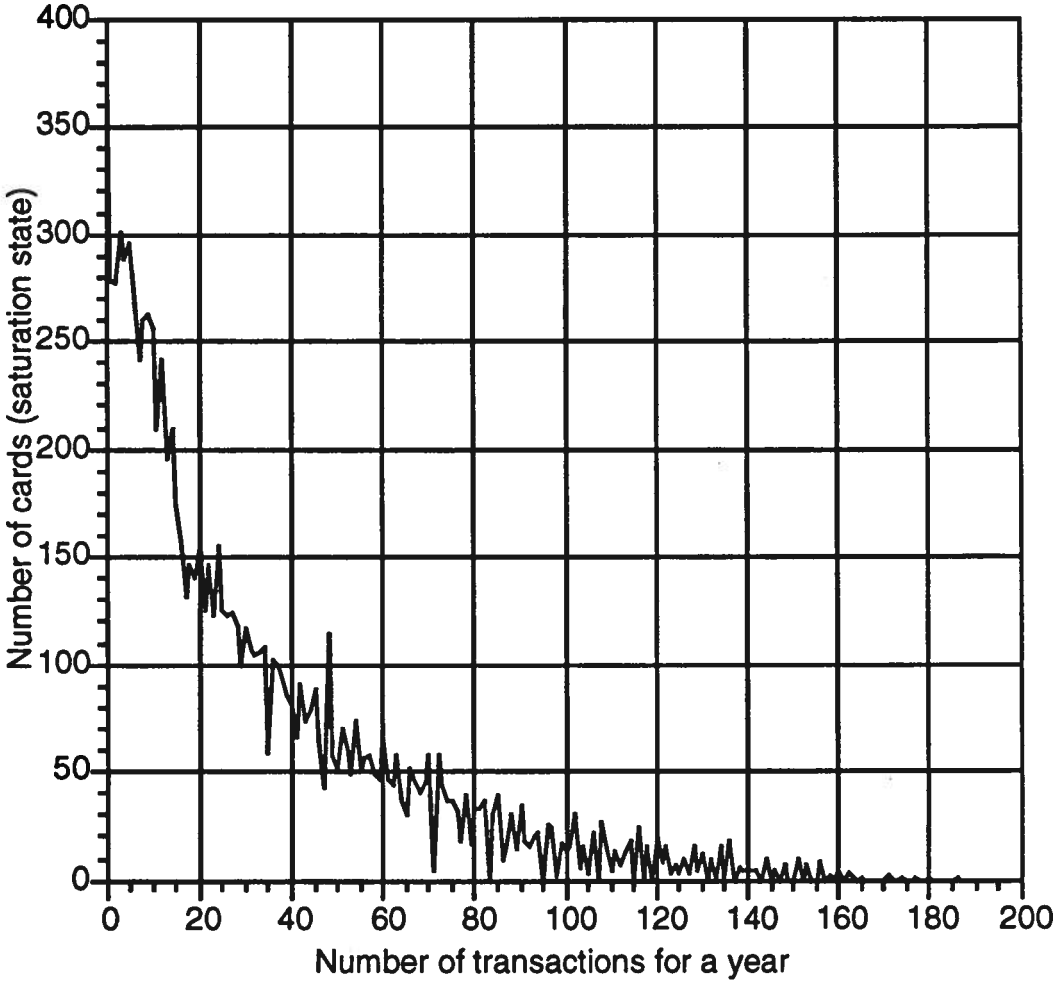
Saturation vs time



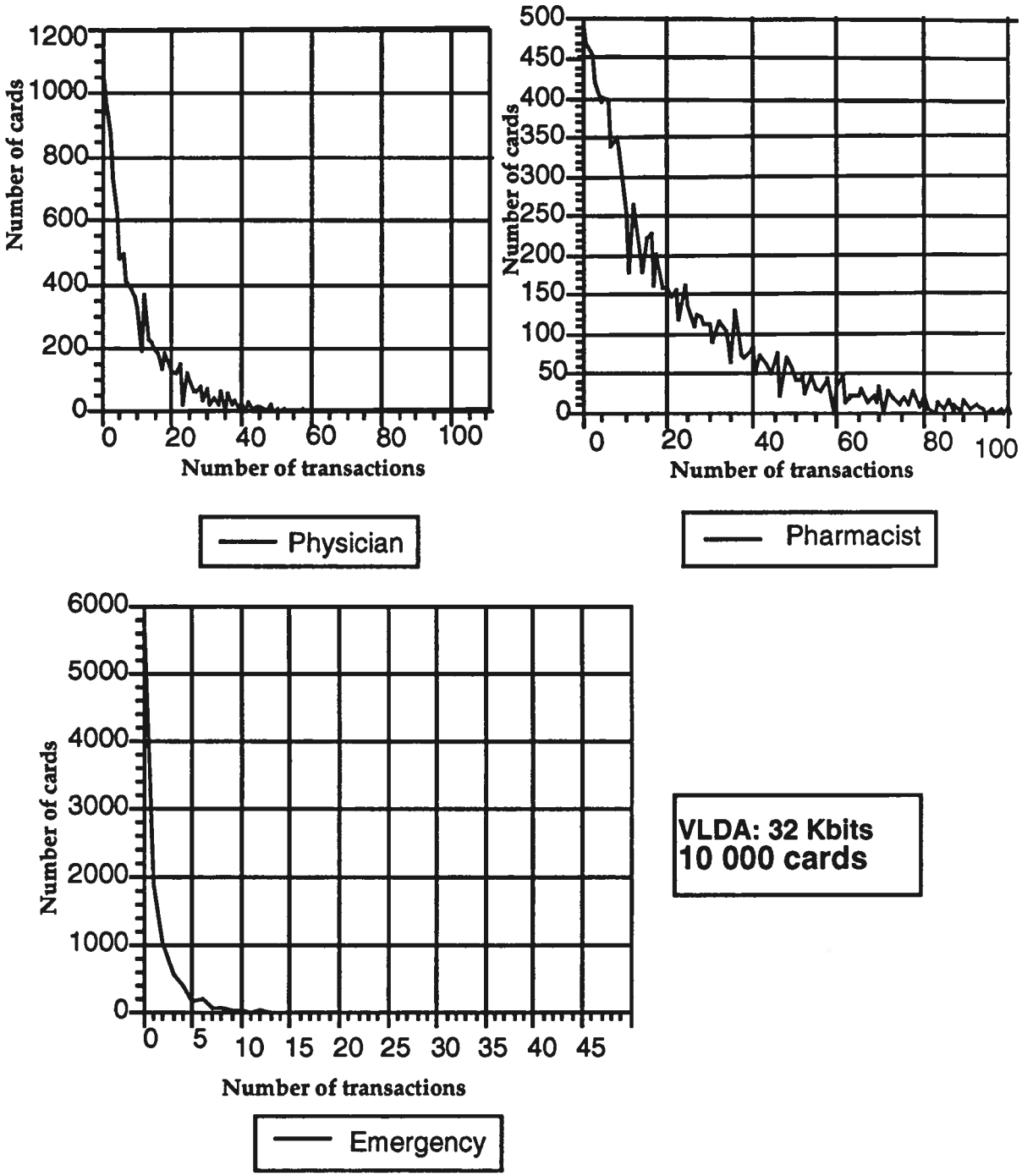
↑

VLDA: 32 Kbits
10 000 cards

Distribution of cards over the number of transactions



VLDA : 32 kbits
10 000 cards


1200
1000
800
600
400
200
0
Number of cards
0 20 40 60 80 100
Number of transactions
Physician
500
450
400
350
300
250
200
150
100
50
0
Number of cards
0 20 40 60 80 100
Number of transactions
Pharmacist
6000
5000
4000
3000
2000
1000
0
Number of cards
0 5 10 15 20 25 30 35 40 45
Number of transactions
Emergency
VLDA: 32 Kbits
10 000 cards

Typical Results from Simulation

Effect of memory allocation and saturation resolution policies

Memory allocation policy	Saturation resolution policy	Life cycle (months)	
		For a 32 Kb card	For a 64 Kb card
VLSA	None	11,7 m.	30,7 m.
DAZI	None	15,8 m.	36,4 m.
VLDA	None	19,2 m.	39,0 m.
VLSA	ESZ	19,7 m.	39,5 m.
VLDA	ESZ	25,4 m.	49,7 m.
DAZI	ESH	30,2 m.	70,1 m.

Number of transactions in different zones at card saturation

Memory allocation policy	Number of transactions when saturation occurs (32 Kb card and no saturation resulution policy)		
	Emergency	Medication	Medical
VLSA	1,8 tr.	24,0 tr.	10,5 tr.
DAZI	2,1 tr.	33,1 tr.	15,4 tr.
VLDA	2,7 tr.	40,3 tr.	17,7 tr.

Effect of memory size on proportion of cards reaching saturation

Memory size	Proportion of cards reaching saturation within 18 months
32 K	30 %
64 K	8 %
128 K	< 0.1 %

Some comments on results:

- The simulation indicates that a problematic situation exists and the proposed system should take somewhat into consideration these simulation results;

- Better results would be available soon based on real data obtained from the RAMQ (1989-1991);

- The life cycle seems to be long enough to support the pilot project;

- The final saturation could be resolved by:

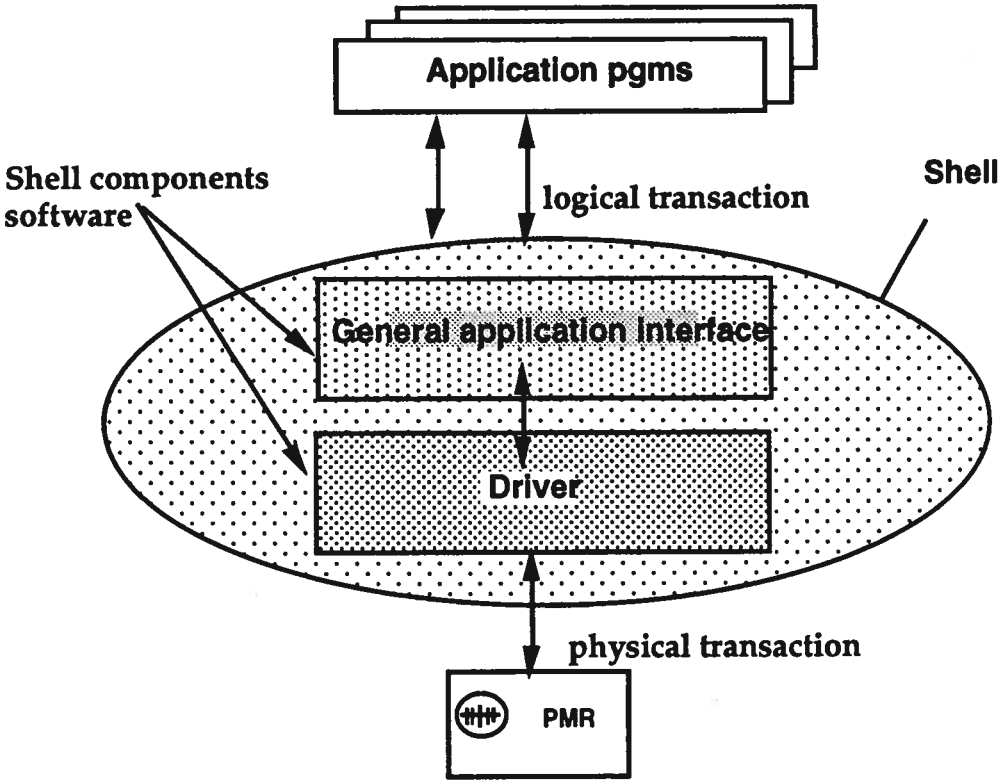
- an adequate memory allocation policy and

- centrally issuing a continuation card.

Part 3:

General architecture of the software system [4]

An application program writes or reads on any card via the shell.



Shell functions:

- **Channeling all communications towards the card;**
- **Reinforcing the atomicity of a logical and physical transaction;**
- **Managing the view of each application with respect to the PMR;**
- **Controling the authorization procedure: PMR and HCP ID card;**
- **Implementing the mapping between logical and physical transactions;**

- Performing physical read and write of each physical transaction;

- Gathering statistical data to back the system evalation work.

Communication protocol between application and card

With a multi-tasking system, an application sends and receives a message to and from another process, the shell, in order to have some work done on the card:

open/ close a card
write/ read a logical transaction
check the card status
etc

Standard set of card instructions

We have implemented a minimum set of 12 instructions (implemented as C functions) to access and use the smart card. These are the only instructions to be used by any application using the QPMR. They set up the message to be sent to the shell and check the return code. This approach is general by nature and can integrate various computer environment.

General syntax of a message:

action_ID [card_type] [actor], [logical zone_ID] [data]

action_ID: opening, closing, eject, reading, writing, card_availability, card status.

card_type: user and HCP card

actor: shell, medical doctor, pharmacist, ...

logical zone_ID: ID number for a logical zone

data: string of data

General syntax of an acknowledge message:

action_ID [card_type] [actor], [logical zone_ID] [data] [return code]

The return code is managed by the application program to reinforce the atomicity of a logical transaction. A logical transaction is either fully executed or nothing is done.

Standard instructions set :

- **CQ_OuvrirCarteActeur (&Acteur) : to open an HCP card and to get a token from the shell**

- **CQ_FermerCarteActeur(Acteur): to close an HCP card**

- **CQ_LireCarteActeur(Acteur, Zone_logique, Pointeur_Data)**

- **CQ_EcrireCarteActeur(Acteur, Zone_logique, Pointeur_Data)**

- **CQ_EjecterCarteActeur() :to eject the HCP card**

When an Actor's card (HCP card) is signed-on, a token is registered in an internal table and the actor's card is kept opened even if the card is withdrawn from the reader.

Instructions to get access to PMR's data:

- CQ_OuvrirCartePorteur(Acteur)

- CQ_FermerCartePorteur()

-CQ_LireCartePorteur(Acteur, Zone_logique, Pointeur_Data)

- CQ_EcrireCartePorteur (Acteur, Zone_logique, Pointeur_Data)

Special instruction to delete a record on the card:

-CQ_AnnulerRecordPorteur(Acteur, Zone_logique, Record)

the record is marked as being deleted but still on the card and may be accessible to a special actor.

other instructions

-CQ_StatutCarte(Buffer_statut): to get the free space still available on a PMR and the actor class

-CQ_CartePresente() : to check if a PMR is signed-on in the reader

Effects on the development of applications:

- the developer is no longer concerned with a particular technology;

- the atomicity of a logical transaction is implemented by the shell;

- any change at the hardware level is managed by the driver and has no effect on the application;

General Structure of an application program using the standard set of instructions

```
BEGIN PGM
Open_Actor_session     = False
Patient_card_present   = False
Actor_card_present     = False

If NOT  CQ_EtablirCommunication() Then EXIT PGM
            /* process interconnexion */
if  (an actor wants the status of the card) then
      CQ_StatutCarte(&Variable_statut)
end_if
if (an actor wants to open a work session)  then
      If  CQ_OuvrirCarteActeur(&Jeton) = 1 then
                  /*habilitation process */
            Open_actor_session  = True
            Actor_card_present  = True
      endif
end_if

While (Open_actor_session) Do
            if NOT Actor_card_present  then
                  if CQ_OuvrirCarteActeur(&Jeton) = 1then
                  /* check actor's token and actor's habilitation and turn  on power*/
                        Actor_card_present = True
                  end_if
            end_if

            While (Actor_card_present) Do
                        if CQ_LireCartePorteur(Jeton,ZoneLogique, &Data) = 2 then
                              Actor_card_present  = False
                              EXIT While
                        end_if
                        if CQ_EcrireCartePorteur(Jeton,ZoneLogique,Data) = 2 then
                              Actor_card_present  = False
                             EXIT While
                        end_if

                        if CQ_StatutCarte(&Variable_statut) = 2 then
                              Actor_card_present  = False
                             EXIT While
```

```
            if CQ_CartePresente() = 0 then
                  Actor_card_present  = False
                  EXIT While

 if (the actor wants to pull out his card from the reader) then
                  CQ_EjecterCarteActeur()
                        /* actor session is kept opened */
                  Actor_card_present = False
            end_if
END_While

if (the actor wants to work on patient card) then
      if CQ_OuvrirCartePorteur(Jeton) = 1then
            Patient_card_present = True
      end_if
end_if

While (Patient_card_present) Do

            if CQ_LireCartePorteur(Jeton,ZoneLogique,&Data) = 2 then
                  Patient_card_present  = False
                     EXIT While
            end_if
            if CQ_EcrireCartePorteur(Jeton,ZoneLogique,Data) = 2 then
                  Patient_card_present  = False
                  EXIT While
            end_if
            if CQ_AnnulerRecordPorteur(Jeton,ZoneLogique,Record) = 2
                  then
                  Patient_card_present  = False
                  EXIT While

            if CQ_StatutCarte(&Variable_statut) = 2 then
                  Patient_card_present  = False
                  EXIT While

            if CQ_CartePresente() = 0 then
                  Patient_card_present  = False
                  EXIT While
```

```
            if (the actor wants to close the patient card) then
                    return_code = CQ_FermerCartePorteur()

                    if return_code = 1 OR return_code = 2 then
                     Patient_card_present  = False
                    end_if

                end_if
        END

    if (the actor wants to close his work session) then
            if CQ_FermerCarteActeur(Jeton) = 1 then EXIT PGM
        end_if

END_While

END PGM
```

return Code:	**0 = unable to execute** **1 = success** **2 = card is pulled out from the reader** **3 = protocol communition problem** **4 = access rigth violation** **5 = full memory card** **6 = execution stopped by user**

General Application Interface

Some functions to be executed, following a message sent by a standard instruction and received by the shell:

- Receives a message from the application;

- Sign-on the actor and build various internal tables;

- Implements the atomicity at the logical transaction level;

- Checks the requesting application's access rights;

- Opens the PMR at the logical level;

- Implements the mapping from logical to physical transactions;

- Manages special dialogs with the user during sign-on procedures;

- Generates a set of physical transactions from one logical transaction;

- Manages the logical deletion of a record under specialized control;

- Detects the saturation in a physical zone;

- Control the resolution of a potential saturation;

Driver

Main functions:

- Recognizes the card technology used on a site; our generalized approach is able to support heterogeneous nodes possibly using different types of cards, refered to, from now on, as technologies;

- Implements the atomicity of a physical transaction;

- Does the physical reading and writing of data in the physical zones;

- Does the physical opening and closing of the card;

- Generates a set of specific physical orders to be executed;

- Compresses data to be stored in a physical zone;

Part 4

Compression techniques used in the PMR [7]

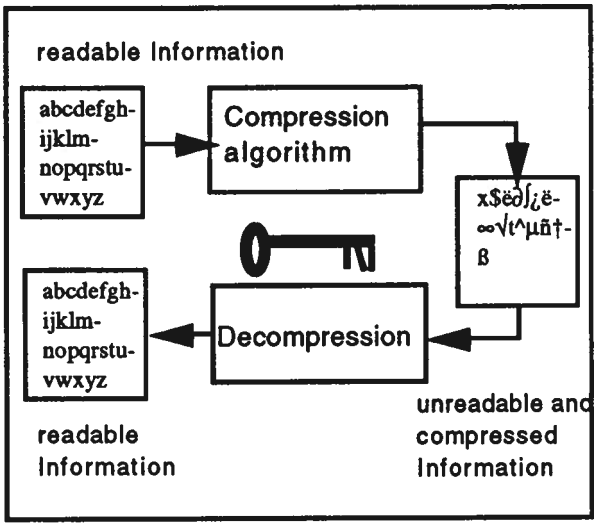
Some advantages in using compression techniques:

- To save memory space: good compression may further delay the saturation state in a physical zone;

---> many algorithms available which give various compression factors depending on input information

- To decrease the communication time on serial line: may be useful to get on-line access to laboratory results;

- to improve security by reducing redundancy in the information;



Text compression reversible algorithms

Many of them are available [8]; our first choice is composed of the following coding classes (assuming a noiseless channel):

- **Huffman coding**
- **Lempel, Ziv, Welch algorithm**
- **Arithmetic coding**
- **LZHUF algorithm**

Text files used

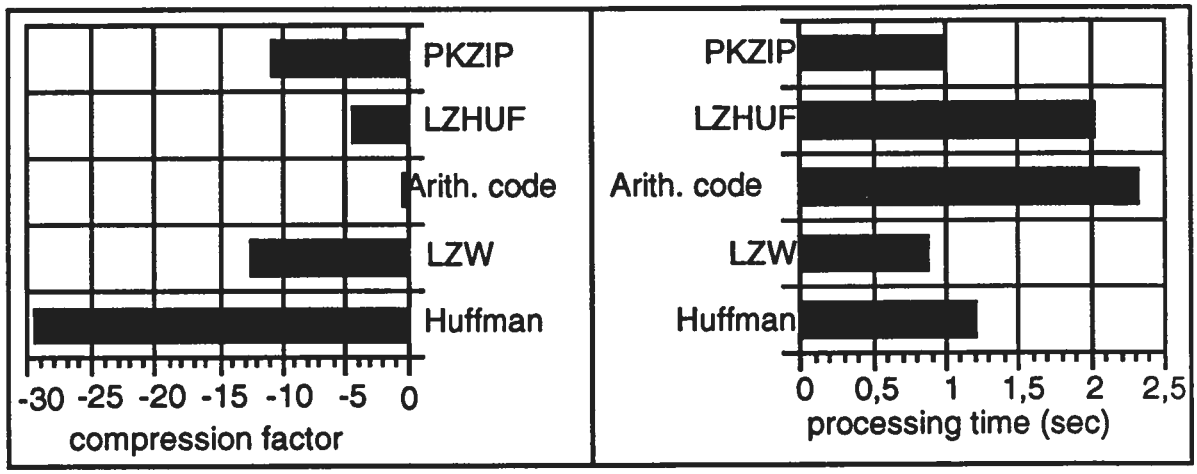
- **random binary digits file**
- **pre-coded medical file with minimal redundancy**
- **pre-coded medical file with redundancy**
- **free-text medical file**

Compression factor = (Li - Lc) * 100 / Li

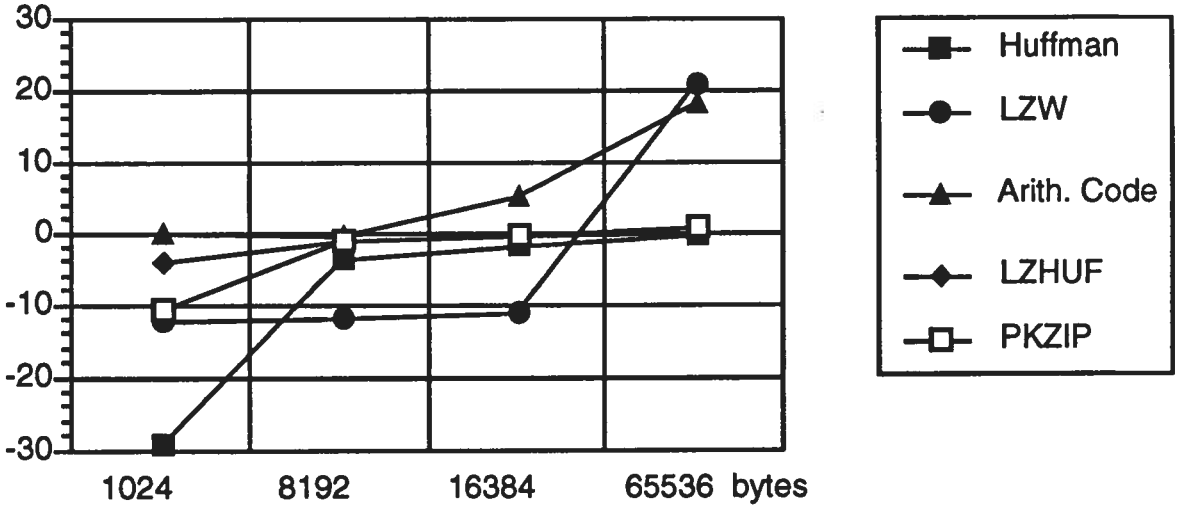
Li = length of the plain text Lc = length of compressed text

A positive CF means saving space on memory.

File of random binary digits: L= 1024 bytes



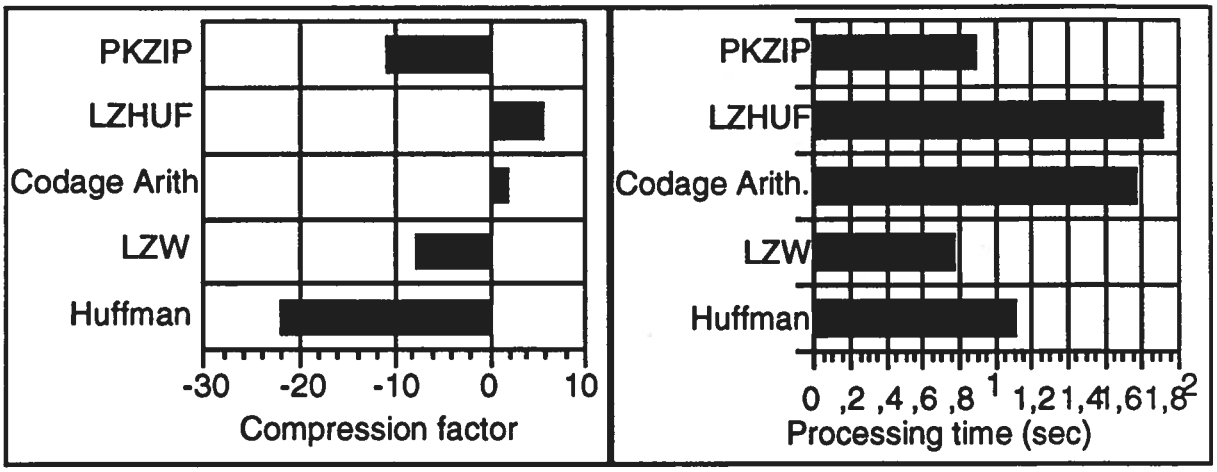
(a) Buffer of 1024 bytes (b) Processing time

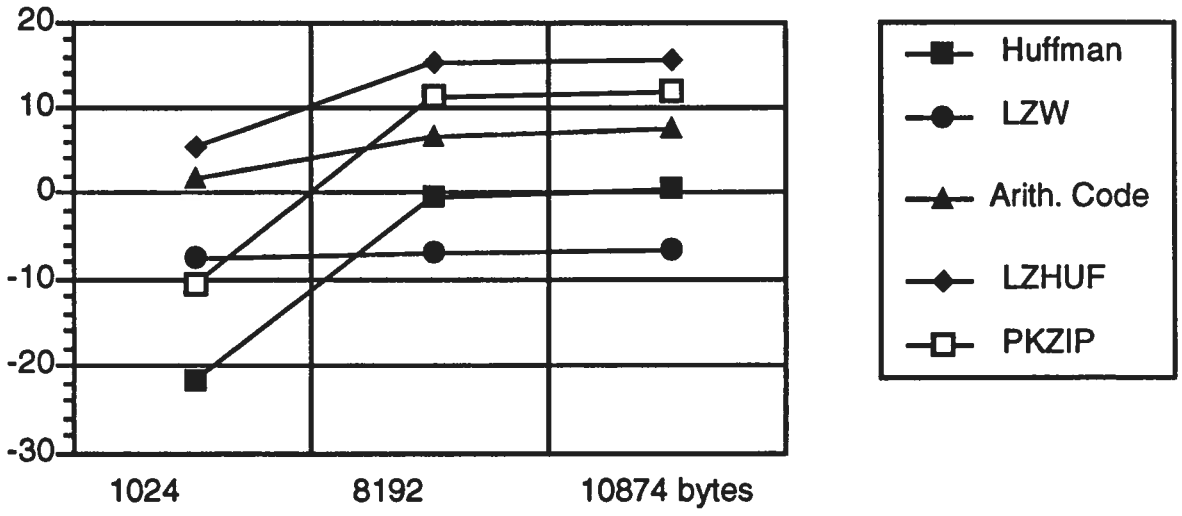


(c)Compression factor versus size of plain text

Space is increased by Huffman algorithm: 30% of space for a processing time of 1.5 sec.

File of pre-coded medical records (L= 1024 bytes)

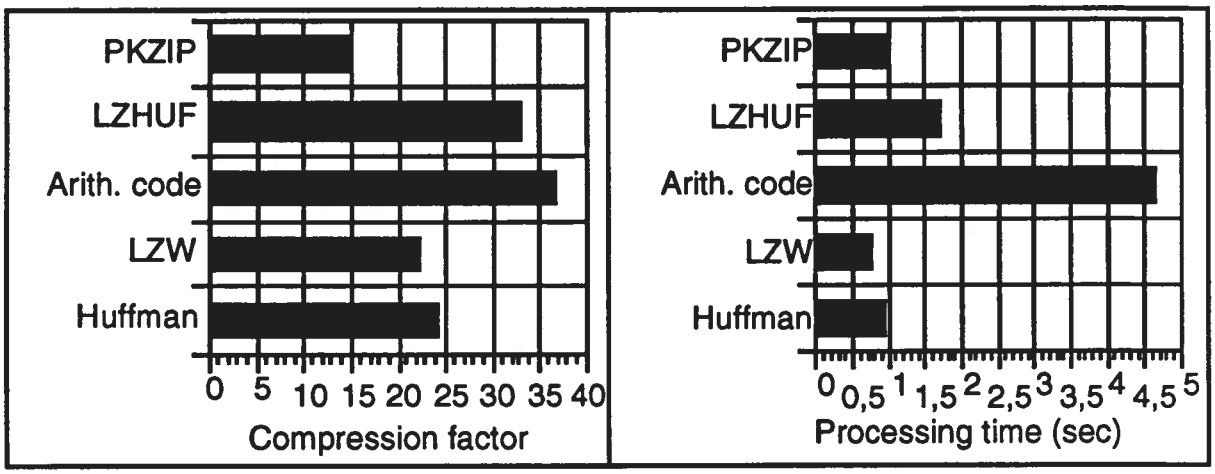


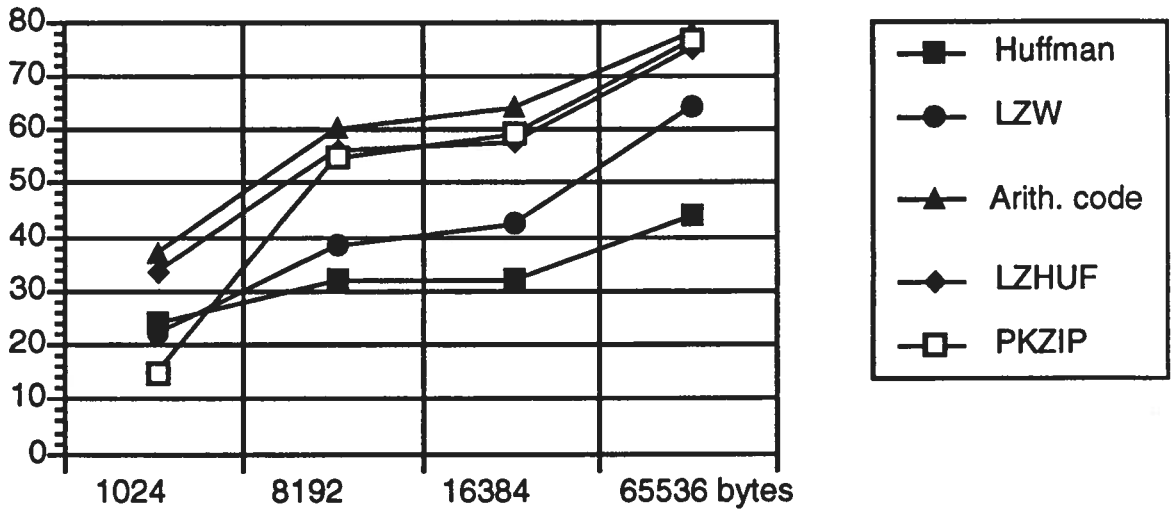


(c) Compression factor versus size of text

With a file of pre-coded medical records, much compression has been done manually and the LZHUF algorithm is still able to compress (6%) the information to be stored in a physical zone.

File of medical records in free text form (L= 1024 bytes)





(c) Compression factor versus size of text

With a file of real medical records in free text form, the LZHUF does a good compression (35%) within a very short period (1,5 sec).

Conclusions

1- Compression techniques appear as a valuable tool to further delay the saturation state of a card;

2- Algorithm LZHUF is the best buy: good compression factor and small processing time;

3- With the imposed constraints to a PMR, memory management policy should be considered (if implementation is feasible) with a given smart card;

4- The two-layered shell is a good structure to immune applications with regard to major upgrades at node level and to allow, eventualy the use of various cards in the network.

* * *

Acknowledgements

This work has evolved from a project funded by the RAMQ. Members of the research group have contibuted to this work by providing various ideas and making comments, namely Eric Bellavance, Guy Girard, Christian Boudreault, Jocelyn Bérubé, Jean-Paul Fortin, Suzanne Hamelin, André Hémond, Guy Lavoie, Marc St-Pierre, Luc Tremblay and Alain Vanasse.

* * *

References

[1] Cantin, Réjean **Conférence au Séminaire international sur les applications de la carte à mémoire,** Stéria Canada, Montréal, mai 1990.

[2] A. Gamache, P. Ardouin et al **Caractéristiques systémiques et techniques d'une carte-santé réalisée avec la carte intelligente,** Actes du Colloque de l'Université Laval sur les cartes Intelligentes, Septembre 1990, 27p.

[3] Cantin, Rejean, **Projet Carte-Santé du Québec,** Conférence à l'Expo-Congrès International des Technologies d'Information, Montrèal, septembre 91

[4] Durant, Pierre **Un serveur généralisé de carte à microprocesseur,** Mémoire de maîtrise, Université Laval, 1991.

[5] P. Ardouin, A. Gamache and al **Effects of memory saturation in the design of a portable medical record,** Proceedings of the Third Global Conference on Patient Cards, March 1991, Spain, 5p.

[6] A. Gamache, P. Ardouin **A Formal Model for Analysis of Memory Allocation and Saturation in the Design of a Smart Card Based System,** Proceedings of SCAT-91, Washington, May 1991, p.134-145.

[7] Zaïane, Rachid, **Mise en oeuvre d'une architecture de driver pour les cartes à microprocesseur et techniques de compression,** Mémoire de maîtrise, Département d'informatique, Université Laval, 1991.

[8] Bell T., Cleary, J. and Witten I.H., **Text Compression,** Prentice Hall Advanced Reference Series, ISBN 0-13-911991-4, 1990